Conventional, Adaptive and Fuzzy Control of Robot Manipulators

THESIS

Submitted in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

By

Sudeept Mohan

Under the Supervision of

Prof. Surekha Bhanot

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN) INDIA

2007

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN) INDIA

C E R T I F I C A T E

This is to certify that the thesis entitled "**Conventional, Adaptive and Fuzzy Control of Robot Manipulators"** submitted by **Sudeept Mohan,** ID. No. **1994PHXF011** for award of **Ph.D. Degree** of the Institute, embodies original work done by him under my supervision.

(Signature in full of the supervisor)

DR. SUREKHA BHANOT

Professor

Birla Institute of Technology and science

Pilani – 333 031 (Rajasthan) INDIA

Date:

Place: Pilani

ABSTRACT

The problem of robot manipulator control is a complex and challenging task. The complexity and challenge arise mainly from the fact that the accurate manipulator dynamic model is difficult to formulate and the manipulator itself might be working in an environment, where it is required to pick different and unknown loads at different times. Under these circumstances, accurate and high-speed motion control of manipulator is a difficult task. Many different control strategies have been proposed in the past to achieve this goal, and it is presently an active area of research. These control strategies range from conventional to adaptive to soft computing techniques like artificial neural networks, fuzzy, genetic algorithms and their combinations.

In this thesis a simulation study of these different control strategies has been undertaken. First the conventional control strategies like the non-model based PD and PID, and model based strategies like, Computed Torque (CT), Feed Forward Inverse Dynamics (FFID) and Critically Damped Inverse Dynamics (CDID) control were studied. These controllers were tested against different trajectories and also for the case where manipulator parameters change during motion due to picking up of payload. It was seen that model based controllers give good performance only if the model parameters are known accurately. It was also seen that the performance of controller improves if we use reference or desired trajectory values for model calculation rather than the actual trajectory values which are obtained from the sensors.

A modification is proposed to the model based control strategies in terms of introduction of modified integral error compensation. The integral action sums the errors for every five iterations of control loop for a given set point. When the set point changes, the error summation is reset to zero. It was seen that the performance of model based controllers improved with inclusion of modified integral action.

Next some adaptive control algorithms for manipulator control were studied. The advantage of adaptive approach is that the accuracy of a manipulator carrying unknown load improves with time because the adaptation mechanism keeps extracting the parameter information from tracking errors. The adaptive controllers studied were Adaptive Critically Damped Inverse Dynamics Controller (ACDID), Model Reference Adaptive Controller (MRAC) and Decentralized Adaptive Controller (DAC). These controllers were also tested for different trajectories and different situations like cold start (no initial estimate of parameters available), warm start (some rough initial estimate of parameters available) and manipulator picking unknown load during the course of motion. It was seen that adaptive controllers give best performance in face of parameter variations. Moreover, the performance is better if some initial estimate of manipulator parameters is available as in case of warm start. Like the conventional controllers these adaptive controllers were also tested for effect of including modified integral error compensation in the control law. It was observed that the performance of adaptive controllers also improves with addition of the modified integral action.

Finally, many different Fuzzy control algorithms for manipulator control were studied. Different hybrid fuzzy control algorithms were tested, which are essentially combinations of conventional or adaptive control algorithms with a lookup table based fuzzy controller. It was found that hybrid fuzzy plus conventional controllers provide performance comparable to adaptive controllers at lesser computational cost.

The Self Organizing Fuzzy Controller (SOC), which builds up the look up table, based on trajectory errors through a modifier algorithm was investigated. It was found that this controller gave best performance amongst all the fuzzy controllers studied in this thesis. The Self Tuning Fuzzy controller (STFC), which changes the output denormalization factor depending on the current trajectory errors, was also investigated. Its performance was not found to be as good as that of Self Organizing Fuzzy controller; however, the overall manipulator motion is smoother for this controller. This is because the controller is not based on lookup table. A modification to the STFC is suggested in terms of changing both the input and output gains by zooming the universe of discourse. This modified controller is known as Coarse/Fine Adaptive Fuzzy controller (CFAF). It was found that CFAF gives better performance than STFC although it is still not as good as SOC. Lastly a new hybrid Fuzzy plus Integral Error controller (HFIE) was investigated. The modified integral action used for this controller was the same as that used earlier for conventional and adaptive controllers. It was found that this simple controller gives a very good performance, next only to SOC and better than STFC or CFAF.

Amongst all the controllers investigated it was found that the hybrid Adaptive Critically Damped Inverse Dynamics (ACDID) + Fuzzy controller gives the best performance.

It is our view that some of the control strategies and techniques that have been extensively investigated for manipulator control in this thesis can also be used in fields like Process control systems which exhibit nonlinear, nonstationary behavior and are difficult to model and control. Experimental implementation of various control schemes studied in this thesis is also recommended for future work.

ACKNOWLEDGEMENTS

First of all I would like to thank my supervisor Prof. Surekha Bhanot for her valuable guidance, encouragement and moral support. It has been a great pleasure to be associated with her on this work.

Next I want to thank Prof. I.J. Nagrath who introduced me to the field of robotics. This work was started under his guidance and he has been a role model and a great source of encouragement to me.

Special thanks are due to Prof. L. K. Maheshwari, Vice Chancellor, Prof. K. E. Raman, Deputy Director, Prof. V. S. Rao Deputy Director, Prof. R. Prakash, Dean, Research and Consultancy Division, for giving me an opportunity and encouragement for completing this thesis. Thanks are also due to Prof. R. K. Mittal for providing valuable comments while going through initial drafts of the thesis. I sincerely acknowledge the help received from Prof. J. P. Misra, Unit Chief, Information Processing center Unit. He helped me with suggestions on trying out various new control strategies.

Thanks are also due to Prof. Sanjoy Roy and Prof. H.V. Manjunath for their critical comments, which helped me improve the quality of manuscript.

I would also like to thank Prof. S. Balasubramanium and Prof. S. Gurunarayanan, Dean, Admissions, for their constant support.

I received encouragements and support from my colleagues namely Dr. Navneet Goyal, and Mr.Yashvardhan Sharma. I express sincere thanks to them.

Special thanks are due to Dr. Dinesh Saini, who helped me in various stages of the thesis. He constantly exhorted me to finish this mammoth task and made me believe that I can do it.

Lastly I also thank my ex-colleagues and friends Mr. Puneet Teja, Mr. Sartaj Singh and Mr. R. R. Ulanganathan for making the time spent at Center for Robotics and Intelligent Systems (CRIS) so memorable.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Abbreviations

ACDID	Adaptive Critically Damped Inverse Dynamics
CDID	Critically Damped Inverse Dynamics
CFAF	Coarse/Fine Adaptive Control
CT	Computed Torque
DAC	Decentralized Adaptive Control
FFID	Feed Forward Inverse Dynamics
FL	Fuzzy Logic
FLC	Fuzzy Logic Controller
HFIE	Hybrid Fuzzy with Integral Error
LTI	Linear Time Invariant
LUT	Look Up Table
MRAC	Model Reference Adaptive Control
NZLUT	Non Zero Look Up Table
PD	Proportional Derivative
PI	Proportional Integral
PID	Proportional Integral Derivative
SOC	Self Organizing Control
ZLUT	Zero Look Up Table

INTRODUCTION

The design of intelligent, autonomous machines to perform tasks that are dull, repetitive, hazardous, or that require skill, strength, or dexterity beyond the capability of humans is the ultimate goal of robotics research. Examples of such tasks include manufacturing, excavation, construction, undersea, space and planetary exploration, toxic waste cleanup, and robotic assisted surgery.

Robotics research is highly interdisciplinary, requiring the integration of control theory with mechanics, electronics, artificial intelligence, and sensor technology. Table 1.1 shows a brief history of robotics and also highlights its interdisciplinary nature.

- 1920 – Czechoslovakian playwright Karel Capek introduces the word robot in the play *R.U.R. - Rossum's Universal Robots.* The word comes from the Czech *robota*, which means tedious labor.
- 1938 – The first programmable, paint spraying mechanism is designed by Americans, Willard Pollard and Harold Roselund for the DeVilbiss Company.
- 1942 – Isaac Asimov publishes *Runaround*, in which he defines the Three Laws of Robotics.
- 1946 – Emergence of the computer.
- 1950 – I, *Robot*, a landmark collection of Asimov's stories, is published.
- 1951 – In France, Raymond Goertz designs the first teleoperated articulated arm for the Atomic Energy Commission. The design was based entirely on mechanical coupling between the master and slave arms (using steel cables and pulleys).
- 1954 – George Devol designs the first programmable robot and coins the term Universal Automation, planting the seed for the name of his future company - Unimation.
- 1959 – Marvin Minsky and John McCarthy establish the Artificial Intelligence Laboratory at MIT.
- 1962 – General Motors purchases the first industrial robot from Unimation and installs it on a production line. Hardyman is born!
- 1964 – Artificial intelligence research laboratories are opened at Stanford Research Institute (SRI), Stanford University, and the University of Edinburgh.
- 1965 – Carnegie Mellon University establishes the Robotics Institute.
- 1970's – Robots begin to be used in industrial applications.
- 1980's – Several robotics companies are founded: CRS, Adept, Computer Motion etc.

- 1990’s – Walking robots, mobile robots, and new innovations emerge: Haptics, Humanoids, Rovers etc.
- 2000’s – Robots are mainstream…ex: Space station arm, Sony Aibo, Palm robot, Telesurgery.

Table 1.1 Brief history of Robotics

The term robot has been applied to a wide variety of mechanical devices, from children's toys to guided missiles. An important class of robots is the manipulator arms, such as the CRS A255 robot shown in Figure 1.1. These manipulators are used primarily in materials handling, welding, assembly, spray painting, grinding, deburring, and other manufacturing applications. The research work in this thesis discusses the aspects related to control of such manipulators.

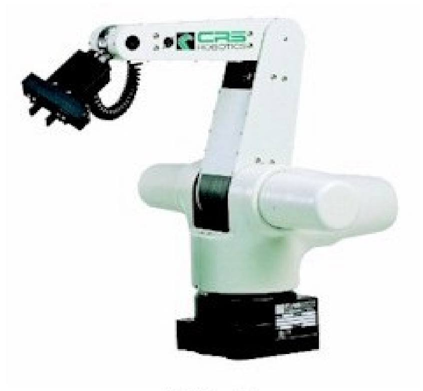


Fig 1.1 The CRS A255 articulated manipulator

This thesis exclusively considers a commonly accepted class of robot plants - rigid body open kinematic chains. A rigid body open kinematic chain consists of a serial arrangement of finitely many (n) rigid bodies fixed by either prismatic or rotational joints, and whose proximal link is joined to an inertial reference system. It is assumed that each joint is instrumented with an actuator capable of delivering a commanded torque, and sensors for sensing both position and velocity. This class of holonomic[*]

[*] In robotics, holonomicity refers to the relationship between the controllable and total degrees of freedom of a given robot. If the controllable degrees of freedom are equal to the total degrees of freedom then the robot is said to be holonomic. If the controllable degrees of freedom are less than the total degrees of freedom it is non-holonomic. A robot is considered to be redundant if it has more controllable degrees of freedom than degrees of freedom in its task space.

systems has the useful property that the joint positions and velocities provide a natural set of generalized coordinates, for which by applying straightforward techniques of classical mechanics a finite dimensional, second order, ordinary differential equation of motion is obtained. The equations of motion, though nonlinear, have the desirable structure that they are completely controllable, observable, and are integrable. The control problem considered is ***reference trajectory tracking*** [Tarokh and Seraji (1988)]***;*** the task is to analyze/design controllers which causes the plant output (robot position and velocity) to asymptotically match a specified reference signal. This thesis does *not* address a great number of other problems in robot control e.g. impedance control, grasping, assembly, collision avoidance, task encoding, sensing environment, vision, user-interaction, running, hitting, catching, flexible joints, flexible links, etc.

The initial attempts to solve the problem of manipulator control fall under the '*conventional*' category. The controllers here consisted of simple PD controllers and some model-based feedback linearizing controllers [Kelly (1998), Heredia and Wen (2000)]. The PD controllers are widely used in industrial robots and treat each joint of manipulator as decoupled and driving a constant inertia load. These controllers work well if the manipulator joints are highly geared in which case the cross coupling effects of dynamics diminish and can be neglected. These controllers give poor performance in case of direct drive, high-speed robots.

Feedback linearization is a useful paradigm because it allows the extensive body of knowledge from linear systems to be used to design controllers for nonlinear systems. The roots of feedback linearization in robotics predate the general theoretical development by nearly a decade, going back to the early notion of feed forward computed torque [An et al. (1989)]. The basic idea of feedback linearization control is to transform a given nonlinear system into a linear system by use of a nonlinear coordinate transformation and nonlinear feedback.

In the robotics context, feedback linearization is known as *inverse dynamics*. The idea is to exactly compensate for all of the coupling nonlinearities in the Lagrangian dynamics in a first stage so that a second stage compensator may be designed based on a linear and decoupled plant. Any number of techniques may be used in the second stage. The feedback linearization may be accomplished with respect to the Joint Space coordinates or with respect to the Task Space coordinates.

The feedback linearization approach exploits important structural properties of robot dynamics. However, the practical implementation of such controllers requires consideration of various sources of uncertainties such as modeling errors, computation errors, external disturbances, unknown loads, and noise. Moreover as the manipulator functions in its workspace, its parameters change with every new payload it picks up. This makes the applicability of conventional controllers very limited. However these controllers give us a good insight into the problem of manipulator control and also form the framework on which more advanced adaptive controllers are based.

In this thesis we have done an in-depth study of these conventional controllers and have also suggested a method of adding integral action to these controllers to improve their performance.

Robust and adaptive control are concerned with the problem of maintaining precise tracking under uncertainty [Slotine (1985), Yao (1997), Imura et al. (1994)]. We distinguish robust from adaptive control in the sense that an adaptive algorithm typically incorporates some sort of on-line parameter estimation scheme while a robust, non-adaptive scheme does not.

An advantage of adaptive approach is that the accuracy of a manipulator carrying unknown loads improves with time because the adaptation mechanism keeps extracting parameter information from tracking errors. Thus the adaptive controllers hold promise of consistent performance in face of large load variations and inaccuracies in initial parameter estimations. While many adaptive controllers have been proposed in literature [Johansson (1990), Hsia (1986), Yuh et al. (1998)], most of them rely on assumptions such as local linearization, time invariance, decoupled dynamics etc., to guarantee their tracking convergence. However in recent past, attempts have been made and control strategies have been proposed which do not resort to these assumptions for proving global stability. These schemes mostly make use of linear parameterization property of manipulator dynamics to synthesize the adaptation law and to prove global stability [Sadegh and Horowitz (1990)]. In this thesis we have studied through simulations a few adaptive controllers. These controllers were tested for both warm and cold start situations. In case of warm start, the controller starts with some rough initial estimate of values of the parameters whereas in case of cold start it is assumed that no such initial estimate is available. A comparison of performance is also done. We have also investigated the effect of introducing integral error compensation on the performance of these controllers.

While conventional controllers suffer in performance because they do not take care of many uncertainties that a manipulator faces, the adaptive controllers have their own drawbacks. They are:

- Requirement of a reasonably accurate manipulator model, even though the parameter values are not required to be known exactly.
- Requirement of fast processors to implement computationally intensive algorithms.
- The adaptive laws are derived mainly by trial and error.
- It is difficult to prove the stability and robustness of these controllers.

In order to overcome the problems of inaccurate dynamics model and computational time constraints, a lot of work has been done in the area of fuzzy logic control of robot manipulators [Emami et al. (2000), Ham et al. (2000), Koo (1995)]. Fuzzy Logic control provides an extensive freedom for control designers to exploit their understanding of the problem and to construct intelligent control strategies. Nonlinear controllers can be devised easily by using fuzzy logic principles. It makes fuzzy controller a powerful tool to deal with nonlinear systems.

Many forms of fuzzy controllers for robotic manipulators have been proposed in literature. These include conventional controllers [Ya-Chen et al. (1997), Nedungadi and Wenzel (1991)], controllers with gravity compensation, self-organizing [Koh et al. (1990), Kazemian (1998)] and self-tuning fuzzy controllers [Llama et al. (2000)], hybrid fuzzy controllers [Meng and Swee (2000), Karner and Janocha (1997), Ya and Meng (2004)] etc. Some of these controllers have been investigated in this thesis. Of particular promise is the self-organizing controller, which builds up the rule base on-line as the manipulator operates. It has a very simple structure, is computationally not intensive and gives very good performance. Also of interest is a new hybrid fuzzy controller proposed in the thesis, which too gives commendable performance. This controller is combination of conventional fuzzy and integral error compensator. The integration of errors is done in a novel way so as to avoid any chances of instability.

As can be seen from above discussion, the robot manipulator control problem is a wide and open area of research. In this thesis we have studied some of the many control strategies used for manipulator control and have also proposed some new control methods as well.

1.1 OBJECTIVES

This thesis is mainly about the computer control of motion, and represents an infinitesimal advance in the human capacity to both understand and to synthesize devices capable of performing useful work. Specifically, it addresses the problem of constructing and analyzing control systems for robot arms, which reliably and accurately follow, prespecified trajectory.

The main objectives of thesis are as follows:

(a) Conventional Control

- (i) To study various conventional control algorithms used for manipulator control. These algorithms include non-model based simple PD and PID as well as model based Computed Torque, Feed-forward inverse dynamics etc.
- (ii) To analyze the performance of various conventional controllers through simulations on predefined trajectories.
- (iii) To investigate the effect of parameter variations on controller performance.
- (iv) To study the effect of inclusion of integral error compensation on controller performance

(b) Adaptive Control

- (i) To study different adaptive control algorithms proposed for robot manipulator control. These include Adaptive Critically damped inverse dynamics controller, Model reference adaptive scheme and Decentralized adaptive controller.

(ii) To study the performance of these controllers for situations like, parameter change, warm start, cold start etc.

(iii) To study the effect of inclusion of integral error compensation on controller performance

(c) Fuzzy Control

(i) To study some different Fuzzy control schemes which are used for manipulator control. These include Conventional Fuzzy, Adaptive Fuzzy and Hybrid Fuzzy schemes.

(ii) To investigate the performance of these controllers under situations like manipulator picking up a load, starting with zero/non-zero entries in rule base etc.

(iii) Propose some new hybrid controllers which combine fuzzy with conventional, and fuzzy with adaptive controllers.

(d) Comparative analysis of performance of the aforesaid controllers.

1.2 ORGANIZATION OF THE THESIS

This thesis is organized in seven chapters. **Chapter 2** reviews literature in the area of robot manipulator control. The review includes work done by researchers across the globe on conventional, adaptive and fuzzy control. Review is also done of schemes, which have not been directly tested on manipulators.

Chapter 3 deals with the general structure of manipulator dynamics. It discusses some important properties of the component matrices of manipulator dynamics. These properties are extensively exploited for design of controllers and for proving their stability. The dynamics equations of a two-link manipulator are derived, and error norms used for comparison of performance are discussed.

Chapter 4 deals with Conventional control strategies for manipulators. These controllers are tested on trajectories for cases like, parameters exactly known, parameters not exactly known, parameters changing during the course of trajectory etc. Effect of adding integral error compensation to these controllers is also investigated.

Chapter 5 discusses Adaptive control strategies for manipulators. These controllers are tested for performance through simulations. The situations tested include: manipulator parameter estimate available (warm start), manipulator parameter estimate not available (cold start), manipulator picking up a load during motion etc. Effect of adding integral error compensation to these controllers is also investigated.

Chapter 6 deals with Fuzzy control methods for robot control. Various controllers like Conventional, Hybrid and Adaptive Fuzzy are discussed. Both, self-organizing and self-tuning varieties of adaptive fuzzy controllers are discussed. Also both lookup table and non-lookup table based controllers are investigated. Some new Hybrid fuzzy controllers are also proposed. All these controllers are tested through simulations.

Finally **Chapter 7** presents the main conclusions of the thesis and provides recommendations for future work.

CHAPTER II

LITERATURE REVIEW

2.0 INTRODUCTION

As explained in Chapter 1, control of robot manipulator is a complex and challenging task. The motion of each joint of manipulator is usually produced by actuators that produce torque or force. If the actuator used (e.g. Stepper motors) could directly execute the trajectory commands then open loop control would suffice. However such actuators are usually not used in manipulators because of their high weight to torque ratios and slow speeds.

The complexity of manipulator control problem is compounded by many factors, some of which are listed below:

- The highly nonlinear dynamics of both manipulator and actuator, arising due to inertia, gravitational, coriolis and centrifugal effects, friction, mechanical flexibility, backlash, hysteresis and actuator geometry.
- Accurate control is required over a wide range of operating conditions.
- There is cross coupling between neighboring inputs and outputs of the system.
- The system dynamics parameters are time varying, for example due to changes in payload, configuration, speed of motion and component wear.

There are many control schemes proposed in literature for robot manipulators. The use of a particular scheme is very much situation dependent. Simple PD controller can accurately control a highly geared industrial robot, performing pick and place operation on a known load. On the other hand a manipulator working in unknown environment or a high speed direct drive manipulator would require more complex Adaptive or Fuzzy controller.

The various controllers proposed by researchers for manipulator motion control can be broadly classified into two categories based on the co-ordinate system they deal with. These two categories are *Joint space* based schemes and *Cartesian space* based schemes. The latter are also known as Resolved motion controllers in literature.

Another possible classification of robot controllers is in terms of their structure. In this scheme the controllers are classified into one of the following categories:

- Conventional
- Robust
- Adaptive
- Fuzzy, Neural, GA based etc.

A good taxonomy of robot manipulator controllers is provided by Miljanovic and Croft (1999). In this thesis we have explored controllers belonging to Conventional, Adaptive and Fuzzy categories. This chapter presents a brief literature review of controllers in these categories.

2.1 CONVENTIONAL CONTROL

The conventional controllers for robot manipulators consist of simple PD/PID controllers, minimum time controllers, variable structure controllers and non-linear decoupled feedback controllers etc.

The PD/PID controllers are very popular because of their simple structure and are still widely used in many industrial robots.

Tarokh and Seraji (1988) have proposed a simple scheme for control of manipulators. The scheme has two loops: an inner PD loop and an outer PID loop. The PD controller stabilizes the robot by classical pole assignment technique, while the outer PID loop achieves input-output decoupling for easy reference trajectory tracking. The PD and PID gains are easily tunable and are related directly to the linearized manipulator model.

One of the main weaknesses of PD controllers is that they require measurement of velocity for calculating the control law. Velocity measurement is often a problem and is noise prone. One solution to overcome this problem is to implement a velocity observer. Heredia and Wen (2000) have proposed a high gain observer for estimating the velocity. They use the singular perturbation method to analyze the PD controller with high gain observer. They have proved that observer error and tracking error become stable and have also given the conditions to show the asymptotic stability of the PD controller.

The proportional–derivative (PD) control plus gravity compensation together with the PD control plus *desired* gravity compensation are the simplest global regulators for robot manipulators. The best feature of these controllers is that the tuning procedure to achieve global asymptotic stability reduces to selecting the proportional and derivative gains in a straightforward manner. However, a drawback of both control strategies is that the knowledge of the gravitational torque vector of the robot dynamics, which depends on some parameters such as mass of the payload, usually uncertain, is required. Kelly (1998) introduced a new class of global position controllers for robots, which do not include their dynamics in the control laws. He developed a new class of regulators leading to a linear PD feedback plus an integral action driven by a class of nonlinear functions of the position error. He characterizes the class of function and gives simple explicit conditions on the controller parameters which guarantee global positioning.

Another important class of conventional controllers is those, which use the model of manipulator to accomplish feedback linearization. This idea was explained in some detail in previous chapter. These controllers are known as *model-based* controllers. These controllers use schemes, which range from simple gravitational compensation to feedback linearization of the full manipulator dynamics. Clearly the suitability of a model-based controller is dependent upon how well the system under control is known.

An ideal model based controller consists of the inverse of the system dynamics, used as a pre-compensator to the actual system. The control inputs required to meet the desired positions, velocities and accelerations can then be calculated directly from the inverse system model. Thus, the system is driven open loop with perfect cancellation between the inverse dynamics and the real system. This simple scheme suffers from the drawback that manipulator dynamics is usually never known perfectly. All the unmodelled effects are thus not compensated. To overcome this problem feedback is used where the open loop model-based controller is combined with a classical feedback controller (ex. PID). The two controllers are then known as *Primary* and *Secondary* respectively. The purpose of secondary controller is to maintain trajectory tracking in presence of modeling errors and unmodelled disturbances. The primary controller is designed using any available knowledge of system dynamics. The overall controller is also known as *feedforward* model based controller. Paul (1974) and Bejczy (1976) proposed such controllers which use full manipulator dynamics model in primary part. These controllers are also known as Computed Torque controllers because the primary part of controller ‘computes’ the torque to be applied at joints depending on desired position, velocity and acceleration. These controllers are thus computationally very intensive. To reduce the computational burden, schemes have been suggested which use only part of manipulator dynamics, such as, gravity terms. The gravity part of dynamics is simple and provides the holding torque information thereby reducing the integral action required of the secondary controller.

However to cancel the gravity terms we should have their exact knowledge. This at times is a difficult task. Khorrami and Ozguner (1988) showed that asymptotic exact tracking of trajectory could be achieved with state feedback and PI controllers. No exact knowledge of gravity terms is required. Only the nominal parameter values and bounds on their variations are required. However global asymptotic stability is guaranteed only for the case of constant set point trajectory.

Another way of reducing the computational burden of these model-based schemes is to use a linearized model of the system under control. This can take the form of a state space controller designed to position the poles of the closed loop system, or to optimize some performance criterion. However, the linearized model quickly becomes inappropriate as the manipulator moves throughout its workspace, and hence degrades the control. This approach may be effective if deviations from the linearization point are small, or alternatively if different linearized models are used as the robot moves along its trajectory. An et al. (1989) give a good comparative experimental study on some feedforward and computed torque controllers.

Another approach to manipulator control is to use the model in feedback part of controller. It is thus known *feedback* model based controller. Here, the inner primary controller is designed using the inverse system dynamics to give an ideally decoupled and linearized system. It is then a simple task to design a secondary controller that regulates the nominally linear system, giving the required closed loop system response. The secondary controller also compensates for errors in the model based primary controller, to ensure set point tracking and disturbance rejection. This approach is also occasionally referred to as computed torque control, but differs from previous law since it uses feedback rather than feedforward.

Song et al. (1989) proposed a scheme, which tackles the problem of manipulator picking up unknown loads at different times. They incorporate the load dynamics in the manipulator dynamics itself and then exploit its properties to design the controller. Their scheme compensates for effects of nonlinearities, couplings and varying payloads and also ensures tracking convergence.

Leung et al. (1990) proposed a sliding mode robot controller based on the variable structure control theory. Spong et al. (1986) proposed a robot controller which uses optimal decision strategy**,** to derive a pointwise optimal control law which minimizes the deviation between the vector of actual joint accelerations and a desired joint acceleration vector, subject to the input constraints.

For a class of robot manipulators, which contain nonlinear couplings and uncertainties, Mao-Lin and Meng (2000) proposed decentralized stabilizing controllers and tracking controllers. In the former case, the system state is ensured to lie ultimately in a prescribed neighborhood of the origin and this neighborhood can be made arbitrarily small. In the latter case, the system is guaranteed to ultimately track a desired model with a prescribed error and this tracking error can also be made arbitrarily small. Moreover, this approach can admit larger nonlinear couplings and uncertainties in the robot manipulator system. Zhang et al. (1990) present a digital implementation of an optimal PID controller of linearly interpolated joint trajectories. The controller obtains optimal performance by reformulating the PID control law to minimize the time delay between the position transducer reading and the application of the corrective torque.

Some authors have also proposed globally stable controllers where the motor dynamics is included in the manipulator model. Su and Stepanenko (1994) proposed one such controller and proved its stability using the Lyapunov method. They only assumed that the inertial parameters of manipulator and electrical parameters of actuators are bounded.

An exhaustive discussion on advanced robot control techniques has been given by Ge (1998) and Er (1993).

2.2 ADAPTIVE CONTROL

The controllers discussed previously have constant parameters, and are designed to be stable even when there are variations in the system under control. An alternative approach, termed *adaptive control* [Ortega and Spong (1988), Tosunoglu and Tesar (1988), Sinha et al. (1990), Yu et al. (1992), Ham (1993), Zhang et al. (2000), Zergeroglu et al. (2000)], automatically adjusts the controller gains as the system changes, as shown in Figure 2.2.1. The controller therefore acts to maintain the closed loop system response in the presence of variations in the system.

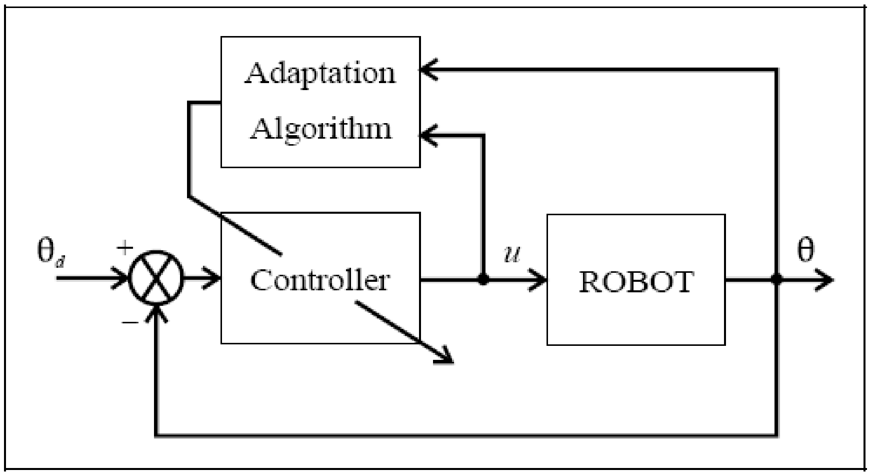


Fig. 2.2.1 Block diagram of Adaptive controller

The first adaptive robot control algorithms in the literature addressed simplified approximations to the full rigid body model.

One of the earliest papers specifically addressing adaptive robot control is by Dubowski and DesForges (1979). This paper address a robot model where the joints are modeled as decoupled linear time invariant (LTI) plants. Horowitz and Tomizuka (1986) proposed to adaptively compensate for time varying nonlinear elements of the full nonlinear plant with the assumption that these elements are slowly time varying in comparison to the rate of parameter adaptation. Takegaki and Arimoto (1981) proposed a different approach using an approximation to the full plant model, which omits some of its nonlinear terms. Many researchers working with adaptive control algorithms addressed a great variety of approximations to the full rigid-body nonlinear model. Hsia (1986) and Landau (1988) give a complete account of these results, which, because of the plant approximations employed, can at best provide only local stability with respect to the full nonlinear system.

Several authors made explicit use of the linearity of robot inertial parameters in developing experimental techniques for adaptive gravity cancellation [Koditschek (1985)] as well as the off-line and on-line identification of these parameters [Khosla and Kanade (1985)]. Shortly thereafter, several authors reported stable direct adaptive tracking controllers that were correct with respect to the full nonlinear robot model. These adaptive controllers, which still require accurate plant structure parameter values, compensated for either partial or complete lack of knowledge of plant inertial parameter values.

Craig et al. (1986) reported the fist adaptive robot control algorithm, which is globally convergent in tracking error. This algorithm is an adaptive version of the familiar "computed torque" exact linearization inverse dynamics control law [Luh et al. (1980)]. It has the advantage that it provides for linear tracking error dynamics. It has the disadvantage that (i) it requires measurement of joint acceleration in addition to position and velocity, and (ii) it is only locally stable in controller parameter error, due to a required inversion of its estimated inertia matrix in the parameter update law. Ortega and Spong (1988) have reported versions of these control algorithms, which do not require measurement of joint acceleration, but remain only locally stable in the controller parameter error.

Many model based adaptive schemes have also been proposed [Ham (1993), Slotine (1986), Tso et al. (1991)], where those coefficients of the robot model that are not well known or are changing, are updated automatically. This is achieved using a *system identification* algorithm, which uses past input and output values of the system to estimate the parameters, for example payload mass. The nonlinear equations of motion of the robot are expressed as a linear function of joint outputs and model parameters. These parameters are estimated using a Lyapunov function candidate approach [Wen (1990)], and they converge to their true values provided certain constraints are met. This method requires measurement of the joint angles, velocities and accelerations that can be problematic due to noise.

A different approach to stable adaptive model based control which achieves stable tracking ***without exact linearization*** was reported independently, by Slotine and Li (1988), and by Horowitz and Sadegh (1987), using a sliding-mode type of stability proof. This algorithm has the perceived disadvantage that it ***does not*** provide linear tracking error dynamics, but it has the advantage that it is globally stable in plant parameter tracking error, asymptotically exact tracking, and does not require measurement of joint acceleration. This particular area of research has seen much work and is still active, addressing issues of convergence, stability and computational burden. However, these model based adaptive controllers are generally only practical if the number of estimated parameters is restricted. The problem becomes complex if the full manipulator model is to be estimated.

A simpler version of this "non-linearizing" approach, possessing a ***local*** lyapunov stability proof, was reported independently by Koditschek (1987). A ***globally*** stable version of this simpler approach, with proof of global stability, was provided by Whitcomb et al. (1993).

A second variant of this "non-linearizing" approach with a feedforward structure admitting off-line tabulation was reported by Wen et al. (1987). DeWit and Fixot (1992) proposed an adaptive controller based on estimated velocity feedback thereby removing the need of actual velocity measurement, which may be contaminated by noise.

In deriving the adaptive and control laws, many times the motor dynamics are ignored, which essentially is an approximation. Yu and Lloyd (1995) and Jing (1995) proposed an adaptive controller that takes the manipulator motor dynamics into consideration. The adaptation law proposed overcomes uncertainties of both manipulator and motor parameters.

As mentioned earlier, the need of developing advanced control methods for robot manipulators has been stressed by many researchers, and a lot of papers have appeared during the last decade. One of these approaches is the decentralized adaptive control**.** This scheme, also known as "independent joint control," is motivated by the notion of the decentralized location of the actuator and sensor on each link. In practice, however, the control gets somewhat complicated since the robot is a highly coupled, nonlinear multivariable system. Also their operating environment is often poorly known and their parameters cannot be calculated accurately enough to be used in real-time control applications. As a result, its stability analysis becomes more complicated and its accurate model parameters have to be known a priori**.** For these reasons, adaptive schemes are very susceptible to these uncertainties and complexities. Several development and applications have been presented**,** where adaptive controllers are used to enhance stability and improve operating conditions of robot manipulator systems [Hsu and Fu (2002), Oh et al. (1988)].

Oh et al. (1988) proposed a decentralized adaptive controller where the controller gain is derived by using model reference adaptive control theory based on Lyapunov's direct method. The adaptive gains consist of proportional, and integral combination of the measured and reference values of the corresponding subsystems. Colbaugh et al. (1993) proposed a controller, which is extremely simple computationally and does not require knowledge of either the mathematical model or the parameter values of the robot dynamics. The controller was shown to be globally stable in the presence of bounded disturbances. Furthermore the control strategy is very general and is implementable for either position regulation or trajectory tracking in either joint space or task space. Gavel and Hsia (1987) presented a decentralized adaptive controller based on high gain feedback approach. Convergence is local in the state parameter space. Because of a high gain feedback approach, the algorithm is tolerant to the nonlinear, time varying interaction among joints, and also to the interaction among control channels due to the nondiagonal inverse Jacobian matrix. The positive definiteness of inverse Jacobian matrix is exploited to make this approach successful.

Tarokh (1990) proposed a decentralized adaptive controller based on discrete time model of the robot manipulator. In his scheme each local controller utilizes only its own joint angle measurement and reference position, and does not require knowledge of the payload, robot characteristics or other joint angles. Due to the decentralized structure of the controller and the simplicity of the control algorithm, computation of joint torques can be performed in parallel in a real time environment. The adaptation laws are derived using hyperstability theory, which guarantees asymptotic trajectory tracking despite gross robot parameter variations. The controller gains are independent of the robot parameters provided that the gain adaptation is sufficiently fast. In the independent joint controller scheme proposed by Seraji (1988) for the development of the decentralized control scheme, each joint is viewed as a subsystem of the entire manipulator system. These subsystems are interconnected by “disturbance torques” representing the inertial coupling terms and the coriolis, centrifugal, friction and gravity terms. The proposed decentralized control scheme consisting of a number of independent joint controllers has several advantages over a single centralized controller for the entire manipulator. A major advantage is that the joint control algorithms require much less computations than the single algorithm resulting from a centralized control law. Furthermore, due to the possibility of parallel processing and distributed computing, the decentralized control scheme can be implemented on a number of simple and fast microprocessors with a high sampling rate, thus improving the system performance. Another major advantage of the decentralized control scheme is its reliability and fault tolerant feature. In case one joint encoder gives erroneous readings of the joint position, in a centralized control system, this would affect the entire control action for all joint motors; whereas in a decentralized system, only one control loop is affected and the remaining joint controllers are unaffected. A variant of this control strategy proposed by Magana and Tagami (1994) is investigated in detail in Chapter 5. Recently Parra-Vega (2003) proposed another simple decentralized continuous sliding PID controller for tracking tasks that yields semi global stability of all closed-loop signals with exponential convergence of tracking errors.

Besides the main approaches to adaptive control discussed above, many other variations of adaptive control have been proposed in literature. Variable structure adaptive controllers have been proposed by Yu (1998), Yu and Lloyd (1997) and Tso et al. (1991). Adaptive learning controller has been proposed by Messner et al. (1991). An adaptive controller designed using input-output approach is presented by Kelly and Ortega (1988). Trusca and Lazea (2003) have proposed an adaptive PID learning control algorithm for periodic robotic motion. Their controller consists of an adaptive PID feedback part and a feedforward input learning part. The feedback part overcomes the disturbances while the feedforward part produces the desired torques. Some authors have also presented adaptive control schemes based on manipulator task space. In one such scheme Feng (1995) presents a composite law, which uses the prediction error and tracking error to derive parameter estimates without requiring inverse of Jacobian or estimated inertia matrix. Burkan (2005) developed adaptive controller using trigonometric functions depending on manipulator kinematics, inertia parameters and tracking error, and both system parameters and adaptation gain matrix are updated in time. The control law includes a PD feed forward part and a full dynamics feed forward compensation part with the unknown manipulator and payload parameters.

As can be seen, many versions of adaptive control strategies are available for use today. To decide on merits and demerits of one adaptive strategy vis-à-vis another, and also that of adaptive with respect to conventional or robust schemes, some results are available in literature. Erlic and Lu (1990) presented an experimental comparison of Adaptive, Robust and Classical Feedback Controllers used in unconstrained trajectory tracking for robot manipulators. In their study it was found that the adaptive controller outperforms the other controllers. Good performance was also achieved using the computed torque method. The proportional-derivative controller was found to perform poorly for velocity tracking. The adaptive algorithm however was found to be computationally demanding. Kim and Hori (1995) have presented an experimental evaluation of adaptive and robust schemes for robot manipulator control. They have classified the adaptive control laws in three groups and have shown that the main difference between groups is in terms of their PD gains. They have further shown that the controllers can give matching performances by proper adjustment of the PD gains. They have also investigated a two degree-of-freedom robust controller and have demonstrated its strong disturbance rejection properties.

Burdet et al. (1998) have also given a comparative evaluation of nonlinear adaptive controllers. They have shown that a version of Feedforward adaptive controller is well suited for learning the parameters of dynamic equation even in presence of friction and noise. However, if the task consists of executing a repeated trajectory, a Lookup table based memory controller is simpler to implement. Niemeyer and Slotine (1988) have commented on performance in adaptive control, specifically about issues related to computational efficiency and recursive implementation of the control algorithms.

To summarize, the adaptive control theory for robot manipulators is at present seeing wide interest and many researchers are trying alternative controller designs. In this thesis we have studied some of the approaches to adaptive control of robot manipulators.

2.3 FUZZY CONTROL

Real time control of a robot manipulator has been the topic of research for a long time. Many theoretical results have been and are being published showing various efficient and accurate control strategies. These include adaptive control, non-linear feedback control, resolved motion rate control, inverse dynamics control etc. But all of them have problems from practical applicability point of view because of:

1) Complex non-linear mathematical model of the manipulator and
2) Extremely involved computational requirements.

In such cases, where conventional and other control methods prove inadequate and complex, it is worthwhile to investigate the control policies of a human operator. Fuzzy Logic is one such control method, which is based on human intuition and experience. Fuzzy algorithms are easy to implement on a computer, do not involve any major computational problems, and do not require a detailed mathematical model of the system [Jamshidi (1997), Kazemian (2001)]. Fuzzy algorithms find wide use in robotic control systems [de Silva (1995), Banerjee and Woo (1993)].

Many Fuzzy control strategies for manipulator control have been proposed in literature. Erbatur et al. (1995) provide a comparative analysis of four different kinds of fuzzy controllers. The controllers studied are: Straight forward conventional fuzzy control, fuzzy control with gravity compensation, fuzzy control with nonlinear state feedback and a self organizing fuzzy control. The problem of gain adjustment in basic fuzzy controller is overcome by using a self-organizing fuzzy controller. The self-organizing controller is able to adjust its gains in a single run. Abdessemed and Benmahammed (1998) have proposed a two layer fuzzy controller. The first layer is the familiar PID controller, and the second is the precompensator, designed on the basis of decision making rules and tuned to minimize the output error when the conventional controller exhibits significant steady state error and a loss in control. The control strategy proposed by Lim and Hiyama (1991) for robotic manipulators incorporates a proportional plus integral (PI) controller with a simple fuzzy logic (FL) controller. In the proposed strategy, the PI controller is used to ensure fast transient response and zero steady-state error for step inputs, or end-point control, whereas the FL controller is used to enhance the damping characteristics of the overall system. A good classification of fuzzy PID controllers is provided by Hu et al. (2001).

Researchers have proposed many versions of adaptive fuzzy controllers. Zhao et al. (1993) showed that the gains of a simple PID controller could be adapted using fuzzy logic. Fuzzy rules and reasoning are utilized on-line to determine the controller parameters based on the error signal and its first difference. Tzafestas and Papanikolopoulos (1990) presented an approach to intelligent PID control, which is based on the application of fuzzy logic. Their approach assumes that nominal controller parameter settings are available through some classical tuning technique (Ziegler-Nichols, Kalman, etc.). By using an appropriate fuzzy matrix (which is similar to Macvicar-Whelan matrix), they determine small changes on these values during the system operation, that lead to improved performance of the transient and steady state behavior of the closed-loop system. Visoli (2001) presents a comparison between different methods, based on fuzzy logic, for the tuning of PID controllers. Yoo and Ham (1999) proposed an adaptive controller that uses a fuzzy logic system to approximate any nonlinear system. There is no need to derive the linear robot dynamic formulation. Their controller is robust not only to the structured uncertainties such as payload parameters, but also to the unstructured ones such as friction model and disturbances. Neo and Er (1995) present a controller that employs tracking errors of the joint motion to estimate the robot dynamics, which are subsequently used in the control law. In particular, it requires no feedback of joint accelerations. This adaptive controller does not require the exact robot dynamics but only the boundary of the dynamics. The controller guarantees the global stability of resulting closed-loop system in the sense that all the signals are bounded. A good reference, which discusses stability issues related to fuzzy controllers, is by Kandel et al. (1999). A controller for adaptive fuzzy tracking control of manipulator is proposed by Lin et al. (2003). Their adaptive fuzzy compensator performs on-line learning to approximate and compensate the unknown nonlinear dynamics of the system so that minimizing a quadratic performance index can obtain the optimal tracker. The proposed controller and the associated learning algorithm require no preliminary off-line learning for initialization and guarantee the output-tracking error to be uniformly bounded. Commuri and Lewis (1996) propose a learning algorithm that learns the stabilizing membership functions online from initial membership functions that are selected using simple design criteria. The controller requires no regression matrix and is essentially model free.

The approach proposed by Kwan and Liu (1999) uses quantitative control schemes to ensure global stability and qualitative control scheme to approximate any non-linear functions caused by disturbances, system uncertainties and interconnections. With the PD control as a preliminary component in maintaining the local stability, non-linear feedback is added to ensure global stability of the entire system. An adaptive fuzzy logic controller is incorporated into the robot arm control system as a function approximator to compensate interconnections effect, unmodelled dynamics, friction, gravity force and uncertainties. The stability criterion of the proposed controller is developed using the Lyapunov synthesis approach. Sun and Wan (2004) have used a controller output error method to design adaptive fuzzy control system. The proposed control strategy employs a gradient descent algorithm to minimize a cost function, which is based on the error of the controller output and is minimized by tuning some or all of the parameters of fuzzy controller. The underlying idea of controller output error method is that each time the response of a plant to a set-point signal is observed, it is learnt how to repeat that response when it is required in future.

Santibanez et al. (2000) extend the idea of PD+ controller to fuzzy. The structure of PD+ control consists of a linear PD feedback plus a specific compensation of the robot dynamics. Furthermore, this control strategy has the distinguishing feature that it reduces to the PD control with gravity compensation in the particular case of set-point control. The authors show that the gains of PD+ can be varied according to a fuzzy logic system, which depends on the robot state. Global Stability of the system is also shown. Kim (2002) has proposed an independent joint fuzzy controller, which does not require an accurate manipulator dynamic model and the joint acceleration measurement, yet it guarantees asymptotic trajectory tracking despite gross robot manipulator variations. No inversion of the estimated mass matrix is also involved. It incorporates an integral term in the control law, which eliminates steady state error. The feedback control loop is guaranteed to be stable. The use of sliding mode control theory in developing an adaptive fuzzy controller is shown by Hsu and Fu (1995). They have presented adaptive robust fuzzy control architecture for robot manipulators. The control objective is to adaptively compensate for the unknown nonlinearities of robot manipulator, which is represented as a fuzzy rule-base consisting of a collection of if-then rules. The algorithm embedded in the proposed architecture can automatically update fuzzy rules and, consequently, it is guaranteed to be locally stable and to drive the tracking errors to a neighborhood of zero. An adaptive fuzzy controller, which does not require measurement of joint velocities, is discussed by Kim (2004). In this controller, adaptive fuzzy logic allows approximation of uncertain and nonlinear robot dynamics. Only one fuzzy system is used to implement the observer-controller structure of the output feedback robot system.

A self-tuning adaptive fuzzy version of the computed torque controller is discussed by Llama et al. (1998). They have shown that the computed-torque control scheme can also yield a globally asymptotically stable closed-loop system not only for constant positive definite gain matrices, but also for a class of manipulator state dependent gain matrices. This is a theoretical result with useful implications to handle real constraint of robot manipulators such as friction in the manipulator joints and torque capability limitations of their actuators. They also show application of fuzzy logic to design a self-tuner for the computed-torque control taking into account specifications of allowable actuator torques limits and desired tracking accuracy in presence of friction.

Colbaugh (1994) proposed another approach to adaptive motion control called as *performance-based adaptive control.* It is known so because the adaptive laws adjust the controller gains directly based on system performance. The development of this schemes proceeds by assuming that very little information is available concerning either the structure or the parameter values of the manipulator dynamic model. As a consequence, these methods are equally applicable to trajectory tracking in joint-space or task-space. The controller proposed is extremely simple, require very little model information, and shows good tracking performance and robustness characteristics. Loc et al. (2004) proposed an adaptive fuzzy controller based on optimal control theory. The controller does not require exact mathematical model of manipulator and takes the error vector as control input.

Many model-following adaptive fuzzy schemes have been investigated by researchers [Tsai et al. (2000)]. Koo (1995) proposed a model reference adaptive fuzzy scheme and showed that it is capable of achieving reference model tracking of a two-link robot manipulator system. He has shown that model reference scheme is capable of manipulator control by achieving adaptive feedback linearization, i.e. to asymptotically cancel the nonlinearities in the system and to place system poles in the desired locations as specified in the reference model. Golea (2002) proposed another such model-following fuzzy adaptive scheme. In his scheme the adaptive fuzzy system is trained to approximate the robot dynamic and then, based on the estimated model, a controller is designed to ensure the tracking of a stable reference model. It is proven, using Lyapunov stability, that this adaptive scheme is robust against uncertainty, external disturbance and approximation error, and achieves asymptotic tracking of a stable reference model. Kuswadi et al. (2003) also proposed another such scheme with particular reference to a hopping robot. Their approach uses linearized model to design a state feedback servo controller. Thereafter, by using fuzzy networks they have developed model reference adaptive fuzzy control in which a fuzzy network is used to compensate the nonlinearities of robot dynamics. The role of the fuzzy network is to construct a linearized model by minimizing the output error caused by nonlinearities in the robot control system through a learning mechanism.

Another class of fuzzy controllers widely reported in literature is the hybrid fuzzy controllers. These controllers combine the action of fuzzy controllers with that of some conventional control algorithm. Butkiewicz (2000) gives a comparative study of different conventional, hybrid fuzzy and adaptive fuzzy controllers. Brehm and Rattan (1993) proposed a hybrid fuzzy PID controller, which takes advantage of the properties of the fuzzy PI, and PD controllers and another method, which adds the fuzzy PD control action to the integral control action. Li et al. (2001) have proposed a hybrid P +ID controller for manipulator control. The structure of the FUZZY P+ID controller is very simple, since it is constructed by replacing the proportional term in the conventional PID controller with an incremental fuzzy logic controller. On the basis of the PID type controllers, only two additional parameters have to be adjusted to implement the FUZZY P+ID controller. These two parameters allow the controller to behave differently, depending on the values of error and error derivative. A very similar approach to hybrid fuzzy controller design has been discussed by Li (1998).

Ordonez et al. (1997) provide a good comparative study of adaptive fuzzy, conventional adaptive and nonlinear non-adaptive controllers. Lin et al. (1995) provide a comparative analysis of fuzzy and PID controllers. Their study shows the simplicity and superiority of fuzzy controllers over their PID counterpart. Khoury et al. (2004) provide a comparative evaluation of the fuzzy PID control method with respect to other methods of nonlinear control, i.e., the computed torque control method and the direct adaptive control method. They emphasize that the main advantage of the fuzzy control approach is its non-dependency on the dynamic model of the plant.

2.4 MOTIVATION FOR PRESENT STUDY

As can be seen from the brief literature survey presented in the previous sections, the problem of manipulator control is a complex and challenging task. Many methods of manipulator control have been proposed by researchers, which range from conventional to adaptive to fuzzy control etc. Furthermore each of these strategies have their own wide and varied flavors.

In view of different kinds of control strategies available, work needs to be done which could test and compare these different control strategies against a common background and suggest the advantages and disadvantages of these strategies. As can be seen from literature survey, some researchers have attempted these comparative studies but they are not very exhaustive.

As discussed in section 2.1 many conventional, model based strategies have been proposed for manipulator control. These model based schemes give good performance in case the manipulator model is known accurately enough and is working in an known environment. A study however needs to be done to compare these conventional model based control strategies for their performance against each other and also against the non-model based algorithms under various situations. With this motivation we undertake the following tasks in Chapter 4:

- Compare three model based control strategies against each other for same manipulator model and similar test trajectories
- Analyze the effect of using approximate rather than accurate parameter values in manipulator model
- Compare the performance of model based and non model based conventional control algorithms
- Study and compare the effect of manipulator picking up an unknown load during the course of its motion on its performance
- Propose and study the effect of including a modified integral action to the model based conventional control algorithms

As the efficacy of conventional control algorithms goes down with increase in uncertainty in the manipulator model, adaptive control is often cited in literature as the way out. As discussed in section 2.2 many adaptive control algorithms have been proposed in literature for manipulator control. But a comprehensive comparative study of these algorithms is by and large missing in the literature. This is the motivation for chapter 5, where we undertake the task of an exhaustive comparative study of three popular adaptive control algorithms used for manipulator control. In particular we

- Compare the performance of adaptive control algorithms for the case when the manipulator picks up and releases an unknown load during the course of its motion
- Compare the performance of adaptive control algorithms when no initial estimate of manipulator parameters are available
- Compare the performance of adaptive control algorithms when some initial estimate of manipulator parameters are available
- Investigate the effect of adding modified integral action to these adaptive controllers

The adaptive control algorithms give good performance but have their own drawbacks like being computationally expensive and difficult to prove to be stable. This has led many researchers to investigate strategies like fuzzy and neural control for manipulators. As discussed in section 2.3 many different fuzzy controllers have been proposed in literature but their efficacy for manipulator control has been rarely investigated.

Moreover there is no good comparative study existing in literature for these different fuzzy controllers. Also there is lot of scope to investigate new hybrid fuzzy controllers, which are obtained as combinations of fuzzy with conventional or adaptive controllers. This forms the main motivation for chapter 6 where we have undertaken the following tasks:

- Investigate lookup table based and non lookup table based pure fuzzy controllers
- Investigate a self organizing fuzzy controller for situation where the lookup tables start with zero and non zero values
- Propose and investigate some new hybrid fuzzy controllers
- Investigate a self tuning adaptive fuzzy controller
- Investigate a coarse/fine adaptive fuzzy controller
- Investigate the performance of these controllers for the cases when manipulator parameters change during motion and when they do not change
- Do a comparative study of performance of the above controllers

The main motivation thus, for the present study is absence of good comparative study of different control algorithms for manipulator control against a common background. We have made an attempt in this thesis to do the same and in the process have also proposed some new controllers and some modifications to the existing controllers with a view to improve their performance.

2.5 CONCLUDING REMARKS

In this chapter we have presented a brief overview of studies carried out in the field of manipulator control by different researchers. In particular we have presented the work done in areas of conventional, adaptive and fuzzy control. The literature survey represents a small and important portion of a vast body of literature available in this area. We have not included in the survey work done on manipulator control in the areas of neural networks, genetic algorithms, impedance control etc, as we do not intend to investigate these schemes in this thesis.

CHAPTER III

ROBOT DYNAMICS AND ISSUES IN CONTROL

3.0 INTRODUCTION

A robot manipulator consists of number of links interconnected by joints to form a kinematic chain. Figure 3.1 shows a serial link (left) and a parallel link (right) manipulator. A parallel link robot, by definition, contains two or more independent serial link chains. In this thesis for simplicity of analysis we confine ourselves to serial link manipulators with only rotational or revolute joints. Also most of the robots used in industry today have this serial open kinematic chain structure. Anyway, most of our discussion about control strategies in the thesis remains valid for parallel robots and for robots with sliding or prismatic joints as well.

In the serial open kinematic chain structure, a number of links are connected in series through joints, which are either revolute or prismatic (linear) in nature. Each joint usually has a single degree of freedom.

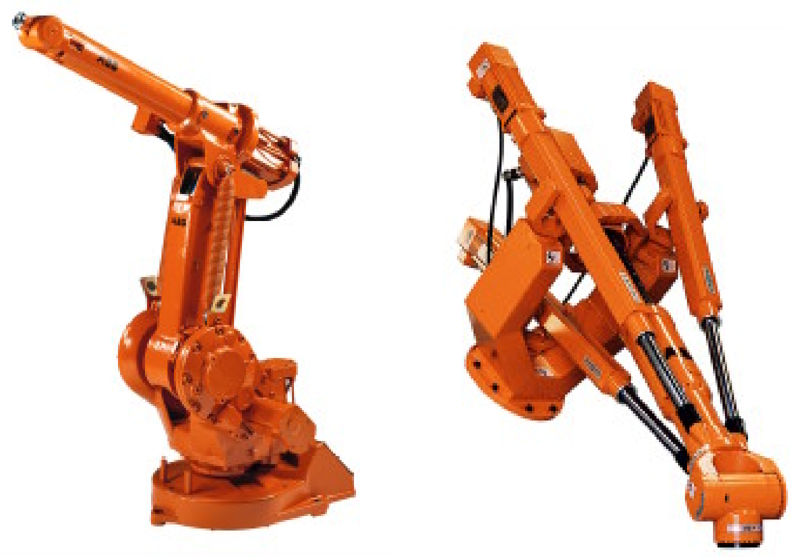


Fig. 3.1. A serial manipulator (left), the ABB IRB1400, and a parallel manipulator (right), the ABB IRB940Tricept.

For the purpose of studying manipulator dynamics, it is usually assumed that the manipulator consists of rigid links with no flexibility. This assumption is quite true for the industrial grade manipulator and does not hold good only for large arms designed for space applications etc. Rigid robot manipulators are fully actuated, i.e., there is an independent control input for each degree–of–freedom. By contrast, robots possessing joint or link flexibility are no longer fully actuated and the control problem is more difficult, in general.

The problem of flexibility, if any, in the industrial manipulator is avoided by setting the controller gains in such a way that the natural frequency of the system lies far away from the lowest resonance frequency of the structure so as not to excite them.

3.1 MANIPULATOR DYNAMICS EQUATIONS

The two most common methods used to derive the manipulator inverse dynamics are the **Newton-Euler** and the **Lagrange** methods [Spong and Vidyasagar (1989)]. The Newton-Euler method is based on force balance approach while the Lagrangian method is based on energy conservation approach. The Lagrangian approach is easier if the number of degrees of freedom, i.e., the number of joints of the manipulator is less than four. The Newton-Euler approach is more suitable for implementation on computer because of its iterative nature. However as the dynamics equations are very explicit and cumbersome even for the simplest of manipulator, it is always better to use the closed form solution. This saves a lot of processor time, thereby making the real time implementation much more easier [Paul (1972)].

The general state space representation of the manipulator inverse dynamics is given by equation 3.1.1 below.

$$\tau = M\left(\theta\right)\ddot{\theta} + V\left(\theta,\dot{\theta}\right) + F\left(\theta,\dot{\theta}\right) + G\left(\theta\right) \qquad (3.1.1)$$

Where:

τ is $n \times 1$ vector of joint torques,

$M\left(\theta\right)$ is $n \times n$ matrix called the manipulator mass matrix,

$V\left(\theta,\dot{\theta}\right)$ is $n \times 1$ matrix consisting of terms arising due to centrifugal and coriolis forces,

$F\left(\theta,\dot{\theta}\right)$ is $n \times 1$ matrix consisting of terms arising due to friction forces,

$G\left(\theta\right)$ is $n \times 1$ matrix consisting of terms arising due to gravity,

$\ddot{\theta}$ is $n \times 1$ vector of joint accelerations,

$\dot{\theta}$ is $n \times 1$ vector of joint velocities,

θ is $n \times 1$ vector of joint positions and

n is the degrees of freedom of the manipulator, equal to the number of joints.

Equation 3.1.1 can also be written in another form as:

$$\tau = M(\theta)\ddot{\theta} + V_M(\theta,\dot{\theta})\dot{\theta} + F_M(\theta,\dot{\theta})\dot{\theta} + G(\theta) \ . \tag{3.1.2}$$

Where $V_M(\theta,\dot{\theta})$ and $F_M(\theta,\dot{\theta})$ are now $n \times n$ matrices.

The various matrices in equations 3.1.1 and 3.1.2 have some typical properties and relations with each other. These are often exploited for designing a controller and for proving its stability [Craig (1988)]. Some of the important properties are listed below:

Mass matrix, $M(\theta)$

- It is symmetric.
- It is positive definite and bounded above and below, i.e., for an n x n identity matrix I_n and for scalars α_m and β_m which satisfy $0<\alpha_m<\beta_m$ we can say that $\alpha_m I_n \le M(\theta) \le \beta_m I_n$.
- Its inverse exists and is positive definite and bounded.
- Its time derivative is given by $2V_M(\theta,\dot{\theta}) - J$, where J is some skew symmetric matrix. This implies that $0.5X^T\dot{M}(\theta)X = X^T V_M(\theta,\dot{\theta})X$ where X is an n x 1 vector.
- The mapping $\tau \to \dot{q}$ is passive, i.e., there exists $\beta \ge 0$ such that

$$\int_0^T \dot{q}^T(u)\tau(u)\,du \ge -\beta$$

Centrifugal and Coriolis force matrix, $V(\theta,\dot{\theta})$

- It has a bound, which is independent of θ but increases quadratically with $\dot{\theta}$.
- It is related to time derivative of manipulator mass matrix as above.

Friction force matrix, $F\left(\theta,\dot{\theta}\right)$

- Position dependence comes only when eccentricity of gears is present.
- In highly geared robots, the friction forces can account for almost 25% of the total torque required.
- Friction is a local effect, so $F\left(\theta,\dot{\theta}\right)$ is uncoupled.
- Friction forces are dissipative, i.e. $\dot{\theta}^T F\left(\theta,\dot{\theta}\right)\dot{\theta} \geq 0$.
- Friction forces are largely viscous in nature.
- If only viscous friction is modelled, then $F\left(\theta,\dot{\theta}\right)$ is a diagonal matrix with viscous friction coefficients as the elements.

Gravity force matrix, $G(\theta)$

- It consists of all gravity related terms.
- It has a bound that is independent of θ.

The dynamic equations of the manipulator used for our simulations are now derived.

3.2. TWO LINK MANIPULATOR DYNAMICS

The manipulator used for simulations is a simple two degree-of-freedom articulated arm. This is a very standard test bed used for studies on control in robotics literature. The two joints of this manipulator are assumed to be driven by DC permanent magnet servomotors. The manipulator is assumed to be of direct drive type, i.e., no gearings are used at the joints. This is usually the case for high speed and high precision manipulators. The control strategies were tested on this two link planar manipulator. This allows us primarily to reduce the amount of calculations that have to be made during runtime while still including the effect of gravity on motion. Figure 3.2.1 shows the manipulator with frames assigned to the links.

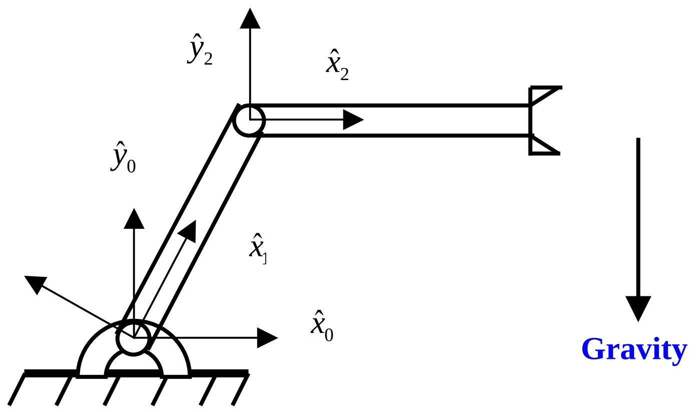


Fig 3.2.1 Manipulator used for experiments with frames attached

In Fig. 3.2.1 the axes $\hat{z}_0, \hat{z}_1$ and $\hat{z}_2$ are perpendicular to the plane of the paper and point out.

The inverse dynamics is derived using the Lagrange method [Craig (1989)]. The joints were assumed to have only viscous friction. This model was used for all simulations. The various manipulator parameters and variables used in the model are:

m_i = Mass of the *i*-th link (kg)

l_i = Length of the *i*-th link (m)

x_i = Location of the centre of mass of the *i*-th link along the respective x- axis (m)

${}^{i}v_{ci}$ = Linear velocity of the centre of mass of the *i*-th link as seen in the *i*-th frame (m/sec)

${}^{i}\omega_i$ = Angular velocity of the *i*-th link expressed in the *i*-th frame (rad/sec)

I_{zzi} = Moment of inertia of the *i*-th link about z_i axis (kg-m^2)

${}^{ci}I_i$ = Inertia tensor of the *i*-th link with respect to a frame having its origin at centre of mass of *i*-th link and axes parallel to the faces of the link (kg-m^2)

${}^{i}P_{ci}$ = Vector location of centre of mass of *i*-th link with respect to *i*-th frame (m)

${}^{i}v_j$ = Linear velocity of origin of *j*-th frame with respect to *i*-th frame

${}^{i}_{j}R$ = Orientation difference between frames *i* and *j*

${}^{0}g$ = Gravity vector with respect to base frame

As both the links of the manipulator are symmetric cuboids, we have approximated all the off diagonal terms of the inertia tensor to zero by proper selection of frames. The kinetic energy, K_i of the i-th link is given by equation 3.2.1.

$$K_i = 0.5\left(m_i \, {}^{i}v_{ci}^{T} + {}^{i}\omega_i^{T} \, {}^{ci}I_i \, {}^{i}\omega_i \right) \qquad (3.2.1)$$

For the first link i=1. The kinetic energy for this link is given by equation 3.2.2

$$K_1 = 0.5\left(m_1 \, {}^{1}v_{c1}^{T} + {}^{1}\omega_1^{T} \, {}^{c1}I_1 \, {}^{1}\omega_1 \right) \qquad (3.2.2)$$

Also

$${}^{1}\omega_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \quad \text{and} \qquad (3.2.3)$$

$${}^{1}v_{c1} = {}^{1}\omega_1 \times {}^{1}P_{c1}$$

$$= \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \times \begin{bmatrix} x_1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ x_1\dot{\theta}_1 \\ 0 \end{bmatrix} \qquad (3.2.4)$$

Substituting 3.2.3 and 3.2.4 in 3.2.2 we get

$$K_1 = 0.5\left(m_1 x_1^2 \dot{\theta}_1^2 + I_{zz1}\dot{\theta}_1^2 \right) \qquad (3.2.5)$$

Similarly for the second link, the kinetic energy, K_2 is given by equation 3.2.6

$$K_2 = 0.5\left(m_2 \, {}^{2}v_{c2}^{T} \, {}^{2}v_{c2} + {}^{2}\omega_2^{T} \, {}^{c2}I_2 \, {}^{2}\omega_2 \right) \qquad (3.2.6)$$

Also

$${}^{2}\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 + \dot{\theta}_2 \end{bmatrix} \quad \text{and} \qquad (3.2.7)$$

$${}^{2}v_{c2} = {}^{2}_{1}R \, {}^{1}v_2 + {}^{2}\omega_2 \times {}^{2}P_{c2}$$

$$= \begin{bmatrix} C_2 & S_2 & 0 \\ -S_2 & C_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ l_1\dot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ x_2\left(\dot{\theta}_1 + \dot{\theta}_2\right) \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} S_2 l_1 \dot{\theta}_1 \\ C_2 l_1 \dot{\theta}_1 + x_2\left(\dot{\theta}_1 + \dot{\theta}_2\right) \\ 0 \end{bmatrix} \qquad (3.2.8)$$

Here C_2 is same as $\cos\theta_2$ and S_2 is same as $\sin\theta_2$.

Substituting 3.2.7 and 3.2.8 in 3.2.6 we get kinetic energy for second link as

$$K_2 = 0.5 m_2 \left[l_1^2 \dot{\theta}_1^2 + x_2^2\left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 + 2C_2 l_1 x_2 \dot{\theta}_1\left(\dot{\theta}_1 + \dot{\theta}_2\right)\right] + 0.5 I_{zz2}\left(\dot{\theta}_1 + \dot{\theta}_2\right)^2 \qquad (3.2.9)$$

The total kinetic energy, *K*, of the manipulator is given by equation 3.2.10 as

$$K = K_1 + K_2 \qquad (3.2.10)$$

For calculating the potential energy of the manipulator, we know that the potential energy of the *i*-th link, P_i, is given by

$$P_i = -m_i\, {}^0g^T\, {}^0P_{ci} + C_i \qquad (3.2.11)$$

Where C_i is a constant chosen such that the potential energy never becomes negative.

Using equation 3.2.11 the potential energy of the first link is given by

$$P_1 = -m_1\, {}^0g^T\, {}^0P_{c1} + C_1$$

$$= m_1 g x_1 \sin\theta_1 + C_1 \qquad (3.2.12)$$

Similarly the potential energy of second link is given by

$$P_2 = -m_2\, {}^0g^T\, {}^0P_{c2} + C_2$$

$$= m_2 g\left(l_1 \sin\theta_1 + x_2 \sin\left(\theta_1 + \theta_2\right)\right) + C_2 \qquad (3.2.13)$$

The total potential energy, P, of the manipulator is now given by

$$P = P_1 + P_2 \qquad (3.2.14)$$

The Lagrangian, L, for the manipulator can be calculated as

$$L = K - P \qquad (3.2.15)$$

The torques, τ, required at the joints to give the desired acceleration and velocity can be calculated as

$$\tau = \frac{d}{dt}\left(\frac{\partial L}{\partial \dot{\theta}}\right) - \frac{\partial L}{\partial \theta} \qquad (3.2.16)$$

or $$\tau = \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}}\right) - \frac{\partial K}{\partial \theta} + \frac{\partial P}{\partial \theta} \qquad (3.2.17)$$

From equation 3.2.17 we calculate torque required at joint 1, τ_1, as

$$\begin{aligned}\tau_1 &= \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}_1}\right)) + \frac{\partial P}{\partial \theta_1} - \frac{\partial K}{\partial \theta_1} \\ &= [(m_2 x_2^2 + 2m_2 x_2 c_2 l_1 + m_1 x_1^2 + m_2 l_1^2) + I_{zz1} + I_{zz2}]\ddot{\theta}_1 - m_2 x_2 s_2 l_1 \dot{\theta}_2^2 - \\ &\quad (2m_2 x_2 s_2 l_1 \dot{\theta}_2 + F_1)\dot{\theta}_1 + m_2 x_2 g c_{12} + (m_1 x_1 + m_2 l_1) g c_1 + (m_2 x_2^2 + m_2 x_2 c_2 l_1 + I_{zz2})\ddot{\theta}_2\end{aligned}$$

(3.2.18)

Similarly the torque at joint 2, τ_2 is given by

$$\begin{aligned}\tau_2 &= \frac{d}{dt}\left(\frac{\partial K}{\partial \dot{\theta}_2}\right) + \frac{\partial P}{\partial \theta_2} - \frac{\partial K}{\partial \theta_2} \\ &= (m_2 l_1 x_2 c_2 + m_2 x_2^2 + I_{zz2})\ddot{\theta}_1 + (m_2 x_2^2 + I_{zz2})\ddot{\theta}_2 + m_2 x_2 l_1 s_2 \dot{\theta}_1^2 + m_2 x_2 g c_{12} + F_2 \dot{\theta}_2\end{aligned}$$

(3.2.19)

F_1 in equation 3.2.18 and F_2 in equation 3.2.19 are coefficients of viscous friction present at joints 1 and 2 respectively.

Equations 3.2.18 and 3.2.19 can be written in standard state space format of equation 3.1.2 as

$$\begin{bmatrix}\tau_1 \\ \tau_2\end{bmatrix} = \begin{bmatrix} [(m_2 x_2^2 + 2m_2 x_2 c_2 l_1 + m_1 x_1^2 + m_2 l_1^2) + I_{zz1} + I_{zz2}] & (m_2 x_2^2 + m_2 x_2 c_2 l_1 + I_{zz2}) \\ (m_2 l_1 x_2 c_2 + m_2 x_2^2 + I_{zz2}) & m_2 x_2^2 + I_{zz2}\end{bmatrix}\begin{bmatrix}\ddot{\theta}_1 \\ \ddot{\theta}_2\end{bmatrix} +$$

$$\begin{bmatrix} -2m_2 x_2 s_2 l_1 \dot{\theta}_2 & -m_2 x_2 s_2 l_1 \dot{\theta}_2 \\ m_2 x_2 l_1 s_2 \dot{\theta}_1 & 0 \end{bmatrix}\begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix} + \begin{bmatrix}F_1 \\ F_2\end{bmatrix}\begin{bmatrix}\dot{\theta}_1 \\ \dot{\theta}_2\end{bmatrix} +$$

$$\begin{bmatrix} m_2 x_2 g c_{12} + (m_1 x_1 + m_2 l_1) g c_1 \\ m_2 x_2 g c_{12}\end{bmatrix}$$

$$= M(\theta)\ddot{\theta} + V_M(\theta,\dot{\theta})\dot{\theta} + F_M \dot{\theta} + G(\theta)$$

(3.2.20)

The manipulator dynamics in equation 3.2.20 can be rewritten in a linear form by simple rearrangement of terms. In other words, there is a constant vector $P \in \mathbb{R}^m$ and a function $W\left(\theta.\dot{\theta}.\ddot{\theta}\right) \in \mathbb{R}^{n\times m}$ such that

$$M\left(\theta\right)\ddot{\theta} + V\left(\theta,\dot{\theta}\right)\dot{\theta} + F\left(\theta,\dot{\theta}\right)\dot{\theta} + G\left(\theta\right) = W\left(\theta,\dot{\theta},\ddot{\theta}\right)P = \tau$$

Here, m is the dimension of the parameter space and is non-unique in general. The function $W\left(\theta,\dot{\theta},\ddot{\theta}\right)$ is called the ***Regressor***. The parameter vector P is comprised of link masses, moments of inertia etc. The properties of linear parameterisation and passivity are very important from point of view of controller design. Using these properties researchers have been able to prove elegant global convergence and stability results for robust and adaptive control [Yu and Arteaga (1994)].

The linear form of manipulator dynamics is particularly suitable for derivation of adaptation laws. To derive the linear form for our two-link manipulator, we define manipulator parameter constants as:

$$P_1 = I_{zz1} + I_{zz2} + m_2(x_2^2 + l_1^2) + m_1 x_1^2 \tag{3.2.21}$$

$$P_2 = 2m_2 l_1 x_2 \tag{3.2.22}$$

$$P_3 = m_2 x_2^2 + I_{zz2} \tag{3.2.23}$$

$$P_4 = F_1 \tag{3.2.24}$$

$$P_5 = F_2 \tag{3.2.25}$$

$$P_6 = g(m_1 x_1 + m_2 l_1) \tag{3.2.26}$$

$$P_7 = g m_2 x_2 \tag{3.2.27}$$

It can be seen that all these constants are functions of manipulator parameters like, mass of link, moment of inertia, link length etc.

The manipulator inverse dynamics can now be written in a linear form as

$$\tau = W(\theta,\dot{\theta},\ddot{\theta})P \tag{3.2.28}$$

Where $P = \begin{bmatrix} P_1 & P_2 & P_3 & .. & .. & P_7 \end{bmatrix}^T$ is the vector of manipulator parameters and $W\left(\theta,\dot{\theta},\ddot{\theta}\right)$ is the 2 x 7 manipulator regressor matrix that has terms that are non-linear in nature and that depend on manipulator kinematics.

For the two-link manipulator under consideration, the various terms of W matrix are:

$W_{11} = \ddot{\theta}_1$

$W_{12} = C_2(\ddot{\theta}_1 + 0.5\ddot{\theta}_2) - 0.5S_2\dot{\theta}_2(\dot{\theta}_2 - \dot{\theta}_1)$

$W_{13} = \ddot{\theta}_2$

$W_{14} = \dot{\theta}_1$

$W_{15} = 0$

$W_{16} = C_1$

$W_{17} = C_{12}$

$W_{21} = 0$

$W_{22} = 0.5C_2\ddot{\theta}_1 + 0.5S_2\dot{\theta}_1^2$

$W_{23} = \ddot{\theta}_1 + \ddot{\theta}_2$

$W_{24} = 0$

$W_{25} = \dot{\theta}_2$

$W_{26} = 0$

$W_{27} = C_{12}$

Here C_{12} is same as $\cos(\theta_1 + \theta_2)$.

The model derived above has been used throughout for simulation studies. The above model includes all effects considering the rigid body behaviour of the individual links.

The actual values of the various manipulator parameters used for simulation are given below in Table.3.2.1. These values are based on the Link 2 and Link 3 parameter values for the CRS Plus manipulator.

$m_1 = 2.0\ kg$
$m_2 = 2.0\ kg$
$l_1 = 0.26\ m$
$x_1 = 0.13\ m$
$x_2 = 0.14\ meters$
$I_{zz1} = 0.09\ kg - m^2$
$I_{zz2} = 0.09\ kg - m^2$
$F_1 = 2.5\ N - m/rad/\sec$
$F_2 = 2.5\ N - m/rad/\sec$

Table.3.2.1. Actual manipulator parameters

3.3 ACTUATOR DYNAMICS

The simplest modification to the rigid robot model given by equation 3.2.20 is the inclusion of the actuator inertia matrix I [Craig (1989)]. The actuator inertia matrix I is an *n* x *n* diagonal matrix,

I = diag(I1; . . .; In) (3.3.1)

where Ii is the actuator inertia of the *i*-th joint.

