

GENETIC FUZZY CONTROL OF PUMA 560 ROBOT MANIPULATOR

A DISSERTATION

*Submitted in partial fulfillment of the
requirements for the award of the degree*

of

MASTER OF TECHNOLOGY

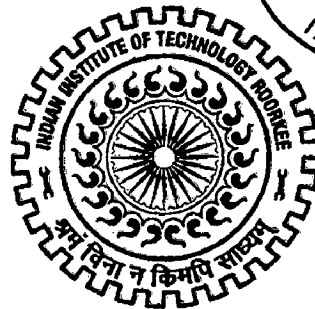
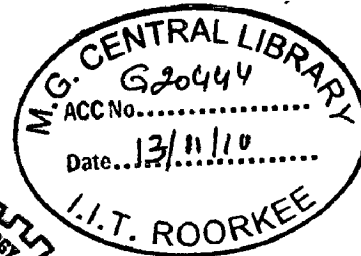
in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Control and Guidance)

By

MONA SUBRAMANIAM A



**DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)**

JUNE, 2010

CANDIDATE'S DECLARATION

I hereby declare that the work, which is presented in this dissertation report entitled, "**Genetic Fuzzy control of PUMA 560 Robot Manipulator**" towards the partial fulfillment of the requirements for the award of the degree of **Master of Technology** with specialization in **Control and Guidance**, submitted in the Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee, Roorkee (India) is an authentic record of my own work carried out during the period from July 2009 to June 2010, under the guidance of **Dr. M. J. Nigam, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology Roorkee.**

I have not submitted the matter embodied in this dissertation for the award of any other Degree or Diploma.

Date: 28/06/2010

Place: Roorkee



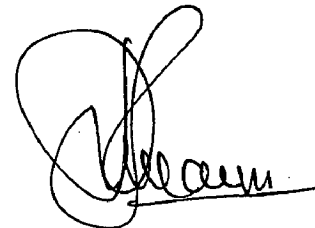
MONA SUBRAMANIAM A

CERTIFICATE

This is to certify that the above statement made by the candidate is correct to the best of my knowledge and belief.

Date: 28.06.2010

Place: Roorkee



Dr. M. J. NIGAM,
Professor, E&C Department,
IIT Roorkee,
Roorkee -247 667 (India).

ACKNOWLEDGEMENTS

I would like to extend gratitude and indebtedness to my guide and supervisor, **Dr. M. J. NIGAM** for his guidance, attention and constant encouragement that inspired me throughout my dissertation work.

I would also like to thank the Lab staff of Control Systems Lab, Department of Electronics and Communication Engineering, IIT Roorkee for providing necessary facilities.

I gratefully acknowledge my sincere thanks to my family members for their inspirational impetus and moral support during course of this work.

I am greatly indebted to all my friends, who have graciously applied themselves to the task of helping me with ample morale support and valuable suggestions. Finally, I would like to extend my gratitude to all those persons who directly or indirectly contributed towards this work.



MONA SUBRAMANIAM A

Abstract

Designing of control systems for a robot manipulator has many practical and theoretical challenges due to the complexities of the robot dynamics involved while achieving high precision- high velocity trajectory tracking in varying load conditions. Conventional robot control methods require highly accurate mathematical modeling, analysis and synthesis. Fuzzy Logic Controllers are a class of non-linear controllers that make use of human expert knowledge. Genetic algorithm is a class of evolutionary algorithms that can be applied to any problem which can be formulated as function optimization problem and it provides a way of optimizing fuzzy controller design.

PUMA 560 is a six Degree of Freedom robot arm that has to be controlled. A Fuzzy PD+I controller is used for the control of this robot arm. In this dissertation Genetic Algorithm is used to optimize, approximate and minimize the Fuzzy Logic Controllers. An algorithm is proposed which has a faster convergence when compared to Genetic Algorithm. Simulations and computations are carried out using MATLAB version 7.0.1.24704(R14).

Tuning of fuzzy parameters using genetic algorithm is carried out on reference Fuzzy PD+I controller. The parameters tuned are Rules, Rule-weights and Membership functions. As a part of the initial study, the parameters mentioned are tuned independently and it is seen that there is improvement in the performance. With this preliminary study, progressive tuning of parameters in three stages is carried out. At the end of second stage, tuning the rules in the rule base and then assigning weights to individual rule antecedent, considerable effect on the performance of the system is observed. Proceeding further with the third stage, membership function tuning is performed. There is an improvement in the performance from the second stage. In this study it is also observed that progressive tuning performs better than simultaneous tuning of parameters.

One of the objectives in machine learning is to learn any system from data. Approximation of system from input-output data presents a way to learn a system. With the input and output data points collected from the reference Fuzzy PD controllers of individual joints, bi-variate polynomial approximation function and Weighted-rule

Fuzzy approximation function is generated through interpolation process using genetic algorithm. The results of above two approximation functions are compared. The results obtained with that of a weighted Fuzzy approximation function is found to be superior with respect to the other.

For any given input trajectory, all the rules in the rulebase are not fired and those fired may not be the best. This suggests that the rulebase can be minimized with only the best rule entries. The rule base so obtained is an optimally minimum rulebase which is achieved by tuning the rulebase in such a way that the number of rules is minimized and the rules in them are optimized simultaneously. A small disturbance is given to the robot arm and the trajectory tracking is evaluated.

One of the main drawbacks of Genetic algorithm is its slow convergence rate. A simple stochastic optimization algorithm is proposed which has a good convergence rate. This algorithm is used for tuning the gains of the Fuzzy PD+I controller. The convergence is compared with that of Genetic algorithm and results of the proposed algorithm are quite encouraging. All the results are presented in Chapter 6.

Contents

| | |
|---|------------|
| Candidate's Declaration and Certificate | i |
| Acknowledgements | ii |
| Abstract | iii |
| Contents | v |
| List of Figures | vii |
| List of Tables | x |
| | |
| Chapter 1. Introduction | 01 |
| 1.1 Problem statement | 03 |
| 1.2 Literature review | 03 |
| 1.3 Motivation | 04 |
| 1.4 Organization of the Thesis | 05 |
| Chapter 2. Robot Manipulator control | 07 |
| 2.1 Robot manipulator | 07 |
| 2.2 Robot manipulator control problem | 09 |
| 2.3 Dynamic Model | 11 |
| 2.4 Manipulator control schemes | 13 |
| 2.5 PUMA 560 | 15 |
| Chapter 3. Fuzzy Logic Systems | 16 |
| 3.1 Fuzzy logic | 16 |
| 3.2 Fuzzy controllers | 18 |
| 3.3 Fuzzy PD+I controller | 20 |
| 3.4 Weighted-rule fuzzy control systems | 23 |
| Chapter 4. Genetic algorithm and Genetic Fuzzy Systems | 25 |
| 4.1 Genetic Algorithm | 25 |
| 4.2 Genetic Fuzzy Systems | 29 |
| Chapter 5. Implementations | 34 |
| 5.1 Parameter tuning | 34 |
| 5.1.1 Preliminary tuning | 34 |
| 5.1.1.1 Rule tuning | 35 |

| | |
|--|-----------|
| 5.1.1.2 Rule-Weight tuning | 35 |
| 5.1.1.3 Membership function tuning | 35 |
| 5.1.2 Two stage tuning | 36 |
| 5.1.3 Three stage tuning | 37 |
| 5.1.4 Simultaneous tuning | 37 |
| 5.2 Approximations of a system | 38 |
| 5.2.1 Bi-Variate Polynomial Approximation | 39 |
| 5.2.2 Weighted-Rule Fuzzy Approximation | 41 |
| 5.3 Optimally Minimum Rulebase | 42 |
| 5.4 A Novel Stochastic Algorithm for optimization | 43 |
| Chapter 6. Simulation results and discussions | 47 |
| 6.1 Results for parameter tuning | 47 |
| 6.1.1 Preliminary tuning | 48 |
| 6.1.1.1 Base system | 48 |
| 6.1.1.2 Rule tuning | 51 |
| 6.1.1.3 Rule weight tuning | 53 |
| 6.1.1.4 Membership function tuning | 55 |
| 6.1.2 Two stage tuning | 58 |
| 6.1.3 Three stage tuning | 61 |
| 6.1.4 Simultaneous tuning | 63 |
| 6.2 Results for Approximations of systems | 68 |
| 6.2.1 Bivariate polynomial approximation | 68 |
| 6.2.2 Weighted-rule Fuzzy approximation | 70 |
| 6.3. Results for Optimally minimum rulebase | 76 |
| 6.4 Results for proposed Stochastic Algorithm | 81 |
| Chapter 7. Conclusions and Future Scope | 86 |
| References | 87 |
| Research and publications by the author | 91 |
| Appendix A.1: PUMA560 Dynamic parameters | 92 |
| Appendix A.2: Default Genetic Algorithm settings | 95 |
| Appendix A.3: Generated Bivariate Polynomial coefficients | 96 |
| Appendix A.4 Pseudo-random joint angle generation | 99 |

List of Figures

| | |
|---|----|
| Figure 2.1 Classification of robots | 08 |
| Figure 2.2 Input- output representation of a robot | 11 |
| Figure 2.3 General structure of Joint space control | 13 |
| Figure 2.4 General structure of operational space control | 14 |
| Figure 2.5 Structure of PUMA 560 | 15 |
| Figure 3.1 Fuzzy controller architecture | 19 |
| Figure 3.2 Structure of the Fuzzy PD+I controller | 21 |
| Figure 3.3 Block diagram of the Fuzzy PD+I controlled PUMA560 robot model | 23 |
| Figure 4.1 Pseudo-code for structure of a Basic GA | 29 |
| Figure 4.2 Genetic Fuzzy System | 31 |
| Figure 4.3 Variation in the mean of a Gaussian MF | 32 |
| Figure 4.4 Variation in the spread of a Gaussian MF | 33 |
| Figure 5.1 Structure of the approximate of Fuzzy PD+I controller | 39 |
| Figure 5.2 Symmetric rulebase used in the Weighted-Rule Fuzzy Approximation | 41 |
| Figure 5.3 Flow chart of the proposed stochastic optimization algorithm | 46 |
| Figure 6.1 Structure of the Fuzzy PD+I controller used in base system | 48 |
| Figure 6.2 (a) The surface view, (b) membership function and (c) rulebase of the reference Fuzzy PD controller used | 49 |
| Figure 6.3 Input trajectory signal given to the individual joints of Puma 560 | 50 |
| Figure 6.4 Joint error generated by the given input trajectory for reference Fuzzy PD+I controller | 50 |
| Figure 6.5 Rule base after Rule tuning using Genetic Algorithm | 51 |
| Figure 6.6 Control surfaces of the rulebase after rule tuning | 52 |
| Figure 6.7 Joint error generated by the robot arm with Rule-tuned Fuzzy PD+I controller | 52 |
| Figure 6.8 Weight tuned rulebase, with weights written within brackets | 53 |
| Figure 6.9 Control surface of weight tuned rulebase | 54 |
| Figure 6.10 Joint error generated by the robot arm with Rule-weight tuned Fuzzy PD+I controller | 54 |
| Figure 6.11 Membership functions of the Fuzzy PD+I controller after MF tuning | 55 |

| | |
|--|----|
| Figure 6.12 control surfaces of the Fuzzy PD+I controller after MF tuning | 56 |
| Figure 6.13 Joint error generated by the robot arm with Membership functions tuned Fuzzy PD+I controller | 56 |
| Figure-6.14 Joint error generated by the robot arm after stage II tuning | 58 |
| Figure 6.15 Weighted-rule rulebase after Stage II tuning | 59 |
| Figure 6.16 Weighted-rule control surface after Stage II tuning | 60 |
| Figure-6.17 Membership functions(MF) of the Fuzzy PD+I controller after Stage III | 61 |
| Figure-6.18 Control surface of the Fuzzy PD+I controller after Stage III | 62 |
| Figure-6.19 Joint error plot of the Fuzzy PD+I controller after Stage III | 62 |
| Figure 6.20 Joint error of the Fuzzy PD+I controller after Simultaneous tuning | 63 |
| Figure 6.21 Weighted-rule rulebase after simultaneous tuning | 64 |
| Figure 6.22 Control surfaces of the Fuzzy PD+I controller after simultaneous tuning | 65 |
| Figure 6.23 Membership functions (MF) after simultaneous tuning | 66 |
| Figure 6.24 Surface plots of the bi-variate polynomials generated | 69 |
| Figure 6.25 (a) Joint error by the bivariate polynomial approximated controller and (b) joint error of the reference Fuzzy PD+I controller | 70 |
| Figure 6.26 Membership functions of the weighted-rule fuzzy approximate system | 71 |
| Figure 6.27 Rulebase of the weighted-rule fuzzy approximate system | 72 |
| Figure 6.28 Control surface of the weighted-rule fuzzy approximate system | 73 |
| Figure 6.29 (a) Joint error by the weighted-rule fuzzy approximate system and (b) joint error of the reference Fuzzy PD+I controller. | 74 |
| Figure 6.30 Input joint trajectories for genetic algorithm based optimally gain tuning of reference system used for optimally minimum rulebase study | 76 |
| Figure 6.31 Joint errors of the reference system (for optimally minimum rulebase study) after genetic algorithm gain tuning | 77 |
| Figure 6.32 Optimally minimum rulebase created by genetic algorithm | 78 |
| Figure 6.33 Control surface of optimally minimum rulebase | 79 |
| Figure 6.34 Joint errors for optimally minimum rulebase | 79 |
| Figure 6.35 Plot of the disturbance signal given to the robot arm at 1 sec | 80 |

| | |
|--|----|
| Figure 6.36 Joint errors for optimally minimum rulebase with disturbance signal given at 1 sec | 80 |
| Figure 6.37 Input and output joint trajectory for optimally minimum rulebase | 81 |
| Figure 6.38 Plot of the Rastrigin's Function | 82 |
| Figure 6.39 Comparative plot for objective function value of Rastrigin's function by Genetic algorithm and proposed algorithm | 83 |
| Figure 6.40 Pseudo-random input joint angle trajectories | 83 |
| Figure 6.41 Convergence plot of Genetic algorithm and the proposed algorithm | 84 |
| Figure 6.42 Joint errors plot for gain tuning by Genetic algorithm | 84 |
| Figure 6.43 Joint errors plot for gain tuning by proposed algorithm | 85 |

List of Tables

| | |
|---|----|
| Table 6.1 Tabulated results of the preliminary tuning | 57 |
| Table 6.2 Tabulated results of the stage wise and simultaneous tuning | 67 |
| Table 6.3 Tabulated results of the joint errors of polynomial approximation and weighted-rule fuzzy approximation methods | 75 |
| Table 6.4 Tabulated results of the approximation errors of polynomial approximation and weighted-rule fuzzy approximation methods | 75 |
| Table 6.5 Results for Rastrigin's function by Genetic algorithm and proposed algorithm | 82 |

Chapter 1: Introduction

Designing of control systems for a robot manipulator has many practical and theoretical challenges due to the complexities of the robot dynamics involved while achieving high precision- high velocity trajectory tracking in varying load conditions. Conventional robot control methods require highly accurate mathematical modeling, analysis and synthesis. These methods are not suitable for controlling robots in unstructured environment, which is a major challenge. There are always uncertainties present in the dynamics, unpredictability in environmental characteristics and also due to sensor impressions.

Fuzzy set techniques are a powerful tool for solving demanding real world problems with uncertain and unpredictable environment. Fuzzy Logic Controllers are a class of non-linear controllers that make use of human expert knowledge and an implicit imprecision to apply control to such systems. The construction of these controllers can be quick and effective in the presence of expert knowledge; conversely, in the absence of such knowledge, their design can be slow and based on trial-and-error rather than a guided approach.

When the traditional PID controller has to be replaced by a fuzzy equivalent, generally Fuzzy PD+I controllers are used as it has the following advantages of being Simple, having Less overshoot, Removes steady state error, and smoothens control signal. A three-input and one-output fuzzy system is too complex to construct the PID controller. It is very difficult to decide the fuzzy control rules intuitively. Fuzzy PD + I control system uses two-input fuzzy system for the proportional and derivative gain and linear control for the integral gain. Two-input fuzzy controller uses "error" and "change in error" as the input variables. In this structure, the proportional and derivative gains vary with the output of the system under control. The integral gain is kept constant. The proportional signal and the derivative signal are dominant to decide the transient response. But the integral signal whose major roll is to eliminate the steady state error has fixed gain. This is due to difficulty of designing the rules for the integral action. In fact, these methods do not require the knowledge of the dynamic model of the controlled system. This feature becomes of major importance when dealing with complex non-linear systems. The

dynamic modeling of robot arms shows a dependency on their mechanical parameters, subject to lifetime modifications (friction factors affected by the abuse of joints), and on their dynamical parameters that vary with the performed task (centers of gravity of the links affected by tool's replacements). These considerations also give advantage to fuzzy control methods on other non-linear method.

Initially when the fuzzy controller is designed by an expert, it is done heuristically based on his experience. Most of the times the controller designed will have certain amount of error present in it since different experts will have varying experience, thereby resulting in a non optimum performance by the controller. In order to get better results, tuning of the Fuzzy controller is required. Fuzzy controllers can be tuned by various strategies, like changing the scaling factor, Modifying the support and spread of membership functions, modifying the rules of the rulebase and changing the type of a membership function itself. In addition to these the Rule-weights can also be changed to perform a local tuning of linguistic rules, this enables the linguistic fuzzy models to cope with inefficient and/or redundant rules and thereby enhances the robustness, flexibility and system modeling capability. If a rule weight is applied to the consequent part of the rule, it modifies the size of the rule's output value. By assigning a rule weight to each of the fuzzy rules, complexity is increased while its accuracy is improved. This suggests a tradeoff relation between the accuracy and complexity.

Genetic algorithm is a class of evolutionary algorithms; they can be applied to any problems that can be formulated as function optimization problems. Genetic Algorithms provide a way of overcoming the shortcoming with fuzzy controller design. These algorithms use some of the concepts of evolutionary theory, and provide an effective way of searching a large and complex solution space to give close to optimal solutions in much faster times than random trial-and-error.

A PUMA 560 robot arm is the system that needs to be controlled. It is a 6 Degree of Freedom robot. A Fuzzy PD+I controller is used for control of the robot arm. In this report Genetic Algorithm is used to optimize, minimize and approximate Fuzzy Logic Controllers. An algorithm is proposed in the end that will have a faster convergence when compared to Genetic Algorithm.

1.1 Problem statement

In light of the discussion, the prime objectives of the research work will focus on using genetic algorithm in combination with fuzzy controllers for control of PUMA 560 robot arm. The objectives of the thesis can be summarized as follows:

- 1 – Study the various Tuning strategies of weighted fuzzy system by genetic algorithm and provide a comparison among them. The various tuning strategies are listed below
 1. Fuzzy rules tuning
 2. Fuzzy weights tuning
 3. Fuzzy membership functions tuning
 4. Two stage tuning (rules followed by weights)
 5. Three stage tuning (rules followed by weights and then membership function)
 6. Simultaneous tuning of rules, weights and membership functions.
- 2 – Study Approximation of the controller by
 1. Bivariate polynomial approximation
 2. Weighted fuzzy system approximationAnd provide a comparative study of both the cases.
- 3 – Generate an optimally minimum rule fuzzy control system using genetic algorithm for the PUMA560 arm
- 4 – Develop a novel stochastic algorithm for optimization which has a faster convergence compared to genetic algorithm.

1.2 Literature review

One of the main contenders of the control field which has benefited from the study of the computational intelligence has been robotic control. The main reason for this is that non linear and coupled complexities are present in the dynamics of robots [1]. And computational intelligence provides an efficient way to combat with this. Study of hybridization of the techniques in computational intelligence is done in [2]. The Fuzzy PD+I controllers are the most general use controller as it has the following advantages of being Simple , having Less overshoot, Removes steady state error, smoothens control

signal [3]. The most popular technique in evolutionary computation research has been the genetic algorithm. Evolutionary algorithms can be applied to any problems that can be formulated as function optimization problems [4]. The design of fuzzy PD+I control was discussed *bong et all* [5]. The study of fuzzy PD+I controller for PUMA560 shows better results when compared with normal PID controller [6]. By tuning the gains of the Fuzzy PD+I controller using genetic algorithm, Simulated Annealing and Generalized Pattern Search Techniques better results are obtained [7]. Gain tuning of the Fuzzy PD+I controller using heuristic search is carried out in [8].

The various structures of Genetic Fuzzy systems were discussed in [9] [10]. Fuzzy controllers can be tuned by various strategies, like changing the scaling factor, Modifying the support and spread of membership functions, modifying the rules of the rulebase and changing the type of a membership function itself. In addition to these the Rule-weights can also be changed to perform a local tuning of linguistic rules, this enables the linguistic fuzzy models to cope with inefficient and/or redundant rules and thereby enhances the robustness, flexibility and system modeling capability [11]. By assigning a rule weight to each of the fuzzy rules, complexity is increased while its accuracy is improved. This suggests a tradeoff relation between the accuracy and complexity [12]. If a rule weight is applied to the consequent part of the rule, it modifies the size of the rule's output value [13].

The interpolation and approximation theory are quite mature fields in by themselves which has received and continues receiving not deep but constant attention [14]. Various methods of polynomial interpolation give efficient results but are mathematically quite rigorous and difficult for a multi-variate case [15]. The methodology of optimally minimizing the rule base was examined in [16], where it was used it for control of the cart pole problem.

1.3 Motivation

Fascinated by the abilities of genetic algorithm, a stochastic optimization method, the various operations on fuzzy control design is carried out. Studying the tuning of individual parameters should give a fair idea about fuzzy systems. Performing the tuning

of the rules of fuzzy controller initially and then fine tuning its weight for fuzzy PD+I control of PUMA560 robot will suggest how the system performance varies. It is expected that the tuning of rules will produce considerable change in the control of the fuzzy controller, but changing the rule weight results will result in finer details getting adjusted.

In order to make a comparison between stage-wise tuning and simultaneous tuning, the parameters for weighted-rule fuzzy controller are tuned using genetic algorithm. Parameters like rules, membership functions and rule-weights play an important role in any fuzzy controller, and optimizing them is a necessary task, since these parameters are always built by designers with trial and error along with their experience or experiments. By stage-wise, it is meant that one set of parameters are taken and tuned at a time. And in the simultaneous tuning, all the parameters considered are tuned in one go. This study is done on Fuzzy PD+I controller of Puma 560 robot. After comparison an inference is drawn as to which procedure is better than the other with reference to ISE criterion.

Genetic algorithms can solve any problem that can be formulated as an optimization problem; with this in mind it is employed to determine the coefficients of polynomial that approximates a system without getting into rigorous mathematical analysis. It is also employed for creating a Weighted-Rule fuzzy approximate model and a comparison of both will throw some light on their individual abilities.

Rule base optimization and minimization task an important facet in the study of genetic fuzzy systems. This method creates only the best possible rules in the rule base and eliminates the unnecessary ones, with this approach a lot of computation and memory resources are conserved.

Proposing an algorithm for optimization and evaluating it can be a worthwhile undertaking. If the algorithm performs better than any standard one then it suggests that there is scope for it to be improved still further.

1.4 Organization of the Thesis

The report has been organized into seven chapters. Chapter 1 gives an introduction to this thesis work, Problem statement, literature survey and Motivation and organization of the thesis. Chapter 2 briefly discusses the Robot Manipulator control. Chapter 3 contains

the Introduction to Fuzzy Logic Systems. Chapter 4 briefly talks of Genetic algorithm and Genetic Fuzzy Systems. Chapter 5 describes Implementation of the topics discussed in the problem statement Chapter 6 presents the Simulation results and discussions. Chapter 7 gives Conclusions and suggestions for future work.

Chapter 2: Robot Manipulator control

The word robot has its origins from Czech where *robota* means executive labour. It was Karel Capek, a science fiction writer who introduced the word 'robot' in his play "*Rossum's Universal Robots*". Ever since, the concept has transformed from the idea of an artificial superhuman into the reality of animated autonomous machines. Robots today are making a considerable impact on many aspects of modern life, from industrial manufacturing to healthcare, transportation, and exploration of the deep space and sea. The study of robotics is inter-disciplinary science covering domains of mechanical, electrical, electronics and computer science. Robot manipulators are basically multi-degree of freedom positioning devices. The robot, as the plant to be controlled, is a multi-input/multi-output, highly coupled, nonlinear dynamic system. Robot control is the backbone of robotics. The tasks of robot control is to find the force(torque) or the actuator input vectors which results in the desired motion tracking with required accuracy by the robot end-effectors [17].

2.1 Robot manipulator

Robotics is concerned with the study of those machines that can replace human beings in the execution of a task, with respect to both physical activity and decision making [1]. Robots can be classified into two, those with a fixed base known as robot manipulators, and the other with a mobile base called mobile robots. The Figure 2.1 gives the classification of robots.

Both mobile robots and manipulators play an important role in the field of robotics. This thesis is exclusively devoted to robot manipulators. Robot manipulators are designed to perform a wide variety of tasks in automotive industry which are used primarily in material handling, welding, assembly, spray painting, grinding and other manufacturing applications. In industrial application, robot manipulators are commonly employed in repetitive tasks of precision and which may be hazardous for human beings. Manipulators are used in industries primarily because they reduce the production cost, enhance precision, quality and productivity with an added advantage of having greater flexibility over

specialized machines. In addition to this there are applications which can be done only by robot manipulators, these applications involve tasks in hazardous conditions such as in radioactive, toxic zones or where a risk of explosion exists, as well as deep space and submarine applications.

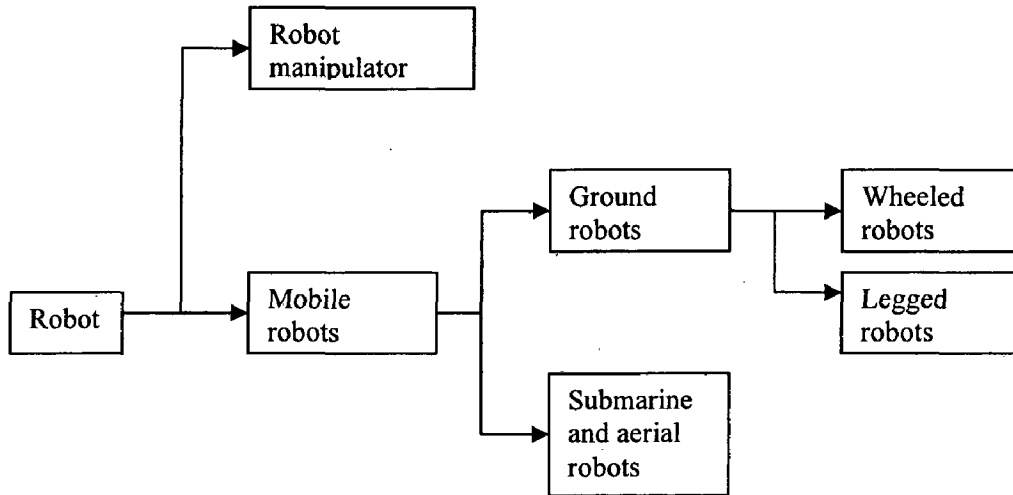


Figure 2.1 Classification of robots

The definition of an industrial robot according to the International Federation of Robotics is as follows:

'A manipulating industrial robot is an automatically controlled, programmable multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications'

The mechanical structure of a robot manipulator consists of a sequence of rigid bodies known as links interconnected by means of articulations called joints; a manipulator is characterized by an arm that ensures mobility, a wrist that confers dexterity, and an end-effector that performs the task required of the robot. The fundamental structure of a manipulator is the serial or open kinematic chain. From a topological viewpoint, a kinematic chain is termed open when there is only one sequence of links connecting the two ends of the chain. Alternatively, a manipulator contains a closed kinematic chain when a sequence of links forms a loop. In an open kinematic chain, each prismatic or revolute joint provides the structure with a single degree of freedom (DOF). A prismatic joint creates a relative translational motion between the two links, whereas a revolute joint creates a relative rotational motion between the two links. Revolute joints are usually

preferred to prismatic joints in view of their compactness and reliability. On the other hand, in a closed kinematic chain, the number of DOFs is less than the number of joints in view of the constraints imposed by the loop. In this work we consider robot manipulators formed by an open *kinematic* chain.

The degrees of freedom should be properly distributed along the mechanical structure in order to have a sufficient number to execute a given task. In the most general case of a task consisting of arbitrarily positioning and orienting an object in three-dimensional (3D) space, six DOFs are required, the first three joints determine the position of the end of the last link in the Cartesian space and the last three specify its orientation with respect to a reference coordinate frame. If more DOFs than task variables are available, the manipulator is said to be redundant from a kinematic viewpoint and are of prime importance when obstacle avoidance is concerned.

2.2 Robot manipulator control problem

Manipulator control has been the subject of many years of research, and continues to attract much attention. The main challenges in the manipulator control problem are the complexity of the dynamics, and uncertainties, both parametric and dynamic. Parametric uncertainties arise from imprecise knowledge of the dynamics, while dynamic uncertainties arise from joint and link, actuator dynamics, friction, sensor noise and unknown environment dynamics.

The reasons posed by these systems which add to the challenges are:

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

Control of robot manipulators is the problem of determining the time history of joint inputs required to cause the end effector to execute a commanded motion. For the analytical purposes, considering an n-DOF robot manipulator, the joint positions are collected in the vector q ,

$$q = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_n \end{bmatrix} \quad (2.1)$$

Physically, the joint positions q is measured by sensors conveniently located on the robot. The corresponding joint velocities may also be measured or estimated from joint position evolution. To each joint, an actuator which may be electromechanical, pneumatic or hydraulic is in contact. The actuators generate the forces or torques which produce the movement of the links and in turn the movement of the robot as a whole. For analytical purposes these torques and forces are collected in the vector τ ,

$$\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_n \end{bmatrix} \quad (2.2)$$

For robots moving freely in their workspace, i.e. without interacting with their environment, the output y to be controlled, may correspond to the joint positions q and joint velocities \dot{q} or alternatively, to the position and orientation of the end-effector (also called end-tool). For robots that have physical contact with their environment, the output y may include the torque τ and force f exerted by the end-tool over its environment. Hence, the corresponding output y of a robot system – involved in a specific class of tasks in general, be of the form,

$$y = y(q, \dot{q}, f) \quad (2.3)$$

On the other hand, the input variables that may be modified to affect the evolution of the output are basically the torques τ and forces f applied by the actuators over the robots joint. Figure 2.2 shows the block diagram corresponding to the case when the outputs are

the joint positions and velocities while τ is the input. In this case, robot with n joints has $2n$ outputs and n inputs.

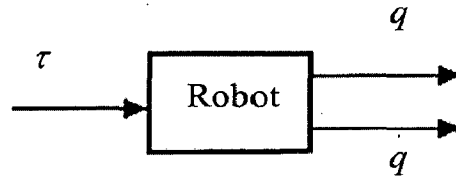


Figure 2.2 Input- output representation of a robot

2.3 Dynamic Model

Robot manipulators are articulated mechanical systems composed of links connected by joints [17]. The dynamic model of robot manipulators is typically derived in the analytic form, using the laws of mechanics due to the mechanical nature. The dynamic models of robot manipulators are highly nonlinear and non autonomous (depend on the state variables and time) differential equations.

Consider a robot manipulator of n -DOF composed of rigid links interconnected by frictionless joints. The kinetic energy function $K(q, \dot{q})$ associated with such an articulated mechanism may always be expressed as,

$$K(q, \dot{q}) = \frac{1}{2} \dot{q}^T M(q) \dot{q} \quad (2.4)$$

where $M(q)$ is a matrix of dimension $n \times n$ referred to as the mass inertia matrix. $M(q)$ is symmetric and positive definite for all $q \in \mathbb{R}^n$. The potential energy $U(q)$ does not have a specific form as in the case of the kinetic energy but it is known that it depends on the vector of joint positions q .

The *Lagrangian* $L(q, \dot{q})$ of a robot manipulator of n -DOF is the difference between its kinetic energy K and its potential energy U ,

$$L(q, \dot{q}) = \frac{1}{2} \dot{q}^T M(q) \dot{q} - U(q) \quad (2.5)$$

With this *Lagrangian*, the Lagrange's equations of motion is written as

$$\frac{d}{dt} \left[\frac{\partial}{\partial \dot{q}} \left[\frac{1}{2} \dot{q}^T M(q) \dot{q} \right] \right] - \frac{\partial}{\partial q} \left[\frac{1}{2} \dot{q}^T M(q) \dot{q} \right] + \frac{\partial U(q)}{\partial q} = \tau \quad (2.6)$$

On the other hand, it holds that

$$\frac{\partial}{\partial \dot{q}} \left[\frac{1}{2} \dot{q}^T M(q) \dot{q} \right] = M(q) \dot{q} \quad (2.7)$$

$$\frac{d}{dt} \left[\frac{\partial}{\partial \dot{q}} \left[\frac{1}{2} \dot{q}^T M(q) \dot{q} \right] \right] = M(q) \ddot{q} + \dot{M}(q) \dot{q} \quad (2.8)$$

Considering these expressions, the equation of motion takes the form

$$M(q) \ddot{q} + \dot{M}(q) \dot{q} - \frac{1}{2} \frac{\partial}{\partial q} \left[\dot{q}^T M(q) \dot{q} \right] + \frac{\partial U(q)}{\partial q} = \tau \quad (2.9)$$

or, in compact form,

$$M(q) \ddot{q} + C(q, \dot{q}) \dot{q} + g(q) = \tau \quad (2.10)$$

where

$$C(q, \dot{q}) \dot{q} = \dot{M}(q) \dot{q} - \frac{1}{2} \frac{\partial}{\partial q} \left[\dot{q}^T M(q) \dot{q} \right] \quad (2.11)$$

$$g(q) = \frac{\partial U(q)}{\partial q} \quad (2.12)$$

Equation (2.10) is the dynamic equation for robots of n -DOF. $C(q, \dot{q}) \dot{q}$ is a vector of dimension n called the vector of centrifugal and Coriolis forces, $g(q)$ is a vector of dimension n of gravitational forces or torques and τ is a vector of dimension n called the vector of external forces, which in general corresponds to the torques and forces applied by the actuators at the joints. Each element of $M(q)$, $C(q, \dot{q})$ and $g(q)$ is in general, a relatively complex expression of the positions and velocities of all the joints q and \dot{q} . The elements of $M(q)$, $C(q, \dot{q})$ and $g(q)$ depend of course, on the geometry of the robot in question. Note that computation of the vector $g(q)$ for a given robot may be carried out as its simply the gradient of the potential energy function $U(q)$.

2.4 Manipulator control schemes

Manipulator controllers can be classified into two broad categories, namely joint space control scheme and Cartesian space control scheme [1]. The joint space control problem is actually articulated in two sub-problems. First, manipulator inverse kinematics is solved to transform the motion requirements x_d from the operational space into the corresponding motion q_d in the joint space. Then, a joint space control scheme is designed that allows the actual motion q to track the reference inputs. However, this solution has the drawback that a joint space control scheme does not influence the operational space variables x_e which are controlled in an open-loop fashion through the manipulator mechanical structure. It is then clear that any uncertainty of the structure (construction tolerance, lack of calibration, gear backlash, elasticity) or any imprecision in the knowledge of the end-effector pose relative to an object to manipulate causes a loss of accuracy on the operational space variables. Figure 2.3 shows the general structure of Joint space control.

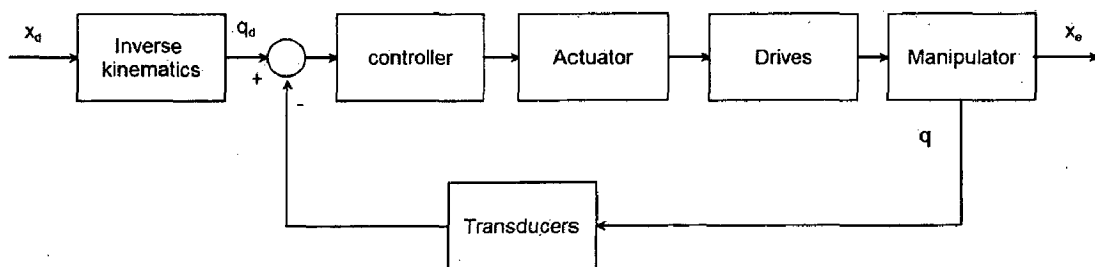


Figure 2.3 General structure of Joint space control

The operational space control problem follows a global approach that requires a greater algorithmic complexity; the General structure of operational space control is shown in Figure 2.4, inverse kinematics is now embedded into the feedback control loop. Its conceptual advantage regards the possibility of acting directly on operational space variables; this is somewhat only a potential advantage, since measurement of operational space variables is often performed not directly, but through the evaluation of direct kinematics functions starting from measured joint space variables. This work focuses on the joint space control scheme and it is assumed that the inverse kinematic is already performed and the trajectory is available at hand.

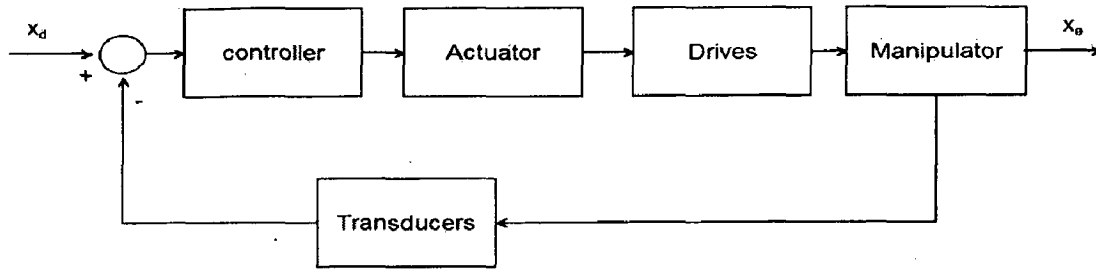


Figure 2.4 General structure of operational space control

There are two main control objectives of robot manipulator Position control (regulation) and motion control (trajectory tracking) [17]. The simplest way to specify the movement of a manipulator is the so called “point to Point” method. This methodology consists in determining a series of points in the manipulator workspace, which the end effector is required to pass through. The position control problem consists of making the end effector to reach a specified point regardless of the trajectory followed from its initial configuration. A more general way to specify a robot motion is by continuous trajectory. In this case, a (continuous) curve, or path in the state space and parameterized in time, is available to achieve a desired task. Then, the motion control problem consists of making the end-effector follow this trajectory as closely as possible. This control problem, whose study is our central objective, is also referred to as trajectory tracking control. The main interest of this work is the study of motion controllers and therefore, we assume that the problems of path planning and trajectory generation are previously solved.

The problem of motion control in joint space for robot manipulators may be formulated in the following terms.

Given a set of vectorial bounded functions q_d, \dot{q}_d and \ddot{q}_d referred to as desired joint positions, velocities and accelerations, find a vectorial function τ such that the positions q , associated to the robot’s joint coordinates follow q_d accurately.

In more formal terms, the objective of motion control consists of finding τ such that

$$\lim_{t \rightarrow \infty} q(t) = q_d(t) \quad (2.13)$$

Where $q_d \in R^n$ stands for the desired joint position vector, or in other words,

$$\lim_{t \rightarrow \infty} e(t) = 0 \quad (2.14)$$

Where $e \in R^n$ stands for the joint position errors vector called position error defined by,

$$e(t) = q_d(t) - q(t) \quad (2.15)$$

The control objective is achieved if the manipulator joint variables follow asymptotically the trajectory of the desired motion.

The computation of the vector τ involves, a vectorial nonlinear function of q_d, \dot{q}_d and \ddot{q}_d . This function is called the “control law” or simply, “controller”. In general, a motion control law may be expressed as

$$\tau = \tau(q, \dot{q}, \ddot{q}, q_d, \dot{q}_d, \ddot{q}_d, M(q), C(q, \dot{q}), g(q)) \quad (2.16)$$

2.5 PUMA 560

PUMA 560 is one of the most popular industrial robots. It is used in most robotics publications to illustrate various concepts, computational developments and research issues on robot manipulators. PUMA stands for “Programmable Universal Machine for Assembly”. It was created by Unimation. PUMA 560 is a six DOF robot manipulator with six revolute joints. Its structure bears close similarities with the human arm in other words it is *anthropomorphic*, articulated robot arm. It consists of a waist rotation, a shoulder rotation, an elbow rotation and a three-DOF wrist that allows arbitrary orientation of the gripper within its workspace. The Figure 2.5 shows the structure of PUMA560 [18]. The dynamic parameters are given in the Appendix A.1

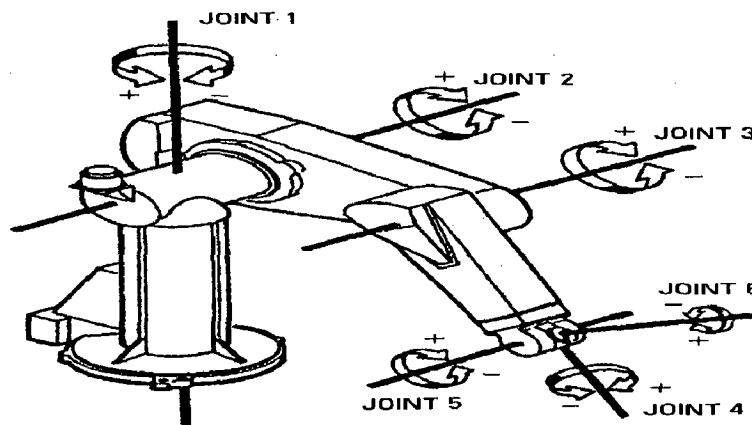


Figure 2.5 Structure of PUMA 560

Chapter 3: Fuzzy Logic Systems

Conventional control methods require highly accurate mathematical modeling, analysis and synthesis, described using one or more differential equations that define the system response to its inputs. This involves assumptions being made with respect to the system dynamics and any non-linear behavior that may occur. Fuzzy set techniques are a powerful tool for solving demanding real world problems with uncertain and unpredictable environment. Fuzzy Logic Controllers are a class of non-linear controllers that make use of human expert knowledge and an implicit imprecision to apply control to such systems. The construction of these controllers can be quick and effective in the presence of expert knowledge. Fuzzy controller are particularly useful in the case where the mathematical model of the control process may not exist, or may be too "expensive" in terms of computer processing power and memory, and a system based on empirical rules may be more effective.

3.1 Fuzzy logic

Fuzzy logic was first developed by Zadeh in the mid-1960s for representing uncertain and imprecise knowledge [2]. It provides an approximate but effective means of describing the behavior of systems that are too complex, ill-defined, or not easily analyzed mathematically. Fuzzy logic is a form of multi-valued logic derived from fuzzy set theory to deal with reasoning that is approximate rather than precise. Fuzzy logic is a superset of conventional (Boolean) logic that has been extended to handle the concept of partial truth—truth values between “completely true” and “completely false” [19]. In binary sets with *binary logic*, in contrast to fuzzy logic named also *crisp logic*, the variables may have a membership value of only 0 or 1. Just as in fuzzy set theory with fuzzy logic the set membership values can range (inclusively) between 0 and 1, in fuzzy logic the degree of truth of a statement can range between 0 and 1 and is not constrained to the two truth values {true (1), false (0)} as in classic predicate logic. Zadeh argues that the attempts to automate various types of activities from assembling hardware to medical diagnosis have been impeded by the gap between the way human beings reason and the way computers

are programmed [2]. Fuzzy logic uses graded statements rather than ones that are strictly true or false. It attempts to incorporate the “rule of thumb” approach generally used by human beings for decision making. Thus, fuzzy logic provides an approximate but effective way of describing the behavior of systems that are not easy to describe precisely. Fuzzy logic controllers, for example, are extensions of the common expert systems that use production rules like “if-then.” With fuzzy controllers, however, linguistic variables like “tall” and “very tall” might be incorporated in a traditional expert system. The result is that fuzzy logic can be used in controllers that are capable of making intelligent control decisions in sometimes volatile and rapidly changing problem environments.

Fuzzy logic has been applied to diverse fields, from control theory to artificial intelligence. Fuzzy logic addresses such applications perfectly as it resembles human decision making with an ability to generate precise solutions from certain or approximate information. It fills an important gap in engineering design methods left vacant by purely mathematical approaches (e.g. linear control design), and purely logic-based approaches (e.g. expert systems) in system design.

While other approaches require accurate equations to model real-world behaviors, fuzzy design can accommodate the ambiguities of real-world human language and logic. It provides both an intuitive method for describing systems in human terms and automates the conversion of those system specifications into effective models.

The various definitions of the terms used in a fuzzy system are given below.

- Fuzzy set - A set that can contain elements with only a partial degree of membership.
- Membership function (MF) - A function that specifies the degree to which a given input belongs to a set or is related to a concept. It represents the degree of truth as an extension of valuation.
- Degree of membership - The output of a membership function, this value is always limited to between 0 and 1. Also known as a membership value or membership grade.
- Linguistic variables - A *linguistic variable* is a collection of fuzzy sets representing linguistic terms of a concept. Variables in mathematics usually take numerical values, in fuzzy logic applications, the non-numeric linguistic variables are often

used to facilitate the expression of rules and facts. A linguistic variable such as age may have a value such as young or its antonym old.

- Antecedent - The initial (or "if") part of a fuzzy rule.
- Consequent - The final (or "then") part of a fuzzy rule.
- Fuzzification - The process of generating membership values for a fuzzy variable using membership functions. In other words, the process of converting a crisp input value to a fuzzy value.
- Defuzzification - The process of transforming a fuzzy output of a fuzzy inference system into a crisp output.
- Implication - The process of shaping the fuzzy set in the consequent based on the results of the antecedent in a Mamdani-type FIS.
- Aggregation - The combination of the consequents of each rule in a Mamdani fuzzy inference system in preparation for defuzzification.

3.2 Fuzzy controllers

The core of a fuzzy controller is a collection of verbal or linguistic rules of the if-then form [3]. Several variables may occur in each rule, both on the if -side and the then-side. Reflecting expert opinions, the rules can bring the reasoning used by computers closer to that of human beings.

The fuzzy controller has four main components: 1. The "rule-base" holds the knowledge, in the form of a set of rules, of how best to control the system. 2. The inference mechanism evaluates which control rules are relevant at the current time and then decides what the input to the plant should be. 3. The fuzzification interface simply modifies the inputs so that they can be interpreted and compared to the rules in the rule-base. 4. The defuzzification interface converts the conclusions reached by the inference mechanism into a crisp control action [20]. Figure 3.1 shows the Fuzzy controller architecture.

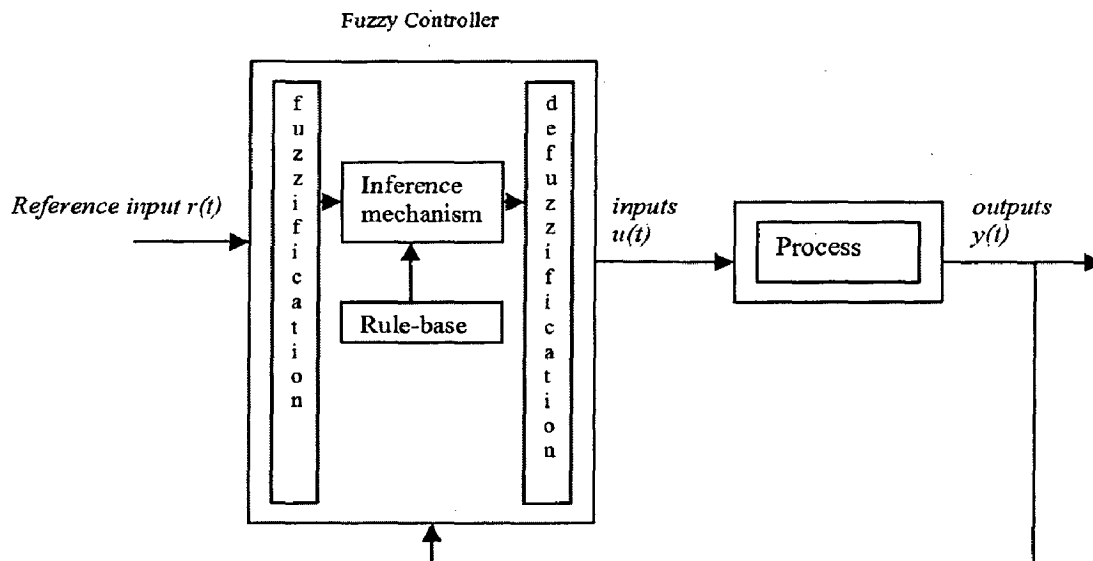


Figure 3.1 Fuzzy controller architecture

To design the fuzzy controller, the control engineer must gather information on how the artificial decision maker should act in the closed-loop system. Sometimes this information can come from a human decision maker who performs the control task, while at other times the control engineer can come to understand the plant dynamics and write down a set of rules about how to control the system without outside help. These “rules” basically say, “If the plant output and reference input are behaving in a certain manner, then the plant input should be some value.” A whole set of such “If-Then” rules is loaded into the rule-base, and an inference strategy is chosen, then the system is ready to be tested to see if the closed-loop specifications are met.

Two forms of FLC are,

- Mamdani
- Sugeno

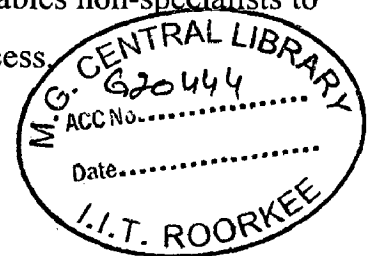
Both of these architectures are similar in all respects except for the formulation of the output crisp value. In the Mamdani FLC, the output is formulated using fuzzy sets whereas the Sugeno type FLC uses single -spike output MFs (i.e. singletons) rather than distributed functions. In this work a Mamdani type fuzzy system is used.

Even though fuzzy controllers are widely used there are a few reasons as to why one would not use a fuzzy controller:

- The PID controller is well understood, easy to implement – both in its digital and analog forms – and it is widely used. By contrast, the fuzzy controller requires some knowledge of fuzzy logic. It also involves building arbitrary membership functions.
- The fuzzy controller is generally nonlinear. It does not have a simple equation like the PID, and it is more difficult to analyze mathematically; approximations are required, and it follows that stability is more difficult to guarantee.
- The fuzzy controller has more tuning parameters than the PID controller. Furthermore, it is difficult to trace the data flow during execution, which makes error correction more difficult.

The main reasons for the success of fuzzy controller's usage in industry are

- Since the control strategy consists of if-then rules, it is easy for a plant operator to read. The rules can be built from a vocabulary containing everyday words such as 'high', 'low', and 'increasing'. Plant operators can embed their experience directly.
- The fuzzy controller accommodates many inputs and many outputs. Variables can be combined in an if-then rule with the connectives 'And' and 'Or'. Rules are executed in parallel, implying a recommended action from each. Fuzzy logic enables non-specialists to design control systems, and this is one of the key reasons for its success.



3.3 Fuzzy PD+I controller

In this structure, the fuzzy system is applied only to the proportional and derivative signal of the linear PID controller [5]. The integral signal uses conventional linear method. The major roll of the integral signal is to eliminate the steady state error. The transient response is affected mostly by the proportional signal and the derivative signal. For the enhancement of the transient response, the varying gains are implemented on the proportional and derivative parts using two-input fuzzy system. The nonlinearities that make the varying gains possible are added by the fuzzy control rules and the membership functions. The nonlinearities emphasize the proportional gain when the tracking error is relatively large and accelerates decreasing speed of the tracking error. The nonlinearities in the derivative gain suppress the overshoot and increases damping as the signals starts to

settle down. In this structure, the fuzzy system is normalized with respect to the maximum range of the signal. Figure 3.2 shows the structure of the Fuzzy PD+I controller

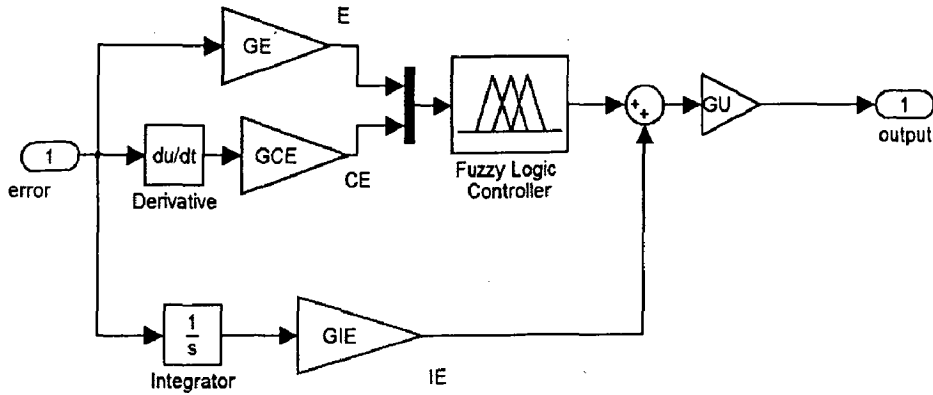


Figure 3.2 Structure of the Fuzzy PD+I controller

If the closed-loop system exhibits a sustained error in steady state, integral action is necessary. The integral action will increase (decrease) the control signal if there is a positive (negative) error, even for small magnitudes of the error. Thus, a controller with integral action will always return to the reference in steady state

The integral error $IE = GIE \int e(t)dt$ is proportional to the accumulation of all previous error measurements in discrete time, with [3].

$$\int e(t)dt \approx \sum_{j=1}^n e(j)T_s \quad (3.1)$$

the control signal $U(n)$ after the gain GU , at the time instant n , is a nonlinear function of error, change in error, and integral error,

$$U(n) = \left[f(GE * e(n), GCE * e(n)) + GIE \sum_{j=1}^n e(j)T_s \right] * GU \quad (3.2)$$

The function f is again the control surface of a PD rule base. The mapping is usually nonlinear, but with a favorable choice of design, a linear approximation is Equation (3.3)

$$f(GE * e(n), GCE * e(n)) \approx GE * e(n) + GCE * e(n) \quad (3.3)$$

Substituting this in Equation (3.2) yields the control action,

$$U(n) \approx \left[GE * e(n) + GCE * e(n) + GIE \sum_{j=1}^n e(j)T_s \right] * GU \quad (3.4)$$

$$U(n) = GE * GU * \left[e(n) + \frac{GCE}{GE} * e(n) + \frac{GIE}{GE} \sum_{j=1}^n e(j)T_s \right] \quad (3.5)$$

In the last line we have assumed that GE is non-zero.

Ideal continuous PID controller is given by the Equation (3.6)

$$u = K_p \left(e + \frac{1}{T_i} \int e(t)dt + T_d \frac{de}{dt} \right) \quad (3.6)$$

In digital controllers, the Equation (3.6) must be approximated. Replacing the derivative term by a backward difference and the integral by a sum using rectangular integration, and given a constant – preferably small – sampling time T_s , the simplest approximation is

$$u(n) = K_p \left(e(n) + \frac{1}{T_i} \sum_{j=1}^n e(j)T_s + T_d \frac{e(n) - e(n-1)}{T_s} \right) \quad (3.7)$$

Comparing Equations (3.5) and (3.7) the gains are related as follows:

$$GE * GU = K_p \quad (3.8)$$

$$\frac{GCE}{GE} = T_d \quad (3.9)$$

$$\frac{GIE}{GE} = \frac{1}{T_i} \quad (3.10)$$

The FPD+I controller provides all the benefits of PID control, but also the disadvantages regarding derivative kick. The integral error removes any steady state error, but can also cause integrator windup.

Figure 3.3 shows the block diagram of the Fuzzy PD+I controlled PUMA560 robot.

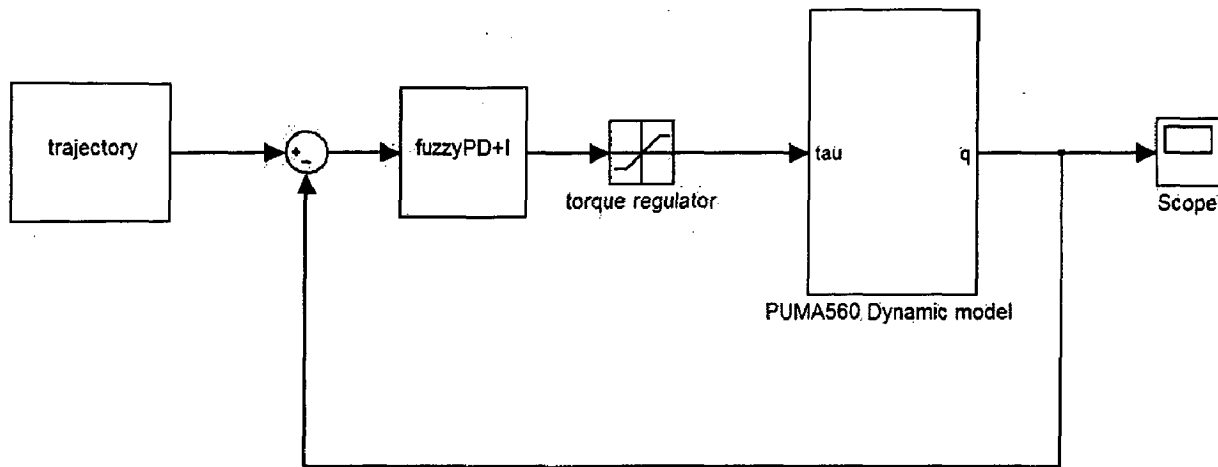


Figure 3.3 Block diagram of the Fuzzy PD+I controlled PUMA560 robot model

3.4 Weighted-rule fuzzy control systems

The use of weights flexibilizes the rule structure, whose output lies in an explicit point between the most distant consequents which is determined by the corresponding rule weights [21]. Rule weights suppose an effective extension of the conventional fuzzy reasoning process that allows tuning of the system to be developed at the rule level. This approach improves the accuracy of the learned model since they induce a good cooperation among rules. However, they come with the drawback of a small interpretability loss which lies in the difficulty to interpret the actual action performed by each rule in the interpolative reasoning process. From other point of view (rule level), when weights are applied to complete rules, the corresponding weight is used to modulate the firing strength of a rule in the process of computing the defuzzified value. For human beings, it is very close to consider this weight as an importance degree associated to the rule, determining how this rule interacts with its neighbouring ones. In addition, only weight values in range $[0, 1]$ are considered, since this preserves the model readability. In this way, the use of rule weights represents an ideal framework for extended Linguistic Fuzzy Modeling while searching for a trade-off between accuracy and interpretability. If a rule weight is applied to the consequent part of a rule, it modifies the size of a rule's output value [13]. By assigning a rule weight to each fuzzy rule, the complexity is increased while its accuracy is

improved. This also suggests a tradeoff relation between the accuracy and the complexity [12].