Defining, $D(\theta) = M(\theta) + I$, we may modify the dynamics to include these additional terms as

$$\tau = D(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + F(\theta,\dot{\theta}) + G(\theta) \tag{3.3.2}$$

As can be seen, the inclusion of the actuator inertias and friction does not change the order of the equations.

If the joints are actuated with permanent magnet DC motors we may write the actuator dynamics as

$$L\frac{di}{dt} + Ri = V - K_b\dot{q} \tag{3.3.3}$$

where *i*, *V* are vectors representing the armature currents and voltages, and *L*, *R*, K_b are matrices representing, respectively, the armature inductances, armature resistances, and back e.m.f constants.

Since the joint torque τ and the armature current *i* are related by $\tau = K_m i$, where K_m is the torque constant of the motor, we may write the complete system (3.3.1)-(3.3.3) as

$$D(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + F(\theta,\dot{\theta}) + G(\theta) = K_m i \tag{3.3.4}$$

$$L\frac{di}{dt} + Ri = V - K_b\dot{q} \tag{3.3.5}$$

In addition, whenever the manipulator is in contact with the environment, the complete dynamic description includes the dynamics of the environment and the coupling forces between the environment and the manipulator. Modeling all of these effects produces an enormously complicated model. The key in robot control system design is to model the most dominant dynamic effects for the particular manipulator under consideration and to design the controller so that it is insensitive or robust to the neglected dynamics.

In the model considered in this thesis, we have neglected all the environment interaction effects and also all the joint flexibilities. We have only considered the inertia of motors and have included it in the dynamic model. However some work has been done by researchers on manipulator control with actuator dynamics included in the manipulator model [Purwar et al. (2004), Ham et al. (1995)].

3.4 ERROR NORMS

For a quantitative comparison of performance of various controllers, three values of errors in position have been used. They are:

- Maximum absolute error at any time during the course of entire trajectory.
- Steady state error of the joints, which is formally defined as

$$e_{s.s} = \lim_{t\to\infty} e(t)$$

- Root Mean Square average of the error (*e*) or the L^2 norm. This norm is defined as

$$L^2\left[e(t)\right] = \left(\left(\frac{1}{t}\right)\int_{t_0}^{t}\left\|e(t)\right\|^2 dt\right)^{\frac{1}{2}}$$

$$= \sqrt{\frac{\int_{t_0}^{t}\left\|e(t)\right\|^2 dt}{t}}$$ which for discrete time case becomes,

$$= \sqrt{\frac{\sum_i \left\|e_i(t)\right\|^2 t_s}{t}}$$

$$= \sqrt{\frac{\sum_i \left\|e_i(t)\right\|^2}{N}}$$ where

$N = \dfrac{t}{t_s}$ is the total number of samples over the entire trajectory

$t =$ total time taken for the trajectory

$t_s =$ sampling time

These error norms cover well the various aspects of manipulator performance for the test trajectories that we have chosen. The test trajectories are described in detail in Chapter 4.

3.5 CONCLUDING REMARKS

In this chapter we have highlighted a few important properties of the individual matrices that comprise the manipulator dynamics equation. These properties are widely used for controller design and in particular for proving their stability.

We have also derived the two-link planar manipulator dynamics equations using the Lagrangian method. These equations comprise the mathematical model of the manipulator and are used in all simulation studies.

Finally the various error norms used for comparative analysis of performance of various controllers are discussed.

CONVENTIONAL CONTROL OF ROBOT MANIPULATORS

4.0 INTRODUCTION

The problem of manipulator control is a highly complex problem of controlling a system which is multi-input, multi-output, non-linear and time variant. The general structure of a manipulator with controller is shown in figure 4.1 below.

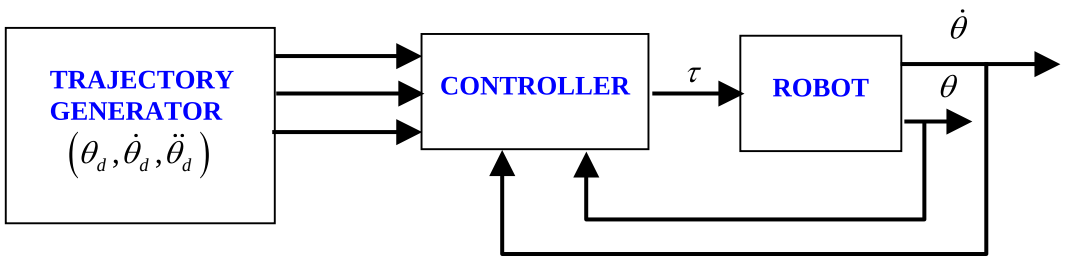


Fig 4.1 General structure of robot control system

Because of the complexity of both the kinematics and dynamics of the manipulator and of the task to be carried out, the motion control problem is generally decomposed into three stages, Motion Planning, Trajectory Generation, and Trajectory Tracking [Spong et al. (1992)]. In the motion planning stage, desired paths are generated in the Task Space without timing information, i.e., without specifying velocity or acceleration along the paths. Of primary concern is the generation of collision free paths in the workspace. In the trajectory generation stage, the desired position, velocity, and acceleration of the manipulator along the path, as a function of time are computed. The trajectory planner may parameterize the end-effector path directly in Task Space or it may compute a trajectory for the individual joints of the manipulator as a curve in the Configuration Space.

In order to compute a Joint Space trajectory, the given end-effector path must be transformed into a Joint Space path via the inverse kinematics mapping. Because of the difficulty of computing this mapping on-line, the usual approach is to compute a discrete set of joint vectors along the end-effector path and to perform an interpolation in Joint Space among these points in order to complete the Joint Space trajectory. Common approaches to trajectory interpolation include polynomial spline interpolation, using trapezoidal velocity trajectories or cubic/quintic polynomial trajectories.

The computed reference trajectory is then presented to the controller, whose function is to cause the robot to track the given trajectory as closely as possible. This thesis is mainly concerned with the design and analysis of the tracking controllers assuming that the path and trajectory have been precomputed.

The trajectory generator provides the controller with information about the desired position, velocity and acceleration $\left(\theta_d, \dot{\theta}_d, \ddot{\theta}_d\right)$ for each joint and keeps updating this information at the path update rate, which usually lies in the range of 20 to 200 Hz [Khosla (1987)]. The controller takes this information and compares it with the present (actual) position and velocity (sometimes acceleration also) of joints $\left(\theta, \dot{\theta}, \ddot{\theta}\right)$, which are provided as feedback through the sensors (usually optical encoders and tachogenerators). Based upon the error between the desired and actual values, the controller calculates a vector of joint torques (τ) that should be applied at respective joints by the actuators to minimise these errors. The torques are calculated using a ***control law***. The goal of the controller is thus, minimisation of the error, e and its first derivative $\dot{e}$ (and sometimes the second derivative $\ddot{e}$ also). Here e is calculated as

$$e = \theta_d - \theta \tag{4.1}$$

and $\dot{e}$ as

$$\dot{e} = \dot{\theta}_d - \dot{\theta} \tag{4.2}$$

where θ is the vector of actual joint positions and $\dot{\theta}$ that of actual joint velocities.

There are various possible controller configurations for manipulator control. In this chapter we analyse some common conventional manipulator controller architectures [Luh (1983)].

The conventional control strategies can be broadly classified as **Linear** and **Non Linear** strategies. We first discuss the essence of linear control of manipulators.

4.1. LINEAR CONTROL OF MANIPULATORS

The use of linear control techniques for any system is valid only when the system to be controlled can be modelled by **linear differential equations**. Thus the linear control of robot manipulators is essentially an approximation, as the manipulator dynamics is described by highly non linear equations. The linear control strategies for robots give excellent performance for manipulators having highly geared joints. This is the case with most of the industrial robots in use today. These controllers assume that each joint is a decoupled, independent entity. Further it is also assumed that each actuator on each joint is driving a constant inertia load. All these assumptions hold good if the manipulator joints are highly geared. One common linear control strategy known as PD (proportional

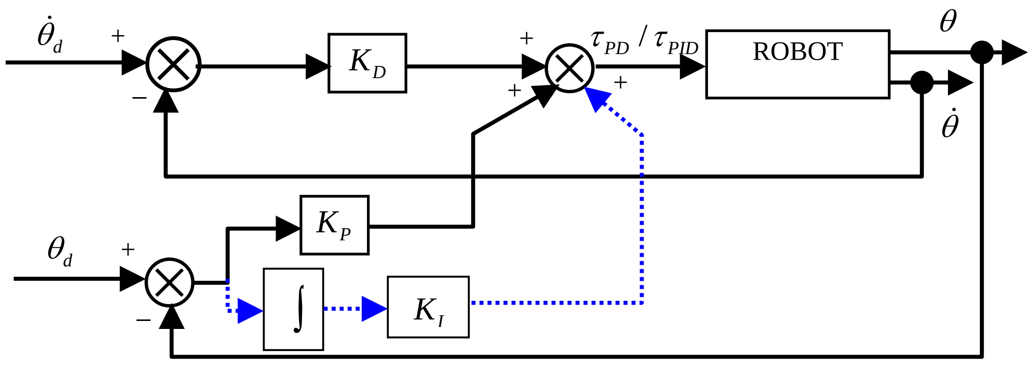


Fig 4.1.1. Block diagram of fixed proportional plus derivative plus integral (PID) feedback control

plus derivative) control is shown in Fig 4.1.1.

The control law used for this strategy is given by equation 4.1.1.

$$\tau_{PD} = K_D \dot{e} + K_P e \qquad (4.1.1)$$

where $\dot{e}$ and e are error in velocity and position, and K_P and K_D are the controller gain matrices. τ_{PD} is the vector of joint torques. Usually the gain matrices are chosen to be diagonal because of the assumption of decoupled nature of the joints, and the diagonal elements are chosen to be greater than zero (required for stability).

All the control strategies investigated in this section and the next section (4.2) were tested against two trajectories. In the first test trajectory (Fig. 4.1.2(a)), the direction of rotation of the joints was not reversed. Here the first joint moves from its initial position of 0° to 90° degrees in 5 seconds and then stays there for another 5 seconds. The second joint on the other hand was made to move from 0° degrees to –90° degrees in 5 seconds and was held there for next 5 seconds.

In the second test trajectory, the direction of motion of joints was reversed in between the motion. This produces a greater stress on the controllers. In this trajectory (Fig.4.1.2 (b)), the first joint of the manipulator was required to move from its initial position at 0° to 90° and then back to 0° in 10 seconds. It was further required to hold this joint at 0° degrees for another 5 seconds. The second joint had to trace a similar path starting from 0°, going to –90° and then again coming back to its initial position of 0° in 10 seconds. The second joint then stays at 0° for a further time of 5 seconds.

Both the test trajectories were defined using quintic polynomials, which satisfied the conditions of zero velocity and acceleration at the beginning and at the end of motion. The path update rate was selected as 333 Hz. Thus the trajectory generator supplies the controller a new set point at every 3msec interval. The manipulator control loop runs 5 times during this interval between two set points.

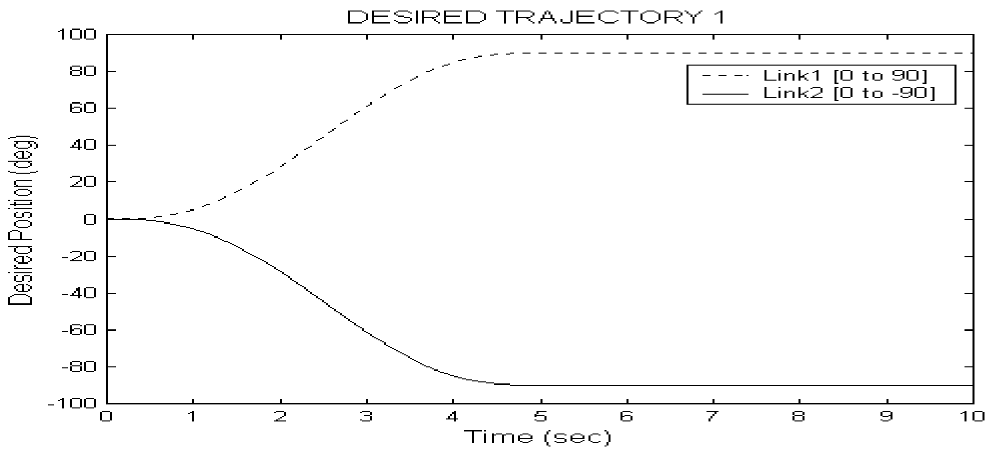


Fig. 4.1.2(a) Desired trajectory 1 (Fixed Parameters)

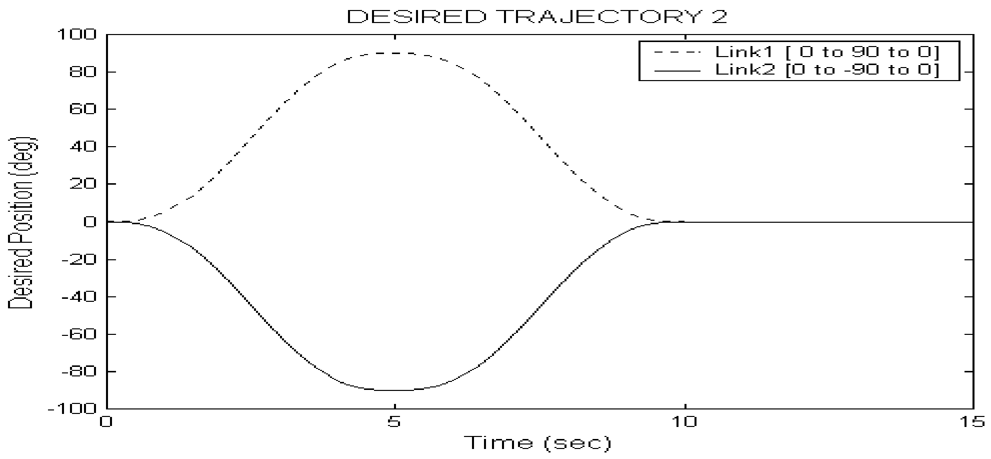


Fig. 4.1.2(b) Desired trajectory 2 (Fixed Parameters)

The performance of PD controller was tested for the above two trajectories. For simulation it was assumed that we have perfect knowledge of manipulator parameters and their values were taken as shown in Table 3.2.1.

The simulation results (error profiles) for this simple control strategy are shown in Fig.4.1.3 (a) and Fig. 4.1.3 (b) for the two different trajectories. The root mean square (RMS) value and the steady state (S.S) value of the errors for the two joints are tabulated in Table 4.1.1.

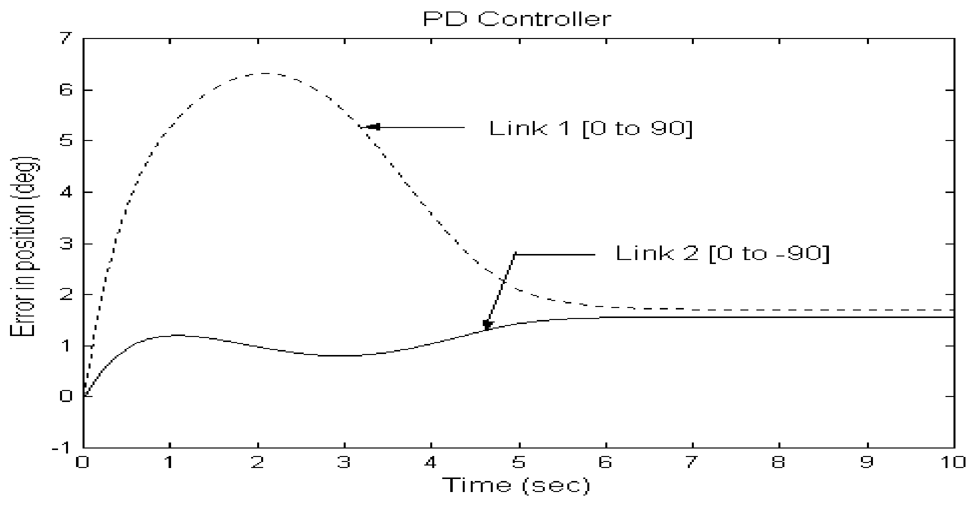


Fig. 4.1.3(a) Errors for PD control (Trajectory 1, Fixed Parameters)

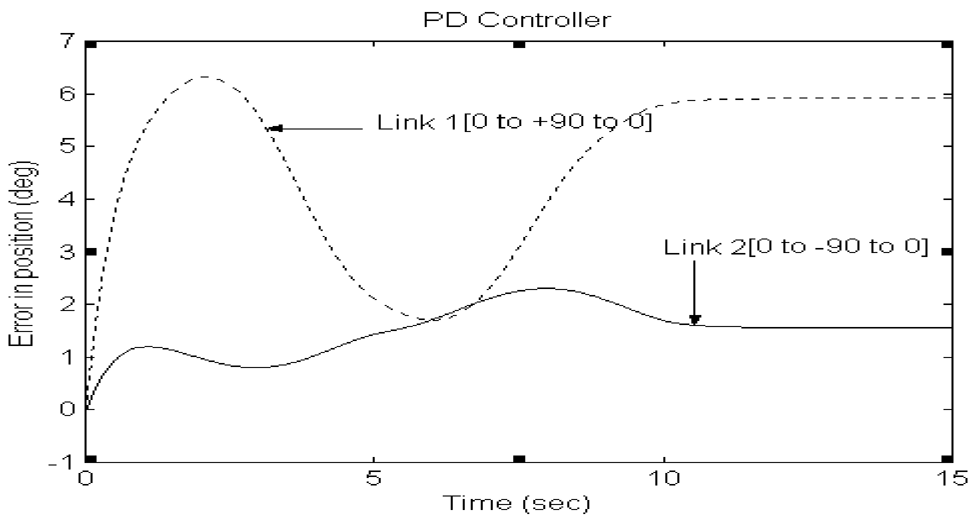


Fig. 4.1.3(b) Errors for PD control (Trajectory 2, Fixed Parameters)

PD Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°		**Link2 0°→ -90°**		**Link1 0°→ 90°→0°**		**Link2 0°→ -90°→0°**	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
3.6101	1.6990	1.3115	1.5660	4.8932	5.9127	1.5740	1.5583

Table 4.1.1 Errors for PD control (Fixed Parameters)

For simulation the PD controller gain matrices were chosen to be diagonal with $K_P = 100$ and $K_D = 50$. Increasing the values of K_P and K_D lead to smaller errors but increase the chances of exciting the manipulator resonance. We chose these values through repeated simulations mainly through trial and error.

The problem with PD controllers is that they do not guarantee exact trajectory tracking, i.e. $\lim_{t\to\infty} e(t) \neq 0$

This can also be seen Fig 4.1.3 (a) and 4.1.3 (b), which show a finite steady state errors. They only guarantee the error $\|e(t)\|$ to be bounded. The steady state magnitude of $\|e(t)\|$ may be reduced to some extent by selecting higher gains [Wen et al. (1987)]. The upper limit to the value of these gains is dictated by the unmodelled flexibility of the manipulator. Higher gains may excite the natural resonance frequencies of the manipulator and cause the whole structure to become unstable.

Usually an integral term is also used in the control law and is shown by the dashed line in Fig. 4.1.1. Equation 4.1.1 is then modified as

$$\tau_{PID} = K_D\dot{e} + K_P e + K_I \int e dt \qquad (4.1.2)$$

where K_I once again is a diagonal matrix with small scalar values to keep the higher order effects at minimum.

The simulation results (error profiles) of the PID control strategy are shown in Fig.4.1.4(a) and Fig. 4.1.4 (b) for the two different trajectories. The root mean square and the steady state values of the errors for the two joints are tabulated in Table 4.1.2. In this simulation the K_I gain matrix was chosen as diagonal with elements equal to 0.25.

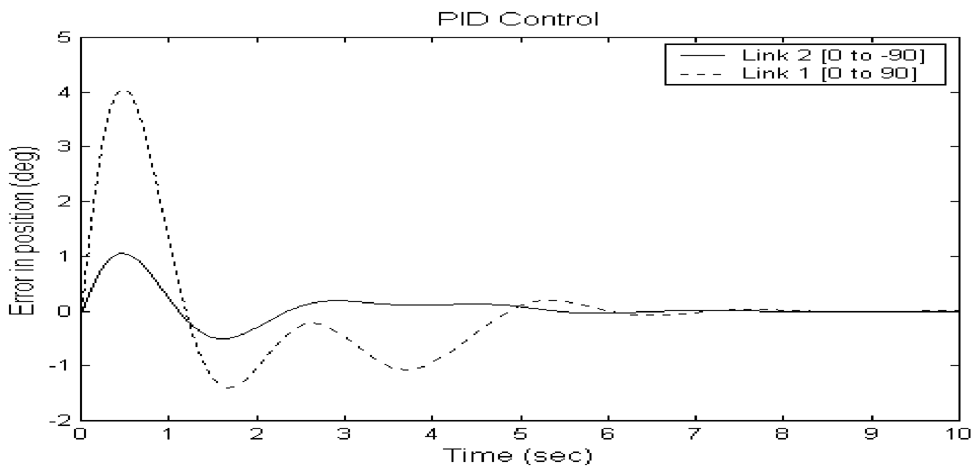


Fig. 4.1.4(a) Errors for PID control (Trajectory 1, Fixed Parameters)

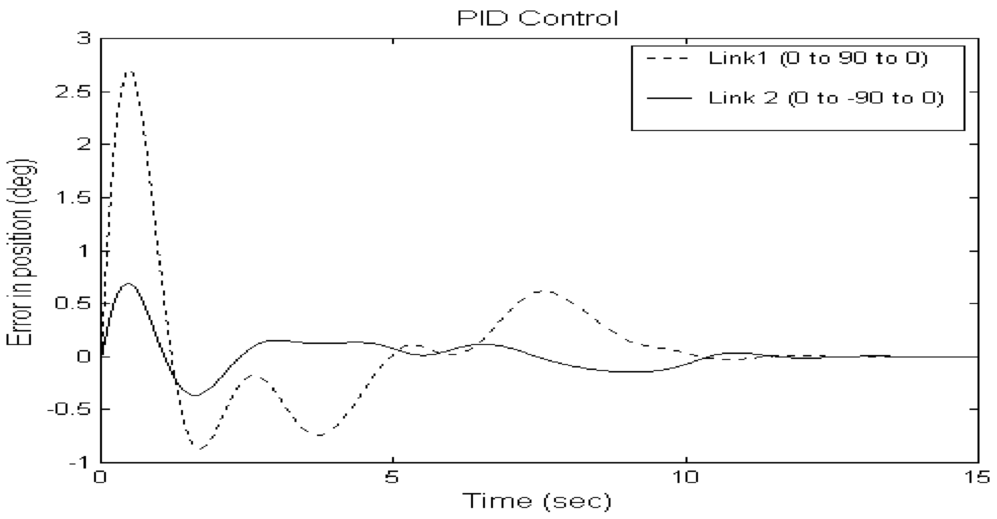


Fig. 4.1.4(b) Errors for PID control (Trajectory 2, Fixed Parameters)

PID Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°**		**Link2** **0°→ -90°**		**Link1** **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
0.7248	0.0	0.1944	0.0	0.6195	-0.0001	0.1682	-0.0003

Table 4.1.2 Errors for PID control (Fixed Parameters)

Comparing Tables 4.1.1 and 4.1.2, although the introduction of the integral term in the control law results in significantly low values of errors as compared to PD control, it also increases the order of the system, which can result in system instability. As shown in Fig 4.1.4(c), even a small value of $K_I = 5$ introduces appreciable oscillations in the system and degrades its performance. The integral term thus is almost always avoided in the real manipulator controller implementations.

All these limitations make linear control of manipulators unfit for tasks requiring high degree of accuracy and/or high speed of operation.

In such situations recourse is taken to more accurate non-linear control techniques, some of which are discussed below.

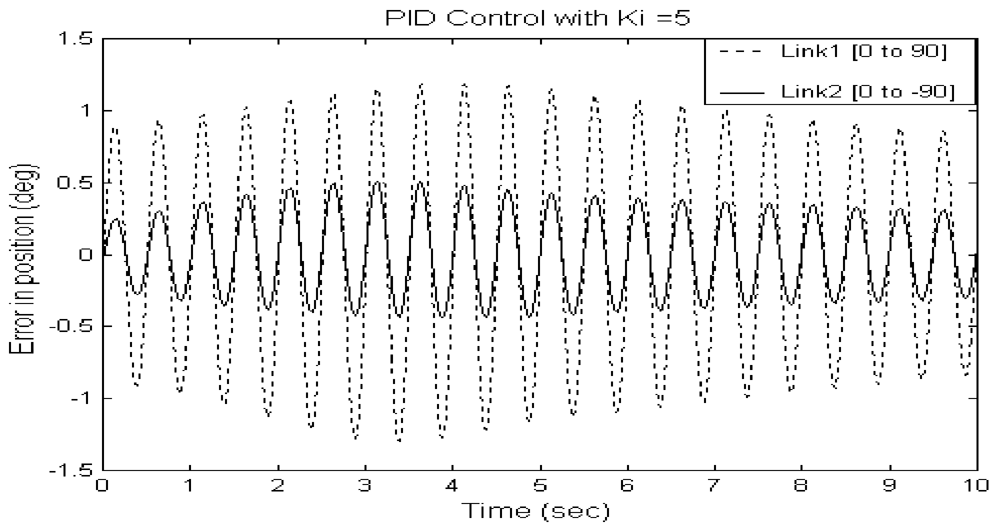


Fig. 4.1.4(c) Errors for PID Control with $K_I = 5$ (Trajectory 1)

4.2 NON-LINEAR CONTROL OF MANIPULATORS

The linear control of manipulators is non-model based in the sense that the control law does not take into consideration the robot mathematical model at all. In the non-linear control of manipulators the manipulator dynamics equation is taken in its complete form, usually without omitting or approximating any of the constituent matrices. The only approximation still used is that the links are still assumed to be perfectly rigid. The manipulator model may be in feedback or in forward path of the control loop. When used in forward path the aim is to provide a non-linear component torque in accordance with manipulator nonlinearities. On the other hand when used in feedback path the main aim is to cancel the manipulator dynamics nonlinearities and make the system linear and decoupled. Some common non-linear control techniques are now discussed.

4.2.1 COMPUTED TORQUE CONTROL (CT)

The most common control technique in the category of non-linear control is the Computed torque control proposed by Paul (1972). The block diagram representation of computed torque control strategy is shown below in Figure 4.2.1.1. As can be seen the basic idea of computed torque control is that of feedback linearization.

Here the computed torque τ_{ct} is given by

$$\tau_{ct} = M(\theta)\left[\ddot{\theta}_d + K_D\dot{e} + K_P e\right] + V_M\left(\theta,\dot{\theta}\right)\dot{\theta} + G(\theta) \tag{4.2.1.1}$$

If the manipulator model is known exactly then this scheme results in asymptotically stable, linear time invariant error dynamics and provides asymptotically exact tracking [Campa et al. (2001)].

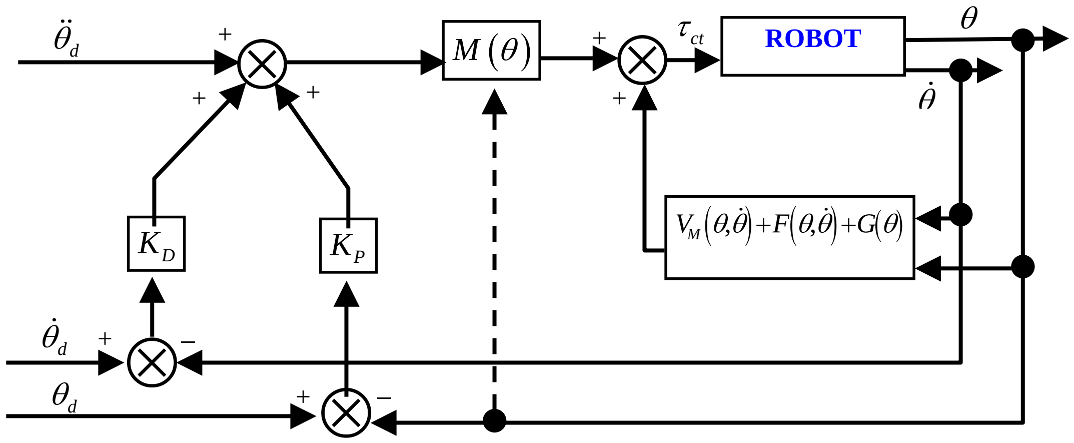


Fig. 4.2.1.1 Block diagram of Computed torque control

The simulation results for the Computed Torque Control method, with an exact model, i.e., assuming that the parameter values are exactly known, are shown in Fig. 4.2.1.2(a) and Fig 4.2.1.2(b). The RMS and Steady State values of errors are tabulated in Table 4.2.1.1(a). As can be seen, the performance of this controller is really good with both the steady state and the RMS values of the errors appreciably low without any danger of instability as in the PID controller. In fact the Steady state errors are almost zero.

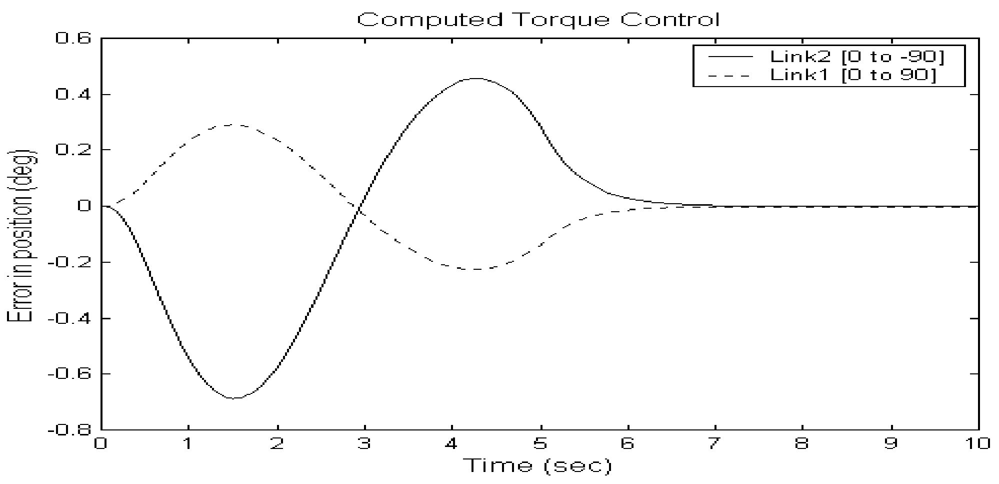


Fig. 4.2.1.2(a) Errors for Computed torque control (Trajectory 1, Exact model)

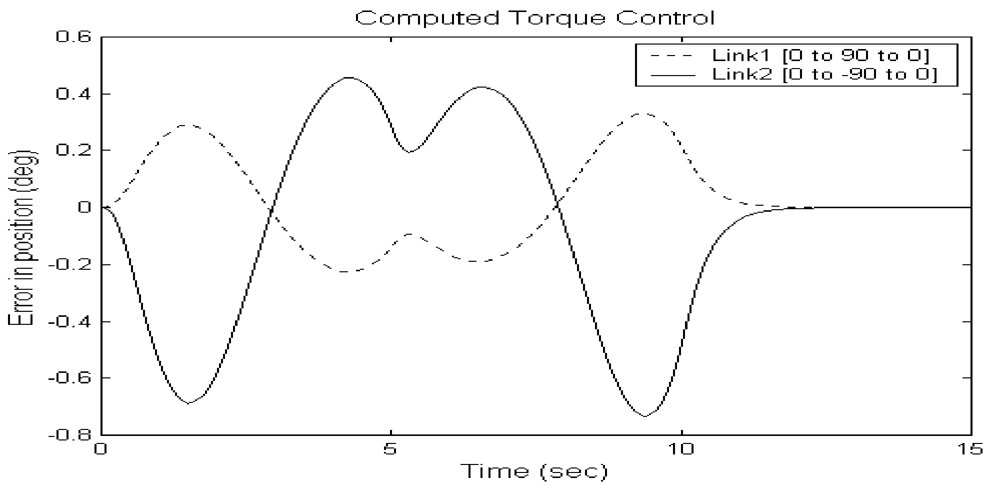


Fig. 4.2.1.2(b) Errors for Computed torque control (Trajectory2, Exact model)

Computed Torque Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°		**Link2 0°→ -90°**		**Link1 0°→ 90°→0°**		**Link2 0°→ -90°→0°**	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
0.1332	-0.0023	0.2990	0.0022	0.1579	0.0000	0.3513	0.0000

Table 4.2.1.1(a) Errors for Computed torque control with exact model

The effectiveness of this controller unfortunately lasts only till the model used is accurate. Even a slightly inexact model, if used, can lead to drastic degradation in performance. This can be seen from the Fig. 4.2.1.2(c) and Fig. 4.2.1.2(d), which depict the errors in position for a Computed Torque Controller with inexact model. The inexactness in modelling in this case was limited to just the masses of the two links, which were taken to be 1.8 kg each, instead of their exact value of 2 kg each. All other manipulator parameters, like the link lengths, the positions of the centre of masses of the links, etc., were taken accurately. But even this small inexactness in the value of just two parameters degrades the transient as well as steady state performance of the controller to a great degree. Table 4.2.1.1(b) tabulates the values of these errors.

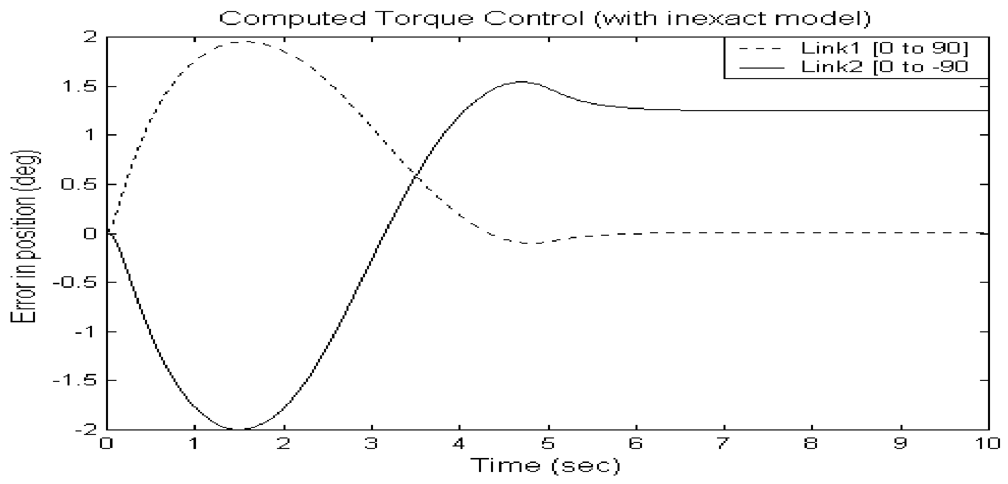


Fig. 4.2.1.2(c) Errors for Computed torque control (Trajectory1, Inexact model)

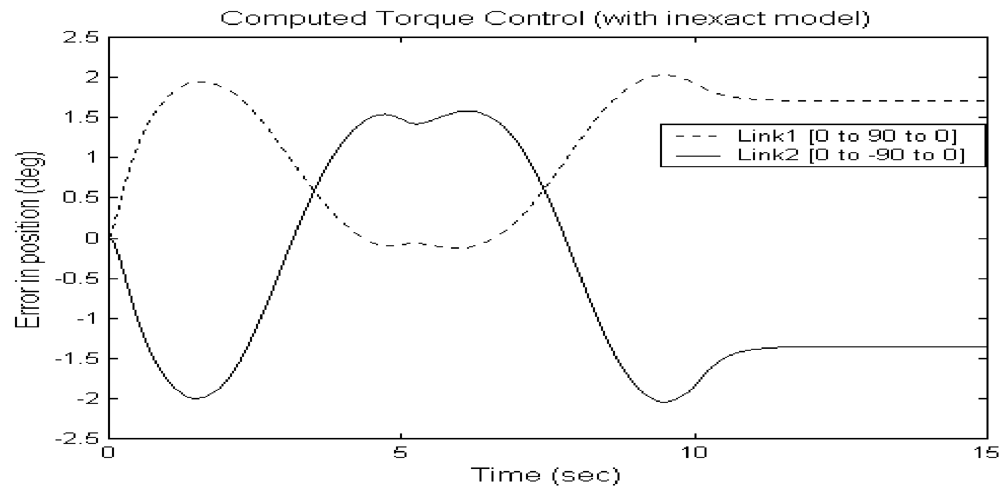


Fig. 4.2.1.2(d) Errors for Computed torque control (Trajectory2, Inexact model)

Computed Torque Control Errors (deg)							
TRAJECTORY NO.1				TRAJECTORY NO.2			
Link1 $0° \rightarrow 90°$		Link2 $0° \rightarrow -90°$		Link1 $0° \rightarrow 90° \rightarrow 0°$		Link2 $0° \rightarrow -90° \rightarrow 0°$	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
0.8839	0.0044	1.3060	1.2498	1.4132	1.7100	1.3908	-1.3497

Table 4.2.1.1(b) Errors for Computed torque control with inexact model

As the estimation of the parameters of the manipulator exactly is a difficult, if not an impossible task, and as the manipulator parameter change when it picks up a load, this controller clearly cannot be relied upon to give a good performance under practical circumstances.

4.2.2 FEED FORWARD INVERSE DYNAMICS (FFID) CONTROL

A slightly different approach that is more suitable to adaptation is sometimes used instead of computed torque scheme [Liegeois et al. (1980)]. This scheme uses the inverse dynamics in feed forward mode. The block diagram of this scheme is shown below in figure 4.2.2.1.

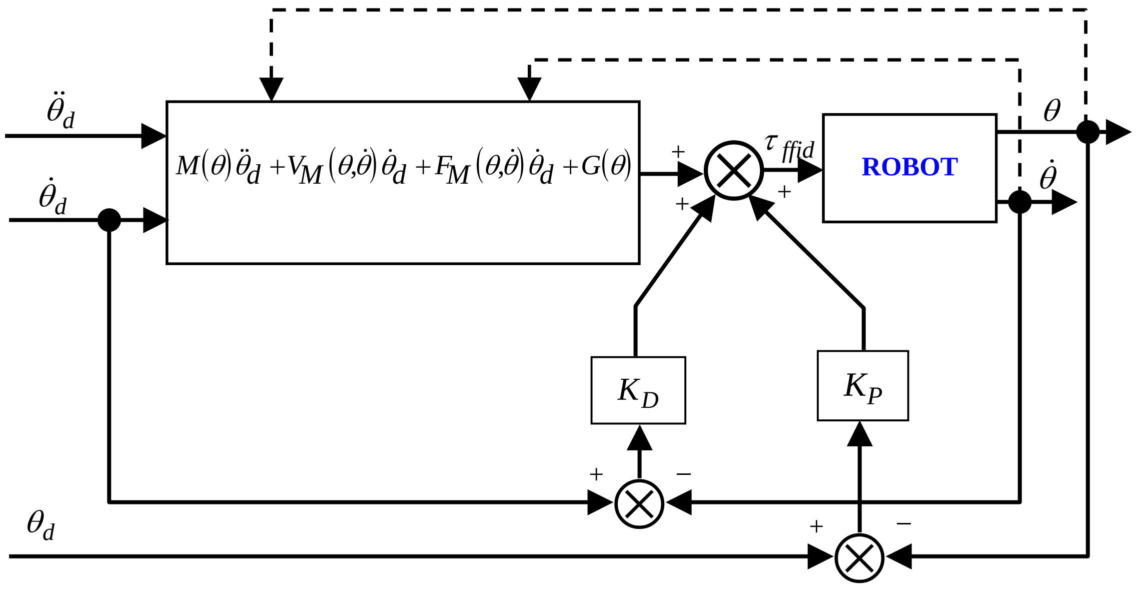

$\ddot{\theta}_d$
$\dot{\theta}_d$
$M(\theta)\ddot{\theta}_d + V_M(\theta,\dot{\theta})\dot{\theta}_d + F_M(\theta,\dot{\theta})\dot{\theta}_d + G(\theta)$
τ_{ffid}
ROBOT
θ
$\dot{\theta}$
K_D
K_P
θ_d
+
−

Fig. 4.2.2.1 Block diagram of Feed Forward Inverse Dynamics Control

In this strategy the torque is calculated as

$$\begin{aligned}\tau_{ffid} &= M(\theta)\ddot{\theta}_d + V_M(\theta,\dot{\theta})\dot{\theta}_d + F_M(\theta,\dot{\theta})\dot{\theta}_d + G(\theta) + K_D\dot{e} + K_P e \\ &= W(\theta,\dot{\theta},\dot{\theta}_d,\ddot{\theta}_d)P + K_D\dot{e} + K_P e\end{aligned} \qquad (4.2.2.1)$$

Equation 4.2.2.1 uses the inverse dynamics model with $W(\theta,\dot{\theta},\dot{\theta}_d,\ddot{\theta}_d)$ as the regressor matrix and P as the vector of manipulators parameters. The W and P matrices are as defined in equation 3.2.28. The regressor matrix is dependent both, on actual and the desired values of acceleration and velocity instead of the actual values alone as can be seen from equation 4.2.2.1. The error system resulting from this controller can be shown to be globally asymptotically stable when K_P and K_D are diagonal and all the scalar values are positive.

The simulation results for this controller, under the assumption that the manipulator model is known accurately, are shown in Fig. 4.2.2.2(a) and Fig. 4.2.2.2(b). The values of the RMS and the steady state errors are listed in Table 4.2.2.1(a).

It can be seen from the error profiles of Fig. 4.2.2.2(a) and (b) that the overall motion of the manipulator, with this controller, is smoother compared to that with the Computed Torque controller. This is indicated by a lesser number of sign changes in the error gradient. Also the magnitude of the RMS error for both the trajectories has decreased appreciably as compared to the Computed Torque controller.

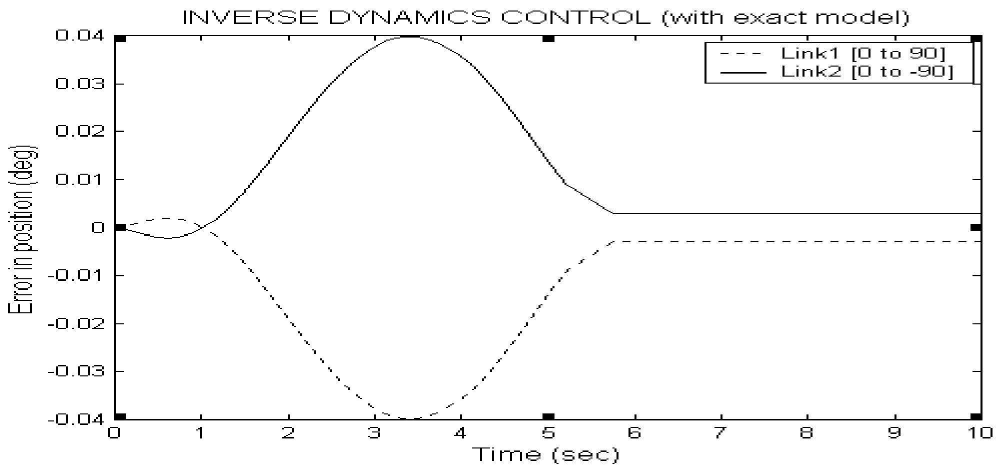


Fig. 4.2.2.2(a) Errors for FFID Control (Exact model, Trajectory1)

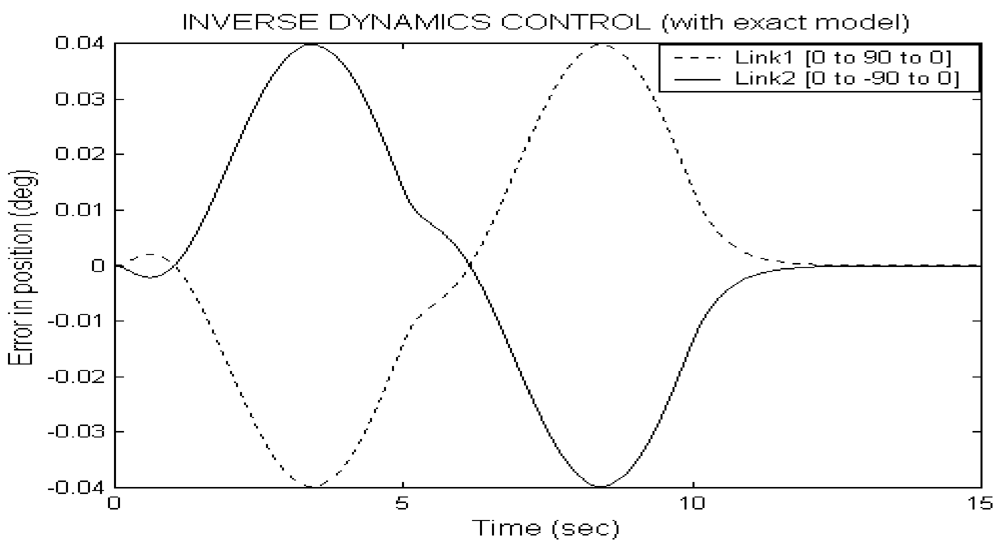


Fig. 4.2.2.2(b) Errors for FFID Control (Exact model, Trajectory2)

FFID Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°**		**Link2** **0°→ -90°**		**Link1** **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
0.0181	-0.0029	0.0181	0.0030	0.0208	0.0000	0.0208	0.0000

Table 4.2.2.1(a) Errors for FFID Control with exact model

The performance of the Feed Forward Inverse Dynamics controller was further tested by using an inexact model of the manipulator, for the feed forward torque calculations. The inexactness in modelling in this case, as for the previous Computed Torque controller, was once again limited to just the masses of the two links. The two links were taken to be of 1.8 kg each, instead of their exact value of 2 kg each. All other manipulator parameters, like the link lengths, the positions of the centre of masses of the links, etc., were taken accurately. The error profiles of the two joints for this case are shown in Fig. 4.2.2.2(c) and Fig. 4.2.2.2(d). The RMS and steady state values of the errors are listed in Table 4.2.2.1(b). It is observed that the performance of the controller shows degradation with larger errors for both the joints and for both the trajectories. However, the deterioration in the performance of the controller is not as marked and pronounced as it was for the Computed Torque scheme.

This clearly indicates the merits of using the inverse dynamics in the feed forward mode and further illustrates the advantage of using the desired velocity and acceleration instead of the actual ones for calculating the manipulator regressor matrix. The use of desired acceleration instead of the actual value has further advantage in terms of real implementation. Measuring the actual acceleration of the manipulator joints is a difficult task, as easy to use acceleration sensors are not readily available. Further, if the acceleration is found by differentiating the velocity information given by a tachogenerator or by double differentiating the position information given by an optical encoder, there always are possibilities of getting wrong values due to even a very low noise signal whose differentiation may result in very large and incorrect values of the actual acceleration.

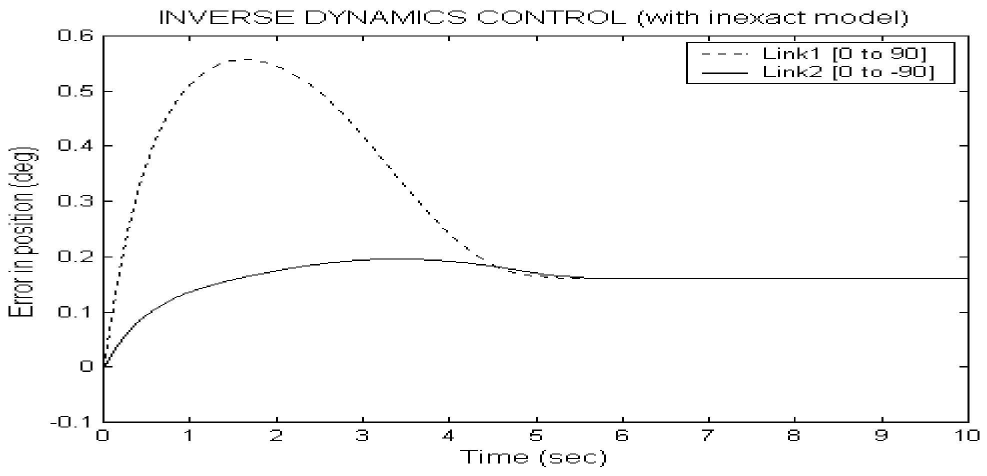


Fig. 4.2.2.2(c) Errors for FFID Control (Inexact model, Trajectory1)

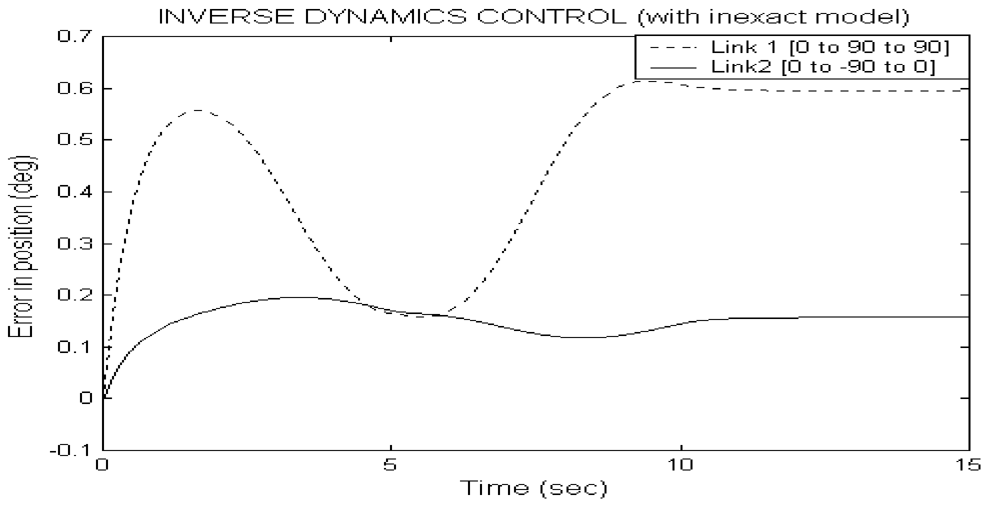


Fig. 4.2.2.2(d) Errors for FFID Control (Inexact model, Trajectory2)

FFID Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°**		**Link2** **0°→ -90°**		**Link1** **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
0.3083	0.1610	0.1638	0.1601	0.4849	0.5949	0.1544	0.1571

Table 4.2.2.1(b) Errors for FFID Control with inexact model

We next investigate another control strategy, which calculates the manipulator regressor matrix, W, in slightly different way, leading to further improvement in performance.

4.2.3 CRITICALLY DAMPED INVERSE DYNAMICS (CDID) CONTROL

This control strategy is almost same as the previous feedforward inverse dynamics, except that the regressor matrix, W, is calculated using reference velocity and reference acceleration instead of the desired values [Slotine and Li (1988)]. These reference values are defined as

$$\dot{\theta}_R = \dot{\theta}_d + \Lambda\left(\theta_d - \theta\right) \tag{4.2.3.1}$$

$$\ddot{\theta}_R = \ddot{\theta}_d + \Lambda\left(\dot{\theta}_d - \dot{\theta}\right) \tag{4.2.3.2}$$

The torque is calculated as

$$\tau_{cdid} = W\left(\theta, \dot{\theta}, \dot{\theta}_R, \ddot{\theta}_R\right) - K_D \dot{e}' \tag{4.2.3.3}$$

where the error , $\dot{e}'$, is defined as

$$\begin{aligned} \dot{e}' &= \dot{\theta} - \dot{\theta}_R \\ &= \dot{\theta} - \dot{\theta}_d - \Lambda\left(\theta_d - \theta\right) \end{aligned} \tag{4.2.3.4}$$

This control law results in a system of stable first order subspace. An exponentially stable system forced by an input that decays to zero has an output that decays to zero. Then $\lim_{t\to\infty} e\left(t\right) \to 0$. This result is used to prove the stability of the controller [Sadegh and Horowitz (1987)]. The block diagram of this control scheme is given in Fig. 4.2.3.1.

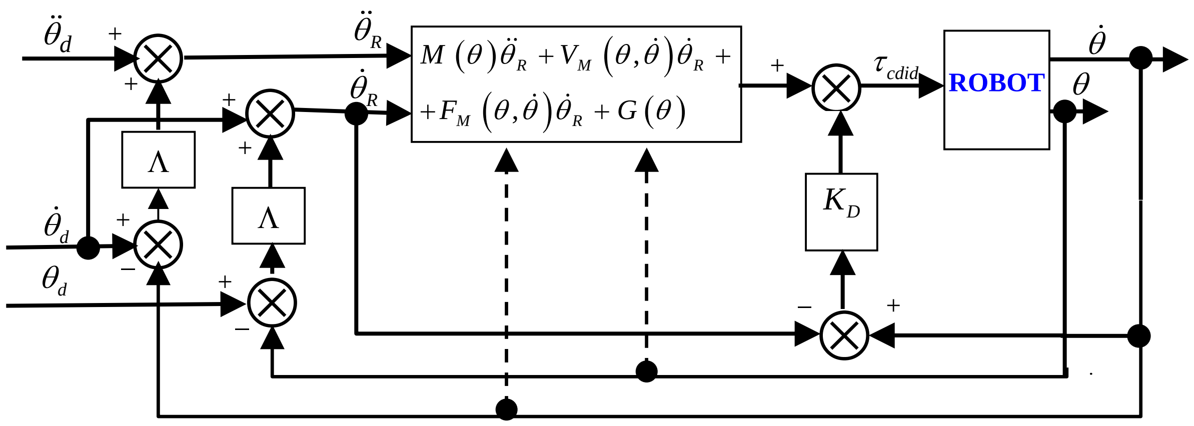


Fig.4.2.3.1 Block diagram of Critically damped inverse dynamics control

The CDID controller was also tested for performance, using both the exact and inexact models of the manipulator as for the previous controllers. As can be seen from equations 4.2.3.3 and 4.2.3.4 there are two main differences between this controller and the previous FFID controller. First, in this controller, the manipulator regressor matrix is calculated as a function of actual positions and velocities and also as a function of reference velocities and accelerations, while in the FFID controller the regressor matrix was calculated as a function of actual positions and velocities, and desired velocities and accelerations. Second, the effective proportional gain of the CDID controller is increased in comparison to that of FFID controller. To see the effective increase in gain, we know that the FFID control law from equation 4.2.2.1 is

$$\begin{aligned}\tau_{ffid} &= M(\theta)\ddot{\theta}_d + V_M(\theta,\dot{\theta})\dot{\theta}_d + F_M(\theta,\dot{\theta})\dot{\theta}_d + G(\theta) + K_D\dot{e} + K_P e \\ &= W(\theta,\dot{\theta},\dot{\theta}_d,\ddot{\theta}_d)P + K_D\dot{e} + K_P e\end{aligned} \tag{4.2.3.5}$$

and from equation 4.2.3.3, the control law for the CDID controller can be written as

$$\begin{aligned}\tau_{cdid} &= W(\theta,\dot{\theta},\dot{\theta}_R,\ddot{\theta}_R)P + K_D(\dot{\theta}_R - \dot{\theta}) \\ &= M(\theta)\ddot{\theta}_R + V_M(\theta,\dot{\theta})\dot{\theta}_R + F_M(\theta,\dot{\theta})\dot{\theta}_R + G(\theta) + K_D(\dot{\theta}_R - \dot{\theta})\end{aligned} \tag{4.2.3.6}$$

Substituting equation 4.2.3.1 and 4.2.3.2 in equation 4.2.3.6 we get,

$$\begin{aligned}\tau_{cdid} &= M(\theta)\ddot{\theta}_R + V_M(\theta,\dot{\theta})\dot{\theta}_R + F_M(\theta,\dot{\theta})\dot{\theta}_R + G(\theta) + K_D\dot{e} + K_D\Lambda e \\ &= M(\theta)\ddot{\theta}_d + V_M(\theta,\dot{\theta})\dot{\theta}_d + F_M(\theta,\dot{\theta})\dot{\theta}_d + G(\theta) + M(\theta)\Lambda\dot{e} + \\ &\quad + V_M(\theta,\dot{\theta})\Lambda e + K_D\dot{e} + K_D\Lambda e \\ &= W(\theta,\dot{\theta},\dot{\theta}_d,\ddot{\theta}_d)P + \left[M(\theta)\Lambda + K_D\right]\dot{e} + \left[V_M(\theta,\dot{\theta})\Lambda + K_D\Lambda\right]e\end{aligned} \tag{4.2.3.7}$$

Comparing equations 4.2.3.5 and 4.2.3.7 we see the equivalent controller gains as

$$\begin{aligned} K_{Deq} &\equiv M(\theta)\Lambda + K_D \\ K_{Peq} &\equiv V_M(\theta,\dot{\theta})\Lambda + K_D\Lambda \end{aligned} \tag{4.2.3.8}$$

As the manipulator mass matrix $M(\theta)$ is positive definite and if the Λ matrix is also chosen to be positive definite, then their product $M(\theta)\Lambda$ is also positive definite. Thus the resultant derivative controller gain for CDID controller is increased compared to the derivative gain for FFID controller. The Λ matrix of the CDID controller effectively has the same role as the K_P matrix has in the FFID controller. The matrix product $V_M(\theta,\dot{\theta})\Lambda$ in equation 4.2.3.8 can have both positive and negative valued elements, but usually the magnitudes of these elements are small. As a result, if we choose $\Lambda = K_P$, the effective proportional gain constant of the CDID controller increases approximately by a factor of K_D.

The simulation results for the CDID controller with exact manipulator model are shown in Fig. 4.2.3.2(a) and (b). The RMS and the Steady State values of the errors for the two links for two different trajectories are listed in Table 4.2.3.1(a).

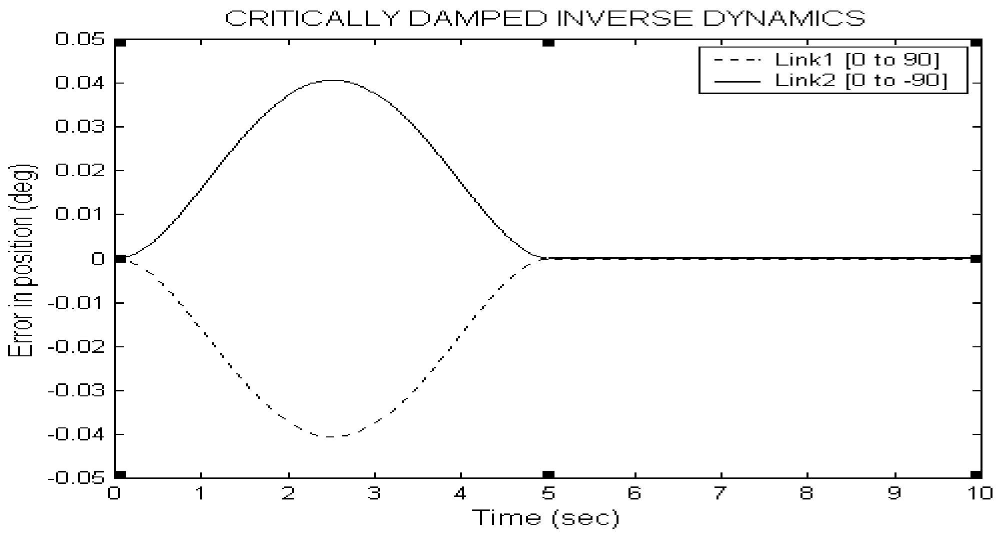


Fig. 4.2.3.2(a) Errors for CDID Control (Exact model, Trajectory1)

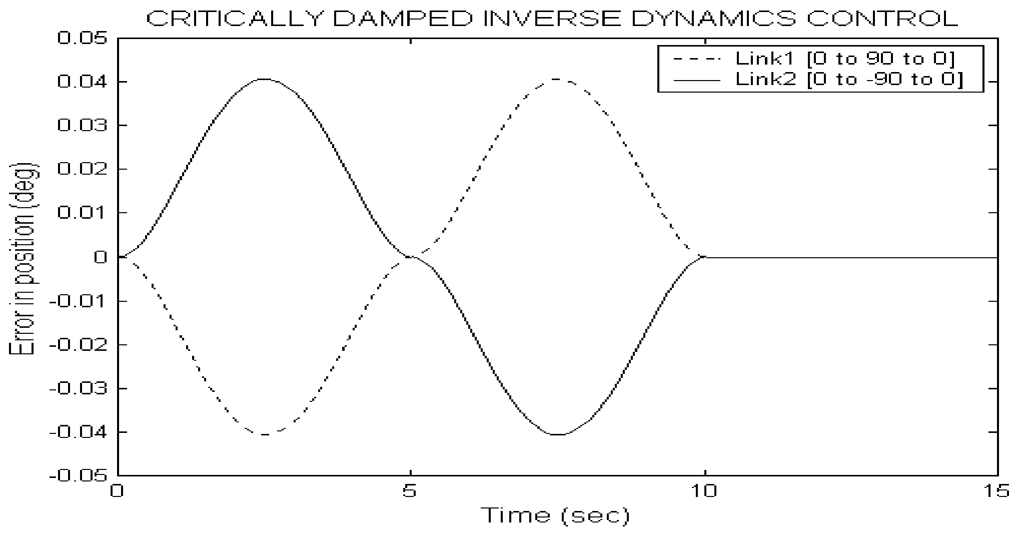


Fig. 4.2.3.2(b) Errors for CDID Control (Exact model, Trajectory2)

CDID Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°		**Link2 0°→ -90°**		**Link1 0°→ 90°→0°**		**Link2 0°→ -90°→0°**	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
0.0183	-0.0001	0.0183	**0.0000**	0.0211	**0.0000**	0.0212	**0.0000**

Table 4.2.3.1(a) Errors for CDID Control with exact model

As can be expected and as is also seen in Fig. 4.2.3.2(a) and (b), with the model known exactly, the CDID controller does not give any performance improvement over the FFID controller as far as the RMS value of the transient portion of motion is concerned. However, as the effective proportional gain constant for the CDID controller is larger than that for the FFID controller, it results in a better steady state performance. In fact, the steady state errors in this simulation are seen to have been completely removed.

The strength of the CDID controller comes to the fore when the manipulator model is not known exactly. The error profiles for the simulation with inexact model are shown in Fig. 4.2.3.2(c) and (d). Table 4.2.3.1(b) lists the quantitative values of the errors for different cases. As can be seen, both the RMS and steady state values of errors, for both the links for both the trajectories are considerably reduced in magnitude over the corresponding values for FFID. As explained earlier, the greater value of the proportional gain constant results in a better steady state performance, while, the use of reference velocities and accelerations for calculation of the manipulator regressor matrix instead of the actual values, results in an improved transient performance of the arm. This can be mainly attributed to the fact that the reference values are ‘cleaner’ compared to the actual values, which are sensor derived and hence always ridden with noise. Moreover if only an optical encoder is used for feedback (as is usual), the values of velocity and acceleration of the joints has to be derived by differentiating the position information provided by the encoders. This numerical differentiation can further reduce the validity of data and the problem becomes more severe with increase in the noise in the environment where the manipulator is working.

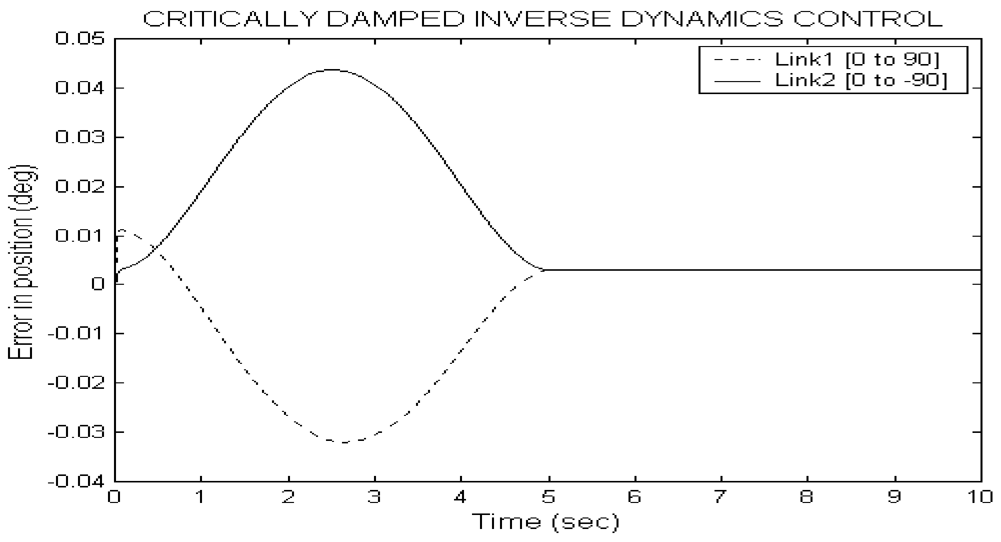


Fig. 4.2.3.2(c) Errors for CDID Control (Inexact model, Trajectory1)

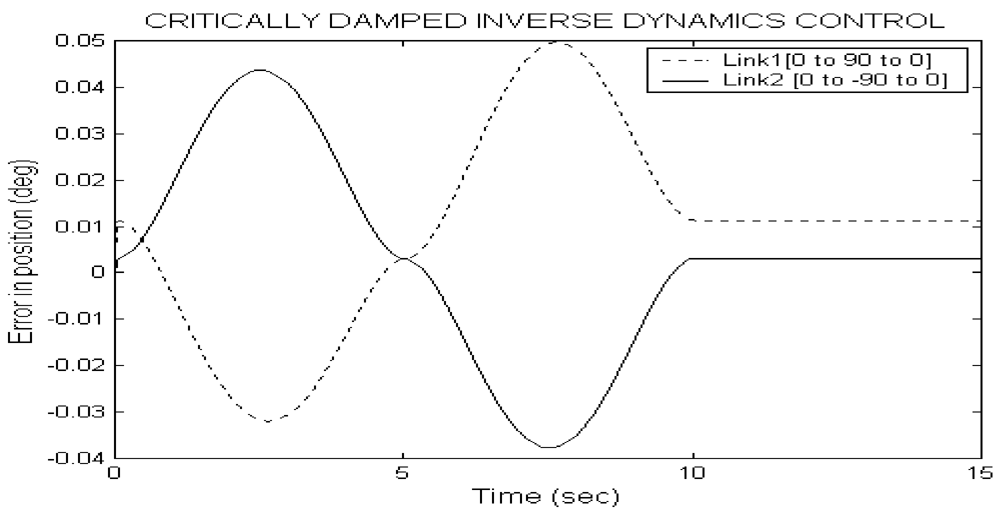


Fig. 4.2.3.2(d) Errors for CDID Control (Inexact model, Trajectory2)

CDID Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°**		**Link2** **0°→ -90°**		**Link1** **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**	
RMS	S.S	RMS	S.S	RMS	S.S	RMS	S.S
0.0140	0.0029	0.0203	0.0030	0.0232	0.0113	0.0214	0.0030

Table 4.2.3.1(b) Errors for CDID Control with inexact model

4.2.4 PURE FEEDFORWARD CONTROLLER

As the name suggests, the inverse dynamics in this scheme is calculated using only the desired values of trajectory. The torque here is calculated as

$$\tau_{idp} = W\left(\theta_d, \dot{\theta}_d, \dot{\theta}_d, \ddot{\theta}_d\right) P + \tau_{PD} + \sigma_n \|e\|^2 K_2^{-1} Ke \tag{4.2.4.1}$$

The last term in the above equation is added to guarantee the stability of the system [Wen and Bayard (1987)].

The simulation results of this controller showed that in performance this controller is essentially at par with the Critically Damped Inverse Dynamics Controller. The error profiles and magnitudes for both the cases of exact and inexact models of the manipulator matched to a good degree to those of CDID controller. Hence the results for this controller are not shown separately.

4.2.5 COMMENTS ON PERFORMANCE OF VARIOUS MODEL BASED CONTROLLERS

The consolidated results for the simulations are presented in Table 4.2.5.1 (Exact model) and Table 4.2.5.2 (Inexact model) for easy comparison of the controllers' performances.

S.No	**STRATEGY**	**TRAJECTORY NO.1**				**TRAJECTORY NO.2**			
		link1 $0° \rightarrow 90°$		**link2** $0° \rightarrow -90°$		**link1** $0° \rightarrow 90° \rightarrow 0°$		**link2** $0° \rightarrow -90° \rightarrow 0°$	
		RMS	**SS**	**RMS**	**SS**	**RMS**	**SS**	**RMS**	**SS**
1.	**PD Control**	3.6101	1.6990	1.3115	1.5660	4.8932	5.9127	1.5740	1.5583
2.	**PID control**	0.7248	0.0	0.1944	0.0	0.6195	-0.0001	0.1682	-0.0003
3.	**CT**	0.1332	-0.0023	0.2990	0.0022	0.1579	0.0	0.3513	0.0
4.	**FFID**	0.0181	-0.0029	0.0181	0.0030	0.0208	0.0	0.0208	0.0
5.	**CDID**	0.0183	-0.0001	0.0183	0.0	0.0211	0.0	0.0212	0.0

Table 4.2.5.1. Errors for various controllers (Exact model)

S.No	**STRATEGY**	**TRAJECTORY NO.1**				**TRAJECTORY NO.2**			
		link1 $0° \rightarrow 90°$		**link2** $0° \rightarrow -90°$		**link1** $0° \rightarrow 90° \rightarrow 0°$		**link2** $0° \rightarrow -90° \rightarrow 0°$	
		RMS	**SS**	**RMS**	**SS**	**RMS**	**SS**	**RMS**	**SS**
1.	**CT**	0.8839	0.0044	1.3060	1.2498	1.4132	1.7100	1.3908	-1.3497
2.	**FFID**	0.3083	0.1610	0.1638	0.1601	0.4849	0.5949	0.1544	0.1571
3.	**CDID**	0.0140	0.0029	0.0203	0.0030	0.0232	0.0113	0.0214	0.0030

Table 4.2.5.2. Errors for various controllers (Inexact model)

Following are the observations made based on the simulation studies done in sections 4.1 and 4.2.

1. PD controller does not give acceptable performance in case the cross coupling effects of manipulator dynamics are not negligible. The errors can be reduced by increasing the controller gains but at the risk of exciting unmodelled dynamics. Anyway, the steady state errors cannot be totally eliminated by even increasing the controller gains.
2. PID controller gives a very good performance but has the risk of instability. Hence it is generally not used in its pure form for manipulator control.

3. If the manipulator model and parameters are known exactly, then the FFID and CDID controllers give comparable performance.
4. If the manipulator parameters are not known exactly, then the CDID outperforms FFID controller.
5. Model based controllers perform better if the values used in the model are of reference or desired trajectory input rather than the sensor values. (Ex. CDID or FFID vs. CT)
6. The model-based controllers are sensitive to incorrect parameter values, and sometimes their performance may degrade below PD performance level if inexactness in parameter values is too high.
7. For CDID the effective controller gain is increased and hence it gives lower error than FFID.
8. The performance of these model-based controllers depends to a great extent on the accuracy of modelling.

The flowchart for the algorithm used for software simulations done in section 4.1 and 4.2 is shown in Fig. 4.2.5.1. The variable ‘K’ keeps track of the iteration number for a given set point. When K is equal to four, a new set point is calculated.
In the next section we investigate the performance of various controllers for the case when manipulator picks up a load sometime during its motion. This changes the values of parameters during motion and demands more from the controllers to maintain their performance.

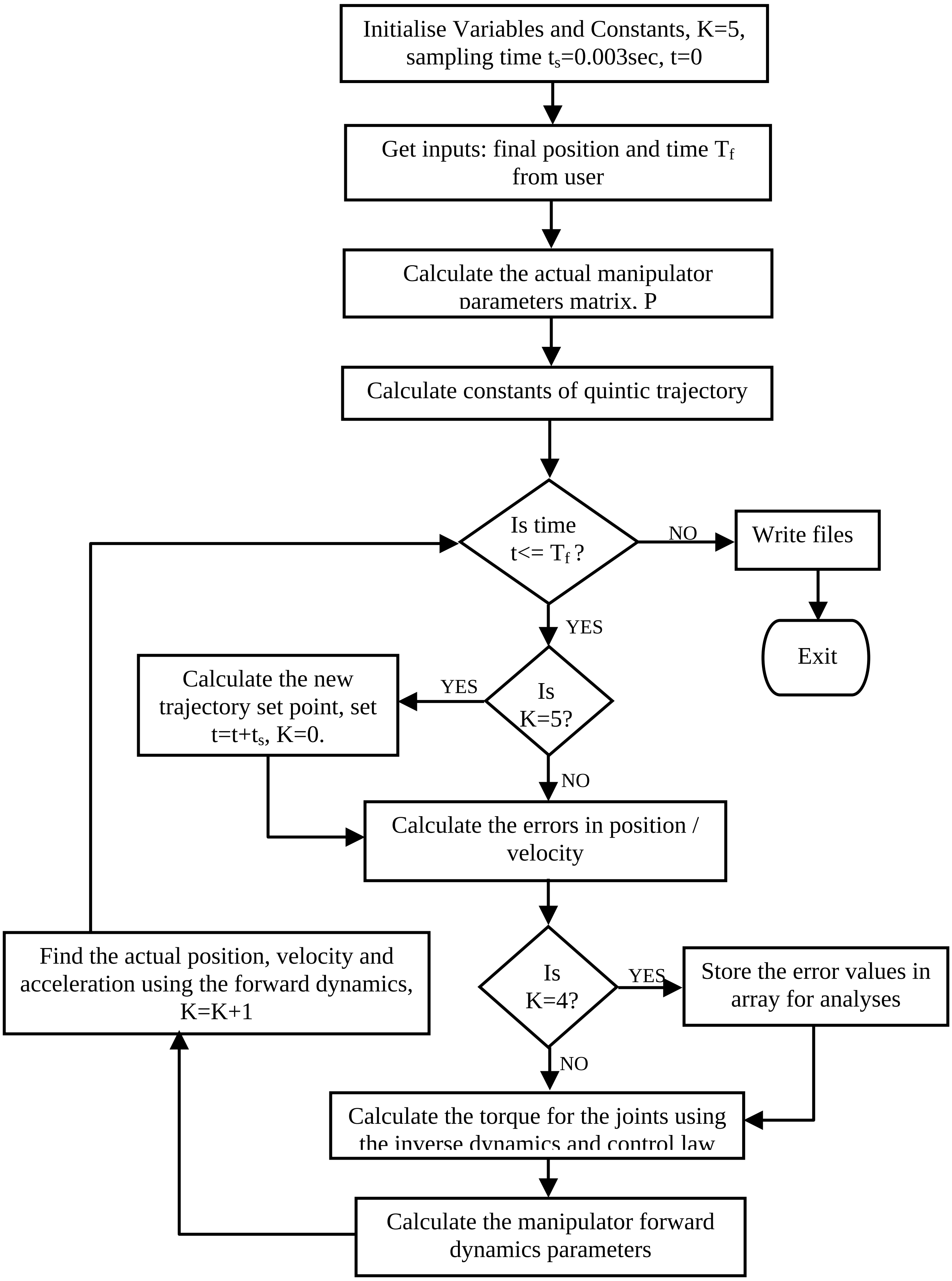

Initialise Variables and Constants, K=5, sampling time t_s=0.003sec, t=0
Get inputs: final position and time T_f from user
Calculate the actual manipulator parameters matrix, P
Calculate constants of quintic trajectory
Is time t<= T_f ?
NO
Write files
Exit
YES
Is K=5?
YES
Calculate the new trajectory set point, set t=t+t_s, K=0.
NO
Calculate the errors in position / velocity
Is K=4?
YES
Store the error values in array for analyses
NO
Find the actual position, velocity and acceleration using the forward dynamics, K=K+1
Calculate the torque for the joints using the inverse dynamics and control law
Calculate the manipulator forward dynamics parameters

Fig.4.2.5.1. Software Simulation Algorithm Flowchart

4.3. EFFECT OF PARAMETER VARIATION ON CONTROLLER PERFORMANCE

A great cause of stress on the controller is the change in the values of manipulator parameters when the arm picks up a load and moves it in its workspace to place it at some destination point. We have already said that the determination of manipulator parameters exactly, is a very difficult task. At most we may have only a *good estimate* of these parameter values but only rarely their exact values. Now as the manipulator may pick up different loads during the course of its operation, and these loads may not be known in the most general case, the model based controllers discussed previously will certainly result in a degraded performance. These controllers need information about the parameter values to control the manipulator motion effectively. Any deviation from the values used in calculating the control law will lead to a poor performance of these controllers.