In order to do so, the weighted rule structure and the inference system extended for multiple output variables is followed which is taken from [22] and given by the statement below:

IF X_1 is A_1 and . . . and X_n is A_n

THEN Y_1 is B_1 and . . . and Y_m is B_m with $[w]$,

Where, X_i and Y_j are the linguistic input and output variables respectively, A_i and B_j are the linguistic labels used in the input and output variables respectively, w is the real-valued rule weight, and *with* is the operator modeling the weighting of a rule.

With this structure, the fuzzy reasoning must be extended. The classical approach is to infer with the FITA (First Infer, Then Aggregate) scheme and compute the defuzzified output of the j -th variable as the following weighted sum:

$$y(j) = \frac{\sum_h m_h \cdot w_h \cdot P_h(j)}{\sum_h m_h \cdot w_h} \quad (3.11)$$

with m_h being the matching degree of the h -th rule, w_h being the weight associated to the h -th rule, and $P_h(j)$ being the characteristic value of the output fuzzy set corresponding to that rule in the j -th variable. In this contribution, center of gravity will be considered as characteristic value and the minimum t-norm will play the role of the implication and conjunctive operators.

Chapter 4: Genetic algorithm and Genetic Fuzzy Systems

The most popular technique in evolutionary computation research has been the genetic algorithm. Evolutionary algorithms can be applied to any problems that can be formulated as function optimization problems. The genetic algorithm is a method for solving optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. Genetic algorithm can be applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non differentiable, stochastic, or highly nonlinear[23].

4.1 Genetic Algorithm

Genetic Algorithms are general purpose search algorithms which use principles inspired by natural genetics to evolve solutions to problems [24]. Genetic Algorithms were envisaged by Holland in the 1970s as an algorithmic concept based on a Darwinian-type survival-of-the-fittest strategy [2], where stronger individuals in the population have a higher chance of creating an offspring. A genetic algorithm is implemented as a computerized search and optimization procedure that uses principles of natural genetics and natural selection. The basic approach is to model the possible solutions to the search problem as strings of ones and zeros. Various portions of these bit-strings represent parameters in the search problem. If a problem-solving mechanism can be represented in a reasonably compact form, then GA techniques can be applied using procedures to maintain a population of knowledge structure that represent candidate solutions, and then let that population evolve over time through competition (survival of the fittest and controlled variation). The GA will generally include the three fundamental genetic operations of selection, crossover and mutation. These operations are used to modify the chosen

solutions and select the most appropriate offspring to pass on to succeeding generations. GAs consider many points in the search space simultaneously and have been found to provide a rapid convergence to a near optimum solution in many types of problems; in other words, they usually exhibit a reduced chance of converging to local minima. GAs show promise but suffer from the problem of excessive complexity if used on problems that are too large.

Generic algorithms are an iterative procedure that consists of a constant-sized population of individuals, each one represented by a finite linear string of symbols, known as the genome, encoding a possible solution in a given problem space. This space, referred to as the search space, comprises all possible solutions to the optimization problem at hand. Standard genetic algorithms are implemented where the initial population of individuals is generated at random. At every evolutionary step, also known as generation, the individuals in the current population are decoded and evaluated according to a fitness function set for a given problem. The expected number of times an individual is chosen is approximately proportional to its relative performance in the population. Crossover is performed between two selected individuals by exchanging part of their genomes to form new individuals. The mutation operator is introduced to prevent premature convergence.

Every member of a population has a certain fitness value associated with it, which represents the degree of correctness of that particular solution or the quality of solution it represents. The initial population of strings is randomly chosen. The strings are manipulated by the GA using genetic operators, to finally arrive at a quality solution to the given problem. GA converges rapidly to quality solutions. Although they do not guarantee convergence to the single best solution to the problem, the processing leverage associated with GAs make them efficient search techniques. The main advantage of a GA is that it is able to manipulate numerous strings simultaneously, where each string represents a different solution to a given problem. Thus, the possibility of the GA getting stuck in local minima is greatly reduced because the whole space of possible solutions can be simultaneously searched. A basic genetic algorithm comprises three genetic operators.

- Selection
- Crossover, and
- Mutation.

Starting from an initial population of strings (representing possible solutions), the GA uses these operators to calculate successive generations. First, pairs of individuals of the current population are selected to mate with each other to form the offspring, which then form the next generation. Selection is based on the survival-of-the-fittest strategy, but the key idea is to select the better individuals of the population, as in tournament selection, where the participants compete with each other to remain in the population. The most commonly used strategy to select pairs of individuals is the method of roulette-wheel selection, in which every string is assigned a slot in a simulated wheel sized in proportion to the string's relative fitness. This ensures that highly fit strings have a greater probability to be selected to form the next generation through crossover and mutation. After selection of the pairs of parent strings, the crossover operator is applied to each of these pairs.

The crossover operator involves the swapping of genetic material (bit-values) between the two parent strings. In single point crossover, a bit position along the two strings is selected at random and the two parent strings exchange their genetic material as illustrated below.

Parent A = $a_1 a_2 a_3 a_4 | a_5 a_6$

Parent B = $b_1 b_2 b_3 b_4 | b_5 b_6$

The swapping of genetic material between the two parents on either side of the selected crossover point, represented by "|", produces the following offspring:

Offspring A' = $a_1 a_2 a_3 a_4 | b_5 b_6$

Offspring B' = $b_1 b_2 b_3 b_4 | a_5 a_6$

The two individuals (children) resulting from each crossover operation will now be subjected to the mutation operator in the final step to forming the new generation.

The mutation operator alters one or more bit values at randomly selected locations in randomly selected strings. Mutation takes place with a certain probability, which, in accordance with its biological equivalent, typically occurs with a very low probability. The mutation operator enhances the ability of the GA to find a near optimal solution to a given problem by maintaining a sufficient level of genetic variety in the population, which is needed to make sure that the entire solution space is used in the search for the best solution. In a sense, it serves as an insurance policy; it helps prevent the loss of genetic material.

Genetic algorithms are most appropriate for optimization type problems, and have been applied successfully in a number of automation applications including job shop scheduling, proportional integral derivative (PID) control loops, and the automated design of fuzzy logic controllers. The reason for a great part of this success is their ability to exploit the information accumulated about an initially unknown search space in order to bias subsequent searches into useful subspaces, i.e., their adaptation. This is their key feature, particularly in large, complex and poorly understood search space, where classical search tools (enumerated, heuristic...) are inappropriate, offering a valid approach to problems requiring efficient and effective search techniques. The pseudo-code in Figure 4.1 shows the structure of a Basic GA [24], where $P(t)$ denotes the population at generation t and $Recombine(t)$ will consist of both the crossover and mutation operations.

A genetic algorithm is typically initialized with a random population consisting of between 20-100 individuals. This population is usually represented by a real-valued number or a binary string. How well an individual performs a task is measured by the objective function. The objective function assigns each individual a corresponding number called its fitness value. The fitness of each chromosome is assessed and a survival of the fittest strategy is applied.

Procedure Genetic Algorithm

Begin (1)

t=0;

Initialize P(t);

Evaluate P(t);

While (Not *termination-condition*) do

Begin (2)

t=t+1;

Select P(t) from P(t-1);

Recombine P(t);

Evaluate P(t);

End (2)

End (1)

Figure 4.1 Pseudo-code for structure of a Basic GA

4.2 Genetic Fuzzy Systems

One of the major drawbacks of Fuzzy Rule Base Systems (FRBS) discussed in the chapter 3 is that they are not able to learn, but require the Knowledge Base (KB) to be derived from expert knowledge [10]. The first step in designing a Genetic FRBS is to decide which parts of the (KB) are subject to optimization. The KB of a descriptive Mamdani-type FRBS is comprised of two components: a Data Base (DB), containing the definitions of the scaling factors and the membership functions of the fuzzy sets associated with the linguistic labels, and a Rule Base (RB), constituted by the collection of fuzzy rules. Fuzzy Logic Controller (FLC) contains a number of sets of parameters that can be altered to modify the controller performance, they are 1. Scaling factors for each variable, 2. The fuzzy sets representing the meaning of linguistic values, 3. The If-Then rules. Each of these parameters has been used as the controller parameter to be adapted in different adaptive FLCs. GAs have been used to modify the fuzzy set definitions, to alter the shapes of the fuzzy sets defining the meaning of the linguistic terms, to determine the membership functions that produce maximum control performance according to the inference system

(fuzzy implication and conjunctive operators) and the defuzzification strategy used. That is, to tune the Fuzzy set, in order to make the FLC behaves as closely as possible to the operator or expert behavior [9]. Changing any of the above parameter will result in considerable change in fuzzy control system. In addition to these the Rule-weights can also be changed to perform a local tuning of linguistic rules. In Linguistic Fuzzy Modeling, tuning of any fuzzy set will influence all the rules that are involved [21].

The objective of a genetic fuzzy system is to automate the knowledge acquisition step in fuzzy system design, a task that is usually accomplished through an interview or observation of a human expert controlling the system [25]. An evolutionary algorithm adapts either part or all of the components of the fuzzy knowledge base. At this point, it is important to notice that a fuzzy knowledge base is not a monolithic structure but is composed of the data base and the rule base which each play a specific role in the fuzzy reasoning process. According to the distinction between data base and rule base, genetic fuzzy systems are discriminated along two major approaches, genetic tuning processes and genetic learning processes. The first method is targeted at optimizing the performance of an already existing fuzzy system. The tuning process involves the adaptation of the fuzzy database, namely parameters of membership functions and input-output scaling factors. The second method is concerned with the automatic derivation of fuzzy rules in the rule base. A genetic learning process faces a much more difficult task as it has to establish the proper relationship between input and output states from scratch, rather than optimizing the performance of a fuzzy system that already operates at least approximately correct. Designing a fuzzy rule based system is equivalent to finding an optimal configuration of fuzzy sets and/or rules, and in that sense can be regarded as an optimization problem. The optimization criterion is the problem to be solved at hand and the search space is the set of parameters that code the membership functions, scaling functions and fuzzy rules. The genetic learning process emerges from the hybridization of an evolutionary algorithm, which by means of selection and genetic operators optimizes parameters of the knowledge base, with the fuzzy system supposed to demonstrate a desired behavior.

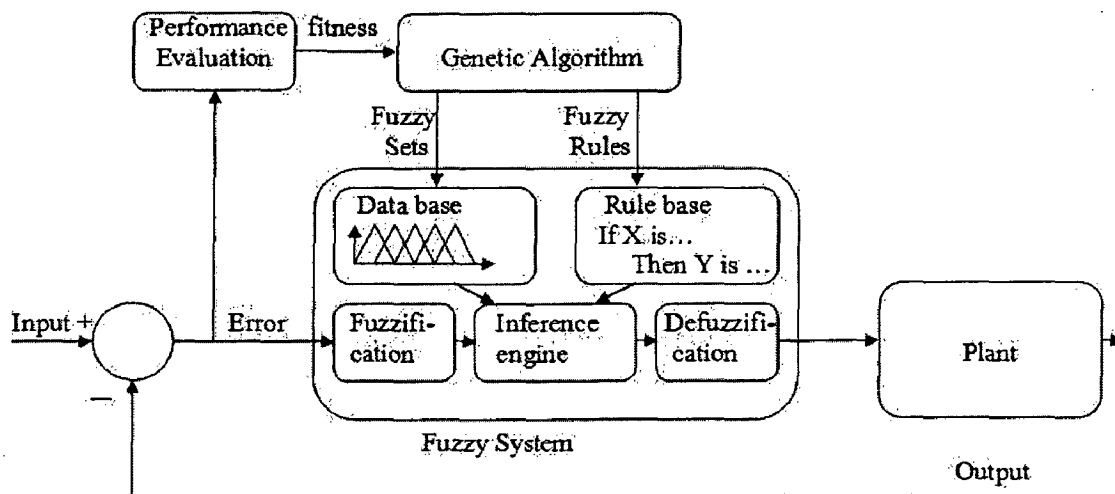


Figure 4.2 Genetic Fuzzy System

The fuzzy system lies at the core of the hybrid structure; it fuzzifies the input state, performs the inference based on the fuzzy rules and aggregates the result of the inference process into a crisp output. Depending on the context, the environment can be a plant to be controlled, a system to be modeled or a set of data to be classified. An external critic or trainer evaluates the performance of the fuzzy system with regard to the control task, the model accuracy or the classification error. The performance is aggregated into a scalar fitness value on which basis the evolutionary algorithm selects better adapted chromosomes. A chromosome either codes parameters of membership functions, scaling factors, fuzzy rules or a combination thereof. By means of crossover and mutation, the evolutionary algorithm generates new parameters for the database and/or rule base which usefulness is tested in the fuzzy system.

It is important to distinguish between tuning and learning problems. Tuning is more concerned with optimization of an existing FRBS, whereas learning constitutes an automated design method for fuzzy rule sets that starts from scratch. Tuning processes assume a predefined RB and have the objective to find a set of optimal parameters for the membership and/or the scaling functions. Learning processes perform a more elaborated search in the space of possible RBs or whole KBs and do not depend on a predefined set of rules.

In the case of tuning membership functions, an individual represents the entire DB as its chromosome encodes the parameterized membership functions associated to the

linguistic terms. Triangular membership functions are usually encoded by their left, centre and right point, whilst Gaussian membership functions by their centre and width. When tuning the membership functions in a linguistic model, the whole fuzzy partitions is encoded into the chromosome and they are globally adapted to maintain the global semantic in the RB. On the other hand, tuning the membership functions of an approximate model is a particular instantiation of KB learning since the rules are completely defined by their membership functions instead of referring to linguistic terms in the DB. In this thesis work, Gaussian membership function is used. A small analysis of how the variations in the mean and spread change the shape of the membership function. Gaussian membership function is characterized by Equation 5, where μ is the center and σ denotes the spread.

$$f(x, \sigma, c) = e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (4.1)$$

Figure 4.3 shows the variation of the Gaussian membership function when the mean (center) is varied. Initially the MF is centered at ' μ ', a change of ' δ ' introduced in ' μ ' resulting in the center getting shifted to ' $\mu+\delta$ '. Figure 4.4 shows the variation of the Gaussian membership function when the spread ' σ ' is varied. The plot ' $\sigma = b$ ' is the initial MF, when ' σ ' is increased to ' a ' the plot ' $\sigma = a$ ' is formed and when ' σ ' is decreased to ' c ' the plot ' $\sigma = c$ ' is formed.

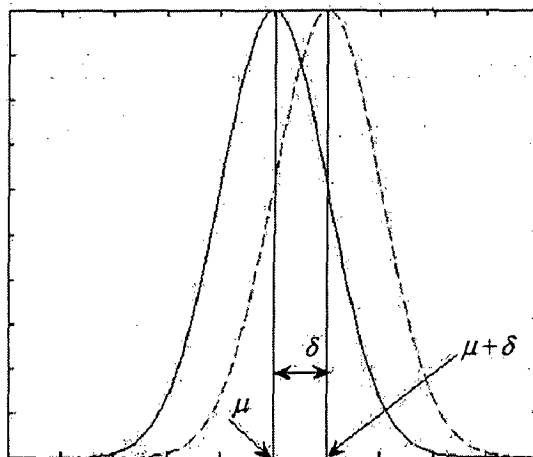


Figure 4.3 Variation in the mean of a Gaussian MF

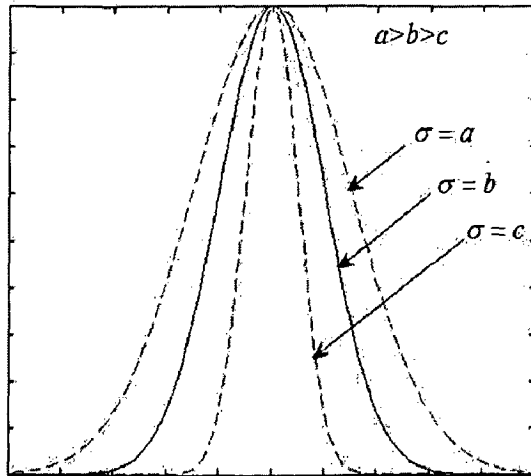


Figure 4.4 Variation in the spread of a Gaussian MF

On the other hand, Genetic learning of the RB assumes a predefined set of fuzzy membership functions in the DB to which the rules refer to by means of linguistic labels. The GA adapts the RB, either working with chromosomes that describe a single fuzzy rule or an entire RB. The RB is either represented by a relational matrix, a decision table or a list of rules. In case each chromosome represents an individual rule, the population as a whole constitutes the solution, namely the optimal set of rules. The rules that form the RB are either evolved simultaneously.

Chapter 5: Implementations

In this chapter, implementation of parameter tuning methods, approximation of systems, obtaining minimally optimal rule base and proposal of a novel algorithm is discussed.

5.1 Parameter tuning

Fuzzy controllers can be tuned by various strategies, like changing the scaling factor, modifying the support and spread of membership functions, modifying the If-Then rules of the rulebase and changing the type of a membership function itself. Tuning the scaling factors, rules and shape and support of a membership function will result in change of the control surface and hence the output of the fuzzy controller [26]. GAs have been used to modify the fuzzy set definitions, to alter the shapes of the fuzzy sets defining the meaning of the linguistic terms, to determine the membership functions that produce maximum control performance according to the inference system (fuzzy implication and conjunctive operators) and the defuzzification strategy used. That is, to tune the Fuzzy set, in order to make the FLC behaves as closely as possible to the operator or expert behavior [9]. In addition to these the Rule-weights can also be changed to perform a local tuning of linguistic rules, which enables the linguistic fuzzy models to cope with inefficient and/or redundant rules thereby enhancing the robustness, flexibility and system modeling capability [11]. By assigning a rule weight to each of the fuzzy rules, complexity is increased while its accuracy is improved. This suggests a tradeoff relation between the accuracy and complexity [12]. If a rule weight is applied to the consequent part of the rule, it modifies the size of the rule's output value [13]. In Linguistic Fuzzy Modeling the tuning of any fuzzy set will influence all the rules that are involved [21].

5.1.1 Preliminary tuning

As part of the preliminary studies three types of tuning are performed. Namely, Rule tuning, Rule-weight tuning, and membership function tuning. This is performed on the reference Fuzzy PD+I control system mentioned in the section 3.3. After this

preliminary studies and inference is drawn as to which of the tuning strategy is better than the other.

5.1.1.1 Rule tuning

In this subsection the chromosomes are encoded with the values of consequent part of the if-then rule of the fuzzy rule base, in other words contains the parameters of the consequent. Genetic algorithm is run until the terminating condition. Genetic algorithm varies these parameters stochastically and on convergence produces a rule base which will have optimal rules in it. Since we are using a 3x3 rulebase we will require chromosomes to consist of 9 variables per fuzzy system corresponding to the 9 rules.

Structure of chromosome: C_1, C_2, \dots, C_n .

Where C_i is the consequent, i varies from 1 to n . In this case $n=54$ (i.e. $9\text{rule} \times 6\text{joints}$).

5.1.1.2 Rule-Weight tuning

In this subsection the chromosomes are encoded with the values of weighting for consequent part of the if-then rule of the fuzzy rule base. Genetic algorithm is run until the terminating condition. Genetic algorithm varies these parameters stochastically and on convergence produces a rule base which will have rules with optimal rule-weights in it. Since we are using a 3x3 rulebase we will require chromosomes to consist of 9 variables per fuzzy system corresponding to the 9 rule weights.

Structure of chromosome: W_1, W_2, \dots, W_n .

Where W_i is the rules-weight, i varies from 1 to n . In this case $n=54$ (i.e. $9\text{rule-weights} \times 6\text{joints}$).

5.1.1.3 Membership function tuning

In this subsection the chromosomes are encoded with the values of parameters of the membership function. Gaussian membership function is used. A Gaussian membership function is characterized by a mean and a spread. The centers of the extreme end membership functions are kept as it is since changing them will affect the universe of discourse and it is desired that the universe of discourse is kept a constant. Genetic

algorithm is run until the terminating condition. Genetic algorithm varies these parameters stochastically and on convergence produces a fuzzy system which will have membership functions with mean and spread. Since we are using a 3x3 rule fuzzy system there will be 9 membership functions altogether, 6 for input and 3 for output. Considering this there will be 9 means and 9 spreads, of these 6 extreme centers are kept untouched hence the total number of parameters tuned is 12. We will require chromosomes to consist of 12 variables per fuzzy system

Structure of chromosome: MF_1, MF_2, \dots, MF_n .

Where MF_i is the membership function parameter, i varies from 1 to n . In this case $n=72$ (i.e. $12Mf$ parameters* $6joints$).

5.1.2 Two stage tuning

With the foundations of the preliminary studies, an attempt is made to tune the Fuzzy controller in two stages. In the first stage the rules are tuned and in the second stage the rule-weights is tuned. This progressive approach is studied to see if there are any performance improvements. Changing the rules will produce considerable change in the control of the fuzzy controller, but changing the rule weight results in finer details getting adjusted.

Stage 1: The chromosomes are encoded with the values of consequent part of the if-then rule of the fuzzy rule base, in other words contains the parameters of the consequent. Genetic algorithm varies these parameters stochastically and on convergence produces a rule base which will have optimal rules in it. Since we are using a 3x3 rulebase we will require chromosomes to consist of 9 variables per fuzzy system corresponding to the 9 rules.

Structure of chromosome: C_1, C_2, \dots, C_n .

Where C_i is the consequent, i varies from 1 to n . In this case $n=54$ (i.e. $9rule*6joints$).

Stage 2: With the rulebase obtained in the previous stage weight tuning is performed as follows. The chromosomes are encoded with the values of weighting for consequent part of the if-then rule of the fuzzy rule base Genetic algorithm varies these parameters stochastically and on convergence produces a rule base which will have rules

with optimal rule-weights in it. Since we are using a 3x3 rulebase we will require chromosomes to consist of 9 variables per fuzzy system corresponding to the 9 rule weights.

Structure of chromosome: W_1, W_2, \dots, W_n .

Where W_i is the rules-weight, i varies from 1 to n . In this case $n=54$ (i.e. 9rule-weights*6joints).

5.1.3 Three stage tuning

As an extension to the previous subsection a third stage tuning is performed. In this section the membership function of the fuzzy system obtained from the two-stage tuning process is tuned. All the constraints described in the preliminary study of tuning for membership function applies here as well.

Structure of chromosome: MF_1, MF_2, \dots, MF_n .

Where MF_i is the membership function parameter, i varies from 1 to n . In this case $n=72$ (i.e. 12Mf parameters*6joints).

5.1.4 Simultaneous tuning

In contrast to the previous two subsection where tuning was carried out progressively, tuning one parameter at a time, in this section the implantation of simultaneously tuning Rule, Rule-weight and Membership functions is discussed. The chromosomes are encoded with the parameters of the consequent, weighting of consequent part and membership function. Genetic algorithm is run until the terminating condition. Genetic algorithm varies these parameters stochastically and on convergence produces a fuzzy system which will have optimal rules, rule-weights and membership function in it. Since we are using a 3x3 rulebase we will require chromosomes to consist of 30 variables per fuzzy system corresponding to the 9 rules, 9 rule weights and 12 membership function parameters.

Structure of chromosome: $C_1, C_2, \dots, C_n, W_1, W_2, \dots, W_n, MF_1, MF_2, \dots, MF_m$

Where C_i is the consequent, W_i is the rules-weight, i varies from 1 to n . MF_j is the membership function parameter, j varies from 1 to m . In this case $n=54, m=72$, total

number of parameters per structure is 180 (9 rule*6joints+9 rule weights*6joints+12 Mf parameters*6joints)

5.2 Approximations of a System

Many applications exist in the control and signal processing areas that utilize nonlinear function approximation. One such application is system identification, which is the process of constructing a mathematical model of a dynamic system using experimental data from that system [20]. The interpolation and approximation theory are quite mature fields in by themselves which has received and continues receiving not deep but constant attention [14].

This section begins by defining the function approximation problem, in which a synthesis of a function to approximate another function that is inherently represented via a finite number of input-output associations is required [20]. Appropriate input-output data points that allow for the construction of an approximate model are gathered.

Given some function

$$g : \bar{X} \rightarrow \bar{Y} \quad (5.1)$$

Where $\bar{X} \subset R^n$ and $\bar{Y} \subset R$, it is desired to construct an approximate model of the system

$$f : X \rightarrow Y \quad (5.2)$$

Where $X \subset \bar{X}$ and $Y \subset \bar{Y}$ are some domain and range of interest, by choosing parameter vector θ (which includes, polynomial coefficients in case of polynomial interpolation and rules, membership function centers, widths, etc in case of a fuzzy system.)

So that

$$g(x) = f(x | \theta) + e(x) \quad (5.3)$$

For all $x = [x_1, x_2, \dots, x_n]^T \in X$ where the approximation error $e(x)$ is as small as possible. If it is required to refer to the input at time k , then use $x(k)$ for the vector and $x_j(k)$ for its j^{th} component. In this work $x_j(k)$ will consists of input data pair, this input data pair set is the training data 'T'. Evaluation of the error in approximation between g and an approximate function $f(x|\theta)$ based on a training data set may or may not be a true measure of the error between g and f for every $x \in X$, but it is the only evaluation we can make based on known information. Hence, measures like 'sum of squares of error' given in

Equation (5.4) to measure the approximation error are used. Accurate function approximation requires that expression of this nature be small;

$$\sum_{x_i \in T} (g(x_i) - f(x_i, \theta))^2 \quad (5.4)$$

While the method for adjusting the parameters θ of $f(x|\theta)$ is critical to the overall success of the approximation method, there is virtually no way of succeeding at having f approximate g if there is no appropriate information present in the training data set 'T'. Basically, we would like 'T' to contain as much information as possible about g . Unfortunately, most often the number of training data pairs is relatively small, or it is difficult to use too much data since this affects the computational complexity of the algorithms that are used to adjust θ .

In this thesis work, two types of approximations, namely Bivariate polynomial approximation and Weighted-rule Fuzzy system approximation are studied and their results are compared. Figure 5.1 shows the Structure of approximate of Fuzzy PD+I controller.

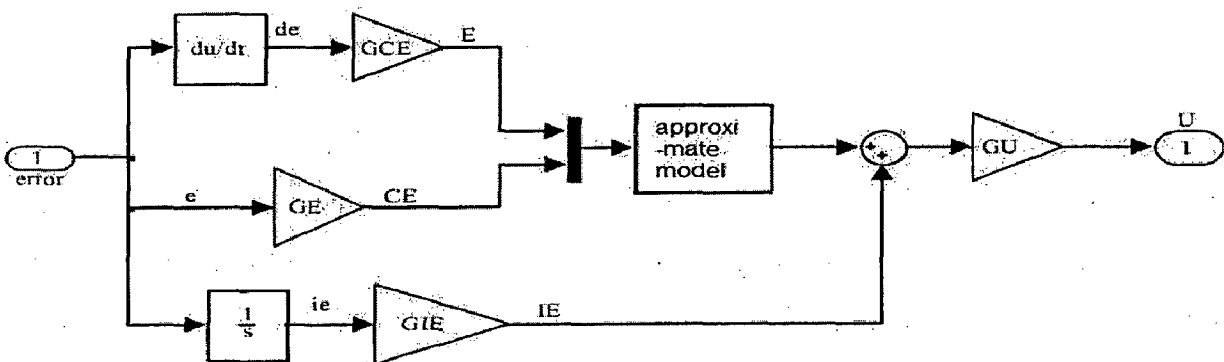


Figure 5.1 Structure of the approximate of Fuzzy PD+I controller

5.2.1 Bi-Variate Polynomial Approximation

Various methods of polynomial interpolation give efficient results but are mathematically quite rigorous and difficult for a multi-variate case [15]. The simplest context to study here is interpolation by uni-variate polynomials. Hence interpolation by uni-variate polynomials is a very classical topic. However, interpolation by polynomials of several variables is much more intricate and is a subject which is currently an active area of research. A bi-variate polynomial is a polynomial in two variables [27]. Bi-variate polynomials have the form as given in Equation (5.5).

$$f(x, y) = \sum_{i,j} a_{i,j} x^i y^j \quad (5.5)$$

A bi-variate polynomial when evaluated over a grid of input values for both x and y, traces a surface defined by $f(x,y)$ of the Equation (5.5). This section proposes to approximate the control surface of the fuzzy controller using bi-variate polynomial function. Approximation is performed using genetic algorithm, which is one of the stochastic optimization algorithm, to determine the coefficients of polynomial that approximates a system without getting into rigorous mathematical analysis.

In this work the bi-variate polynomial chosen for the approximation of the reference fuzzy control surface is given as $(x+y+1)^{10}$, with all the coefficients being replaced by variables that can get modified by genetic algorithm so that an optimal value of these coefficients can be found. The polynomial so found closely approximates fuzzy controller behavior in the region of operation for which the data points were collected.

Let us assume that the data points collected from the reference fuzzy controller are x_1 and x_2 for inputs and y for outputs, where x_1 , x_2 and y are vectors of same size. With these data points an estimate bi-variate polynomial model is created, bi-variate in terms of x_1 and x_2 variables. The squared error between the estimate model output vector $f(x,y)$ and the y vector is calculated. This squared error is minimized by the Genetic Algorithm. This minimization is done by manipulating the coefficients of the polynomial by genetic algorithm to obtain an approximate model of fuzzy controller in terms of bi-variate polynomial function. This approximate model is employed in the reference Fuzzy PD+I control block by replacing the Fuzzy PD controller as shown in Figure 5.1. Equation 5.6 shows the relation between the polynomial $P(x_1, x_2)$ and Fuzzy PD function $F(e, e)$.

$$P(x_1, x_2) \approx F(e, e) \quad (5.6)$$

Where $P(x_1, x_2)$ is of the form specified in Equation 5.5

Structure of chromosome : P_1, P_2, \dots, P_n .

Where P_i is the polynomial coefficients, W_i is the rules-weight, i varies from 1 to n . In this case $n=67$ the number of coefficients present for polynomial with structure as that equation $(x+y+1)^{10}$

5.2.2 Weighted-Rule Fuzzy Approximation

The basic problem to be studied here is how to construct a Weighted-Rule fuzzy system from numerical data, where linguistics is used as the starting point to specify a fuzzy system. If the numerical data is plant input-output data obtained from an experiment, we may identify a fuzzy system model of the plant. This may be useful for simulation purposes and sometimes for use in a controller [20]. On the other hand, the data may come from other sources, and a fuzzy system may be used to provide for a parameterized nonlinear function that fits the data by using its basic interpolation capabilities. For instance, suppose that we have a human expert who controls some process and we observe how he or she does this by observing what numerical plant input the expert picks for the given numerical data that she or he observes.

Let us assume that the data points collected from the reference fuzzy controller are x_1 and x_2 for inputs and y for outputs, where x_1 , x_2 and y are vectors of same size. From the numerical data, the maximum and minimum are calculated and the universe of discourse (UOD) is assigned ± 1.5 times the maximum of the absolute values of maximum and minimum. The membership function chosen is Gaussian membership functions for both input and output. A 3x3 rulebase system is being developed so the centers of the Gaussian membership function are assigned as $[-UOD \text{ value}, 0, +UOD]$. The spread for these membership functions are assigned as $\text{Range}/(1.5*\pi)$, where Range is (2 times UOD), this is done so that the cross-point of the membership functions occurs at 0.5. A symmetric rulebase as shown in the Figure 5.2 is chosen and assigned to the fuzzy system being created.

| | | | |
|---------|---|---|---|
| de e | N | Z | P |
| N | N | N | Z |
| Z | N | Z | P |
| P | Z | P | P |

Figure 5.2 Symmetric rulebase used in the Weighted-Rule Fuzzy Approximation

The weights are assigned to this rulebase genetically to obtain the approximate model of the reference system. The sum of squared errors between the estimate model output vector $F_w(x1,x2)$ and the y vector is calculated. This sum of squared errors is minimized by the Genetic Algorithm. The chromosomes are encoded with the values of weighting for consequent part of the if-then rule of the fuzzy rule base. Genetic algorithm is run until the terminating condition. Genetic algorithm varies these parameters stochastically and on convergence produces a rule base which will have rules with optimal rule-weights in it. Since we are using a 3x3 rulebase we will require chromosomes to consist of 9 variables per fuzzy system corresponding to the 9 rule weights.

Structure of chromosome: W_1, W_2, \dots, W_n .

Where W_i is the rules-weight, i varies from 1 to n . In this case $n=9$.

This approximate model is employed in the reference Fuzzy PD+I control block by replacing the Fuzzy PD controller as shown in Figure 5.1. Equation (5.7) shows the relation between the reference Fuzzy function $F(x1,x2)$ and Weighted-rule Fuzzy PD approximate function $F_w(e,e)$.

$$F(x1,x2) \approx F_w(e,e) \quad (5.7)$$

Where $F(x1, x2)$ is the fuzzy function of the reference Fuzzy PD+I for the inputs $x1$ and $x2$. $F_w(e,e)$ is the fuzzy function of the weighted-rule Fuzzy approximate model.

5.3 Optimally Minimum Rulebase

For any given input trajectory, all the rules in the rulebase are not fired. This suggests that the rulebase can be minimized. Along with minimization if the rules present in the minimal set are tuned, then an optimally minimum rulebase is obtained. Rulebase is tuned in such a way that the number of rules is minimized and optimized simultaneously.

In this section the chromosomes are encoded in such a way that the chromosome is broken down into two sub-chromosomes, the first contains the parameters of the consequent and the second contains binary weights [16]. Genetic algorithm is run until the terminating condition. Genetic algorithm varies these parameters stochastically and on

convergence produces a rule base which will have an Optimally Minimum Rules in it. In other words, with the various genetic operations the best consequent and the binary weights are obtained. The consequents with weights '0' is equivalent to considering this rule non existent.

Structure of chromosome: $C_1, C_2, \dots, C_n, W_1, W_2, \dots, W_n$

Where C_i is the consequent and W_i , is the associated binary weight, i varies from 1 to n . Overall to find an optimally minimum rulebase for a rule base of size 'n' the length of the chromosome required is '2n', 'n' for consequents and 'n' for weights. Here we are using a 3x3 rulebase and hence we will have $n=9$. In case of consequent sub-chromosome each consequent requires as much size as a positive integer and for weights one bit is sufficient since it is a binary weight.

A small disturbance is given to the robot arm and the trajectory tracking is evaluated to check if the created Optimally Minimum Rulebase is able to cope up with the disturbance.

5.4 A Novel Stochastic Algorithm for optimization

Given a space Ω of individual solutions $\omega \in R^n$ and an objective function $f, f(\omega) \rightarrow R$, optimizing f is the process of finding the solution ω^* which minimizes (maximizes) f .

Random search consists of picking up random potential solutions and evaluating them. The best solution over a number of samples is the result of random search. Stochastic algorithm is nothing other than a random search, with hints by a chosen heuristics (or meta-heuristics) to guide the next potential solution to evaluate [28]. Stochastic optimization algorithms were designed to deal with highly complex optimization problems. There are a number of stochastic search algorithms present, Genetic algorithm, simulated annealing, ant colony optimization, etc to name a few. In this section the proposed algorithm is compared with the genetic algorithm, the scope of work is restricted to comparison with genetic algorithm only.

The proposed stochastic search algorithm is initially tested on Rastrigin's function and then later is used to minimize the ISE of the joint for trajectory tracking control of the PUMA560.

The search algorithm is divided into three phases. In the first phase the search space is searched thoroughly and if the solution exists outside this search space then the space is extended, this can be considered as a global search. In the second phase the search space is restricted but still able to change its neighborhood considerably, this can be considered as a local search. And in the third phase the search is completely restricted to a very small space in the neighborhood of the solution, this can be considered as a highly restricted local search phase.

Before starting the algorithm, as part of initialization a random number vector 'x' is generated within the range specified as the search space, this is done just once. With the start of the algorithm, value of x is used as the centre and absolute value of x is used as spread of the normal distribution in the first iteration. The spread should always be positive. The random numbers are generated by a Gaussian function, in other words they are normally distributed. After the first iteration the number generated by the normal distribution is taken as 'x'. The algorithm is discussed phase by phase. Let y be the function to be minimized.

Phase I: This is the phase of global search, so the neighborhood to be searched from the present solution has to be very large. This is ensured by making the spread of the normal distribution equal to the arithmetic mean of the best solution of y so far and the absolute value of random number x generated in the previous iteration by the normal distribution. By using the number generated in the previous iteration in calculating the spread, it is ensured that there is a certain degree of randomization or perturbation in search neighborhood, which will ensure that unknown better solutions can be explored. The mean of the normal distribution will be the values of x corresponding to the solution of best y. If a better solution is obtained the best x and best y are updated. When best x is updated the centre of the distribution also moves to this location. This way it is ensured that the search space is made variable moving towards better solution. The reason for using arithmetic mean for spread in this stage is that the value spread will be half way between the x and y. For example let us consider that y is a function of x where x is a vector of size 1. Now if for x=10, best y = 0.2, then the spread that will be used for next search will be $(10+0.2)/2$ which is a value 5.1. This is a pretty large value of spread considering the value of x.

Phase II: This is the phase of moderate local search, so the neighborhood to be searched from the present solution has to be moderate. This is ensured by making the spread of the normal distribution equal to the geometric mean of the best solution of y so far and the absolute value of random number x generated in the previous iteration by the normal distribution. The mean of the normal distribution will be the values of x corresponding to the solution of best y . If a better solution is obtained the best x and best y are updated. When best x is updated the centre of the distribution also moves to this location. The reason for using geometric mean for spread in this stage is that the value of spread will be very close to the minimum value of x and y . For example let us consider that y is a function of x where x is a vector of size 1. Now if for $x=10$, best $y = 0.2$, then the spread that will be used for next search will be $\sqrt{10*0.2}$ which is 1.41. This is a moderate value of spread considering the value of x .

Phase III: This is the phase of extreme local search, so the neighborhood to be searched from the present solution has to be very small. This is ensured by making the spread of the normal distribution equal to the harmonic mean of the best solution of y so far and the absolute value of random number x generated in the previous iteration by the normal distribution. The mean of the normal distribution will be the values of x corresponding to the solution of best y . If a better solution is obtained the best x and best y are updated. When best x is updated the centre of the distribution also moves to this location. The reason for using harmonic mean for spread in this stage is that the value of spread will be very close to the minimum value of x and y . For example let us consider that y is a function of x where x is a vector of size 1. Now if for $x=10$, best $y = 0.2$, then the spread that will be used for next search will be $\frac{2*(10*0.2)}{(10+0.2)}$ which is 0.39. This is a small value of spread considering the value of x .

The value of the x corresponding to the best y is returned as a result of possible solution. Flow chart of the proposed algorithm is shown in Figure 5.3.

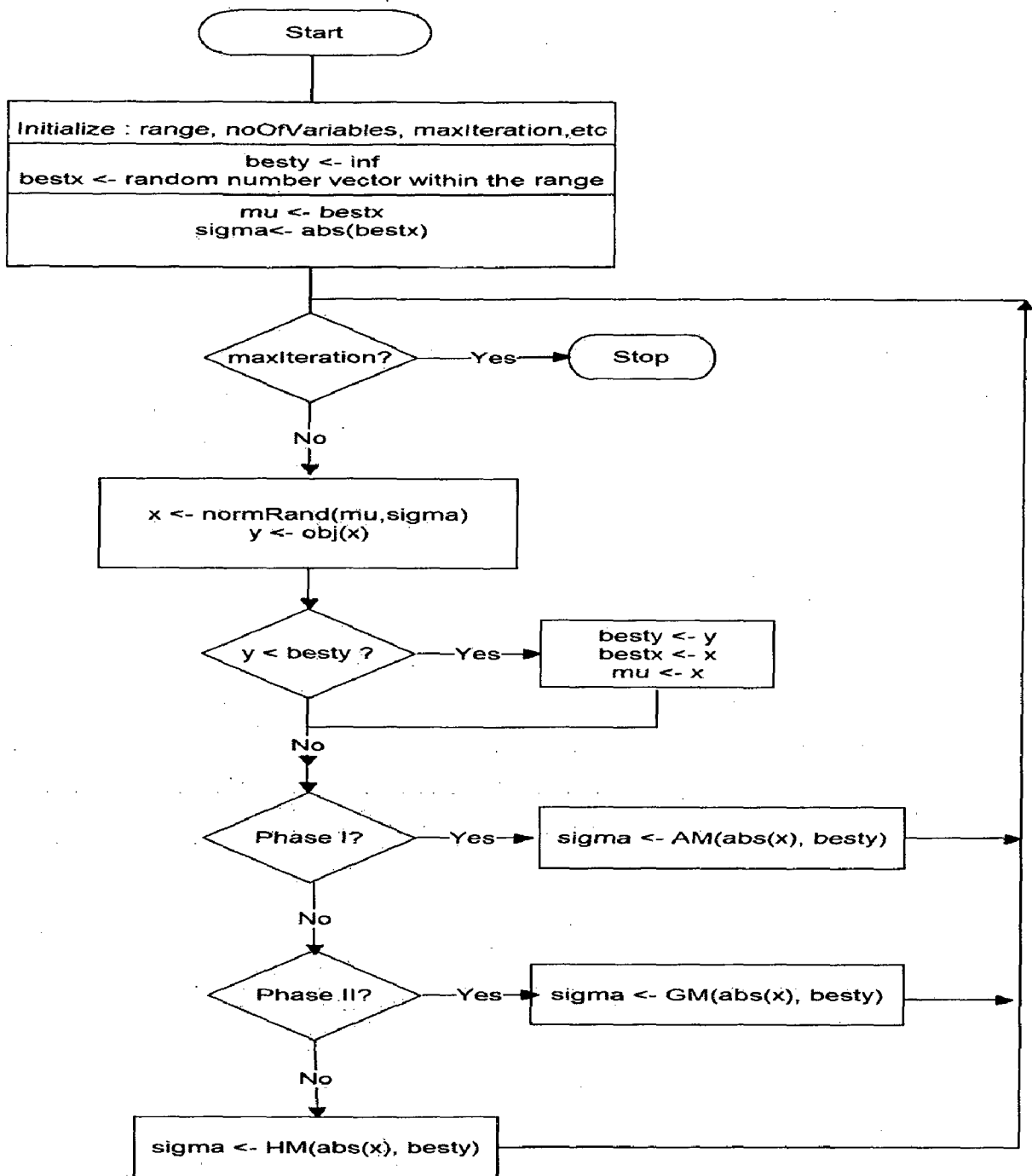


Figure 5.3 Flow chart of the proposed stochastic optimization algorithm

This algorithm is initially used to find the minimum of Rastrigin's function. A comparison of Rastrigin's function results by the proposed algorithm and Genetic algorithm is made. Based on the success it is implemented for tuning the gains of the Fuzzy PD+I controlled Puma 560 arm for pseudo-random joint trajectories. A comparison of results by the proposed algorithm and Genetic algorithm is made.

Chapter 6: Simulation results and discussions

In this chapter, the results of different methodologies mentioned in chapters 5 are discussed. Initially a preliminary study of tuning individual parameters in fuzzy PD+I controller for joint control of PUMA560 robot arm is carried out and an inference is drawn. Next two stage tuning procedure and three stage tuning procedure are carried out and three stage tuning is compared with simultaneous tuning procedure. Following this, approximations of the Fuzzy PD+I are studied. Two methods of approximating the fuzzy PD+I from input-output data, namely Bivariate Polynomial approximation and weighted fuzzy logic approximation are carried out and a comparison is made. Further, optimally minimum rulebase generation is discussed. Finally an algorithm is proposed which has faster convergence than genetic algorithm.

Default Genetic Algorithm settings for the MATLAB GA toolbox are used and are given in the Appendix A.2

The objective function considered here is based on the error criterion. In this dissertation, performance of membership functions, rules and weight tuning are evaluated in terms of Integral Square Error (ISE) error criteria. The error criterion is given as a measure of performance index. The ISEs of individual joints are added together to obtain an overall ISE. This is done to simplify the task of Genetic Algorithm. The objective of Genetic Algorithm is to minimize this overall ISE. The overall ISE is given by Equation (6.1).

$$ISE = \sum_{i=1}^6 \int e_i^2(t) dt \quad (6.1)$$

Where $e_i(t)$ is the error signal for the i^{th} joint. Here i can take values from 1 to 6 corresponding to 6 joints.

6.1 Results for parameter tuning

Tuning of rules, weights and membership functions has been carried out genetically till a best fitness is achieved. Some of the details that are to be taken care of are, the weights need to be within the range [0 1], the rules generated must be valid, the universe of

discourse should be kept same as the base system, the centers of the membership functions mf1 and mf3 are kept at -1 and 1 respectively and the center of the membership function mf2 is varied by the genetic algorithm to obtain an optimized location. The spread of all the membership functions are changed by the genetic algorithm. Both the spread and center are optimized in parallel while MF tuning. In this section the results for the section 5.1 are presented. For the preliminary tuning the minimum number of generation for which the GA is run is 200 and there after a stall limit of 50 is used upto a maximum of 500 generations.

6.1.1 Preliminary tuning

In this section the results of Base system, Rule tuning, Rule weight tuning and Membership function tuning are discussed.

6.1.1.1 Base system

Figure 6.1 shows the structure of the Fuzzy PD+I controller, where GCE, GE, GIE and GU are the gains of Fuzzy PD+I controller and more often called scaling factors which can be varied to tune the controller. Here Genetic Algorithm is used to coarsely tune (to represent manual tuning) these gains initially in order to produce base or reference system. All other parameter tuning in this section is carried out on this base system. Figure 6.2 shows the surface view, membership function the rulebase of the reference Fuzzy PD controller used.

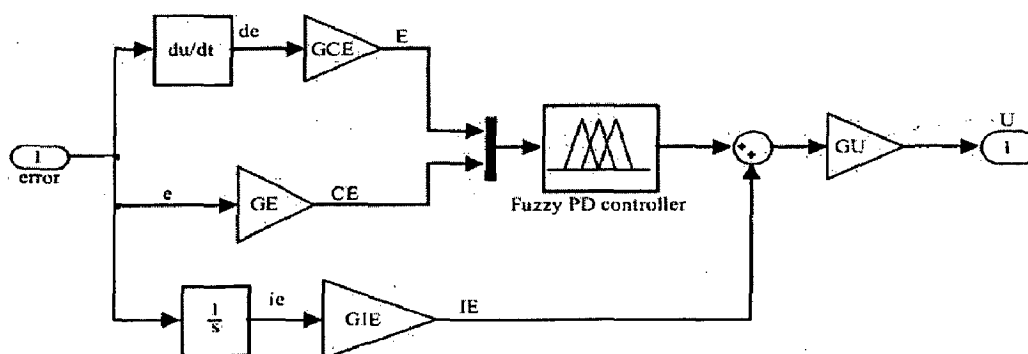
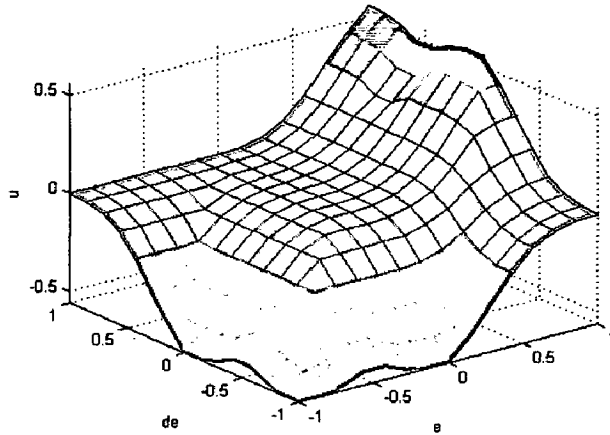
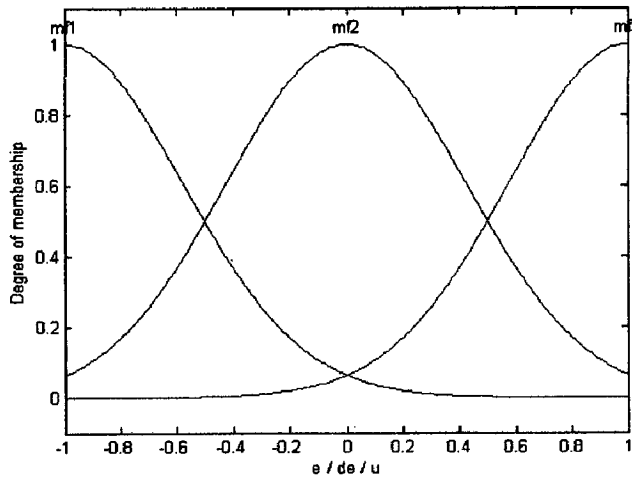


Figure 6.1 Structure of the Fuzzy PD+I controller used in base system



(a)



(b)

| $\begin{matrix} de \\ e \end{matrix}$ | mf1 | mf2 | mf3 |
|---------------------------------------|-----|-----|-----|
| mf1 | mf1 | mf1 | mf2 |
| mf2 | mf1 | mf2 | mf2 |
| mf3 | mf2 | mf3 | mf3 |

(c)

Figure 6.2 (a) The surface view, (b) membership function and (c) rulebase of the reference Fuzzy PD controller used.

While providing the input to the robot, care has to be taken that the trajectory used is continuous and smooth. The input trajectory used in this subsection is $1 - \cos(\pi/t)$ since it is a continuous and smooth function which can be double differentiated, t is time in seconds. This trajectory is used for all the joints.

Figure 6.3 shows Input trajectory signal given to the individual joints and Figure 6.4 shows the joint error generated by the given input trajectory for reference Fuzzy PD+I controller.

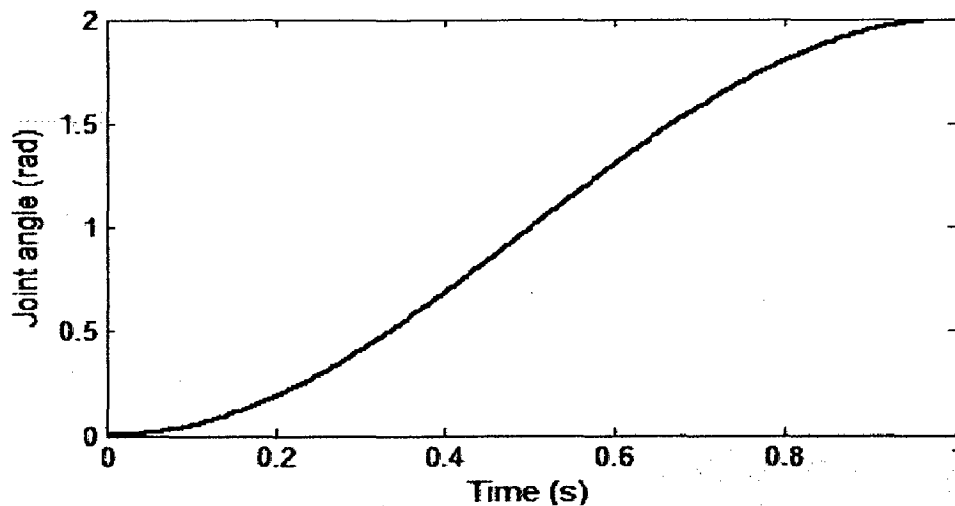


Figure 6.3 Input trajectory signal given to the individual joints of Puma 560

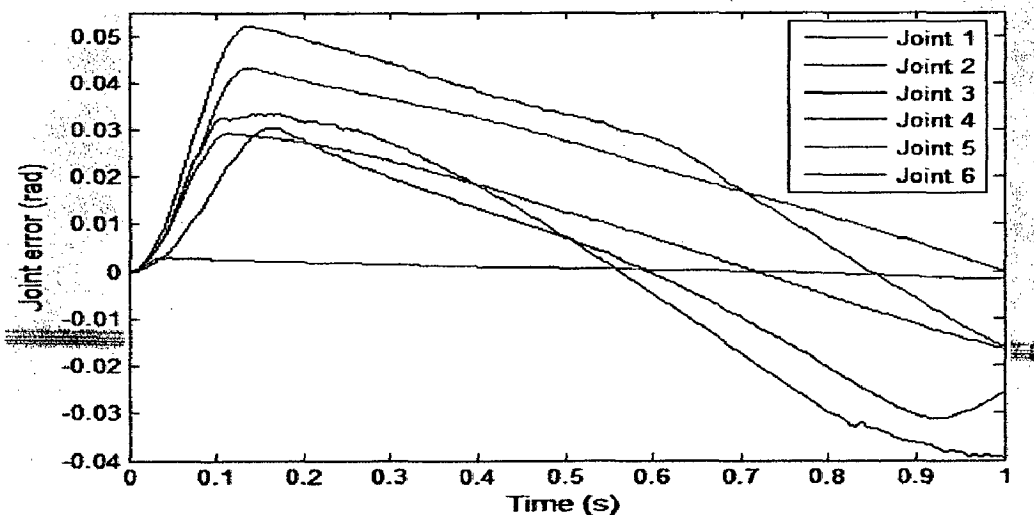


Figure 6.4 Joint error generated by the given input trajectory for reference Fuzzy PD+I controller.

6.1.1.2 Rule tuning

The result of rule tuning is presented here. The number of parameters tuned is 54 (9 rules \times 6 joints). Figure 6.5 shows the Rule base after tuning of the rules and Figure 6.6 shows the control surface view of this rule base. Figure 6.7 represents the joint error generated by the robot arm with Rule tuned Fuzzy PD+I controller.

| Joint 1 rule base | | | | Joint 2 rule base | | | |
|-------------------|-----|-----|-----|-------------------|-----|-----|-----|
| de e | mf1 | mf2 | mf3 | de e | mf1 | mf2 | mf3 |
| mf1 | mf1 | mf2 | mf3 | mf1 | mf2 | mf1 | mf3 |
| mf2 | mf1 | mf2 | mf3 | mf2 | mf1 | mf2 | mf3 |
| mf3 | mf3 | mf3 | mf3 | mf3 | mf2 | mf3 | mf2 |

| Joint 3 rule base | | | | Joint 4 rule base | | | |
|-------------------|-----|-----|-----|-------------------|-----|-----|-----|
| de e | mf1 | mf2 | mf3 | de e | mf1 | mf2 | mf3 |
| mf1 | mf3 | mf3 | mf1 | mf1 | mf2 | mf2 | mf1 |
| mf2 | mf1 | mf2 | mf3 | mf2 | mf1 | mf2 | mf3 |
| mf3 | mf2 | mf1 | mf2 | mf3 | mf3 | mf2 | mf2 |

| Joint 5 rule base | | | | Joint 6 rule base | | | |
|-------------------|-----|-----|-----|-------------------|-----|-----|-----|
| de e | mf1 | mf2 | mf3 | de e | mf1 | mf2 | mf3 |
| mf1 | mf2 | mf1 | mf1 | mf1 | mf1 | mf2 | mf2 |
| mf2 | mf2 | mf2 | mf3 | mf2 | mf1 | mf2 | mf3 |
| mf3 | mf2 | mf2 | mf2 | mf3 | mf1 | mf2 | mf3 |

Figure 6.5 Rule base after Rule tuning using Genetic Algorithm

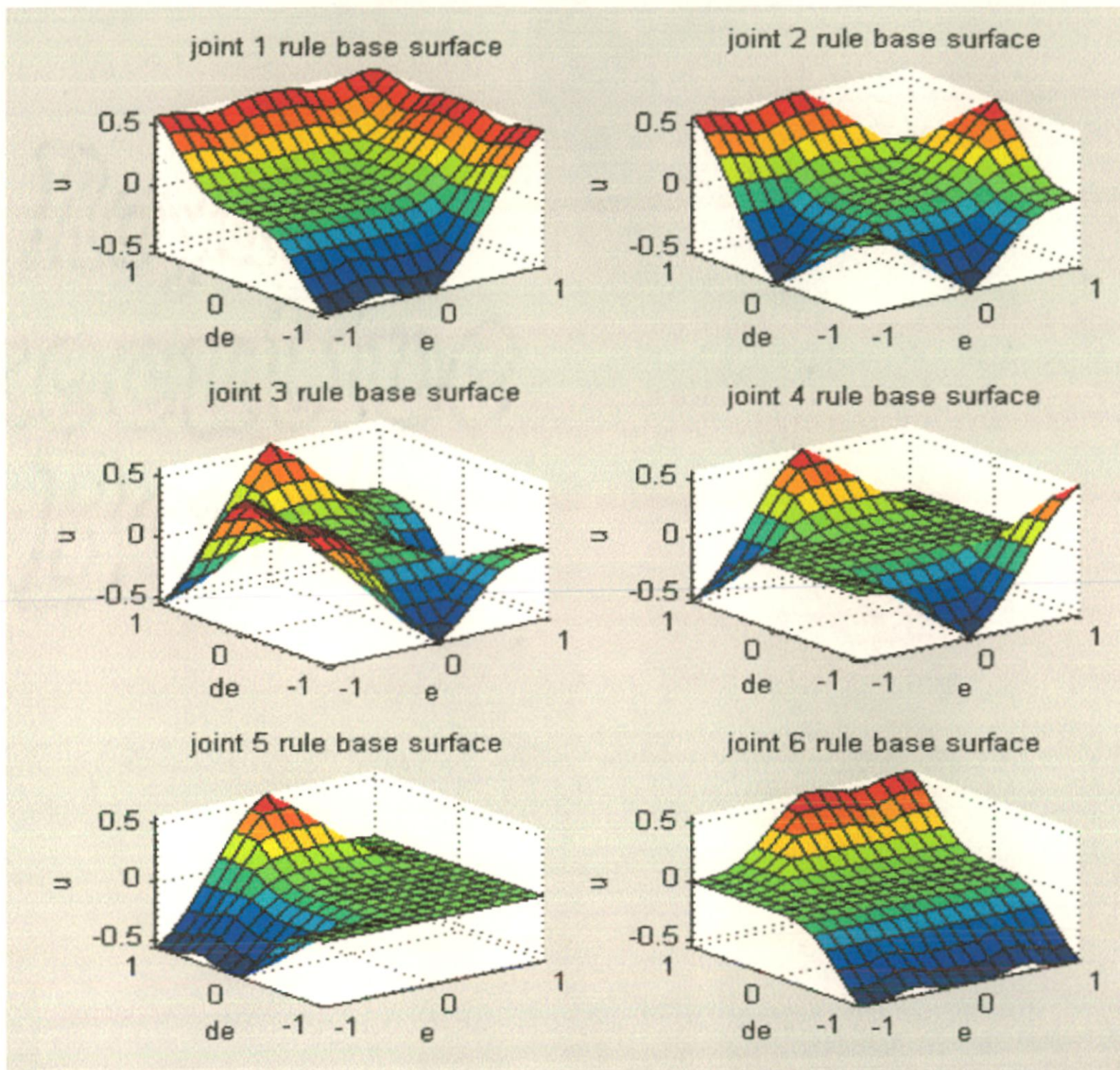


Figure 6.6 Control surfaces of the rulebase after rule tuning

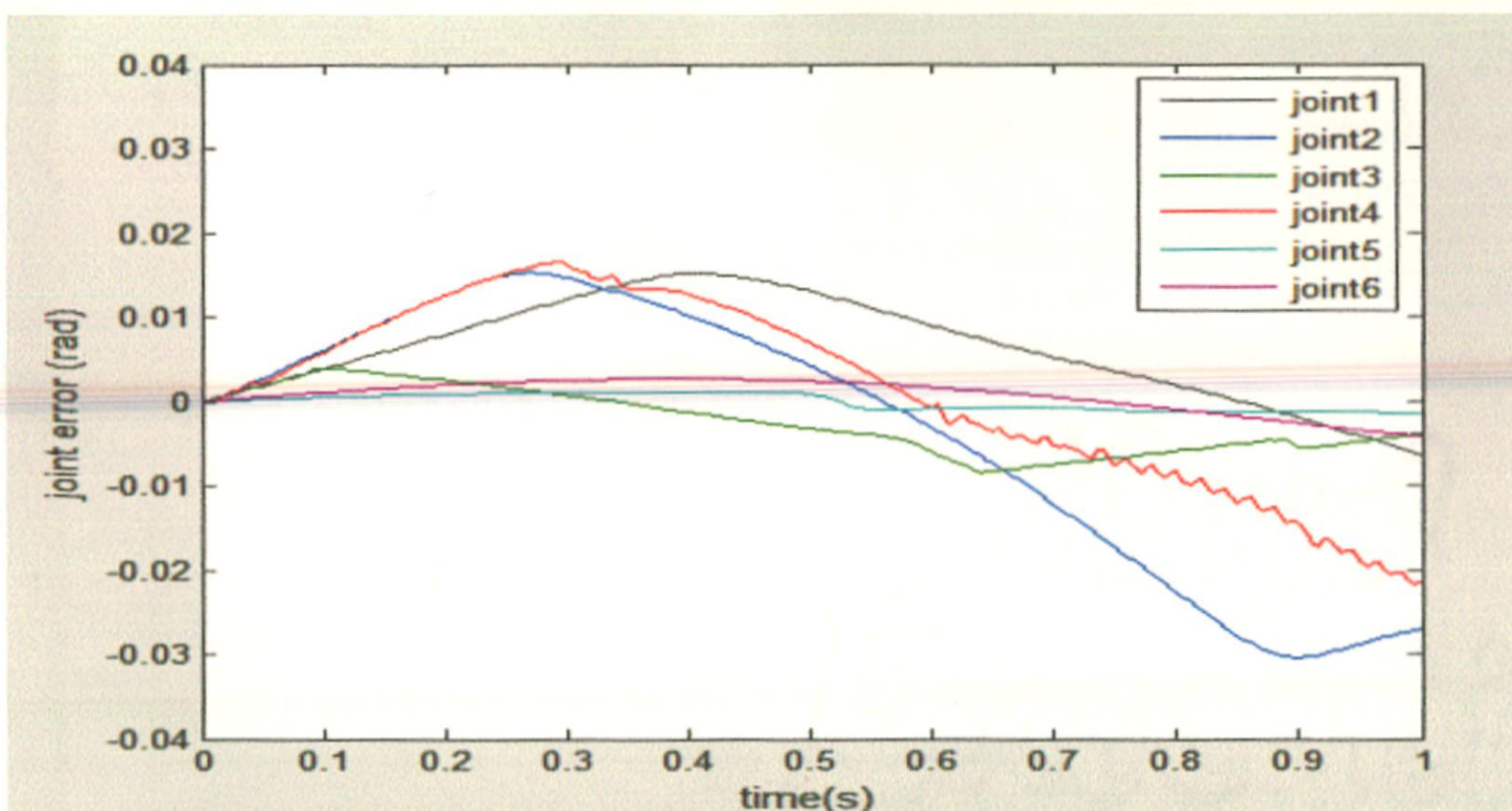


Figure 6.7 Joint error generated by the robot arm with Rule-tuned Fuzzy PD+I controller

6.1.1.3. Rule weight tuning

The result of Rule weight tuning is presented here. The weights of the rules are tuned and the number of parameters tuned is 54 (9 weights \times 6 joints). Figure 6.8 shows the weight tuned rulebase, with weights written within brackets.