4.3.1 TEST TRAJECTORIES AND PARAMETER VALUES

The various control strategies discussed previously in sections 4.1 and 4.2 were tested for performance against two trajectories. The first trajectory consisted of only a single quintic polynomial, which moved the two joints from their initial positions (0°) to a final position of 90° or -90°. The second trajectory on the other hand consisted of two quintic polynomials. The first polynomial takes the joints from their initial home position (0°) to either 90° or –90°, while the second polynomial moves the joints back to their home positions. This ***trajectory switching*** is a typical occurrence during the course of manipulator motion and puts a great stress on the controller.

In the following sections we further tested the performance of these control strategies by incorporating the fact that the manipulator picks up a load during its motion. This results in changed values of its parameters. We assume that in the beginning of manipulator motion the parameter values were known exactly, and at some time during its motion, the manipulator picks up a load thereby changing its parameter values. The controller performance under this situation was tested by simulation.

We once again used two trajectories for testing the control strategies. In the first case, the joint 1 of the manipulator moves from 0° to 90° in 5 seconds. At this time the manipulator picks up a load. The joint then moves back to 0° in 5 seconds and then stays there for another 5 seconds. The motion of joint 2 is exactly same as of joint 1 except that it moves to –90° in place of +90°. The interpolating polynomials used were quintic. This first trajectory is shown in Fig. 4.3.1.1(a). For this trajectory the quantitative measures of performance were taken as the RMS and the steady state values of the errors.

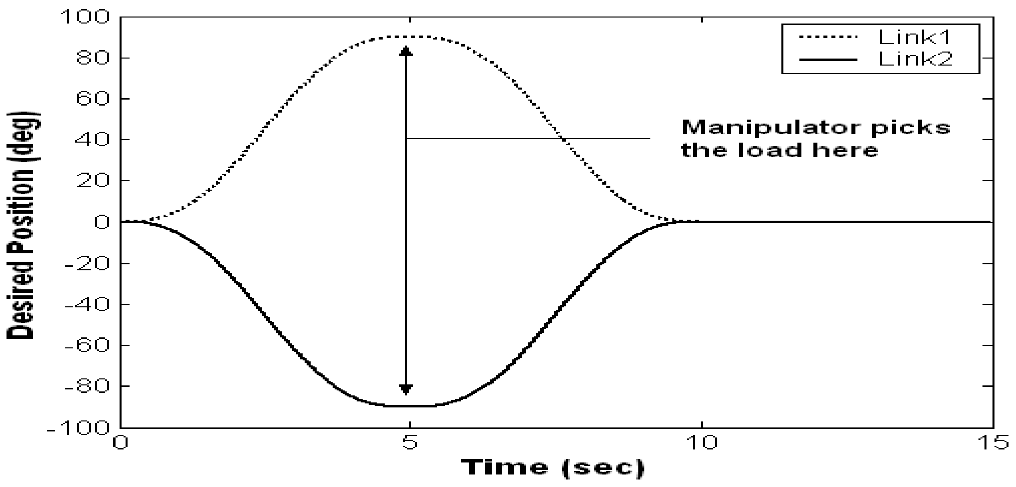


Fig. 4.3.1.1(a) Desired Trajectory 1 (Changing Parameters)

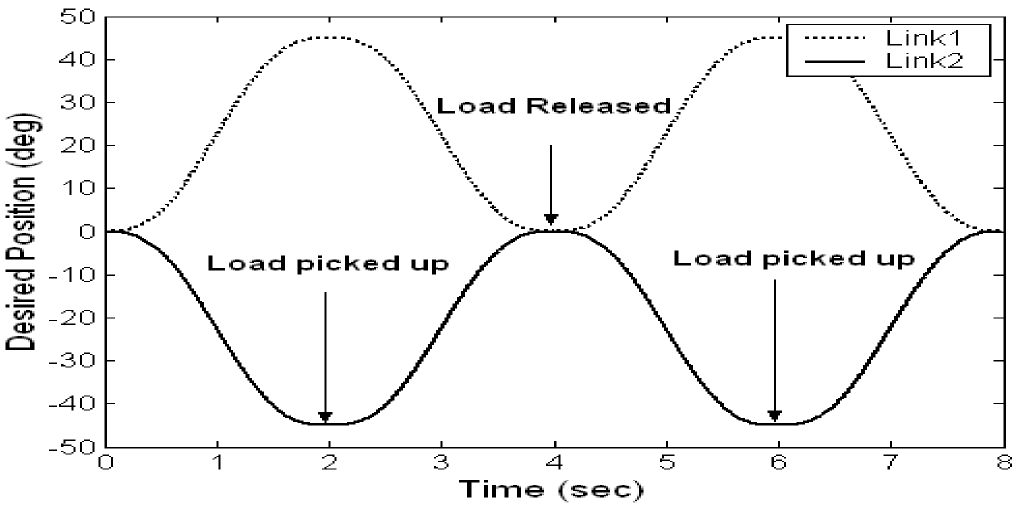


Fig. 4.3.1.1(b) Desired Trajectory 2 (Changing Parameters)

The second trajectory used for testing is shown in Fig. 4.3.1.1(b). In this trajectory the two joints of the manipulator were required to move in a cyclic fashion. The first joint moves from its home position of 0° to +45° in 2 seconds. The manipulator then picks up a payload and returns home in next 2 seconds and upon reaching home it drops its payload. This operation is then repeated over time. The second joint has a trajectory profile similar to the first one except that it moves from home to –45° and back to home. The motion of manipulator over a period of 8 seconds was used for finding the RMS values of the errors. Since in this case the set point is always changing, the measure of steady state error was replaced by the maximum error over this time period. This kind of motion is commonly found in industrial manipulators used for 'Pick and Place' kind of operation. The original values of various manipulator parameters were taken to be same as in the case of previous simulations. When the manipulator picks up a load, the values of these parameters undergo a change. For the task of simulation, the original and the changed values of the parameters were taken as shown in Tables 4.3.1.1 and 4.3.1.2. The values in Table 4.3.1.1 are based on Link 2 and Link 3 parameters of the CRS Plus manipulator.

$m_1 = 2.0\ kg$
$m_2 = 2.0\ kg$
$l_1 = 0.26\ m$
$x_1 = 0.13\ m$
$x_2 = 0.14\ meters$
$I_{zz1} = 0.09\ kg - m^2$
$I_{zz2} = 0.09\ kg - m^2$
$F_1 = 2.5\ N - m / rad / \sec$
$F_2 = 2.5\ N - m / rad / \sec$

Table 4.3.1.1. Original manipulator parameter values

$m_1 = 3.0\ kg$
$m_2 = 3.0\ kg$
$l_1 = 0.26\ m$
$x_1 = 0.15\ m$
$x_2 = 0.16\ meters$
$I_{zz1} = 1.5\ kg - m^2$
$I_{zz2} = 0.09\ kg - m^2$
$F_1 = 2.5\ N - m / rad / \sec$
$F_2 = 2.5\ N - m / rad / \sec$

Table 4.3.1.2. Changed manipulator parameter values (on picking up load)

4.3.2 PD / PID CONTROLLERS

The PD and the PID control algorithms are non-model based control strategies. These two controllers were tested for the effect of parameter variation on their performance to form a basis for comparison of performance for model-based algorithms.

The PD controller error profiles are shown in Fig. 4.3.2.1(a) and (b) for trajectory 1 and trajectory 2 respectively. The magnitudes of various error norms are tabulated in Table. 4.3.2.1. As expected, the effect of parameter variation on the performance of PD controller is drastic. Both the RMS and the Steady State values of errors for both the trajectories show a marked increase. It can be seen from Fig.4.3.2.1 (b) that the errors increase every time the manipulator picks up the load and they decrease when the load is released. Overall the errors are large and unacceptable.

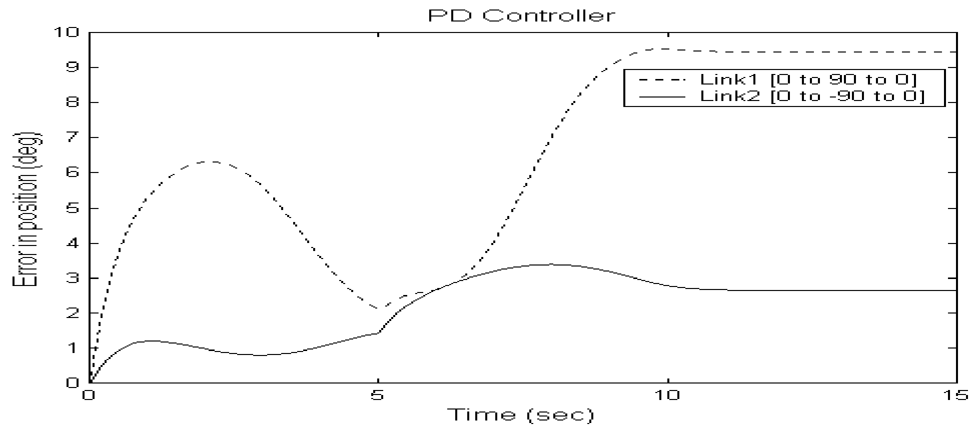


Fig. 4.3.2.1(a) PD Control errors for Trajectory 1 (Changing Parameters)

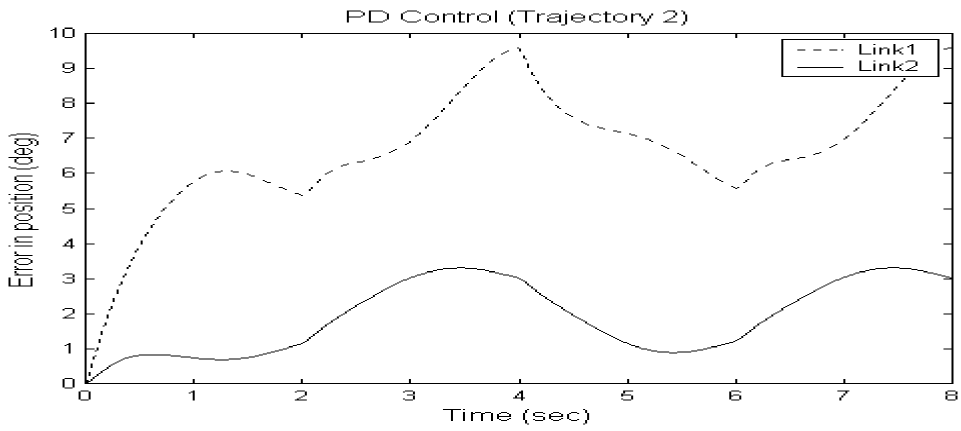


Fig. 4.3.2.1(b) PD Control errors for Trajectory 2 (Changing Parameters)

PD Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
7.1415	9.4447	2.3758	2.6345	6.8624	9.5918	2.1554	3.3037

Table 4.3.2.1 Errors for PD control (Changing Parameters)

Adding an integral term and modifying the controller to PID substantially reduces the large errors of PD controller. The simulation results for the PID controller are shown in Fig. 4.3.2.2(a) and (b). The magnitudes of the errors are listed in Table. 4.3.2.2. The integral gain constant K_I for this controller is taken to be a small value equal to 0.25. K_P and K_D were taken as 100 and 50 respectively as for previous simulations. It can be seen that even this small value of K_I results in a marked improvement in performance. The RMS values of the errors are substantially reduced and the steady state errors are eliminated almost totally.

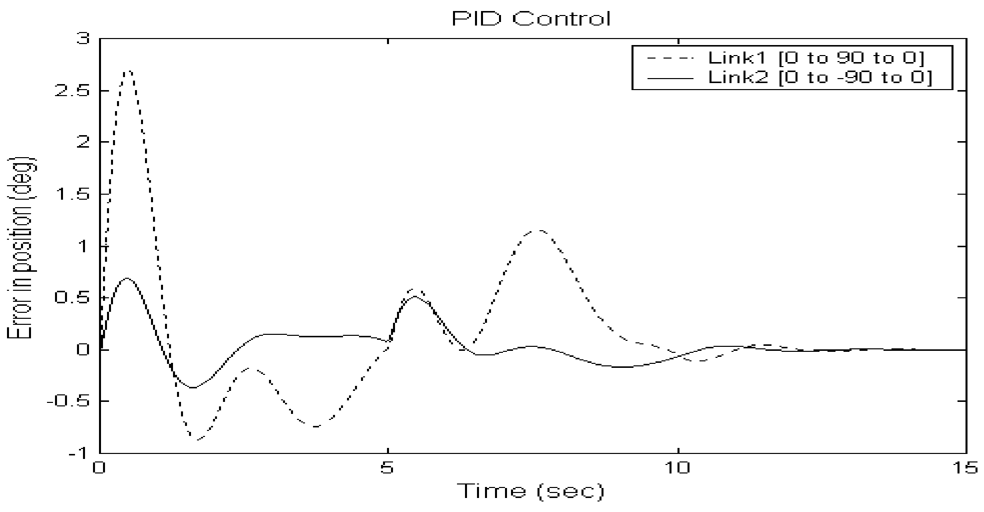


Fig. 4.3.2.2(a) PID Control errors for Trajectory 1 (Changing Parameters)

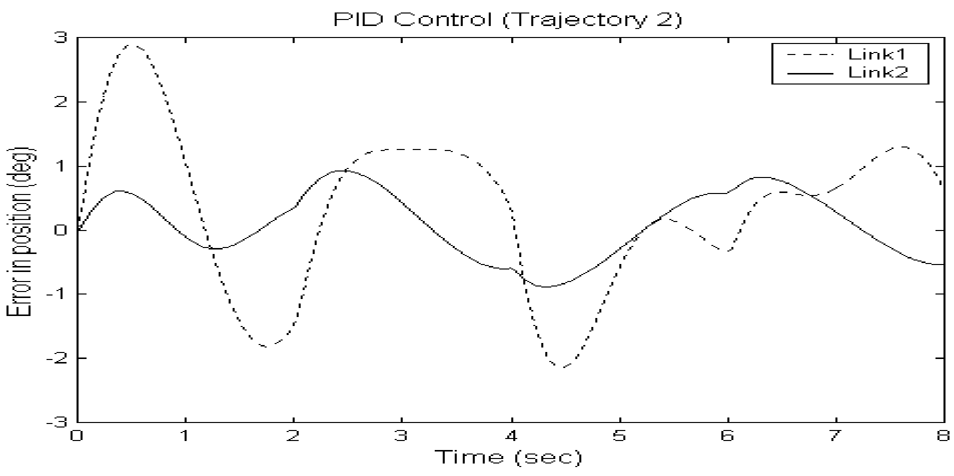


Fig. 4.3.2.2(b) PID Control errors for Trajectory 2 (Changing Parameters)

PID Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.6783	-0.0024	0.1960	-0.0004	1.2578	2.8973	0.5218	0.9304

Table 4.3.2.2 Errors for PID control (Changing Parameters)

But unfortunately this performance improvement is not without peril. Even a small increase in the integral gain constant results in an unstable system. Fig. 4.3.2.2(c) is the plot of PID controller errors versus time for a value of K_I =3.25. It can be seen that the system has become unstable with the errors increasing with time. This instability of PID controllers makes them unfit for actual use for controlling the manipulator.

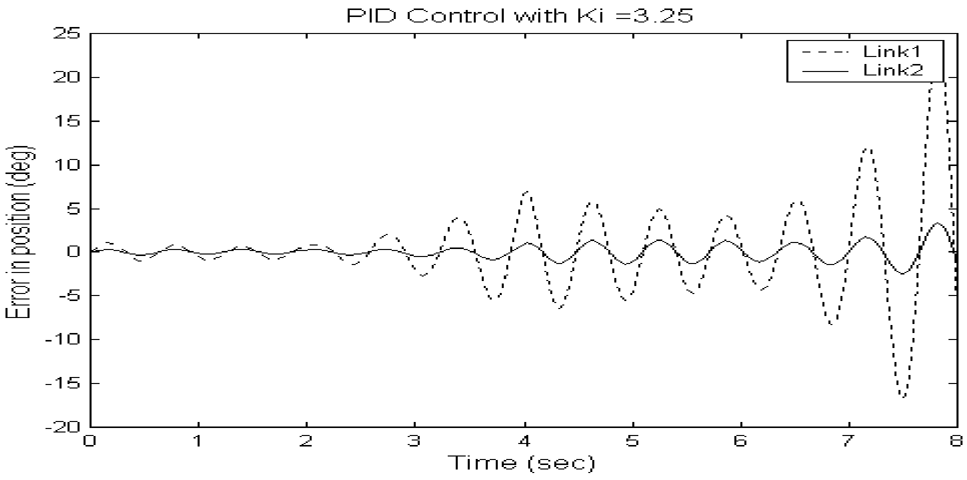


Fig. 4.3.2.2(c) PID Control errors for Trajectory 2 with K_I=3.25 (Changing Parameters)

The above PID controller can be modified in the way errors are summed up for the integral action. Instead of summing the errors during the entire duration of trajectory, the errors are summed up only for the five iterations of the control loop. These five iterations are associated with every new set point supplied by the trajectory generator. Whenever the new set point arrives from trajectory generator, the error summation is reset to zero. This change in the way errors are summed up, result in great advantage from stability point of view for the controller. It can be shown that this controller is stable [Loria (2000)].

The position error profiles for this modified PID controller are shown in Fig. 4.3.2.2(d) and 4.3.2.2(e) for trajectory 1 and trajectory 2 respectively. Table 4.3.2.3 lists the values of various errors for the two trajectories. It can be seen that the error magnitudes have gone down appreciably for this modified PID controller when compared to the PD controller. Of course the performance improvement is not as marked as in case of ‘normal’ PID, but is still good enough to warrant its use. In fact the errors have gone down by more than 50% compared to PD control errors. Moreover the fact that this scheme can be proved to be stable makes it all the more appealing.

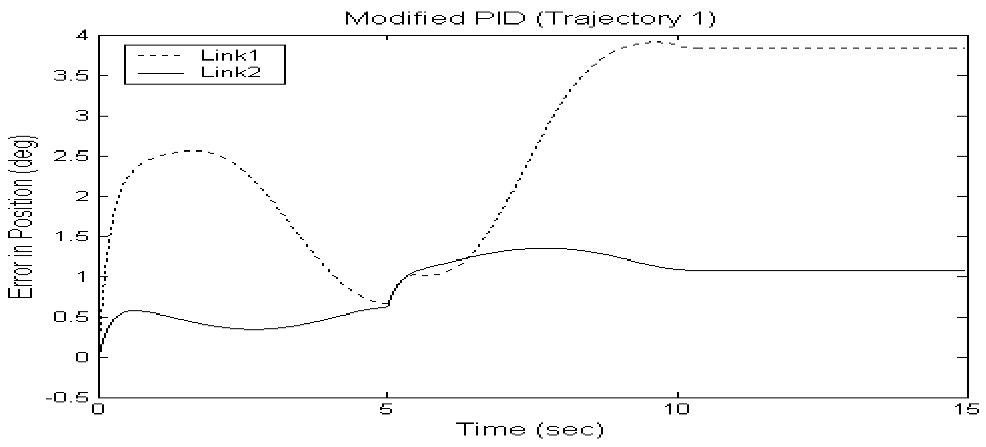


Fig. 4.3.2.2(d) Modified PID Control errors for Trajectory 1 (Changing Parameters)

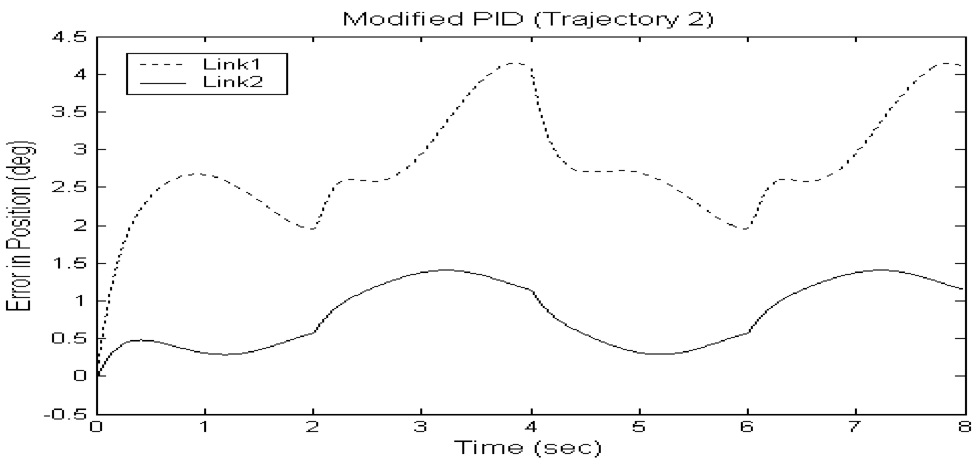


Fig. 4.3.2.2(e) Modified PID Control errors for Trajectory 2 (Changing Parameters)

Modified PID Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
2.9360	3.8290	0.9742	1.0737	2.8617	4.1457	0.9264	1.4036

Table 4.3.2.3 Errors for Modified PID control (Changing Parameters)

4.3.3. COMPUTED TORQUE CONTROL

The computed torque control is a model based control strategy, in which the model of the manipulator is in the feedback loop. We have already seen that the performance of this controller depends largely on the exactness of the model being used. For this simulation we assumed that to begin with the model is exactly known and that this model changes when the manipulator picks up a load. The errors versus time plot for this controller is shown in Fig 4.3.3.1(a) and (b) for the two trajectories. Table 4.3.3.1 lists the magnitude of errors for the two trajectories used for the simulated testing.

It can be seen from the error plots that in the beginning, when the model is known exactly, the tracking errors of the two joints are small. These errors show a sharp increase in magnitude when the manipulator picks up a load, when time is 5 seconds for the first trajectory and 2 seconds for the second trajectory. It can be further seen that the transient errors, which build up, tend to increase the steady state error as well. Overall the errors are large and the motion is jerky with the errors increasing and decreasing alternately. Computed torque control as a result performs poorly in case the manipulator is working in a dynamic, unknown environment.

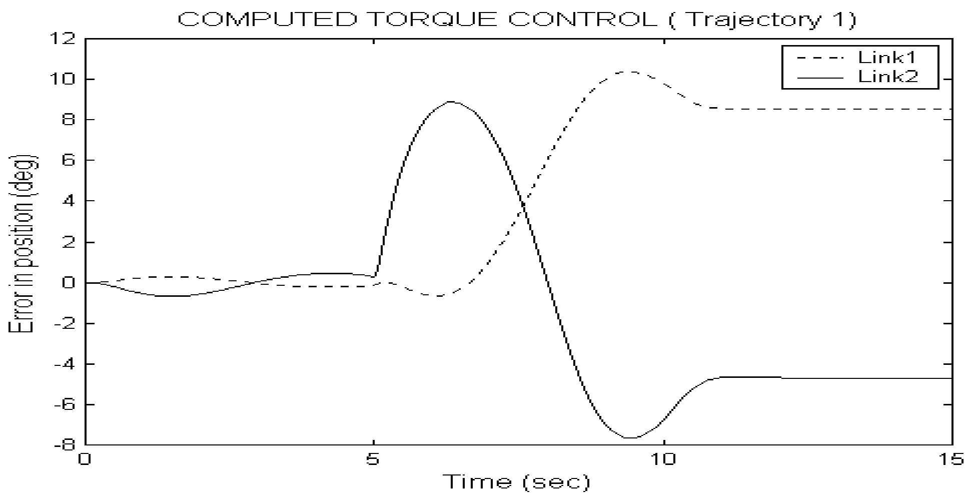


Fig. 4.3.3.1(a) Computed Torque Control errors for Trajectory 1 (Changing Parameters)

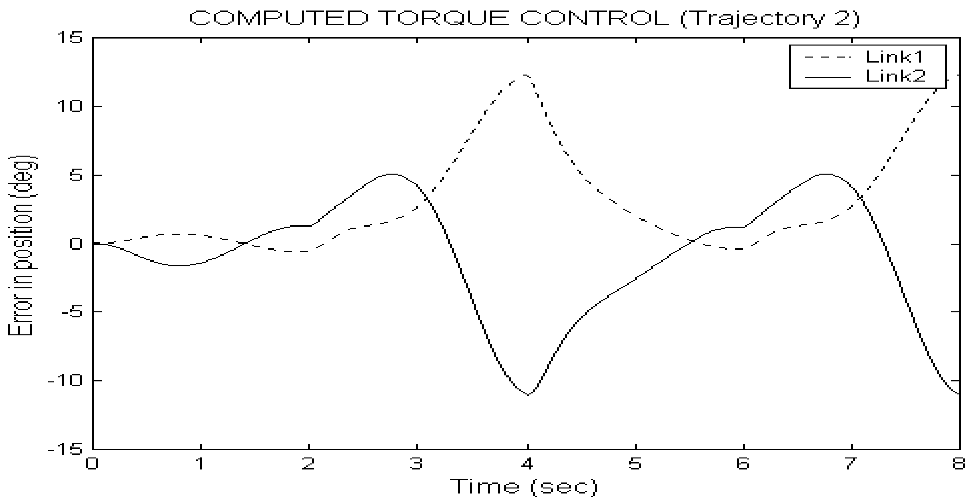


Fig. 4.3.3.1(b) Computed Torque Control errors for Trajectory 2 (Changing Parameters)

Computed Torque Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
6.1340	8.5617	4.6260	-4.7008	4.9007	12.2731	4.4288	5.0733

Table 4.3.3.1 Errors for Computed Torque control (Changing Parameters)

4.3.4. FFID CONTROL

The Feed Forward Inverse Dynamics controller, as opposed to the Computed Torque controller, is a model-based strategy in which the model is in the forward path of the control loop. Moreover as stated earlier, the model-based part is evaluated as a function of actual position and velocity as well as the desired velocity and acceleration i.e., $W\left(\theta,\dot{\theta},\dot{\theta}_d,\ddot{\theta}_d\right)$. For this controller too the initial errors at the beginning of motion are small values, as the model is assumed to be known perfectly. However, as the manipulator picks up the load and the model becomes inexact, the errors show a sudden increase. These errors again decrease when the manipulator releases the load and its original model again becomes valid. This fluctuation of errors is akin to the one seen in the case of Computed Torque control, but the magnitude of these errors is considerably reduced, as can be seen from Table 4.3.4.1. This can be mainly attributed to the fact that this controller uses comparatively 'cleaner' desired velocity and acceleration information instead of the actual values, which are always tainted with noise signals. The error profiles for the two joints for the two trajectories are shown in Fig. 4.3.4.1(a) and (b).

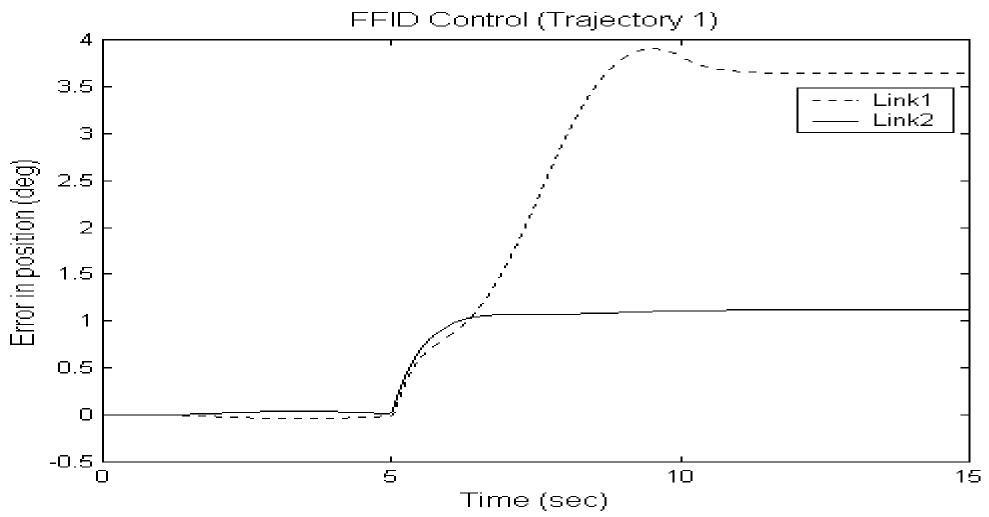


Fig. 4.3.4.1(a) FFID Control errors for Trajectory 1 (Changing Parameters)

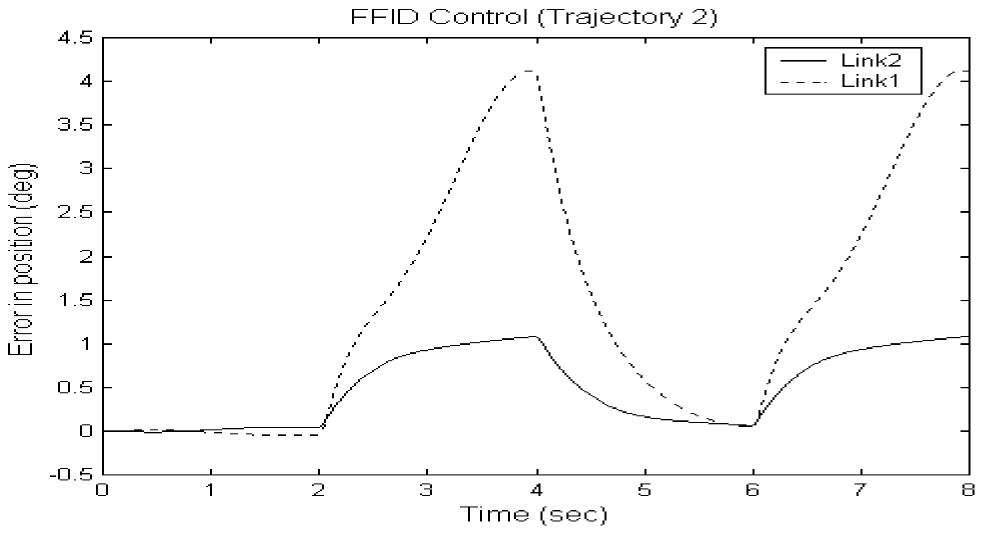


Fig. 4.3.4.1(b) FFID Control errors for Trajectory 2 (Changing Parameters)

FFID Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
2.5940	3.6395	0.8689	1.1186	2.0083	4.1268	0.6431	1.0846

Table 4.3.4.1 Errors for FFID control (Changing Parameters)

4.3.5 CDID CONTROL

The simulation results for the Critically Damped Inverse Dynamics controller are shown in Fig. 4.3.5.1(a) and Fig. 4.3.5.1(b). The overall profiles of the errors have the same basic characteristics as those for the FFID controller. There are some noticeable differences however. First, the overall magnitudes of the different error norms for CDID controller have decreased considerably. This can be seen from Table. 4.3.5.1. Secondly, the errors for joint 2 have also been limited in magnitude to a large extent. But the oscillation of the errors is still very much present and they tend to increase whenever the manipulator picks up a load. The larger effective gains of the CDID controller as compared to the FFID controller are mainly responsible for this improved performance of the controller.

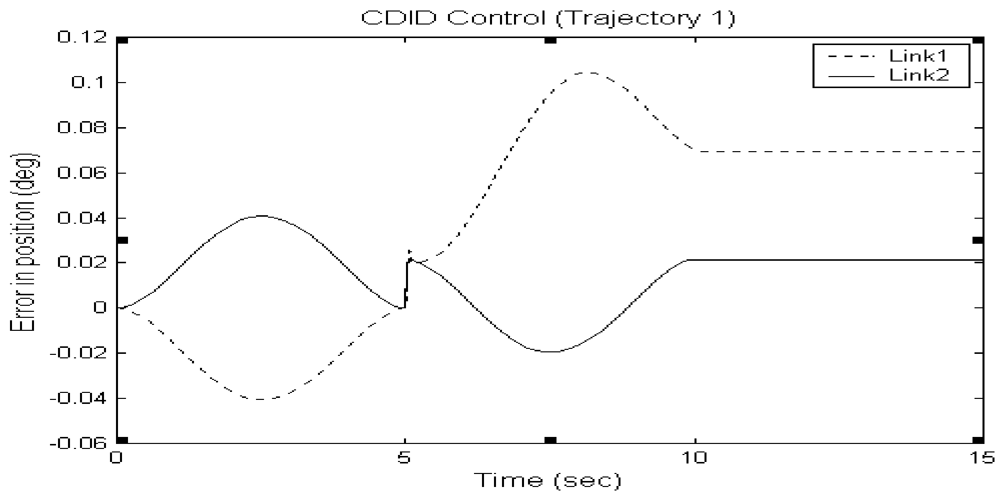


Fig. 4.3.5.1(a) CDID Control errors for Trajectory 1 (Changing Parameters)

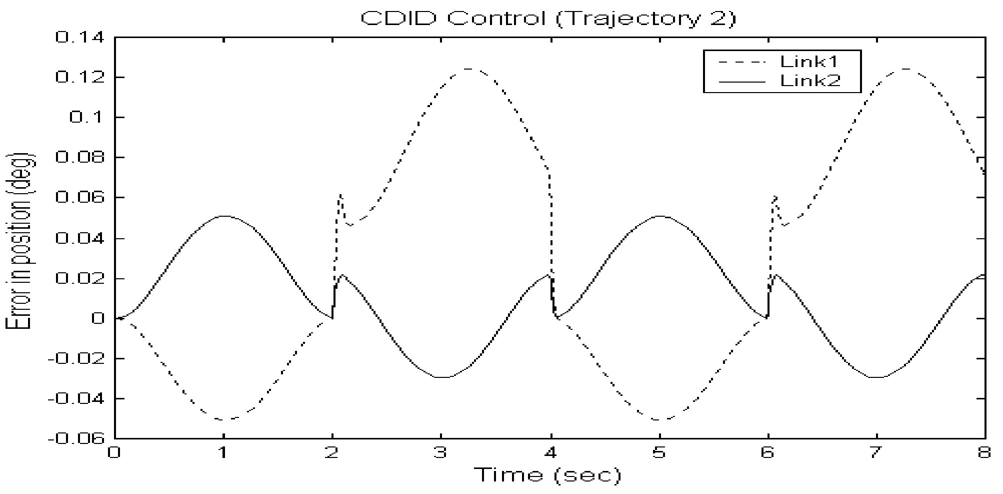


Fig. 4.3.5.1(b) CDID Control errors for Trajectory 2 (Changing Parameters)

CDID Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.0616	0.0695	0.0211	0.0214	0.0707	0.1244	0.0265	0.0509

Table 4.3.5.1 Errors for CDID control (Changing Parameters)

4.3.6 COMPARISON OF PERFORMANCE

The consolidated results for the simulations carried out in section 4.3 are presented in Table 4.3.6.1 for easy comparison of the performance of various controllers.

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°		link2 0°→ -90°→0°		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	PD Control	7.1415	9.4447	2.3758	2.6345	6.8624	9.5918	2.1554	3.3037
2.	PID control (Modified)	2.9360	3.8290	0.9742	1.0737	2.8617	4.1457	0.9264	1.4036
3.	CT control	6.1340	8.5617	4.6260	-4.7008	4.9007	12.2731	4.4288	5.0733
4.	FFID	2.5940	3.6395	0.8689	1.1186	2.0083	4.1268	0.6431	1.0846
5.	CDID	0.0616	0.0695	0.0211	0.0214	0.0707	0.1244	0.0265	0.0509

Table 4.3.6.1. Errors of different controllers for changing parameter case

Following observations are made based on simulations carried out in this section:

1. The performance of PD controller degrades further for the case when parameters of manipulator change during motion. This is mainly due to the fact that the controller gains chosen for good performance for one set of parameters are no longer optimal when the parameters of the manipulator change. Clearly one set of controller gains cannot give good performance if parameters change during motion.
2. The modified PID scheme results in better controller performance with an additional advantage of guaranteed system stability. The controller errors are reduced significantly when compared to PD controller. Modified PID could be a good choice for manipulator control using non-model based controllers.
3. The CT controller gives errors almost matching the PD controller. This shows that this controller does not perform well if the model used has inaccuracies. Moreover it just adds to the calculations required to be performed to calculate the manipulator model equations used in feedback loop.

4. The FFID controller performs appreciably better than the CT controller and slightly better than the modified PID controller. This once again indicates the merits of using the inverse dynamics in the feed forward mode and further illustrates the advantage of using the desired velocity and acceleration instead of the actual ones for composing the manipulator regressor matrix. As already stated earlier, the use of desired acceleration instead of the actual value has further advantage in terms of real world implementation.
5. The CDID controller is the best performer for the case of varying manipulator parameters. It is observed that both the RMS and steady state values of errors, for both the links for both the trajectories are considerably reduced in magnitude over the corresponding values for FFID. As explained earlier, the greater value of the proportional gain constant results in a better steady state performance, while, the use of reference velocities and accelerations for calculation of the manipulator regressor matrix instead of the actual values, results in an improved transient performance of the arm.
6. Model-based controllers (CT) do not always give a better performance than a non-model based controller (PID).

We next investigate the effect of adding modified integral error compensation on the performance of various model-based controllers.

4.4 EFFECT OF ADDING INTEGRAL COMPENSATION TO MODEL BASED CONTROLLERS

In this section we propose adding modified integral error compensation to the model based controllers namely CT, FFID and CDID, which were discussed earlier. We also investigate the effect of adding this modified integral action on the performance of these controllers. The integral action in these controllers was limited to the five iterations of the control loop performed for every new set point supplied by the trajectory generator. The errors were thus summed up for only these five iterations and the summation was reset to zero whenever the trajectory generator supplied a new set point. This was done primarily to keep the higher order effects introduced by integral error compensation from dominating the response and resulting in possible instability of the system. The system can be proved to be stable if the summation of errors is done as described earlier [Loria (2000)]. Moreover as a precaution, provision was made in the software to switch off the integral action completely in case of errors growing beyond a presettable upper bound.

4.4.1 COMPUTED TORQUE + INTEGRAL ERROR CONTROL

The first controller investigated in this section is the Computed Torque controller discussed previously in section 4.2.1. The block diagram of this controller with integral error compensation is shown in Fig. 4.4.1.1. The control law for this controller is given by equation 4.4.1.1 as

$$\tau_{ctie} = M(\theta)\left[\ddot{\theta}_d + K_D\dot{e} + K_P e\right] + V_M(\theta,\dot{\theta})\dot{\theta} + G(\theta) + K_I\int edt \qquad (4.4.1.1)$$

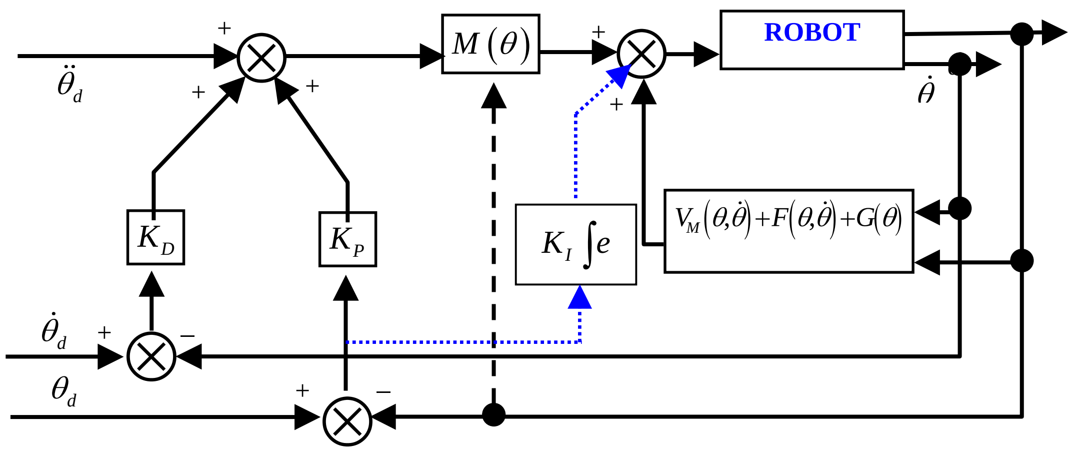


Fig. 4.4.1.1. Block diagram of Computed Torque + Integral Error Control

Fig. 4.4.1.2(a) and 4.4.1.2(b) show the error profiles for this controller for trajectory 1 and trajectory 2 respectively. Table 4.4.1.1 list the various errors for this controller for the two trajectories. When the two error profiles of Fig 4.4.1.2(a) and 4.4.1.2(b) are compared with the error profiles for the CT controller without integral error compensation, given in Fig.4.3.3.1(a) and 4.3.3.1(b), a marked improvement in performance is noticed.

For the first trajectory the RMS values of the errors are brought down considerably and so are the steady state errors. The steady state errors however have only been reduced and not removed altogether because of the special nature of integral action. It is mainly because the summation of errors is not over the entire trajectory but only for five iterations of the control loop, after which the summation is reset to zero.

For the second trajectory, which depicts pick-and-place kind of motion, we notice a similar improvement as seen for the first trajectory. Both the RMS and maximum values of errors reduce considerably when compared to Computed Torque controller without integral error compensation.

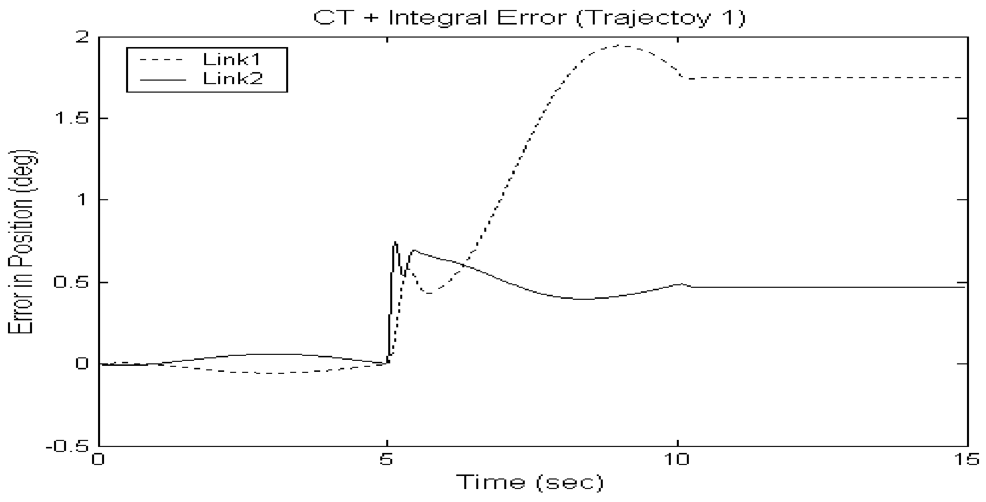


Fig. 4.4.1.2(a) CT + Integral Error Control errors for Trajectory 1

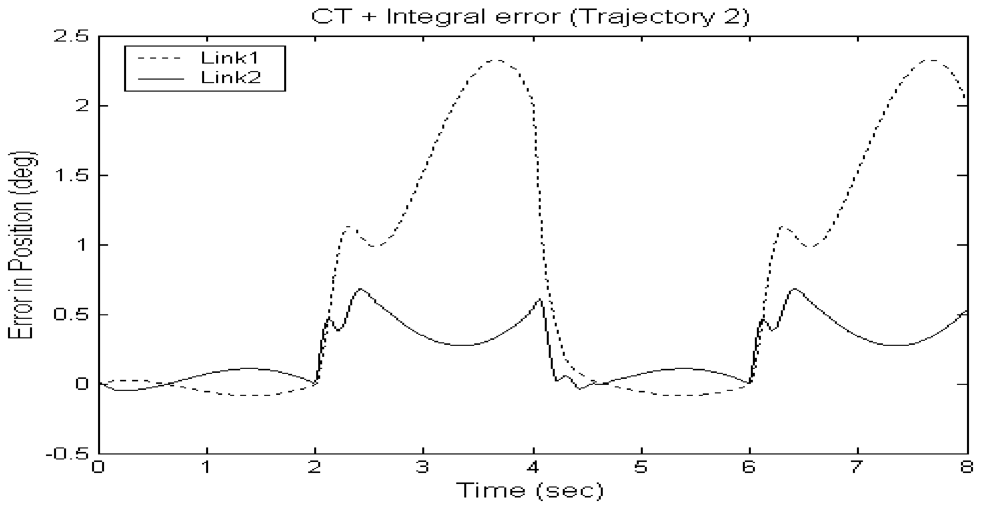


Fig. 4.4.1.2(b) CT + Integral Error Control errors for Trajectory 2

CT + Integral Error Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2** 0°→ -90°→0°		**Link1** 0°→ 45°→0°→		**Link2** 0°→ -45°→0°→	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.2919	1.7460	0.4019	0.4721	1.1894	2.3245	0.3118	0.6798

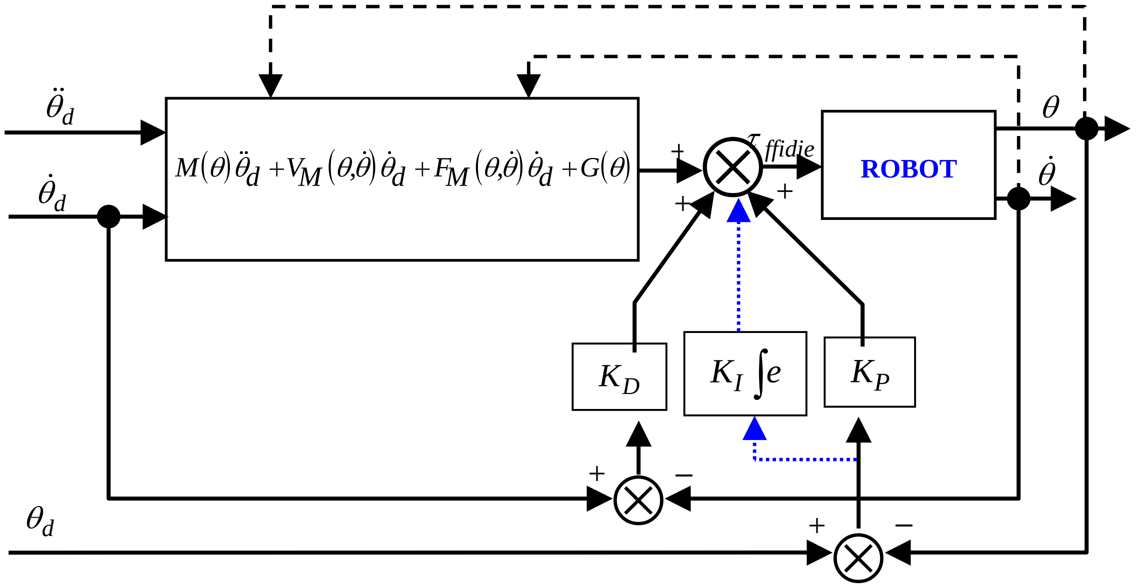
Table 4.4.1.1 Errors for CT + Integral Error control

4.4.2 FEED FORWARD INVERSE DYNAMICS + INTEGRAL ERROR CONTROL

The second controller investigated in this section is the Feed Forward Inverse Dynamics controller discussed previously in section 4.2.2. The block diagram of this controller with integral error compensation is shown in Fig. 4.4.2.1. The control law for this controller is given by equation 4.4.2.1 as

$$\tau_{ffidie} = M(\theta)\ddot{\theta}_d + V_M(\theta,\dot{\theta})\dot{\theta}_d + F_M(\theta,\dot{\theta})\dot{\theta}_d + G(\theta) + K_D\dot{e} + K_P e + K_I\int edt \qquad (4.4.2.1)$$

Fig. 4.4.2.2(a) and 4.4.2.2(b) show the error profiles for this controller for trajectory 1 and trajectory 2 respectively. Table 4.4.2.1 list the various errors for this controller for the two trajectories. When the two error profiles of Fig 4.4.2.2(a) and 4.4.2.2(b) are compared with the error profiles for the FFID controller without integral error compensation, given in Fig.4.3.4.1 (a) and 4.3.4.1(b), a marked improvement in



$\ddot{\theta}_d$
$\dot{\theta}_d$
θ_d
$M(\theta)\ddot{\theta}_d + V_M(\theta,\dot{\theta})\dot{\theta}_d + F_M(\theta,\dot{\theta})\dot{\theta}_d + G(\theta)$
τ_{ffidie}
ROBOT
θ
$\dot{\theta}$
K_D
$K_I\int e$
K_P
+
+
+
+
–
+
–

Fig. 4.4.2.1 Block diagram of Feed Forward Inverse Dynamics + Integral Error Control

performance can be noticed.

For the first trajectory the RMS values of the errors are brought down considerably and so are the steady state errors. The steady state errors however have only been reduced and not removed altogether because of the same reason as stated previously.

For the second trajectory, we notice a similar improvement as seen for the first trajectory. Both the RMS and maximum values of errors reduce considerably when compared to FFID controller without integral error compensation. The reduction in errors for this controller however, is not as marked as that for the CT controller.

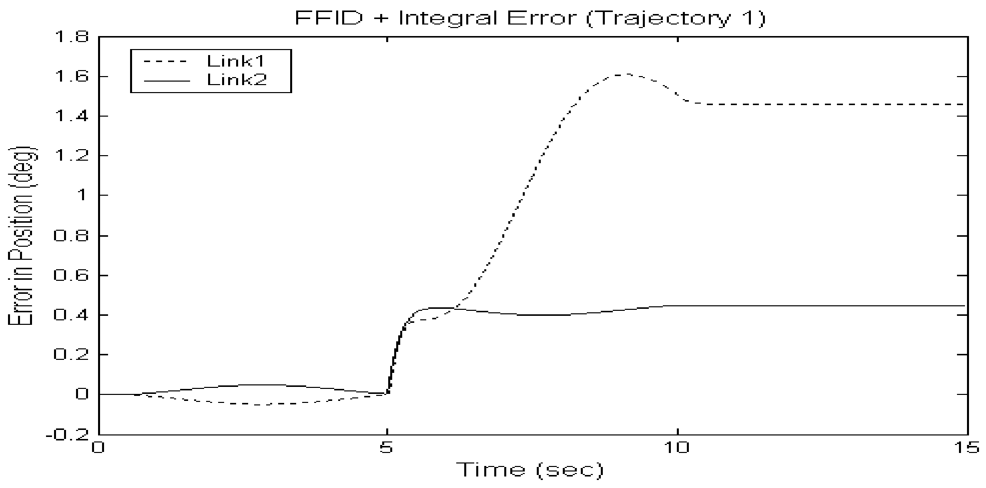


Fig. 4.4.2.2(a) FFID + Integral Error Control errors for Trajectory 1

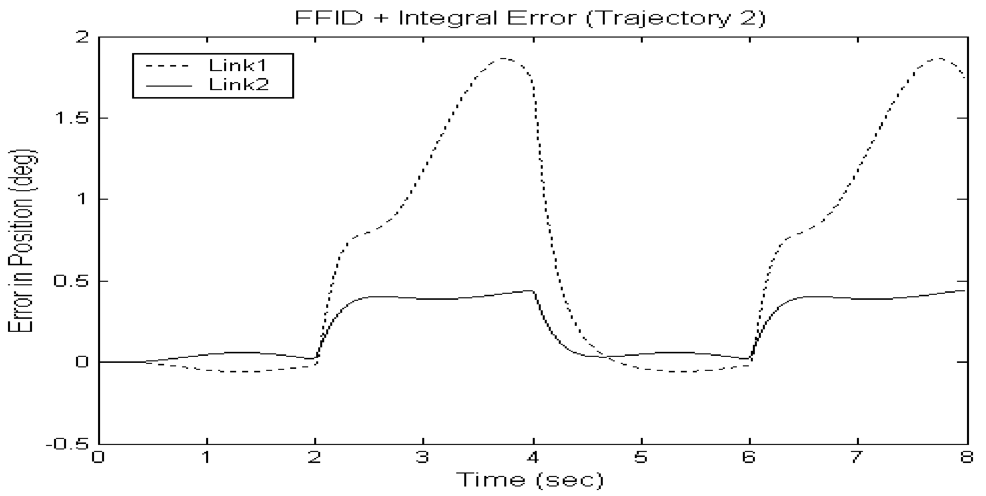


Fig. 4.4.2.2(b) FFID + Integral Error Control errors for Trajectory 2

FFID+ Integral Error Control Errors (degrees)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.0687	1.4588	0.3508	0.4488	0.9466	1.8639	0.2759	0.4419

Table 4.4.2.1 Errors for FFID + Integral Error control

4.4.3 CRITICALLY DAMPED INVERSE DYNAMICS + INTEGRAL ERROR CONTROL

The last controller investigated in this section is the Critically Damped Inverse Dynamics controller discussed previously in section 4.2.3. The block diagram of this controller with integral error compensation is shown in Fig. 4.4.3.1. The control law for this controller is given by equation 4.4.3.1 as

$$\tau_{cdidie} = M(\theta)\ddot{\theta}_R + V_M\left(\theta,\dot{\theta}\right)\dot{\theta}_R + F_M\left(\theta,\dot{\theta}\right)\dot{\theta}_R + G(\theta) + K_D\dot{e} + K_D\Lambda e + K_I\int e \qquad (4.4.3.1)$$

Fig. 4.4.3.2(a) and 4.4.3.2(b) show the error profiles for this controller for trajectory 1 and trajectory 2 respectively. Table 4.4.3.1 list the various errors for this controller for the two trajectories. When the two error profiles of Fig 4.4.3.2(a) and 4.4.3.2(b) are compared with the error profiled for the CDID controller without integral error compensation, given in Fig.4.3.5.1 (a) and 4.3.5.1(b), we do not notice any marked improvement in performance. In fact the improvement in errors is only discernible when we compare the values in Table 4.4.3.1 and Table 4.3.5.1. This is mainly due to the fact that the errors for CDID controller without integral error compensation are already pretty low and do not sum up to a substantial value over the five iterations of the control loop. Choosing a higher value of can reduce the errors further K_I but we have not done so here because the intention is to compare the performance of different controller under 'similar' conditions.

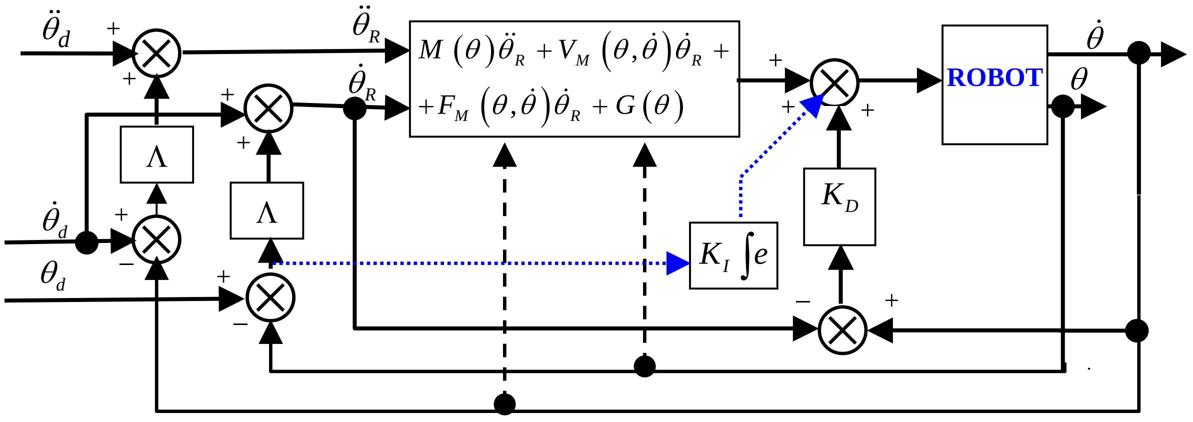


Fig 4.4.3.1 Block diagram of CDID + Integral Error Controller

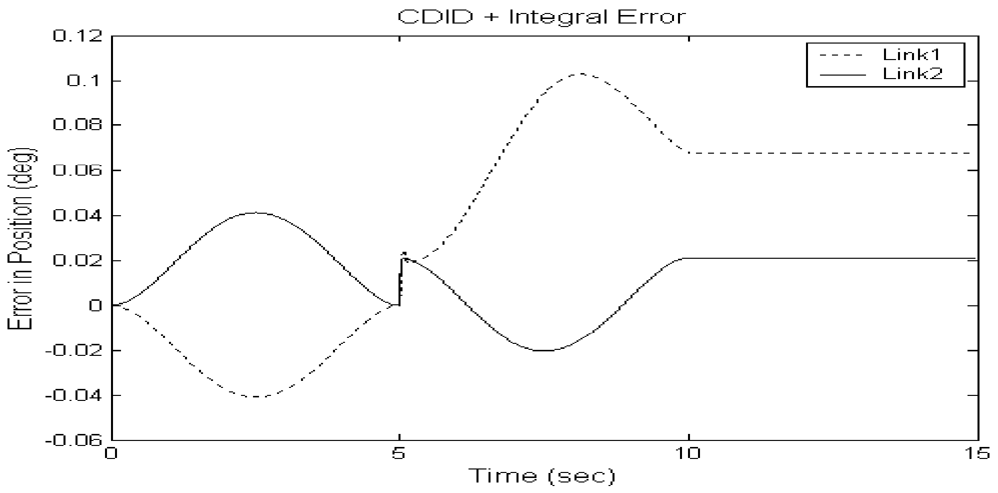


Fig. 4.4.3.2(a) CDID + Integral Error Control errors for Trajectory 1

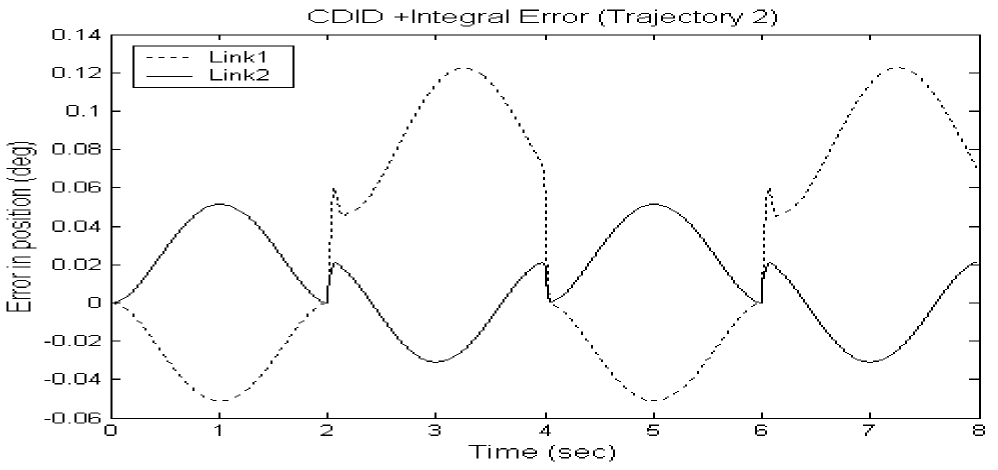


Fig. 4.4.3.2(b) CDID + Integral Error Control errors for Trajectory 2

CDID+ Integral Error Control Errors (degrees)							
TRAJECTORY NO.1				TRAJECTORY NO.2			
Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.0604	0.0676	0.0210	0.0208	0.0697	0.1226	0.0268	0.0513

Table 4.4.3.1 Errors for CDID + Integral Error control

4.4.4. COMPARISON OF PERFORMANCE

The consolidated results for the simulations carried out in section 4.3 and 4.4 are presented in Table 4.4.4.1 for easy comparison of the performance of various controllers.

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	PD Control	7.1415	9.4447	2.3758	2.6345	6.8624	9.5918	2.1554	3.3037
2.	PID control	2.9360	3.8290	0.9742	1.0737	2.8617	4.1457	0.9264	1.4036
3.	CT	6.1340	8.5617	4.6260	-4.7008	4.9007	12.2731	4.4288	5.0733
4.	FFID	2.5940	3.6395	0.8689	1.1186	2.0083	4.1268	0.6431	1.0846
5.	CDID	0.0616	0.0695	0.0211	0.0214	0.0707	0.1244	0.0265	0.0509
6.	CT+IE*	1.2919	1.7460	0.4019	0.4721	1.1894	2.3245	0.3118	0.6798
7.	FFID+IE*	1.0687	1.4588	0.3508	0.4488	0.9466	1.8639	0.2759	0.4419
8.	CDID+IE*	0.0604	0.0676	0.0210	0.0208	0.0697	0.1226	0.0268	0.0513

* Integral Error compensation

Table 4.4.4.1. Errors (in degrees) for different controllers with modified IE compensation

Following observations are made based on simulations carried out in this section:

1. The use of integral error compensation improves the performance of CT and FFID controllers appreciably. Hence a ‘judicious’ use of integral error compensation as discussed previously, seems advisable.
2. The use of integral error compensation does not show any appreciable performance gain in case of CDID controller, as the errors of the original controller were already low.
3. The CDID + IE controller gives an overall best performance in the category of conventional controllers.

4.5 CONCLUDING REMARKS

In this chapter we investigated some conventional manipulator controllers for their performance under different situations. These situations were: manipulator model known exactly, model not known exactly and model changing during the course of motion. We also investigated the effect of adding a modified integral action to these conventional controllers.

The results for various controllers presented in this chapter are for the case when the manipulator parameters change only slightly. In real practice, the parameter values may change by over 200% or more, as the manipulator operates in its work environment and picks up large loads. As a result the ensuing errors due to large parameter variations would also be large. This fact forms the basis of development of some non-conventional control strategies, which can absorb the effect of parameter variation in its performance. These controllers can adapt themselves depending upon the changing parameter values. We discuss adaptive control of manipulators in the next chapter.

ADAPTIVE CONTROL OF ROBOT MANIPULATORS

5.0 INTRODUCTION

As seen in the last chapter, due to the highly non-linear nature of manipulator dynamics and the variable nature of manipulator parameters, the conventional non-linear control of manipulators falls short of performance expectations in applications requiring very accurate and precise motion control. This problem is further compounded in the case of direct drive robots, which do not use any torque amplifying gearings for high speed and high precision tasks. This in turn means that we cannot in general neglect the cross coupling effects of manipulator dynamics for these direct drive robots.

The variable nature of manipulator parameters suggests the use of Adaptive controllers for the control of manipulators [Pagilla and Biao (2000), Song (1994)]. These controllers can either estimate the unknown manipulator parameters [Datta and Ming (1996), Kawasaki et al. (1996)] or they can change the controller gains depending on the prevailing position and/or velocity errors in the system [Spong and Ortega (1996), Colbaugh and Seraji (1994)]. Some of the important desirable goals for design of adaptive controllers for robot manipulators are:

- Insensitivity to parameter uncertainties
- Insensitivity to unknown payload variations
- Low demand for on-line computations
- Decoupled joint response

In general the adaptive controller design problem is as follows: given the desired joint position $\theta_d(t)$, and with some or all the manipulator parameters unknown, derive a control law for the actuator torques, and an estimation law for the unknown parameters, such that the manipulator joint position $\theta(t)$ precisely tracks $\theta_d(t)$ after an initial adaptation process. Adaptive control design approaches can be broadly classified into two categories [Astrom and Wittenmark (1995)]:

(i) Model reference adaptive control (MRAC) and

(ii) Self-tuning adaptive control

The structures of these two types of adaptive control systems are shown in Fig. 5.1(a) and 5.1(b). Existing robot adaptive schemes are derived from the applications of these two approaches. The adaptive controller can tackle the problem of parameter variation to a great extent and give good performance even in the face of very large load variation. In fact the adaptive controller goes on improving with time as it keeps on extracting the parameter information while executing a trajectory.

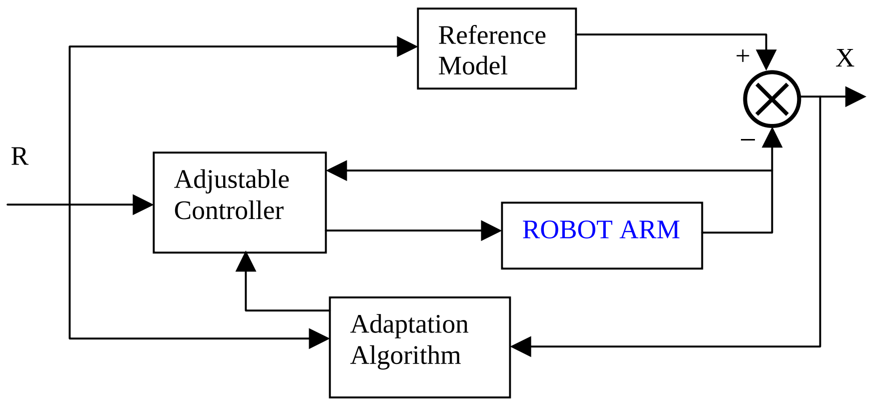


Fig. 5.1(a) Model reference adaptive controller

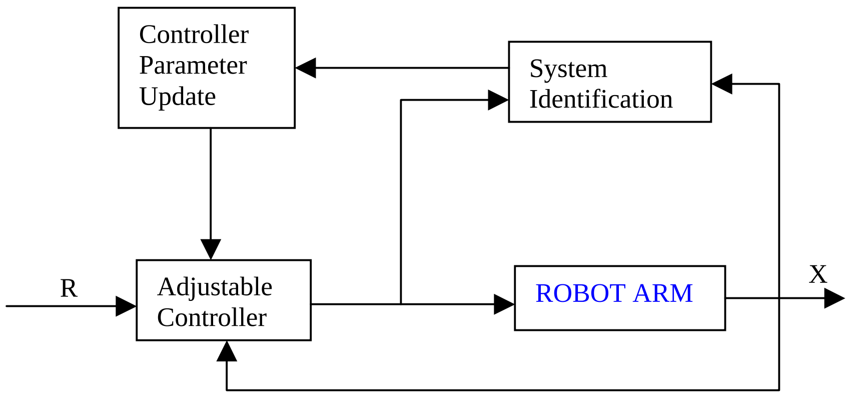


Fig. 5.1(b) Self tuning adaptive controller

In this chapter we study few adaptive control strategies for their performance in face of different operating conditions. We also study the effect of adding a modified integral action on performance of these controllers and do a comparative analysis of performance of these controllers.

5.1 ISSUES IN ADAPTIVE CONTROL OF MANIPULATORS

Some of the main issues associated with adaptive control of manipulators are listed below:

- The strong non-linearity of robot dynamics makes the analysis of adaptive controllers difficult.
- Most of the controllers still rely on approximations and assumptions such as local linearization, time invariance of parameters or decoupled nature of dynamics to prove stability.
- These adaptive schemes are computation intensive and require a fast processor for their implementation.
- The direct adaptive approach is computationally much less expensive than the indirect or the composite approach.
- Theoretical analysis and computer simulations of an adaptive controller are important but not sufficient. This is because of inherent factors such as unmodelled high frequency dynamics and measurement noise are generally neglected in stability analysis.
- For convergence of parameter values, the reference signal should be rich enough, i.e., it should contain sufficiently high frequency components.
- The parameter values may converge to different magnitudes for different trajectories. This implies transients during switching from one trajectory to another.
- The controller parameters may not always converge to ‘true’ plant values.

The importance and significance of these issues are highlighted in following sections, where we simulate the behavior of these controllers for different situations.

5.2. TESTING METHOD

The Adaptive controllers simulated in this section were tested for two different trajectories. These two trajectories are same as those used for testing the conventional controllers and described in section 4.3.1. We briefly describe the salient points of these trajectories again for sake of clarity in the context of adaptive motion control of manipulators.

In the first trajectory, the first joint was required to move from its initial home position (0°) to a final position of +90° in 5 seconds. On reaching the final position the manipulator picks up a load and returns back to its home position in another 5 seconds. On reaching the home position the manipulator was required to stay there with the load for another 5 seconds. Thus the desired position of first joint remains constant at 0° for the last 5 seconds of its motion. This kind of trajectory enables us to test the steady state performance of the controller. The desired motion for the second joint is exactly the same as for the first one except that it is required to move from 0° to -90° and then back to 0° in a total time of 15 seconds. Fig. 4.3.1.1(a) shows the desired joint position profiles for this trajectory.

The second test trajectory was chosen to simulate the motion of manipulator during a typical pick and place operation. Here the manipulator's first joint was required to move from its home position of 0° to a final position of +45° in 2 seconds. At this point the manipulator picks up a load and returns back to its home position in the next 2 seconds. On reaching home the manipulator releases the load and this cycle is repeated all over again. The second joint of the manipulator has a motion similar to the first one except that it moves to a final position of -45°. The errors for this trajectory were traced for two cycles, i.e., 8 seconds. The RMS and the maximum values of the errors were used for quantitative performance comparisons of various controllers for this trajectory. Fig 4.3.1.1(b) shows the joint motion profiles for this trajectory.

The controllers were tested using the above trajectories for two cases. In the first case we assumed that some initial estimate is available, of the various manipulator parameter values. This estimate is a rough approximation of the real, actual values and can be arrived at by some elementary measurements. For simulation the actual and the estimated values of the manipulator parameters were taken as shown in Tables 5.2.1 and 5.2.2 respectively. The actual manipulator parameters as same as those used previously in Chapter 4, and are also given in Table 4.3.1.1.

$m_1 = 2.0\ kg$
$m_2 = 2.0\ kg$
$l_1 = 0.26\ m$
$x_1 = 0.13\ m$
$x_2 = 0.14\ meters$
$I_{zz1} = 0.09\ kg - m^2$
$I_{zz2} = 0.09\ kg - m^2$
$F_1 = 2.5\ N - m / rad / \sec$
$F_2 = 2.5\ N - m / rad / \sec$

Table 5.2.1. Actual manipulator parameter values

$m_1 = 1.0\ kg$
$m_2 = 1.0\ kg$
$l_1 = 0.26\ m$
$x_1 = 0.11\ m$
$x_2 = 0.12\ meters$
$I_{zz1} = 0.05\ kg - m^2$
$I_{zz2} = 0.05\ kg - m^2$
$F_1 = 2.0 N - m / rad / \sec$
$F_2 = 2.0\ N - m / rad / \sec$

Table 5.2.2. Estimated manipulator parameter values

The adaptive algorithms in this case start with this apriori estimate and then adapt to the true values as the motion progresses. We thus say that the manipulator makes a ***warm start*** in this case. In the second case we assume the worst-case condition of having no information about the manipulator parameters. Here the adaptive algorithms have to start with no knowledge whatsoever of the values of different parameters, i.e., all the parameters are initialized to zero value. This situation is referred to as ***cold start.*** The parameters of the manipulator were further assumed to have changed to new values whenever it picked up a load. These new values of the parameters of manipulator with load were taken as shown in Table 5.2.3. The actual manipulator parameters as same as those used previously in Chapter 4, and are also given in Table 4.3.1.2. These values are repeated here for easy reference.

$m_1 = 3.0\ kg$
$m_2 = 3.0\ kg$
$l_1 = 0.26\ m$
$x_1 = 0.15\ m$
$x_2 = 0.16\ meters$
$I_{zz1} = 1.5\ kg - m^2$
$I_{zz2} = 0.09\ kg - m^2$
$F_1 = 2.5 N - m / rad / \sec$
$F_2 = 2.5\ N - m / rad / \sec$

Table 5.2.3. Changed manipulator parameter values (on picking up load)

In the following sections we present the results of simulation studies on some adaptive controllers tested for situations discussed above.

5.3. ADAPTIVE COMPUTED TORQUE CONTROLLER

This controller is the adaptive version of Computed torque controller discussed in detail in section 4.2.1. It was one of the first adaptive controllers proposed for adaptive manipulator control by Craig (1988). This adaptive controller suffered from many problems and is generally not preferred because of its three main disadvantages listed below:

- The algorithm requires inversion of the manipulator mass matrix, which is computationally very intensive.
- Implementation requires measurement of acceleration. Good and relatively inexpensive acceleration sensors are difficult to get and if acceleration is found from numerical differentiation of position or velocity information, then the values may be spurious in the presence of even slightest noise.
- The controller can be proved to be only locally stable in parameter error. This requires a constant check on the values of the parameters to keep them within acceptable range.

We did not investigate this controller because the aforesaid problems make its practical implementation very difficult.

5.4 ADAPTIVE CRITICALLY DAMPED INVERSE DYNAMICS CONTROLLER (ACDID)

The first adaptive controller investigated in this work is the adaptive version of the conventional Critically Damped Inverse Dynamics controller (CDID), described in detail in section 4.2.3. This controller was proposed by Slotine and Li (1988). This is a direct adaptive controller in the sense that the parameter values are adapted directly from the information about position and velocity errors of the different joints. The adaptation law is derived starting from the manipulator dynamics equation written in a linear form as in equation 3.2.28. If we define

$$\tilde{P} = \hat{P} - P \tag{5.4.1}$$

as the parameter estimation error vector, with P as the true parameter values vector and $\hat{P}$ as the vector of parameter estimates, then the linearity property of the robot dynamics enables us to write

$$\tilde{M}(\theta)\ddot{\theta}_R + \tilde{V}_M\left(\theta,\dot{\theta}\right)\dot{\theta}_R + \tilde{F}_M\left(\theta,\dot{\theta}\right)\dot{\theta}_R + \tilde{G}(\theta) = W\left(\theta,\dot{\theta},\dot{\theta}_R,\ddot{\theta}_R\right)\tilde{P} \tag{5.4.2}$$

Where

$$\begin{aligned}\tilde{M} &= \hat{M} - M\\ \tilde{V}_M &= \hat{V}_M - V_M\\ \tilde{F}_M &= \hat{F}_M - F_M\\ \tilde{G} &= \hat{G} - G\end{aligned}$$

and $\dot{\theta}_R$ and $\ddot{\theta}_R$ are reference trajectories as defined in equations 4.2.3.1 and 4.2.3.2. The control law used can then be written as

$$\tau = \hat{M}(\theta)\ddot{\theta}_R + \hat{V}_M\left(\theta,\dot{\theta}\right)\dot{\theta}_R + \hat{F}_M\left(\theta,\dot{\theta}\right)\dot{\theta}_R + \hat{G}(\theta) - K_D\dot{e} \tag{5.4.3}$$

where $\dot{e}$ is as defined in equation 4.2.3.4 and K_D is uniformly positive definite controller gain matrix. The stability of the controller can be proved, by considering the Lyapunov function candidate,

$$V(t) = \frac{1}{2}\left[\dot{e}^T M(\theta)\dot{e} + \tilde{P}^T\Gamma^{-1}\tilde{P}\right] \tag{5.4.4}$$

where Γ is a constant positive definite adaptation gain matrix.

Differentiating $V(t)$ with respect to time leads to equation

$$\dot{V}(t) = \dot{e}^T\left(\tau - M(\theta)\ddot{\theta}_R - V_M\left(\theta,\dot{\theta}\right)\dot{\theta}_R - G(\theta)\right) + \tilde{P}\Gamma^{-1}\dot{\tilde{P}} \tag{5.4.5}$$

Substituting the control law in the above equation results in

$$\dot{V}(t) = -\dot{e}^T K_D \dot{e} + \tilde{P}^T \left[\Gamma^{-1} \dot{\hat{P}} + W^T \dot{e} \right] \tag{5.4.6}$$

Now if we choose the adaptation law as

$$\dot{\hat{P}} = -\Gamma W^T \dot{e} \tag{5.4.7}$$

then equation 5.4.6 reduces to

$$\dot{V}(t) = -\dot{e}^T K_D \dot{e} \le 0 \tag{5.4.8}$$

Thus the system is proved to be stable if the gain matrix K_D is chosen to be positive definite. The block diagram for this controller is shown in Fig.5.4.1.