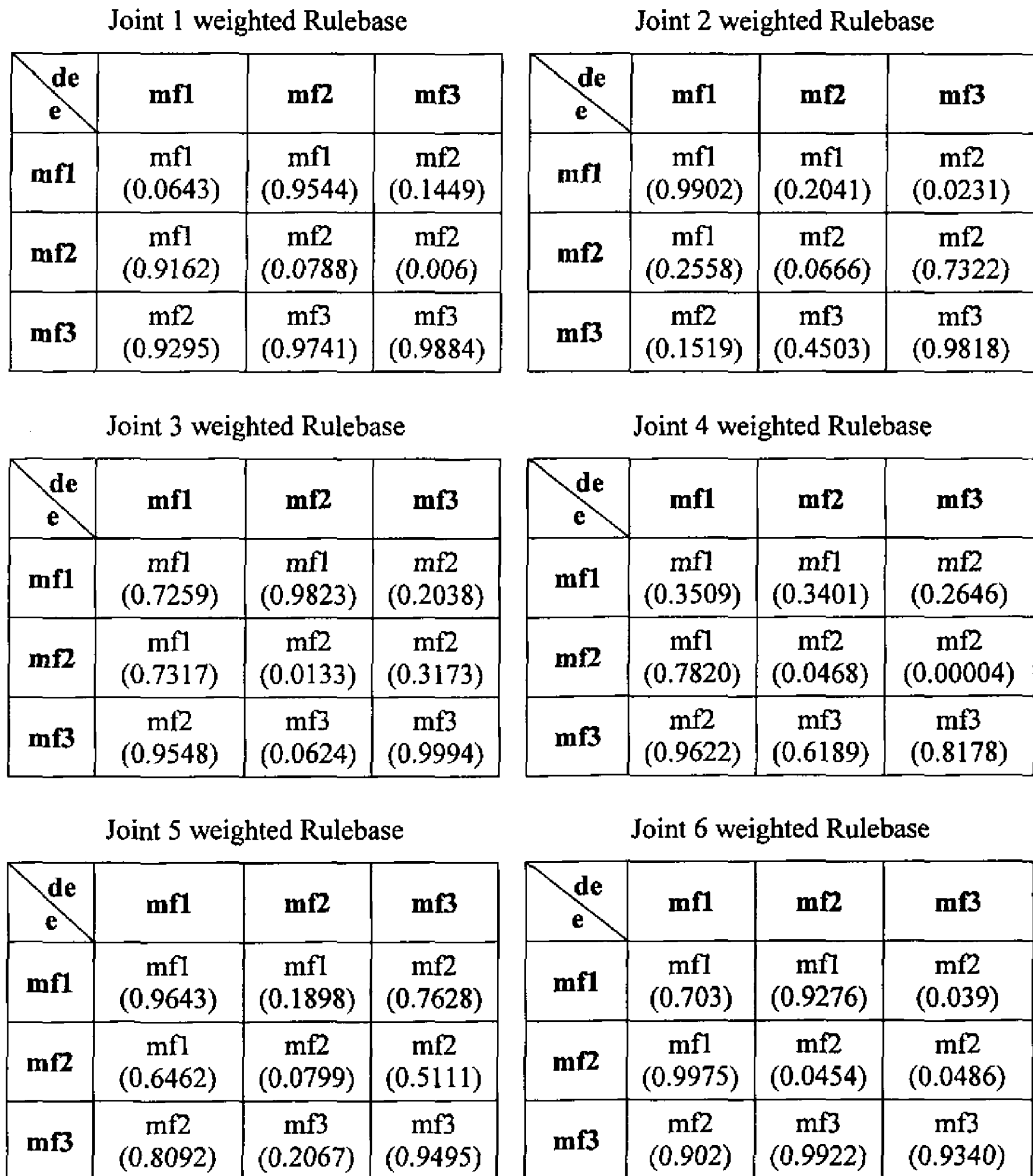


Figure 6.8 Weight tuned rulebase, with weights written within brackets

Figure 6.9 shows the control surface of weight tuned rulebase and Figure 6.10 represents the joint error generated by the robot arm with Rule-weight tuned Fuzzy PD+I controller.

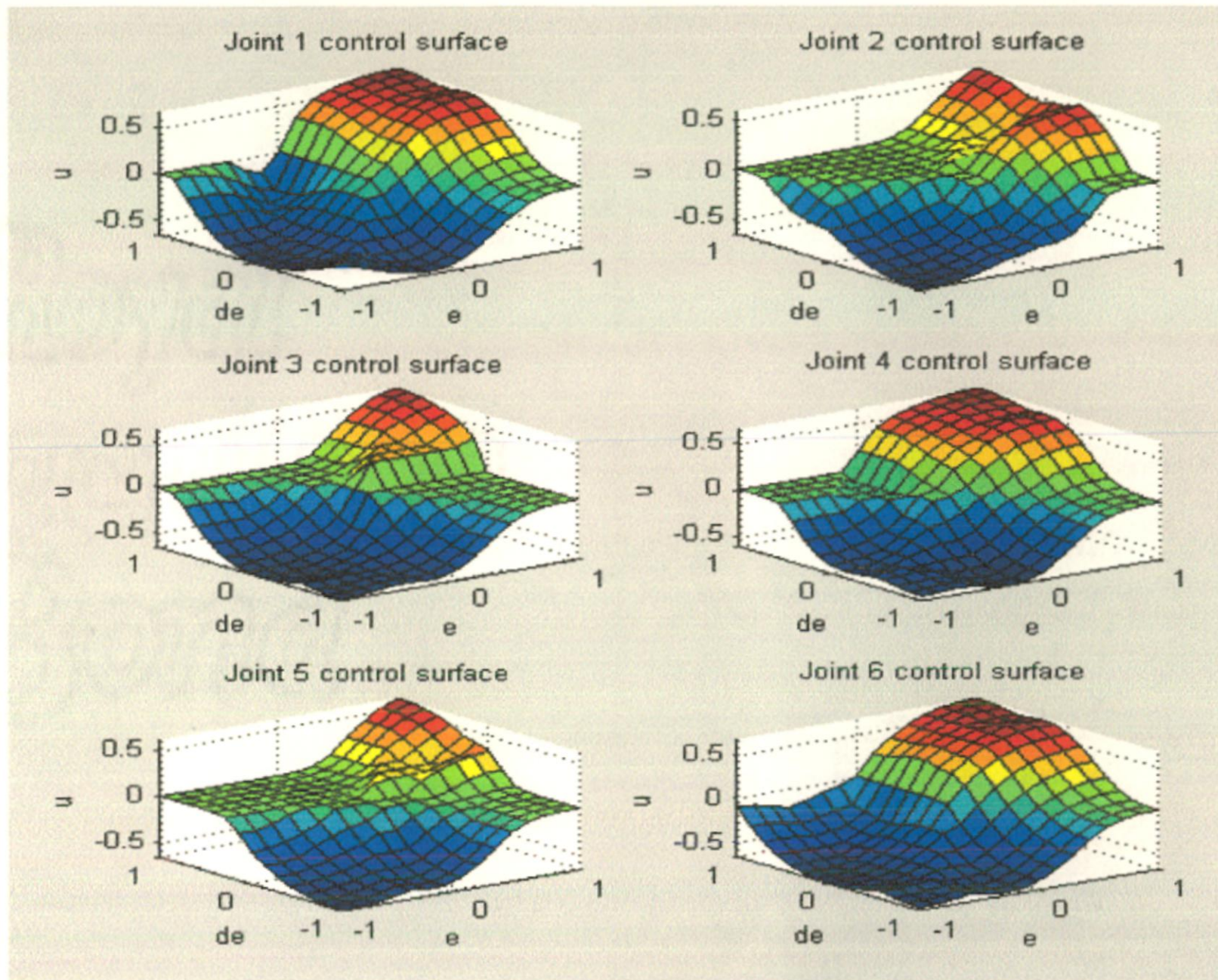


Figure 6.9 Control surface of weight tuned rulebase

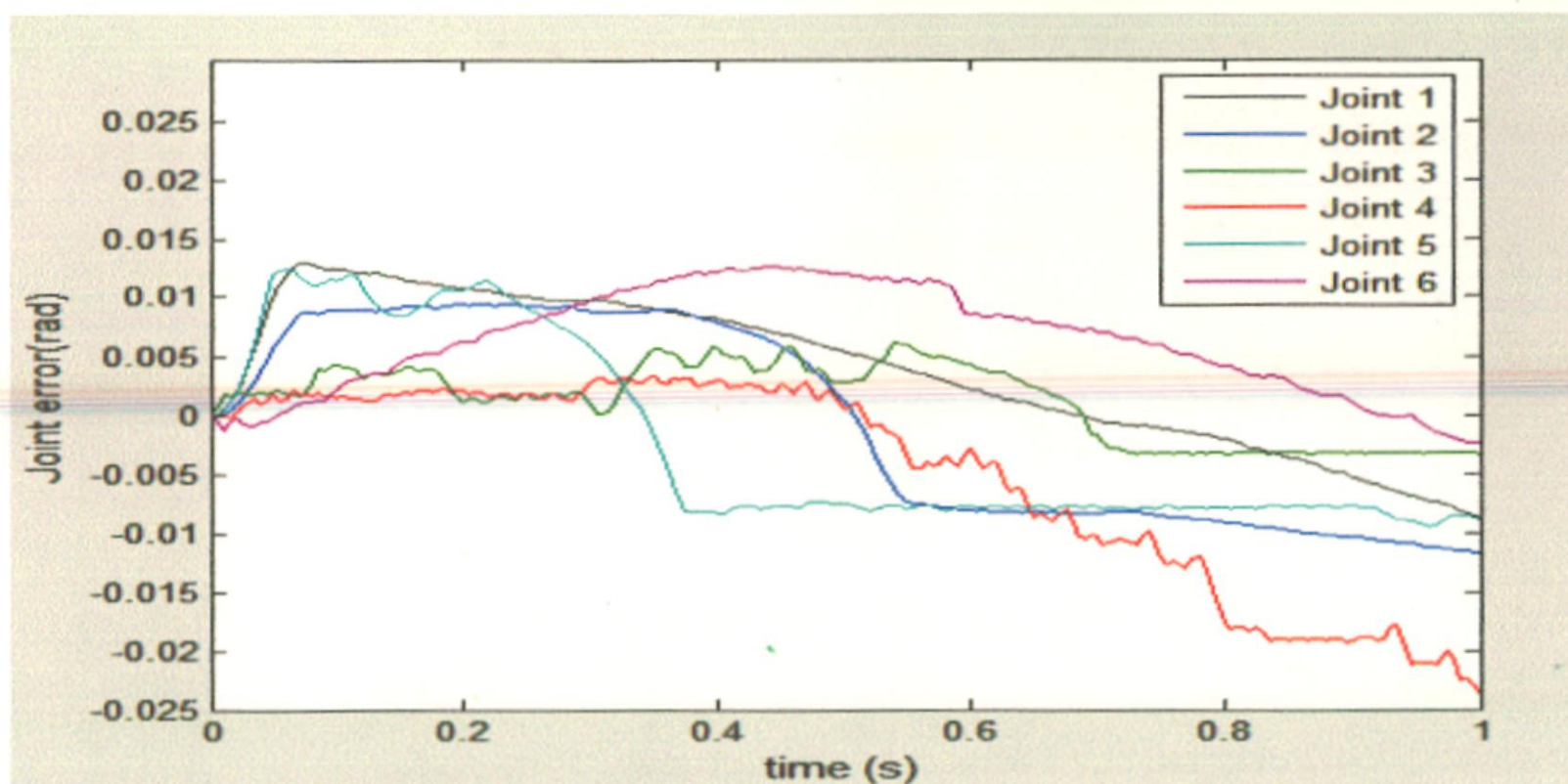


Figure 6.10 Joint error generated by the robot arm with Rule-weight tuned Fuzzy PD+I controller

6.1.1.4. Membership function tuning

The result of membership function tuning is presented here. The number of parameters tuned are 72 ((9 for ' σ ' + 3 for ' μ ') \times 6 joints). Figure 6.11 shows the Membership functions (MF) of the Fuzzy PD+I controller after MF tuning. Figure 6.12 shows the control surface after MF tuning and Figure 6.13 represent the joint error generated by the robot arm with Membership functions tuned Fuzzy PD+I controller.

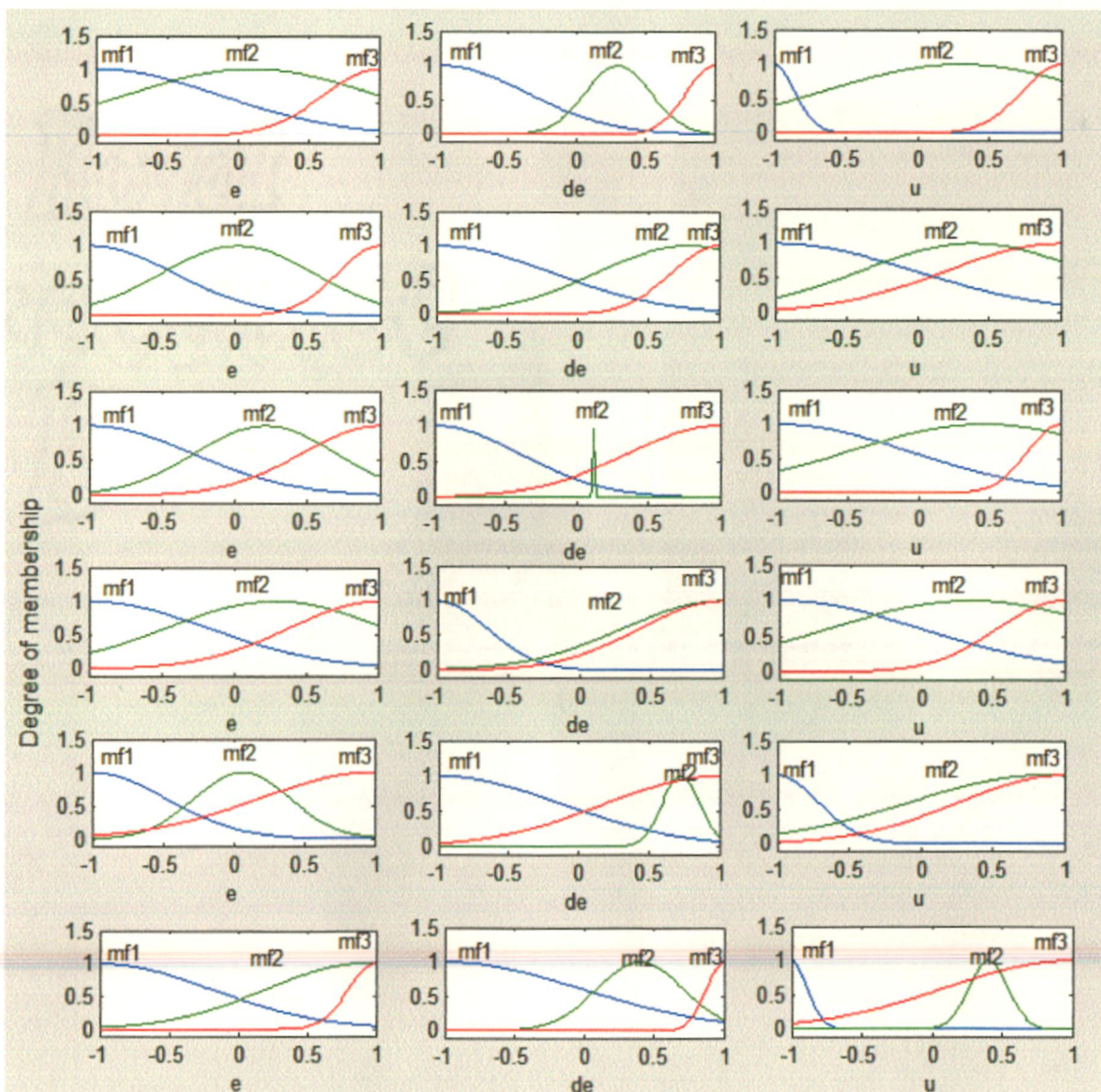


Figure 6.11 Membership functions of the Fuzzy PD+I controller after MF tuning

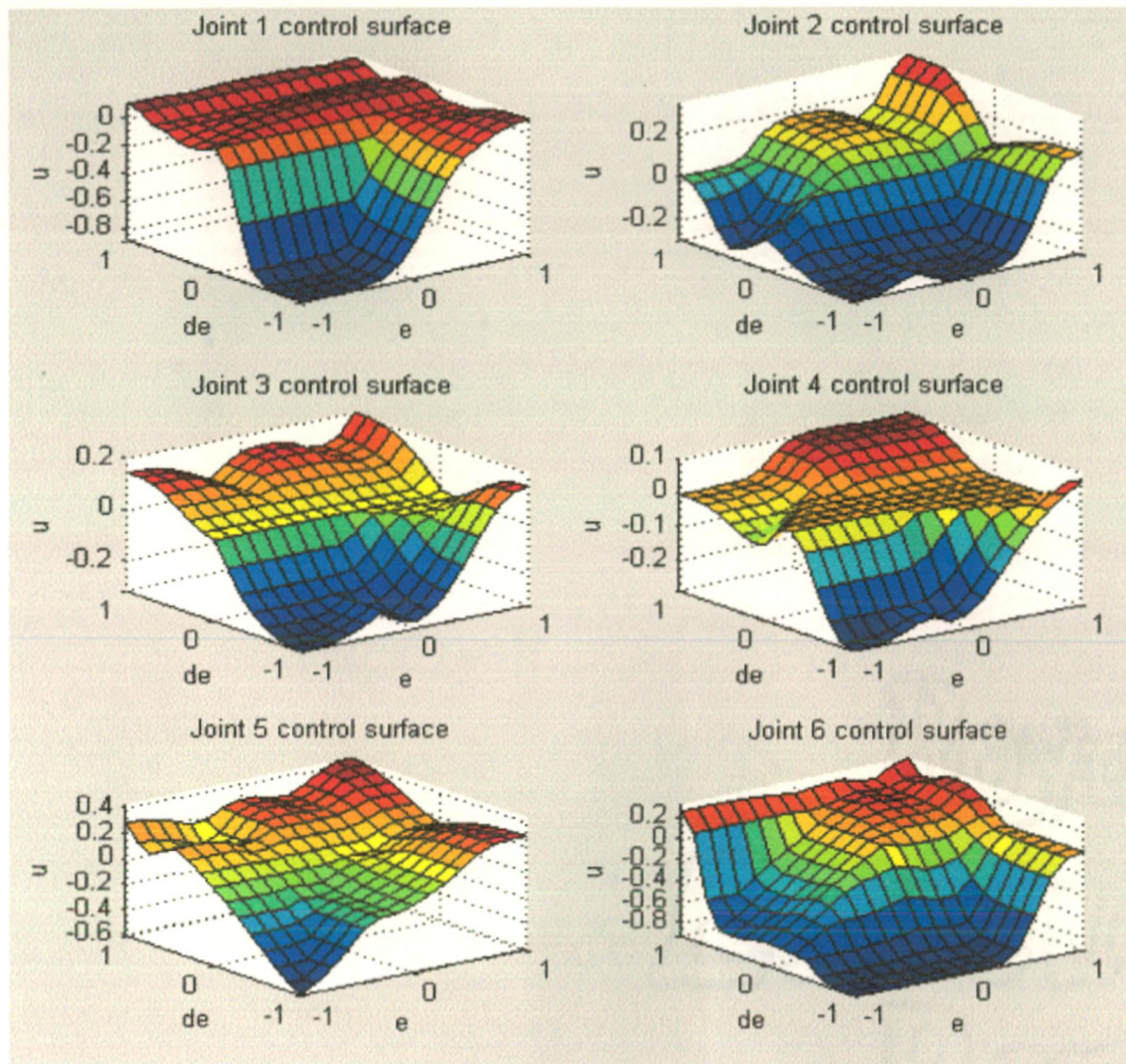


Figure 6.12 control surfaces of the Fuzzy PD+I controller after MF tuning

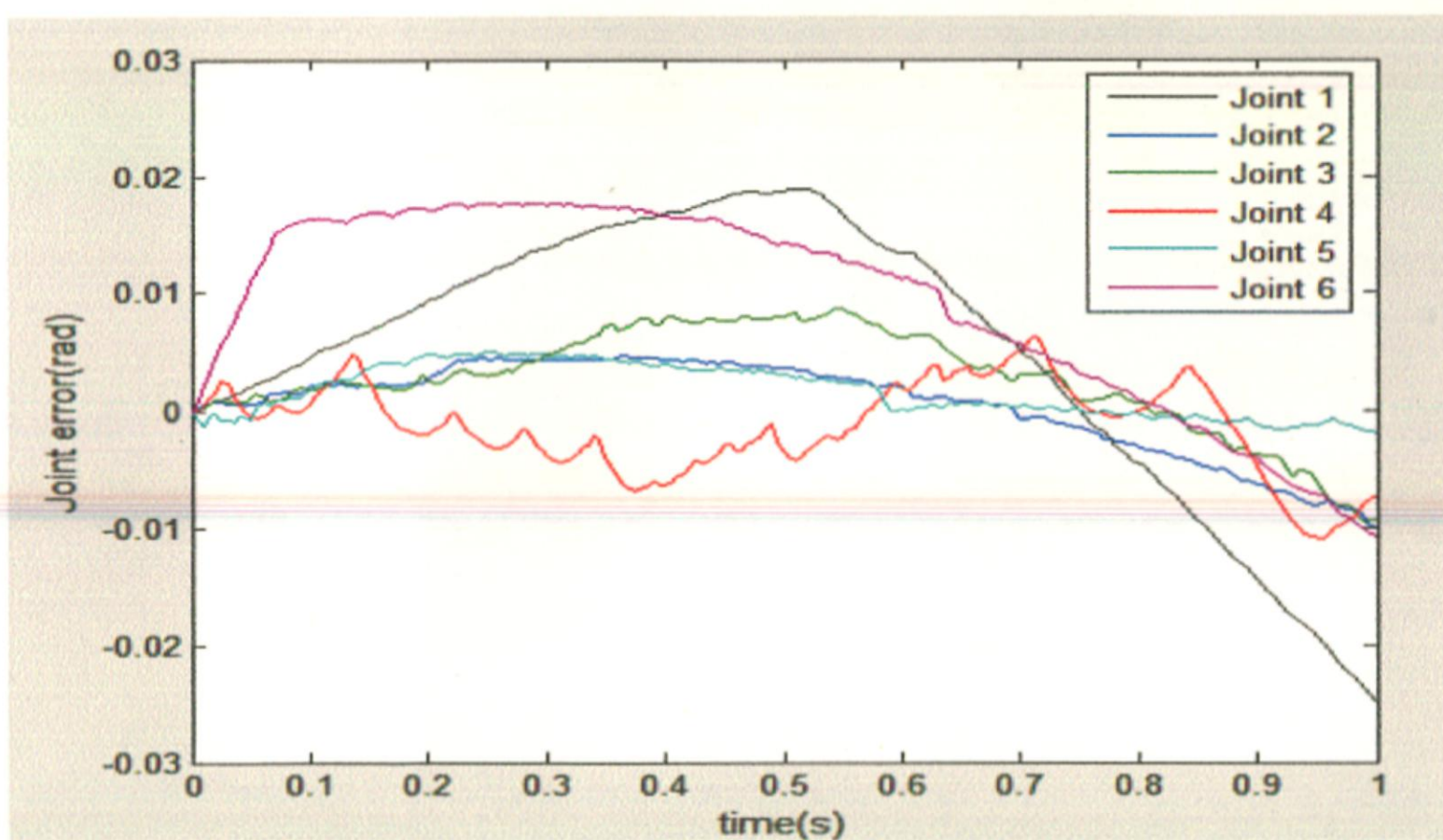


Figure 6.13 Joint error generated by the robot arm with Membership functions tuned Fuzzy PD+I controller

Table 6.1 Tabulated results of the preliminary tuning

| Type | No. of parameters tuned | Joint 1 ISE $\int e_1^2(t)dt$ | Joint 2 ISE $\int e_2^2(t)dt$ | Joint 3 ISE $\int e_3^2(t)dt$ | Joint 4 ISE $\int e_4^2(t)dt$ | Joint 5 ISE $\int e_5^2(t)dt$ | Joint 6 ISE $\int e_6^2(t)dt$ | Overall ISE $\sum_{i=1}^6 \int e_i^2(t)dt$ |
|---------------|-------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|---|
| Base system | - | 0.000691 | 0.000361 | 0.00099 | 0.000624 | 1.88E-06 | 0.000276 | 2.94E-03 |
| Rule tuning | 54 | 7.51E-05 | 0.00025 | 1.88E-05 | 0.000115 | 1.01E-06 | 4.25E-06 | 4.64E-04 |
| Weight tuning | 54 | 5.33E-05 | 7.14E-05 | 8.55E-06 | 9.53E-05 | 6.85E-05 | 4.75E-05 | 3.45E-04 |
| MF tuning | 72 | 0.000163 | 1.53E-05 | 2.46E-05 | 1.62E-05 | 3.11E-06 | 0.000147 | 3.69E-04 |

Table 6.1 gives the number of parameters tuned; break up of the ISEs of the individual joints and the overall ISE that is minimized by Genetic Algorithm. It provides results of the preliminary tuning strategies in a tabular form.

The inference which is drawn from Table 6.1 is that the weights tuning provides better results (overall ISE) than others for this system.