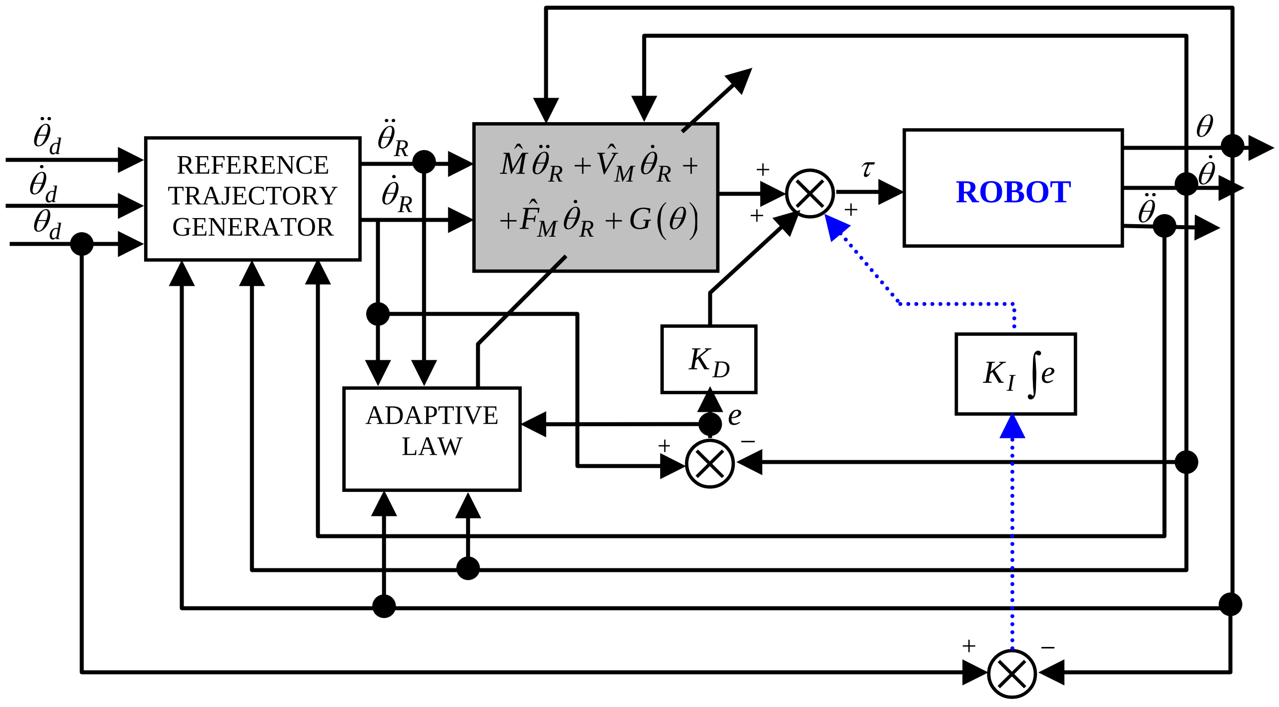


Fig.5.4.1 Block diagram of Adaptive Critically Damped Controller (with Integral Error Feedback)

The simulation for ACDID controller was carried out for the two trajectories for the warm and cold start cases. Fig 5.4.2(a) shows the error profiles for first trajectory, warm start, while Fig 5.4.2(b) shows the error profiles for second trajectory, warm start. Figures 5.4.2(c) and 5.4.2(d) are for the cold start case.

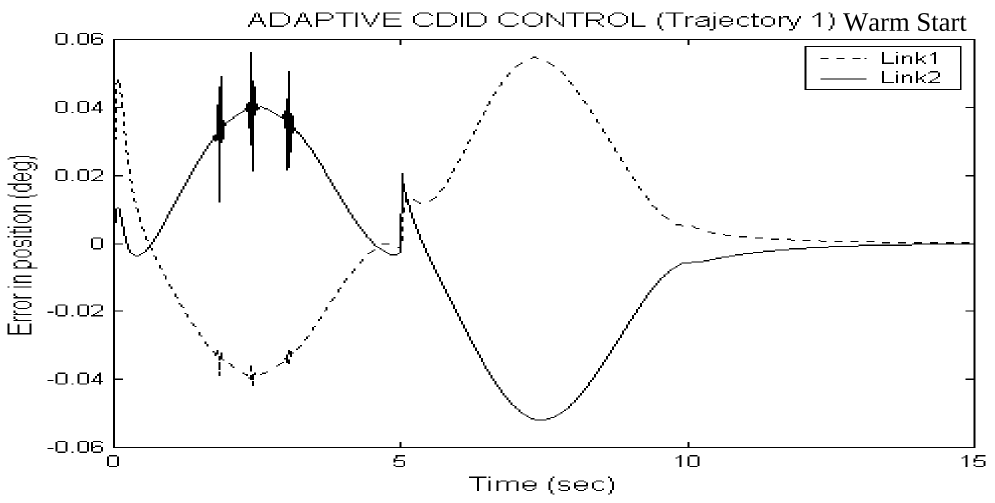


Fig. 5.4.2(a) Errors for ACDID, Trajectory 1 (warm start)

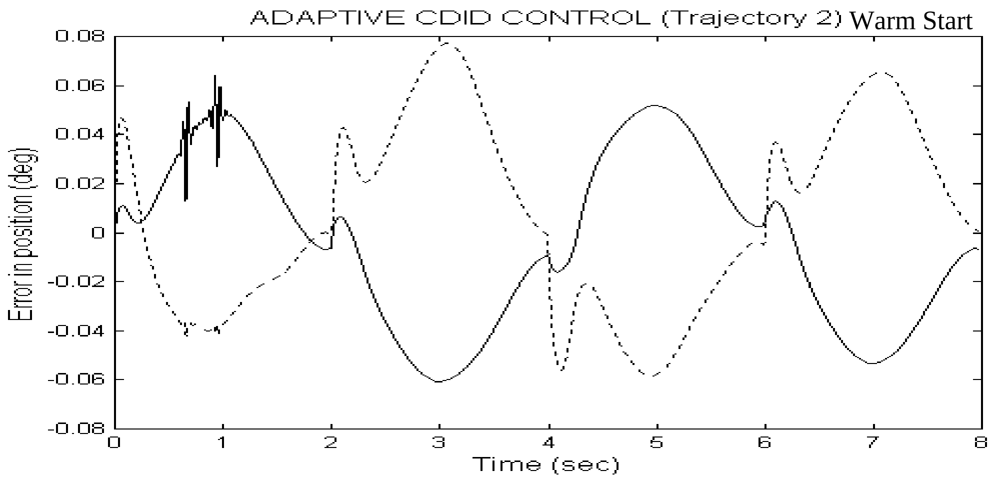


Fig. 5.4.2(b) Errors for ACDID, Trajectory 2 (warm start)

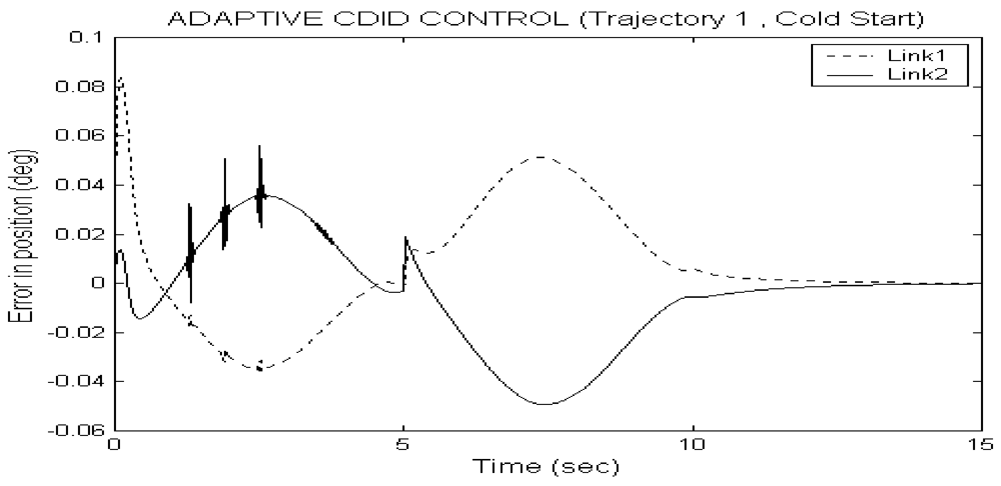


Fig. 5.4.2(c) Errors for ACDID, Trajectory 1 (cold start)

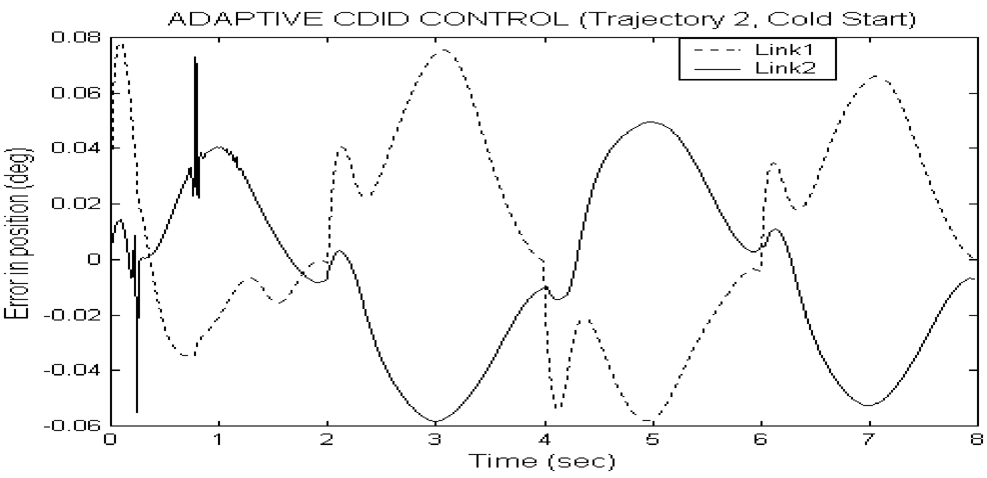


Fig. 5.4.2(d) Errors for ACDID, Trajectory 2 (cold start)

	ADAPTIVE CDID Control Errors (degrees)							
	TRAJECTORY NO.1				TRAJECTORY NO.2			
	Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
	RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
Warm Start	0.0247	0.0001	0.0238	-0.0002	0.0391	0.0772	0.0344	0.0642
Cold Start	0.0244	0.0001	0.0220	-0.0002	0.0390	0.0785	0.0323	0.0732

Table 5.4.1 Errors for ACDID (warm and cold start)

Table 5.4.1 summarizes the errors for ACDID for the two trajectories for the warm and cold start cases. As can be seen from the table and the error profiles, the errors for ACDID controller are minimal and performance is much better than any conventional controller discussed in chapter 4. Moreover as the errors are already low there is hardly any perceptible difference between the cold and warm start cases. Also it can be seen that the errors are higher for the cold start case when compared to warm start in the beginning of motion. The controller then learns the parameter values within first few iterations and after that the two profiles for cold and warm start almost match.

A modification done to ACDID controller was inclusion of integral error term in calculation of final controller output. This is indicated by blue dotted line in Fig. 5.4.1. The integral action was limited to only five iterations of the control loop, which are done for every new set point produced by the trajectory generator. After every five iterations the summation of errors was reset to zero. This is done primarily to keep the summation term from growing without bound or in other words to keep the system stable. This idea was discussed in detail in section 4.4. This modified ACDID controller was tested for the same two trajectories as above, for warm and cold start cases.

Fig 5.4.3(a) shows the error profiles for first trajectory, warm start, while Fig 5.4.3(b) shows the error profiles for second trajectory, warm start. Figures 5.4.3(c) and 5.4.3(d) are for the cold start case. Table 5.4.2 summarizes the errors for ACDID for the two trajectories for the warm and cold start cases.

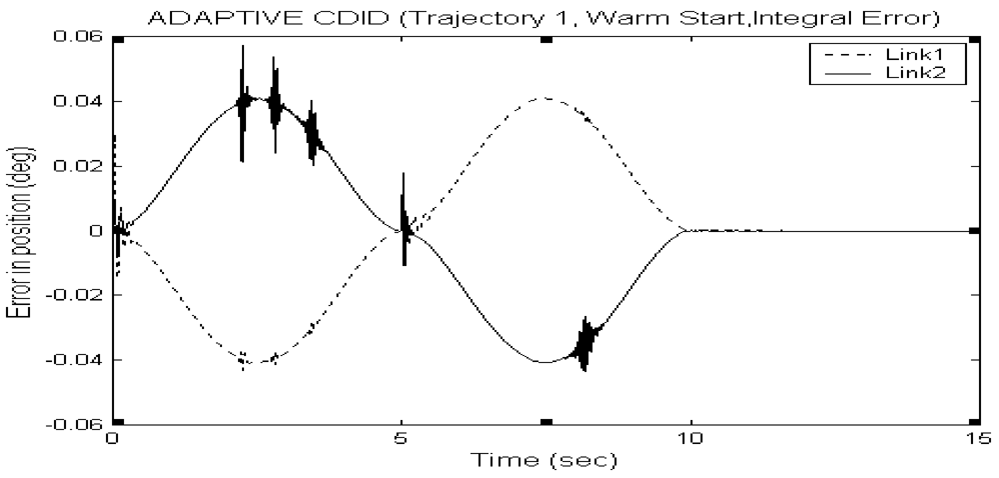


Fig. 5.4.3(a) Errors for ACDID trajectory 1 (warm start, integral error)

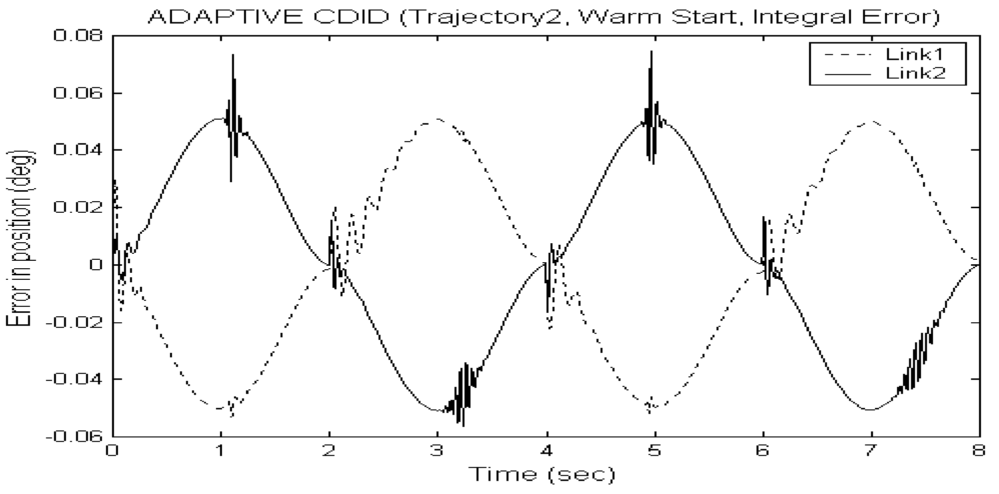


Fig. 5.4.3(b) Errors for ACDID trajectory 2 (warm start, integral error)

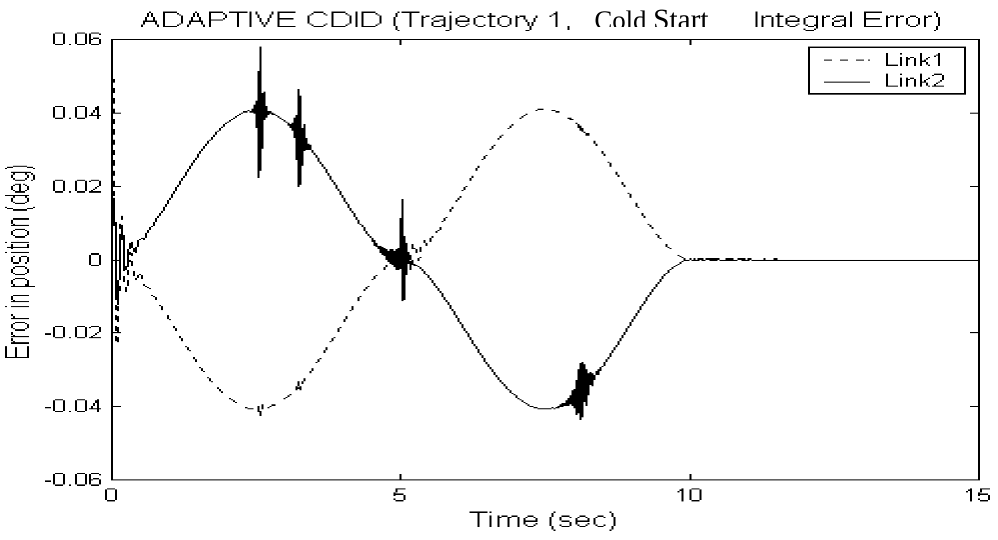


Fig. 5.4.3(c) Errors for ACDID trajectory 1 (cold start, integral error)

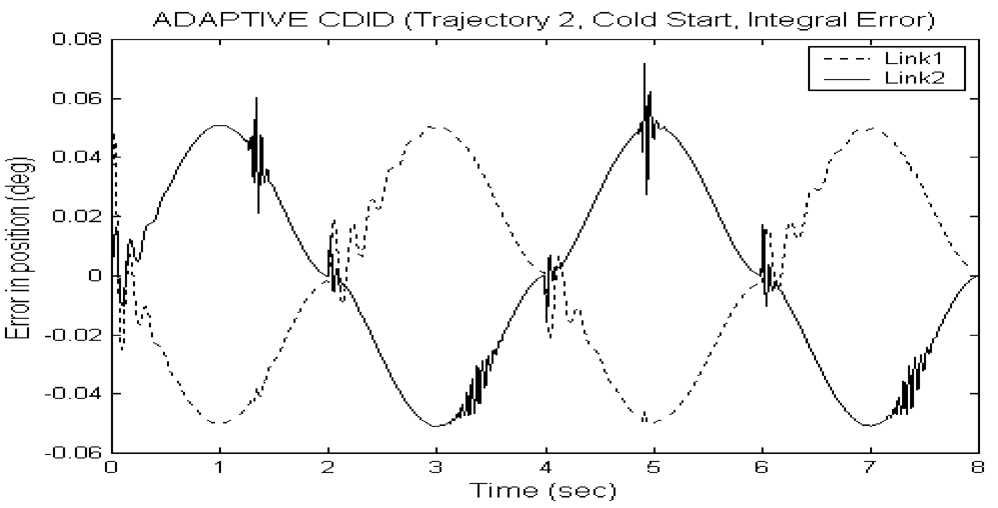


Fig. 5.4.3(d) Errors for ACDID trajectory 2 (cold start, integral error)

	ADAPTIVE CDID Control Errors (degrees)							
	TRAJECTORY NO.1				**TRAJECTORY NO.2**			
	Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
	RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
Warm Start	0.0213	0.0	0.0212	0.0	0.0323	0.0508	0.0325	0.0745
Cold Start	0.0215	0.0	0.0213	0.0	0.0324	0.0505	0.0326	0.0716

Table 5.4.2 Errors for ACDID (warm and cold start with integral error)

The introduction of integral error compensation further improves the performance of ACDID controller. For the first trajectory the major improvement is in terms of steady state errors. The steady state error goes to zero almost immediately after the manipulator reaches home at the end of ten seconds. This can be seen from Figures 5.4.3(a) and 5.4.3(c). Even for the second trajectory, there is improvement in terms of both the Maximum error and the RMS error. As the introduction of an integral error compensation term does not add much to the computational complexity of the controller and on the other hand gives improved trajectory tracking, its use with ACDID controller is advisable.

5.5 MODEL REFERENCE ADAPTIVE CONTROLLER (MRAC)

The second controller investigated is a model reference adaptive controller [Lewis et al. (1988), Maliotis and Lewis (1989)]. It is synthesized in two stages. First, the known dynamics are separated out and used to perform a global linearization on the nonlinear system. Second, a model reference adaptive controller, based on the Lyapunov stability criterion, is designed for the remaining unknown portion of the plant. This controller takes advantage of structure and any known dynamics of the system in order to increase the speed of adaptation and relax the conditions required for convergence.

The adaptation law is derived starting from the manipulator dynamic equation written as

$$M(\theta)\ddot{\theta} + F(\theta,\dot{\theta})\dot{\theta} + G(\theta)\theta = \tau \qquad (5.5.1)$$

where $M(\theta)$ is *n x n* inertia matrix, $F(\theta,\dot{\theta})$ is *n x n* matrix containing the centrifugal, coriolis and friction terms, $G(\theta)$ is a *n x 1* vector containing the gravity terms, θ is an *n x 1* joint position variable vector and τ is *n x 1* input torque vector. If the system described in 5.5.1 has some known and some unknown plant dynamics then we may write:

$$\begin{aligned} M &= M_k + M_u^* = M_k(I + M_k^{-1}M_u^*) = M_k M_u \\ G &= G_k + G_u \\ F &= F_k + F_u \end{aligned} \qquad (5.5.2)$$

where subscript '*k*' stands for known part and subscript '*u*' stands for unknown part of manipulator dynamics.

Substituting 5.5.2 in 5.5.1 results in equation

$$M_u\ddot{\theta} + M_k^{-1}F_u\dot{\theta} + M_k^{-1}G_u\theta = u \tag{5.5.3}$$

where

$$u = M_k^{-1}(\tau - F_k\dot{\theta} - G_k\theta) \tag{5.5.4}$$

If we define

$$x = \begin{bmatrix} \theta \\ \dot{\theta} \end{bmatrix} \tag{5.5.5}$$

we can write equation 5.5.3 as

$$\begin{aligned} \dot{x} &= \begin{bmatrix} 0 & I \\ -M_u^{-1}M_k^{-1}G_u & -M_u^{-1}M_k^{-1}F_u \end{bmatrix} x + \begin{bmatrix} 0 \\ M_u^{-1} \end{bmatrix} u \\ &= Ax + Bu \end{aligned} \tag{5.5.6}$$

Next we choose a reference model given by

$$\dot{x}_d = A_m x_d + B_m v \tag{5.5.7}$$

where

$$A_m = \begin{bmatrix} 0 & 0 \\ -K_1 & -K_2 \end{bmatrix}, B_m = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{5.5.8}$$

If the errors between actual and desired trajectories is defined as

$$e = x_d - x \tag{5.5.9}$$

the error dynamics will be given by

$$\begin{aligned} \dot{e} &= A_m e + (A_m - A)x - Bu + B_m v \\ &= \begin{bmatrix} e_1 \\ e_2 \end{bmatrix} \end{aligned} \tag{5.5.10}$$

The control objective is to make the error decrease asymptotically. This can be achieved if the adaptive control law chosen is

$$u = u_k + u_a \qquad 5.5.11)$$

where the linear feedback portion of the controller is given by

$$u_k = -[K_1 \quad K_2]x + v \tag{5.5.12}$$

and the adaptive portion of the control is given by

$$u_a = -[\Delta_1 \quad \Delta_2]x + \Delta_v v \tag{5.5.13}$$

where Δ_1, Δ_2 and Δ_v are adaptive gains chosen using a Lyapunov approach.

The adaptive gains are calculated as

$$\begin{aligned} \dot{\Delta}_1 &= -aw\theta^T \\ \dot{\Delta}_2 &= -aw\dot{\theta}^T \\ \dot{\Delta}_v &= bwv^T \end{aligned} \tag{5.5.14}$$

where a and b are positive scalar gains and w is the filtered error defined as

$$w = P_2 e_1 + P_3 e_2 \tag{5.5.15}$$

where

$$P = \begin{bmatrix} P_1 & P_2^T \\ P_2 & P_3 \end{bmatrix} \tag{5.5.16}$$

is the positive definite solution of the Lyapunov equation

$$A_m^T P + PA_m = -Q \tag{5.5.17}$$

with $Q > 0$.

The block diagram for this controller is shown in Fig.5.5.1.

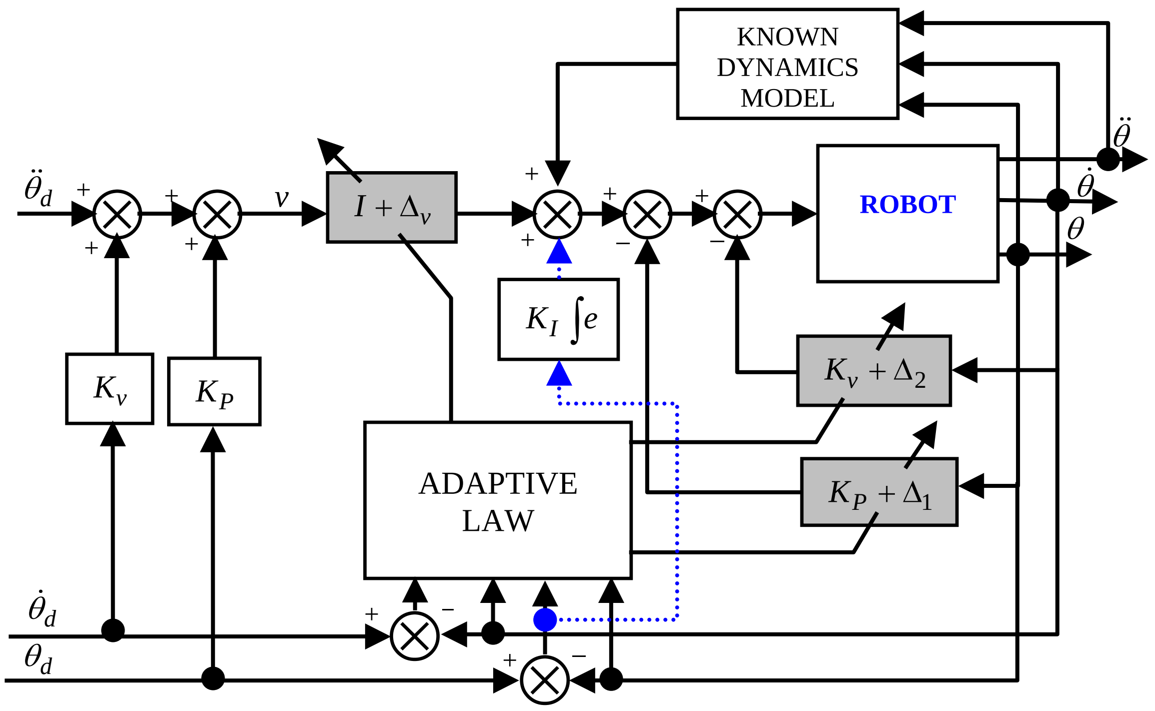


Fig.5.5.1 Block diagram of Direct Adaptive Model Reference Control (with Integral Error Feedback)

The simulation for MRAC controller was carried out for the two trajectories for the warm and cold start cases. Fig 5.5.2(a) shows the error profiles for first trajectory, warm start, while Fig 5.5.2(b) shows the error profiles for second trajectory, warm start. Figures 5.5.2(c) and 5.5.2(d) are for the cold start case.

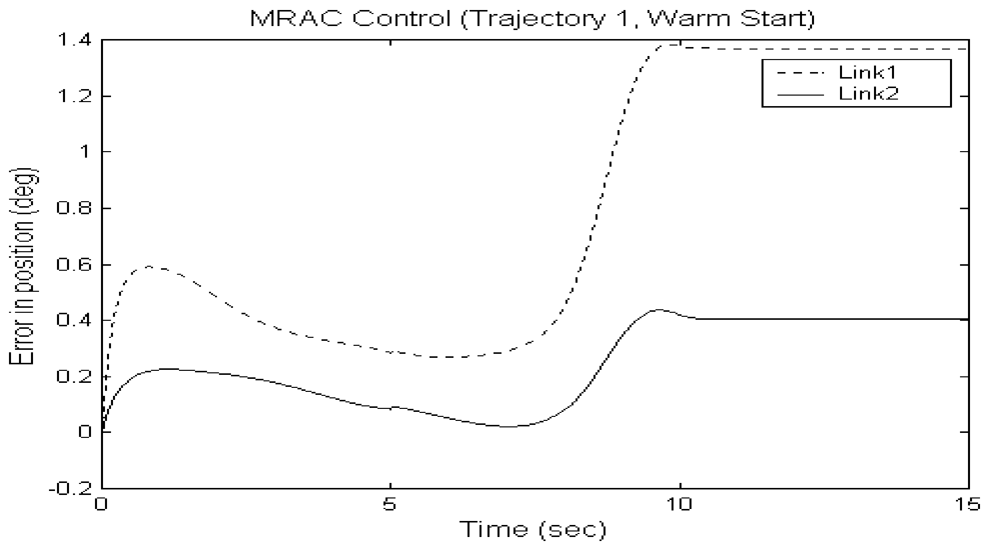


Fig. 5.5.2(a) Errors for MRAC, trajectory 1 (warm start)

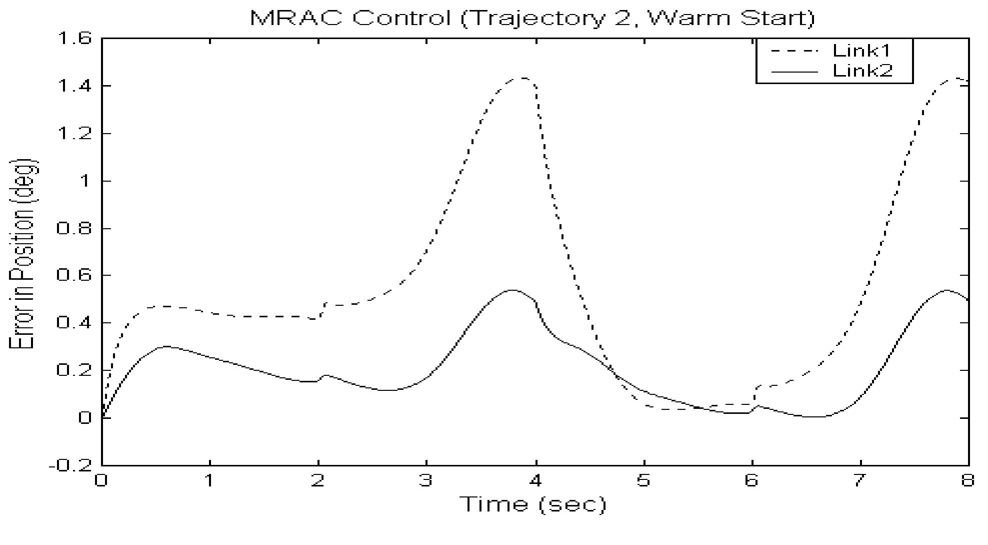


Fig. 5.5.2(b) Errors for MRAC, trajectory 2 (warm start)

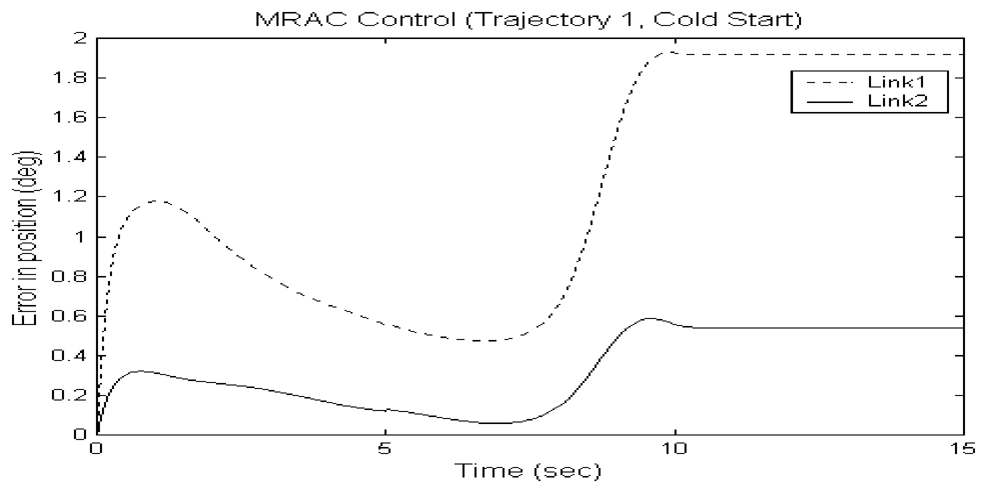


Fig. 5.5.2(c) Errors for MRAC, trajectory 1 (cold start)

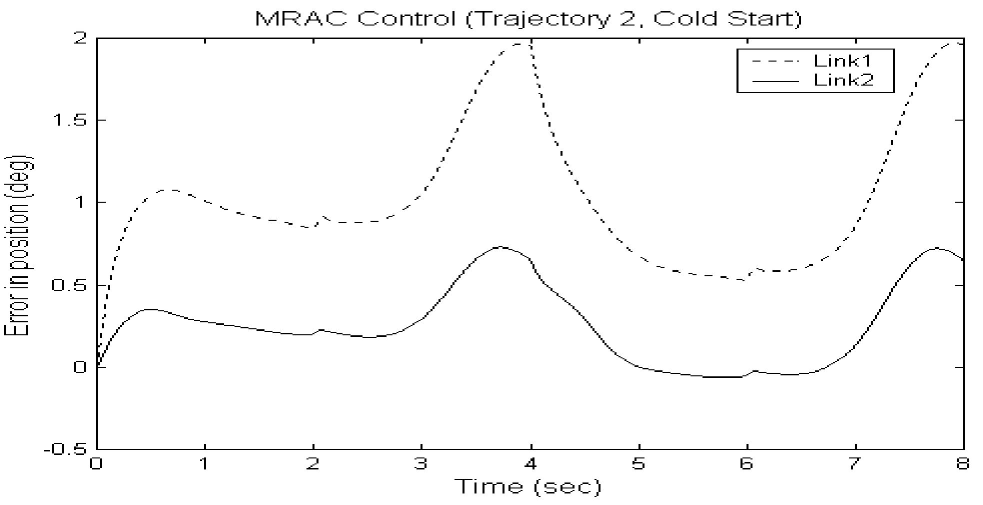


Fig. 5.5.2(d) Errors for MRAC, trajectory 2 (cold start)

	MRAC Control Errors (degrees)							
	TRAJECTORY NO.1				**TRAJECTORY NO.2**			
	Link1 0°→ 90°→0°		**Link2** 0°→ -90°→0°		**Link1** 0°→ 45°→0°→		**Link2** 0°→ -45°→0°→	
	RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
Warm Start	0.9283	1.3691	0.2827	0.4039	0.6935	1.4323	0.2628	0.5390
Cold Start	1.3565	1.9183	0.3808	0.5383	1.1114	1.9689	0.3510	0.7286

Table 5.5.1 Errors for MRAC (warm and cold start)

Table 5.5.1 summarizes the errors for MRAC for the two trajectories for the warm and cold start cases. As can be seen from the table, the errors for MRAC are quite different for the two cases of cold and warm start. The errors are appreciably lower in case of warm start as compared to cold start. This is expected because the control law is explicitly dependent upon torques due to known dynamics. Hence use of known portion of manipulator dynamics is advisable for this controller. The errors of this controller however are larger as compared to ACDID controller as can be seen from Tables 5.4.1 and 5.5.1.

A modification done to MRAC was inclusion of integral error term in calculation of final controller output. This is indicated by blue dotted line in Fig. 5.5.1. The integral action was, as previously, limited to only five iterations of the control loop, which are done for every new set point produced by the trajectory generator. After every five iterations the summation of errors was reset to zero. This modified MRAC controller was also tested for the two trajectories described earlier.

Fig 5.5.3(a) shows the error profiles for first trajectory, warm start, while Fig 5.5.3(b) shows the error profiles for second trajectory, warm start. Figures 5.5.3(c) and 5.5.3(d) are for the cold start case. Table 5.5.2 summarizes the errors for MRAC for the two trajectories for the warm and cold start cases.

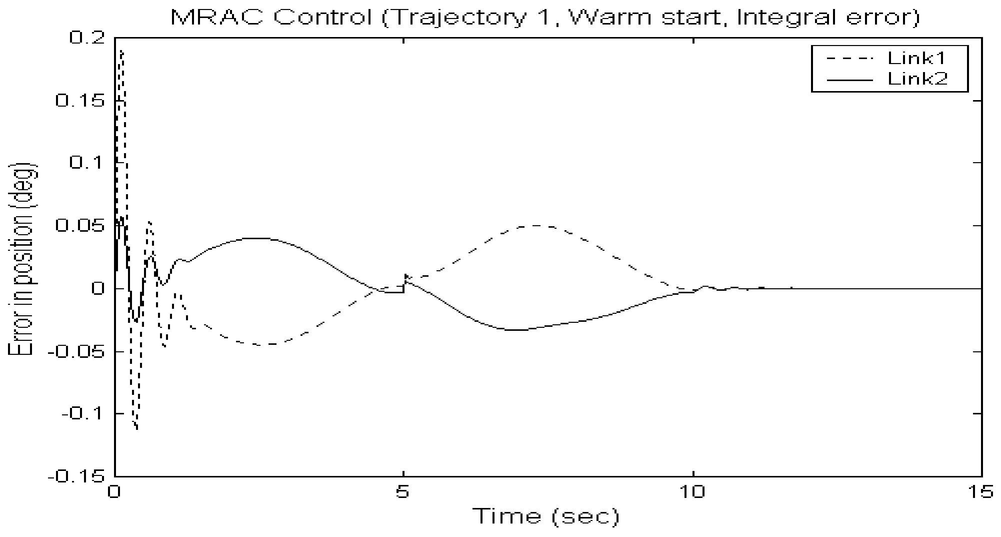


Fig. 5.5.3(a) Errors for MRAC trajectory 1 (warm start, integral error)

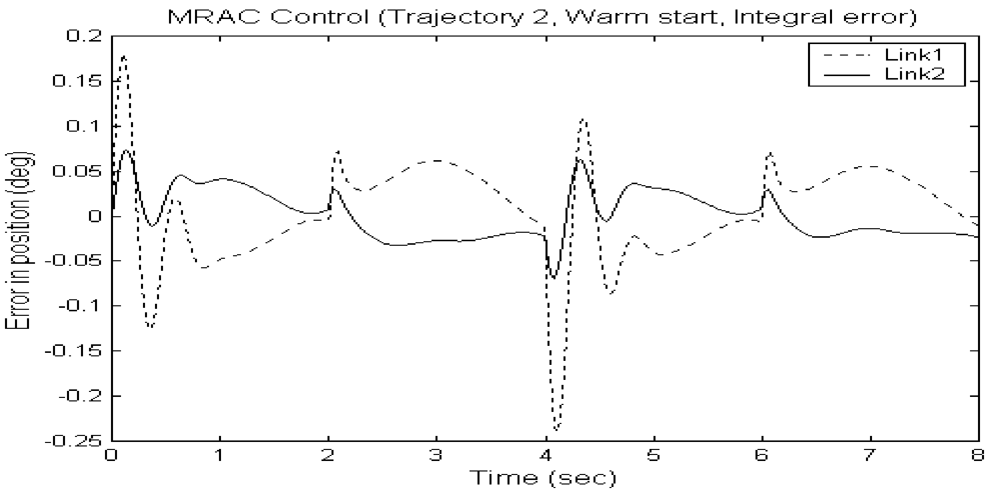


Fig. 5.5.3(b) Errors for MRAC trajectory 2 (warm start, integral error)

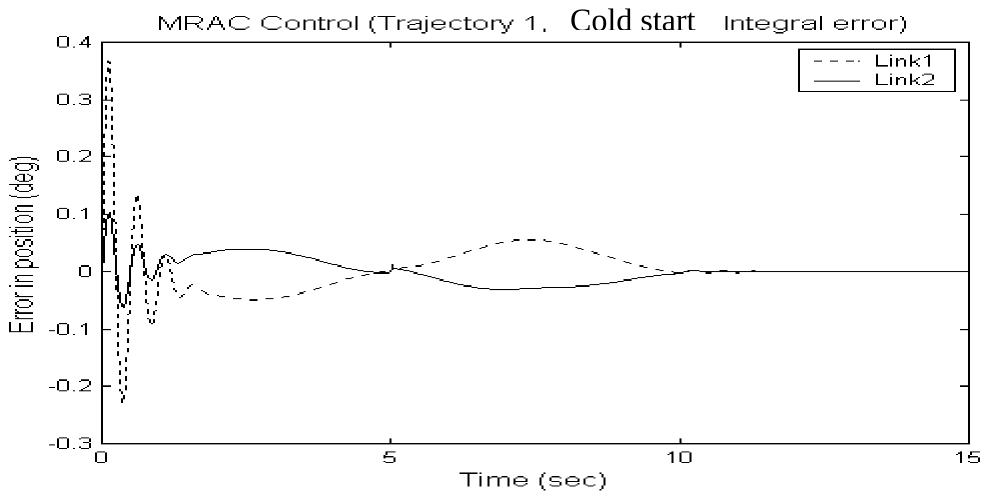


Fig. 5.5.3(c) Errors for MRAC trajectory 1 (cold start, integral error)

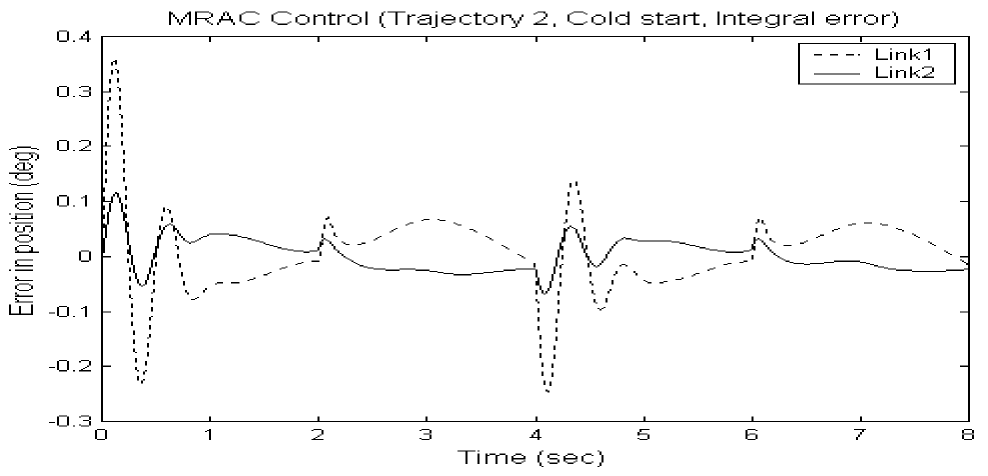


Fig. 5.5.3(d) Errors for MRAC trajectory 2 (cold start, integral error)

	MRAC Control with Integral Errors (degrees)							
	TRAJECTORY NO.1				TRAJECTORY NO.2			
	Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
	RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
Warm Start	0.0321	0.0	0.0200	0.0	0.0559	0.1800	0.0270	0.0732
Cold Start	0.0496	-0.0001	0.0221	0.0	0.0755	0.3604	0.0300	0.1158

Table 5.5.2 Errors for MRAC (warm and cold start with integral error)

As expected, the introduction of integral error compensation improves the performance of MRAC controller considerably. For the first trajectory the major improvement is in terms of both, the RMS and steady state errors. The steady state error goes to zero almost immediately after the manipulator reaches home at the end of ten seconds. This can be seen from Figures 5.5.3(a) and 5.5.3(c). For the second trajectory also there is improvement in terms of both, the Maximum error and the RMS error. As the introduction of an integral error compensation term does not add much to the computational complexity of the controller and at the same time gives improved trajectory tracking, its use with MRAC controller is advisable.

5.6 DECENTRALIZED ADAPTIVE CONTROLLER (DAC)

Decentralized control has been widely accepted by the robotics industry due to ease of implementation and tolerance to failure. Conventional controllers for industrial robots are based on independent joint control schemes in which each joint is controlled separately by a simple position servo loop with predefined constant gains. This control scheme is adequate for simple pick-and-place tasks, for which industrial robots are often used, where only point-to-point motion is of concern. However, in tasks where precise tracking of fast trajectories under different payloads is required, the independent joint, conventional robot control systems are severely inadequate.

The controller investigated in this section uses a technique for advanced manipulator control based on **adaptive independent joint control** [Magana and Tagami (1994)]**.** A major point of departure in this approach from the centralized approaches is the formulation of the problem in a decentralized control context at the outset. This control scheme has two major features. First, due to its adaptive nature, knowledge of manipulator dynamic model and parameter values or the payload parameters are not required. Second, due to its decentralized structure and controller simplicity, the scheme is computationally very fast and is amenable to parallel processing implementation within a distributed computing architecture, with one microprocessor dedicated to each joint.

The centralized model of an *n* link manipulator is given by equation 5.6.1 as:

$$M(\theta)\ddot{\theta} + V(\theta,\dot{\theta})\dot{\theta} + F(\theta)\dot{\theta} + G(\theta)\theta = \tau \qquad (5.6.1)$$

where $M(\theta)$ is *n x n* inertia matrix, $V(\theta,\dot{\theta})$ is *n x n* matrix containing the centrifugal and coriolis terms, $F(\theta)$ is *n x n* matrix containing friction terms, $G(\theta)$ is a *n x 1* vector containing the gravity terms, θ is an *n x 1* joint variable vector and τ is *n x 1* input vector.

To design the controller the centralized model of equation 5.6.1 is decomposed into *n* interconnected systems as:

$$m_{ii}(\theta)\ddot{\theta}_{ii} + \sum_{\substack{j=1 \\ j\neq i}}^{n} m_{ij}\ddot{\theta}_j + V_i(\theta,\dot{\theta}) + F_i(\dot{\theta}) + G_i(\theta) = \tau_i \qquad (5.6.2)$$

The coupling effects from each subsystem are then lumped together (in *d*) and treated as disturbance. Each subsystem then becomes:

$$m_{ii}(\theta)\ddot{\theta}_{ii} + d_i = \tau_i \qquad (5.6.3)$$

The main objective of the controller design is to control each joint independently in a decentralized fashion and to track the prescribed trajectories. This controller uses an adaptive PID control law given by equation 5.6.4.

$$\tau = K_i + K_P e + K_d \dot{e} + K_f \ddot{\theta}_d \qquad (5.6.4)$$

where *e* is the error given by

$$e = \theta_d - \theta \qquad (5.6.5)$$

Substituting equation 5.6.5 in equation 5.6.3 yields

$$m\ddot{\theta} + d = K_i + K_P e + K_d \dot{e} + K_f \ddot{\theta}_d \qquad (5.6.6)$$

which on simplification gives the state space model of the system as

$$\dot{X} = \begin{bmatrix} \dot{e} \\ \ddot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\dfrac{K_P}{m} & -\dfrac{K_d}{m} \end{bmatrix} \begin{bmatrix} e \\ \dot{e} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{d - K_i}{m} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{m - K_f}{m} \end{bmatrix} \ddot{\theta}_d \qquad (5.6.7)$$

If the error model is defined as

$$\dot{X}_m = \begin{bmatrix} \dot{e}_m \\ \ddot{e}_m \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\xi\omega \end{bmatrix} \begin{bmatrix} e_m \\ \dot{e}_m \end{bmatrix} = AX_m \qquad (5.6.8)$$

then error model tracking error is given by

$$\dot{E} = \begin{bmatrix} \dot{e}_m - \dot{e} \\ \ddot{e}_m - \ddot{e} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -\omega^2 & -2\xi\omega \end{bmatrix} E + \begin{bmatrix} 0 & 0 \\ \dfrac{K_P}{m} - \omega^2 & \dfrac{K_d}{m} - 2\xi\omega \end{bmatrix} X + \begin{bmatrix} 0 \\ \dfrac{K_i - d}{m} \end{bmatrix} + \begin{bmatrix} 0 \\ \dfrac{K_f - m}{m} \end{bmatrix} \ddot{\theta}_d \tag{5.6.9}$$

To ensure the stability of the system given by equation 5.6.9, using the Lyapunov method, the various gains can be adapted as given by equation 5.6.10.

$$\begin{aligned} K_i &= C_0 \int_0^t r dt + f_0 r \\ K_P &= C_1 \int_0^t re dt + f_1 re \\ K_d &= C_2 \int_0^t r\dot{e} dt + f_2 r\dot{e} \\ K_f &= C_3 \int_0^t r\ddot{\theta}_d dt + f_3 r\ddot{\theta}_d \end{aligned} \tag{5.6.10}$$

where

$C_i = \dfrac{m_{ii}}{Q_i}, i = 0,1,2,3$ and Q_i are positive constants

$r = p_2 e + p_3 \dot{e}$ where p_2 and p_3 are positive constants

and

$$\begin{aligned} f_0 &= k_0 |r|, \quad k_0 > 0 \\ f_1 &= k_1 |r|, \quad k_1 \geq 0 \\ f_2 &= k_2 |r|, \quad k_2 \geq 0 \\ f_3 &= k_3 |r|, \quad k_3 \geq 0 \end{aligned}$$

The block diagram for this controller is shown in Fig.5.6.1.

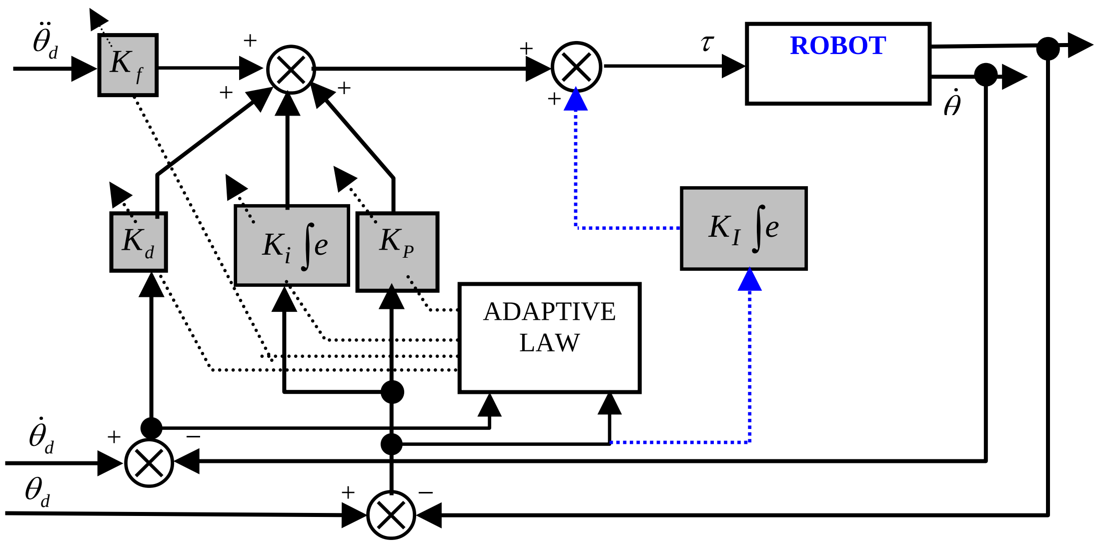


Fig. 5.6.1 Block diagram of Decentralized Adaptive Control

The simulation for DAC was carried out for the two trajectories as for the previous controllers. Fig 5.6.2(a) shows the error profiles for first trajectory, while Fig 5.6.2(b) shows the error profiles for second trajectory. However for DAC there are no cases of warm and cold start, as this controller does not make use of any manipulator model whatsoever. It only tunes the controller gains depending on the current position and velocity errors.

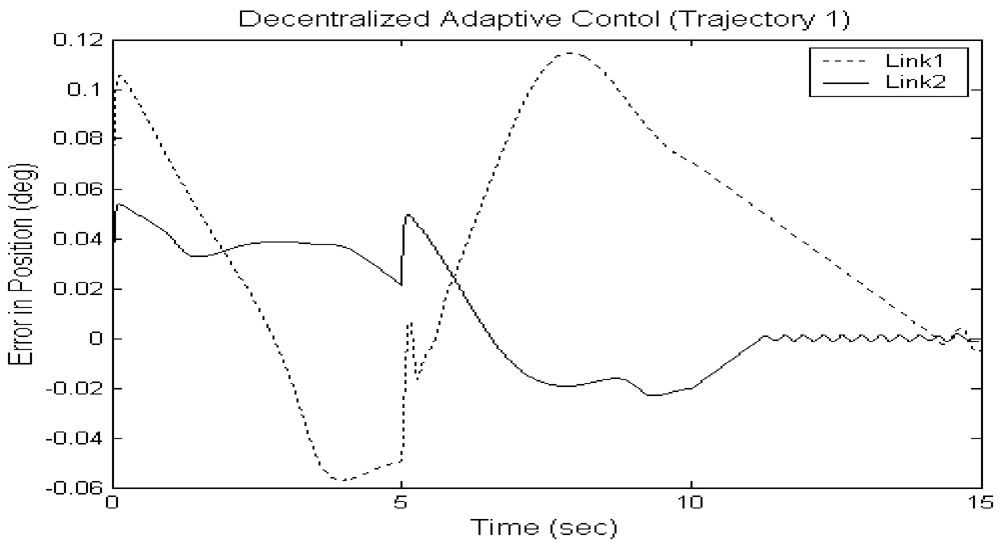


Fig. 5.6.2(a) Errors for DAC, trajectory 1

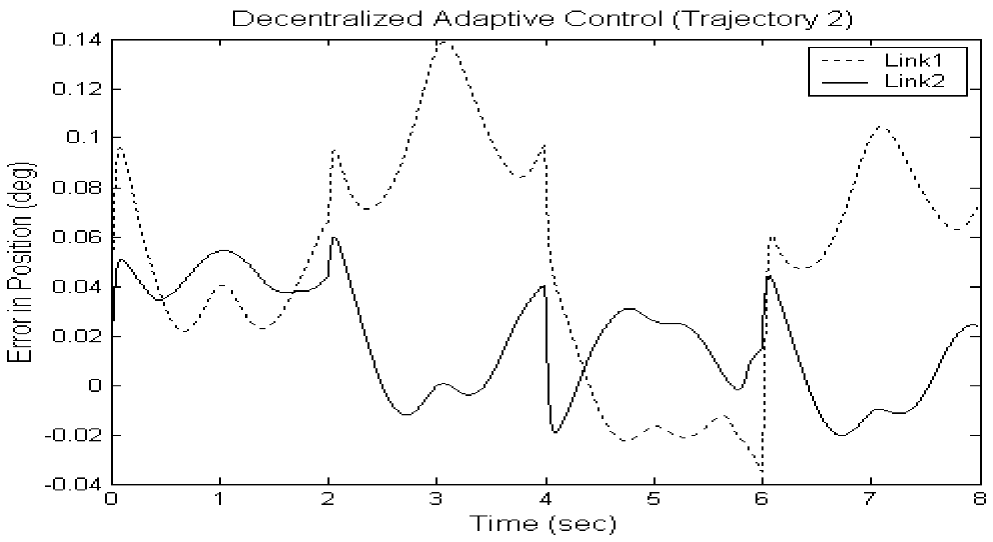


Fig. 5.6.2(b) Errors for DAC, trajectory 2

Decentralized Adaptive Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 $0° \rightarrow 90° \rightarrow 0°$		**Link2** $0° \rightarrow -90° \rightarrow 0°$		**Link1** $0° \rightarrow 45° \rightarrow 0° \rightarrow$		**Link2** $0° \rightarrow -45° \rightarrow 0° \rightarrow$	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.0613	0.0043	0.0259	-0.0007	0.0674	0.1387	0.0284	0.0600

Table 5.6.1 Errors for DAC

Table 5.6.1 summarizes the errors for DAC for the two trajectories. As can be seen from the various errors, the performance of controller is pretty good keeping in mind the fact that no model is being used. Because this controller uses no model, the amount of calculations to be performed in the control loop is considerably reduced. The performance of this controller is better than MRAC but not as good as ACDID. For situations where we cannot afford a fast processor or where such a processor is not available, DAC is a viable option.

An additional modified integral error compensation term was introduced in this controller to see if it provides improved trajectory tracking as in the case of previous controllers. Figures 5.6.3(a) and 5.6.3(b) show the error profiles for the two trajectories for DAC with integral error compensation.

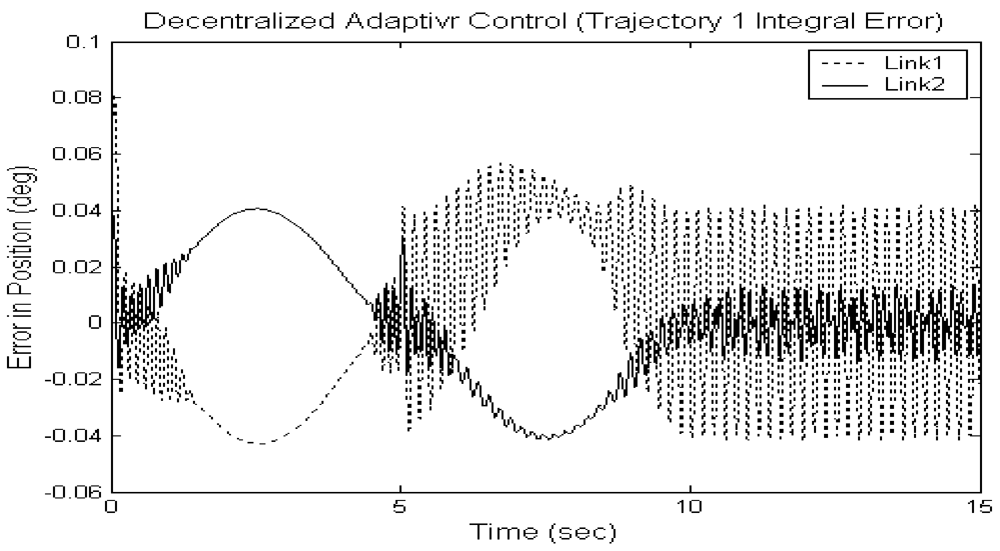


Fig. 5.6.3(a) Errors for DAC, trajectory 1 (Integral error)

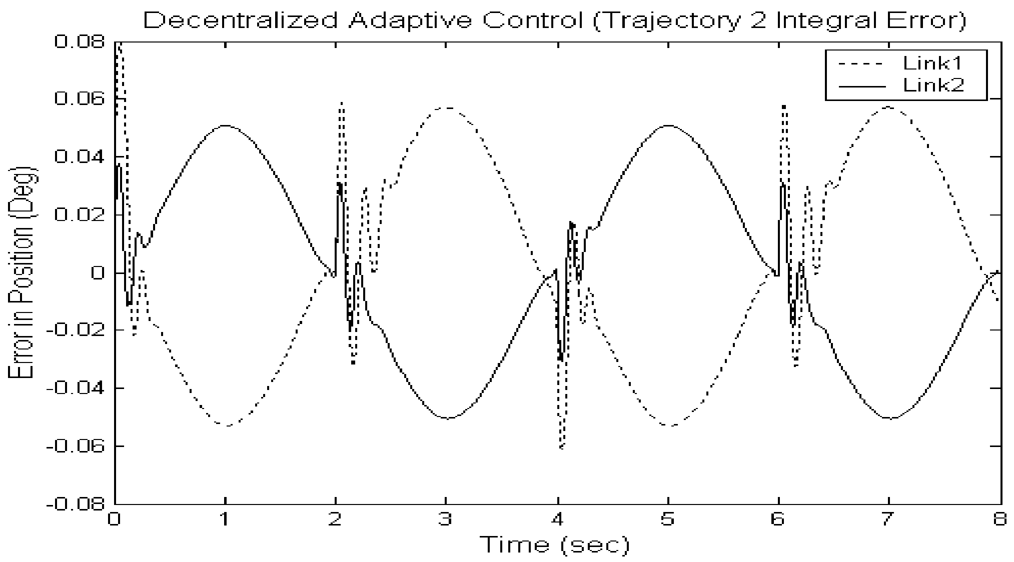


Fig. 5.6.3(b) Errors for DAC, trajectory 2 (Integral error)

Integral Decentralized Adaptive Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.0313	0.0418	0.0220	0.0117	0.0371	0.0798	0.0329	0.0507

Table 5.6.2 Errors for DAC (Integral Error)

Table 5.6.2 list the values of various errors for the two trajectories for DAC with integral error compensation. As can be seen from Fig 5.6.3(a), the introduction of integral error compensation does bring down the trajectory tracking errors but it also introduces oscillations in the system. As these oscillations do not grow in amplitude, because the controller is stable, and as the magnitude of these oscillations is very small, integral error compensation can still be used with this controller.

5.7 COMPARISON OF PERFORMANCE

The consolidated results for the simulations carried out in section 4.3, 4.4, 5.4, 5.5 and 5.6 are presented in Table 5.7.1 for easy comparison of the performance of various controllers.

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
CONVENTIONAL CONTROLLERS									
1.	PD Control	7.1415	9.4447	2.3758	2.6345	6.8624	9.5918	2.1554	3.3037
2.	PID control	2.9360	3.8290	0.9742	1.0737	2.8617	4.1457	0.9264	1.4036
3.	CT	6.1340	8.5617	4.6260	-4.7008	4.9007	12.2731	4.4288	5.0733
4.	FFID	2.5940	3.6395	0.8689	1.1186	2.0083	4.1268	0.6431	1.0846
5.	CDID	0.0616	0.0695	0.0211	0.0214	0.0707	0.1244	0.0265	0.0509
6.	CT+IE*	1.2919	1.7460	0.4019	0.4721	1.1894	2.3245	0.3118	0.6798
7.	FFID+IE*	1.0687	1.4588	0.3508	0.4488	0.9466	1.8639	0.2759	0.4419
8.	CDID+IE*	0.0604	0.0676	0.0210	0.0208	0.0697	0.1226	0.0268	0.0513
ADAPTIVE CONTROLLERS									
9.	ACDID	0.0247	0.0001	0.0238	-0.0002	0.0391	0.0772	0.0344	0.0642
10.	ACDID+IE*	0.0213	0.0	0.0212	0.0	0.0323	0.0508	0.0325	0.0745
11.	MRAC	0.9283	1.3691	0.2827	0.4039	0.6935	1.4323	0.2628	0.5390
12.	MRAC+IE*	0.0321	0.0	0.0200	0.0	0.0559	0.1800	0.0270	0.0732
13.	DAC	0.0613	0.0043	0.0259	-0.0007	0.0674	0.1387	0.0284	0.0600
14.	DAC+IE*	0.0313	0.0418	0.0220	0.0117	0.0371	0.0798	0.0329	0.0507

* Integral Error Compensation (modified)

Table 5.7.1. Errors (in degrees) for different controllers

Following observations are made based on simulations carried out in this section:

1. The adaptive controllers give better performance than the conventional controllers studied in chapter 4. This is mainly because of the learning capability of these controllers. These controllers can learn the unknown or the

changed parameter values as they execute the trajectory. Alternatively they can adjust the controller gains depending on the current system errors.

2. The adaptive controllers also perform better than conventional controllers with modified integral error compensation.
3. Model based adaptive controllers perform better for the warm start case than for the cold start case. Thus it is advisable to use whatever knowledge one may have about the manipulator parameter values at the start of motion. This knowledge may be a rough estimate of the actual values.
4. Addition of modified integral error compensation to the adaptive controllers further improves their performance. The steady state errors for these controllers are almost zero, while the maximum and RMS values of errors are also reduced.
5. Model Based Adaptive controllers are computationally expensive. The DAC discussed above is computationally least expensive of all the adaptive controllers studied.
6. The ACDID controller with modified integral error compensation gives the best performance amongst all controllers investigated in this chapter and in chapter 4. This is indicated by the shaded cells in Table 5.7.1.

5.8 CONCLUDING REMARKS

In this chapter we studied the efficacy of adaptive algorithms for manipulator control. Both model based and non-model based adaptive controllers were investigated. It was seen that all the adaptive controllers outperform the conventional controllers.

The model based adaptive controllers were also tested for two different cases. In first case it was assumed that a rough estimate of parameters was available while in second case it was assumed that no such estimate is available.

Further, the effect of inclusion of a modified integral compensation to these adaptive controllers was studied. The integral action is such that it maintains the stability of the controller. It was seen that this integral action further improves the performance of these controllers.

Although the adaptive controllers give a very good performance, it comes at the price of computational complexity. All these controllers are computationally intensive and require a fast processor for practical implementation. We thus need to investigate other control schemes, which can give comparable results at lesser computational expense. One such scheme is the Fuzzy control. We study the fuzzy controller in detail in the next chapter.

FUZZY CONTROL OF ROBOT MANIPULATORS

6.0 INTRODUCTION

Fuzzy control of robotic manipulators has found vast interest in the control literature. Unlike Boolean logic, fuzzy logic deals with concepts of vagueness, uncertainty or imprecision. It provides an extensive freedom for control designers to exploit their understanding of the problem and to construct intelligent control strategies [Bonissone and Chiang (1993), Ken et al. (1988)]. Nonlinear controllers can be devised easily by using fuzzy logic principles [Zhou and Coiffet (1992)]. This makes fuzzy controllers powerful tools to deal with nonlinear systems [Chun Fei and Chin-Teng (2004), Mamdani (1993)].

The fuzzy control strategy consists of situation and action pairs, similar to how a human operator uses his experience to interpret the situation and initiate the control action. A human operator usually looks at the error and the change of error so as to arrive at a particular control action. A block diagram for the fuzzy controller is shown in Fig.6.1. The fuzzy controller here defines error (e) as

$$e = \theta_d - \theta \tag{6.1}$$

and rate of change of error ($\dot{e}$) as

$$\dot{e} = \dot{\theta}_d - \dot{\theta} \tag{6.2}$$

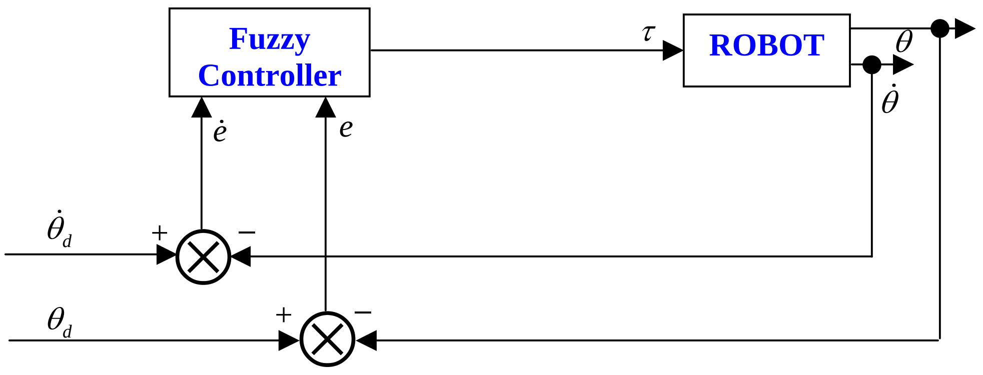


Fig. 6.1. Block diagram of Fuzzy Controller

τ is the output of fuzzy controller applied as control input to the robot system.

A detailed view of internal of the Fuzzy controller block shown in Fig. 6.1 is shown in Fig.6.2.

The input variables to the fuzzy controller (e, $\dot{e}$) are quantized into thirteen levels represented by –6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, and a set of linguistic variables such as Negative Big (NB), Negative Medium (NM), Negative Small (NS)**,** Zero (ZE), Positive Small (PS), Positive Medium (PM), Positive Big (PB) are assigned.

The next step in the design of the fuzzy controller is to decide the membership functions for the linguistic variables. The decision regarding the type of the membership function is arbitrary and depends on the choice of the user. Here, we have selected the triangular membership function as shown in Fig. 6.3. The control rules are formulated in a manner to represent the operator's experience regarding the system behavior [Dubois and Prade (1996)].

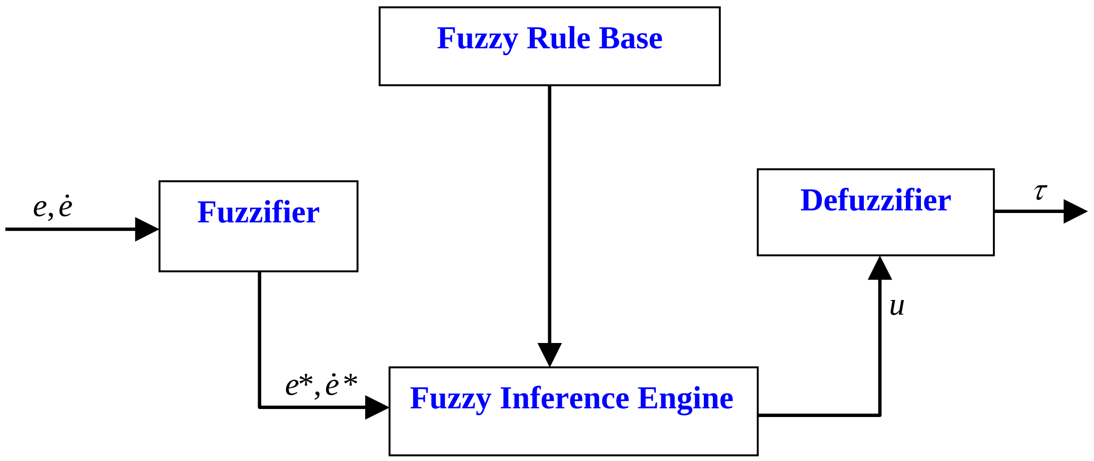


Fig. 6.2. Details of Fuzzy controller block

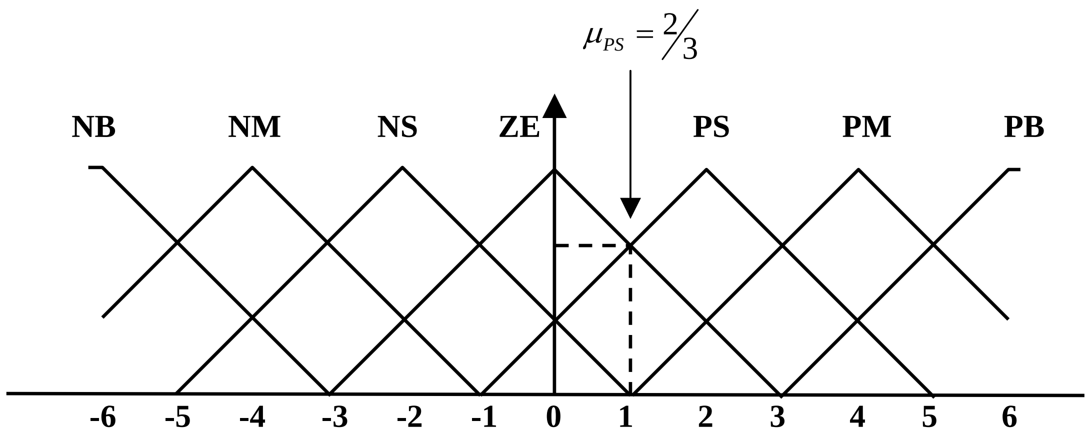


Fig. 6.3. Membership functions of the Linguistic variables

Some of the rules that were formulated are

R1: If e is ZE and $\dot{e}$ is ZE, then u is ZE.

R2**:** If e is ZE and $\dot{e}$ is NS**,** then u is NS.

R3: If e is NM and $\dot{e}$ is ZE, then u is NM**.**

R4: If e is NM and $\dot{e}$ is NB**,** then u is NB.

These rules constitute the knowledge base of the fuzzy controller [Nagrath et al. (1995)]. The rule strength of the individual rule is evaluated using the intersection operation defined as

$$\mu_{NB}(u) = \min(\mu_{NM}(e^*), \mu_{NB}(\dot{e}^*)) \qquad (6.3)$$

where $\mu_{NB}(u)$ is the rule strength of the rule R4, $\mu_{NM}(e^*)$ is the membership of the crisp input e^* in the fuzzy set NM and $\mu_{NB}(\dot{e}^*)$ is the membership of $\dot{e}^*$ in the fuzzy set NB. For each possible combination of e^* and $\dot{e}^*$, the rules are fired individually to give the degree to which the rule antecedent has been matched by the crisp value. The clipped values for the individual rules thus obtained are aggregated forming the overall control values. The output value is then defuzzified by using the center of gravity method, which, for the discrete case, is given by

$$u^* = \frac{\sum_{Ri} \mu_{Ri}(u_{Ri}).u_{Ri}}{\sum_{Ri} \mu_{Ri}(u_{Ri})} \qquad (6.4)$$

The output values thus obtained for all the $(e^*,\ \dot{e}^*)$ pairs are stored in the form of a lookup table (LUT) as shown in Table 6.1.

The array implementation improves execution speed, as the run-time inference is reduced to a table look-up which is a lot faster, at least when the correct entry can be found without too much searching [Albertos et al. (2000)].

$\dot{e}$ \ e	**Membership Function**												
	-6	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
-6	-5.6	-5.4	-5.0	-4.8	-4.8	-4.7	-4.7	-4.6	-4.5	-4.4	-4.3	-4.3	-4.2
-5	-4.7	-4.5	-4.4	-4.3	-4.2	-4.1	-4.0	-3.9	-3.8	-3.8	-3.7	-3.6	-3.5
-4	-3.7	-3.6	-3.5	-3.2	-3.0	-3.0	-3.0	-2.9	-2.9	-2.8	-2.8	-2.7	-2.7
-3	-2.0	-2.0	-1.9	-1.9	-1.8	-1.8	-1.7	-1.7	-1.6	-1.5	-1.4	-1.3	-1.3
-2	0.0	0.0	-0.8	-1.0	-1.2	-1.7	-2.3	-2.2	-2.2	-2.0	-2.0	-1.0	-1.0
-1	1.0	1.0	0.0	0.0	-0.5	-0.5	-0.5	-1.0	-1.2	-1.5	-1.7	-1.0	-1.0
0	1.3	1.2	1.0	0.8	0.6	0.0	-0.2	-0.4	-0.6	-0.8	-1.0	-1.0	-1.0
1	2.0	2.0	1.9	1.8	1.8	1.8	1.8	1.8	1.5	0.0	-0.3	-1.0	-0.8
2	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	2.0	1.2	0.8	0.0	0.0
3	2.0	2.1	2.3	2.5	2.5	2.5	2.6	2.7	2.8	2.8	2.9	2.9	3.0
4	2.7	2.7	2.8	3.1	3.2	3.3	3.5	3.6	3.6	3.8	3.8	3.9	3.9
5	3.6	3.3	3.7	4.0	4.1	4.3	4.3	4.4	4.4	4.5	4.5	4.6	4.7
6	4.4	4.4	4.3	4.8	5.0	5.0	5.1	5.2	5.3	5.4	5.6	5.6	5.6

Table 6.1. Lookup Table for the Fuzzy Controller

The controller output values shown in the Table 6.1 were obtained after some manual adjustment through trial and error to give best possible results. This was required because the manipulator control problem is highly nonlinear and the rules formulated through user experience are not always correct under different situations.

6.1. PURE FUZZY CONTROL

The first investigation that was carried out concerned the performance of fuzzy controller under the circumstance that the manipulator parameters do not change throughout the motion. In other words, the manipulator does not pick up or release any load during its motion. The manipulator parameters were kept same as for all previous simulations and are given in Table 4.3.1.1. The two trajectories used for simulation are same as shown in Fig. 4.3.1.1(a) and Fig. 4.3.1.1(b). The lookup table used for fuzzy controller is given in Table 6.1.

The error profiles for the two links are shown in Fig. 6.1.1(a) for the first trajectory, and Fig. 6.1.1(b) for the second trajectory. Table 6.1.1 lists the various error measures magnitudes for the two trajectories. As mentioned earlier these profiles are obtained by using the lookup table given in Table 6.1, which was obtained after some manual adjustments to the original table obtained from the rule base. As can be seen from the error magnitudes, the performance of this controller is pretty good. Except for somewhat large maximum error the RMS values of errors and steady state errors are quite small. The large amount of error in the beginning of a trajectory segment is mainly because the set points are changing rapidly during this time or the manipulator is picking/releasing load. The amount of calculations to be done by this controller is small compared to Model based adaptive or conventional controllers. This means that this controller can be run at higher sampling rates giving even better performance. We have however, in this simulation, kept the sampling rates same as for previous simulations for the sake of 'JUST' comparison.