6.1.2 Two stage tuning

The result of the section 5.1.2 is presented here. As a first step of progressively tuning the fuzzy controller, rule tuning is carried out. The results of rule tuning of section 6.1.1.2 are taken as the stage I results and the proceeded further with the stage II. The number of generations of optimization in genetic algorithm is 500 for the stage I (maximum generation in 6.1.1.2). After the rule tuning, the weights of the rules are tuned in the second stage. The number of generations of optimization in genetic algorithm is 1000. The number of generation is so chosen such that weight tuning is done more thoroughly. The number of parameters tuned is 54 (9 weights \times 6 joints). Figure 6.14 represents the joint error generated by the robot arm with Rule-weight tuned Fuzzy PD+I controller, Figure 6.15 shows the weight tuned rulebase, with weights written within brackets and Figure 6.16 shows the control surface of weight tuned rulebase.

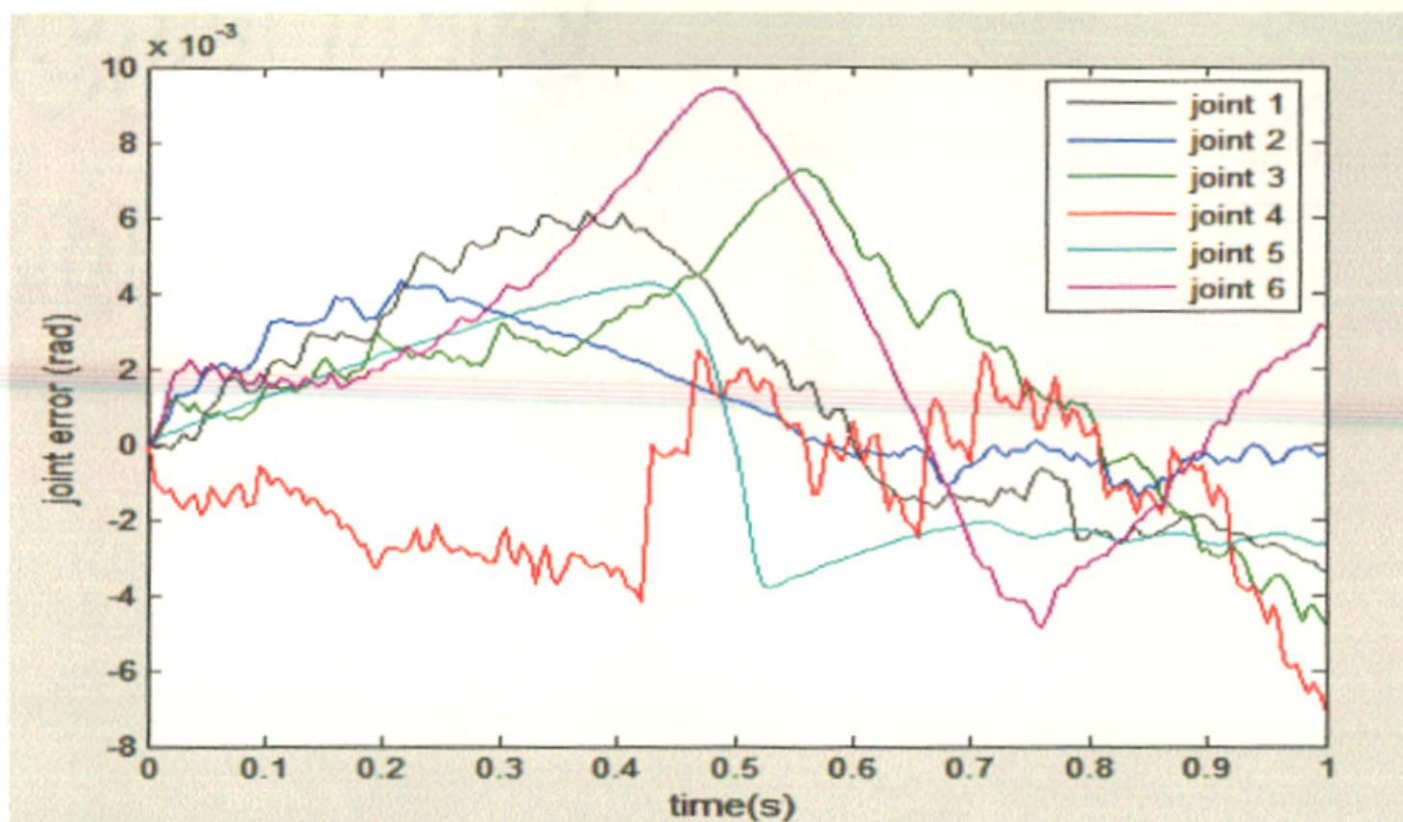


Figure-6.14 Joint error generated by the robot arm after stage II tuning

Joint 1 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|-----------------------|-----------------|-----------------|-----------------|
| mf1 | mf1 (0.0306) | mf2 (0.9316) | mf3 (0.1008) |
| mf2 | mf1 (0.8930) | mf2 (0.0443) | mf3 (0.1213) |
| mf3 | mf3 (0.0289) | mf3 (0.9404) | mf3 (0.0100) |

Joint 2 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|-----------------------|-----------------|-----------------|-----------------|
| mf1 | mf2 (0.5867) | mf1 (0.9734) | mf3 (0.0853) |
| mf2 | mf1 (0.9697) | mf2 (0.0270) | mf3 (0.9136) |
| mf3 | mf2 (0.8997) | mf3 (0.9754) | mf2 (0.7711) |

Joint 3 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|-----------------------|-----------------|-----------------|-----------------|
| mf1 | mf3 (0.3153) | mf3 (0.0281) | mf1 (0.2572) |
| mf2 | mf1 (0.9398) | mf2 (0.1810) | mf3 (0.9393) |
| mf3 | mf2 (0.2207) | mf1 (0.1864) | mf2 (0.8373) |

Joint 4 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|-----------------------|-----------------|-----------------|-----------------|
| mf1 | mf2 (0.0796) | mf2 (0.8884) | mf1 (0.9974) |
| mf2 | mf1 (0.7155) | mf2 (0.0229) | mf3 (0.9551) |
| mf3 | mf3 (0.7841) | mf2 (0.9899) | mf2 (0.3016) |

Joint 5 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|-----------------------|-----------------|-----------------|-----------------|
| mf1 | mf2 (0.7691) | mf1 (0.9942) | mf1 (0.2588) |
| mf2 | mf2 (0.009) | mf2 (0.9882) | mf3 (0.9936) |
| mf3 | mf2 (0.8763) | mf2 (0.9907) | mf2 (0.9653) |

Joint 6 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|-----------------------|-----------------|-----------------|-----------------|
| mf1 | mf1 (0.9584) | mf2 (0.0570) | mf2 (0.0353) |
| mf2 | mf1 (0.9719) | mf2 (0.2378) | mf3 (0.9400) |
| mf3 | mf1 (0.9382) | mf2 (0.7952) | mf3 (0.9465) |

Figure 6.15 Weighted-rule rulebase after Stage II tuning

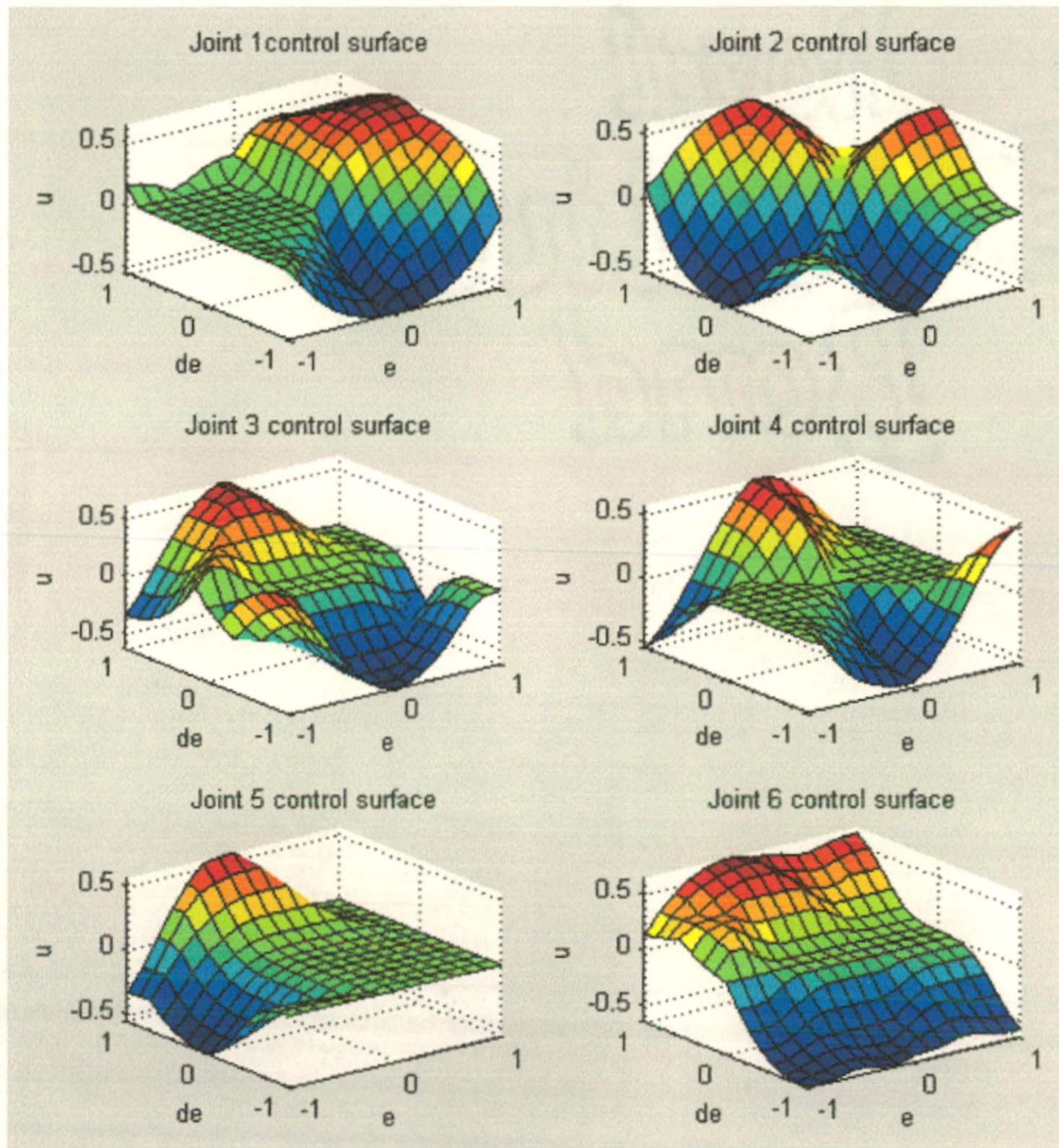


Figure 6.16 Weighted-rule control surface after Stage II tuning

Rule tuning followed by rule-weight tuning using Genetic Algorithm is successfully applied to the Fuzzy PD+I controller of the PUMA560. It is seen from the plots of Figure 6.4 and Figure 6.7 that tuning the rules of rule base results in a good performance. Further from Figure 6.14 it is evident that tuning the weights of rule tuned rulebase makes fine tuning of the fuzzy controller possible. The control surface obtained after the Stage II is smoother than that of Stage I, observed from Figure 6.6 and Figure 6.16. With this two-stage approach better and finer tuning of fuzzy controllers can be implemented even for complex coupled systems.

6.1.3 Three stage tuning

The result of the section 5.1.3 is discussed here. After the rules and their weights are tuned in two stages above, the tuned rulebase is taken and tuning is extended in stage 3 by tuning the membership functions. The number of generations of optimization in genetic algorithm is 500 for this case. The number of parameters tuned are 72 ((9 for ' σ ' + 3 for ' μ ') \times 6 joints). Figure 6.17 shows the Membership functions (MF) of the Fuzzy PD+I controller after Stage III, top row denoting joint 1 MF, next row for joint 2 MF, and so on. Figure-6.18 shows the control surface after Stage III. Figure-6.19 shows the joint error plot of the Fuzzy PD+I controller after Stage III.

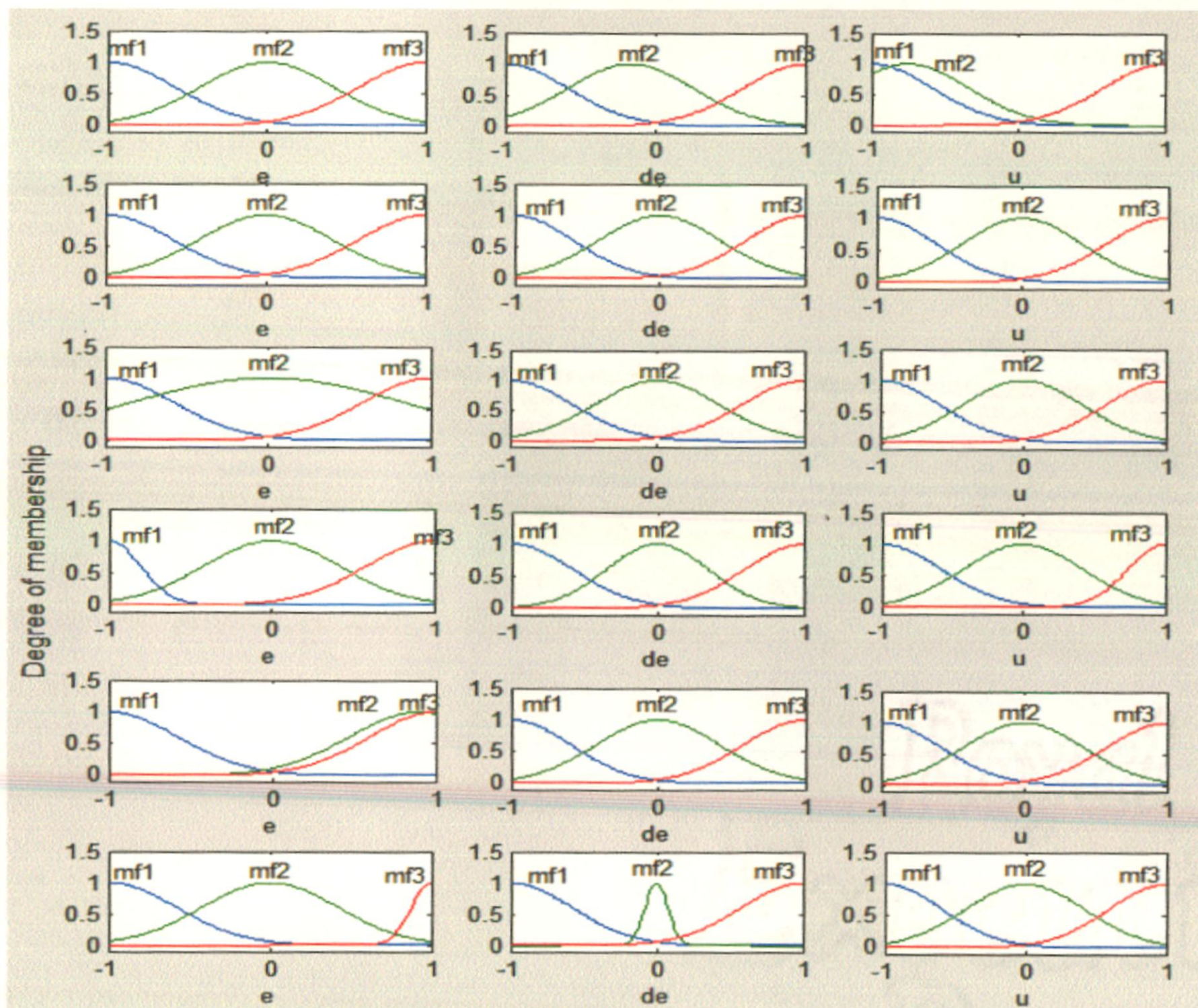


Figure 6.17 Membership functions (MF) of the Fuzzy PD+I controller after Stage III

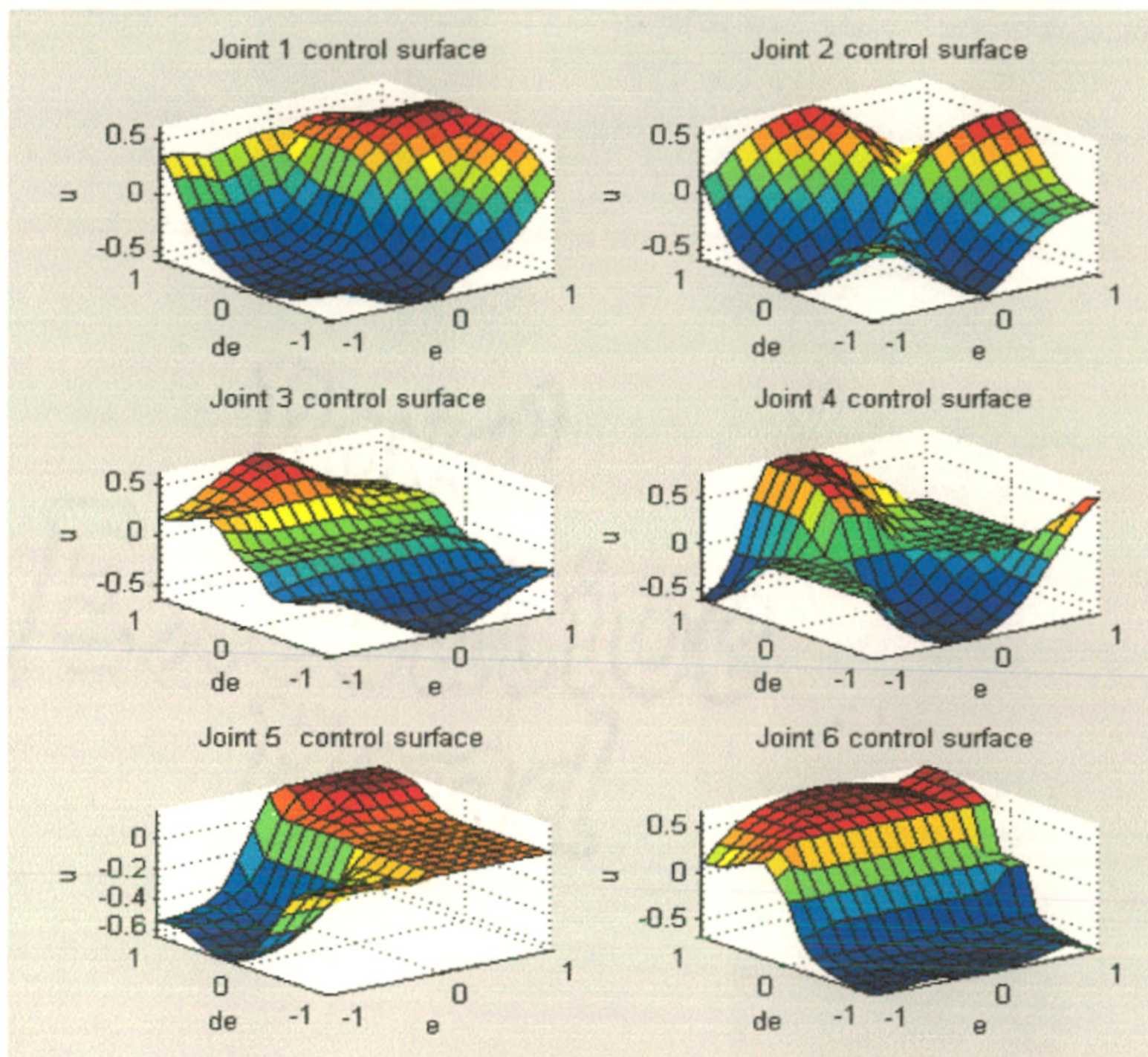


Figure-6.18 Control surface of the Fuzzy PD+I controller after Stage III

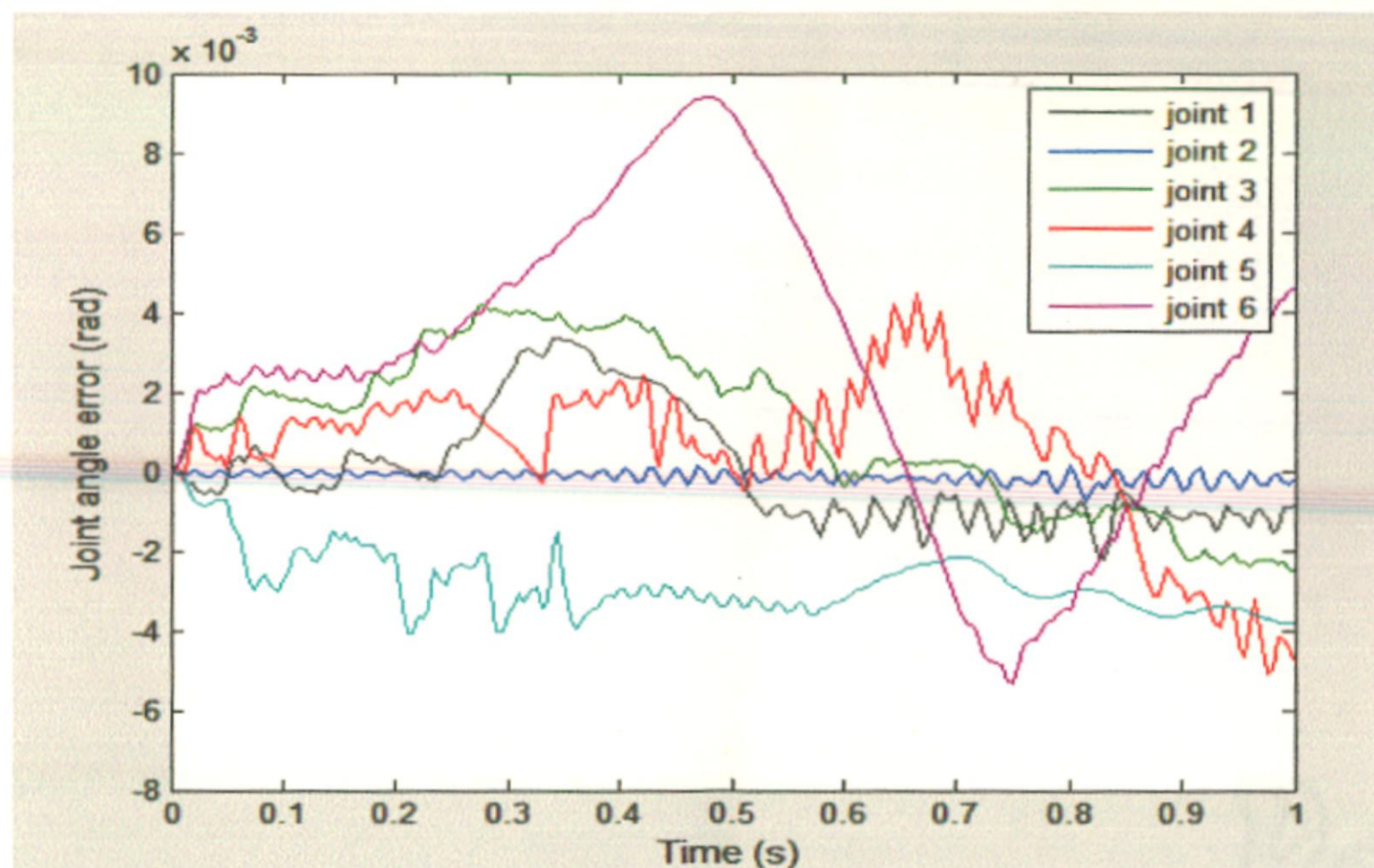


Figure-6.19 Joint error plot of the Fuzzy PD+I controller after Stage III.

6.1.4 Simultaneous tuning

The result of the section 5.1.4 is presented here. In this section, Rules, weights and MFs of the reference Fuzzy PD+I controller are tuned simultaneously. The number of generations of optimization in genetic algorithm is 4000 for this case. The number of parameters tuned is 180 ((9 rules + 9 weights + (9 for ' σ ' + 3 for ' μ ')) \times 6 joints). Figure 6.20 represents the joint error generated in simultaneous tuning. Figure 6.21 shows the weighted rulebase, with weights written within brackets. Figure 6.22 shows the control surface. Figure 6.23 shows the Membership functions (MF) generated in simultaneous tuning, top row denoting joint 1 MF, next row for joint 2 MF, and so on.

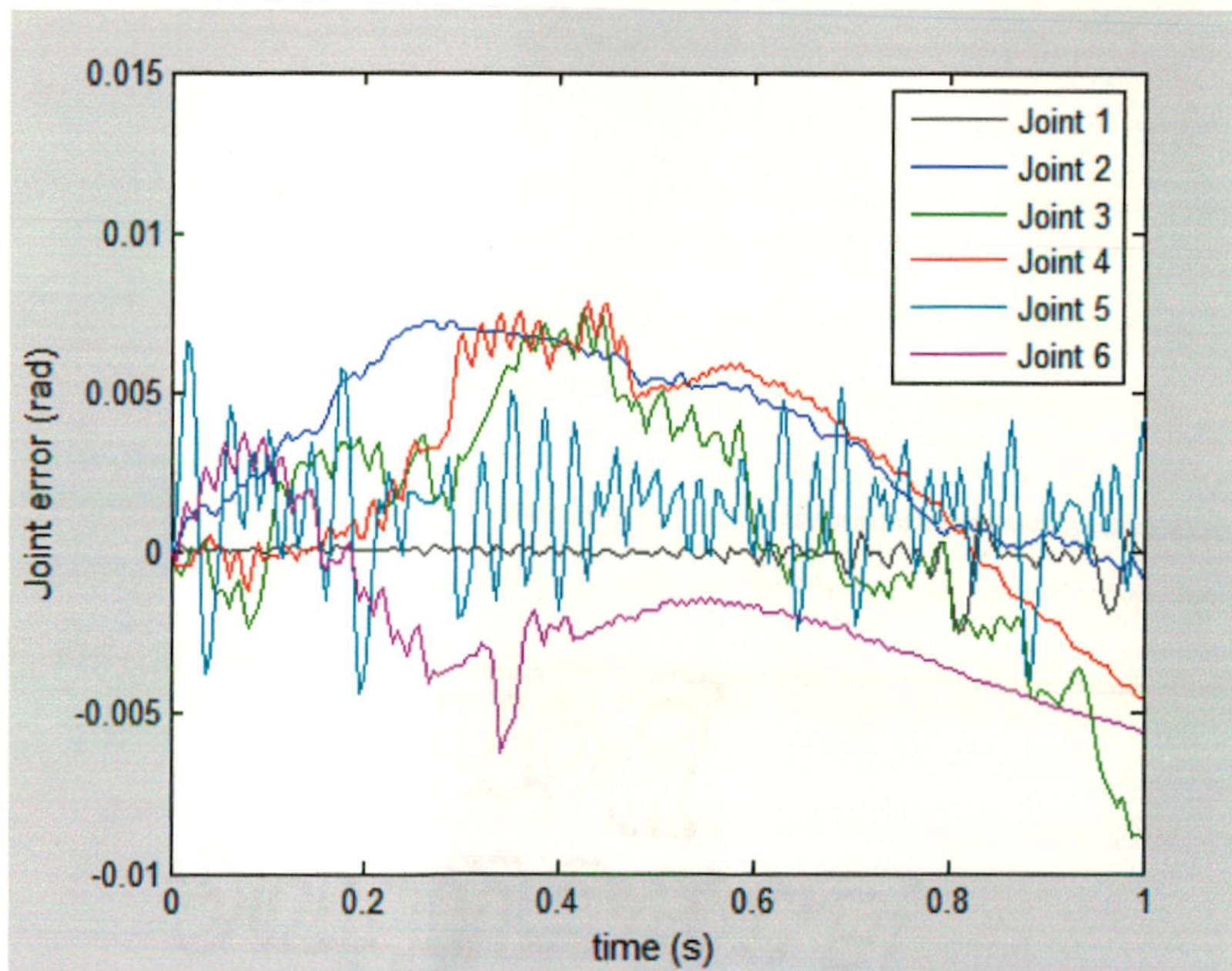


Figure 6.20 Joint error of the Fuzzy PD+I controller after Simultaneous tuning.