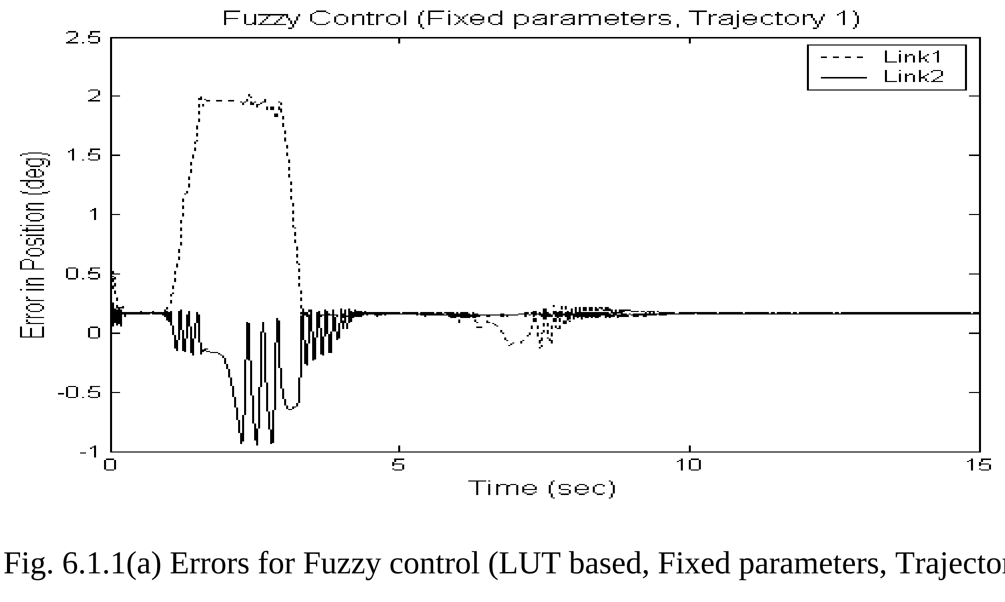


Fig. 6.1.1(a) Errors for Fuzzy control (LUT based, Fixed parameters, Trajectory 1)

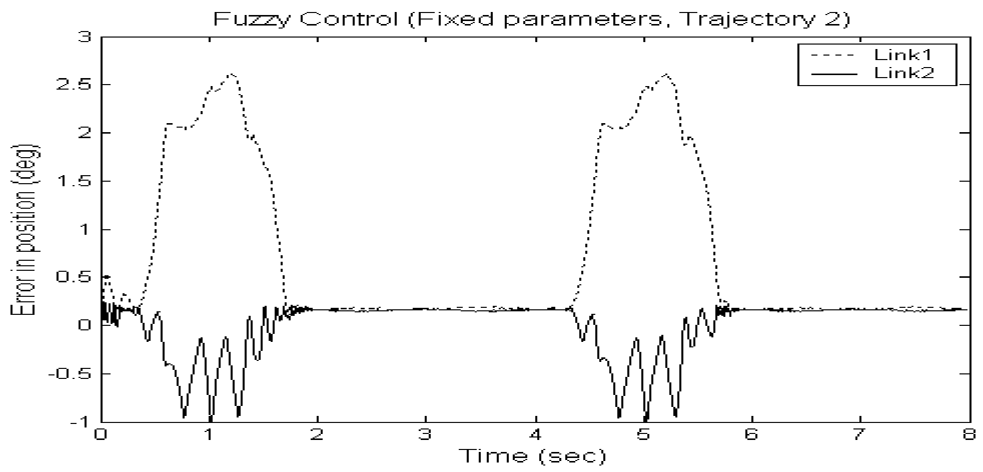


Fig. 6.1.1(b) Errors for Fuzzy control (LUT based, Fixed parameters, Trajectory 2)

Fuzzy Control (Fixed parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.6781	0.1676	0.2270	0.1664	1.1374	2.6153	0.2901	-0.9998

Table 6.1.1. Errors for Fuzzy control (LUT based, Fixed parameters)

The ‘good’ performance of the Fuzzy controller deteriorates considerably if the manipulator parameters change during motion. The two trajectories used to investigate this case are same as before. The only difference is that now in these two trajectories the manipulator picks up and releases load during its motion. This Picking up and releasing of load changes manipulator parameters during motion. The changed parameters of manipulator are listed in the Table 4.3.1.2.

The error profiles for the two links are shown in Fig. 6.1.2(a) for the first trajectory, and Fig. 6.1.2(b) for the second trajectory. Table 6.1.2 lists the magnitudes of various error measures for the two trajectories. It can be seen from Fig. 6.1.2(a) the steady state error for link 1 has increased considerably from the previous value of 0.1676 to 2.0 degrees. This also results in larger RMS value of error. From Fig. 6.1.2(b) we notice a similar increase in errors for the second trajectory. This is mainly due to the fact that the lookup table used for control is not optimally tuned for the new values of manipulator parameters.

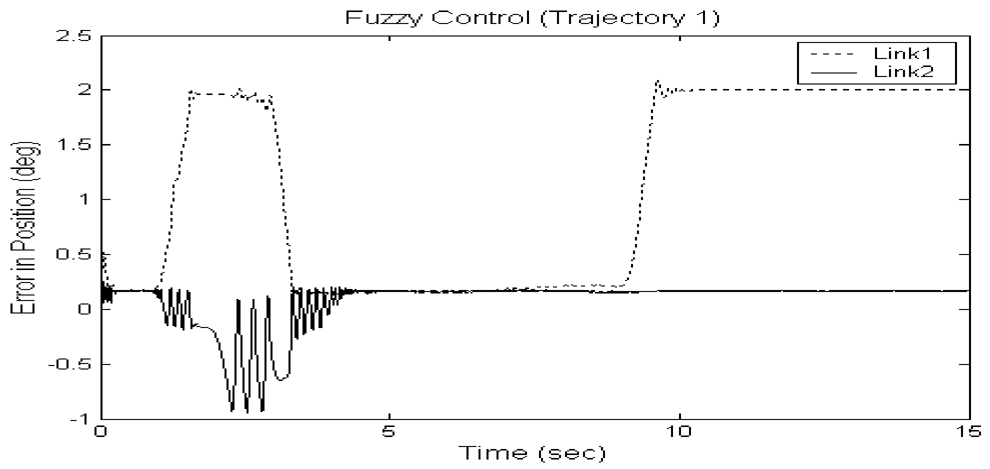


Fig. 6.1.2(a) Errors for Fuzzy control (LUT based, Changing parameters, Trajectory 1)

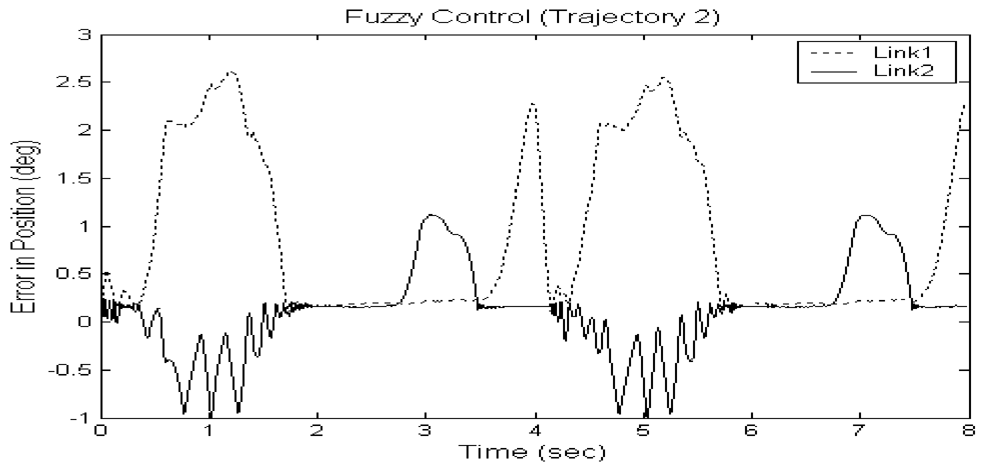


Fig. 6.1.2(b) Errors for Fuzzy control (LUT based, Changing parameters, Trajectory 2)

Pure Fuzzy Control Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183

Table 6.1.2. Errors for Fuzzy control (LUT based, Changing parameters)

This problem of having to retune the Fuzzy lookup table every time the trajectory changes or manipulator parameters change can be solved by primarily by two methods. These two methods are Adaptive Fuzzy and Self Organizing Fuzzy control methods. These methods will be discussed in later sections. The problem can also be alleviated by use of Hybrid fuzzy controllers, which we would investigate next.

6.2 HYBRID FUZZY CONTROL

In this section we propose and investigate some new hybrid fuzzy control schemes. The primary characteristic of these controllers is that in these schemes the final control output applied to the plant is summation of individual output of two controllers. One of them is the Fuzzy controller while the other could be a Conventional or Adaptive controller [Butkiewicz (2000), Chin and Er (1998)]. The general block diagram of the controller is shown in Fig. 6.2.1. As both the controllers are individually stable, the combination is also stable. We first discuss the results of combining Fuzzy and Conventional controllers and then Fuzzy and Adaptive controllers.

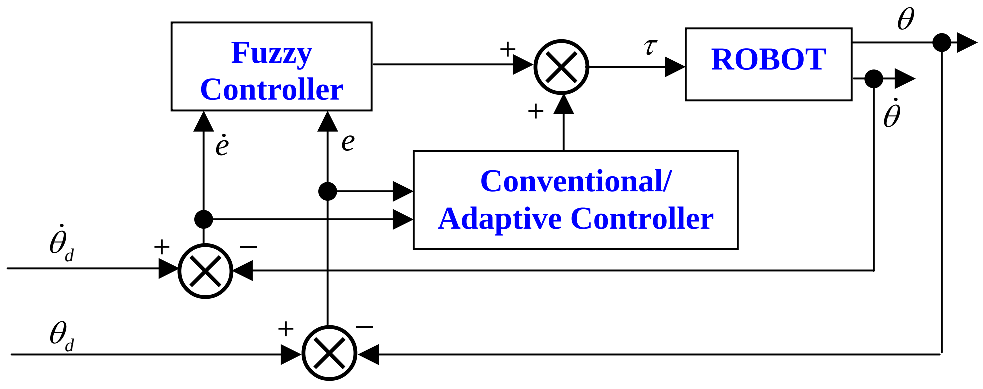


Fig. 6.2.1. Block diagram of Hybrid Fuzzy Controller

All the controllers discussed in the following subsections were tested for the case when the parameters of the manipulator change during motion. The two trajectories used to investigate the controllers are shown in Fig. 4.3.1.1(a) and Fig. 4.3.1.1(b). In these two trajectories the manipulator picks up and releases load during its motion. This Picking up and releasing of load changes manipulator parameters during motion. The changed parameters of manipulator are listed in the Table 4.3.1.2.

6.2.1 FUZZY PLUS COMPUTED TORQUE CONTROLLER

The Computed Torque controller discussed in section 4.2.1 is combined with the Fuzzy controller in this scheme. Fig. 6.2.1.1 shows the block diagram of this controller. The error profiles for the two links are shown in Fig. 6.2.1.2(a) for the first trajectory, and Fig. 6.2.1.2(b) for the second trajectory. Table 6.2.1.1 lists the magnitudes of various error measures for the two trajectories for this controller and some other related controllers. As can be seen from the table, this controller performs much better than Pure Fuzzy, CT and CT+IE controllers. Both the RMS and steady state values of the errors have reduced considerably for this controller when compared to the other controllers. Even the maximum values of errors have reduced considerably for the second trajectory.

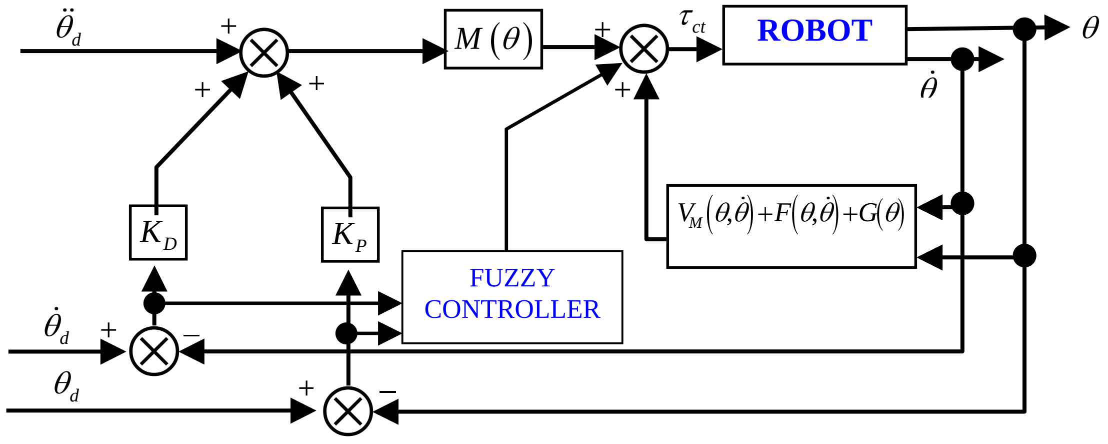


Fig. 6.2.1.1 Block diagram of Fuzzy + Computed Torque Controller

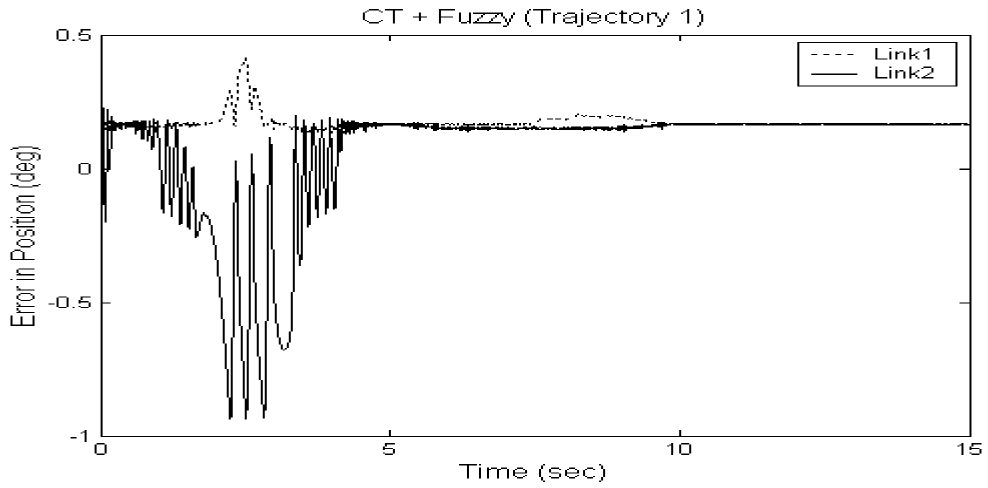


Fig. 6.2.1.2(a) Errors for CT + Fuzzy control (Trajectory 1)

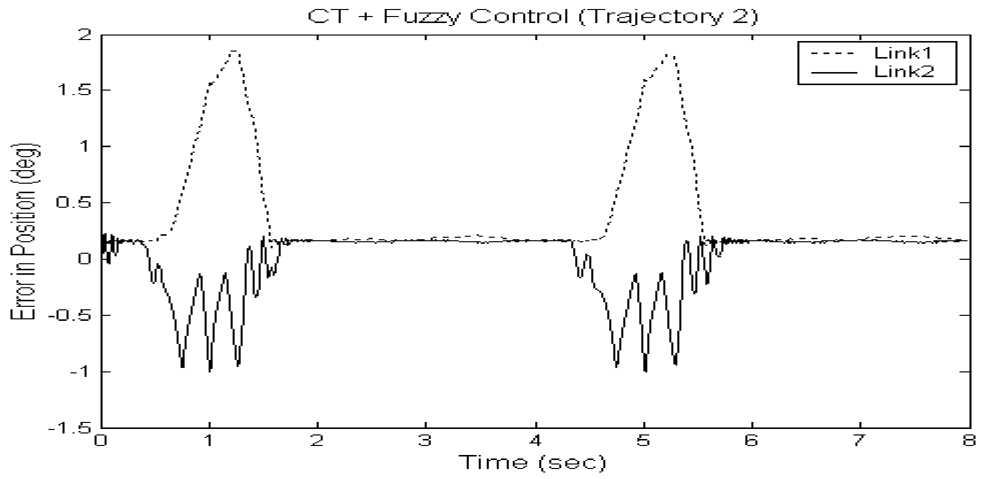


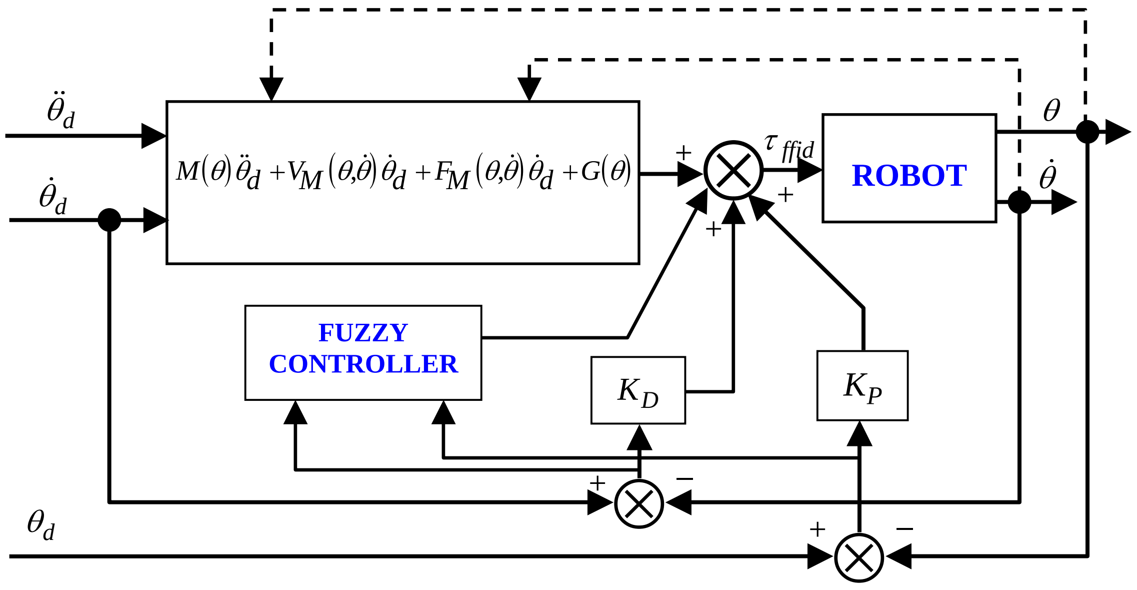
Fig. 6.2.1.2(b) Errors for CT + Fuzzy control (Trajectory 2)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
2.	CT	6.1340	8.5617	4.6260	-4.7008	4.9007	12.2731	4.4288	5.0733
3.	CT+IE	1.2919	1.7460	0.4019	0.4721	1.1894	2.3245	0.3118	0.6798
4.	CT+Fuzzy	0.1761	0.1670	0.2361	0.1667	0.6324	1.8571	0.2895	-1.0030

Table 6.2.1.1. Comparison of Errors for CT + Fuzzy control

6.2.2 FUZZY PLUS FFID CONTROLLER

The FFID controller discussed in section 4.2.2 is combined with the Fuzzy controller in this scheme. Fig. 6.2.2.1 shows the block diagram of this controller. The error profiles for the two links are shown in Fig. 6.2.2.2(a) for the first trajectory, and Fig. 6.2.2.2(b) for the second trajectory. Table 6.2.2.1 lists the various error measures magnitudes for the two trajectories for this controller and some other related controllers. As can be seen from the table, this controller performs much better than Pure Fuzzy, FFID and FFID+IE controllers.



$\ddot{\theta}_d$
$\dot{\theta}_d$
$M(\theta)\ddot{\theta}_d + V_M(\theta,\dot{\theta})\dot{\theta}_d + F_M(\theta,\dot{\theta})\dot{\theta}_d + G(\theta)$
τ_{ffid}
ROBOT
θ
$\dot{\theta}$
FUZZY CONTROLLER
K_D
K_P
θ_d
+
−

Fig. 6.2.2.1 Block diagram of Fuzzy + FFID Controller

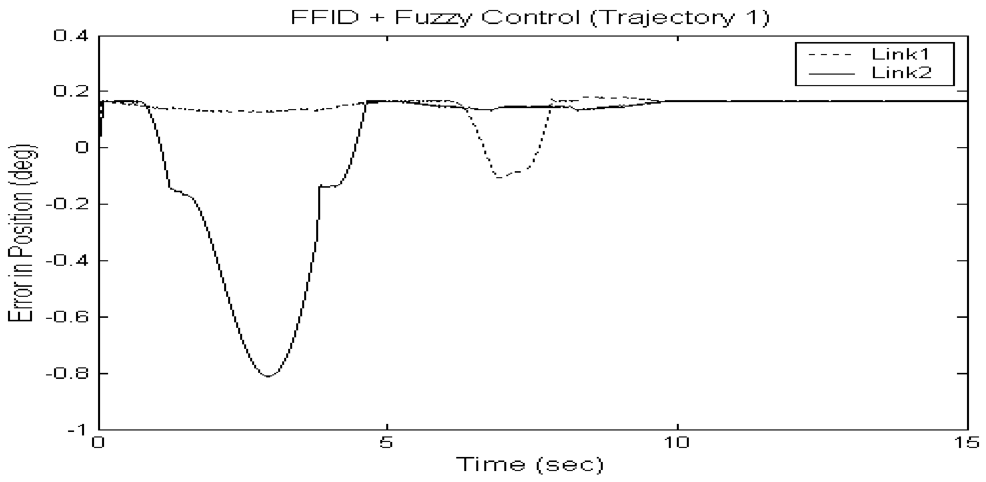


Fig. 6.2.2.2(a) Errors for FFID + Fuzzy control (Trajectory 1)

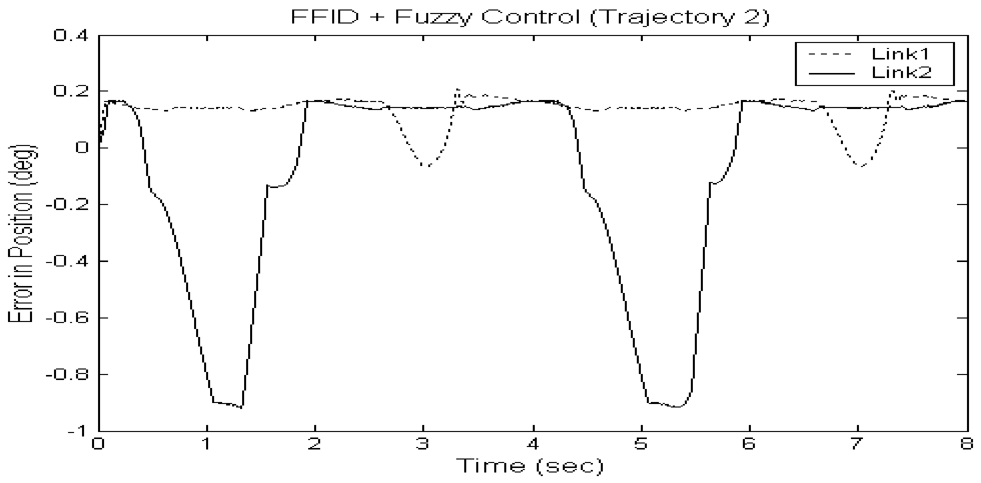


Fig. 6.2.2.2(b) Errors for FFID + Fuzzy control (Trajectory 2)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
2.	FFID	2.5940	3.6395	0.8689	1.1186	2.0083	4.1268	0.6431	1.0846
3.	FFID+IE*	1.0687	1.4588	0.3508	0.4488	0.9466	1.8639	0.2759	0.4419
4.	FFID+Fuzzy	0.1534	0.1667	0.2716	0.1663	0.1471	0.2104	0.3702	-0.9186

Table 6.2.2.1. Comparison of Errors for FFID + Fuzzy control

6.2.3 FUZZY PLUS CDID CONTROLLER

The CDID controller discussed in section 4.2.3 is combined with the Fuzzy controller in this scheme. Fig. 6.2.3.1 shows the block diagram of this controller. The error profiles for the two links are shown in Fig. 6.2.3.2(a) for the first trajectory, and Fig. 6.2.3.2(b) for the second trajectory. Table 6.2.3.1 lists the magnitudes of various error measures for the two trajectories for this controller and some other related controllers. It is seen from the table, that this controller performs better than Pure Fuzzy, CDID and CDID+IE controllers.

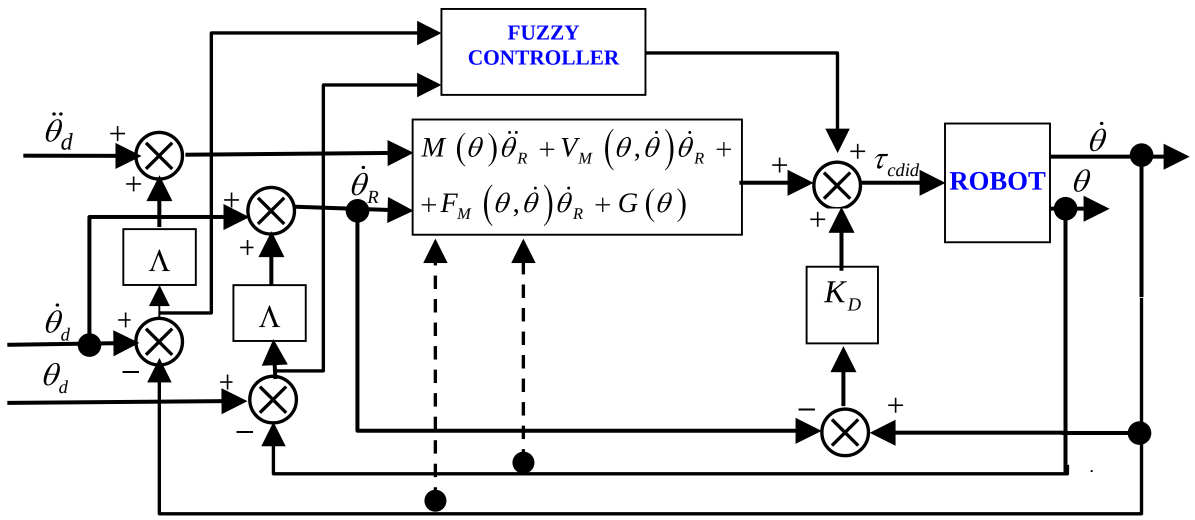


Fig. 6.2.3.1 Block diagram of Fuzzy + CDID Controller

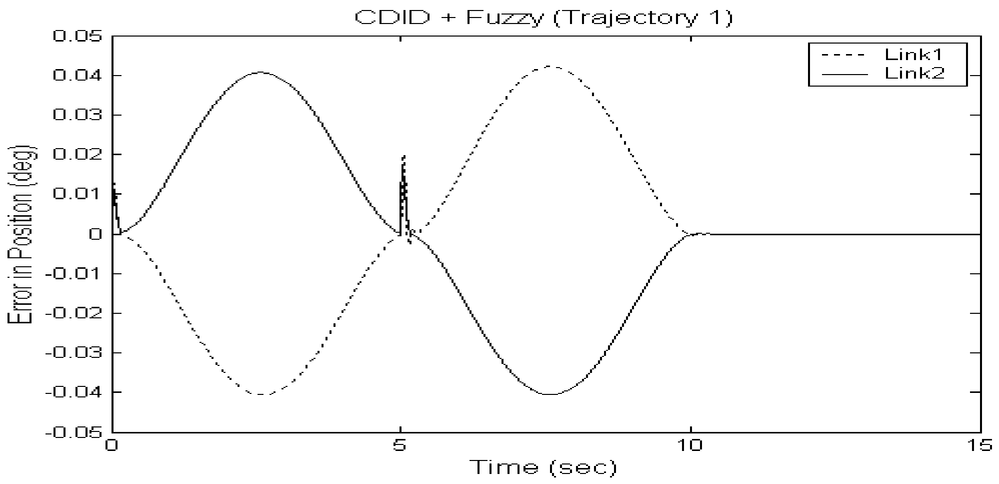


Fig. 6.2.3.2(a) Errors for CDID + Fuzzy control (Trajectory 1)

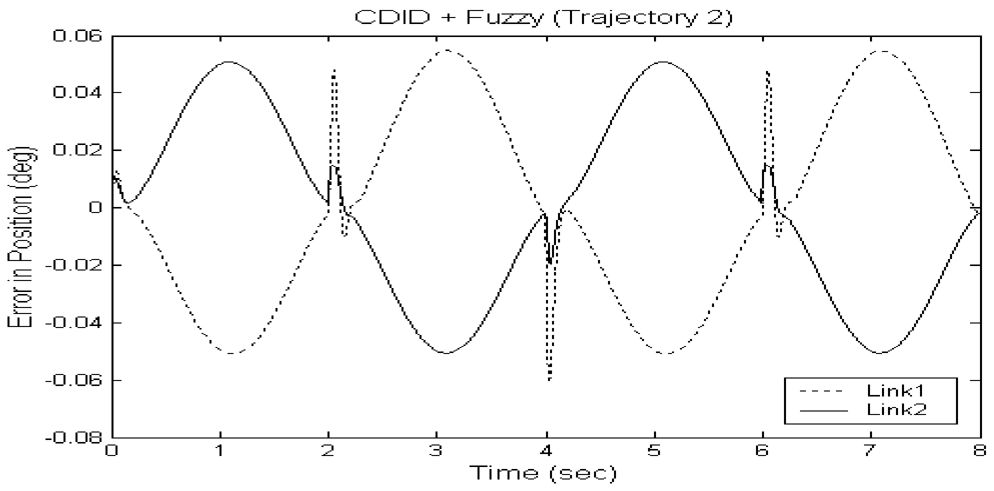


Fig. 6.2.3.2(b) Errors for CDID + Fuzzy control (Trajectory 2)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
2.	CDID	0.0616	0.0695	0.0211	0.0214	0.0707	0.1244	0.0265	0.0509
3.	CDID+IE	0.0604	0.0676	0.0210	0.0208	0.0697	0.1226	0.0268	0.0513
4.	FFID+Fuzzy	0.0216	0.0	0.0212	0.0	0.0343	-0.0610	0.0325	0.0507

Table 6.2.3.1. Comparison of Errors for CDID + Fuzzy control

6.2.4 FUZZY PLUS ACDID CONTROLLER

Adaptive controllers can also be combined with fuzzy to give a hybrid controller [Lin and Mon (2003), Hojati and Gazor (2002)]. The ACDID controller discussed in section 5.4 is combined with the Fuzzy controller in this scheme. Fig. 6.2.4.1 shows the block diagram of this controller. The error profiles for the two links are shown in Fig. 6.2.4.2(a) for the first trajectory, and Fig. 6.2.4.2(b) for the second trajectory. Table 6.2.4.1 lists the various error measures magnitudes for the two trajectories for this controller and some other related controllers. As can be seen from the table, this controller performs almost same as ACDID+IE controller with slightly lower errors.

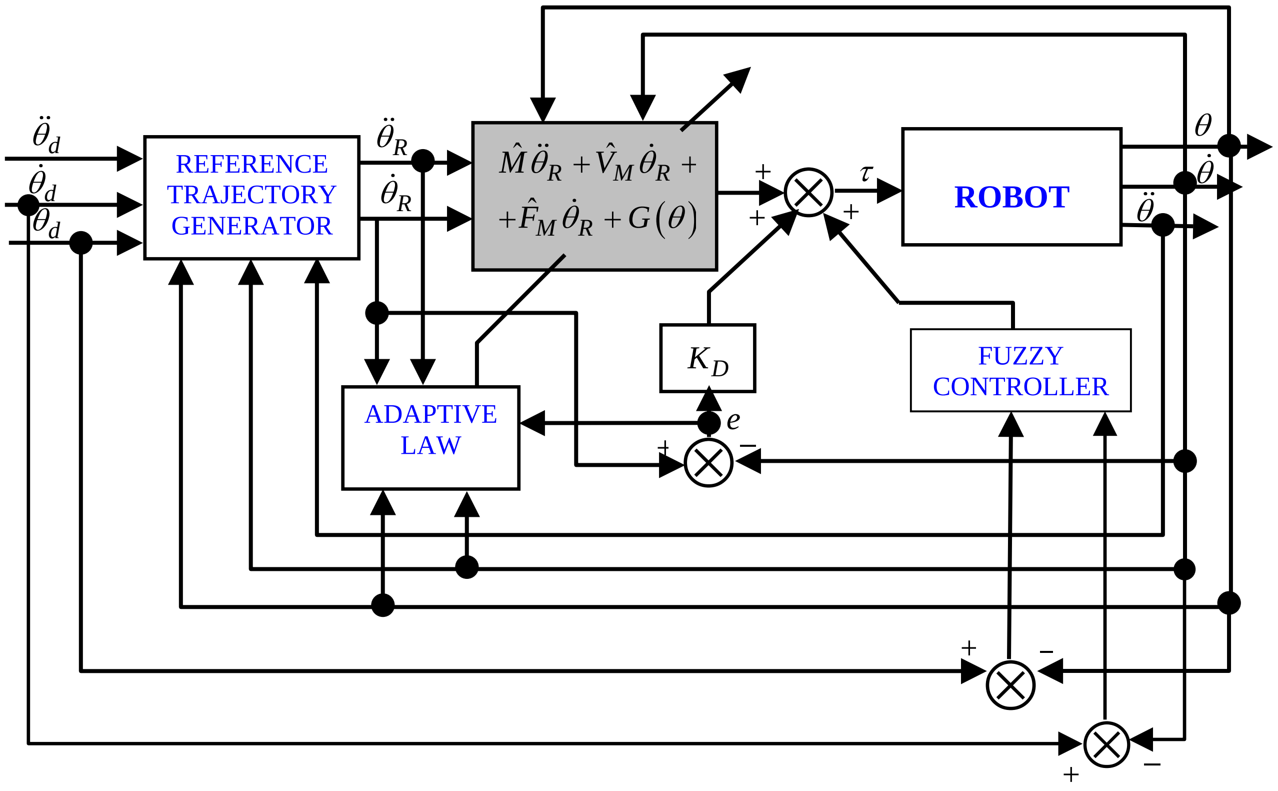


Fig. 6.2.4.1 Block diagram of Fuzzy + ACDID Controller

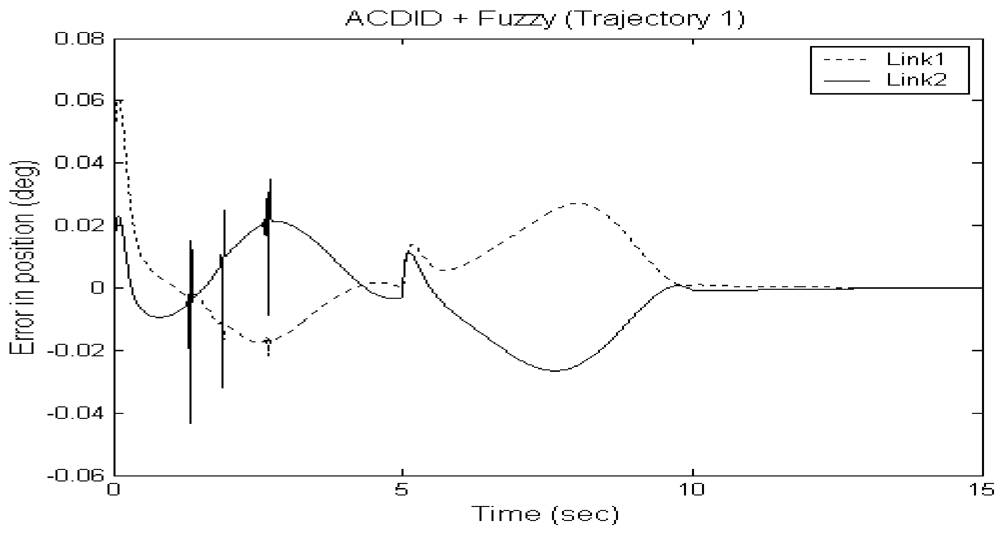


Fig. 6.2.4.2(a) Errors for ACDID + Fuzzy control (Trajectory 1)

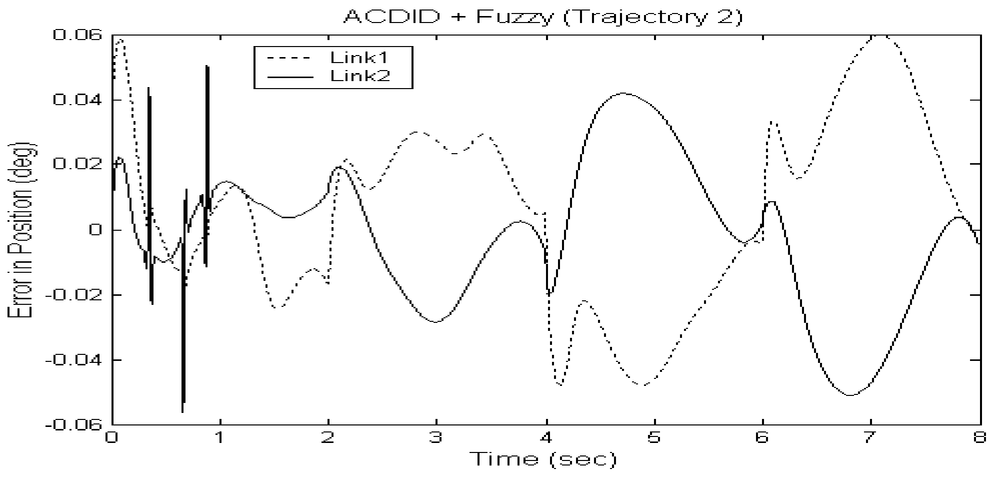


Fig. 6.2.4.2(b) Errors for ACDID + Fuzzy control (Trajectory 2)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
2.	ACDID	0.0247	0.0001	0.0238	-0.0002	0.0391	0.0772	0.0344	0.0642
3.	ACDID+IE	0.0213	0.0	0.0212	0.0	0.0323	0.0508	0.0325	0.0745
4.	ACDID+Fuzzy	0.0130	0.0	0.0119	-0.0001	0.0291	0.0599	0.0226	-0.0564

Table 6.2.4.1. Comparison of Errors for ACDID + Fuzzy control

6.2.5 FUZZY PLUS MRAC

Some work has been done on hybrid MRAC Fuzzy controllers by researchers [Jen-Yang (2002). The MRAC discussed in section 5.5 is combined with the Fuzzy controller in this scheme. Fig. 6.2.5.1 shows the block diagram of this controller. The error profiles for the two links are shown in Fig. 6.2.5.2(a) for the first trajectory, and Fig. 6.2.5.2(b) for the second trajectory. Table 6.2.5.1 lists the magnitudes of various error measures for the two trajectories for this controller and some other related controllers. As can be seen from the table, this controller performs much better than Pure Fuzzy, MRAC but not as good as MRAC+IE controller.

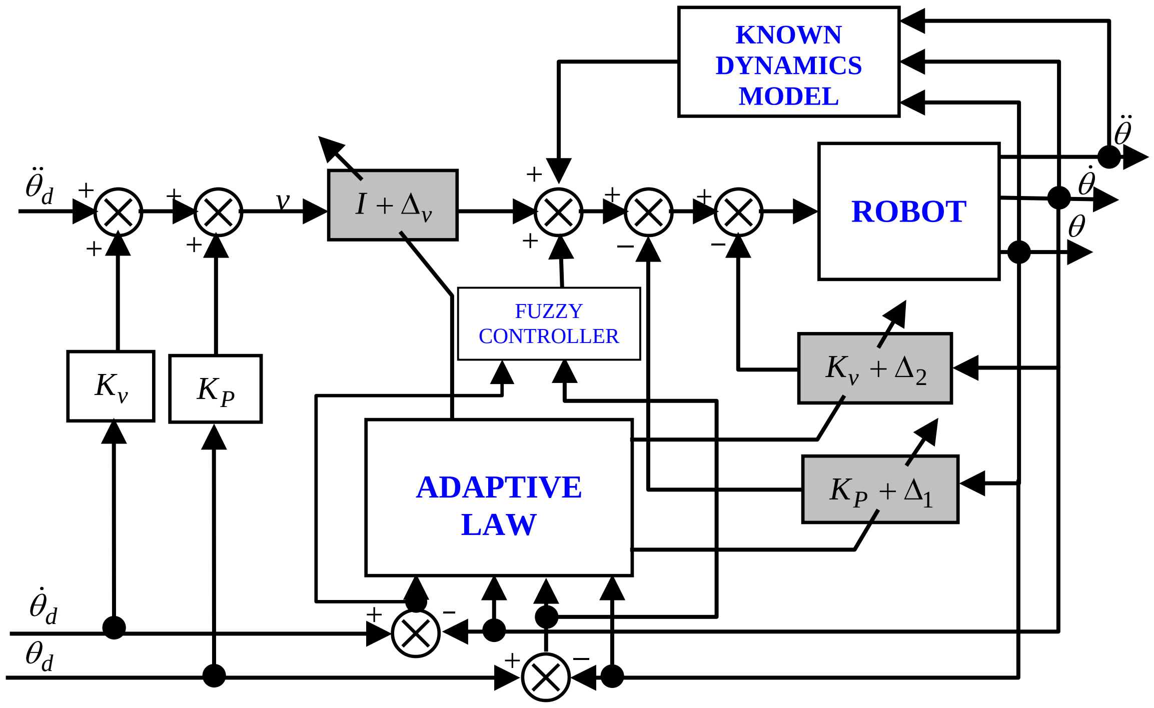


Fig. 6.2.5.1 Block diagram of MRAC + Fuzzy Controller

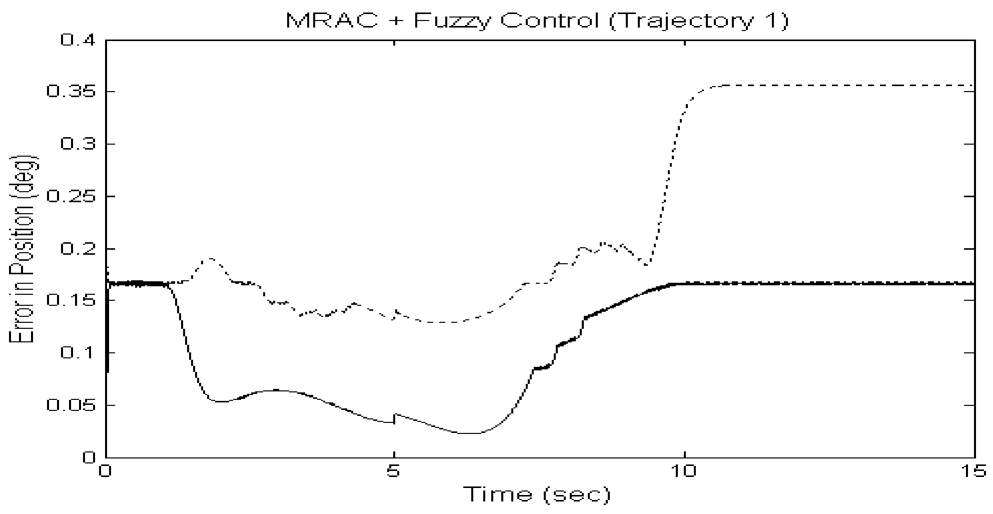


Fig. 6.2.5.2(a) Errors for MRAC + Fuzzy control (Trajectory 1)

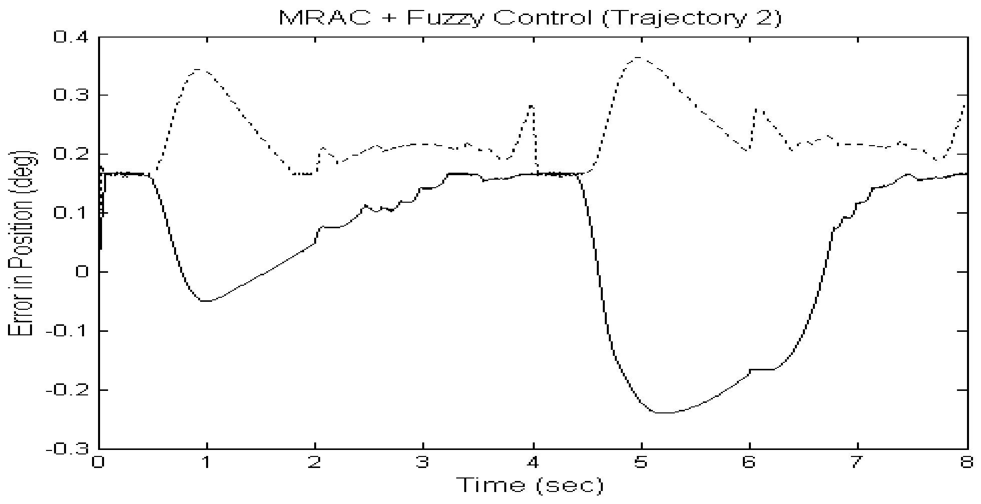


Fig. 6.2.5.2(b) Errors for MRAC + Fuzzy control (Trajectory 2)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
2.	MRAC	0.9283	1.3691	0.2827	0.4039	0.6935	1.4323	0.2628	0.5390
3.	MRAC+IE	0.0321	0.0	0.0200	0.0	0.0559	0.1800	0.0270	0.0732
4.	MRAC+Fuzzy	0.2472	0.3565	0.1261	0.1673	0.2350	0.3648	0.1423	-0.2403

Table 6.2.5.1. Comparison of Errors for MRAC + Fuzzy control

6.2.6 FUZZY PLUS DAC

The DAC discussed in section 5.6 is combined with the Fuzzy controller in this scheme. Fig. 6.2.6.1 shows the block diagram of this controller. The error profiles for the two links are shown in Fig. 6.2.6.2(a) for the first trajectory, and Fig. 6.2.6.2(b) for the second trajectory. Table 6.2.6.1 lists the magnitudes of various error measures for the two trajectories for this controller and some other related controllers. It is seen from the table, that there is no marked improvement in performance of this controller over DAC and DAC+IE controllers.

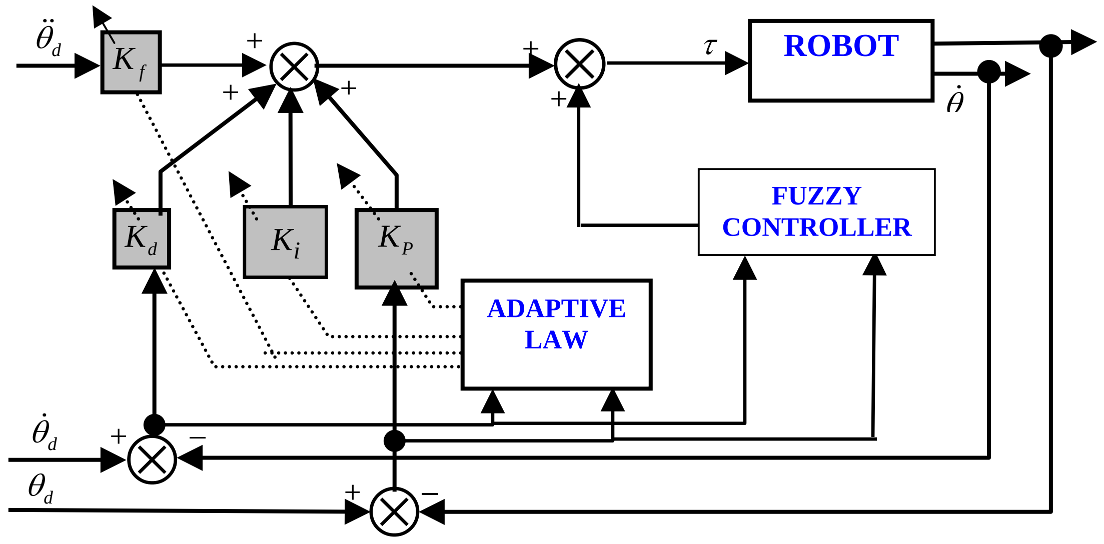


Fig. 6.2.6.1 Block diagram of DAC + Fuzzy Controller

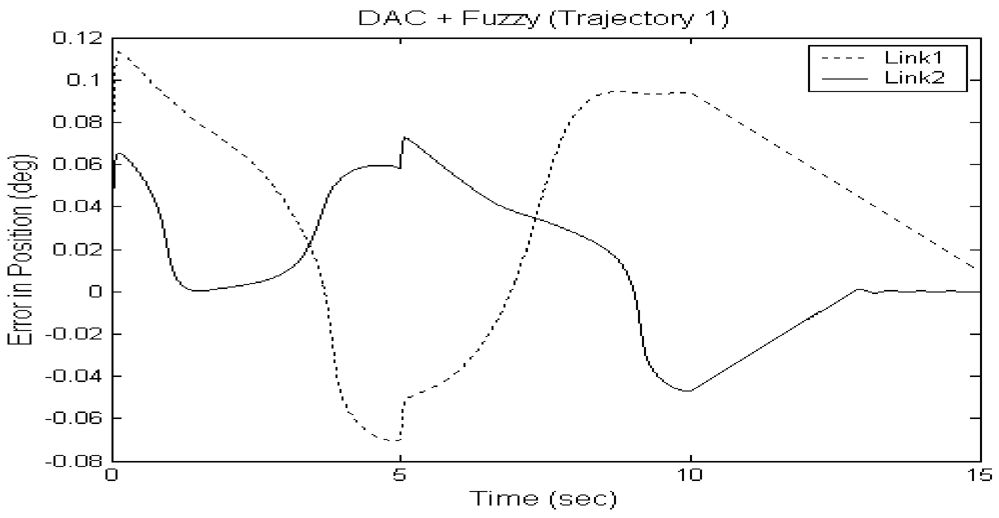


Fig. 6.2.6.2(a) Errors for DAC + Fuzzy control (Trajectory 1)

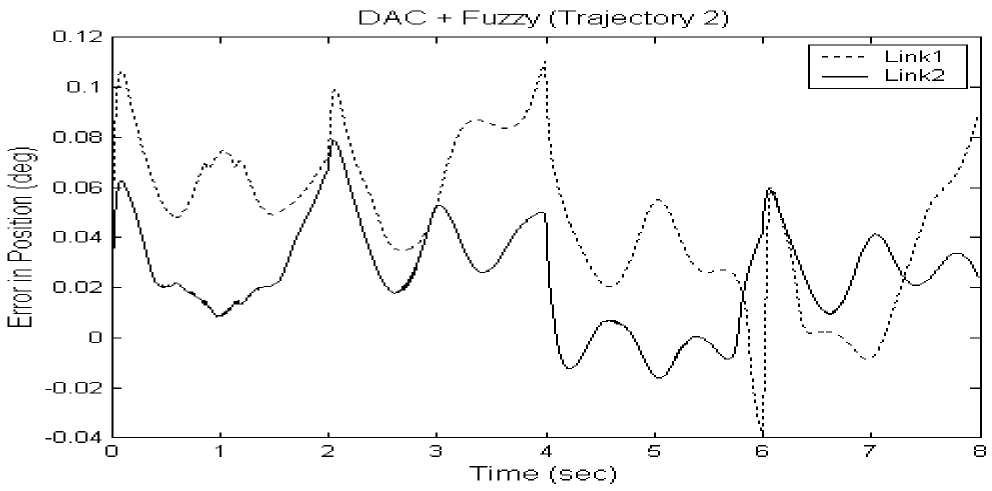


Fig. 6.2.6.2(b) Errors for DAC + Fuzzy control (Trajectory 2)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
2.	DAC	0.0613	0.0043	0.0259	-0.0007	0.0674	0.1387	0.0284	0.0600
3.	DAC+IE	0.0313	0.0418	0.0220	0.0117	0.0371	0.0798	0.0329	0.0507
4.	DAC+Fuzzy	0.0657	0.0101	0.0351	0.0001	0.0567	0.1103	0.0323	0.0791

Table 6.2.6.1. Comparison of Errors for DAC + Fuzzy control

6.2.7 COMPARISON OF PERFORMANCE

The consolidated results for the simulations carried out in section 6.2 and some other relevant simulations carried out in Chapter 4 and 5 are presented in Table 6.2.7.1 for easy comparison of the performance of various controllers.

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
	CONVENTIONAL CONTROLLERS								
1.	PD Control	7.1415	9.4447	2.3758	2.6345	6.8624	9.5918	2.1554	3.3037
2.	PID control	2.9360	3.8290	0.9742	1.0737	2.8617	4.1457	0.9264	1.4036
3.	CT	6.1340	8.5617	4.6260	-4.7008	4.9007	12.2731	4.4288	5.0733
4.	FFID	2.5940	3.6395	0.8689	1.1186	2.0083	4.1268	0.6431	1.0846
5.	CDID	0.0616	0.0695	0.0211	0.0214	0.0707	0.1244	0.0265	0.0509
6.	CT+IE	1.2919	1.7460	0.4019	0.4721	1.1894	2.3245	0.3118	0.6798
7.	FFID+IE	1.0687	1.4588	0.3508	0.4488	0.9466	1.8639	0.2759	0.4419
8.	CDID+IE	0.0604	0.0676	0.0210	0.0208	0.0697	0.1226	0.0268	0.0513
	ADAPTIVE CONTROLLERS								
9.	ACDID	0.0247	0.0001	0.0238	-0.0002	0.0391	0.0772	0.0344	0.0642
10.	ACDID+IE	0.0213	0.0	0.0212	0.0	0.0323	0.0508	0.0325	0.0745
11.	MRAC	0.9283	1.3691	0.2827	0.4039	0.6935	1.4323	0.2628	0.5390
12.	MRAC+IE	0.0321	0.0	0.0200	0.0	0.0559	0.1800	0.0270	0.0732
13.	DAC	0.0613	0.0043	0.0259	-0.0007	0.0674	0.1387	0.0284	0.0600
14.	DAC+IE	0.0313	0.0418	0.0220	0.0117	0.0371	0.0798	0.0329	0.0507
	HYBRID FUZZY CONTROLLERS								
15.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
16.	CT+Fuzzy	0.1761	0.1670	0.2361	0.1667	0.6324	1.8571	0.2895	-1.0030
17.	FFID+Fuzzy	0.1534	0.1667	0.2716	0.1663	0.1471	0.2104	0.3702	-0.9186
18.	CDID+Fuzzy	0.0216	0.0	0.0212	0.0	0.0343	-0.0610	0.0325	0.0507
19.	ACDID+Fuzzy	0.0130	0.0	0.0119	-0.0001	0.0291	0.0599	0.0226	-0.0564
20.	MRAC+Fuzzy	0.2472	0.3565	0.1261	0.1673	0.2350	0.3648	0.1423	-0.2403
21.	DAC+Fuzzy	0.0657	0.0101	0.0351	0.0001	0.0567	0.1103	0.0323	0.0791

Table. 6.2.7.1 Comparison of errors for various control strategies vs. Hybrid Fuzzy

Following observations are made based on simulations carried out in this section:

1. The Hybrid fuzzy/conventional controllers show significant performance improvement over their conventional counterparts. All model based conventional controllers, i.e., CT, FFID and CDID show marked improvement in performance.

2. The Hybrid fuzzy/conventional controllers also show significant performance improvement over the conventional controllers with integral error compensation.
3. The Hybrid fuzzy/adaptive controllers also show performance improvement over their adaptive counterparts. However the improvement is not that significant as in case of hybrid fuzzy/conventional controllers. This is mainly because of the fact that adaptive controllers by themselves give very good performance leaving little scope for further improvements.
4. Adaptive controllers are computationally intensive and adding a Fuzzy controller to them increases the computational burden even further. Further it does not result in any significant performance improvement. Hence the use of Hybrid fuzzy/adaptive controllers does not seem advisable.
5. The performance of CDID + Fuzzy hybrid controller is almost at par with ACDID+IE controller as can be seen from Table 6.2.7.1. It indicates that Hybrid conventional/fuzzy controllers can perform as good as adaptive controllers. Moreover they are computationally much less expensive than the adaptive controllers.
6. The best performance in hybrid category is that of hybrid Fuzzy + ACDID controller.

6.3 SELF ORGANIZING CONTROLLER (SOC)

Self Organizing Controller (SOC) is based on the original Fuzzy controller [Koh et al (1990), Kazemian (1998, 2002)]. It is termed as self-organizing because it is able to adjust the control strategy in a fuzzy controller automatically without any human intervention [Novakovic (1997)]. The SOC has a layered structure in which the lower layer is a LUT based controller and the higher layer is the adjustment mechanism. Fig. 6.3.1 shows the block diagram of SOC. At the lower layer is a Fuzzy controller. The two inputs to this controller are the error *e* and change in error *ce*. These are multiplied by normalization gains *GE* and *GCE* respectively before being given to the rule base in **F**. The value obtained from lookup table in **F** is the output of the controller *u*. This is multiplied by the output gain *GU* to give the final control signal *U*.

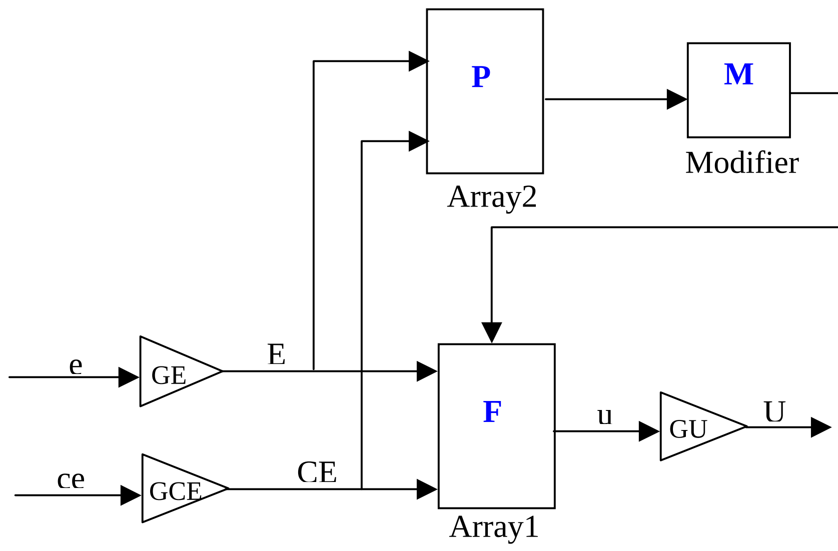


Fig. 6.3.1 Block diagram of SOC [Jantzen (1998)]

The idea behind self-organization is to let an adjustment mechanism update the values in LUT of F, based on current performance of the controller. The updating should be such that the table entry responsible for poor performance is punished, so that the next time this entry is used, the performance is better. If the performance is good, the entries are left unchanged.

The input to higher layer is also *error* and *change in error*, and it modifies the LUT in **F** through a *modifier* algorithm **M** when necessary. It uses a performance measure to decide the magnitude of each change to **F**. The performance measures are numbers; organized in a table **P,** which is of same size as **F**, expressing what is desirable, in a transient response. The table **P** can be built using linguistic rules, but is often built manually, based on experience. The same performance table **P** may be used with a different process, without prior knowledge of the process, since it only expresses the *desired* transient response. The controller can start from scratch with an **F**-lookup table full of zeros; it will, however, converge faster towards a stable table, if **F** is initialized with sensible numbers to begin with.

The SOC learns to control the system in accordance with the desired response. This is called training. At the sampling instant *n*, it records the error between desired performance and the actual performance. Based on this error it modifies the LUT in **F** accordingly. The performance table **P** evaluates the current state and returns a performance measure P(i_n, j_n), where i_n is the index corresponding to E_n, and j_n is the index corresponding to CE_n. Tables 6.3.1 and 6.3.2 are examples of performance tables.

$\dot{e}$	**Membership Function**												
e	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
-6	-6	-6	-6	-6	-6	-6	-6	-5	-4	-3	-2	-1	0
-5	-6	-6	-6	-6	-5	-4	-4	-4	-3	-2	-1	0	0
-4	-6	-6	-6	-5	-4	-3	-3	-3	-2	-1	0	0	1
-3	-6	-6	-5	-4	-3	-2	-2-	-2	-1	0	0	1	2
-2	-6	-5	-4	-3	-2	-1	-1	-1	0	0	1	2	3
-1	-5	-4	-3	-2	-1	-1	0	0	0	1	2	3	4
0	-5	-4	-3	-2	-1	0	0	0	1	2	3	4	5
1	-3	-2	-1	0	0	0	0	1	1	2	3	4	5
2	-2	-1	0	0	0	1	1	1	2	3	4	5	6
3	-1	0	0	0	1	2	2	2	3	4	5	6	6
4	0	0	0	1	2	3	3	3	4	5	6	6	6
5	0	0	1	2	3	4	4	4	5	6	6	6	6
6	0	1	2	3	4	5	6	6	6	6	6	6	6

Table 6.3.1 Example of Performance Table (Yamazaki, 1982)

$\dot{e}$	**Membership Function**												
e	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
-6	-6	-6	-6	-6	-6	-6	-6	0	0	0	0	0	0
-5	-6	-6	-6	-6	-6	-6	-6	-3	-2	-2	0	0	0
-4	-6	-6	-6	-6	-6	-6	-6	-5	-4	-2	0	0	0
-3	-6	-5	-5	-4	-4	-4	-4	-3	-2	0	0	0	0
-2	-6	-5	-4	-3	-2	-2	-2	0	0	0	0	0	0
-1	-5	-4	-3	-2	-1	-1	-1	0	0	0	0	0	0
0	-4	-3	-2	-1	0	0	0	0	0	1	2	3	4
1	0	0	0	0	0	0	1	1	1	2	3	4	5
2	0	0	0	0	0	0	2	2	2	3	4	5	6
3	0	0	0	0	2	3	4	4	4	4	5	5	6
4	0	0	0	2	4	5	6	6	6	6	6	6	6
5	0	0	0	2	2	3	6	6	6	6	6	6	6
6	0	0	0	0	0	0	6	6	6	6	6	6	6

Table 6.3.2 Example of another Performance Table (Procyk and Mamdani, 1979)

It can be seen that the zeros in Table 6.3.1 are almost in a diagonal band. It can be shown that this amounts to a desired first order system behavior. Table 6.3.2 on the other hand has zeros in a ‘Z’ shaped patch. This allows for a zero slope to begin with, and a slight overshoot at the end of transient [Jantzen (1998)].

6.3.1 PURE FUZZY (FIXED PARARAMETERS)

The controller tested here is exactly same as that discussed in section 6.2. The only difference here is that the normalization gain for error was increased by a factor of ten to get an improved performance from the fuzzy controller. This is because the SOC performance can then be compared against ‘almost’ best possible fuzzy performance. The two trajectories used for simulation are same as earlier. It is assumed that the manipulator parameters do not change throughout the motion. The error profiles for the two links are shown in Fig. 6.3.1.1(a) for the first trajectory, and Fig. 6.3.1.1(b) for the second trajectory. Table 6.3.1.1 lists the magnitudes of various error measures for the two trajectories for this controller. As can be seen, the errors for both the trajectories have improved when compared to fuzzy controller investigated in section 6.2.

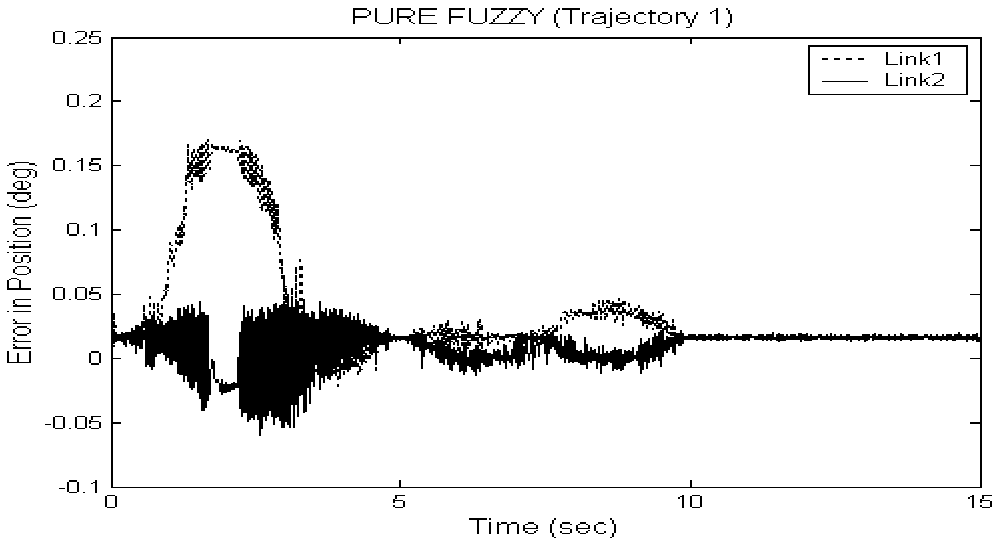


Fig. 6.3.1.1(a) Errors for Fuzzy control (Increased normalization gains, Fixed parameters, Trajectory 1)

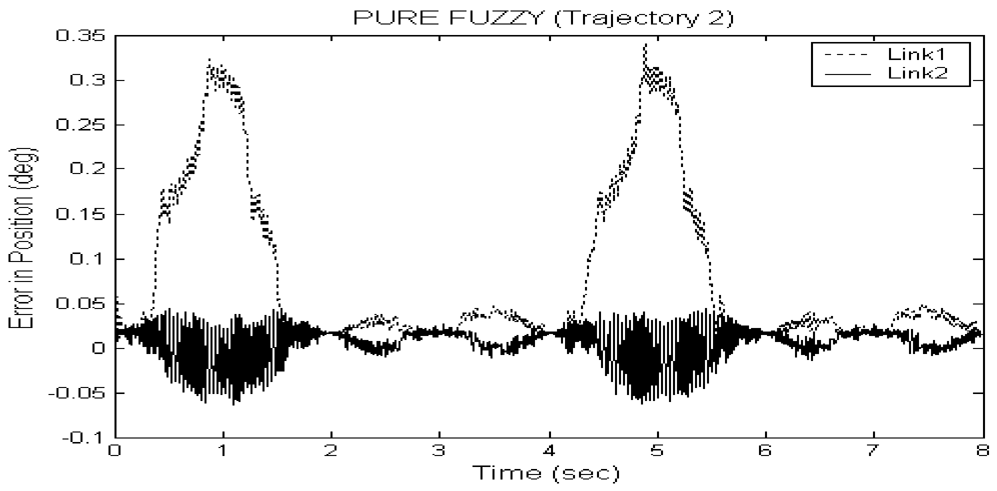


Fig. 6.3.1.1(b) Errors for Fuzzy control (Increased normalization gains, Fixed parameters, Trajectory 2)

Fuzzy Control (Fixed parameters) Errors (deg)							
TRAJECTORY NO.1				TRAJECTORY NO.2			
Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.0551	0.0173	0.0167	0.0171	0.1198	0.3397	0.0200	-0.0632

Table 6.3.1.1. Errors for Fuzzy control (Increased normalization gains, Fixed parameters)

The high frequency in the above error profiles is mainly due to increased normalization gains, which leads to increased control activity.

6.3.2 SOC (FIXED PARAMETERS, ZERO LUT)

The SOC with initial LUT empty was investigated next. Here too the parameters do not change throughout the manipulator motion. The error profiles for the two links are shown in Fig. 6.3.2.1(a) for the first trajectory, and Fig. 6.3.2.1(b) for the second trajectory. Table 6.3.2.1 lists the magnitudes of various error measures for the two trajectories for this controller. The SOC controller gives better performance than the Pure Fuzzy controller even with initial LUT empty. This can be seen from Tables 6.3.1.1 and 6.3.2.1.

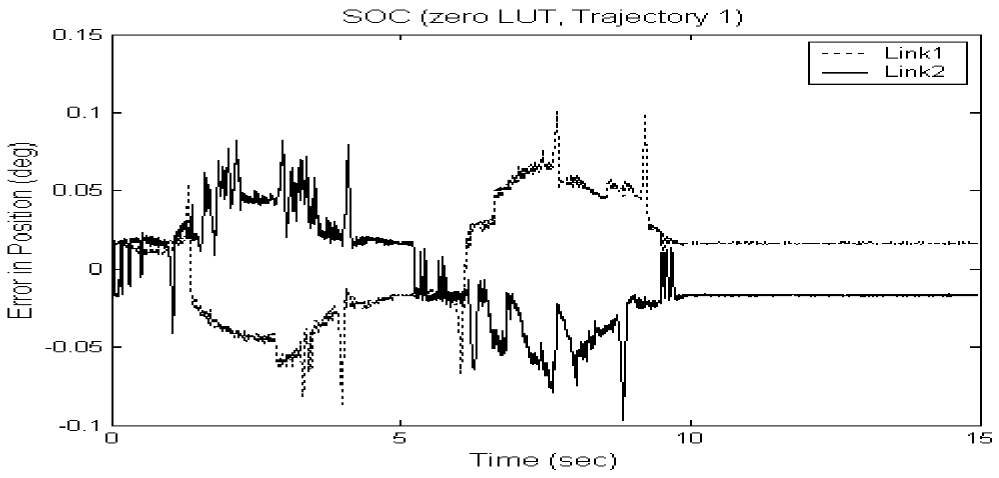


Fig. 6.3.2.1(a) Errors for SOC control (Fixed Parameters, Zero LUT, Trajectory 1)

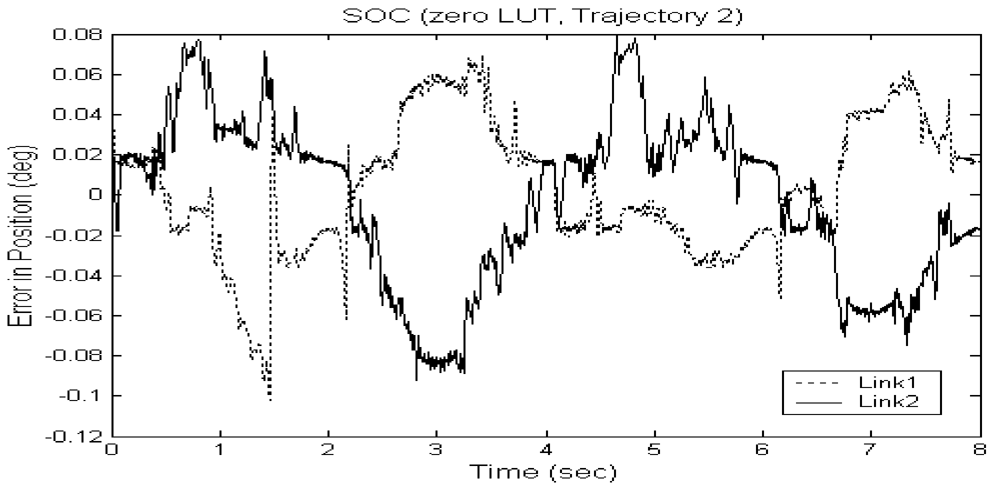


Fig. 6.3.2.1(b) Errors for SOC control (Fixed Parameters, Zero LUT, Trajectory 2)

SOC (Fixed parameters, Zero LUT) Errors (deg)							
TRAJECTORY NO.1				TRAJECTORY NO.2			
Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.0338	0.0164	0.0306	-0.0163	0.0331	-0.1023	0.0409	-0.0923

Table 6.3.2.1. Errors for SOC control (Fixed parameters, Zero LUT)

6.3.3 SOC (FIXED PARAMETERS, NONZERO LUT)

In this case it was assumed that the initial lookup table is not empty and the starting point is same as the lookup table for Pure fuzzy controller (Table 6.1). The error profiles for the two links are shown in Fig. 6.3.3.1(a) for the first trajectory, and Fig. 6.3.3.1(b) for the second trajectory. Table 6.3.3.1 lists the magnitudes of various error measures for the two trajectories for this controller.

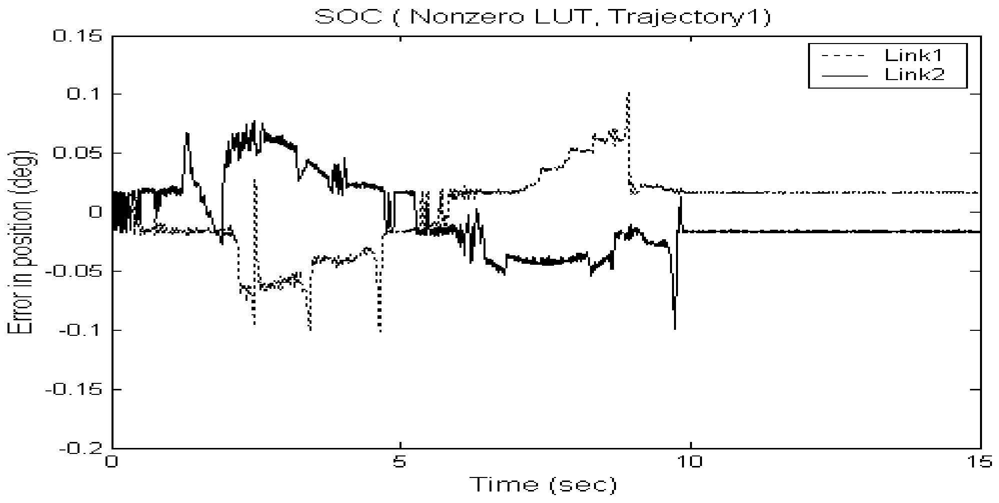


Fig. 6.3.3.1(a) Errors for SOC control (Fixed Parameters, Nonzero LUT, Trajectory 1)

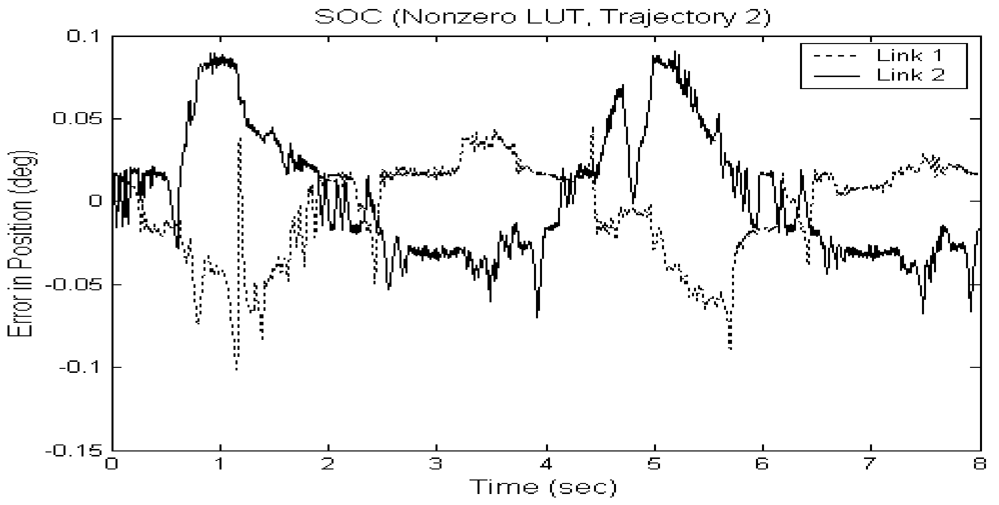


Fig. 6.3.3.1(b) Errors for SOC control (Fixed Parameters, Nonzero LUT, Trajectory 2)

SOC (Fixed parameters, Nonzero LUT) Errors (deg)							
TRAJECTORY NO.1				TRAJECTORY NO.2			
Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.0315	0.0164	0.0306	-0.0163	0.0293	-0.1014	0.0385	0.0908

Table 6.3.3.1. Errors for SOC control (Fixed parameters, Nonzero LUT)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	0.0551	0.0173	0.0167	0.0171	0.1198	0.3397	0.0200	-0.0632
2.	SOC Zero LUT	0.0338	0.0164	0.0306	-0.0163	0.0331	-0.1023	0.0409	-0.0923
3.	SOC Nonzero LUT	0.0315	0.0164	0.0306	-0.0163	0.0293	-0.1014	0.0385	0.0908

Table 6.3.3.2. Errors for SOC and Fuzzy (Fixed Parameters)

Table 6.3.3.2 lists the errors for the above three cases for comparison. It is seen that the errors decrease from Pure fuzzy to SOC with zero LUT to SOC with nonzero LUT. It is however observed that this improvement is marginal. This is because in these cases the manipulator parameters do not change during motion and therefore the pure fuzzy controller gives a good performance as it is tuned for these fixed parameters of the manipulator.

6.3.4 PURE FUZZY (CHANGING PARARAMETERS)

The next three simulations are similar to the previous three except that in these cases we assume that the manipulator picks up and releases load during motion. Hence the manipulator parameters change here. The first simulation is for Pure Fuzzy controller. It is observed here as in section 6.1, that the performance of controller degrades compared to the fixed parameter case. The error profiles for the two links are shown in Fig.

6.3.4.1(a) for the first trajectory, and Fig. 6.3.4.1(b) for the second trajectory. Table 6.3.4.1 lists the various error magnitudes for the two trajectories for this controller.

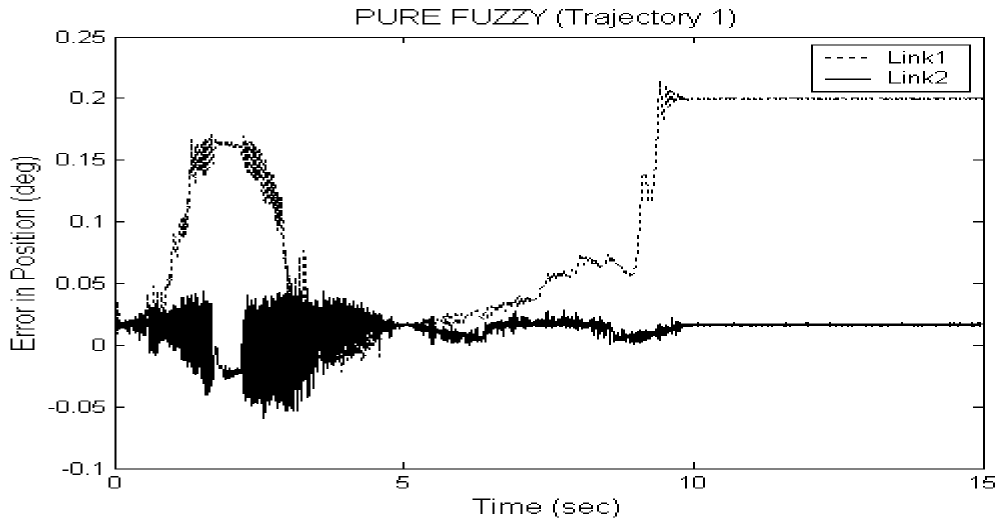


Fig. 6.3.4.1(a) Errors for Fuzzy control (Increased normalization gains, Changing parameters, Trajectory 1)

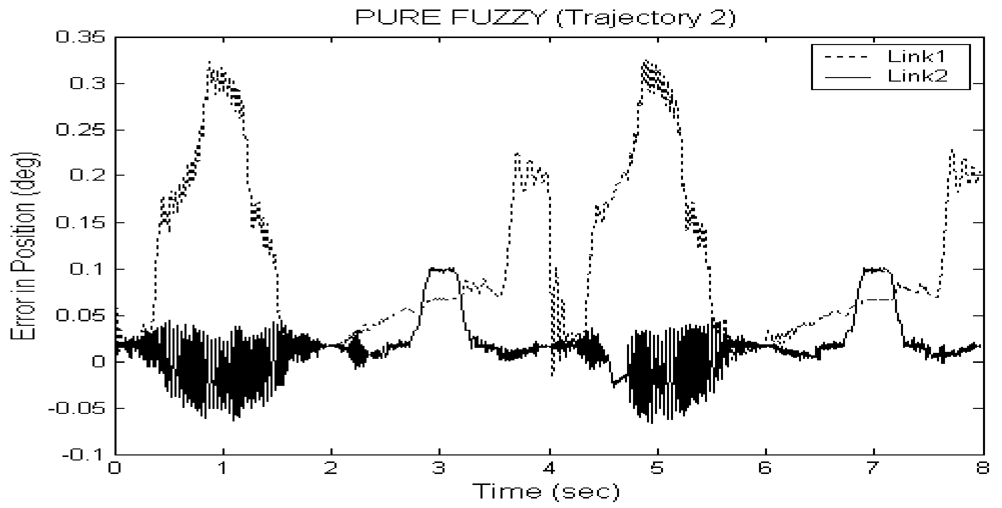


Fig. 6.3.4.1(b) Errors for Fuzzy control (Increased normalization gains, Changing parameters, Trajectory 2

Fuzzy Control (Changing parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 **0°→ 90°→0°**		**Link2** **0°→ -90°→0°**		**Link1** **0°→ 45°→0°→**		**Link2** **0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.1364	0.1998	0.0180	0.0173	0.1382	0.3249	0.0354	0.1012

Table 6.3.4.1 Errors for Fuzzy control (Increased normalization gains, Changing parameters)

6.3.5 SOC (CHANGING PARAMETERS, ZERO LUT)

The performance of SOC under the circumstance of changing parameters is investigated here. The error profiles for the two links are shown in Fig. 6.3.5.1(a) for the first trajectory, and Fig. 6.3.5.1(b) for the second trajectory. Table 6.3.5.1 lists the various error measures for the two trajectories for this controller. It is seen that SOC even with an initially empty LUT performs better than Fuzzy controller. This is primarily because SOC can auto tune the LUT as the parameters of manipulator change.

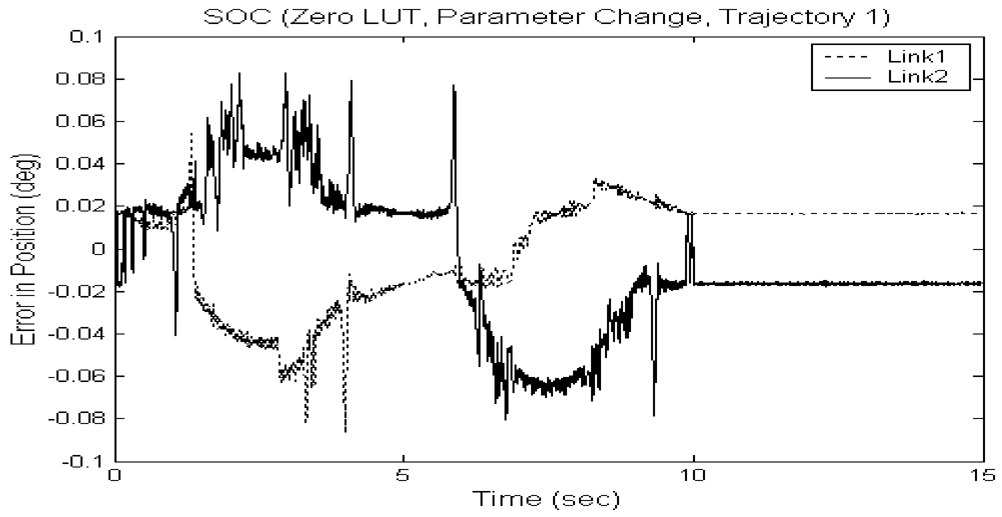


Fig. 6.3.5.1(a) Errors for SOC control (Changing Parameters, Zero LUT, Trajectory 1)

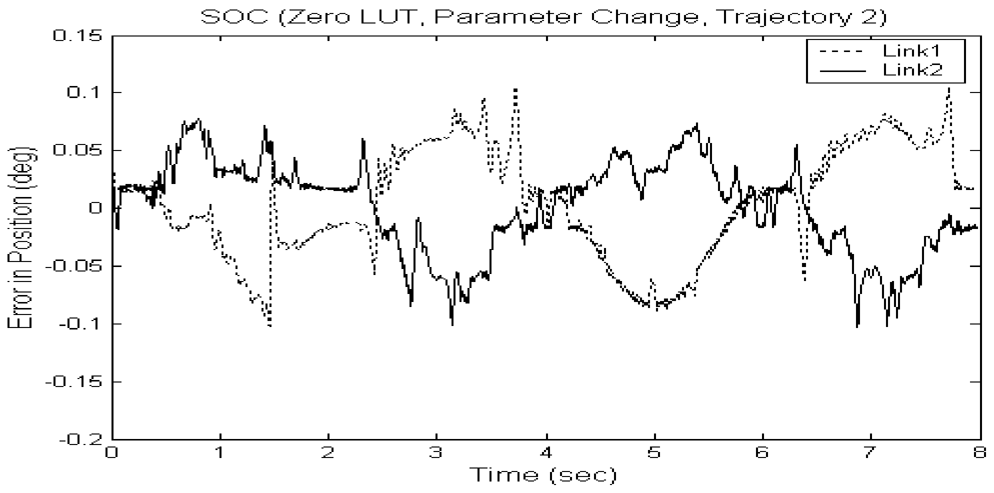


Fig. 6.3.5.1(b) Errors for SOC control (Changing Parameters, Zero LUT, Trajectory 2)

SOC (Variable parameters, Zero LUT) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.0245	0.0166	0.0333	-0.0169	0.0477	0.1059	0.0395	-0.1032

Table 6.3.5.1. Errors for SOC control (Changing parameters, Zero LUT)

6.3.6 SOC (CHANGING PARAMETERS, NONZERO LUT)

The last case investigated was the effect on performance of SOC when the lookup table is initially nonzero. Here we initialized the values in lookup table to be same as those used for Pure fuzzy controller. The error profiles for the two links are shown in Fig. 6.3.6.1(a) for the first trajectory, and Fig. 6.3.6.1(b) for the second trajectory. Table 6.3.6.1 lists the magnitudes of various error measures for the two trajectories for this controller.

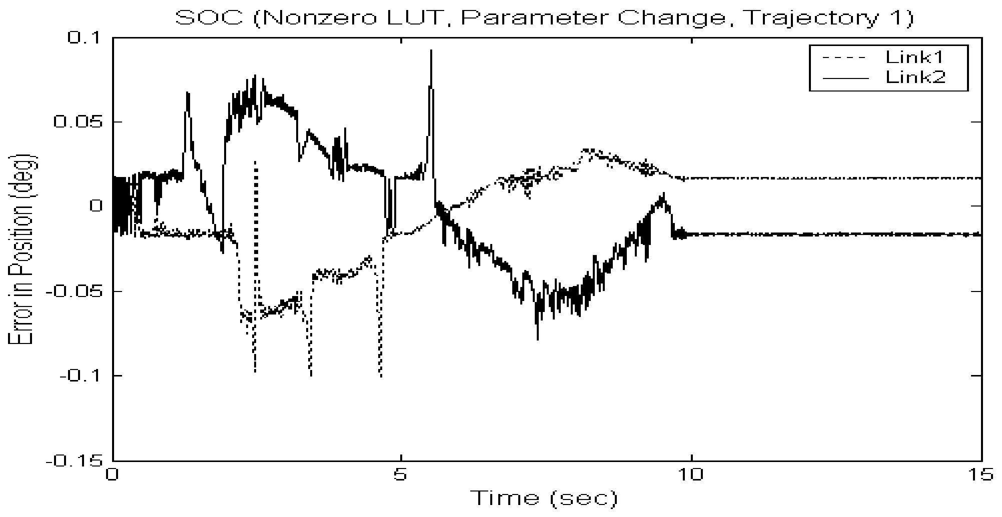


Fig. 6.3.6.1(a) Errors for SOC control (Changing Parameters, Nonzero LUT, Trajectory 1)

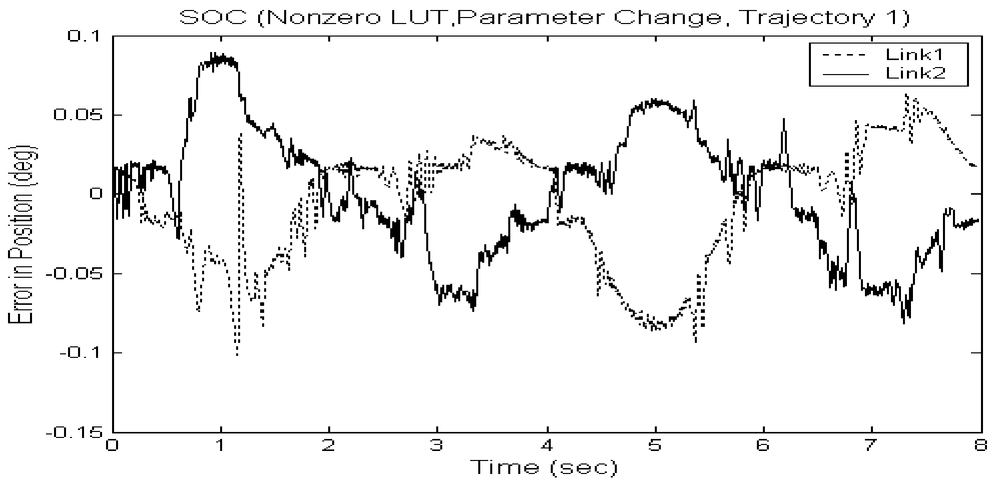


Fig. 6.3.6.1(b) Errors for SOC control (Changing Parameters, Nonzero LUT, Trajectory 2)

SOC (Variable parameters, Nonzero LUT) Errors (deg)							
TRAJECTORY NO.1				TRAJECTORY NO.2			
Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.0271	0.0166	0.0305	-0.0166	0.0380	-0.1014	0.0387	0.0898

Table 6.3.6.1. Errors for SOC control (Changing parameters, Nonzero LUT)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	0.1364	0.1998	0.0180	0.0173	0.1382	0.3249	0.0354	0.1012
2.	SOC Zero LUT	0.0245	0.0166	0.0333	-0.0169	0.0477	0.1059	0.0395	-0.1032
3.	SOC Nonzero LUT	0.0271	0.0166	0.0305	-0.0166	0.0380	-0.1014	0.0387	0.0898

Table 6.3.6.2. Errors for SOC and Fuzzy (Changing Parameters)

Table 6.3.6.2 lists the errors for the above three cases for comparison. It is seen that the errors decrease from Pure fuzzy to SOC with zero LUT to SOC with nonzero LUT.

6.3.7 COMPARISON OF PERFORMANCE

Following observations are made based on simulations carried out in this section (6.3):

1. When the manipulator parameters do not change during the course of trajectory, a fixed LUT based fuzzy controller gives good performance. But building up the LUT requires lot of intuition and experience. Besides it may also requires adjustment of values through trial and error by repeated runs. A SOC under this situation can quickly build its LUT starting from all zero values and give better performance than Pure Fuzzy controller. The performance of SOC further improves if we can start with non zero LUT.
2. When the manipulator parameters change during the course of trajectory, the performance of Pure Fuzzy controller degrades considerably. This is because this controller uses a fixed LUT, which is tuned for one set of parameters. The SOC under these circumstances gives much better performance as it can change its LUT as the parameters of manipulator change. This is seen from simulations done in section 6.3. Thus using SOC for manipulator control seems to be a better option than Pure fuzzy as invariably the manipulator parameters change during motion. Further it does not require any tuning beforehand. However it is better to use a

nonzero lookup table that incorporates some user experience as it can lead to improved performance.

6.4 SELF TUNING (ADAPTIVE) FUZZY CONTROLLER (STFC)

A Fuzzy controller consists of three major components that can be altered to give different controller behaviors. These three components are:

- The normalization and denormalization scaling factors
- The fuzzy set representing the meaning of linguistic values
- The if-then rule base

If the above three components remain fixed the fuzzy controller is of type non-adaptive. If on the other hand any of the above three components are altered when the controller is running, it is known as Adaptive-Fuzzy [Han-Xiong (1996)].

A Controller that changes scaling factors or modifies the fuzzy set definitions is known as ‘self-tuning’ controller. Adaptive Fuzzy controller that modifies the rule base is known as ‘self-organizing’ controller. These controllers can start with an existing rule base and then modify it or they can build the rule base entirely afresh starting with no rules at all. This type of controller was studied in previous section. Figure 6.4.1 shows the classification of fuzzy controllers.

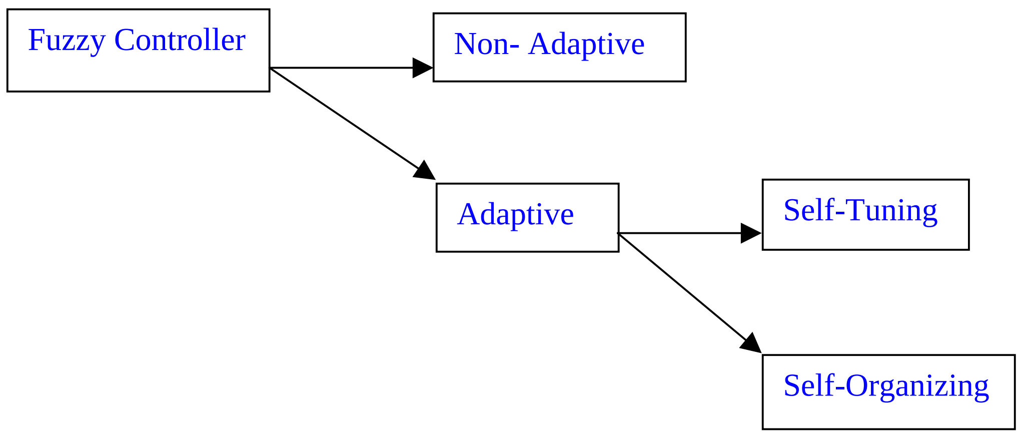


Fig 6.4.1. Classification of Fuzzy controllers

The Adaptive Fuzzy controller that we investigated is of PD type [Mudi and Pal (1999), Chatterjee and Watanabe (2005)]. The output gain (denormalization, GU) of this controller is adjusted on-line depending on the present values of error and error derivative. Thus the controller is of self-tuning type. We also investigated the more general case where both the input (normalization, GE & GCE) and output (denormalization, GU) gains of the controller are adapted on-line. The block diagram of the self-tuning fuzzy controller is shown in Fig. 6.4.2. The membership functions for controller inputs (error and error derivative) and output are defined on the common interval [-6 6] and are same as shown in Fig. 6.3. The membership functions for gain updating factor (α) are defined on [0 1]. These membership functions are as shown in Fig. 6.4.3.

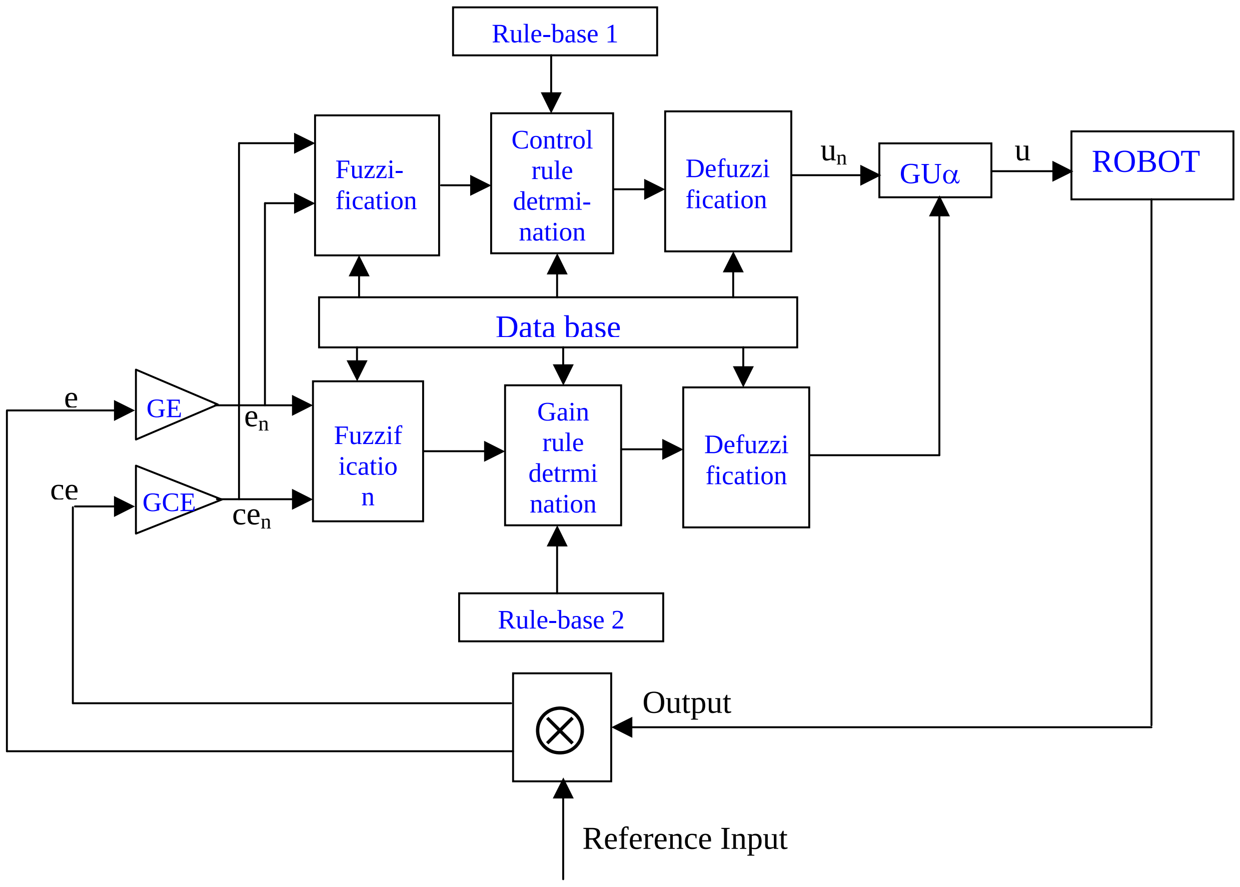


Fig. 6.4.2. Block diagram of the self-tuning fuzzy controller (adapted from Mudi and Pal, 1999)

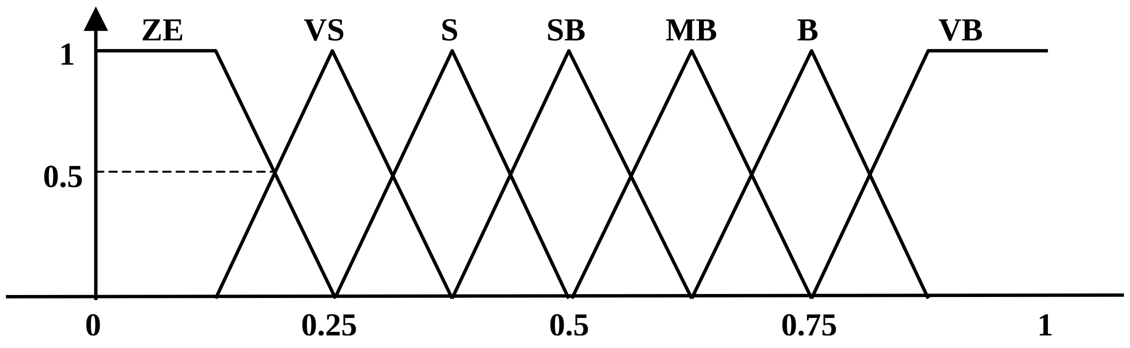


Fig. 6.4.3 Membership functions for gain updating factor (α)

For the conventional fuzzy controller the controller output is mapped to the respective actual output by the output gain GU. On the other hand in the self-tuning fuzzy controller the actual output is obtained by multiplying the controller output with GUα. The gain-updating factor α is calculated on-line using a *model independent* fuzzy rule base which has e and $\dot{e}$ as inputs. The governing equations for this self-tuning fuzzy controller are given below.

$$e_n = GE.e \tag{6.4.1}$$

$$ce_n = GCE.ce \tag{6.4.2}$$

$$u = \alpha.GU.u_n \tag{6.4.3}$$

The fuzzy controller produces output based on rules of the form:

Ri : If e is **E** and ce is **CE** then u is **U**

The complete rule base for the controller is shown in Table. 6.4.1.

$\dot{e}$ \ e	**NB**	**NM**	**NS**	**ZE**	**PS**	**PM**	**PB**
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

Table. 6.4.1. Fuzzy controller Rule Base

The gain updating part of the controller produces output based on rules of the form:

Ri : If e is **E** and ce is **CE** then α is α

The complete rule base used for updating α is shown in Table. 6.4.2.

$\dot{e}$ \ e	**NB**	**NM**	**NS**	**ZE**	**PS**	**PM**	**PB**
NB	VB	VB	VB	B	SB	S	ZE
NM	VB	VB	B	B	MB	S	VS
NS	VB	MB	B	VB	VS	S	VS
ZE	S	SB	MB	ZE	MB	SB	S
PS	VS	S	VS	VB	B	MB	VB
PM	VS	S	MB	B	B	VB	VB
PB	ZE	S	SB	B	VB	VB	VB

Table. 6.4.2. Fuzzy Rule Base for α

The parameter α is independent of any manipulator parameter and depends only on current system states. Thus the self-tuning scheme is largely independent of the process being controlled.

The following steps were used for tuning the controller:

Assuming that α=1, we first adjust the value of GE so that the normalized error covers the entire domain [-6 6] to make efficient use of rule base. We then adjust the values of GCE and GU to make the output as acceptable as possible. This process is done through trial and error for any one trajectory. Now we have a good conventional fuzzy controller, which becomes the initial starting point for fuzzy self-tuning controller.

The output-scaling factor (GU) is now set to three times (to keep the rise time almost same) the value found in previous step. The other two scaling factors are kept same as determined in previous step. α is no longer fixed at 1 but is calculated on-line from its rule base.

6.4.1 PURE FUZZY (FIXED PARARAMETERS)

The controller investigated in this section is a normal Pure Fuzzy controller (Non LUT based), exactly similar to the controller discussed in section 6.3.1. The normalization and denormalization factors are chosen to be exactly same and so are the rule base and membership functions. The complete rule base is shown in Table. 6.4.1.

The difference here is that this controller evaluates the fuzzy rules as it is and does not make use of lookup table while calculating the output torque values for the two links of the manipulator. The disadvantage of not using the lookup table is mainly in terms of considerably more number of calculations required which in turn increases the sampling time of the controller. The advantage on the other hand is in terms of much smoother controller action compared to lookup table based controller. This is of importance when we modify this controller to self-tuning controller. The self-tuning controller gives much better performance if the control action of the original controller is smooth.

The two trajectories used for simulation are same as earlier. For the first simulation it is assumed that the manipulator parameters do not change throughout the motion. The error profiles for the two links are shown in Fig. 6.4.1.1(a) for the first trajectory, and Fig. 6.4.1.1(b) for the second trajectory. Table 6.4.1.1 lists the various error magnitudes for the two trajectories for this controller. A comparison between the error profiles (for trajectory 1) of this controller and the exactly similar LUT based controller studied in 6.3.1 is shown in Fig 6.4.1.1(c). As can be seen from this figure, the errors for LUT based controller are lower compared to Non-LUT based controller. This is mainly because of manual fine-tuning done for LUT based controller. Also as can be expected the trajectory for Non-LUT based controller is much smoother compared to LUT based controller.

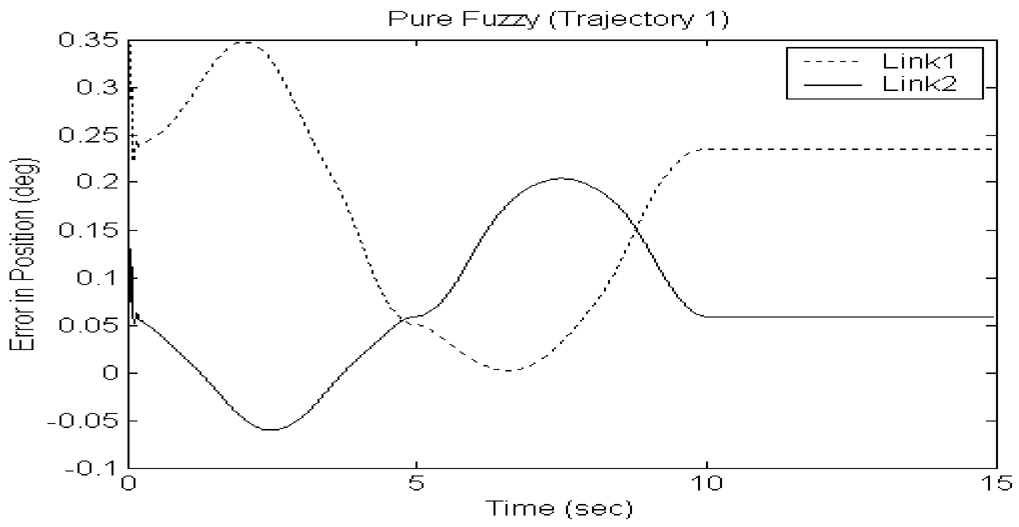


Fig. 6.4.1.1(a) Errors for Fuzzy control (Non LUT based, Fixed parameters, Trajectory 1)

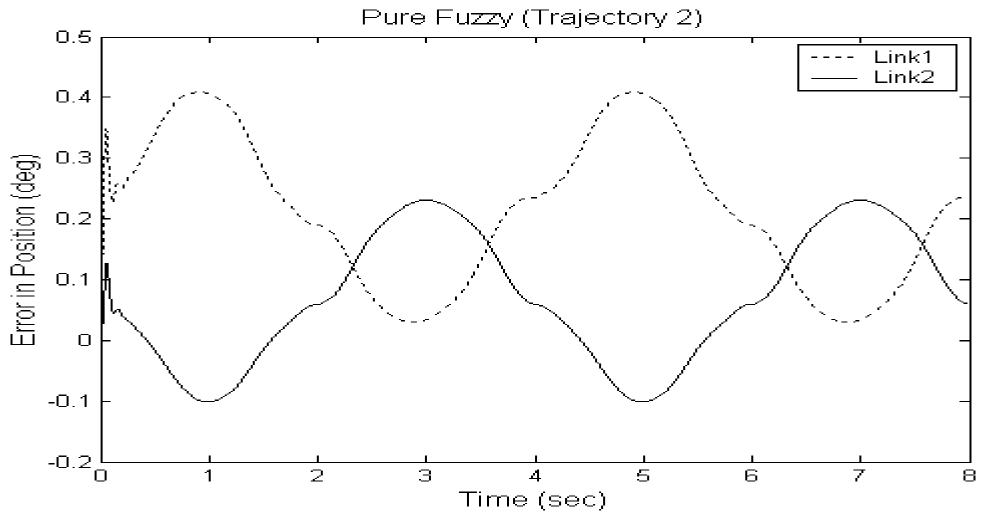


Fig. 6.4.1.1(b) Errors for Fuzzy control (Non LUT based, Fixed parameters, Trajectory 2)

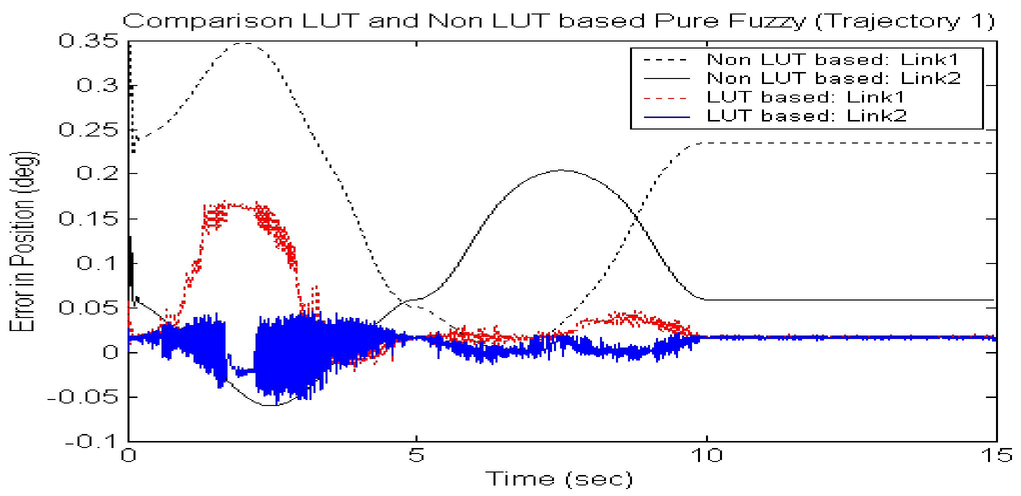


Fig 6.4.1.1(c) Comparison of errors for LUT and Non LUT based Fuzzy controllers (Fixed parameters)

Pure Fuzzy (Fixed parameters Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2** 0°→ -90°→0°		**Link1** 0°→ 45°→0°→		**Link2** 0°→ -45°→0°→	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.2104	0.2345	0.0966	0.0590	0.2453	0.4088	0.1273	0.2308

Table 6.4.1.1. Errors for Fuzzy control (Non LUT based, Fixed parameters)

6.4.2 ADAPTIVE FUZZY (FIXED PARAMETERS)

The self-tuning adaptive fuzzy controller is investigated in this section. This controller is discussed in detail in section 6.4. We run the simulation for two trajectories for the case that manipulator parameters do not change during the entire duration of motion. Fig. 6.4.2.1(a) shows the error profiles for first trajectory. Fig. 6.4.2.1(b) shows a comparison of errors between adaptive fuzzy and pure fuzzy control. As can be seen from the figure, the adaptive fuzzy controller gives improved performance compared to pure fuzzy controller. Errors for both links for adaptive fuzzy controller are reduced. However like the pure fuzzy controller, the adaptive fuzzy controller also has non-zero steady state errors for both links albeit reduced to some extent.

Fig. 6.4.2.1(c) shows the error profiles for the second trajectory. Fig. 6.4.2.1(d) shows a comparison of errors between adaptive fuzzy and pure fuzzy control for the second trajectory. Once again it is observed that adaptive fuzzy controller gives better performance with both r.m.s and maximum errors getting reduced, when compared to pure fuzzy controller. Table 6.4.2.1 lists the various error measures for the two trajectories for this controller.

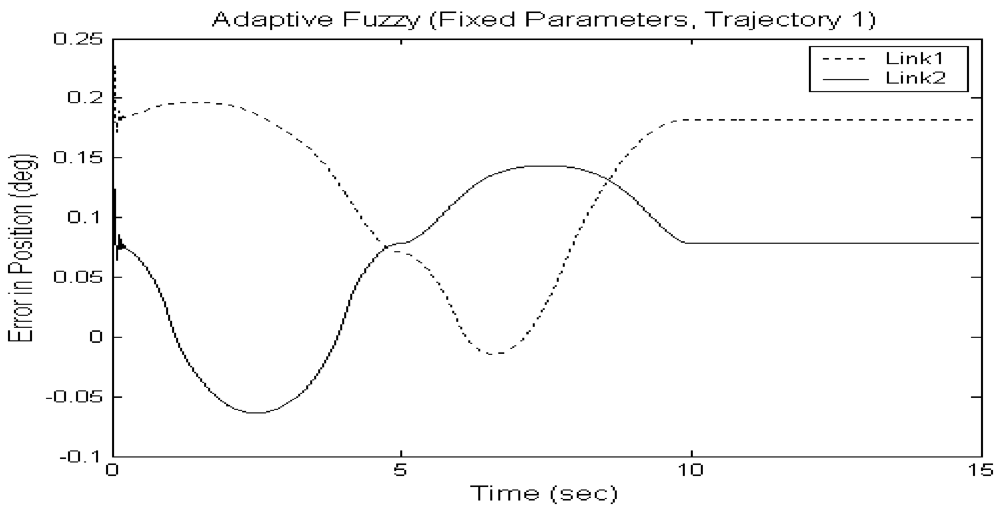


Fig. 6.4.2.1(a) Errors for Adaptive Fuzzy control (Fixed parameters, Trajectory 1)

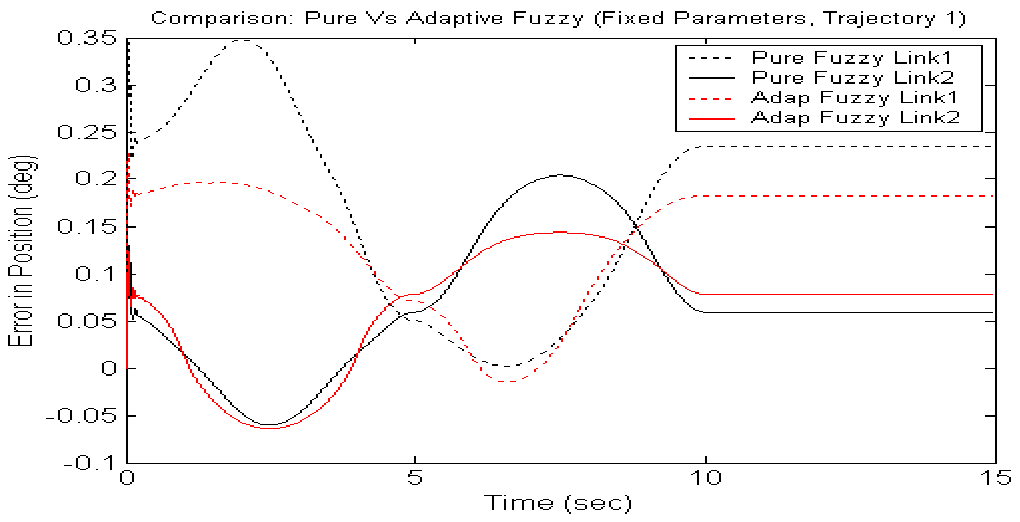


Fig. 6.4.2.1(b) Comparison of Errors for Adaptive Fuzzy Vs Fuzzy control (Fixed parameters, Trajectory 1)

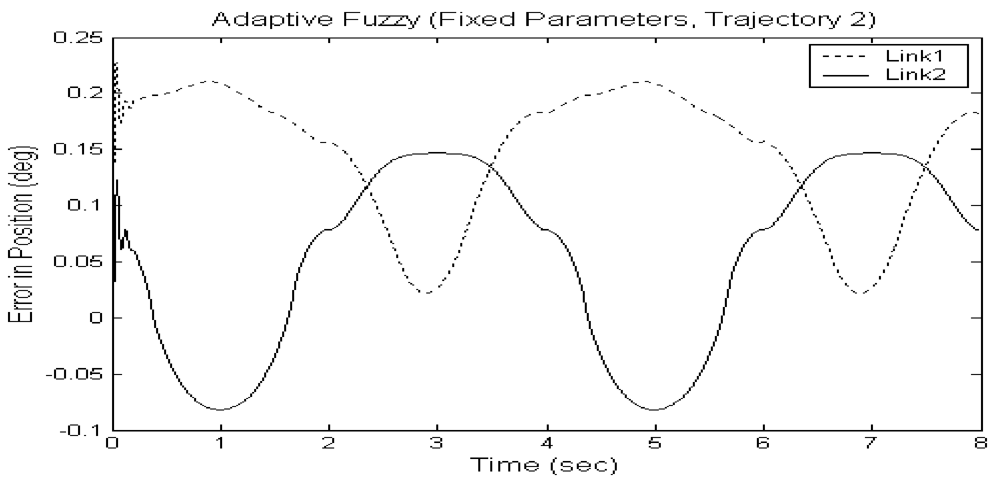


Fig. 6.4.2.1(c) Errors for Adaptive Fuzzy control (Fixed parameters, Trajectory 2)

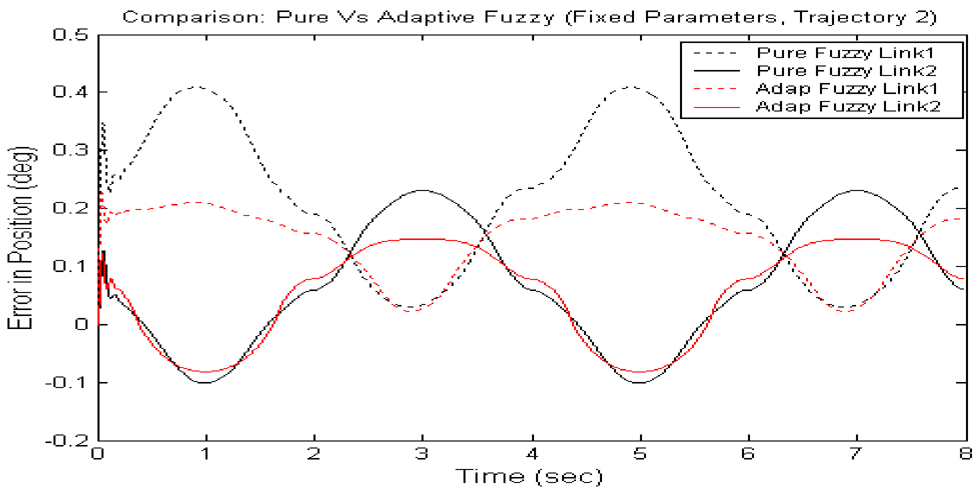


Fig. 6.4.2.1(d) Comparison of Errors for Adaptive Fuzzy Vs Fuzzy control (Fixed parameters, Trajectory 2)

Adaptive Fuzzy (Fixed parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.1541	0.1819	0.0887	0.0784	0.1587	0.2272	0.0990	0.1468

Table 6.4.2.1. Errors for Adaptive Fuzzy control (Fixed parameters)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	0.2104	0.2345	0.0966	0.0590	0.2453	0.4088	0.1273	0.2308
2.	SOC Zero LUT	0.0338	0.0164	0.0306	-0.0163	0.0331	-0.1023	0.0409	-0.0923
3.	SOC Nonzero LUT	0.0315	0.0164	0.0306	-0.0163	0.0293	-0.1014	0.0385	0.0908
4	STFC	0.1541	0.1819	0.0887	0.0784	0.1587	0.2272	0.0990	0.1468

Table 6.4.2.2. Errors for Fuzzy, SOC and STFC (Fixed Parameters)

Table 6.4.2.2 lists the errors for the Pure Fuzzy, SOC and STFC for comparison. It can be seen that STFC performs well but is not as good as SOC.

6.4.3 PURE FUZZY (CHANGING PARARAMETERS)

The simulations done in this section are similar to those done in previous section except that in these cases we assume that the manipulator picks up and releases load during motion. Hence the manipulator parameters change during the course of motion. The first simulation is for pure fuzzy controller. It is observed that the performance of controller degrades compared to the fixed parameter case. In particular it is the transient performance of the controller that suffers. The steady state errors for first trajectory remain same for fixed and changing parameter cases. The maximum errors for second trajectory increase in case of changing parameters. The error profiles for the two links are shown in Fig. 6.4.3.1(a) for the first trajectory, and Fig. 6.4.3.1(b) for the second trajectory. Table 6.4.3.1 lists the magnitudes of various errors for the two trajectories for this controller. A comparison between the error profiles (for trajectory 1) of this controller and the exactly similar LUT based controller studied in 6.3.4 is shown in Fig 6.4.3.1(c). As can be seen from this figure, the errors for LUT based controller are lower compared to Non-LUT based controller. As explained earlier, this is mainly because of manual fine-tuning done for LUT based controller. Also as can be expected the trajectory for Non-LUT based controller is much smoother compared to LUT based controller.

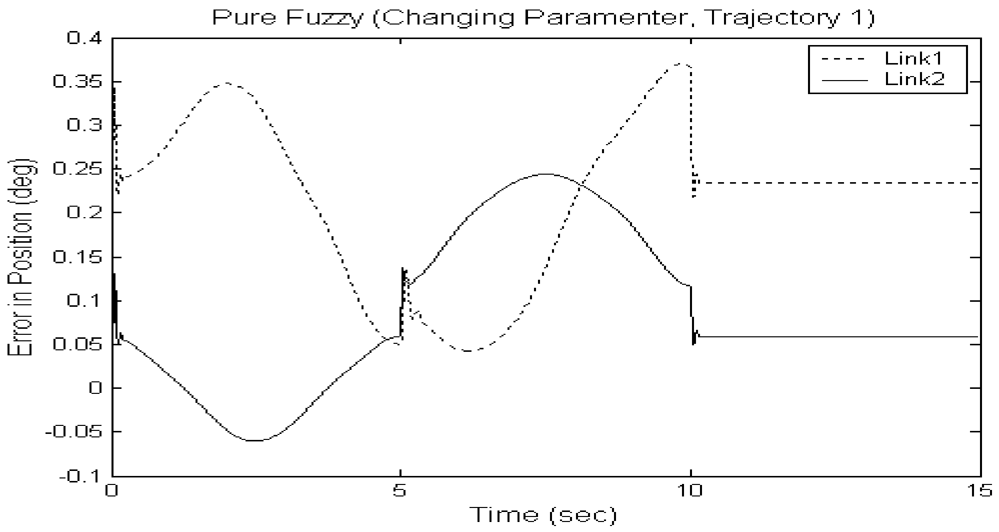


Fig. 6.4.3.1(a) Errors for Fuzzy control (Non LUT based, Changing parameters, Trajectory 1)

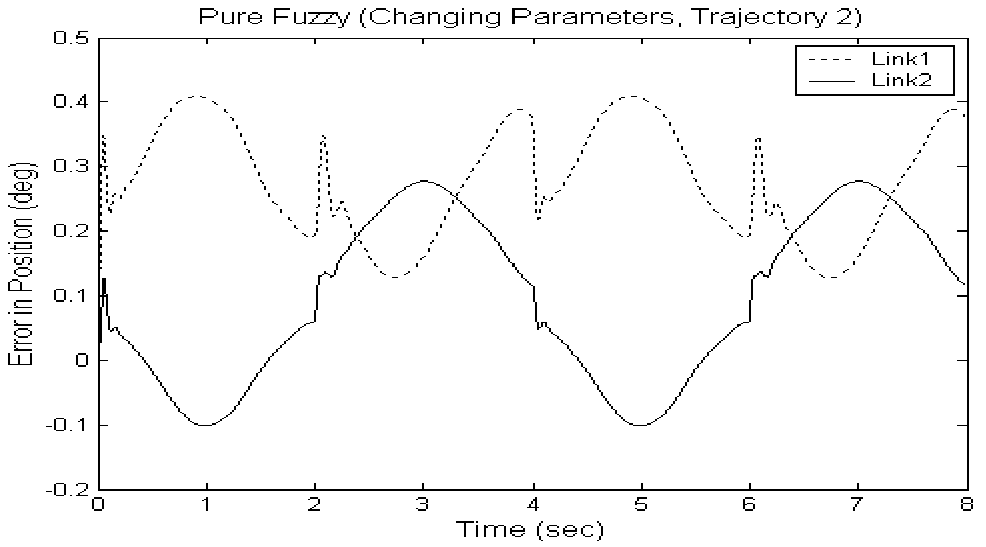


Fig. 6.4.3.1(b) Errors for Fuzzy control (Non LUT based, Changing parameters, Trajectory 2)

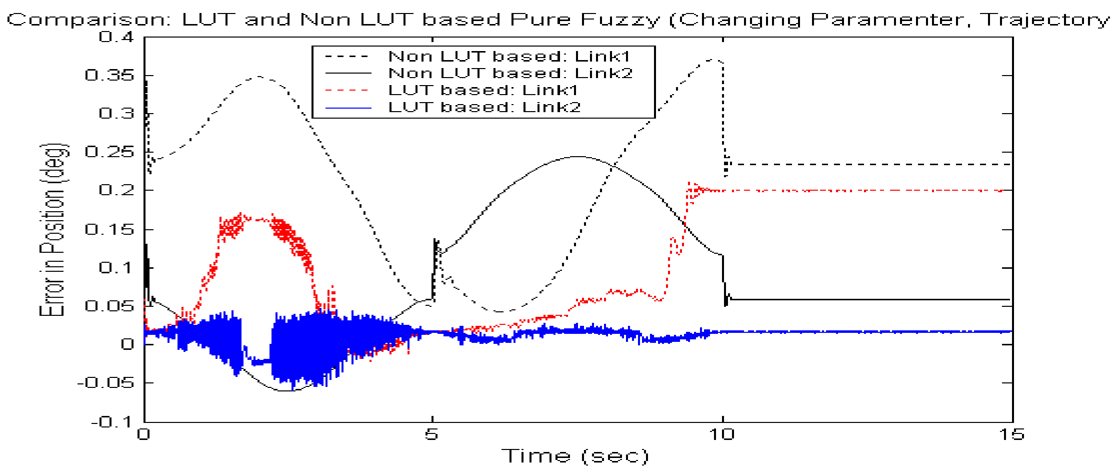


Fig 6.4.3.1(c) Comparison of errors for LUT and Non LUT based Fuzzy controllers (Changing Parameters)

Pure Fuzzy (Changing parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.2347	0.2345	0.1206	0.0590	0.2907	0.4088	0.1579	0.2774

Table 6.4.3.1. Errors for Fuzzy control (Non LUT based, Changing parameters)

6.4.4. ADAPTIVE FUZZY (CHANGING PARAMETERS)

The simulations carried out in this section are similar to those done in section 6.4.2 except for the fact that here the manipulator parameters change during the course of motion. The manipulator is assumed to pick and release load periodically. We run the simulation for two trajectories as used for all simulation. Fig. 6.4.4.1(a) shows the error profiles for first trajectory. Fig. 6.4.4.1(b) shows a comparison of errors between adaptive fuzzy and pure fuzzy control. As can be seen from the figure, the adaptive fuzzy controller gives improved performance compared to pure fuzzy controller. Errors for both links for adaptive fuzzy controller are reduced. However like the pure fuzzy controller, the adaptive fuzzy controller also has non-zero steady state errors for both links although reduced to some extent.

Fig. 6.4.4.1(c) shows the error profiles for the second trajectory. Fig. 6.4.4.1(d) shows a comparison of errors between adaptive fuzzy and pure fuzzy control for the second trajectory. Once again it is observed that adaptive fuzzy controller gives better performance with both r.m.s and maximum errors getting reduced, when compared to pure fuzzy controller. Table 6.4.4.1 lists the various error measures for the two trajectories for this controller.

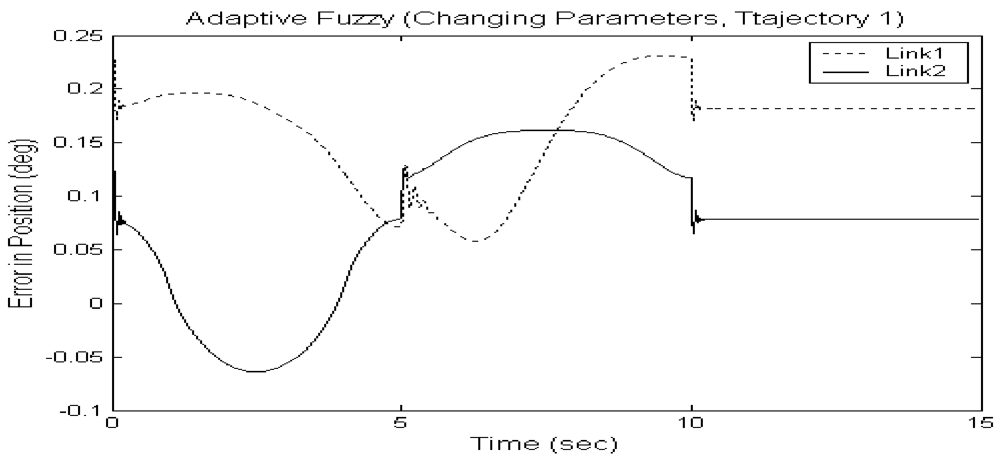


Fig. 6.4.4.1(a) Errors for Adaptive Fuzzy control (Changing parameters, Trajectory 1)

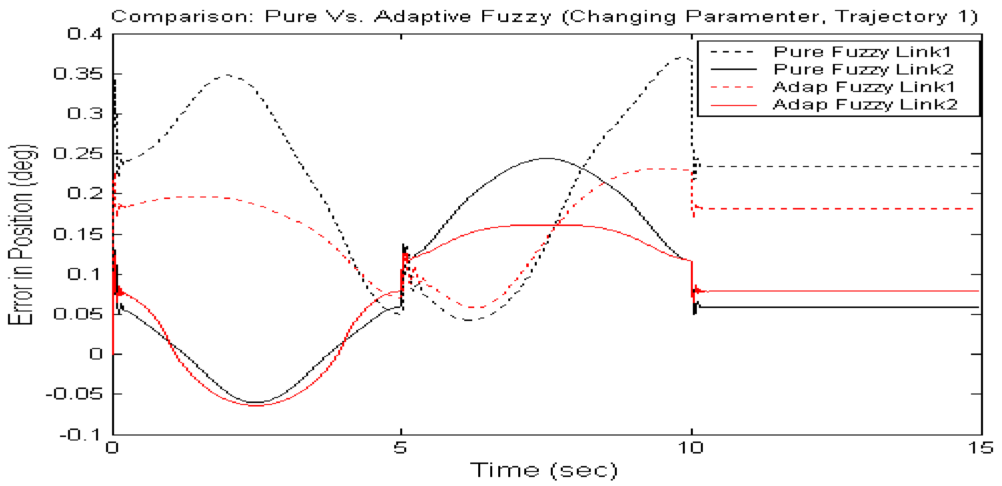


Fig. 6.4.4.1(b) Comparison of Errors for Adaptive Fuzzy Vs Fuzzy control (Changing parameters, Trajectory 1)

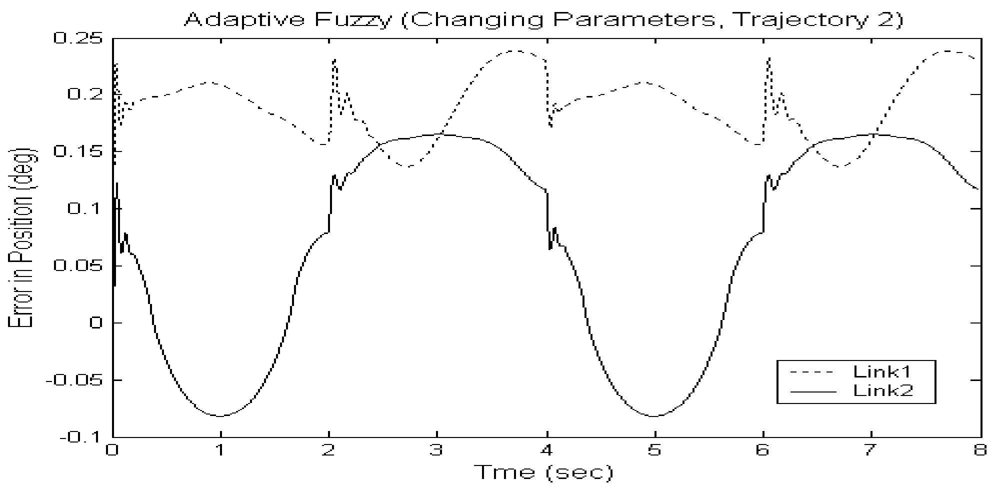


Fig. 6.4.4.1(c) Errors for Adaptive Fuzzy control (Changing parameters, Trajectory 2)

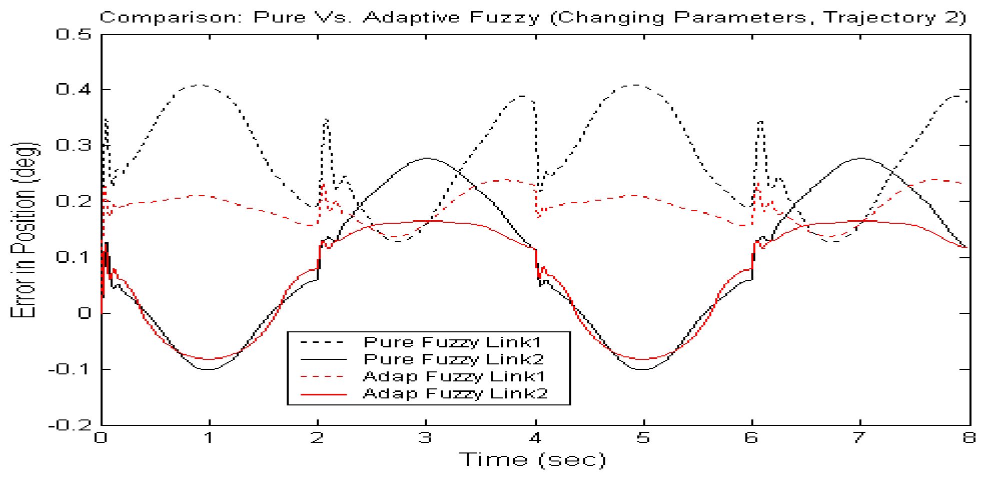


Fig. 6.4.4.1(d) Comparison of Errors for Adaptive Fuzzy Vs Fuzzy control (Changing parameters, Trajectory 2)

Adaptive Fuzzy (Changing parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.1707	0.1819	0.1010	0.0784	0.1923	0.2384	0.1147	0.1651

Table 6.4.4.1. Errors for Adaptive Fuzzy control (Changing parameters)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	0.2347	0.2345	0.1206	0.0590	0.2907	0.4088	0.1579	0.2774
2.	SOC Zero LUT	0.0245	0.0166	0.0333	-0.0169	0.0477	0.1059	0.0395	-0.1032
3.	SOC Nonzero LUT	0.0271	0.0166	0.0305	-0.0166	0.0380	-0.1014	0.0387	0.0898
4	STFC	0.1707	0.1819	0.1010	0.0784	0.1923	0.2384	0.1147	0.1651

Table 6.4.4.2. Errors for Fuzzy, SOC and STFC (Changing Parameters)

Table 6.4.4.2 lists the errors for the Pure Fuzzy, SOC and STFC for comparison. It can be seen that STFC performs well but is not as good as SOC.

6.4.5 COMPARISON OF PERFORMANCE

Following observations are made based on simulations carried out in this section (6.4):

1. The error profiles for non-lookup table based Fuzzy controller are much smoother compared to LUT based Fuzzy controllers. This is mainly because of smoother controller action and translates directly into a smoother manipulator motion. The errors for the LUT based controller however, are smaller.
2. The Adaptive Fuzzy controller improves the performance of Pure Fuzzy controller considerably. The performance improvement is more for the case when the manipulator parameters change during motion.
3. The error profiles for Adaptive Fuzzy controllers are much smooth compared to their SOC counterparts. This is once again mainly due to their non-lookup table based nature.

6.5 HYBRID FUZZY+INTEGRAL ERROR CONTROLLER (HFIE)

The PD self-tuning adaptive controller investigated in previous sections gives a reasonably good performance for both trajectories as far as rms errors are concerned. However it is observed that for the first trajectory, both the manipulator links end up with steady state errors of about 0.1 degrees. Moreover these errors do not decrease even if we consider the case of manipulator parameters not changing during the course of motion.

With a view to reduce this steady state error we propose and investigate a simple hybrid fuzzy controller. The block diagram of this controller is shown in Fig. 6.5.1. This controller consists of two parts. First is a simple non-lookup table based fuzzy controller. This controller is same as that investigated in section 6.4.1 and 6.4.3. It is also the same as the

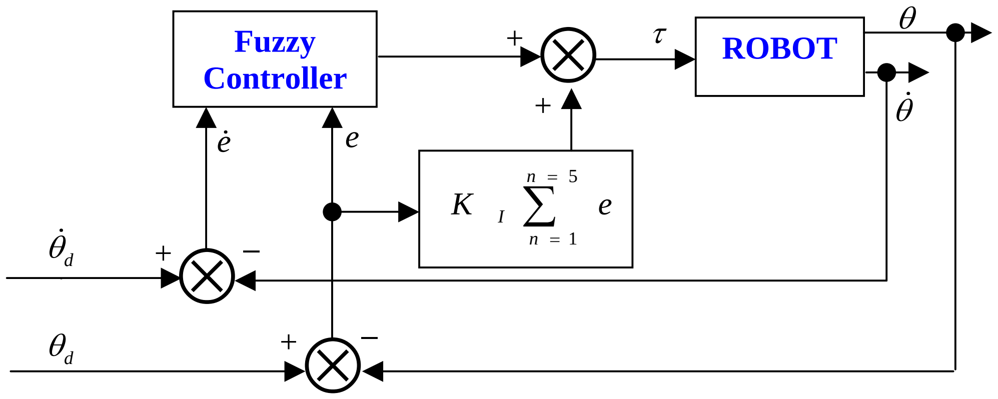


Fig. 6.5.1. Hybrid Fuzzy plus Integral error Controller block diagram

self-tuning adaptive fuzzy controller investigated in section 6.4.2 and 6.4.4 but with adaptation gain α fixed at a constant value of 1. Second is an integral error controller, as shown in Fig. 6.5.1. Usually the integral part of a controller produces an output, which is proportional to integral of error over the entire period of motion. But this simple addition of integral term also increase the order of system and might result in an unstable closed loop system. The controller that we propose ***does not*** perform summation (integration) of error over entire period of motion.

For our simulations the trajectory generator provides the controller with information about the desired position, velocity and acceleration $(\theta_d, \dot{\theta}_d, \ddot{\theta}_d)$ for each joint and keeps updating this information at the path update rate which has been chosen as 3ms (333Hz). The controller takes this information and compares it with the present (actual) position and velocity of joints $\left(\theta, \dot{\theta}\right)$, which are provided as feedback through the sensors. Based upon the error between the desired and actual values, the controller calculates a vector of joint torques $\left(\tau\right)$, which should be applied at respective joints by the actuators to minimize these errors. In the simulations, the control loop runs five times for every set point supplied by the trajectory generator. The integral action of our controller is limited to summing up these five errors for every set point provided by the trajectory generator. The sum of these errors is reset to zero every time the trajectory generator gives a new set point. This type of integral action cannot of course give zero values of steady state error but can nevertheless reduce them. Further the overall resulting controller does not suffer from danger of instability.

6.5.1. HFIE CONTROLLER (FIXED PARAMETERS)

We run the simulation for two trajectories for the case that manipulator parameters do not change during the entire duration of motion. Fig. 6.5.1.1(a) shows the error profiles for first trajectory. Fig. 6.5.1.1(b) shows a comparison of errors between adaptive fuzzy and HFIE control. As can be seen from the figures, the HFIE controller gives much improved performance compared to adaptive fuzzy controller. Errors for both links for HFIE controller are reduced. This includes not only the steady state errors but also the transient, rms and maximum errors as well. The steady state errors although reduced, are still not zero.

Fig. 6.5.1.1(c) shows the error profiles for the second trajectory. Fig. 6.5.1.1(d) shows a comparison of errors between adaptive fuzzy and HFIE control for the second trajectory. Once again it is observed that HFIE controller gives better performance with both r.m.s and maximum errors getting reduced, when compared to adaptive fuzzy controller. Table 6.5.1.1 lists the various error measures magnitudes for the two trajectories for this controller.

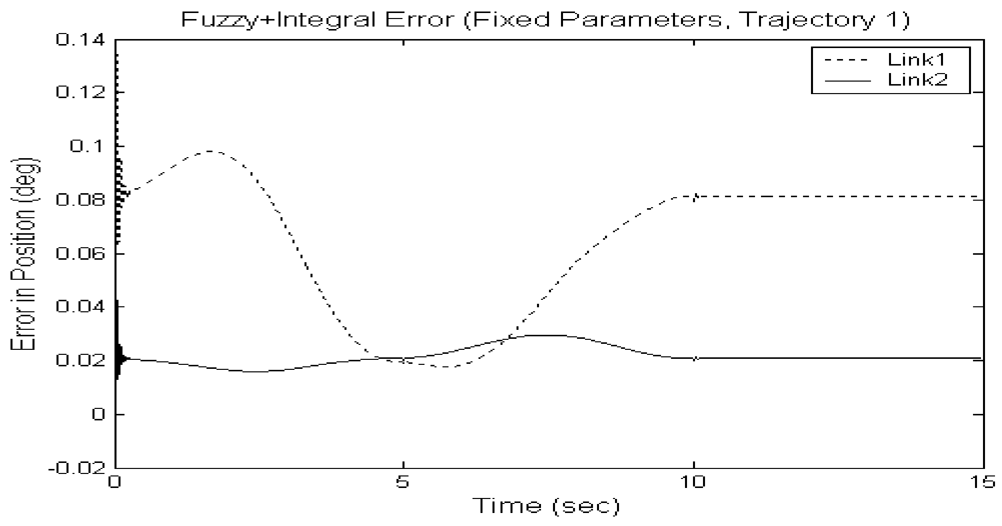


Fig. 6.5.1.1(a) Errors for HFIE controller (Fixed parameters, Trajectory 1)

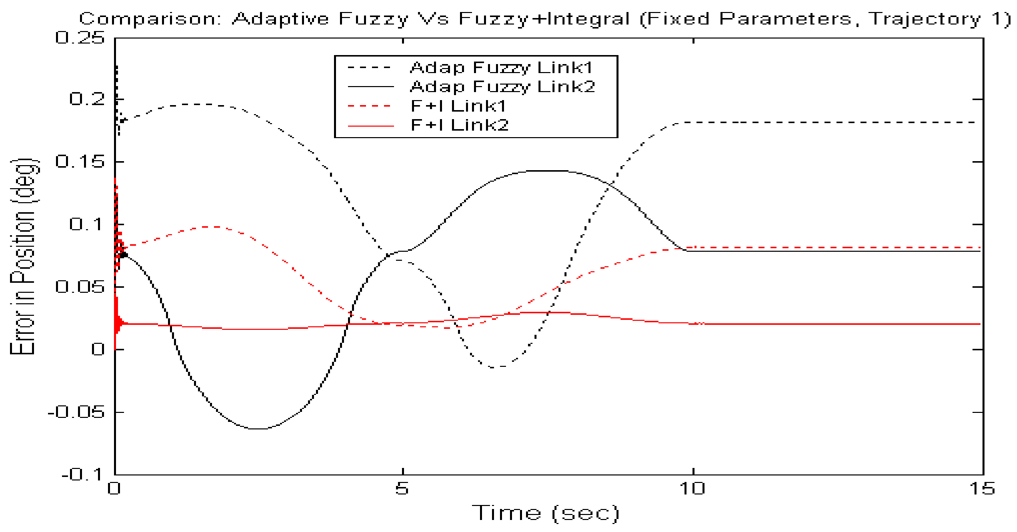


Fig. 6.5.1.1(b) Comparison of Errors for Adaptive Fuzzy Vs HFIE control (Fixed parameters, Trajectory 1)

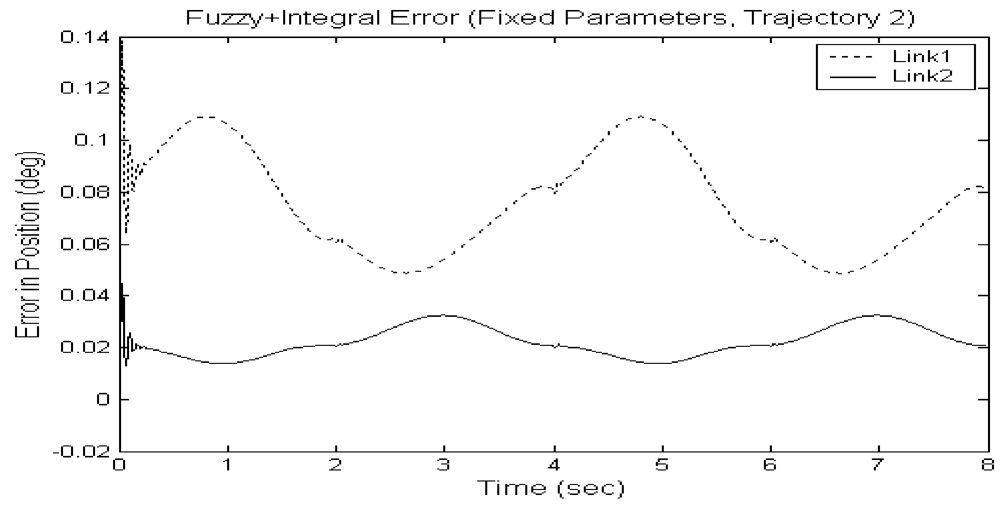


Fig. 6.5.1.1(c) Errors for HFIE controller (Fixed parameters, Trajectory 2)

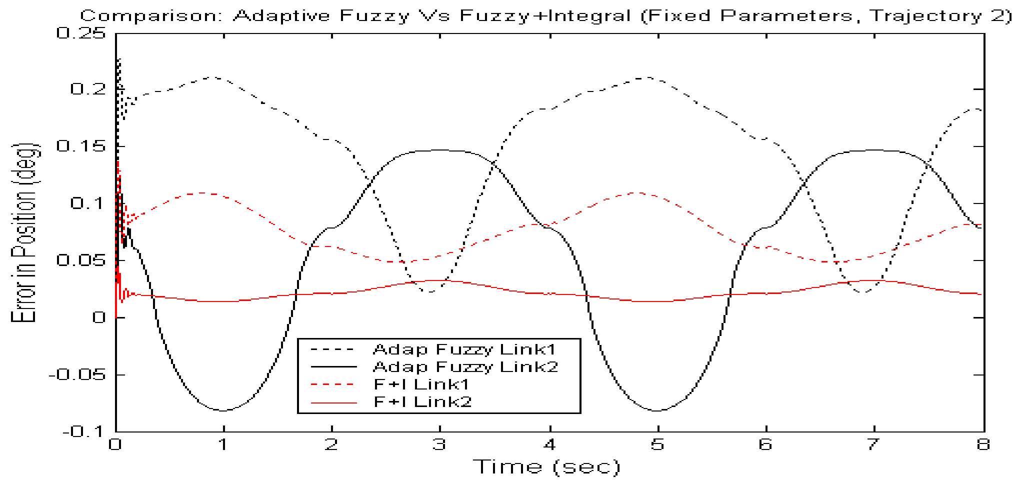


Fig. 6.5.1.1(d) Comparison of Errors for Adaptive Fuzzy Vs HFIE control (Fixed parameters, Trajectory 2)

Pure Fuzzy + Integral Error (Fixed parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.0701	0.0813	0.0218	0.0208	0.0784	0.1384	0.0229	0.0450

Table 6.5.1.1. Errors for HFIE control (Fixed parameters)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	0.2104	0.2345	0.0966	0.0590	0.2453	0.4088	0.1273	0.2308
2.	SOC Zero LUT	0.0338	0.0164	0.0306	-0.0163	0.0331	-0.1023	0.0409	-0.0923
3.	SOC Nonzero LUT	0.0315	0.0164	0.0306	-0.0163	0.0293	-0.1014	0.0385	0.0908
4.	STFC	0.1541	0.1819	0.0887	0.0784	0.1587	0.2272	0.0990	0.1468
5.	HFIE	0.0701	0.0813	0.0218	0.0208	0.0784	0.1384	0.0229	0.0450

Table 6.5.1.2. Errors for Fuzzy, SOC, STFC and HFIE (Fixed Parameters)

Table 6.5.1.2 lists the errors for the Pure Fuzzy, SOC, STFC and HFIE for comparison. It can be seen that HFIE performs better than STFC but is not as good as SOC.

6.5.2. HFIE CONTROLLER (CHANGING PARAMETERS)

We next do the simulation for two trajectories for the case that manipulator parameters change during the duration of motion. Fig. 6.5.2.1(a) shows the error profiles for first trajectory. Fig. 6.5.2.1(b) shows a comparison of errors between adaptive fuzzy and HFIE control. Once again it can be seen from the figures, that the HFIE controller gives much improved performance compared to adaptive fuzzy controller. Errors for both links for HFIE controller are reduced. This includes not only the steady state errors but the transient, rms and maximum errors as well. The steady state errors although reduced are still not zero.

Fig. 6.5.2.1(c) shows the error profiles for the second trajectory. Fig. 6.5.2.1(d) shows a comparison of errors between adaptive fuzzy and HFIE control for the second trajectory. As expected, the HFIE controller gives better performance with both r.m.s and maximum errors getting reduced, when compared to adaptive fuzzy controller. Table 6.5.2.1 lists the various error measures magnitudes for the two trajectories for this controller.

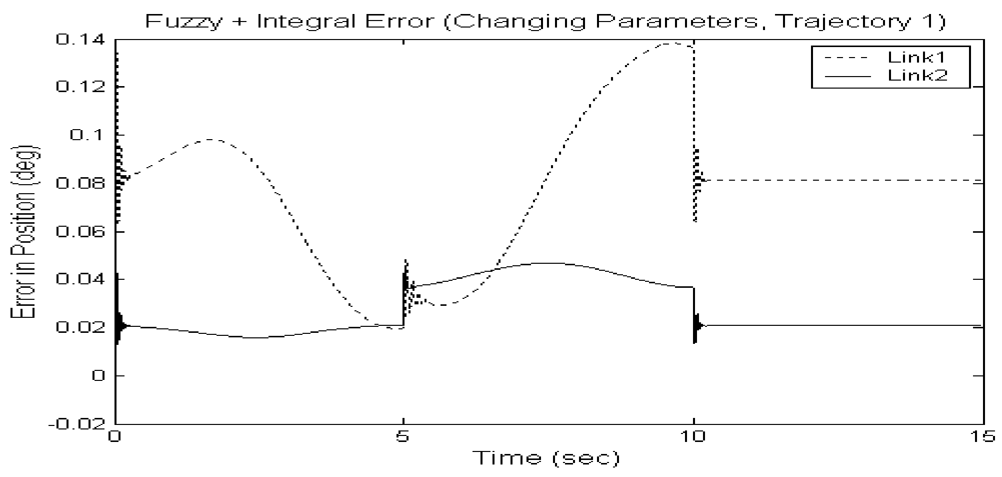


Fig. 6.5.2.1(a) Errors for HFIE controller (Changing parameters, Trajectory 1)

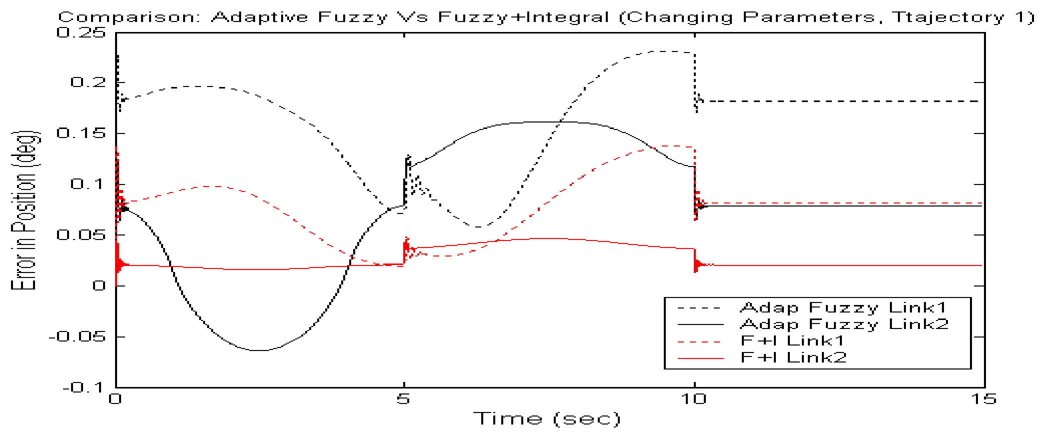


Fig. 6.5.2.1(b) Comparison of Errors for Adaptive Fuzzy Vs HFIE control (Changing parameters, Trajectory 1)

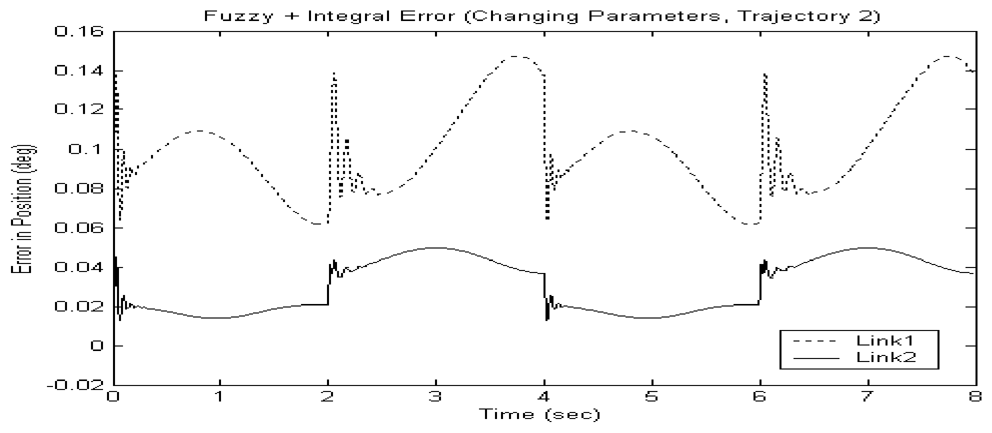


Fig. 6.5.2.1(c) Errors for HFIE controller (Changing parameters, Trajectory 2)

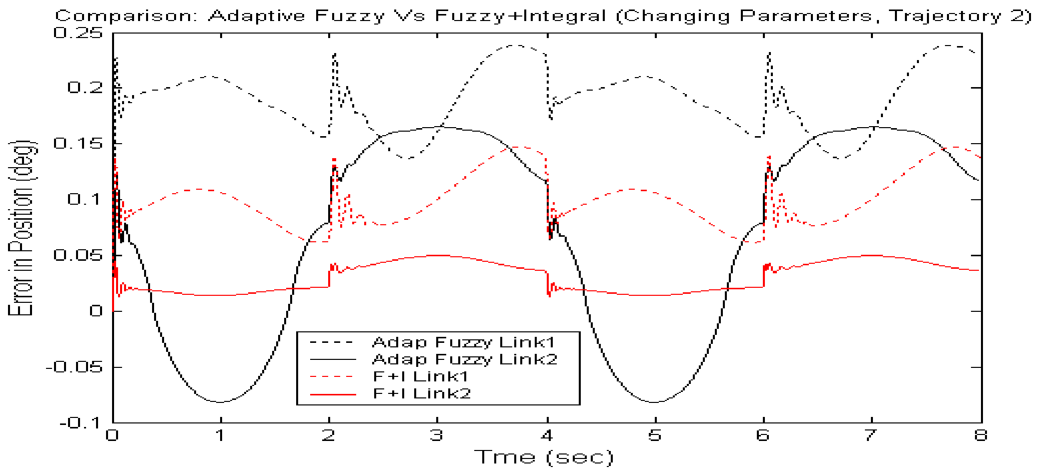


Fig. 6.5.2.1(d) Comparison of Errors for Adaptive Fuzzy Vs HFIE control (Changing parameters, Trajectory 2)

Pure Fuzzy + Integral Error (Changing parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.0381	0.0813	0.0292	0.0208	0.1026	0.1474	0.0335	0.0498

Table 6.5.2.1. Errors for HFIE control (Changing parameters)

S.No	**CONTROL STRATEGY**	**TRAJECTORY NO.1**				**TRAJECTORY NO.2**			
		link1 0°→ 90°→0°→		**link2 0°→ -90°→0°→**		**link1 0°→ 45°→0°→45°→0°**		**link2 0°→ -45°→0°→-45°→0°**	
		RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
1.	**Pure Fuzzy**	0.2347	0.2345	0.1206	0.0590	0.2907	0.4088	0.1579	0.2774
2.	**SOC Zero LUT**	0.0245	0.0166	0.0333	-0.0169	0.0477	0.1059	0.0395	-0.1032
3.	**SOC Nonzero LUT**	0.0271	0.0166	0.0305	-0.0166	0.0380	-0.1014	0.0387	0.0898
4.	**STFC**	0.1707	0.1819	0.1010	0.0784	0.1923	0.2384	0.1147	0.1651
5.	**HFIE**	0.0381	0.0813	0.0292	0.0208	0.1026	0.1474	0.0335	0.0498

Table 6.5.2.2. Errors for Fuzzy, SOC, STFC and HFIE (Changing Parameters)

Table 6.5.2.2 lists the errors for the Pure Fuzzy, SOC, STFC and HFIE for comparison. It can be seen that HFIE performance is almost comparable to that of SOC.