Joint 1 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|---------|-----------------|-----------------|-----------------|
| mf1 | mf2 (0.9898) | mf1 (0.9842) | mf2 (0.1759) |
| mf2 | mf1 (0.9532) | mf2 (0.0309) | mf3 (0.8517) |
| mf3 | mf2 (0.7509) | mf3 (0.8802) | mf3 (0.9341) |

Joint 2 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|---------|-----------------|-----------------|-----------------|
| mf1 | mf2 (0.8871) | mf2 (0.776) | mf2 (0.8782) |
| mf2 | mf1 (0.1292) | mf3 (0.0525) | mf3 (0.9305) |
| mf3 | mf2 (0.8273) | mf3 (0.8607) | mf3 (0.9134) |

Joint 3 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|---------|-----------------|-----------------|-----------------|
| mf1 | mf1 (0.9996) | mf1 (0.9404) | mf2 (0.9771) |
| mf2 | mf1 (0.2116) | mf2 (0.0915) | mf1 (0.1318) |
| mf3 | mf2 (0.6531) | mf1 (0.8848) | mf2 (0.9321) |

Joint 4 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|---------|-----------------|-----------------|-----------------|
| mf1 | mf2 (0.2602) | mf1 (0.6198) | mf1 (0.9148) |
| mf2 | mf1 (0.9816) | mf2 (0.9540) | mf2 (0.9483) |
| mf3 | mf2 (0.9354) | mf3 (0.9065) | mf3 (0.9273) |

Joint 5 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|---------|-----------------|-----------------|-----------------|
| mf1 | mf2 (0.9843) | mf1 (0.9506) | mf3 (0.9409) |
| mf2 | mf1 (0.6940) | mf3 (0.9444) | mf1 (0.5144) |
| mf3 | mf2 (0.9878) | mf1 (0.8708) | mf3 (0.9312) |

Joint 6 weighted Rulebase

| de e | mf1 | mf2 | mf3 |
|---------|-----------------|-----------------|-----------------|
| mf1 | mf1 (0.1206) | mf2 (0.9719) | mf2 (0.8943) |
| mf2 | mf1 (0.9782) | mf2 (0.6448) | mf1 (0.7751) |
| mf3 | mf2 (0.9326) | mf3 (0.9129) | mf2 (0.2038) |

Figure 6.21 Weighted-rule rulebase after simultaneous tuning

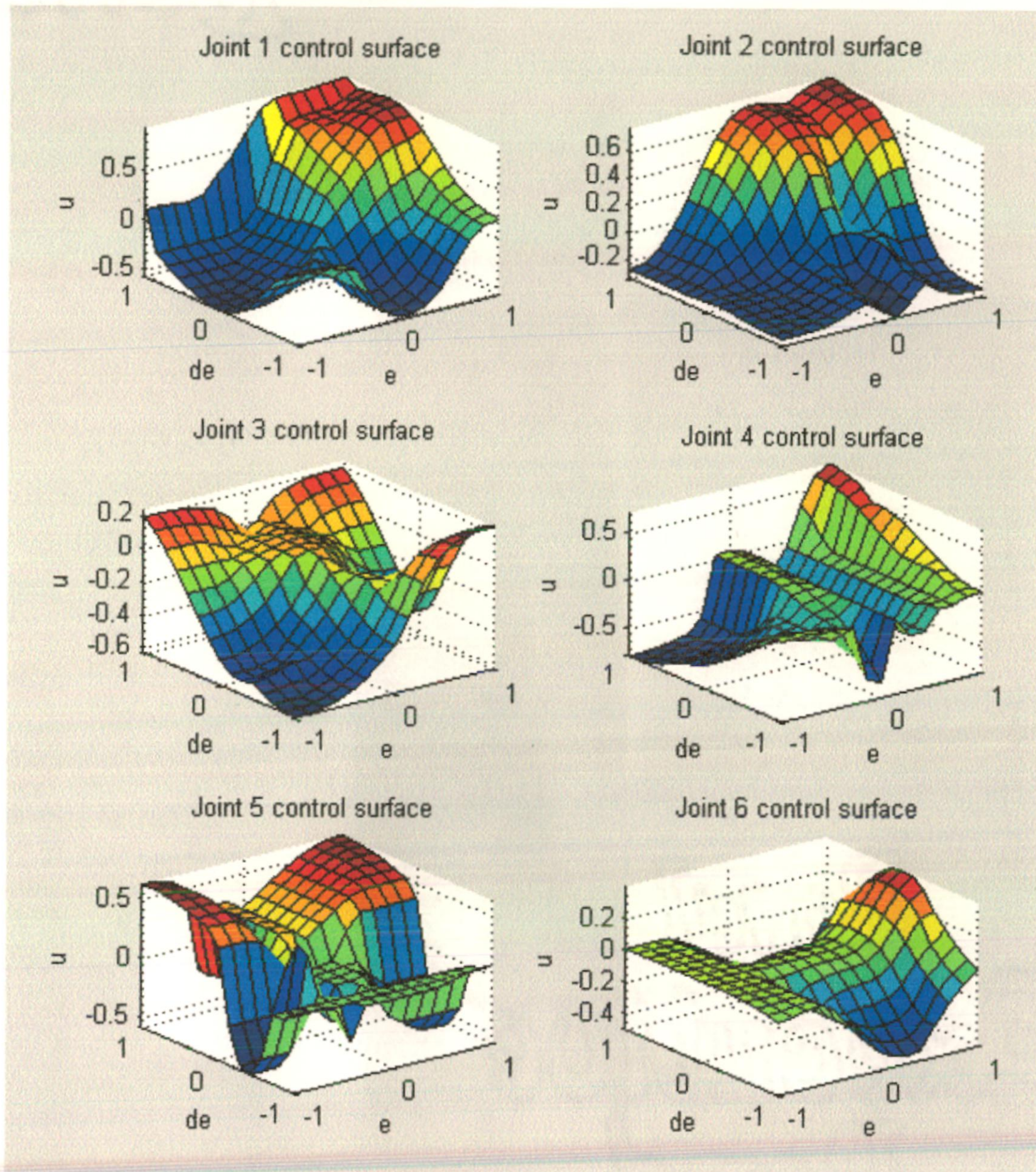


Figure 6.22 Control surfaces of the Fuzzy PD+I controller after simultaneous tuning

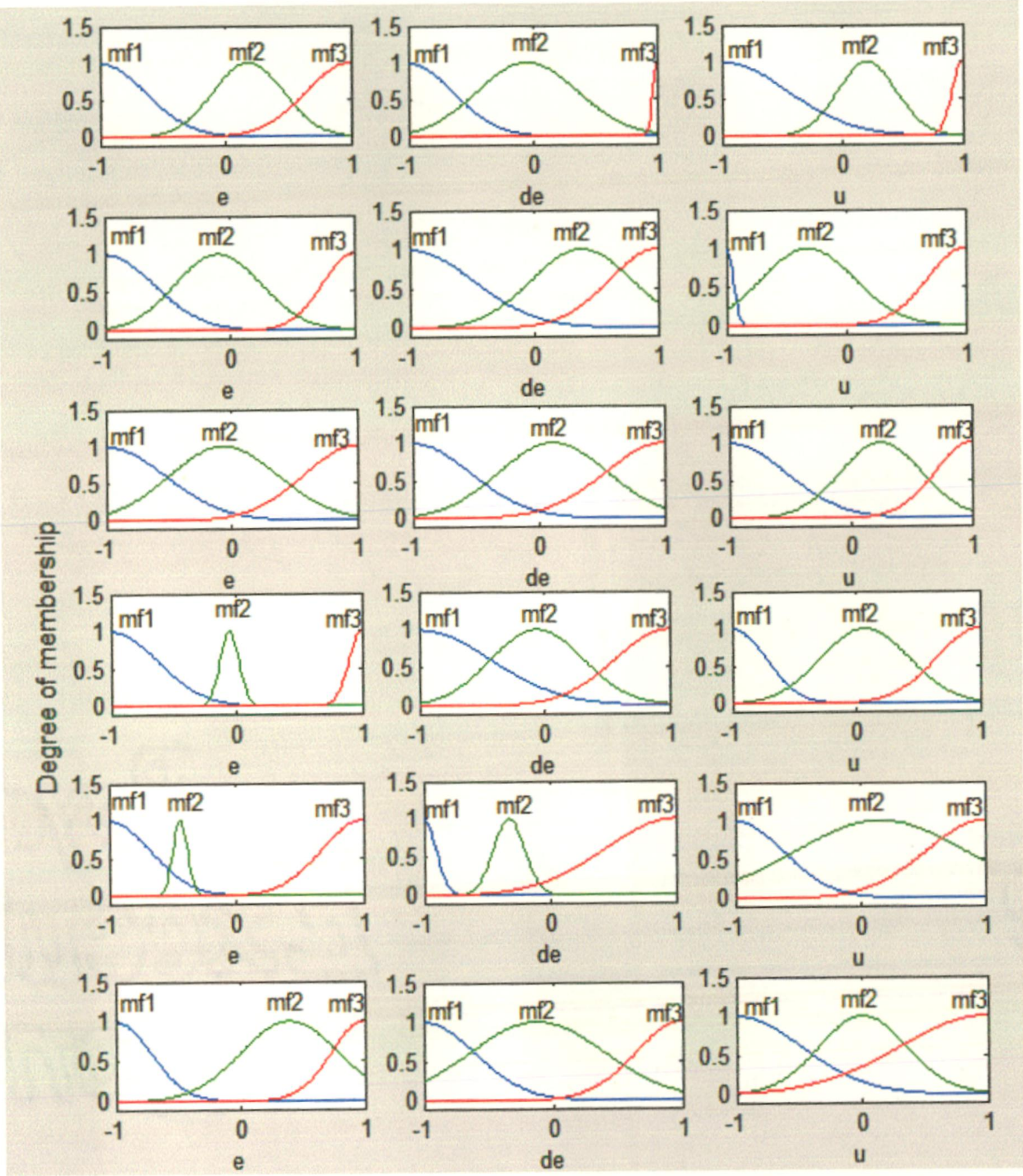


Figure 6.23 Membership functions (MF) after simultaneous tuning

Table 6.2 provides the details of the number of parameters tuned, break up of individual joint errors, and the overall ISE after optimization by GA for stage wise and simultaneous tuning strategies.

Table 6.2 Tabulated results of the stage wise and simultaneous tuning

| Type | No. of parameters tuned | Joint 1 ISE $\int e_1^2(t)dt$ | Joint 2 ISE $\int e_2^2(t)dt$ | Joint 3 ISE $\int e_3^2(t)dt$ | Joint 4 ISE $\int e_4^2(t)dt$ | Joint 5 ISE $\int e_5^2(t)dt$ | Joint 6 ISE $\int e_6^2(t)dt$ | Overall ISE $\sum_{i=1}^6 \int e_i^2(t)dt$ |
|-----------------------------------|-------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|---|
| Base system | - | 0.000691 | 0.000361 | 0.00099 | 0.000624 | 1.88E-06 | 0.000276 | 2.94E-03 |
| Stage I : Rule tuning | 54 | 7.51E-05 | 0.00025 | 1.88E-05 | 0.000115 | 1.01E-06 | 4.25E-06 | 4.64E-04 |
| Stage II : Rule-Weight tuning | 54-54 | 1.07E-05 | 3.99E-06 | 9.91E-06 | 3.62E-07 | 7.28E-06 | 1.58E-05 | 4.80E-05 |
| Stage III : Rule-Weight-MF tuning | 54-54-72 | 1.84E-06 | 2.70E-08 | 3.75E-06 | 1.50E-06 | 1.75E-06 | 1.84E-05 | 2.73E-05 |
| Simultaneous tuning | 180 | 1.04E-07 | 1.99E-05 | 1.13E-05 | 7.38E-06 | 3.07E-06 | 1.48E-06 | 4.32E-05 |

A comparative study of two different parameter tuning methods of the Fuzzy Controller is carried out. In the first method Fuzzy controller tuning in three stages is carried out and in the second method simultaneous tuning of all the mentioned parameters is done. Table 6.2 provides a comparison of the two strategies mentioned in terms of the performance criterion. The performance used for evaluating the performance is according to the Equation (6.1) which is presented in the last column of the Table 6.2. It can be seen from the table that progressively tuning the fuzzy controller for the control of PUMA 560 has lesser overall ISE than simultaneously tuning procedure. The total number of generations of optimization required to obtain the result in case of the progressive tuning is 2000 generations (500 for rules + 1000 for weights + 500 for membership functions), Which is half of that for simultaneous tuning. The main problem with simultaneous tuning procedure is that the number of parameters in use (180 parameters). To find an optimum solution with 180 parameters is searching a space of dimension 180 to find one optimum solution. Due to this the solution gets stuck in a local minimum. On the other hand in the progressive tuning we take only one set of parameters at a time for tuning and then proceed with the next. The problem that could arise in this situation is that we might end up with a sub-optimal solution which tends to get oriented towards the solution of the first stage. This is clearly seen in control surface plots in the Figure 6.6, Figure 6.16 and Figure-6.18.

6.2 Results for Approximations of systems

The results of two approximation methods discussed, namely Bi-variate Polynomial approximation and Weighted Fuzzy approximation are presented here. Approximation of the reference Fuzzy PD+I controller of section 6.1.1.1 is performed using the input-output data points. Both the methods are compared at the end of this section.

6.2.1 Bivariate polynomial approximation

The result of bivariate polynomial approximation is presented here. The bi-variate polynomial chosen for the approximation of fuzzy control surface is given as $(x+y+1)^{10}$, with all the coefficients being replaced by variables that can get modified by genetic algorithm so that an optimal value of these coefficients can be found. Figure 6.24 shows

the surface plot of the bi-variate polynomials generated. Figure 6.25 shows the joint error by the both bivariate polynomial approximated controller and the reference Fuzzy PD+I controller.

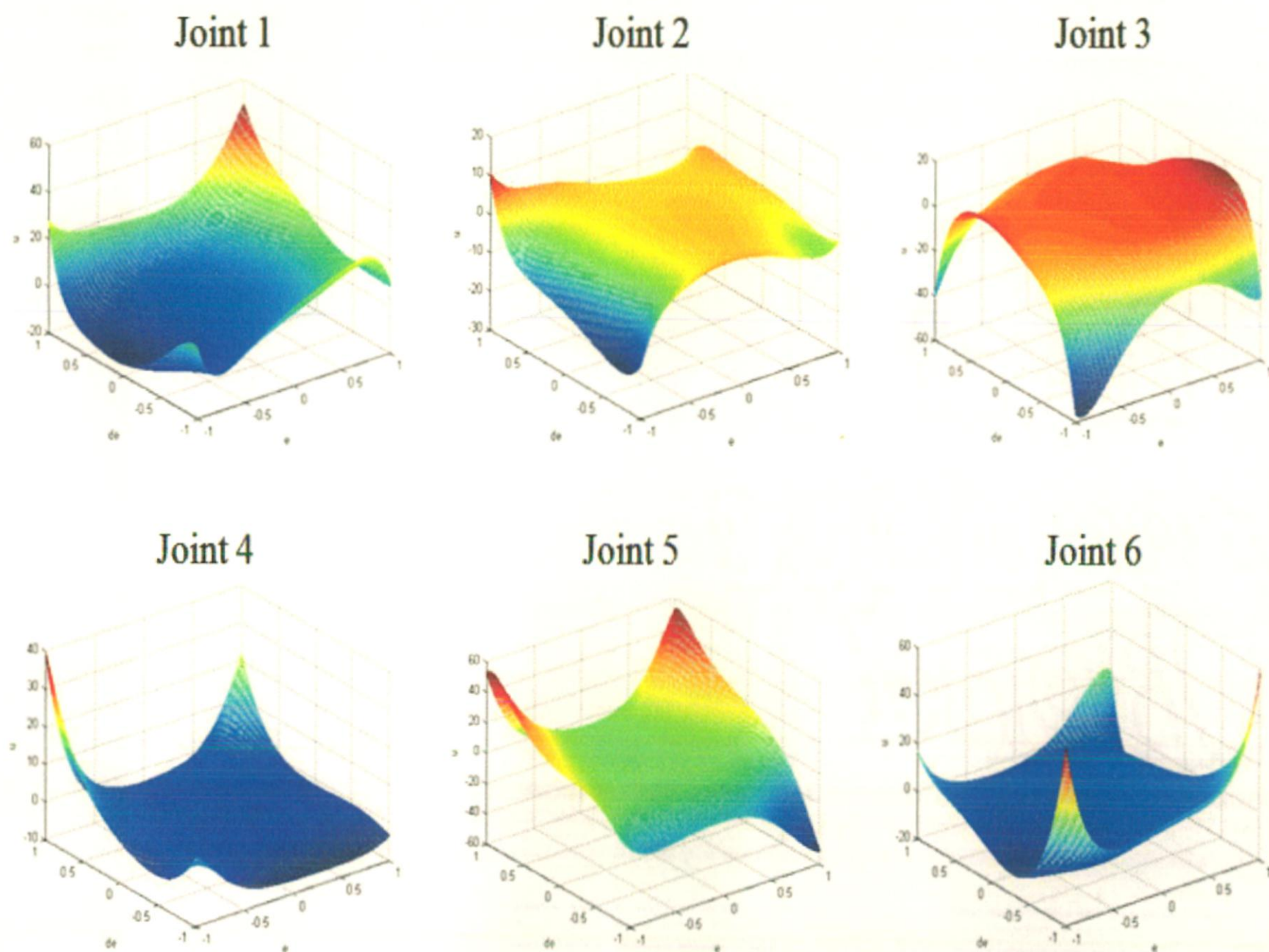
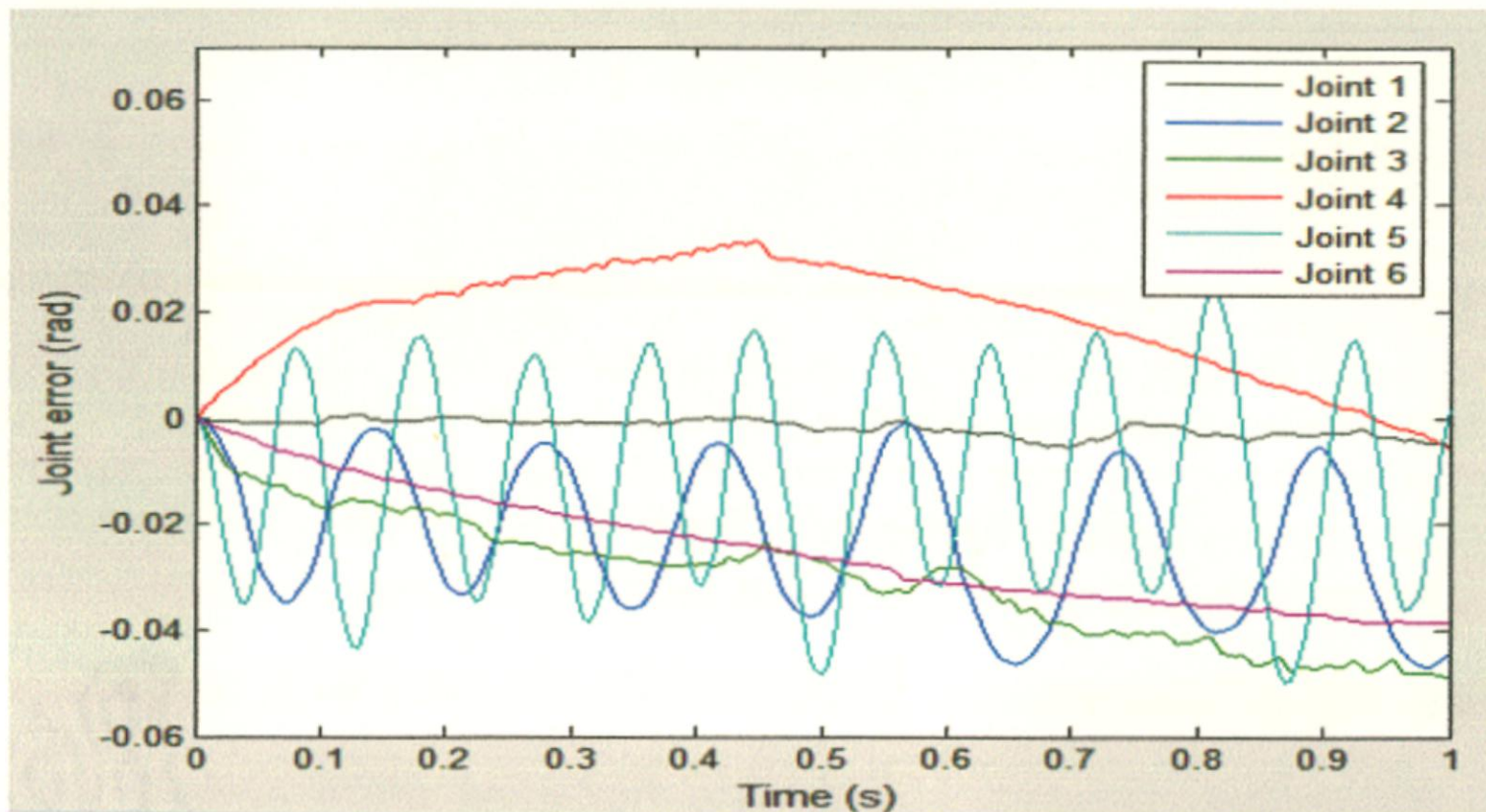
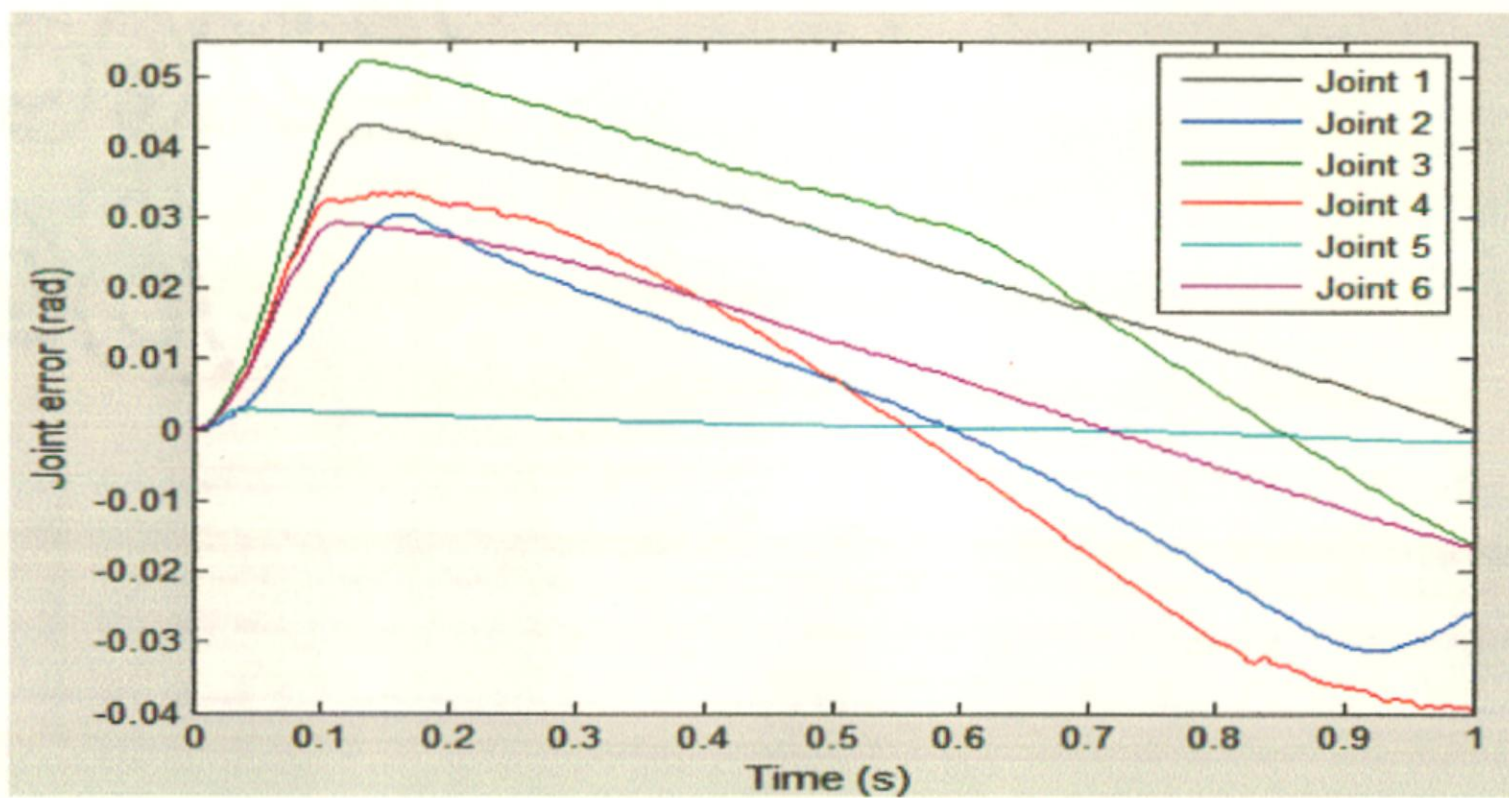


Figure 6.24 Surface plots of the bi-variate polynomials generated.

The Bivariate polynomial equation and its coefficients are given in the Appendix A.3. It is seen from Figure 6.25, that the Bi-variate polynomial of sufficiently high degree can approximate a fuzzy controller with certain approximation error. The approximation error can act both constructively and destructively. The aim of the section was to create a method which will extend to approximating a multivariable and a complex system. With this approximation technique any systems approximate model can be created. This particularly is useful when we need to make an approximate model of a real time system without going through rigorous mathematical procedures.



(a)



(b)

Figure 6.25 (a) Joint error by the bivariate polynomial approximated controller and (b) joint error of the reference Fuzzy PD+I controller.

6.2.2 Weighted-rule Fuzzy approximation

The result of the weighted-rule fuzzy approximate system is presented here. The weighted fuzzy approximate system so found approximates fuzzy controller behavior in the region of operation for which the data points were collected. Figure 6.26 shows the Membership functions (MF) generated, top row denoting joint 1 MF, next row for joint 2 MF, and so on. Figure 6.27 shows the weighted rulebase, with weights written within

brackets. Figure 6.28 shows the control surface. Figure 6.29 shows the joint error by the both weighted rule fuzzy approximate controller and the reference Fuzzy PD+I controller

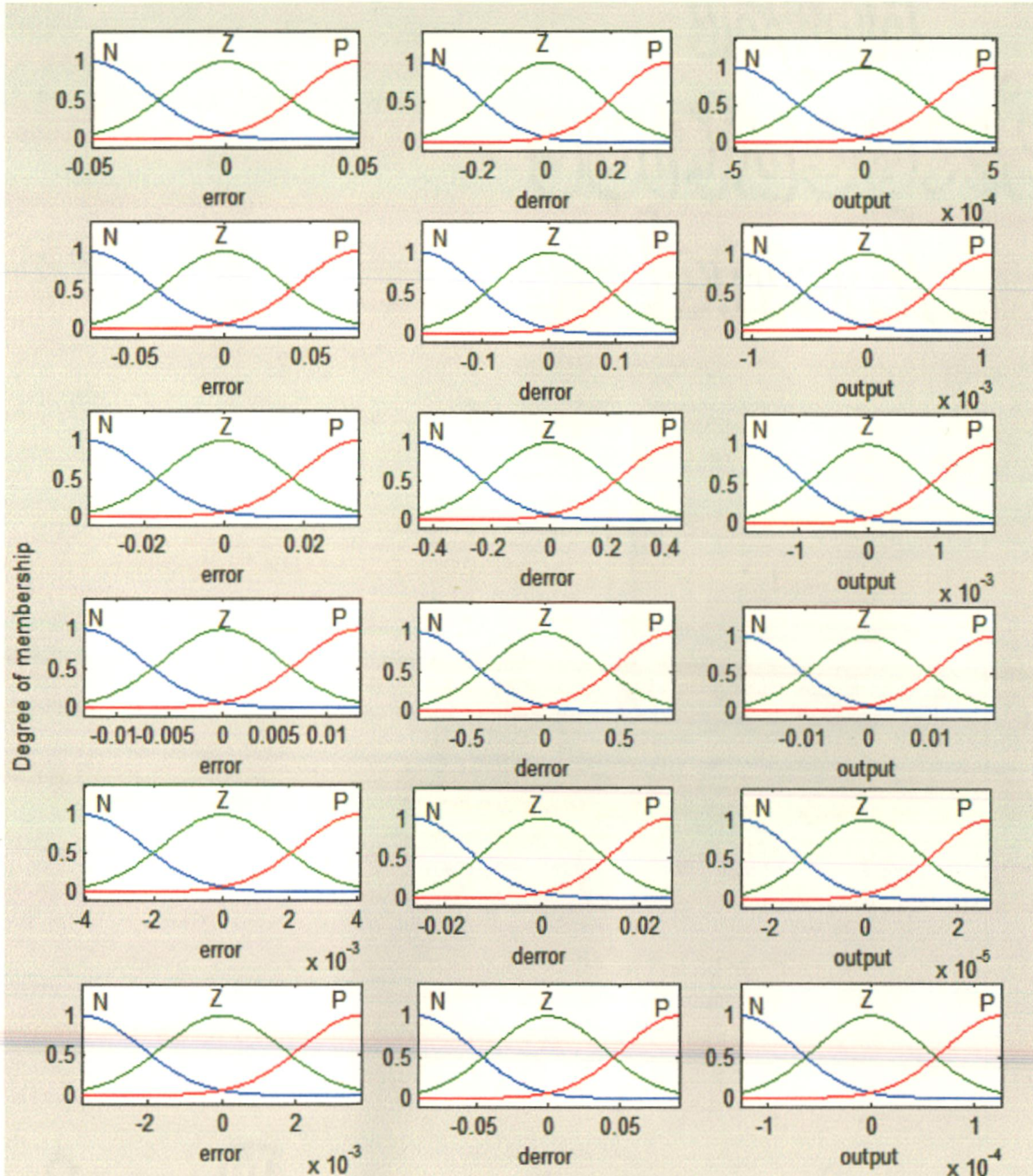


Figure 6.26 Membership functions of the weighted-rule fuzzy approximate system

Joint 1 Rulebase

| de e | N | Z | P |
|---------|---------------|---------------|---------------|
| N | N (0.7482) | N (0.9366) | Z (0.1492) |
| Z | N (0.9828) | Z (0.0195) | P (0.4181) |
| P | Z (0.0331) | P (0.1178) | P (0.5807) |

Joint 2 Rulebase

| de e | N | Z | P |
|---------|---------------|---------------|---------------|
| N | N (0.7919) | N (0.9318) | Z (0.1021) |
| Z | N (0.5848) | Z (0.0118) | P (0.3798) |
| P | Z (0.1012) | P (0.321) | P (0.5908) |