6.5.3 COMPARISON OF PERFORMANCE

Following observations are made based on simulations carried out in this section (6.5):

1. The modified integral action is an effective method to reducing the overall errors for manipulator trajectory tracking.
2. The proposed HFIE controller performs remarkably well when compared to STFC. The HFIE controller gives better performance with r.m.s., maximum and steady state errors all getting reduced, when compared to STFC
3. The improved performance of HFIE controller is further achieved with having much lesser number of calculations to perform compared to adaptive fuzzy controller. Although for our simulations we have kept the sampling rate for both adaptive fuzzy and HFIE controllers same, the much higher possible sampling rates for HFIE controller will improve its performance further.
4. It was also observed that errors for HFIE controller go down as we increase the integral gain constant Ki to a certain value. Any further increase in Ki results in errors increasing again. Hence there is an optimal value for the gain Ki.

6.6 COARSE/FINE ADAPTIVE FUZZY CONTROLLER (CFAF)

When a controller is required to operate under conditions of both large and small excursions of its inputs from their nominal values, it is convenient to use *two* or more sets of fuzzy rules to effect improved control [Dunlop et al. (1994)]. For large excursions of the controller input variables, coarse control is applied with the objective of forcing the plant to return to its nominal operating point as rapidly as possible. Accuracy of control is of secondary importance under these circumstances and only a few rules are required. When the plant variables reach some small region about the nominal operating point then fine control is applied. Here a new set of control rules necessary to effect the desired fine control actions are used and these involve a larger number of rules and fuzzy sets. Under normal operating conditions the controller uses fine control for small excursions about the nominal operating point.

An alternative way of achieving coarse-fine control is through *zooming* of the universe of discourse of each controller input variable. In this case the universe of discourse is varied, either in discrete regions in control space or smoothly as the plant approaches the desired operating point. This approach has been used to great effect for the control of high precision mechatronic devices and is investigated in this section for effectiveness in case of mechanical manipulator.

The basic controller is still the self-tuning adaptive fuzzy controller discussed in section 6.4. In that controller the output-scaling factor alone is adapted via the variable gain factor α. The characteristics of a PI- or PD-type fuzzy logic controller depends on both input and output scaling factors, i.e., for the best performance, simultaneous adjustment of both input and output scaling factors is more justified. To this effect the controller normalizes the position and velocity errors to limit them to domain [-6 6]. It then checks if the position and velocity errors are both within [-3 3]. If they are, then both the position and velocity error are doubled to provide the zooming effect. If the errors are not within [-3 3], then they are used as they are, without being doubled. This simple strategy results in much improved performance of the controller as discussed in the following sections.

6.6.1 COARSE/FINE ADAPTIVE FUZZY CONTROLLER (FIXED PARAMETERS)

We first run the simulation for two trajectories for the case that manipulator parameters do not change during the entire duration of motion. Fig. 6.6.1.1(a) shows the error profiles for first trajectory. Fig. 6.6.1.1(b) shows a comparison of errors between adaptive fuzzy and CFAF control. As can be seen from the figures, the CFAF controller gives much improved performance compared to adaptive fuzzy controller. Errors for both links for CFAF controller are reduced. This includes not only the steady state errors but the transient, rms and maximum errors as well.

Fig. 6.6.1.1(c) shows the error profiles for the second trajectory. Fig. 6.6.1.1(d) shows a comparison of errors between adaptive fuzzy and CFAF control for the second trajectory. Once again it is observed that CFAF controller gives better performance with both r.m.s and maximum errors getting reduced, when compared to adaptive fuzzy controller. Table 6.6.1.1 lists the various error measures magnitudes for the two trajectories for this controller.

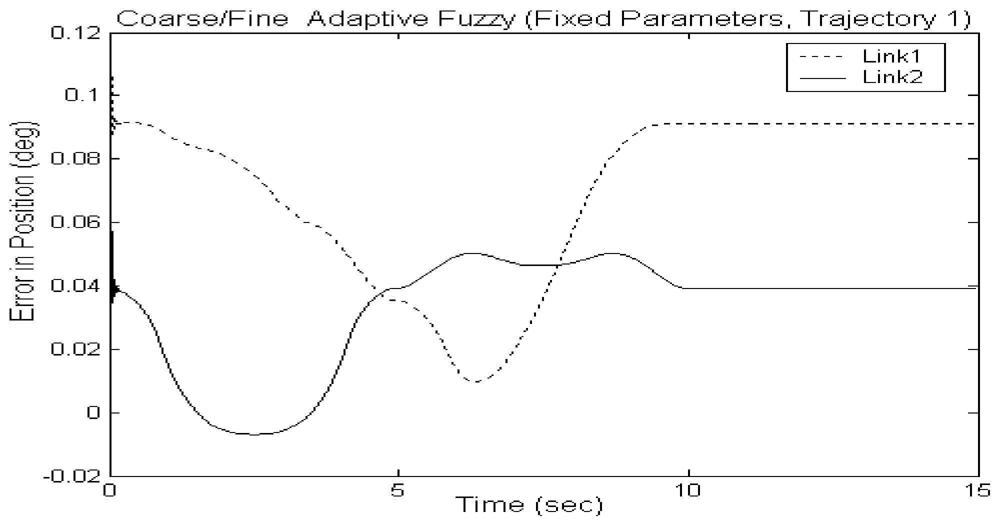


Fig. 6.6.1.1(a) Errors for CFAF controller (Fixed parameters, Trajectory 1)

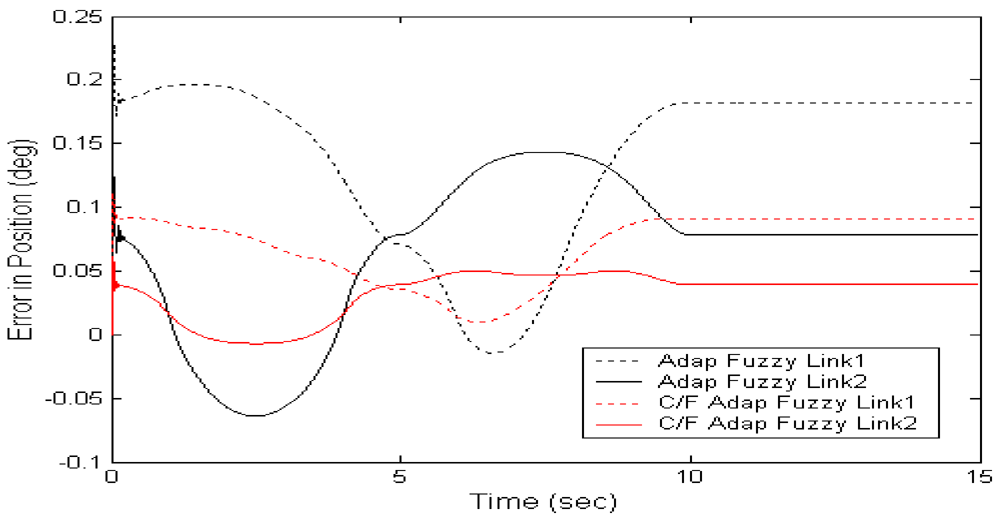


Fig. 6.6.1.1(b) Comparison of Errors for Adaptive Fuzzy Vs CFAF control (Fixed parameters, Trajectory 1)

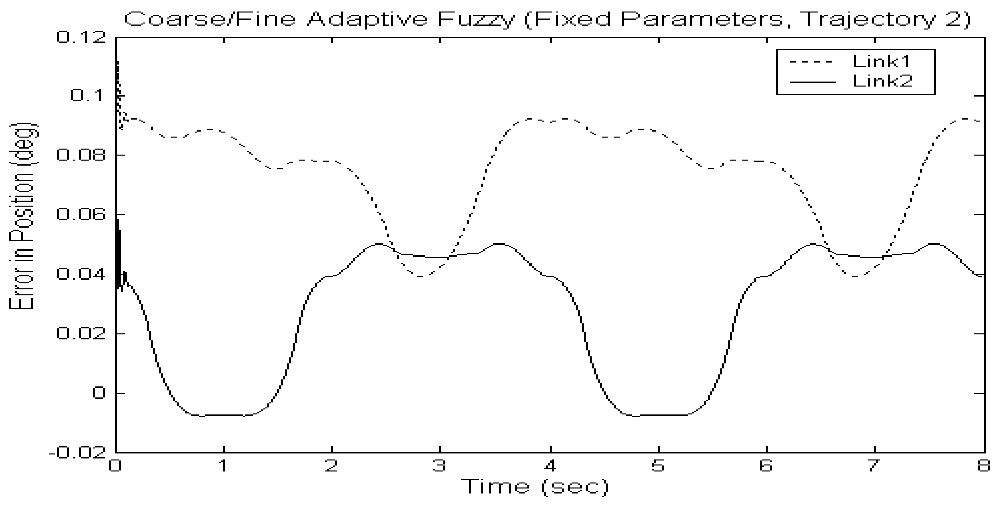


Fig. 6.6.1.1(c) Errors for CFAF controller (Fixed parameters, Trajectory 2)

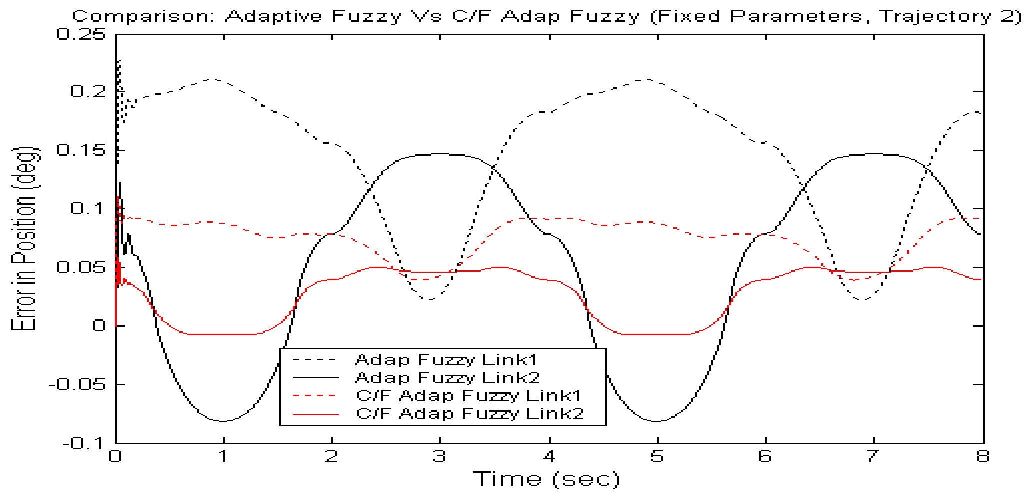


Fig. 6.6.1.1(d) Comparison of Errors for Adaptive Fuzzy Vs CFAF control (Fixed parameters, Trajectory 2)

CF Adaptive Fuzzy (Fixed parameters) Errors (deg)							
TRAJECTORY NO.1				**TRAJECTORY NO.2**			
Link1 0°→ 90°→0°		**Link2 0°→ -90°→0°**		**Link1 0°→ 45°→0°→**		**Link2 0°→ -45°→0°→**	
RMS	**SS**	**RMS**	**SS**	**RMS**	**MAX**	**RMS**	**MAX**
0.0747	0.0910	0.0373	0.0392	0.0766	0.1115	0.0358	0.0583

Table 6.6.1.1. Errors for CFAF control (Fixed parameters)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	0.2104	0.2345	0.0966	0.0590	0.2453	0.4088	0.1273	0.2308
2.	SOC Zero LUT	0.0338	0.0164	0.0306	-0.0163	0.0331	-0.1023	0.0409	-0.0923
3.	SOC Nonzero LUT	0.0315	0.0164	0.0306	-0.0163	0.0293	-0.1014	0.0385	0.0908
4.	STFC	0.1541	0.1819	0.0887	0.0784	0.1587	0.2272	0.0990	0.1468
5.	HFIE	0.0701	0.0813	0.0218	0.0208	0.0784	0.1384	0.0229	0.0450
6.	CFAF	0.0747	0.0910	0.0373	0.0392	0.0766	0.1115	0.0358	0.0583

Table 6.6.1.2. Errors for Fuzzy, SOC, STFC, HFIE and CFAF (Fixed Parameters)

Table 6.6.1.2 lists the errors for the Pure Fuzzy, SOC, STFC, HFIE and CFAF for comparison. It can be seen that CFAF performs better than STFC but is not as good as SOC or HFIE.

6.6.2 COARSE/FINE ADAPTIVE FUZZY CONTROLLER (CHANGING PARAMETERS)

We next do the simulation for two trajectories for the case that manipulator parameters change during the entire duration of motion. Fig. 6.6.2.1(a) shows the error profiles for first trajectory. Fig. 6.6.2.1(b) shows a comparison of errors between adaptive fuzzy and CFAF control. Once again it can be seen from the figures, that the CFAF controller gives much improved performance compared to adaptive fuzzy controller. Errors for both links for CFAF controller are reduced. This includes not only the steady state errors but the transient, r.m.s and maximum errors as well.

Fig. 6.6.2.1(c) shows the error profiles for the second trajectory. Fig. 6.6.2.1(d) shows a comparison of errors between adaptive fuzzy and CFAF control for the second trajectory. As expected, the CFAF controller gives better performance with both r.m.s and maximum errors getting reduced, when compared to adaptive fuzzy controller. Table 6.6.2.1 lists the various error measures magnitudes for the two trajectories for this controller.

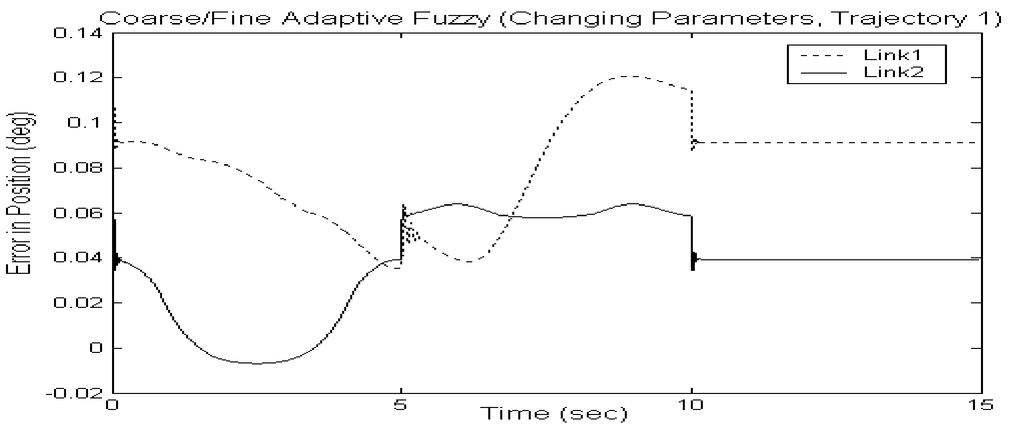


Fig. 6.6.2.1(a) Errors for CFAF controller (Changing parameters, Trajectory 1)

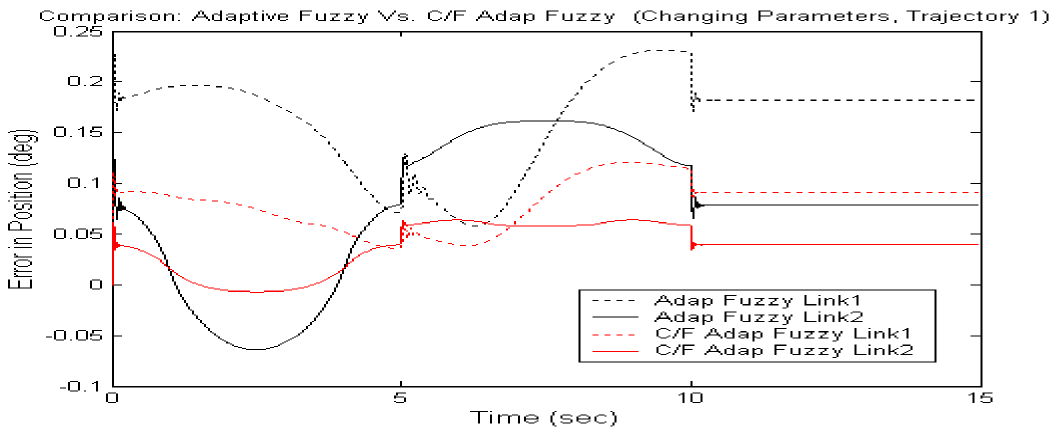


Fig. 6.6.2.1(b) Comparison of Errors for Adaptive Fuzzy Vs CFAF control (Changing parameters, Trajectory 1)

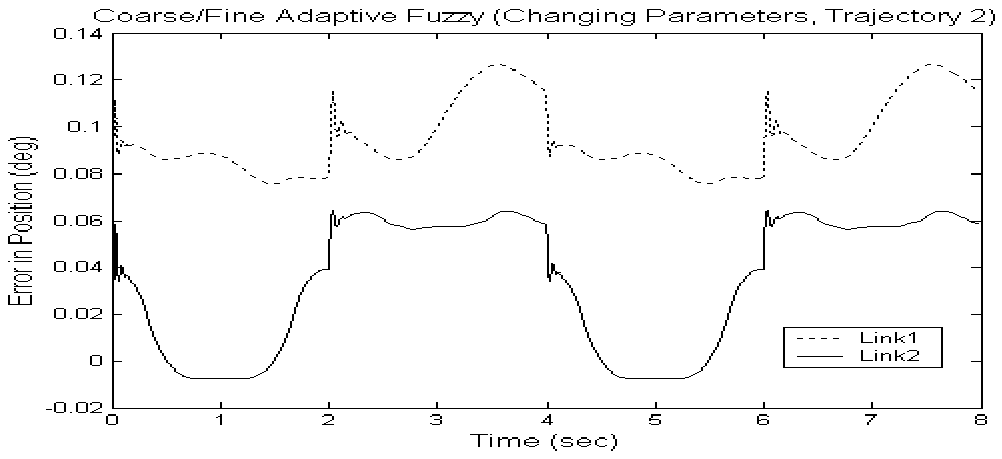


Fig. 6.6.2.1(c) Errors for CFAF controller (Changing parameters, Trajectory 2)

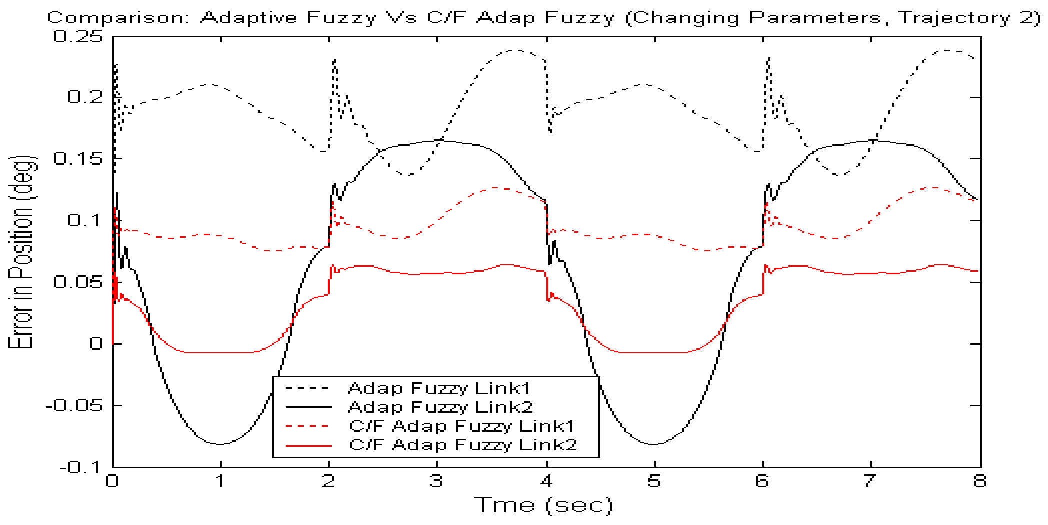


Fig. 6.6.2.1(d) Comparison of Errors for Adaptive Fuzzy Vs CFAF control (Changing parameters, Trajectory 2)

CF Adaptive Fuzzy (Changing parameters) Errors (deg)							
TRAJECTORY NO.1				TRAJECTORY NO.2			
Link1 0°→ 90°→0°		Link2 0°→ -90°→0°		Link1 0°→ 45°→0°→		Link2 0°→ -45°→0°→	
RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
0.0846	0.0910	0.0435	0.0392	0.0961	0.1266	0.0448	0.0644

Table 6.6.2.1 Errors for CFAF control (Changing parameters)

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
1.	Pure Fuzzy	0.2347	0.2345	0.1206	0.0590	0.2907	0.4088	0.1579	0.2774
2.	SOC Zero LUT	0.0245	0.0166	0.0333	-0.0169	0.0477	0.1059	0.0395	-0.1032
3.	SOC Nonzero LUT	0.0271	0.0166	0.0305	-0.0166	0.0380	-0.1014	0.0387	0.0898
4.	STFC	0.1707	0.1819	0.1010	0.0784	0.1923	0.2384	0.1147	0.1651
5.	HFIE	0.0381	0.0813	0.0292	0.0208	0.1026	0.1474	0.0335	0.0498
6.	CFAF	0.0846	0.0910	0.0435	0.0392	0.0961	0.1266	0.0448	0.0644

Table 6.6.2.2. Errors for Fuzzy, SOC, STFC, HFIE and CFAF (Changing Parameters)

Table 6.6.2.2 lists the errors for the Pure Fuzzy, SOC, STFC, HFIE and CFAF for comparison. It can be seen that CFAF performance is better than STFC but not as good as HFIE or SOC.

6.6.3 COMPARISON OF PERFORMANCE

Following observations are made based on simulations carried out in this section (6.6):

1. It is observed that the CFAF controller gives much improved performance compared to STFC. Errors for both links for CFAF controller are reduced for fixed as well as changing parameters case.
2. CFAF does not involve any additional computational burden on the controller. The additional complexity is only in terms of few additional if-then-else statements.
3. The CFAF controller however is still not as good as HFIE controller or SOC.

6.7 CONCLUDING REMARKS

In this chapter we investigated the efficacy of Fuzzy control techniques for manipulator control. We investigated both the lookup table based controller and the conventional fuzzy controller. We found that the fuzzy controller on its own does not give a performance as good as Adaptive controllers. However the Self-Organizing and Self-Tuning versions of the fuzzy controllers give very good performance.

We also investigated the performance of some new hybrid fuzzy controllers. It was found that hybrid conventional-fuzzy controllers give a substantial performance improvement. On the other hand the Hybrid adaptive-fuzzy controllers do not give much performance improvement.

The HFIE and CFAF controllers are also very competitive in their performance. On the whole Fuzzy control is a viable alternative to adaptive control both in terms of good performance and reduced complexity of computations required.

CHAPTER VII

CONCLUSIONS AND RECOMMENDATIONS

7.1 CONCLUSIONS

In this thesis we have attempted to explore the behavior of few robot manipulator control strategies. The controllers investigated in this thesis are of conventional, adaptive and fuzzy kind and their combinations.

We have also suggested some modifications to the existing control strategies, which could lead to improvement in their performance. This is mainly in terms of introducing a *modified* integral error compensation. We have also suggested some new control strategies, like HFIE, which give good performance with minimal computational complexity.

For the sake of ‘just’ comparison, we have kept the manipulator model and test trajectories same throughout. The detailed conclusions for the study carried out on them are already presented in Chapters 4, 5 and 6 respectively. Here we state the main conclusions and present them section wise.

CONVENTIONAL CONTROL

Various manipulator control strategies belonging to the conventional strategies were tested in this section. These included the PD, PID, Computed Torque, Feed Forward Inverse Dynamics and Critically Damped Inverse Dynamics control. These strategies were tested for two different trajectories. In the first trajectory the manipulator had to essentially position itself accurately and in the second trajectory the manipulator is essentially involved in a pick and place task. The first trajectory allows us to investigate the steady state behavior of the controller while the second trajectory allows investigation of transient behavior of the controller more rigorously.

The first situation tested was where the manipulator parameters do not change during the entire trajectory. Under this situation it was seen that although the manipulator parameters are not changing, the fixed gain PD controller perform poorly. This is mainly because the manipulator model used does not include any gearings at the joints, a case which is true for high-speed robots. The PID controller gives much better performance than PD controller but can become unstable very easily even with low gain values. This clearly points to the fact that for high speed and high precision operations these simple non-model based schemes cannot be used.

Amongst the model-based controllers, the performance is greatly dependent upon exactness with which the model structure and its parameters are known. If the model and its parameters are known exactly, the model-based controllers give very good performance. This performance degrades rapidly with increasing inaccuracies of measurement and modeling. So much so that the performance may degrade below that of PD controller. Under the situation of inexactness of modeling and parameter estimation, the CDID controller gives best performance. This is mainly because this controller uses reference values of trajectory rather than actual ones for model computation.

The second situation, which was tested, was when the manipulator picks up an unknown load during the course of its motion and releases it later. This effectively means that the manipulator parameters change during motion and the magnitude of change is unknown. Under this situation, the performance of PD controller and CT controller degrades appreciably. The FFID and CDID controllers also loose on their performance but not as much as PD or CT controllers. This clearly once again highlights the merit of using reference and desired trajectory values, rather than actual ones, for model computation.

Lastly in this section we proposed the use of a modified integral error compensation with the model based controllers for improving their performance. The integral action sums up the errors for every five iterations of control loop for a given set point. When the set point changes, the error summation is reset to zero. This modified integral action does not suffer from stability related problems and greatly improves the performance of the model-based controllers. Moreover as the integral action is implemented entirely in software, it can be 'switched off' in case the manipulator performance degrades below a datum level. It was seen that the performance of CT and FFID controllers improved appreciably with the inclusion of modified integral action, while the performance of CDID controller only improved marginally.

ADAPTIVE CONTROL

To further improve the performance of manipulator controller, we need a controller, which can change or 'learn' the manipulator parameters which are not known exactly because of inaccuracies of estimation or because they change as the manipulator picks up unknown loads in its work environment. Alternatively the controller could adjust its gains as the parameters change with the objective of driving the tracking errors to zero. This is essentially what the adaptive controllers do.

In this section we investigated the behavior of three different adaptive controllers widely quoted in literature for manipulator control. These controllers are the adaptive version of CDID or the ACDID controller, a Model Reference Adaptive Controller (MRAC) and a Decentralized Adaptive Controller (DAC).

We first investigated the performance of these controllers for the situation that the manipulator parameters change during the motion as it picks up and releases load. This case was further analyzed for the situations of warm start (some estimate of parameters known) and cold start (no estimate of parameters is available). It was seen that the adaptive controllers out perform the conventional controllers for all situations. Even the conventional controllers with modified integral action do not perform as well as these adaptive controllers. Further it was seen that the model-based controllers (ACDID and MRAC) perform better for the warm start case. This clearly shows the importance of using 'good' estimates of parameter, whenever available, for performance improvement.

We also investigated the effect of adding modified integral action to these controllers just like we did for conventional controllers. It was seen that addition of this action improves the performance of adaptive controllers both in terms of steady state and transient errors. Overall the ACDID controller with modified integral action gave best performance amongst adaptive controllers.

FUZZY CONTROL

Although the adaptive controllers give very good performance, their practical implementation is not easy because of their computational complexity. This means that high-speed processors need to be used to get sufficiently high sampling rates. This directly translates into high cost of implementation. Hence it is worthwhile to investigate the Fuzzy control method, which promises good performance at much less computational complexity. In this thesis we investigated many different fuzzy controllers, which include self-organizing, self-tuning, hybrid fuzzy etc. We have also proposed some new fuzzy controllers for the robot manipulators.

In this section we proposed and tested several new hybrid fuzzy controllers. These controllers are essentially a combination of pure fuzzy (LUT based) and another conventional or adaptive controller. The control action of the two controllers is summed up to produce the final actuating signal.

It was seen that the performance of conventional controllers studied earlier (CT, CDID and FFID) improves considerably when they are combined in the hybrid structure with fuzzy controller. The performance of these hybrid fuzzy/conventional controllers is also better than conventional controllers with modified integral error compensation. As the fuzzy controller is LUT based, it adds only a minimal computational burden. However this burden is more than that incurred by adding only the modified integral error compensation.

The hybrid fuzzy/adaptive controllers also showed some performance improvement. However this improvement is not as marked as the previous case of hybrid fuzzy/conventional controllers. Moreover these controllers are computationally even more intensive than the adaptive controller alone. Hence we conclude that it is not worthwhile using such controllers. We found that CDID + Fuzzy controller performs almost as good as ACDID with modified integral error compensation with lesser computational complexity. Hence these hybrid fuzzy/conventional controllers may provide a viable alternative to adaptive controllers.

The second variant of fuzzy controller investigated is the fuzzy self-organizing controller (SOC). Here the controller builds up its own LUT starting from all zero entries or modifies the already existing entries according to error profiles. This controller gives an outstanding performance that was not matched by any other controller of the fuzzy category. The performance was good even if the LUT had all entries as zero initially. Of course the performance improved further when we started with non-zero values in LUT.

The self-tuning fuzzy controller (STFC) was also studied. This controller adapts its output denormalization gain online depending on the present error and its derivative. The implementation of this controller is non-LUT based. It was seen that this controller improves the performance beyond that of pure fuzzy controller but it is not at par with the self-organizing controller. However the trajectory for STFC is much smoother compared to that of SOC, mainly because of its non-LUT based nature. We suggest a modification to STFC such that effectively both the input and output gains of the fuzzy controller can be changed. This is achieved by zooming the universe of discourse. This controller is known as coarse/fine adaptive fuzzy controller (CFAF). The CFAF improves on the performance of STFC with minimal additional computational burden. However the performance is still not as good as that of SOC.

Lastly we tried out a very simple hybrid architecture where we combined the fuzzy controller with the modified integral error compensation (HFIE). To our surprise the HFIE gives a very good performance next only to SOC. It also outperforms the STFC and CFAF controllers. The computational burden is lesser than SOC.

Table 7.1.1 lists the errors for various controllers under similar situation of changing parameters. The shaded cells indicate the best performer in a category. Amongst all controllers the hybrid ACDID + Fuzzy controller gives the best performance which is closely matched by ACDID + IE controller.

S.No	CONTROL STRATEGY	TRAJECTORY NO.1				TRAJECTORY NO.2			
		link1 0°→ 90°→0°→		link2 0°→ -90°→0°→		link1 0°→ 45°→0°→45°→0°		link2 0°→ -45°→0°→-45°→0°	
		RMS	SS	RMS	SS	RMS	MAX	RMS	MAX
CONVENTIONAL CONTROL									
1.	PD Control	7.1415	9.4447	2.3758	2.6345	6.8624	9.5918	2.1554	3.3037
2.	PID control	2.9360	3.8290	0.9742	1.0737	2.8617	4.1457	0.9264	1.4036
3.	CT	6.1340	8.5617	4.6260	-4.7008	4.9007	12.2731	4.4288	5.0733
4.	FFID	2.5940	3.6395	0.8689	1.1186	2.0083	4.1268	0.6431	1.0846
5.	CDID	0.0616	0.0695	0.0211	0.0214	0.0707	0.1244	0.0265	0.0509
6.	CT+IE*	1.2919	1.7460	0.4019	0.4721	1.1894	2.3245	0.3118	0.6798
7.	FFID+IE*	1.0687	1.4588	0.3508	0.4488	0.9466	1.8639	0.2759	0.4419
8.	CDID+IE*	0.0604	0.0676	0.0210	0.0208	0.0697	0.1226	0.0268	0.0513
ADAPTIVE CONTROL									
9.	ACDID	0.0247	0.0001	0.0238	-0.0002	0.0391	0.0772	0.0344	0.0642
10.	ACDID+IE*	0.0213	0.0	0.0212	0.0	0.0323	0.0508	0.0325	0.0745
11.	MRAC	0.9283	1.3691	0.2827	0.4039	0.6935	1.4323	0.2628	0.5390
12.	MRAC+IE*	0.0321	0.0	0.0200	0.0	0.0559	0.1800	0.0270	0.0732
13.	DAC	0.0613	0.0043	0.0259	-0.0007	0.0674	0.1387	0.0284	0.0600
14.	DAC+IE*	0.0313	0.0418	0.0220	0.0117	0.0371	0.0798	0.0329	0.0507
HYBRID FUZZY									
15.	Pure Fuzzy	1.3926	2.0000	0.2289	0.1670	1.2304	2.6153	0.4585	1.1183
16.	CT+Fuzzy	0.1761	0.1670	0.2361	0.1667	0.6324	1.8571	0.2895	-1.0030
17.	FFID+Fuzzy	0.1534	0.1667	0.2716	0.1663	0.1471	0.2104	0.3702	-0.9186
18.	CDID+Fuzzy	0.0216	0.0	0.0212	0.0	0.0343	-0.0610	0.0325	0.0507
19.	ACDID+Fuzzy	0.0130	0.0	0.0119	-0.0001	0.0291	0.0599	0.0226	-0.0564
20.	MRAC+Fuzzy	0.2472	0.3565	0.1261	0.1673	0.2350	0.3648	0.1423	-0.2403
21.	DAC+Fuzzy	0.0657	0.0101	0.0351	0.0001	0.0567	0.1103	0.0323	0.0791
ADAPTIVE FUZZY									
22.	SOC ZLUT	0.0245	0.0166	0.0333	-0.0169	0.0477	0.1059	0.0395	-0.1032
23.	SOC NZLUT	0.0271	0.0166	0.0305	-0.0166	0.0380	-0.1014	0.0387	0.0898
24.	STFC	0.1707	0.1819	0.1010	0.0784	0.1923	0.2384	0.1147	0.1651
25.	HFIE	0.0381	0.0813	0.0292	0.0208	0.1026	0.1474	0.0335	0.0498
26.	CFAF	0.0846	0.0910	0.0435	0.0392	0.0961	0.1266	0.0448	0.0644

Table 7.1.1 Errors for all controllers for parameter changing case

7.2 SUMMARY OF CONTRIBUTIONS

In brief we have contributed to the body of existing knowledge in the field of manipulator control in the following way:

1. Done a comparative study of some conventional model based and non-model based controllers. This was done for different situations like manipulator model known exactly, manipulator model not known exactly etc.
2. Proposed and investigated the effect of including a modified integral action to the all model based conventional controllers
3. Showed that including the modified integral error compensation improves the performance of conventional controllers appreciably
4. Showed that CDID controller with modified integral action is the best performer in conventional controllers category
5. Done a comparative study of some existing adaptive controllers for various situations like warm start and cold start etc.
6. Proposed and investigated the effect of including a modified integral action to the adaptive controllers
7. Showed that including the modified integral error compensation improves the performance of adaptive controllers
8. Showed that ACDID controller with modified integral action is the best performer in adaptive controllers category
9. Proposed and investigated some new hybrid fuzzy controllers
10. Done a comparative study of different fuzzy controllers like self-organizing, self-tuning, hybrid etc.
11. Showed that the self-organizing controller with a non zero lookup table is the best performer in fuzzy controllers category

7.3 RECOMMENDATIONS FOR FUTURE WORK

- In any practical implementation of a robot controller, the actuator dynamics plays an important role. Actuator torque saturation for example, would have a direct impact on stability of the system. Not much work has been done in this direction. A complete manipulator model would definitely include not only the robot dynamics but the actuator dynamics as well. Work may be done in the direction of testing the behavior of these controllers with actuator dynamics incorporated in the model.

- With the recent availability of lightweight and precision acceleration sensors, work needs to done on developing control algorithms, which use acceleration feedback. It may be analyzed if controllers using acceleration feedback give better performance than the controllers that do not use it.

- Any simulation can at best be a pointer to expected behavior of a controller. It can never be a substitute to testing by experimentation. All the control algorithms simulated in the thesis may be experimentally tested as well.

- Much work has recently been done in the area of neural networks based controllers for robot manipulators [Patino et al. (2002), Horne et al. (1990)]. A comparative study of these controllers with conventional, adaptive and fuzzy controllers may be carried out.

- Many genetic algorithm based robot controllers have also been proposed of late [Alander (1998)]. A comparative study of these genetic algorithm based controllers with conventional, adaptive, fuzzy and neural networks based controllers may be carried out.

- These various controllers may be compared not only for their trajectory tracking performance but also for their computational complexity. More computational complexity directly implies costlier practical implementation.

LIST OF PUBLICATIONS

1. **Conventional Control Strategies for Robot Manipulators: A Simulation Study**, Proceedings of International Conference on Computer Applications in Electrical Engineering-Recent Advances, IIT, Roorkee, October 2005, pp 362-367.

2. **A Comparative Study of Few Conventional and Adaptive Control Algorithms for Manipulator Control**, Proceedings of the 2nd Indian International Conference on Artificial Intelligence, Pune, December 2005, pp 252-264.

3. **Comparative Study of Some Adaptive Fuzzy Algorithms for Manipulator Control,** International Journal of Computational Intelligence, 2006, Vol.3, Number 4, pp 303-311.

4. **Comparative Study of Some New Hybrid Fuzzy Algorithms for Manipulator Control,** Journal of Control Science and Engineering, Vol. 2007, Article ID 75653, 10 pages, 2007. doi: 10.1155/2007/75653.

REFERENCES

Abdessemed, F., Benmahammed, K., ‘A two layer robot control design’, Proceedings of IEEE International Conference on Fuzzy Systems, 4-9 May 1998, Vol. 1, Pages: 522 – 527

Alander, J. T., ‘ An indexed bibliography on genetic algorithms in robotics’, Report series No. 94-1-ROBOT, University of Vasaa, Finland, 1998

Albertos, P., Olivares, M., Sala, A., ‘ Fuzzy logic based look-up table controller with generalization’, Proceedings of the American Control Conference, June 2000, Pages: 1949-1953

An, C.H., Atkeson, C.G., Griffiths, J.D., Hollerbach, J.M., ‘Experimental evaluation of feed forward and computed torque control’, IEEE Transactions on Robotics and Automation, June 1989, Vol. 5, Issue: 3, Pages: 368 – 373

Astrom, K.J., Wittenmark, B., ‘Adaptive control’, Addison Wesley, 1995

Banerjee, S., Woo, P. Y., ‘Fuzzy logic control of robot manipulator’, Proceedings of IEEE Conference on Control Applications, 13-16 Sept. 1993, Vol.1, Pages: 87 – 88

Bonissone, P.P., Chiang, K.H., ‘Fuzzy logic controllers from development to deployment’, Proceedings of IEEE International Conference on Neural Networks, 28 March-1 April 1993, Vol.1, Pages: 610 – 619

Brehm, T., Rattan, K.S., ‘Hybrid fuzzy logic PID Controller’, Proceedings of the IEEE Aerospace and Electronics Conference, 24-28 May 1993, Vol.2, Pages: 807 – 813

Burdet, E., Codourey, A., Rey, L., ‘Experimental evaluation of nonlinear adaptive controllers’, IEEE Control Systems Magazine, April 1998, Vol. 18, Issue: 2, Pages: 39 – 47

Burkan, R., ‘Design of an adaptive control law using trigonometric functions for robot manipulators’, Robotica , 2005, Vol. 23, Pages: 93–99.

Butkiewicz, B. S., ‘System with hybrid fuzzy-conventional PID controller’, IEEE International Conference on Systems, Man, and Cybernetics, 2000, Vol. 5, Pages: 3705-3709

Campa, R., Kelly, R., Garcia, E., ‘On stability of the resolved acceleration control’, Proceedings of IEEE International Conference on Robotics and Automation, 2001, Vol. 4, Pages: 3523 – 3528

Chatterjee, A., Watanabe, K., ‘An adaptive fuzzy strategy for motion control of robot manipulators’, Journal of Soft Computing, 2005, Vol. 9, Pages: 185–193

Chin, S.H., Er, M.J., ‘Hybrid adaptive fuzzy controllers of robot manipulators’, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 13-17 Oct. 1998, Vol. 2, Pages: 1132 – 1137

Chun-Fei, H., Chin-Teng, L., ‘New techniques for intelligent control’, Proceedings of IEEE International Symposium on Intelligent Control, 2004, Pages: 13 – 18

Colbaugh, R., Glass, K., Seraji, H., ‘Decentralized adaptive control of manipulators theory and experiments’, Proceedings of the 32nd IEEE Conference on Decision and Control, 15-17 Dec. 1993, Vol.1, Pages: 153 – 158

Colbaugh, R., Seraji, H., ‘Adaptive tracking control of manipulators: Theory and Experiments’, Proceedings of IEEE International Conference on Robotics and Automation, 8-13 May 1994, Pages: 2992 – 2999

Colbaugh, R., Seraji, H., Glass, K., ‘Application of adaptive tracking control to industrial robots’, Proceedings of the Third IEEE Conference on Control Applications, 24-26 Aug. 1994, Vol.2, Pages: 915 – 920

Commuri, S., Lewis, F.L., ‘Adaptive-fuzzy logic control of robot manipulators’, Proceedings of IEEE International Conference on Robotics and Automation, 22-28 April 1996, Vol. 3, Pages: 2604 – 2609

Craig, J. J., ‘Adaptive control of mechanical manipulators’, Addison-Wesley, New York, 1988.

Craig, J. J., ‘Introduction to robotics-mechanics and control’, Addison-Wesley, New York, 1989.

Craig, J. J., Hsu, P., Sastry, S., ‘Adaptive control of mechanical manipulators’, Proceedings of IEEE International Conference on Robtics and Automation, 1986, Pages: 190-195

Datta, A., Ming-Tzu Ho, ‘A modified model reference adaptive control scheme for rigid robots’, IEEE Transactions on Robotics and Automation, June 1996, Vol. 12, Issue: 3, Pages: 466 – 470

de Silva C. W., ‘Applications of fuzzy logic in the control of robotic manipulators’, Fuzzy Sets and Systems, Vol. 70, Pages: 223 – 234

De Wit, C.C., Fixot, N., ‘Adaptive control of robot manipulators via velocity estimated feedback’, IEEE Transactions on Automatic Control, Aug. 1992, Vol. 37, Issue: 8, Pages: 1234 – 1237

Dubois, D., Prade. H., ‘What are fuzzy rules and how to use them’, Fuzzy Sets and Systems, 1996, Vol. 84, Pages: 169-185

Dubowski, S., DesForges, D. T., ‘The application of model reference adaptive control to robotic manipulators’, ASME Journal of Dynamic Systems, Measurement, and Control, Vol. 101, 1979, Pages: 193-200

Dunlop, J.A., Burnham, K.J., James, D.J.G., King, P.J., ‘A self-regulating scaling method for fuzzy control’, Proceedings of the Third IEEE Conference on Control Applications, 24-26 Aug. 1994, Vol.1, Pages: 683 - 687

Emami, M. R., Goldenberg, A. A., Turksen, I. B., ‘Systematic design and analysis of fuzzy-logic control and application to robotics, Part I. Modeling’, Robotics and Autonomous Systems, 2000, Vol. 33, Pages: 65-68

Emami, M. R., Goldenberg, A. A., Turksen, I. B., ‘Systematic design and analysis of fuzzy-logic control and application to robotics, Part II. Control’, Robotics and Autonomous Systems, 2000, Vol. 33, Pages: 89-108

Er, M.J., ‘Recent developments and futuristic trends in robot manipulator control’, Proceedings of Asia-Pacific Workshop on Advances in Motion Control, 15-16 July 1993, Pages: 106 - 111

Erbatur, K., Kaynak, O., Rudas, I., ‘A study of fuzzy schemes for control of robotic manipulators’, Proceedings of the 1995 IEEE 21st International Conference on Industrial Electronics, Control, and Instrumentation, 6-10 Nov. 1995, Vol.1, Pages: 63 – 68

Erlic, M., Lu, W.-S., ‘A comparative evaluation of adaptive, robust and classical feedback controllers used in unconstrained trajectory tracking for robot manipulators’, Proceedings of the 33rd Midwest Symposium on Circuits and Systems, 12-14 Aug. 1990, Vol.2, Pages: 661 – 664

Gang, F., ‘A new adaptive control algorithm for robot manipulators in task space’, IEEE Transactions on Robotics and Automation, June 1995, Vol. 11, Issue: 3, Pages: 457 – 462

Gavel, D., Hsia, T., ‘Decentralized adaptive control of robot manipulators’, Proceedings of IEEE International Conference on Robotics and Automation, Mar 1987, Vol: 4, Pages: 1230 – 1235

Ge, S. S., ‘Advanced control techniques of robotic manipulators’, Proceedings of the American Control Conference, June 1998, Pages: 2185-2199

Golea, N., ‘Indirect fuzzy adaptive model-following control for robot manipulators’, Proceedings of International Conference on Control Applications, 18-20 Sept. 2002, Vol. 1, Pages: 198 – 202

Ham, C., Qu, Z., Johnson, R., ‘Robust fuzzy control for robot manipulators’, Proceedings of IEE - Control Theory and Applications, March 2000, Vol. 147, Issue 2, Pages: 212-216

Ham, C., Qu, Z., Kaloust, J., Johnson, R., ‘A new learning control of robot manipulators in the presence of actuator dynamics’, Proceedings of IEEE International Conference on Robotics and Automation, 21-27 May 1995, Vol. 2, Pages: 2144 – 2149

Ham, W., ‘Adaptive control based on explicit model of robot manipulator’, IEEE Transactions on Automatic Control, April 1993, Vol: 38, Issue: 4, Pages: 654 – 658

Heredia, J.A., Wen, Yu, ‘A high-gain observer-based PD control for robot manipulator’, Proceedings of the American Control Conference, 28-30 June 2000, Vol. 4, Pages: 2518 - 2522

Hojati, M., Gazor, S., ‘Hybrid adaptive fuzzy identification and control of nonlinear systems’, IEEE Transactions on Fuzzy Systems, April 2002, Vol. 10, Issue 2, Pages: 198 – 210

Horne, B., Jamshidi, M., Vadiee, N., ‘ Neural networks in robotics: A survey’, Journal of Intelligent and Robotic systems, 1990, Vol. 3, Pages: 61-66

Horowitz, R., Tomizuka, M., ‘An adaptive control scheme for mechanical manipulators - compensation of nonlinearity and decoupling control’, ASME Journal of Dynamic Systems, Measurement, and Control, Vol.108, 1986, Pages: 127-135

Hsia, T., ‘Adaptive control of robot manipulators- A review’, Proceedings of IEEE International Conference on Robotics and Automation, Apr 1986, Vol. 3, Pages: 183 – 189

Hsu, F., Fu, L., ‘Nonlinear control of robot manipulators using adaptive fuzzy sliding mode control’, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 5-9 Aug. 1995, Vol. 1, Pages: 156 – 161

Hsu, S., Fu, L., ‘Globally adaptive decentralized control of robot manipulators’, Proceedings of the 41st SICE Annual Conference, 5-7 Aug. 2002, Vol. 1, Pages: 402 – 407

Hu, B., Mann, G.K.I., Gosine, R.G., ‘A Systematic Study of Fuzzy PID Controllers—Function-Based Evaluation Approach’, IEEE Transactions on Fuzzy Systems, Oct. 2001, Vol. 9, Issue: 5, Pages: 699 - 712

Imura, J.-I., Sugie, T., Yoshikawa, T., ‘Adaptive robust control of robot manipulators-Theory and Experimentation’, IEEE Transactions on Robotics and Automation, Oct. 1994, Vol. 10 , Issue: 5 , Pages:705 – 710

Jamshidi, M., ‘Fuzzy control of complex systems’, Journal of Soft Computing, 1997, Vol.1, Pages: 42-56

Jantzen J., ‘The Self-Organizing Fuzzy Controller’, Tech. report no 98-H 869 (soc), Technical University of Denmark, 19 Aug 1998.

Jen-Yang, C., ‘Hybrid model-based adaptive fuzzy control system’, Proceedings of IEEE International Conference on Fuzzy Systems, 2-17 May 2002, Vol. 2, Pages: 1126 – 1131

Jing, Y., ‘Adaptive control of robotic manipulators including motor dynamics’, IEEE Transactions on Robotics and Automation, Aug. 1995, Vol. 11, Issue: 4, Pages: 612 - 617

Johansson, R., ‘Adaptive control of robot manipulator motion’, IEEE Transactions on Robotics and Automation, Aug. 1990, Vol. 6, Issue: 4, Pages: 483 – 490

Kandel, A., Luo, Y., Zhang, Y.-Q., ‘Stability analysis of fuzzy control systems’, Fuzzy Sets and Systems, 1999, Vol. 105, Pages: 33-48

Karner, J., Janocha H., ‘Hybrid controller for adaptive link control of industrial robots’, Journal of Intelligent and Robotic Systems, September 1997, Vol. 20, Numbers 2-4, Pages: 93 – 104

Kawasaki, H., Bito, T., Kanzaki, K., ‘An efficient algorithm for the model-based adaptive control of robotic manipulators’, IEEE Transactions on Robotics and Automation, June 1996, Vol. 12, Issue: 3, Pages: 496 – 501

Kazemian, H. B., ‘Development of an intelligent fuzzy controller’, Proceedings of IEEE International Conference on Fuzzy Systems, 2001, Vol. 1, Pages: 517-520

Kazemian, H.B., ‘The self organizing fuzzy PID controller’, Proceedings of IEEE International Conference on Computational Intelligence, 4-9 May 1998, Vol. 1, Pages: 319 – 324

Kazemian, H.B., ‘The SOF-PID controller for the control of a MIMO robot arm’, IEEE Transactions on Fuzzy Systems, Aug. 2002, Vol. 10, Issue: 4, Pages: 523 – 532

Kelly, R., ‘Global positioning of robot manipulators via PD control plus a class of nonlinear integral actions’, IEEE Transactions on Automatic Control, July 1998, Vol. 43, Issue: 7, Pages: 934 – 938

Kelly, R., Ortega, R., ‘Adaptive control of robot manipulators an input-output approach’, Proceedings of IEEE International Conference on Robotics and Automation, 24-29 April 1988, Vol.2, Pages: 699 - 703

Ken. C., Jian-Ya, L., Xiang, L.Y., ‘Fuzzy control of robot manipulators’, Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, August 8-12, 1988, Vol: 2, Pages: 1210 – 1212

Khorrami, F., Ozguner, U., ‘Decentralized control of robot manipulators via state and proportional-integral feedback’, Proceedings of IEEE International Conference on Robotics and Automation, 24-29 April 1988, Vol.2, Pages: 1198 – 1203

Khosla, P. K., Kanade, Takeo, ‘Parameter identification of robot dynamics’, Proceedings of IEEE Conference on Decision and Control, 1985, Pages: 777-787

Khosla, P., ‘Choosing Sampling Rates for robot control’, Proceedings of IEEE International Conference on Robotics and Automation, Mar 1987, Vol. 4, Pages: 169 – 174

Khoury, G. M., Saad, M., Kanaan H., Asmar, Y. C., ‘Fuzzy PID control of a five DOF robot arm’, Journal of Intelligent and Robotic Systems, July 2004, Vol. 40, Number 3, Pages: 299 – 320

Kim, K., Hori, Y., ‘Experimental evaluation of adaptive and robust schemes for robot manipulator control’, IEEE Transactions on Industrial Electronics, Dec. 1995, Vol. 42, Issue: 6, Pages: 653 – 662

Kim, V.T., ‘Independent joint adaptive fuzzy control of robot manipulators’, Proceedings of the 5th Biannual World Automation Congress, 9-13 June 2002, Vol. 14, Pages: 645 – 652

Kim,E., ‘Output feedback tracking control of robot manipulators with model uncertainty via adaptive fuzzy logic’, IEEE Transactions on Fuzzy Systems, June 2004, Vol. 12, No. 3, Pages: 368 – 378

Koditschek, Daniel E., ‘Adaptive strategies for the control of natural motion’, Proceedings of IEEE 24th Conference on Decision and Control, Dec 1985, Pages: 1405-1409

Koditschek, Daniel E., ‘High gain feedback and telerobotic tracking’, Workshop on Space Telembotics, Pasadena, CA, Jan 1987, Pages: 355-363

Koh, K.C., Cho, H.S., Kim, S.K., Jeong, I.S., ‘Application of a self-organizing fuzzy control to the joint control of a Puma-760 robot’, Proceedings of IEEE International Workshop on Intelligent Robots and Systems, 3-6 July 1990, Vol.1, Pages: 537 – 542

Koo, T.J., ‘Model reference adaptive fuzzy control of robot manipulator’, Proceedings of IEEE International Conference on Systems, Man and Cybernetics, 22-25 Oct. 1995, Vol. 1, Pages: 424 – 429

Kuswadi, S., Sampei, M., Nakaura, S., ‘Model reference adaptive fuzzy control for one linear actuator hopping robot’, Proceedings of IEEE International Conference on Fuzzy Systems, 25-28 May 2003, Vol. 1, Pages: 254 – 259

Kwan, E., Liu, M., ‘An adaptive fuzzy approach for robot manipulator tracking’, Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, 8-9 Nov. 1999, Pages: 53 – 58

Landau, I.D., ‘Future trends in adaptive control of robot manipulators’, Proceedings of the 27th IEEE Conference on Decision and Control, 7-9 Dec. 1988, Vol.2, Pages: 1604 – 1606

Leung, T.P., Su, C.-Y., Zhou, Q.-J., ‘Sliding mode control of robot manipulators: A case study’, Proceedings of 16th Annual Conference of IEEE, IECON '90, 27-30 Nov. 1990, Vol.1, Pages:671 – 675

Lewis, F.L., Maliotis, G., Abdallah, C., ‘Robust adaptive control for a class of partially known nonlinear systems’, Proceedings of the 27th IEEE Conference on Decision and Control, 7-9 Dec. 1988, Vol.3, Pages: 2425 – 2427

Li, W., ‘Design of a Hybrid fuzzy logic proportional plus conventional integral-derivative controller’, IEEE Transactions on Fuzzy Systems, Nov 1998, Vol. 6, Issue: 4, Pages: 449-463

Li, W., Changa, X. G.,Wahl, F. M., Farrell, J., ‘Design of an enhanced hybrid fuzzy P+ID controller for a mechanical manipulator’, IEEE Transactions on Systems, Man and Cybernetics, Part B, Dec 2001, Vol. 31, Issue: 6, Pages: 938-945

Li, W., Changa, X. G.,Wahl, F. M., Farrell, J., 'Tracking control of a manipulator under uncertainty by FUZZY P+ID controller', Fuzzy Sets and Systems, 2001, Vol. 122, Pages: 125-137

Liegeois, A., Fournier. A., Aldon, M., 'Model reference control of high velocity industrial robots', Proceedings of Joint Automatic Control Conference, 1980.

Lim, C.M., Hiyama, T., 'Application of fuzzy logic control to a manipulator', IEEE Transactions on Robotics and Automation, Oct. 1991, Vol. 7, Issue: 5, Pages: 688 – 691

Lin, C., Mon, Y., 'Hybrid adaptive fuzzy controllers with application to robotic systems', 2003, Fuzzy Sets and Systems, Vol. 139, Pages: 151-165

Lin, P.I., Hwang, S., Chou, J., 'Comparison on fuzzy logic and PID controls for a DC motor position controller', Conference Record of the 1994 IEEE Industry Applications Society Annual Meeting, 2-6 Oct. 1994, Vol.3, Pages: 1930 – 1935

Lin, W., Cheng, C., Chen, C., 'Adaptive fuzzy design for optimal tracking control of robot manipulators', Proceedings of IEEE International Symposium on Intelligent Control, 2003, Pages: 908 – 913

Llama, M.A., Kelly, R., Santibanez, V., 'Stable Computed-Torque control of robot manipulators via fuzzy self tuning', IEEE Transactions on Systems, Man and Cybernetics, Part B, Feb. 2000, Vol. 30, Issue: 1, Pages: 143 – 150

Llama, M.A., Santibanez, V., Kelly, R., Flores, J., 'Stable fuzzy self-tuning computed-torque control of robot manipulators', Proceedings of IEEE International Conference on Robotics and Automation, 16-20 May 1998, Vol. 3, Pages: 2369 – 2374

Loc, H. D., Ha, T. T., Cuong, N. C., ‘An adaptive fuzzy logic controller for robot-manipulator’, International Journal of Advanced Robotic Systems, 2004, Vol. 1 Number 2, Pages: 115-117

Loria, A., Lefeber, A.A.J., Nijmeijer, H., ‘Global asymptotic stability of robot manipulators with linear PID and PI^{2}D control’, Stability and Control.:Theory and Appllications., 2000, Vol. 3(2), Pages: 138-149

Luh, J. Y. S., Walker, M. W., Paul, R. P., ‘Resolved acceleration control of mechanical manipulators’, IEEE Transaction on Automatic Control, Vol.25, 1980, Pages: 468-474

Luh, J.Y.S, ‘Conventional Controller Design for Industrial Robots: A Tutorial’, IEEE Transactions on Sys., Man, and Cybernetics, May/June 1983, Vol. 13, No. 3, Pages: 298-316,

Magana, M.E., Tagami, S., ‘An improved trajectory tracking decentralized adaptive controller for robot manipulators’, IEEE Transactions on Industrial Electronics, Oct. 1994, Vol. 41, Issue: 5, Pages: 477 – 482

Maliotis, G.N., Lewis, F.L., ‘Improved robust adaptive controller for a class of partially known nonlinear systems’, Proceedings of the 28th IEEE Conference on Decision and Control, 13-15 Dec. 1989, Vol.3, Pages: 2155 – 2157

Mamdani, E.H., ‘Twenty years of fuzzy control: Experiences gained and lessons learnt’, Proceedings of IEEE International Conference on Fuzzy Systems, 28 March-1 April 1993, Vol.1, Pages: 339 – 344

Mao-Lin N., Meng, J. E., ‘Decentralized control of robot manipulators with couplings and uncertainties’, Proceedings of American Control Conference, 28-30 June 2000, Vol. 5, Pages: 3326 – 3330

Meng, J.E., Swee Hong, C., ‘Hybrid adaptive fuzzy controllers of robot manipulators with bound estimation’, IEEE Transactions on Industrial Electronics, Oct. 2000, Vol. 47, Issue: 5, Pages: 1151 - 1160

Messner, W., Horowitz, R., Kao, W. -W., Boals, M., ‘A new adaptive learning rule’, IEEE Transactions on Automatic Control, Vol. 36, Issue: 2, Feb. 1991, Pages: 188 – 197

Miljanovic, D.M., Croft, E.A., ‘A taxonomy for robot control’, Proceedings of IEEE International Conference on Robotics and Automation, 10-15 May 1999, Vol. 1, Pages: 176 – 181

Mudi, R.K., Pal, N.R., ‘A robust self-tuning scheme for PI- and PD-type fuzzy controllers’, IEEE Transactions on Fuzzy Systems, Feb. 1999, Vol. 7, Issue: 1, Pages: 2 – 16

Nagrath, I.J., Pahade Paras Shripal, Chand, A., ‘ Development and implementation of intelligent control strategy for robotic manipulator’, Proceedings of IEEE/IAS International Conference on Industrial Automation and Control, 5-7 Jan. 1995, Pages: 215 – 220

Nedungadi, A., Wenzel, D.J., ‘A novel approach to robot control using fuzzy logic’, Proceeding of IEEE International Conference on Systems, Man, and Cybernetics, 13-16 Oct. 1991, Vol.3, Pages: 1925 - 1930

Neo, S.S., Er, M.J., ‘Adaptive fuzzy control of robot manipulators’, Proceedings of the 4th IEEE Conference on Control Applications, 28-29 Sept. 1995, Pages: 724 – 729

Niemeyer,G., Slotine,J.-J.E., ‘Performance in adaptive manipulator control’, Proceedings of the 27th IEEE Conference on Decision and Control, 7-9 Dec. 1988, Vol.2, Pages:1585 - 1591

Oh, B.J., Jamshidi, M., Seraji, H., ‘Decentralized adaptive control [robot]’, Proceedings of IEEE International Conference on Robotics and Automation, 24-29 April 1988, Vol.2, Pages: 1016 – 1021

Ortega, R., Spong, M.W., ‘Adaptive motion control of rigid robots: A tutorial’, Proceedings of the 27th IEEE Conference on Decision and Control, 7-9 Dec. 1988, Vol.2, Pages: 1575 - 1584

Pagilla, P.R., Biao Yu, ‘Adaptive control of a robot carrying a time-varying payload’, Proceedings of the IEEE International Conference on Control Applications, 25-27 Sept. 2000, Pages: 68 – 73

Parra-Vega, V., Arimoto, S., Yun-Hui Liu, Hirzinger, G., Akella, P., ‘Dynamic sliding PID control for tracking of robot manipulators: Theory and experiments’, IEEE Transactions on Robotics and Automation, Dec. 2003, Vol. 19, Issue: 6, Pages: 967 – 976

Patino, H.D., Carelli, R., Kuchen, B.R., ‘Neural networks for advanced control of robot manipulators’, IEEE Transactions on Neural Networks, Mar 2002, Vol. 13, Issue: 2, Pages: 343-354

Paul, R.P., ‘Modeling, Trajectory Calculation, and Servoing of a Computer Controlled Arm’, Stanford A.I. Lab, A.I. Memo 177, Stanford, CA, Nov. 1972.

Purwar, S., Kar, I.N., Jha, A.N., ‘Adaptive control of robot manipulators using fuzzy logic systems under actuator constraints’, Proceedings of IEEE International Conference on Fuzzy Systems, 25-29 July 2004, Vol. 3, Pages: 1449 – 1454

Sadegh, N., Horowitz, R., ‘An exponentially stable adaptive control law for robot manipulators’, IEEE Transactions on Robotics and Automation, Aug. 1990, Vol. 6, Issue: 4, Pages: 491 – 496

Sadegh, N., Horowitz, R., ‘Stability analysis of an adaptive controller for robotic manipulators’, Proceedings of IEEE International Conference on Robotics and Automation, Mar 1987, Vol.: 4, Pages: 1223 – 1229

Santibanez, V., Kelly, R., Llama, M.A., ‘Fuzzy PD+ control for robot manipulators’, Proceedings of IEEE International Conference on Robotics and Automation, 24-28 April 2000, Vol. 3, Pages: 2112 – 2117

Seraji, H., ‘Adaptive independent joint control of manipulators: theory and experiment’, Proceedings of IEEE International Conference on Robotics and Automation, 24-29 April 1988,Vol.2, Pages: 854 - 861

Sinha, A.S.C., Kayalar, S., Yurtseven, H.O., ‘Nonlinear adaptive control of robot manipulators’, Proceedings of IEEE International Conference on Robotics and Automation, 13-18 May 1990, Vol.3, Pages: 2084 – 2088

Slotine, J. -J., ‘Robustness issues in robot control’, Proceedings of IEEE International Conference on Robotics and Automation, Mar 1985, Vol. 2, Pages: 656 – 661

Slotine, J.-J., ‘On modeling and adaptation in robot control’, Proceedings of IEEE International Conference on Robotics and Automation, Apr 1986, Vol. 3 , Pages:1387 – 1392

Slotine, J.-J.E., Li Weiping, ‘Adaptive manipulator control: A case study’, IEEE Transactions on Automatic Control, Nov. 1988,Vol. 33 , Issue: 11 , Pages:995 – 1003

Song, Y. D., Gao, W.B, Cheng, M., ‘Study on path tracking control of robot manipulators with unknown payload’, Proceedings of IEEE International Conference on Systems Engineering, 24-26 Aug. 1989, Pages: 321 – 324

Song, Y.D., ‘Adaptive motion tracking control of robot manipulators: Non-regressor based approach’, Proceedings of IEEE International Conference on Robotics and Automation, 8-13 May 1994, Vol.4, Pages: 3008 – 3013

Spong, M., Thorp, J., Kleinwaks, J., ‘The control of robot manipulators with bounded input’, IEEE Transactions on Automatic Control, Jun 1986, Vol. 31, Issue: 6, Pages: 483 – 490

Spong, M.W., and Vidyasagar, M., ‘Robot Dynamics and Control’, John Wiley & Sons, Inc., New York, 1989.

Spong, M.W., Lewis, F., and Abdallah, C., ‘Robot Control: Dynamics, Motion Planning, and Analysis’, IEEE Press, 1992.

Spong, M.W., Ortega, R., ‘On adaptive inverse dynamics control of rigid robots’, IEEE Transactions on Automatic Control, Jan. 1990, Vol. 35, Issue: 1, Pages: 92 – 95

Su, C., Stepanenko, Y., ‘Guaranteed stability based control of robot manipulators incorporating motor dynamics’, Proceedings of IEEE International Symposium on Industrial Electronics, 25-27 May 1994, Pages: 345 – 350

Takegaki, M., Arimoto, S., ‘An adaptive trajectory control of manipulators’, The International Journal of Control, Vol.102, 1981, Pages: 201-217

Tarokh, M., ‘Decentralized digital adaptive control of robot motion’, Proceedings of IEEE International Conference on Robotics and Automation, 13-18 May 1990, Vol.2, Pages: 1410 – 1415

Tarokh, M., Seraji, H., ‘A control scheme for trajectory tracking of robot manipulators’, Proceedings of IEEE International Conference on Robotics and Automation, 24-29 April 1988, Vol.2, Pages: 1192 – 1197

Tosunoglu, S., Tesar, D., ‘State of the Art in adaptive control of robotic systems’, IEEE Transactions on Aerospace and Electronic Systems, Sept. 1988, Vol. 24, Issue: 5, Pages: 552 – 561

Trusca, M., Lazea, G., ‘An adaptive PID learning controller for periodic robot motion’, Proceedings of IEEE Conference on Control Applications, 23-25 June 2003, Vol. 1, Pages: 686 – 689

Tsai, C., Wang, C., Lin, W., ‘Robust fuzzy model-following control of robot manipulators’, IEEE Transactions on Fuzzy Systems, Aug. 2000, Vol. 8, Issue: 4, Pages: 462 – 469

Tso, S.K., Xu, Y., Shum, H.Y., ‘Variable structure model reference adaptive control of robot manipulators’, Proceedings of IEEE International Conference on Robotics and Automation, 9-11 April 1991, Vol.3, Pages: 2148 – 2153

Tzafestas, S., Papanikolopoulos, N.P., ‘Incremental fuzzy expert PID control’, IEEE Transactions on Industrial Electronics, Oct. 1990, Vol. 37, Issue: 5, Pages: 365 – 371

Visioli, A., ‘Tuning of PID controllers with fuzzy logic’, IEE Proc.-Control Theory Appl., Jan. 2001, Vol. 148, No. I, Pages: 1-8

Wei Sun, Yaonan Wang, ‘An adaptive fuzzy control for robotic manipulators’, Proceedings of Control, Automation, Robotics and Vision Conference, 6-9 Dec. 2004, Vol. 3, Pages: 1952 – 1956

Wen, J. T., and Bayard, D. S., ‘Robust control for robotic manipulators, Part I**:** Non-adaptive case’, Jet Propulsion Lab., Pasadena, CA, Tech. Rep. 347-87-203, 1987.

Wen, J.T., ‘A unified perspective on robot control: the energy Lyapunov function approach’, Proceedings of the 29th IEEE Conference on Decision and Control, 5-7 Dec. 1990, Vol.3, Pages: 1968 – 1973

Wen, J. T., Bayard, David S., ‘Robust control for robotic manipulators part 1: Nonadaptive case’, Technical Report 347-87-203, 1987, Jet Propulsion Laboratory, Pasadena, CA

Whitcomb, L.L., Rizzi, A.A., Koditschek, D.E., ‘Comparative experiments with a new adaptive controller for robot arms’, IEEE Transactions on Robotics and Automation, Feb. 1993,Vol: 9, Issue: 1, Pages: 59 - 70

Ya Lei Sun, Meng Joo Er, ‘Hybrid fuzzy control of robotics systems’, IEEE Transactions on Fuzzy Systems, Dec. 2004, Vol. 12, Issue 6, Pages: 755 – 765

Ya-Chen Hsu, Guanrong Chen, Sanchez, E., ‘A fuzzy PD controller for multi-link robot control stability analysis’, Proceedings of IEEE International Conference on Robotics and Automation, 20-25 April 1997, Vol. 2, Pages: 1412 – 1417

Yao Bin, ‘Adaptive robust control of robot manipulators: Theory and comparative experiments’, The second CWC on ICIA, 1997, Pages: 442-447.

Yoo, B.K., Ham, W.C., ‘Adaptive Control of Robot Manipulators using fuzzy compensator-part 1’, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 17-21 Oct. 1999, Vol. 1, Pages: 35 – 40

Yoo, B.K., Ham, W.C., ‘Adaptive Control of Robot Manipulators using fuzzy compensator-part 2’, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 17-21 Oct. 1999, Vol. 1, Pages: 52 – 56

Yu Tang, Arteaga, M.A., ‘Adaptive control of robot manipulators based on passivity’, IEEE Transactions on Automatic Control, Sept. 1994,Vol. 39, Issue: 9, Pages: 1871 – 1875

Yu, H., ‘Robust Combined adaptive and variable structure adaptive control of robot manipulators’, Robotica, 1998, Vol. 16, Pages: 623–650.

Yu, H., Lloyd, S., ‘Adaptive control of robot manipulator including motor dynamics’, Proceedings of the American Control Conference, 21-23 June 1995, Vol. 5, Pages: 3803 – 3807

Yu, H., Lloyd, S., ‘Variable structure adaptive control of robot manipulators’, IEE Proceedings on Control Theory and Applications, March 1997, Vol. 144, Issue: 2, Pages: 167 – 176

Yu, H., Seneviratne, L.D., Earles, S. W.E., ’Adaptive control of robot manipulators’, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, July 7-10,1992, Pages: 293 – 298

Yuh, J., Nie, J., Lee, W.C., ‘Adaptive control of robot manipulators using bound estimation’, Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 13-17 Oct. 1998, Vol.2, Pages: 1126 – 1131

Zergeroglu, E., Dawson, D.M., de Queiroz, M.S., Krstic, M., ‘On Global output feedback tracking control of robot manipulators’, Proceedings of the 39th IEEE Conference on Decision and Control, 12-15 Dec. 2000, Vol. 5, Pages: 5073 – 5078

Zhang, F., Dawson, D.M., de Queiroz, M.S., Dixon, W.E., ‘Global adaptive feedback tracking control of robot manipulators’, IEEE Transactions on Automatic Control, June 2000, Vol. 45, Issue: 6, Pages: 1203 – 1208

Zhang, H., Trott, G., Paul, R.P., ‘Minimum delay PID control of interpolated joint trajectories of robot manipulators’, IEEE Transactions on Industrial Electronics, Oct. 1990, Vol. 37, Issue: 5, Pages: 358 - 364

Zhao, Z., Tomizuka, M., Isaka, S., ‘Fuzzy gain scheduling of PID controllers’, IEEE Transactions on Systems, Man and Cybernetics, Oct 1993, Vol. 23, Issue: 5, Pages: 1392-1398

APPENDIX A

In this appendix are presented some sample codes used for simulation of different control algorithms. All the codes except for Adaptive fuzzy controller are written in C. That for adaptive fuzzy is written in MATLAB. We present sample codes in each of the three categories, i.e., conventional, adaptive and fuzzy.

A.1 CDID CONTROL CODE

```
/* This control algorithm controls a two-link manipulator. Path update rate is 333 Hz. For
each set point the control loop is executed 5 times. Strategy is CDID Control.  */
/*-----------------------------------------------------------------------------------------------------*/
                                        /*SIMULATION*/
                        /* CRITICALLY DAMPED INVERSE DYNAMICS */
/*-----------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <float.h>
#define pi 3.142857
/*---------------------------------------------------------------------------------------------------*/
                                  /* TRAJECTORY IS QUINTIC */
/*---------------------------------------------------------------------------------------------------*/
void main()
{
int    n=0,i=0,k=5,seg=1;
float tf,t=0.0,ts=0.003,tsp,g=9.8,acc1d,acc2d,v1cap=2.5,v2cap=2.5,a0,b0,
thf1,thf2,vel1=0.0,vel2=0.0,l1=0.26,a3,a4,a5,b3,b4,b5,v22,pos1o,pos2o,
 pos1=0.0,pos2=0.0,p1,p2,p3,p4,p5,p6,p7,izz2=0.09,c1,c2,con11,con12,
 error1s[5000],error2s[5000],acc1=0.0,acc2=0.0,pos1d,pos2d,vel1d,vel2d,
ep1,ep2,ev1,ev2,ep1s=0.0,ep2s=0.0,torque1,torque2,izz1=0.09,izz1cap=0.09,con21,
con22,m1=2.0,m2=2.0,v1=2.5,v2=2.5,x1=0.13,x2=0.14,izz2cap=0.09,e1dot,e2dot,
m11,m12,m21,m22,v11,v12,v21,g11,g21,m2cap=1.8,m1cap=1.8,vel1r,vel2r,
x2cap=0.14,x1cap=0.13,gamma=100.0,kv11=50.0,kv22=50.0,ki11=50.0,ki22=50.0,acc1r
,acc2r,izz1n=1.5,izz2n=0.09,m1n=3.0,m2n=3.0,x1n=0.15,x2n=0.16;
FILE   *er1,*er2;
clrscr();
/*-------------------------------------------------------------------------------------------------------*/
                                        /*GET INPUTS*/
/*-------------------------------------------------------------------------------------------------------*/
tsp=ts/5.0;
highvideo();
textbackground(YELLOW);
window(10,5,80,25);
```

```
textcolor(CYAN);
cprintf("TRAJECTORY IS QUINTIC POLYNOMIAL || Tf(max)= 5sec");
window(15,10,80,25);
textcolor(CYAN);
cprintf("FINAL TIME                    ");
scanf("%f",&tf);
window(15,14,80,25);
cprintf("ANGLE OF THE FIRST JOINT IN DEGREES(+) ");
scanf("%f",&thf1);
window(15,15,80,25);
cprintf("ANGLE OF THE SECOND JOINT IN DEGREES(-)");
scanf("%f",&thf2);
/*----------------------------------------------------------------------------------------------------*/
                          /* INITIALISE */
/*----------------------------------------------------------------------------------------------------*/
thf1=(thf1*pi)/180.0;
thf2=(thf2*pi)/180.0;
if(thf1 < 0.0 || thf2 > 0.0)
{
window(15,18,80,25);
cprintf("GIVE CORRECT ANGLES              ");
goto END;
}
/*----------------------------------------------------------------------------------------------------*/
        /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY*/
/*----------------------------------------------------------------------------------------------------*/
a0=0.0;
b0=0.0;
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
/*----------------------------------------------------------------------------------------------------*/
                      /*ACTUAL PARAMETERS*/
/*----------------------------------------------------------------------------------------------------*/
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
window(15,19,80,25);
cprintf("BUSY WITH CONTROL              \n");
```

```
/*----------------------------------------------------------------------------------------------------*/
                              /*SERVO LOOP*/
/*----------------------------------------------------------------------------------------------------*/
for (;seg<=4;)
{
for (;t<=tf;)
{
            /* DESIRED POSITION, VELOCITY, ACCELERATION*/
if(k==5)
{
pos1d=a0+(a3*t*t*t)+(a4*t*t*t*t)+(a5*t*t*t*t*t);
pos2d=b0+(b3*t*t*t)+(b4*t*t*t*t)+(b5*t*t*t*t*t);
vel1d=(3.0*a3*t*t)+(4.0*a4*t*t*t)+(5.0*a5*t*t*t*t);
vel2d=(3.0*b3*t*t)+(4.0*b4*t*t*t)+(5.0*b5*t*t*t*t);
acc1d=(6.0*a3*t)+(12.0*a4*t*t)+(20.0*a5*t*t*t);
acc2d=(6.0*b3*t)+(12.0*b4*t*t)+(20.0*b5*t*t*t);
t=t+ts;
k=0;
}
ep1=gamma*(pos1d-pos1);
ev1=gamma*(vel1d-vel1);
/*----------------------------------------------------------------------------------------------------*/
                                /*LINK 2 */
/*----------------------------------------------------------------------------------------------------*/
ep2=gamma*(pos2d-pos2);
ev2=gamma*(vel2d-vel2);
ep1s=ep1s+(pos1d-pos1);
ep2s=ep2s+(pos2d-pos2);
vel1r=vel1d+ep1;
vel2r=vel2d+ep2;
acc1r=acc1d+ev1;
acc2r=acc2d+ev2;
e1dot=vel1r-vel1;
e2dot=vel2r-vel2;
/*exact model*/
m11=izz1+izz2+(m2*(x2*x2+l1*l1))+(m1*x1*x1)+2.0*m2*l1*x2*cos(pos2);
m12=(m2*x2*x2)+izz2+(m2*l1*x2*cos(pos2));
m21=m2*x2*l1*cos(pos2)+izz2+(m2*x2*x2);
m22=(m2*x2*x2)+izz2;
v11=v1-(2.0*m2*l1*x2*sin(pos2))*vel2;
v12=(-m2*l1*x2*sin(pos2))*vel2;
v21=m2*l1*x2*sin(pos2)*vel1;
v22=v2;
g11=g*(m1*x1+m2*l1)*cos(pos1)+(m2*x2*g*cos(pos1+pos2));
g21=g*(m2*x2*cos(pos1+pos2));
/*-----------------------------------------------------------------------------------------------------*/
                    /*Calculate torques due to known values*/
/*-----------------------------------------------------------------------------------------------------*/
torque1=m11*acc1r+m12*acc2r+v11*vel1r+v12*vel2r+g11;
torque2=m21*acc1r+m22*acc2r+v21*vel1r+v22*vel2r+g21;
/*-----------------------------------------------------------------------------------------------------*/
                         /*Total torque to be applied*/
/*-----------------------------------------------------------------------------------------------------*/
torque1=torque1+kv11*e1dot+ki11*ep1s;
torque2=torque2+kv22*e2dot+ki22*ep2s;
/*-----------------------------------------------------------------------------------------------------*/
                         /* VALUES FOR LINK 1 */
/*-----------------------------------------------------------------------------------------------------*/
c1=      (p4-p2*sin(pos2)*vel2)*vel1-0.5*p2*sin(pos2)*vel2*vel2+
          p6*cos(pos1)+(p7*cos(pos1+pos2));
con11= (p1+p2*cos(pos2)+0.1498);
con12= (p3+0.5*p2*cos(pos2));
/*-----------------------------------------------------------------------------------------------------*/
                          /*VALUES FOR LINK 2*/
/*-----------------------------------------------------------------------------------------------------*/
c2=      0.5*p2*sin(pos2)*vel1*vel1+p5*vel2+p7*cos(pos1+pos2);
con21= (0.5*p2*cos(pos2)+p3);
con22= (p3+0.1498);
/*-----------------------------------------------------------------------------------------------------*/
                            /*STORE VALUES*/
/*-----------------------------------------------------------------------------------------------------*/
if(k==4)
{
error1s[n]=ep1*180.0/(pi);
error2s[n]=ep2*180.0/(pi);
n=n+1;
ep1s=0.0;
ep2s=0.0;
}
/*-----------------------------------------------------------------------------------------------------*/
           /* CALCULATE ACTUAL POSITION / VELOCITY OF LINKS*/
/*-----------------------------------------------------------------------------------------------------*/
acc2=((torque1-c1)*con21-(torque2-c2)*con11)/(con12*con21-con11*con22);
acc1=((torque1-c1)*con22-(torque2-c2)*con12)/(con11*con22-con12*con21);
vel1=acc1*tsp+vel1;
vel2=acc2*tsp+vel2;
pos1=pos1+vel1*tsp+0.5*acc1*tsp*tsp;
pos2=pos2+vel2*tsp+0.5*acc2*tsp*tsp;
k=k+1;
}
seg=seg+1;
```

```
/*-----------------------------------------------------------------------------------------------------*/
    /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY FOR THE SECOND
                                  SEGMENT*/
/*-----------------------------------------------------------------------------------------------------*/
if (seg==2 || seg==4)
{
p1=izz1n+izz2n+m2n*(x2n*x2n+l1*l1)+m1n*x1n*x1n;
p2=2.0*m2n*l1*x2n;
p3=m2n*x2n*x2n+izz2n;
p4=v1;
p5=v2;
p6=g*(m1n*x1n+m2n*l1);
p7=m2n*x2n*g;
t=0.0;
a0=thf1;
b0=thf2;
a3=-10.0*(thf1)/(tf*tf*tf);
a4=15.0*(thf1)/(tf*tf*tf*tf);
a5=-6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=-10.0*(thf2)/(tf*tf*tf);
b4=15.0*(thf2)/(tf*tf*tf*tf);
b5=-6.0*(thf2)/(tf*tf*tf*tf*tf);
k=5;
}
if (seg==3)
{
t=0.0;
a0=0.0;
b0=0.0;
/*thf1=0.0;
thf2=0.0;*/
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
k=5;
}
}
/*------------------------------------------------------------------------------------------------------*/
                                        /*OPEN FILES*/
/*------------------------------------------------------------------------------------------------------*/
er1=fopen("scerror1.dat","w+");
er2=fopen("scerror2.dat","w+");
/*------------------------------------------------------------------------------------------------------*/
                                        /*WRITE FILES*/
/*------------------------------------------------------------------------------------------------------*/
for(i=0;i<=(n-1);i++)
{
fprintf(er1,"%f\n",error1s[i]/gamma);
fprintf(er2,"%f\n",error2s[i]/gamma);
}
fcloseall();
END:;
/*------------------------------------------------------------------------------------------------------*/
}
```

A.2. ACDID CONTROL CODE

```
/* This control algorithm controls a two-link manipulator. Path update rate is 333 Hz. For
each setpoint the control loop is executed 5 times.kd1=100 and kd2=50. */
/*------------------------------------------------------------------------------------------------------*/
                              /*SIMULATION ACDID CONTROL*/
                           /* POSITION + VELOCITY FEEDBACK */
/*------------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <float.h>
#define pi 3.142857
/*------------------------------------------------------------------------------------------------------*/
                                   /* TRAJECTORY IS QUINTIC */
/*------------------------------------------------------------------------------------------------------*/
void main()
{
int    n=0,i=0,k=5,seg=1;
float  tf,t=0.0,ts=0.003,tsp,kd1=50.0,kd2=50.0,lambda1=100.0,lambda2=100.0,g=9.8,
       vel1=0.0,vel2=0.0,l1=0.26,thf1,thf2,a0,b0,a3,a4,a5,b3,b4,b5,
       pos1,pos2,p1cap,p2cap,p3cap,p4cap,p5cap,p6cap,p7cap,
       p1,p2,p3,p4,p5,p6,p7,izz2=0.09,c1,c2,con11,con12,con21,con22,
       error1s[5000],error2s[5000],w11,w12,w13,w14,w16,w17,pos1o,pos2o,
      gama1=100.0,gama2=100.0,gama3=100.0,gama4=100.0,gama5=100.0,gama6=100.0,
       pos1d,pos2d,vel1d,vel2d,acc1d,acc2d,vel1r,vel2r,acc1r,acc2r,gama7=100.0,
       error1,error2,e1,e2,torque1,torque2,w22,w23,w25,w27,p1capdot=0.0,p2capdot=0.0,
       p3capdot=0.0,p4capdot=0.0,p5capdot=0.0,p6capdot=0.0,p7capdot=0.0,izz1=0.09,
       m1=2.0,m2=2.0,v1=2.5,v2=2.5,x1=0.13,x2=0.14,acc1=0.0,acc2=0.0,m2n=3.0,
       m1n=3.0, x1n=0.15,x2n=0.16,izz1n=1.5,izz2n=0.09,
       x1cap=0.11,x2cap=0.12,v1cap=2.0,v2cap=2.0,izz1cap=0.05,izz2cap=0.05,
       m1cap=1.0,m2cap=1.0;
FILE   *er1,*er2;
clrscr();
/*------------------------------------------------------------------------------------------------------*/
                                        /*GET INPUTS*/
/*------------------------------------------------------------------------------------------------------*/
tsp=ts/5.0;
highvideo();
textbackground(YELLOW);
window(10,5,80,25);
textcolor(CYAN);
cprintf("TRAJECTORY IS QUINTIC POLYNOMIAL || Tf(max)= 5sec");
window(15,10,80,25);
textcolor(CYAN);
```

```
cprintf("FINAL TIME                    ");
scanf("%f",&tf);
/*window(15,11,80,25);
textcolor(GREEN);
cprintf("FIRST ELEMENT OF GAIN MATRIX          ");
scanf("%f", &kd1);
window(15,12,80,25);
cprintf("SECOND ELEMENT OF GAIN MATRIX          ");
scanf("%f", &kd2);*/
window(15,14,80,25);
cprintf("ANGLE OF THE FIRST JOINT IN DEGREES(+) ");
scanf("%f",&thf1);
window(15,15,80,25);
cprintf("ANGLE OF THE SECOND JOINT IN DEGREES(-)");
scanf("%f",&thf2);

/*----------------------------------------------------------------------------------------------------*/
                              /* INITIALISE */
/*----------------------------------------------------------------------------------------------------*/
thf1=(thf1*pi)/180.0;
thf2=(thf2*pi)/180.0;
if(thf1 < 0.0 || thf2 > 0.0)
{
window(15,18,80,25);
cprintf("GIVE CORRECT ANGLES                    ");
goto END;
}
/*----------------------------------------------------------------------------------------------------*/
           /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY*/
/*----------------------------------------------------------------------------------------------------*/
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
pos1=0.0;
pos1o=0.0;
pos2=0.0;
pos2o=0.0;
a0=pos1o;
b0=pos2o;
/*----------------------------------------------------------------------------------------------------*/
                        /*ESTIMATE OF PARAMETERS */
/*----------------------------------------------------------------------------------------------------*/
p1cap=izz1cap+izz2cap+m2cap*(x2cap*x2cap+l1*l1)+m1cap*x1cap*x1cap;
```

```
p2cap=2.0*m2cap*l1*x2cap;
p3cap=m2cap*x2cap*x2cap+izz2cap;
p4cap=v1cap;
p5cap=v2cap;
p6cap=g*(m1cap*x1cap+m2cap*l1);
p7cap=m2cap*x2cap*g;
/*----------------------------------------------------------------------------------------------------*/
                                /*ACTUAL PARAMETERS*/
/*----------------------------------------------------------------------------------------------------*/
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
window(15,19,80,25);
cprintf("BUSY WITH CONTROL                    \n");
/*----------------------------------------------------------------------------------------------------*/
                                      /*SERVO LOOP*/
/*----------------------------------------------------------------------------------------------------*/
for (;seg<=4;)  /* for two segment trajectory*/
{
for (;t<=tf;)
{
                /* DESIRED POSITION, VELOCITY, ACCELERATION*/
if(k==5)
{
pos1d=a0+(a3*t*t*t)+(a4*t*t*t*t)+(a5*t*t*t*t*t);
pos2d=b0+(b3*t*t*t)+(b4*t*t*t*t)+(b5*t*t*t*t*t);
vel1d=(3.0*a3*t*t)+(4.0*a4*t*t*t)+(5.0*a5*t*t*t*t);
vel2d=(3.0*b3*t*t)+(4.0*b4*t*t*t)+(5.0*b5*t*t*t*t);
acc1d=(6.0*a3*t)+(12.0*a4*t*t)+(20.0*a5*t*t*t);
acc2d=(6.0*b3*t)+(12.0*b4*t*t)+(20.0*b5*t*t*t);
t=t+ts;
k=0;
}
error1=pos1d-pos1;
vel1r=(vel1d + lambda1*error1);
acc1r=(acc1d - lambda1*(vel1-vel1d));
e1=(vel1-vel1r);
/*----------------------------------------------------------------------------------------------------*/
                                        /*LINK 2 */
/*----------------------------------------------------------------------------------------------------*/
error2=pos2d-pos2;
vel2r=(vel2d + lambda2*error2);
```

```
acc2r=(acc2d - lambda2*(vel2-vel2d));
e2=(vel2-vel2r);
/*--------------------------------------------------------------------------------------------------------*/
                              /*TORQUE VALUES FOR LINK 1 */
/*--------------------------------------------------------------------------------------------------------*/
torque1=(p1cap+p2cap*cos(pos2))*acc1r+(p3cap+0.5*p2cap*cos(pos2))*acc2r+
        (p4cap-p2cap*sin(pos2)*vel2)*vel1r-0.5*p2cap*sin(pos2)*vel2*vel2r+
         p6cap*cos(pos1)+(p7cap*cos(pos1+pos2))+(0.1498*acc1r)-kd1*e1;
c1=     (p4-p2*sin(pos2)*vel2)*vel1-0.5*p2*sin(pos2)*vel2*vel2+
         p6*cos(pos1)+(p7*cos(pos1+pos2));
con11= (p1+p2*cos(pos2)+0.1498);
con12= (p3+0.5*p2*cos(pos2));
/*--------------------------------------------------------------------------------------------------------*/
                    /*CALCULATE ESTIMATED TORQUE FOR LINK 2*/
/*--------------------------------------------------------------------------------------------------------*/
torque2=(0.5*p2cap*cos(pos2)+p3cap)*acc1r+p3cap*acc2r+(0.1498*acc2r)+
    0.5*p2cap*sin(pos2)*vel1*vel1r+p5cap*vel2r+p7cap*cos(pos1+pos2)-kd2*e2;
c2=     0.5*p2*sin(pos2)*vel1*vel1+p5*vel2+p7*cos(pos1+pos2);
con21= (0.5*p2*cos(pos2)+p3);
con22= (p3+0.1498);
/*--------------------------------------------------------------------------------------------------------*/
                                   /*STORE VALUES*/
/*--------------------------------------------------------------------------------------------------------*/
if(k==4)
{
error1s[n]=error1*180.0/pi;
error2s[n]=error2*180.0/pi;
n=n+1;
}
/*--------------------------------------------------------------------------------------------------------*/
                                    /* W MATRIX */
/*--------------------------------------------------------------------------------------------------------*/
w11=acc1r;
w12=(cos(pos2))*(acc1r+0.5*acc2r)-(0.5*sin(pos2)*vel2*(vel1r+vel2r))-
0.5*sin(pos2)*vel1*vel2r;
w13=acc2r;
w14=vel1r;
w16=cos(pos1);
w17=cos(pos1+pos2);
w22=0.5*(cos(pos2)*vel1r+sin(pos2)*vel1*vel1r);
w23=acc1r+acc2r;
w25=vel2r;
w27=cos(pos1+pos2);
/*--------------------------------------------------------------------------------------------------------*/
                                     /*PCAPDOT*/
/*--------------------------------------------------------------------------------------------------------*/
```

```
p1capdot= -1.0*(gama1*w11*e1);
p2capdot= -1.0*((gama2*w12*e1)+(gama2*w22*e2));
p3capdot= -1.0*((gama3*w13*e1)+(gama3*w23*e2));
p4capdot= -1.0*(gama4*w14*e1);
p5capdot= -1.0*(gama5*w25*e2);
p6capdot= -1.0*(gama6*w16*e1);
p7capdot= -1.0*((gama7*w17*e1)+(gama7*w27*e2));
/*-------------------------------------------------------------------------------------------------------*/
                    /*UPDATE THE ESTIMATED VALUES*/
/*-------------------------------------------------------------------------------------------------------*/
p1cap=(tsp*p1capdot)+p1cap;
p2cap=(tsp*p2capdot)+p2cap;
p3cap=(tsp*p3capdot)+p3cap;
p4cap=(tsp*p4capdot)+p4cap;
p5cap=(tsp*p5capdot)+p5cap;
p6cap=(tsp*p6capdot)+p6cap;
p7cap=(tsp*p7capdot)+p7cap;

/*-------------------------------------------------------------------------------------------------------*/
          /* CALCULATE ACTUAL POSITION / VELOCITY OF LINKS*/
/*-------------------------------------------------------------------------------------------------------*/
acc2=(((torque1-c1)*con21)-((torque2-c2)*con11))/((con12*con21)-(con11*con22));
acc1=(((torque1-c1)*con22)-((torque2-c2)*con12))/((con11*con22)-(con12*con21));
vel1=acc1*tsp+vel1;
vel2=acc2*tsp+vel2;
pos1=pos1+vel1*tsp+0.5*acc1*tsp*tsp;
pos2=pos2+vel2*tsp+0.5*acc2*tsp*tsp;
k=k+1;
}
seg=seg+1;
if (seg==2 || seg==4)
{
t=0.0;
p1=izz1n+izz2n+m2n*(x2n*x2n+l1*l1)+m1n*x1n*x1n;
p2=2.0*m2n*l1*x2n;
p3=m2n*x2n*x2n+izz2n;
p4=v1;
p5=v2;
p6=g*(m1n*x1n+m2n*l1);
p7=m2n*x2n*g;
/*-------------------------------------------------------------------------------------------------------*/
  /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY FOR THE SECOND
                                SEGMENT*/
/*-------------------------------------------------------------------------------------------------------*/
a0=thf1;
b0=thf2;
```

```
a3=-10.0*(thf1)/(tf*tf*tf);
a4=15.0*(thf1)/(tf*tf*tf*tf);
a5=-6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=-10.0*(thf2)/(tf*tf*tf);
b4=15.0*(thf2)/(tf*tf*tf*tf);
b5=-6.0*(thf2)/(tf*tf*tf*tf*tf);
k=5;
}
if (seg==3)
{
t=0.0;
k=5;
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
a0=0.0;
b0=0.0;
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
}
}
/*-----------------------------------------------------------------------------------------------------*/
                                        /*OPEN FILES*/
/*-----------------------------------------------------------------------------------------------------*/
er1=fopen("scerror1.dat","w+");
er2=fopen("scerror2.dat","w+");
/*-----------------------------------------------------------------------------------------------------*/
                                        /*WRITE FILES*/
/*-----------------------------------------------------------------------------------------------------*/
for(i=0;i<=(n-1);i++)
{
fprintf(er1,"%f\n",error1s[i]);
fprintf(er2,"%f\n",error2s[i]);
}
fcloseall();
END:;
/*-----------------------------------------------------------------------------------------------------*/
}
```

A.3 ACDID+FUZZY CONTROL CODE

/* This control algorithm controls a two-link manipulator. Path update rate is 333 Hz. For each setpoint the control loop is executed 5 times.*/

```
/*---------------------------------------------------------------------------------------------------*/
                    /* ACDID + FUZZY */
                    /*SIMULATION*/
            /* POSITION + VELOCITY FEEDBACK */
/*---------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <float.h>
#define pi 3.142857
/*---------------------------------------------------------------------------------------------------*/
                /* TRAJECTORY IS QUINTIC */
/*---------------------------------------------------------------------------------------------------*/
void main()
{
int    n=0,i=0,k=5,row,row2,col,col2,seg=1;
float  tf,t=0.0,ts=0.003,tsp,kd1=50.0,kd2=50.0,lambda1=100.0,lambda2=100.0,g=9.8,
      vel1=0.0,vel2=0.0,l1=0.26,thf1,thf2,a3,a4,a5,b3,b4,b5,a0,b0,
      pos1,pos2,p1cap,p2cap,p3cap,p4cap,p5cap,p6cap,p7cap,izz2cap=0.05,
      p1,p2,p3,p4,p5,p6,p7,izz2=0.09,c1,c2,con11,con12,con21,con22,
      error1s[5000],error2s[5000],w11,w12,w13,w14,w16,w17,pos1o,pos2o,
      gama1=100.0,gama2=100.0,gama3=100.0,gama4=100.0,gama5=100.0,
      gama6=100.0, pos1d,pos2d,vel1d,vel2d,acc1d,acc2d,vel1r,vel2r,acc1r,acc2r,
      gama7=100.0,error1,error2,e1,e2,torque1,torque2,w22,w23,w25,
      w27,p1capdot,p2capdot,p3capdot,p4capdot,p5capdot,p6capdot,
      p7capdot,izz1cap=0.05,izz1=0.09,
      m1cap=1.0,m1=2.0,m2cap=1.0,m2=2.0,v1cap=2.0,v1=2.5,v2cap=2.0,v2=2.5,
      x1cap=0.11,x1=0.13,x2cap=0.12,x2=0.14,acc1=0.0,acc2=0.0,nep,nev,dnc,
      ep1,ep2,ev1,ev2,count1,count2,izz1n=1.5,izz2n=0.09,m1n=3.0,m2n=3.0,
      x1n=0.15,x2n=0.16;
FILE   *er1,*er2;
/*---------------------------------------------------------------------------------------------------*/
                /*FUZZY LOOK UP TABLE*/
/*---------------------------------------------------------------------------------------------------*/
float lookt[13] [13]=
                {
                {-5.6, -5.4, -5.0, -4.8, -4.8, -4.7, -4.7, -4.6, -4.5, -4.4, -4.3, -4.3, -4.2},
                {-4.7, -4.5, -4.4, -4.3, -4.2, -4.1, -4.0, -3.9, -3.8, -3.8, -3.7, -3.6, -3.5},
                {-3.7, -3.6, -3.5, -3.2, -3.0, -3.0, -3.0, -2.9, -2.9, -2.8, -2.8, -2.7, -2.7},
                {-2.0, -2.0, -1.9, -1.9, -1.8, -1.8, -1.7, -1.7, -1.6, -1.5, -1.4, -1.3, -1.3},
                { 0.0,  0.0, -0.8, -1.0, -1.2, -1.7, -2.3, -2.2, -2.2, -2.0, -2.0, -1.0, -1.0},
                { 1.0,  1.0,  0.0,  0.0, -0.5, -0.5, -0.5, -1.0, -1.2, -1.5, -1.7, -1.0, -1.0},
                { 1.3,  1.2,  1.0,  0.8,  0.6,  0.0, -0.2, -0.4, -0.6, -0.8, -1.0, -1.0, -1.0},
                { 2.0,  2.0,  1.9,  1.8,  1.8,  1.8,  1.8,  1.8,  1.5,  0.0, -0.3, -1.0, -0.8},
                { 2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  1.2,  0.8,  0.0,  0.0},
                { 2.0,  2.1,  2.3,  2.5,  2.5,  2.5,  2.6,  2.7,  2.8,  2.8,  2.9,  2.9,  3.0},
                { 2.7,  2.7,  2.8,  3.1,  3.2,  3.3,  3.5,  3.6,  3.6,  3.8,  3.8,  3.9,  3.9},
                { 3.6,  3.3,  3.7,  4.0,  4.1,  4.3,  4.3,  4.4,  4.4,  4.5,  4.5,  4.6,  4.7},
                { 4.4,  4.4,  4.3,  4.8,  5.0,  5.0,  5.1,  5.2,  5.3,  5.4,  5.6,  5.6,  5.6}
                };
clrscr();
/*---------------------------------------------------------------------------------------------------*/
                                        /*GET INPUTS*/
/*---------------------------------------------------------------------------------------------------*/
tsp=ts/5.0;
highvideo();
textbackground(YELLOW);
window(10,5,80,25);
textcolor(CYAN);
cprintf("TRAJECTORY IS QUINTIC POLYNOMIAL || Tf(max)= 5sec");
window(15,10,80,25);
textcolor(CYAN);
cprintf("FINAL TIME                          ");
scanf("%f",&tf);
/*window(15,11,80,25);
textcolor(GREEN);
cprintf("FIRST ELEMENT OF GAIN MATRIX          ");
scanf("%f", &kd1);
window(15,12,80,25);
cprintf("SECOND ELEMENT OF GAIN MATRIX          ");
scanf("%f", &kd2); */
window(15,14,80,25);
cprintf("ANGLE OF THE FIRST JOINT IN DEGREES(+) ");
scanf("%f",&thf1);
window(15,15,80,25);
cprintf("ANGLE OF THE SECOND JOINT IN DEGREES(-)");
scanf("%f",&thf2);
/*---------------------------------------------------------------------------------------------------*/
                                        /* INITIALISE */
/*---------------------------------------------------------------------------------------------------*/
thf1=(thf1*pi)/180.0;
thf2=(thf2*pi)/180.0;
if(thf1 < 0.0 || thf2 > 0.0)
{
window(15,18,80,25);
cprintf("GIVE CORRECT ANGLES                    ");
goto END;
```

```
}
a0=0.0;
b0=0.0;
/*-----------------------------------------------------------------------------------------------*/
              /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY*/
/*-----------------------------------------------------------------------------------------------*/
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
pos1=0.0;
pos2=0.0;
nep=108.0/pi;       /*Normalisation factor for position error*/
nev=10.8/(pi);  /*Normalisation factor for error dot*/
dnc=255.0/5.6;       /*Denormalisation factor for voltage*/
/*-----------------------------------------------------------------------------------------------*/
                              /*ESTIMATE OF PARAMETERS */
/*-----------------------------------------------------------------------------------------------*/
p1cap=izz1cap+izz2cap+m2cap*(x2cap*x2cap+l1*l1)+m1cap*x1cap*x1cap;
p2cap=2.0*m2cap*l1*x2cap;
p3cap=m2cap*x2cap*x2cap+izz2cap;
p4cap=v1cap;
p5cap=v2cap;
p6cap=g*(m1cap*x1cap+m2cap*l1);
p7cap=m2cap*x2cap*g;
/*-----------------------------------------------------------------------------------------------*/
                                /*ACTUAL PARAMETERS*/
/*-----------------------------------------------------------------------------------------------*/
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
window(15,19,80,25);
cprintf("BUSY WITH CONTROL                \n");
/*-----------------------------------------------------------------------------------------------*/
                                    /*SERVO LOOP*/
/*-----------------------------------------------------------------------------------------------*/
for (;seg<=4;)
{
for (;t<=tf;)
{
```

```
                    /* DESIRED POSITION,VELOCITY,ACCELERATION*/
if(k==5)
{
pos1d=a0+(a3*t*t*t)+(a4*t*t*t*t)+(a5*t*t*t*t*t);
pos2d=b0+(b3*t*t*t)+(b4*t*t*t*t)+(b5*t*t*t*t*t);
vel1d=(3.0*a3*t*t)+(4.0*a4*t*t*t)+(5.0*a5*t*t*t*t);
vel2d=(3.0*b3*t*t)+(4.0*b4*t*t*t)+(5.0*b5*t*t*t*t);
acc1d=(6.0*a3*t)+(12.0*a4*t*t)+(20.0*a5*t*t*t);
acc2d=(6.0*b3*t)+(12.0*b4*t*t)+(20.0*b5*t*t*t);
t=t+ts;
k=0;
}
error1=pos1d-pos1;
vel1r=(vel1d + lambda1*error1);
acc1r=(acc1d - lambda1*(vel1-vel1d));
e1=(vel1-vel1r);

/*------------------------------------------------------------------------------------------------------*/
                                        /*LINK 2 */
/*------------------------------------------------------------------------------------------------------*/
error2=pos2d-pos2;
vel2r=(vel2d + lambda2*error2);
acc2r=(acc2d - lambda2*(vel2-vel2d));
e2=(vel2-vel2r);
ep1=(pos1d-pos1)*nep;  /*Normalised errors*/
ev1=(vel1d-vel1)*nev;
ep2=(pos2d-pos2)*nep;
ev2=(vel2d-vel2)*nev;
/*------------------------------------------------------------------------------------------------------*/
                                   /*LABEL THE ERROR*/
/*------------------------------------------------------------------------------------------------------*/
if (ep1 <= 4.8){row=0;}
if (-4.8 < ep1 && ep1 <= -3.6){row=1;}
if (-3.6 < ep1 && ep1 <= -2.4){row=2;}
if (-2.4 < ep1 && ep1 <= -1.2){row=3;}
if (-1.2 < ep1 && ep1 <= -0.6){row=4;}
if (-0.6 < ep1 && ep1 <= -0.1){row=5;}
if (-0.1 < ep1 && ep1 <=  0.1){row=6;}
if ( 0.1 < ep1 && ep1 <=  0.6){row=7;}
if ( 0.6 < ep1 && ep1 <=  1.2){row=8;}
if ( 1.2 < ep1 && ep1 <=  2.4){row=9;}
if ( 2.4 < ep1 && ep1 <= 3.6){row=10;}
if ( 3.6 < ep1 && ep1 <= 4.8){row=11;}
if ( 4.8 < ep1){row=12;}
                                     /*SECOND LINK*/
if (ep2 <= 4.8){row2=0;}
if (-4.8 < ep2 && ep2 <= -3.6){row2=1;}
if (-3.6 < ep2 && ep2 <= -2.4){row2=2;}
if (-2.4 < ep2 && ep2 <= -1.2){row2=3;}
if (-1.2 < ep2 && ep2 <= -0.6){row2=4;}
if (-0.6 < ep2 && ep2 <= -0.1){row2=5;}
if (-0.1 < ep2 && ep2 <=  0.1){row2=6;}
if ( 0.1 < ep2 && ep2 <=  0.6){row2=7;}
if ( 0.6 < ep2 && ep2 <=  1.2){row2=8;}
if ( 1.2 < ep2 && ep2 <=  2.4){row2=9;}
if ( 2.4 < ep2 && ep2 <= 3.6){row2=10;}
if ( 3.6 < ep2 && ep2 <= 4.8){row2=11;}
if ( 4.8 < ep2){row2=12;}
/*----------------------------------------------------------------------------------------------------*/
                                        /*LABEL EDOT*/
/*----------------------------------------------------------------------------------------------------*/
if (ev1 <= 4.8){col=0;}
if (-4.8 < ev1 && ev1 <= -3.6){col=1;}
if (-3.6 < ev1 && ev1 <= -2.4){col=2;}
if (-2.4 < ev1 && ev1 <= -1.2){col=3;}
if (-1.2 < ev1 && ev1 <= -0.6){col=4;}
if (-0.6 < ev1 && ev1 <= -0.1){col=5;}
if (-0.1 < ev1 && ev1 <=  0.1){col=6;}
if ( 0.1 < ev1 && ev1 <=  0.6){col=7;}
if ( 0.6 < ev1 && ev1 <=  1.2){col=8;}
if ( 1.2 < ev1 && ev1 <=  2.4){col=9;}
if ( 2.4 < ev1 && ev1 <= 3.6){col=10;}
if ( 3.6 < ev1 && ev1 <= 4.8){col=11;}
if ( 4.8 < ev1){col=12;}
                                        /*SECOND LINK*/
if (ev2 <= 4.8){col2=0;}
if (-4.8 < ev2 && ev2 <= -3.6){col2=1;}
if (-3.6 < ev2 && ev2 <= -2.4){col2=2;}
if (-2.4 < ev2 && ev2 <= -1.2){col2=3;}
if (-1.2 < ev2 && ev2 <= -0.6){col2=4;}
if (-0.6 < ev2 && ev2 <= -0.1){col2=5;}
if (-0.1 < ev2 && ev2 <=  0.1){col2=6;}
if ( 0.1 < ev2 && ev2 <=  0.6){col2=7;}
if ( 0.6 < ev2 && ev2 <=  1.2){col2=8;}
if ( 1.2 < ev2 && ev2 <=  2.4){col2=9;}
if ( 2.4 < ev2 && ev2 <= 3.6){col2=10;}
if ( 3.6 < ev2 && ev2 <= 4.8){col2=11;}
if ( 4.8 < ev2){col2=12;}
count1=(lookt[row] [col])*dnc;
count2=(lookt[row2] [col2])*dnc;
```

```
/*----------------------------------------------------------------------------------------------------*/
                        /*TORQUE VALUES FOR LINK 1 */
/*----------------------------------------------------------------------------------------------------*/
torque1=(((((24.0*count1)/255.0)-0.066*(vel1*80.0))*0.066)/2.32)*80.0;
/*----------------------------------------------------------------------*/
                /*CALCULATE ESTIMATED TORQUE FOR LINK 2*/
/*----------------------------------------------------------------------------------------------------*/
torque2=(((((24.0*count2)/255.0)-0.066*(vel2*70.0))*0.066)/2.32)*70.0;
/*----------------------------------------------------------------------------------------------------*/
                        /*TORQUE VALUES FOR LINK 1 */
/*----------------------------------------------------------------------------------------------------*/
torque1=(p1cap+p2cap*cos(pos2))*acc1r+(p3cap+0.5*p2cap*cos(pos2))*acc2r+
        (p4cap-p2cap*sin(pos2)*vel2)*vel1r-0.5*p2cap*sin(pos2)*vel2*vel2r+
         p6cap*cos(pos1)+(p7cap*cos(pos1+pos2))+(0.1498*acc1r)-kd1*e1+torque1;
/*if(torque1 > (80.0*10.5*0.066))
{torque1=(80.0*10.5*0.066);}
if(torque1 < (-80.0*10.5*0.066))
{torque1=-(80.0*10.5*0.066);}*/
c1=     (p4-p2*sin(pos2)*vel2)*vel1-0.5*p2*sin(pos2)*vel2*vel2+
         p6*cos(pos1)+(p7*cos(pos1+pos2));
con11= (p1+p2*cos(pos2)+0.1498);
con12= (p3+0.5*p2*cos(pos2));
/*----------------------------------------------------------------------------------------------------*/
                /*CALCULATE ESTIMATED TORQUE FOR LINK 2*/
/*----------------------------------------------------------------------------------------------------*/
torque2=(0.5*p2cap*cos(pos2)+p3cap)*acc1r+p3cap*acc2r+(0.1498*acc2r)+
    0.5*p2cap*sin(pos2)*vel1*vel1r+p5cap*vel2r+p7cap*cos(pos1+pos2)-
kd2*e2+torque2;
c2=     0.5*p2*sin(pos2)*vel1*vel1+p5*vel2+p7*cos(pos1+pos2);
con21= (0.5*p2*cos(pos2)+p3);
con22= (p3+0.1498);
/*----------------------------------------------------------------------------------------------------*/
                             /*STORE VALUES*/
/*----------------------------------------------------------------------------------------------------*/
if(k==4)
{
error1s[n]=error1*180.0/pi;
error2s[n]=error2*180.0/pi;
n=n+1;
}
/*----------------------------------------------------------------------------------------------------*/
                              /* W MATRIX */
/*----------------------------------------------------------------------------------------------------*/
w11=acc1r;
w12=(cos(pos2))*(acc1r+0.5*acc2r)-(0.5*sin(pos2)*vel2*(vel1r+vel2r))-
0.5*sin(pos2)*vel1*vel2r;
```

```
w13=acc2r;
w14=vel1r;
w16=cos(pos1);
w17=cos(pos1+pos2);
w22=0.5*(cos(pos2)*vel1r+sin(pos2)*vel1*vel1r);
w23=acc1r+acc2r;
w25=vel2r;
w27=cos(pos1+pos2);
/*--------------------------------------------------------------------------------------------------------*/
                                        /*PCAPDOT*/
/*--------------------------------------------------------------------------------------------------------*/
p1capdot= -1.0*(gama1*w11*e1);
p2capdot= -1.0*((gama2*w12*e1)+(gama2*w22*e2));
p3capdot= -1.0*((gama3*w13*e1)+(gama3*w23*e2));
p4capdot= -1.0*(gama4*w14*e1);
p5capdot= -1.0*(gama5*w25*e2);
p6capdot= -1.0*(gama6*w16*e1);
p7capdot= -1.0*((gama7*w17*e1)+(gama7*w27*e2));
/*--------------------------------------------------------------------------------------------------------*/
                          /*UPDATE THE ESTIMATED VALUES*/
/*--------------------------------------------------------------------------------------------------------*/
p1cap=(tsp*p1capdot)+p1cap;
p2cap=(tsp*p2capdot)+p2cap;
p3cap=(tsp*p3capdot)+p3cap;
p4cap=(tsp*p4capdot)+p4cap;
p5cap=(tsp*p5capdot)+p5cap;
p6cap=(tsp*p6capdot)+p6cap;
p7cap=(tsp*p7capdot)+p7cap;
/*--------------------------------------------------------------------------------------------------------*/
             /* CALCULATE ACTUAL POSITION / VELOCITY OF LINKS*/
/*--------------------------------------------------------------------------------------------------------*/
acc2=(((torque1-c1)*con21)-((torque2-c2)*con11))/((con12*con21)-(con11*con22));
acc1=(((torque1-c1)*con22)-((torque2-c2)*con12))/((con11*con22)-(con12*con21));
vel1=acc1*tsp+vel1;
vel2=acc2*tsp+vel2;
pos1=pos1+vel1*tsp+0.5*acc1*tsp*tsp;
pos2=pos2+vel2*tsp+0.5*acc2*tsp*tsp;
k=k+1;
}
seg=seg+1;
t=0.0;
pos1o=pos1;
pos2o=pos2;
```

```
/*------------------------------------------------------------------------------------------------------------*/
    /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY FOR THE SECOND
                                  SEGMENT*/
/*------------------------------------------------------------------------------------------------------------*/

if(seg==2 || seg==4)
{
p1=izz1n+izz2n+m2n*(x2n*x2n+l1*l1)+m1n*x1n*x1n;
p2=2.0*m2n*l1*x2n;
p3=m2n*x2n*x2n+izz2n;
p4=v1;
p5=v2;
p6=g*(m1n*x1n+m2n*l1);
p7=m2n*x2n*g;
a0=thf1;
b0=thf2;
a3=-10.0*(thf1)/(tf*tf*tf);
a4=15.0*(thf1)/(tf*tf*tf*tf);
a5=-6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=-10.0*(thf2)/(tf*tf*tf);
b4=15.0*(thf2)/(tf*tf*tf*tf);
b5=-6.0*(thf2)/(tf*tf*tf*tf*tf);
k=5;
}
if(seg==3)
{
t=0.0;
a0=0.0;
b0=0.0;
/*thf1=0.0;
thf2=0.0; */
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
k=5;
}
```

```
}
/*-----------------------------------------------------------------------------------------------------------*/
                                        /*OPEN FILES*/
/*-----------------------------------------------------------------------------------------------------------*/
er1=fopen("scerror1.dat","w+");
er2=fopen("scerror2.dat","w+");
/*-----------------------------------------------------------------------------------------------------------*/
                                        /*WRITE FILES*/
/*-----------------------------------------------------------------------------------------------------------*/
for(i=0;i<=(n-1);i++)
{
fprintf(er1,"%f\n",error1s[i]);
fprintf(er2,"%f\n",error2s[i]);
}
fcloseall();
END:;
/*-----------------------------------------------------------------------------------------------------------*/
}
```

A.4 SOC FUZZY CONTROL CODE

```
/* This control algorithm controls a two-link manipulator. Path update rate is 333 Hz. For
each setpoint the control loop is executed 5 times.*/
/*-----------------------------------------------------------------------------------------------------*/
                              /*SIMULATION*/
                              /* SOC FUZZY */
/*-----------------------------------------------------------------------------------------------------*/
#include <stdio.h>
#include <math.h>
#include <conio.h>
#include <stdlib.h>
#include <float.h>
#define pi 3.142857
/*-----------------------------------------------------------------------------------------------------*/
                         /* TRAJECTORY IS QUINTIC */
/*-----------------------------------------------------------------------------------------------------*/
void main()
{
/*-----------------------------------------------------------------------------------------------------*/
                          /*FUZZY LOOK UP TABLE*/
/*-----------------------------------------------------------------------------------------------------*/
float lookp[13] [13]={
            {-6, -6, -6, -6, -6, -6, -6, -5, -4, -3, -2, -1,  0},
            {-6, -6, -6, -6, -5, -4, -4, -4, -3, -2, -1,  0,  0},
            {-6, -6, -6, -5, -4, -3, -3, -3, -2, -1,  0,  0,  1},
            {-6, -6, -5, -4, -3, -2, -2, -2, -1,  0,  0,  1,  2},
            {-6, -5, -4, -3, -2, -1, -1, -1,  0,  0,  1,  2,  3},
            {-5, -4, -3, -2, -1, -1,  0,  0,  0,  1,  2,  3,  4},
            {-5, -4, -3, -2, -1,  0,  0,  0,  1,  2,  3,  4,  5},
            {-3, -2, -1,  0,  0,  0,  0,  1,  1,  2,  3,  4,  5},
            {-2, -1,  0,  0,  0,  1,  1,  1,  2,  3,  4,  5,  6},
            {-1,  0,  0,  0,  1,  2,  2,  2,  3,  4,  5,  6,  6},
            { 0,  0,  0,  1,  2,  3,  3,  3,  4,  5,  6,  6,  6},
            { 0,  0,  1,  2,  3,  4,  4,  4,  5,  6,  6,  6,  6},
            { 0,  1,  2,  3,  4,  5,  6,  6,  6,  6,  6,  6,  6}
            };
float lookt[13] [13]={
            {-5.6, -5.4, -5.0, -4.8, -4.8, -4.7, -4.7, -4.6, -4.5, -4.4, -4.3, -4.3, -4.2},
            {-4.7, -4.5, -4.4, -4.3, -4.2, -4.1, -4.0, -3.9, -3.8, -3.8, -3.7, -3.6, -3.5},
            {-3.7, -3.6, -3.5, -3.2, -3.0, -3.0, -3.0, -2.9, -2.9, -2.8, -2.8, -2.7, -2.7},
            {-2.0, -2.0, -1.9, -1.9, -1.8, -1.8, -1.7, -1.7, -1.6, -1.5, -1.4, -1.3, -1.3},
            { 0.0,  0.0, -0.8, -1.0, -1.2, -1.7, -2.3, -2.2, -2.2, -2.0, -2.0, -1.0, -1.0},
            { 1.0,  1.0,  0.0,  0.0, -0.5, -0.5, -0.5, -1.0, -1.2, -1.5, -1.7, -1.0, -1.0},
            { 1.3,  1.2,  1.0,  0.8,  0.6,  0.0, -0.2, -0.4, -0.6, -0.8, -1.0, -1.0, -1.0},
            { 2.0,  2.0,  1.9,  1.8,  1.8,  1.8,  1.8,  1.8,  1.5,  0.0, -0.3, -1.0, -0.8},
                    { 2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  1.2,  0.8,  0.0,  0.0},
                    { 2.0,  2.1,  2.3,  2.5,  2.5,  2.5,  2.6,  2.7,  2.8,  2.8,  2.9,  2.9,  3.0},
                    { 2.7,  2.7,  2.8,  3.1,  3.2,  3.3,  3.5,  3.6,  3.6,  3.8,  3.8,  3.9,  3.9},
                    { 3.6,  3.3,  3.7,  4.0,  4.1,  4.3,  4.3,  4.4,  4.4,  4.5,  4.5,  4.6,  4.7},
                    { 4.4,  4.4,  4.3,  4.8,  5.0,  5.0,  5.1,  5.2,  5.3,  5.4,  5.6,  5.6,  5.6}
                    };
/*float lookt[13] [13]={
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0}
                    }; */

float lookt2[13] [13]={
                    {-5.6, -5.4, -5.0, -4.8, -4.8, -4.7, -4.7, -4.6, -4.5, -4.4, -4.3, -4.3, -4.2},
                    {-4.7, -4.5, -4.4, -4.3, -4.2, -4.1, -4.0, -3.9, -3.8, -3.8, -3.7, -3.6, -3.5},
                    {-3.7, -3.6, -3.5, -3.2, -3.0, -3.0, -3.0, -2.9, -2.9, -2.8, -2.8, -2.7, -2.7},
                    {-2.0, -2.0, -1.9, -1.9, -1.8, -1.8, -1.7, -1.7, -1.6, -1.5, -1.4, -1.3, -1.3},
                    { 0.0,  0.0, -0.8, -1.0, -1.2, -1.7, -2.3, -2.2, -2.2, -2.0, -2.0, -1.0, -1.0},
                    { 1.0,  1.0,  0.0,  0.0, -0.5, -0.5, -0.5, -1.0, -1.2, -1.5, -1.7, -1.0, -1.0},
                    { 1.3,  1.2,  1.0,  0.8,  0.6,  0.0, -0.2, -0.4, -0.6, -0.8, -1.0, -1.0, -1.0},
                    { 2.0,  2.0,  1.9,  1.8,  1.8,  1.8,  1.8,  1.8,  1.5,  0.0, -0.3, -1.0, -0.8},
                    { 2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  2.0,  1.2,  0.8,  0.0,  0.0},
                    { 2.0,  2.1,  2.3,  2.5,  2.5,  2.5,  2.6,  2.7,  2.8,  2.8,  2.9,  2.9,  3.0},
                    { 2.7,  2.7,  2.8,  3.1,  3.2,  3.3,  3.5,  3.6,  3.6,  3.8,  3.8,  3.9,  3.9},
                    { 3.6,  3.3,  3.7,  4.0,  4.1,  4.3,  4.3,  4.4,  4.4,  4.5,  4.5,  4.6,  4.7},
                    { 4.4,  4.4,  4.3,  4.8,  5.0,  5.0,  5.1,  5.2,  5.3,  5.4,  5.6,  5.6,  5.6}
                    };
/*float lookt2[13] [13]={
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0},
                    {0,0,0,0,0,0,0,0,0,0,0,0,0}
                    }; */

int    n=0,i=0,j=0,k=5,row=0,row2=0,col=0,col2=0,rowp,row2p,colp,col2p,seg=1;
float  tf,t=0.0,ts=0.003,tsp,g=9.8,a0=0.0,b0=0.0,pos1o,pos2o,vel1=0.0,vel2=0.0,
       l1=0.26,thf1,thf2,a3,a4,a5,b3,b4,b5,pos1,pos2,p1,p2,p3,p4,p5,p6,p7,
       izz2=0.09,c1,c2,con11,con12,con21,con22,error1s[5000],error2s[5000],
       acc1,acc2,pos1d,pos2d,vel1d,vel2d,ep1=0.0,ep2=0.0,ev1=0.0,ev2=0.0,
       izz1=0.09,count1,count2,m1=2.0,m2=2.0,v1=2.5,v2=2.5,x1=0.13,x2=0.14,
       nep,nev,dnc,izz1n=1.5,izz2n=0.09,m1n=3.0,m2n=3.0,x1n=0.15,x2n=0.16,ep1s=0.0,
       ep2s=0.0,ki1=0.0,ki2=0.0, torque1,torque2,;
FILE   *er1,*er2;
clrscr();
/*-----------------------------------------------------------------------------------------------------*/
                                   /*GET INPUTS*/
/*-----------------------------------------------------------------------------------------------------*/
tsp=ts/5.0;
highvideo();
textbackground(YELLOW);
window(10,5,80,25);
textcolor(CYAN);
cprintf("TRAJECTORY IS QUINTIC POLYNOMIAL || Tf(max)= 5sec");
window(15,10,80,25);
textcolor(CYAN);
cprintf("FINAL TIME                          ");
scanf("%f",&tf);
window(15,14,80,25);
cprintf("ANGLE OF THE FIRST JOINT IN DEGREES(+) ");
scanf("%f",&thf1);
window(15,15,80,25);
cprintf("ANGLE OF THE SECOND JOINT IN DEGREES(-)");
scanf("%f",&thf2);
/*-----------------------------------------------------------------------------------------------------*/
                                   /* INITIALISE */
/*-----------------------------------------------------------------------------------------------------*/
thf1=(thf1*pi)/180.0;
thf2=(thf2*pi)/180.0;
if(thf1 < 0.0 || thf2 > 0.0)
{
window(15,18,80,25);
cprintf("GIVE CORRECT ANGLES                   ");
goto END;
```

```
}
/*---------------------------------------------------------------------------------------------------------*/
                 /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY*/
/*---------------------------------------------------------------------------------------------------------*/
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
pos1=0.0;
pos2=0.0;
nep=1080.0/(pi);        /*Normalisation factor for position error*/
nev=1.0*10.8/pi;  /*Normalisation factor for error dot*/
dnc=255.0/6.0;       /*Denormalisation factor for voltage*/
/*---------------------------------------------------------------------------------------------------------*/
                              /*ACTUAL PARAMETERS*/
/*---------------------------------------------------------------------------------------------------------*/
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
window(15,19,80,25);
cprintf("BUSY WITH CONTROL                    \n");
a0=0.0;
b0=0.0;
/*---------------------------------------------------------------------------------------------------------*/
                                  /*SERVO LOOP*/
/*---------------------------------------------------------------------------------------------------------*/
for (;seg<=4;)
{
for (;t<=tf;)
{
                 /* DESIRED POSITION,VELOCITY,ACCELERATION*/
if(k==5)
{
pos1d=a0+(a3*t*t*t)+(a4*t*t*t*t)+(a5*t*t*t*t*t);
pos2d=b0+(b3*t*t*t)+(b4*t*t*t*t)+(b5*t*t*t*t*t);
vel1d=(3.0*a3*t*t)+(4.0*a4*t*t*t)+(5.0*a5*t*t*t*t);
vel2d=(3.0*b3*t*t)+(4.0*b4*t*t*t)+(5.0*b5*t*t*t*t);
t=t+ts;
k=0;
}
```

```
ep1=(pos1d-pos1)*nep;
ep2=(pos2d-pos2)*nep;
ev1=(vel1d-vel1)*nev;
ev2=(vel2d-vel2)*nev;
/*---------------------------------------------------------------------------------------------------------*/
                                        /*LABEL THE ERROR*/
/*---------------------------------------------------------------------------------------------------------*/
if (ep1 <= -4.8){rowp=0;}
if (-4.8 < ep1 && ep1 <= -3.6){rowp=1;}
if (-3.6 < ep1 && ep1 <= -2.4){rowp=2;}
if (-2.4 < ep1 && ep1 <= -1.2){rowp=3;}
if (-1.2 < ep1 && ep1 <= -0.6){rowp=4;}
if (-0.6 < ep1 && ep1 <= -0.1){rowp=5;}
if (-0.1 < ep1 && ep1 <=  0.1){rowp=6;}
if ( 0.1 < ep1 && ep1 <=  0.6){rowp=7;}
if ( 0.6 < ep1 && ep1 <=  1.2){rowp=8;}
if ( 1.2 < ep1 && ep1 <=  2.4){rowp=9;}
if ( 2.4 < ep1 && ep1 <= 3.6){rowp=10;}
if ( 3.6 < ep1 && ep1 <= 4.8){rowp=11;}
if ( 4.8 < ep1){rowp=12;}
                                           /*SECOND LINK*/
if (ep2 <= -4.8){row2p=0;}
if (-4.8 < ep2 && ep2 <= -3.6){row2p=1;}
if (-3.6 < ep2 && ep2 <= -2.4){row2p=2;}
if (-2.4 < ep2 && ep2 <= -1.2){row2p=3;}
if (-1.2 < ep2 && ep2 <= -0.6){row2p=4;}
if (-0.6 < ep2 && ep2 <= -0.1){row2p=5;}
if (-0.1 < ep2 && ep2 <=  0.1){row2p=6;}
if ( 0.1 < ep2 && ep2 <=  0.6){row2p=7;}
if ( 0.6 < ep2 && ep2 <=  1.2){row2p=8;}
if ( 1.2 < ep2 && ep2 <=  2.4){row2p=9;}
if ( 2.4 < ep2 && ep2 <= 3.6){row2p=10;}
if ( 3.6 < ep2 && ep2 <= 4.8){row2p=11;}
if ( 4.8 < ep2){row2p=12;}
/*---------------------------------------------------------------------------------------------------------*/
                                           /*LABEL EDOT*/
/*---------------------------------------------------------------------------------------------------------*/
if (ev1 <= -4.8){colp=0;}
if (-4.8 < ev1 && ev1 <= -3.6){colp=1;}
if (-3.6 < ev1 && ev1 <= -2.4){colp=2;}
if (-2.4 < ev1 && ev1 <= -1.2){colp=3;}
if (-1.2 < ev1 && ev1 <= -0.6){colp=4;}
if (-0.6 < ev1 && ev1 <= -0.1){colp=5;}
if (-0.1 < ev1 && ev1 <=  0.1){colp=6;}
if ( 0.1 < ev1 && ev1 <=  0.6){colp=7;}
if ( 0.6 < ev1 && ev1 <=  1.2){colp=8;}
if ( 1.2 < ev1 && ev1 <=  2.4){colp=9;}
if ( 2.4 < ev1 && ev1 <= 3.6){colp=10;}
if ( 3.6 < ev1 && ev1 <= 4.8){colp=11;}
if ( 4.8 < ev1){colp=12;}
                                        /*SECOND LINK*/
if (ev2 <= -4.8){col2p=0;}
if (-4.8 < ev2 && ev2 <= -3.6){col2p=1;}
if (-3.6 < ev2 && ev2 <= -2.4){col2p=2;}
if (-2.4 < ev2 && ev2 <= -1.2){col2p=3;}
if (-1.2 < ev2 && ev2 <= -0.6){col2p=4;}
if (-0.6 < ev2 && ev2 <= -0.1){col2p=5;}
if (-0.1 < ev2 && ev2 <=  0.1){col2p=6;}
if ( 0.1 < ev2 && ev2 <=  0.6){col2p=7;}
if ( 0.6 < ev2 && ev2 <=  1.2){col2p=8;}
if ( 1.2 < ev2 && ev2 <=  2.4){col2p=9;}
if ( 2.4 < ev2 && ev2 <= 3.6){col2p=10;}
if ( 3.6 < ev2 && ev2 <= 4.8){col2p=11;}
if ( 4.8 < ev2){col2p=12;}
lookt[row][col]=lookt[row][col]+lookp[rowp][colp];
if(lookt[row][col]>6.0)
{lookt[row][col]=6.0;}
if(lookt[row][col]<-6.0)
{lookt[row][col]=-6.0;}
lookt2[row2][col2]=lookt2[row2][col2]+lookp[row2p][col2p];
if(lookt2[row2][col2]>6.0)
{lookt2[row2][col2]=6.0;}
if(lookt2[row2][col2]<-6.0)
{lookt2[row2][col2]=-6.0;}
ep1=(pos1d-pos1)*nep;  /*Normalised errors*/
ev1=(vel1d-vel1)*nev;
ep1s=ep1s+(pos1d-pos1);
/*------------------------------------------------------------------------------------------------------*/
                                           /*LINK 2 */
/*------------------------------------------------------------------------------------------------------*/
ep2=(pos2d-pos2)*nep;
ev2=(vel2d-vel2)*nev;
ep2s=ep2s+(pos2d-pos2);
/*------------------------------------------------------------------------------------------------------*/
                                      /*LABEL THE ERROR*/
/*------------------------------------------------------------------------------------------------------*/
if (ep1 <= -4.8){row=0;}
if (-4.8 < ep1 && ep1 <= -3.6){row=1;}
if (-3.6 < ep1 && ep1 <= -2.4){row=2;}
if (-2.4 < ep1 && ep1 <= -1.2){row=3;}
if (-1.2 < ep1 && ep1 <= -0.6){row=4;}
if (-0.6 < ep1 && ep1 <= -0.1){row=5;}
if (-0.1 < ep1 && ep1 <=  0.1){row=6;}
if ( 0.1 < ep1 && ep1 <=  0.6){row=7;}
if ( 0.6 < ep1 && ep1 <=  1.2){row=8;}
if ( 1.2 < ep1 && ep1 <=  2.4){row=9;}
if ( 2.4 < ep1 && ep1 <= 3.6){row=10;}
if ( 3.6 < ep1 && ep1 <= 4.8){row=11;}
if ( 4.8 < ep1){row=12;}
                                    /*SECOND LINK*/
if (ep2 <= -4.8){row2=0;}
if (-4.8 < ep2 && ep2 <= -3.6){row2=1;}
if (-3.6 < ep2 && ep2 <= -2.4){row2=2;}
if (-2.4 < ep2 && ep2 <= -1.2){row2=3;}
if (-1.2 < ep2 && ep2 <= -0.6){row2=4;}
if (-0.6 < ep2 && ep2 <= -0.1){row2=5;}
if (-0.1 < ep2 && ep2 <=  0.1){row2=6;}
if ( 0.1 < ep2 && ep2 <=  0.6){row2=7;}
if ( 0.6 < ep2 && ep2 <=  1.2){row2=8;}
if ( 1.2 < ep2 && ep2 <=  2.4){row2=9;}
if ( 2.4 < ep2 && ep2 <= 3.6){row2=10;}
if ( 3.6 < ep2 && ep2 <= 4.8){row2=11;}
if ( 4.8 < ep2){row2=12;}
/*---------------------------------------------------------------------------------------------------------*/
                                    /*LABEL EDOT*/
/*---------------------------------------------------------------------------------------------------------*/
if (ev1 <= -4.8){col=0;}
if (-4.8 < ev1 && ev1 <= -3.6){col=1;}
if (-3.6 < ev1 && ev1 <= -2.4){col=2;}
if (-2.4 < ev1 && ev1 <= -1.2){col=3;}
if (-1.2 < ev1 && ev1 <= -0.6){col=4;}
if (-0.6 < ev1 && ev1 <= -0.1){col=5;}
if (-0.1 < ev1 && ev1 <=  0.1){col=6;}
if ( 0.1 < ev1 && ev1 <=  0.6){col=7;}
if ( 0.6 < ev1 && ev1 <=  1.2){col=8;}
if ( 1.2 < ev1 && ev1 <=  2.4){col=9;}
if ( 2.4 < ev1 && ev1 <= 3.6){col=10;}
if ( 3.6 < ev1 && ev1 <= 4.8){col=11;}
if ( 4.8 < ev1){col=12;}
                                    /*SECOND LINK*/
if (ev2 <= -4.8){col2=0;}
if (-4.8 < ev2 && ev2 <= -3.6){col2=1;}
if (-3.6 < ev2 && ev2 <= -2.4){col2=2;}
if (-2.4 < ev2 && ev2 <= -1.2){col2=3;}
if (-1.2 < ev2 && ev2 <= -0.6){col2=4;}
if (-0.6 < ev2 && ev2 <= -0.1){col2=5;}
if (-0.1 < ev2 && ev2 <=  0.1){col2=6;}
if ( 0.1 < ev2 && ev2 <=  0.6){col2=7;}
if ( 0.6 < ev2 && ev2 <=  1.2){col2=8;}
if ( 1.2 < ev2 && ev2 <=  2.4){col2=9;}
if ( 2.4 < ev2 && ev2 <= 3.6){col2=10;}
if ( 3.6 < ev2 && ev2 <= 4.8){col2=11;}
if ( 4.8 < ev2){col2=12;}
count1=(lookt[row] [col])*dnc;
count2=(lookt2[row2] [col2])*dnc;
/*----------------------------------------------------------------------------------------------------------*/
                              /*TORQUE VALUES FOR LINK 1 */
/*----------------------------------------------------------------------------------------------------------*/
torque1=(((((20.0*count1)/255.0)-0.066*(vel1*80.0))*0.066)/2.32)*80.0+ki1*ep1s;
/*if (torque1>55.44)
{torque1=55.44;}
if (torque1<-55.44)
{torque1=-55.44;} */
c1=      (p4-p2*sin(pos2)*vel2)*vel1-0.5*p2*sin(pos2)*vel2*vel2+
          p6*cos(pos1)+(p7*cos(pos1+pos2));
con11= (p1+p2*cos(pos2)+0.1498);
con12= (p3+0.5*p2*cos(pos2));
/*----------------------------------------------------------------------------------------------------------*/
                    /*CALCULATE ESTIMATED TORQUE FOR LINK 2*/
/*----------------------------------------------------------------------------------------------------------*/
torque2=(((((20.0*count2)/255.0)-0.066*(vel2*70.0))*0.066)/2.32)*70.0+ki2*ep2s;
/*if (torque2>48.51)
{torque2=48.51;}
if (torque2<-48.51)
{torque2=-48.51;}  */
c2=      0.5*p2*sin(pos2)*vel1*vel1+p5*vel2+p7*cos(pos1+pos2);
con21= (0.5*p2*cos(pos2)+p3);
con22= (p3+0.1498);
/*----------------------------------------------------------------------------------------------------------*/
                                   /*STORE VALUES*/
/*----------------------------------------------------------------------------------------------------------*/
if(k==4)
{
error1s[n]=ep1*180.0/(pi*nep);
error2s[n]=ep2*180.0/(pi*nep);
n=n+1;
ep1s=0.0;
ep2s=0.0;
}

/*----------------------------------------------------------------------------------------------------------*/
             /* CALCULATE ACTUAL POSITION / VELOCITY OF LINKS*/
/*----------------------------------------------------------------------------------------------------------*/
acc2=((torque1-c1)*con21-(torque2-c2)*con11)/(con12*con21-con11*con22);
```

```
acc1=((torque1-c1)*con22-(torque2-c2)*con12)/(con11*con22-con12*con21);
vel1=acc1*tsp+vel1;
vel2=acc2*tsp+vel2;
pos1=pos1+vel1*tsp+0.5*acc1*tsp*tsp;
pos2=pos2+vel2*tsp+0.5*acc2*tsp*tsp;
k=k+1;
}
t=0.0;
pos1o=pos1;
pos2o=pos2;
seg=seg+1;
/*-----------------------------------------------------------------------------------------------------------*/
   /*CALCULATE CONSTANTS OF QUINTIC TRAJECTORY FOR THE SECOND
                                   SEGMENT*/
/*-----------------------------------------------------------------------------------------------------------*/
if(seg==2 || seg==4)
{
p1=izz1n+izz2n+m2n*(x2n*x2n+l1*l1)+m1n*x1n*x1n;
p2=2.0*m2n*l1*x2n;
p3=m2n*x2n*x2n+izz2n;
p4=v1;
p5=v2;
p6=g*(m1n*x1n+m2n*l1);
p7=m2n*x2n*g;
a0=thf1;
b0=thf2;
a3=-10.0*(thf1)/(tf*tf*tf);
a4=15.0*(thf1)/(tf*tf*tf*tf);
a5=-6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=-10.0*(thf2)/(tf*tf*tf);
b4=15.0*(thf2)/(tf*tf*tf*tf);
b5=-6.0*(thf2)/(tf*tf*tf*tf*tf);
k=5;
}
if(seg==3)
{
t=0.0;
a0=0.0;
b0=0.0;
/*thf1=0.0;
thf2=0.0;  */
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
```

```
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
k=5;
}
}

/*----------------------------------------------------------------------------------------------------*/
                                        /*OPEN FILES*/
/*----------------------------------------------------------------------------------------------------*/
er1=fopen("scerror1.dat","w+");
er2=fopen("scerror2.dat","w+");
/*----------------------------------------------------------------------------------------------------*/
                                        /*WRITE FILES*/
/*----------------------------------------------------------------------------------------------------*/
for(i=0;i<=(n-1);i++)
{
fprintf(er1,"%f\n",error1s[i]);
fprintf(er2,"%f\n",error2s[i]);
}
/*for(i=0;i<=12;i++)
{
for(j=0;j<=12;j++)
{
fprintf(er1,"%f\n",lookt[i][j]);
fprintf(er2,"%f\n",lookt2[i][j]);
}
} */
fcloseall();
END:;
/*----------------------------------------------------------------------------------------------------*/
}
```

A.5 ADAPTIVE FUZZY CONTROL CODE

```
% Declare and initialize the variables
clear error1s;
clear error2s;
ts=0.003;
tsp=ts/5.0;
tf=2.0;
thf1=45;
thf2=-45;
n=1;
i=0;
k=5;
seg=1;
t=0.0;
g=9.8;
a0=0;
b0=0;
vel1=0;
vel2=0;
l1=0.26;
izz2=0.09;
izz1=0.09;
m1=2.0;
m2=2.0;
v1=2.5;
v2=2.5;
x1=0.13;
x2=0.14;
izz1n=1.5;
izz2n=0.09;
m1n=3.0;
m2n=3.0;
x1n=0.15;
x2n=0.16;
ki1=6000;
ki2=6000;
ep1s=0;
ep2s=0;
thf1=(thf1*pi)/180.0;
thf2=(thf2*pi)/180.0;
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
```

```
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
pos1=0.0;
pos2=0.0;
nep=1080.0/(6.0*pi);       %Normalisation factor for position error*/
nev=1.0*10.8/(6.0*pi);     %Normalisation factor for error dot*/
dnc=3*255.0;                %Denormalisation factor for voltage*/
%------------------------------------------------------------------------------------------------------
                                   %ACTUAL PARAMETERS*/
%------------------------------------------------------------------------------------------------------
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
a0=0.0;
b0=0.0;
%------------------------------------------------------------------------------------------------------
                                        %SERVO LOOP*/
%------------------------------------------------------------------------------------------------------
while seg <= 4,

while t <= tf,

                  % DESIRED POSITION, VELOCITY, ACCELERATION*/
if k==5,
pos1d=a0+(a3*t*t*t)+(a4*t*t*t*t)+(a5*t*t*t*t*t);
pos2d=b0+(b3*t*t*t)+(b4*t*t*t*t)+(b5*t*t*t*t*t);
vel1d=(3.0*a3*t*t)+(4.0*a4*t*t*t)+(5.0*a5*t*t*t*t);
vel2d=(3.0*b3*t*t)+(4.0*b4*t*t*t)+(5.0*b5*t*t*t*t);
t=t+ts;
k=0;
end
ep1=(pos1d-pos1)*nep;  %Normalised errors*/
ev1=(vel1d-vel1)*nev;
ep1s=ep1s+(pos1d-pos1);
%------------------------------------------------------------------------------------------------------
                                           %LINK 2 */
ep2=(pos2d-pos2)*nep;
ev2=(vel2d-vel2)*nev;
ep2s=ep2s+(pos2d-pos2);
%------------------------------------------------------------------------------------------------------
if abs(ep1)<0.5 & abs(ev1)<0.5,
   ep1=2*ep1;
   ev1=2*ev1;
```

```
end
if abs(ep2)<0.5 & abs(ev2)<0.5,
    ep2=2*ep2;
    ev2=2*ev2;
end
%-----------------------------------------------------------------------------------------------------
u1=[ep1
    ev1];
u2=[ep2
    ev2];
count1=evalfis(u1,x1x);
count2=evalfis(u2,x1x);
%alpha1=evalfis(u1,x2x);
%alpha2=evalfis(u2,x2x);
%-----------------------------------------------------------------------------------------------------
if abs(ep1)<0.5 & abs(ev1)<0.5,
    alpha1=alpha1;%/2;
end
if abs(ep2)<0.5 & abs(ev2)<0.5,
    alpha2=alpha2;%/2;
end
%-----------------------------------------------------------------------------------------------------

count1=count1*dnc;%*alpha1;
count2=count2*dnc;%*alpha2;
%-----------------------------------------------------------------------------------------------------
if abs(ep1)<0.5 & abs(ev1)<0.5,
    ep1=ep1/2;
    ev1=ev1/2;
end
if abs(ep2)<0.5 & abs(ev2)<0.5,
    ep2=ep2/2;
    ev2=ev2/2;
end
%-----------------------------------------------------------------------------------------------------
                              %TORQUE VALUES FOR LINK 1 */
%-----------------------------------------------------------------------------------------------------
torque1=(((((20.0*count1)/255.0)-0.066*(vel1*80.0))*0.066)/2.32)*80.0+ki1*ep1s;
c1=      (p4-p2*sin(pos2)*vel2)*vel1-
0.5*p2*sin(pos2)*vel2*vel2+p6*cos(pos1)+(p7*cos(pos1+pos2));
con11= (p1+p2*cos(pos2)+0.1498);
con12= (p3+0.5*p2*cos(pos2));
%-----------------------------------------------------------------------------------------------------
                    %CALCULATE ESTIMATED TORQUE FOR LINK 2*/
%-----------------------------------------------------------------------------------------------------
torque2=(((((20.0*count2)/255.0)-0.066*(vel2*70.0))*0.066)/2.32)*70.0+ki2*ep2s;
```

```
c2= 0.5*p2*sin(pos2)*vel1*vel1+p5*vel2+p7*cos(pos1+pos2);
con21= (0.5*p2*cos(pos2)+p3);
con22= (p3+0.1498);
%----------------------------------------------------------------------------------------------------
                                    %STORE VALUES*/
%----------------------------------------------------------------------------------------------------
if k==4,
error1s(n)=ep1*180.0/(pi*nep);
error2s(n)=ep2*180.0/(pi*nep);
n=n+1;
ep1s=0.0;
ep2s=0.0;
end
%----------------------------------------------------------------------------------------------------
            % CALCULATE ACTUAL POSITION / VELOCITY OF LINKS*/
%----------------------------------------------------------------------------------------------------
acc2=((torque1-c1)*con21-(torque2-c2)*con11)/(con12*con21-con11*con22);
acc1=((torque1-c1)*con22-(torque2-c2)*con12)/(con11*con22-con12*con21);
vel1=acc1*tsp+vel1;
vel2=acc2*tsp+vel2;
pos1=pos1+vel1*tsp+0.5*acc1*tsp*tsp;
pos2=pos2+vel2*tsp+0.5*acc2*tsp*tsp;
k=k+1;
end
t=0.0;
pos1o=pos1;
pos2o=pos2;
seg=seg+1;
%----------------------------------------------------------------------------------------------------
   %CALCULATE CONSTANTS OF QUINTIC TRAJECTORY FOR THE SECOND
                                      SEGMENT*/
%----------------------------------------------------------------------------------------------------
if seg==2 | seg==4,

p1=izz1n+izz2n+m2n*(x2n*x2n+l1*l1)+m1n*x1n*x1n;
p2=2.0*m2n*l1*x2n;
p3=m2n*x2n*x2n+izz2n;
p4=v1;
p5=v2;
p6=g*(m1n*x1n+m2n*l1);
p7=m2n*x2n*g;
a0=thf1;
b0=thf2;
a3=-10.0*(thf1)/(tf*tf*tf);
a4=15.0*(thf1)/(tf*tf*tf*tf);
a5=-6.0*(thf1)/(tf*tf*tf*tf*tf);
```

```
b3=-10.0*(thf2)/(tf*tf*tf);
b4=15.0*(thf2)/(tf*tf*tf*tf);
b5=-6.0*(thf2)/(tf*tf*tf*tf*tf);
k=5;
end
if seg==3,

   t=0.0;
a0=0.0;
b0=0.0;
%thf1=0.0;
%thf2=0.0;
a3=10.0*(thf1)/(tf*tf*tf);
a4=-15.0*(thf1)/(tf*tf*tf*tf);
a5=6.0*(thf1)/(tf*tf*tf*tf*tf);
b3=10.0*(thf2)/(tf*tf*tf);
b4=-15.0*(thf2)/(tf*tf*tf*tf);
b5=6.0*(thf2)/(tf*tf*tf*tf*tf);
p1=izz1+izz2+m2*(x2*x2+l1*l1)+m1*x1*x1;
p2=2.0*m2*l1*x2;
p3=m2*x2*x2+izz2;
p4=v1;
p5=v2;
p6=g*(m1*x1+m2*l1);
p7=m2*x2*g;
k=5;
end
end
%---------------------------------------------------------------------------------------------------
                                                  %OPEN FILES*/
%---------------------------------------------------------------------------------------------------
er1=fopen('f:\back\output\scerror1.dat','w');
er2=fopen('f:\back\output\scerror2.dat','w');

%---------------------------------------------------------------------------------------------------
                                                  %WRITE FILES*/
%---------------------------------------------------------------------------------------------------
error1s=error1s';
error2s=error2s';
%i=1;
%while i<n,
fprintf(er1,'%2.5f \n',error1s);
fprintf(er2,'%2.5f \n',error2s);
%i=i+1;
%end
fclose(er1);
fclose(er2);

%-----------------------------------------------------------------------------------------------------
```

Brief Biography of the Supervisor

Surekha Bhanot is currently attached to the Instrumentation Unit of Birla Institute of Technology and Sciences (BITS), Pilani as Professor and Unit Chief. She holds a Ph.D degree from University of Roorkee (now IIT, Roorkee). Previosly, she completed Master of Philosophy (M.Phil) in Instrumentation and Bachelor of Engineering (B.E) in Mechanical from BITS. She has a teaching experience of over 26 years, of which 19 years were at Thapar Institute of Engineering Tectnology (TIET), Patiala and rest at BITS, Pilani. Her current research interests include Instrumentation and AI techniques for process modeling and control.

Brief Biography of the Candidate

Sudeept Mohan completed his Master of Engineering (M.E) degree in Electronics and Control from the Birla Institute of Technology and Sciences (BITS), Pilani. He also holds a Masters (MSc.) degree in Physics and Bachelor of Engineering (B.E) in Electrical and Electronics from the same institute. He has a teaching experience of over fifteen years at BITS, Pilani. Currently he is attached to the Computer Science department at BITS as Assistant Professor. His research interests include Automatic controls and Robotics.