Joint 3 Rulebase

| de e | N | Z | P |
|---------|---------------|---------------|---------------|
| N | N (0.0155) | N (0.8018) | Z (0.9517) |
| Z | N (0.9764) | Z (0.0388) | P (0.2159) |
| P | Z (0.2292) | P (0.1106) | P (0.1669) |

Joint 4 Rulebase

| de e | N | Z | P |
|---------|---------------|---------------|---------------|
| N | N (0.8136) | N (0.0718) | Z (0.9437) |
| Z | N (0.8637) | Z (0.0584) | P (0.0638) |
| P | Z (0.0301) | P (0.1019) | P (0.1282) |

Joint 5 Rulebase

| de e | N | Z | P |
|---------|---------------|---------------|---------------|
| N | N (0.0649) | N (0.9553) | Z (0.6651) |
| Z | N (0.839) | Z (0.1698) | P (0.377) |
| P | Z (0.9653) | P (0.7023) | P (0.4319) |

Joint 6 Rulebase

| de e | N | Z | P |
|---------|---------------|---------------|---------------|
| N | N (0.0449) | N (0.9376) | Z (0.6917) |
| Z | N (0.9659) | Z (0.0053) | P (0.1416) |
| P | Z (0.3410) | P (0.0512) | P (0.0281) |

Figure 6.27 Rulebase of the weighted-rule fuzzy approximate system

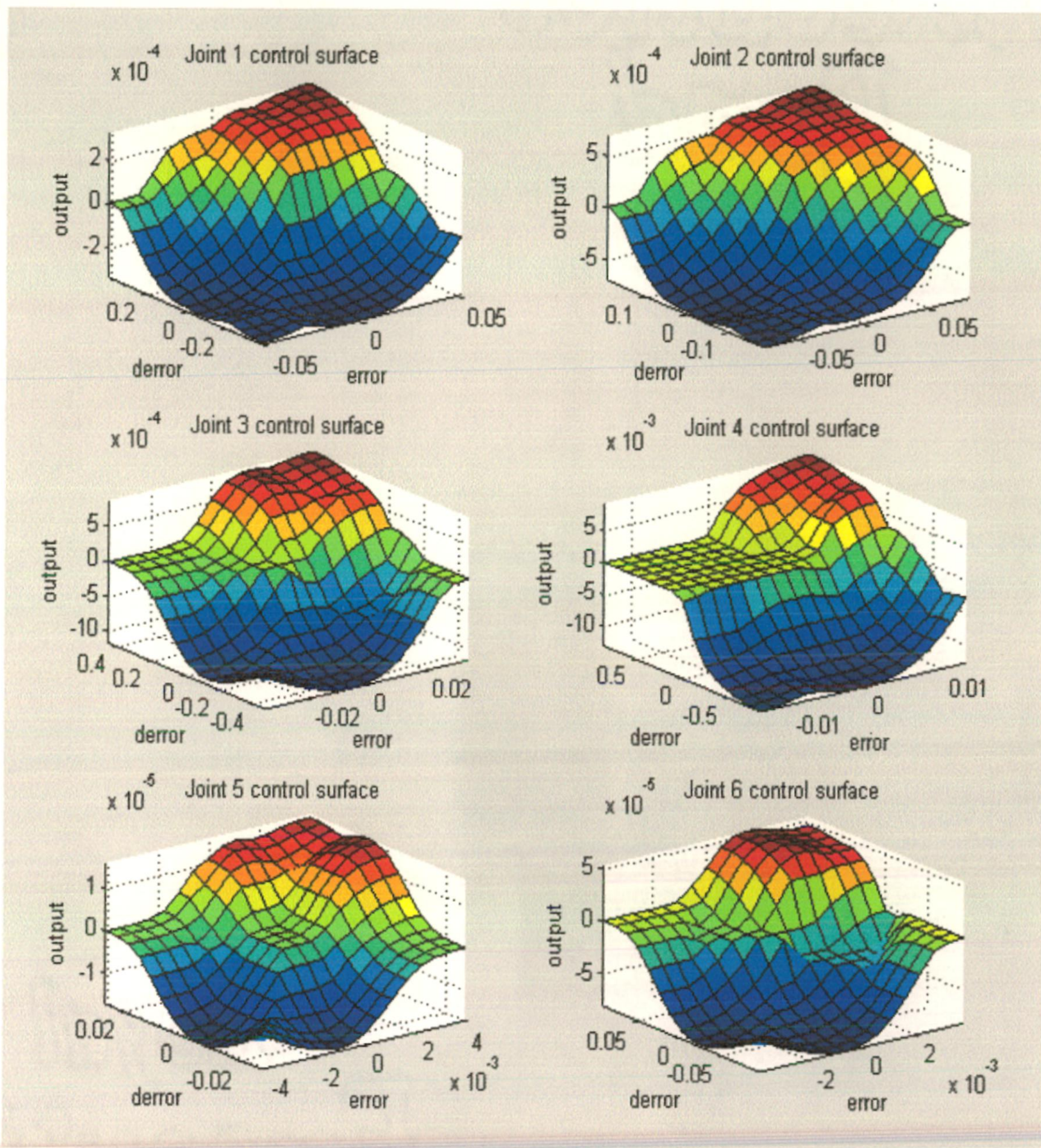
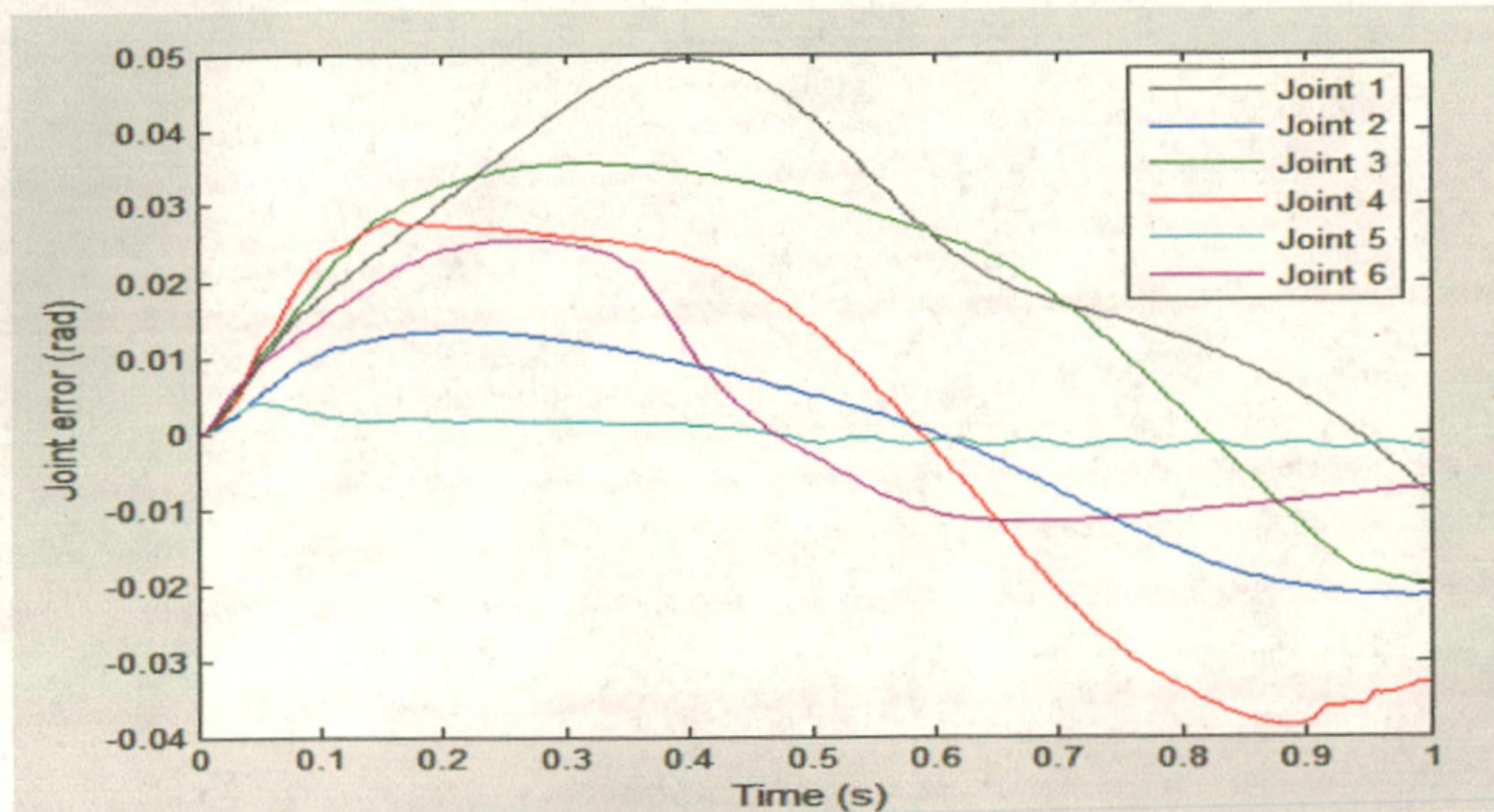
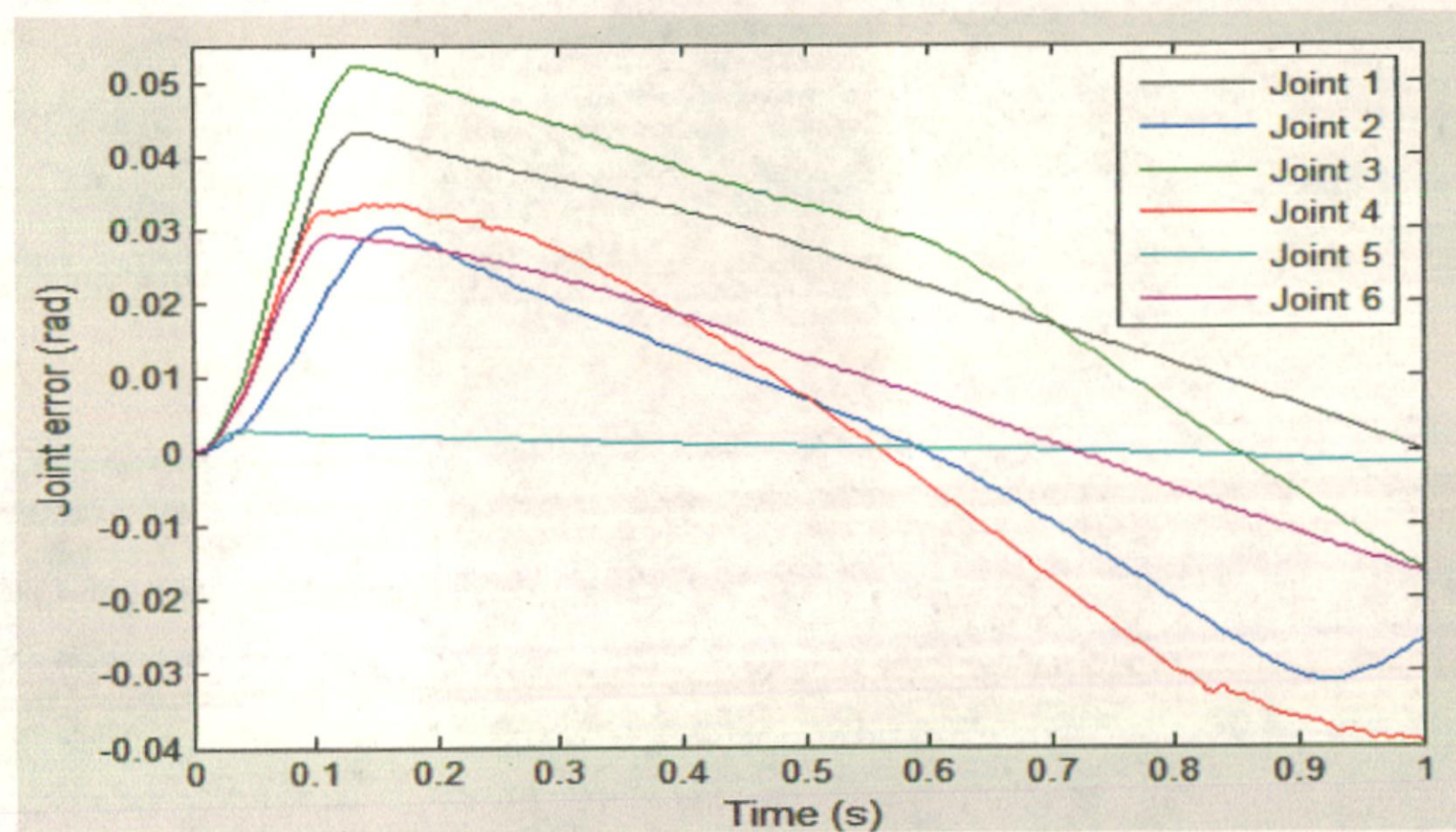


Figure 6.28 Control surface of the weighted-rule fuzzy approximate system



(a)



(b)

Figure 6.29 (a) Joint error by the weighted-rule fuzzy approximate system and (b) joint error of the reference Fuzzy PD+I controller.

Table 6.3 shows the tabulated results of the joint error ISEs of polynomial approximation and weighted-rule fuzzy approximation strategies. Table 6.4 shows the tabulated results of the approximation errors of polynomial approximation and weighted-rule fuzzy approximation strategies

Table 6.3 Tabulated results of the joint errors of polynomial approximation and weighted-rule fuzzy approximation methods

| Type | Joint 1 ISE | Joint 2 ISE | Joint 3 ISE | Joint 4 ISE | Joint 5 ISE | Joint 6 ISE | Total ISE |
|------------------------------------|----------------|----------------|----------------|----------------|----------------|----------------|--------------|
| Base sys | 0.000691 | 0.000361 | 0.00099 | 0.000624 | 1.88E-06 | 0.000276 | 2.94E-03 |
| Bivariate polynomial approximation | 5.20E-06 | 0.000694 | 0.001036 | 0.000449 | 0.000521 | 0.000717 | 3.42E-03 |
| Weighted-Rule fuzzy approximation | 0.000837 | 0.00015 | 0.000624 | 0.00062 | 3.00E-06 | 0.000208 | 2.44E-03 |

Table 6.4 Tabulated results of the approximation errors of polynomial approximation and weighted-rule fuzzy approximation methods

| Type | Joint 1 | Joint 2 | Joint 3 | Joint 4 | Joint 5 | Joint 6 | Total ISE |
|--|----------|----------|----------|----------|----------|----------|--------------|
| Bivariate polynomial approximation error (ISE) | 0.000751 | 0.000959 | 0.003183 | 0.000672 | 0.000537 | 0.001214 | 7.31E-03 |
| Weighted-Rule fuzzy approximation error (ISE) | 9.32E-05 | 5.33E-05 | 8.93E-05 | 1.78E-05 | 9.79E-07 | 9.49E-05 | 3.49E-04 |

Comparing the Figure 6.25 and Figure 6.29 and from the data in Table 6.3 it is evident that the approximation by weighted fuzzy systems is a better option when compared to the bivariate-polynomial approximation technique. The approximation error from the Table 6.4 also verifies the same. Approximation by weighted fuzzy systems is convenient for a two input one output systems, but when the number of inputs is large then approximation by weighted fuzzy system will become tricky because of the explosion in the number of rules. In this case the approximation by an n-variate polynomial will be of use.

6.3. Results for Optimally minimum rulebase

For any given input trajectory, all the rules in the rulebase are not fired and those fired may not be the best. This suggests that the rulebase can be minimized with only the best rule entries. The rule base so obtained is an optimally minimum rulebase which is achieved by tuning the rulebase in such a way that the number of rules is minimized and the rules in them are optimized simultaneously. The result of the section 5.3 is presented here.

The Fuzzy PD controller design used in section 6.1.1.1 is used here, except that the gains are tuned optimally for the given input Sinusoidal signal with frequency of 2 rad/sec and amplitude of [1 2 3 4 5 6] rad, amplitude of 1 for joint 1, 2 for joint 2 and so on. The Fuzzy PD+I controller after genetically gain tuning for the input shown in Figure 6.30 input applied only for 1 sec, the reference system for this section is obtained. The Joint errors plot for this reference system is shown in Figure 6.31.

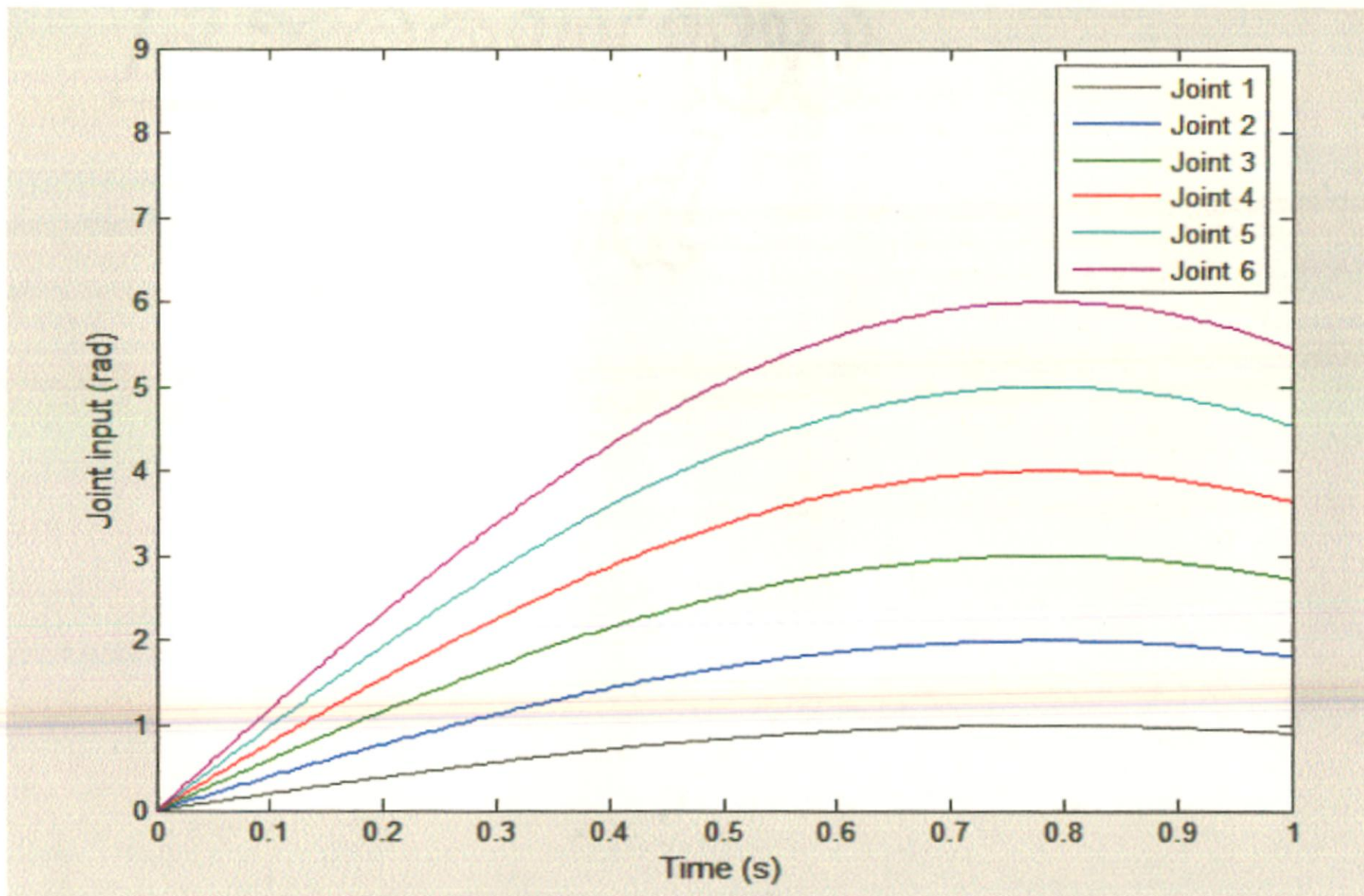


Figure 6.30 Input joint trajectories for genetic algorithm based optimally gain tuning of reference system used for optimally minimum rulebase study.

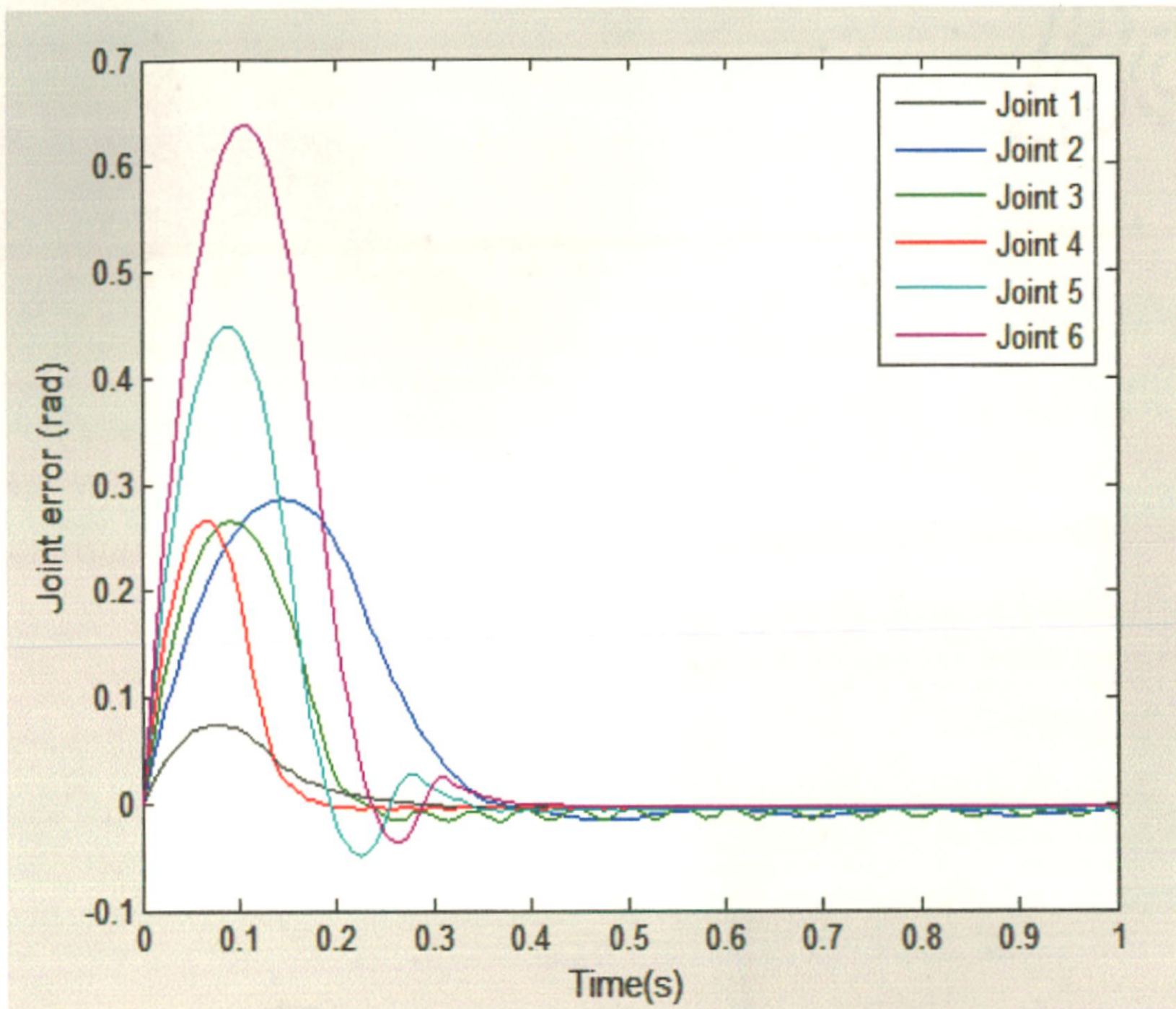


Figure 6.31 Joint errors of the reference system (for optimally minimum rulebase study) after genetic algorithm gain tuning.

After the reference system for this section is obtained, generation of optimally minimum rulebase by genetic algorithm is carried out. Figure 6.32 shows the optimally minimum rulebase of the Fuzzy PD+I controller created by genetic algorithm. Figure 6.33 shows its control surface, and Figure 6.34 shows the joint error plot for input signal shown in Figure 6.30.

Joint 1 Rulebase

| | | | |
|-----------------------|------------|------------|------------|
| de e | mf1 | mf2 | mf3 |
| mf1 | mf1 | | |
| mf2 | | | |
| mf3 | | | mf3 |

Joint 2 Rulebase

| | | | |
|-----------------------|------------|------------|------------|
| de e | mf1 | mf2 | mf3 |
| mf1 | | | mf1 |
| mf2 | mf1 | | mf3 |
| mf3 | mf2 | mf3 | mf2 |

Joint 3 Rulebase

| | | | |
|-----------------------|------------|------------|------------|
| de e | mf1 | mf2 | mf3 |
| mf1 | | | |
| mf2 | mf1 | | |
| mf3 | mf3 | | mf3 |

Joint 4 Rulebase

| | | | |
|-----------------------|------------|------------|------------|
| de e | mf1 | mf2 | mf3 |
| mf1 | | mf1 | |
| mf2 | mf1 | mf2 | mf2 |
| mf3 | | mf3 | mf2 |

Joint 5 Rulebase

| | | | |
|-----------------------|------------|------------|------------|
| de e | mf1 | mf2 | mf3 |
| mf1 | mf1 | | |
| mf2 | | | |
| mf3 | | | mf3 |

Joint 6 Rulebase

| | | | |
|-----------------------|------------|------------|------------|
| de e | mf1 | mf2 | mf3 |
| mf1 | mf1 | | mf1 |
| mf2 | | | mf3 |
| mf3 | | | |

Figure 6.32 Optimally minimum rulebase created by genetic algorithm

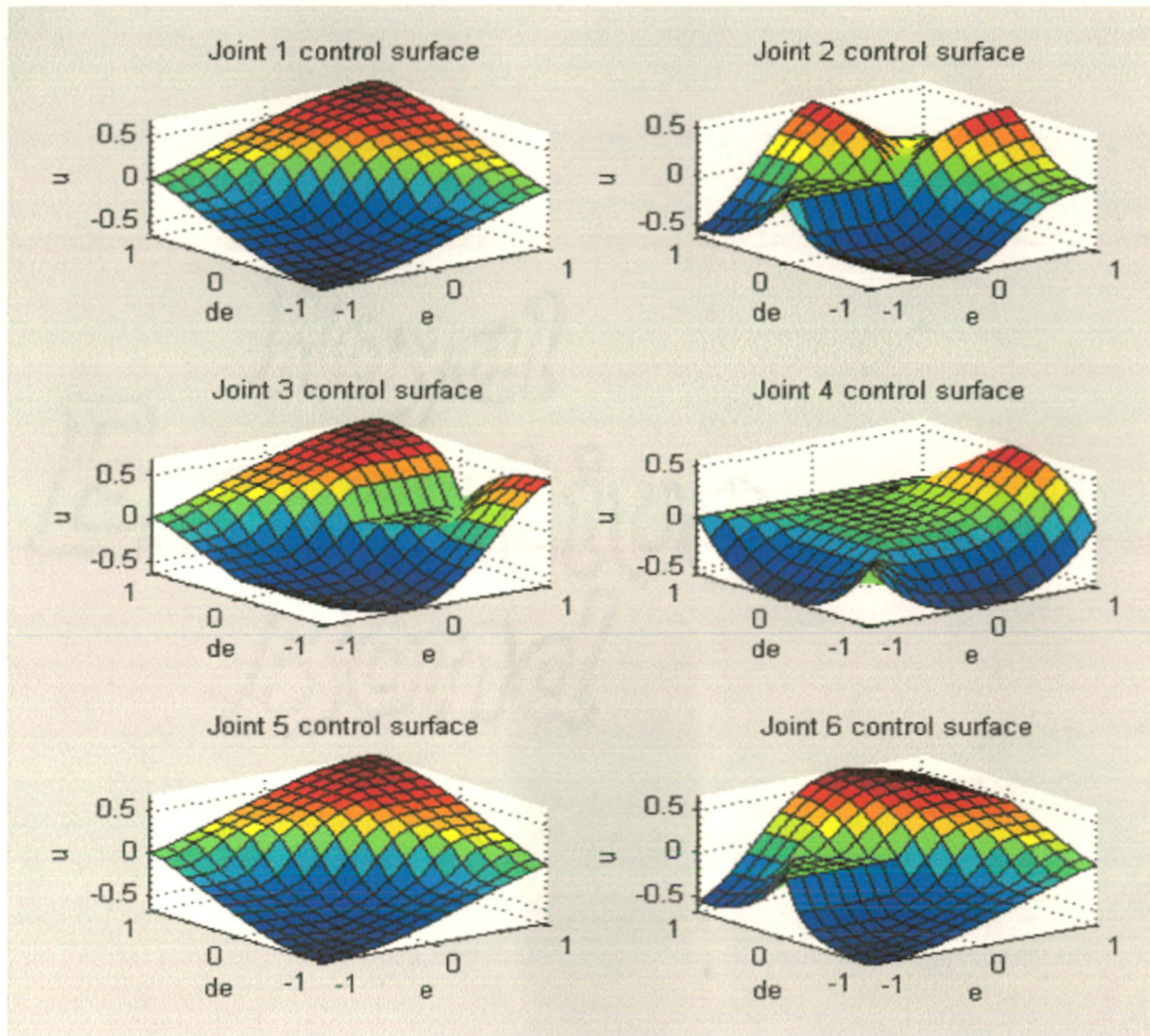


Figure 6.33 Control surface of optimally minimum rulebase

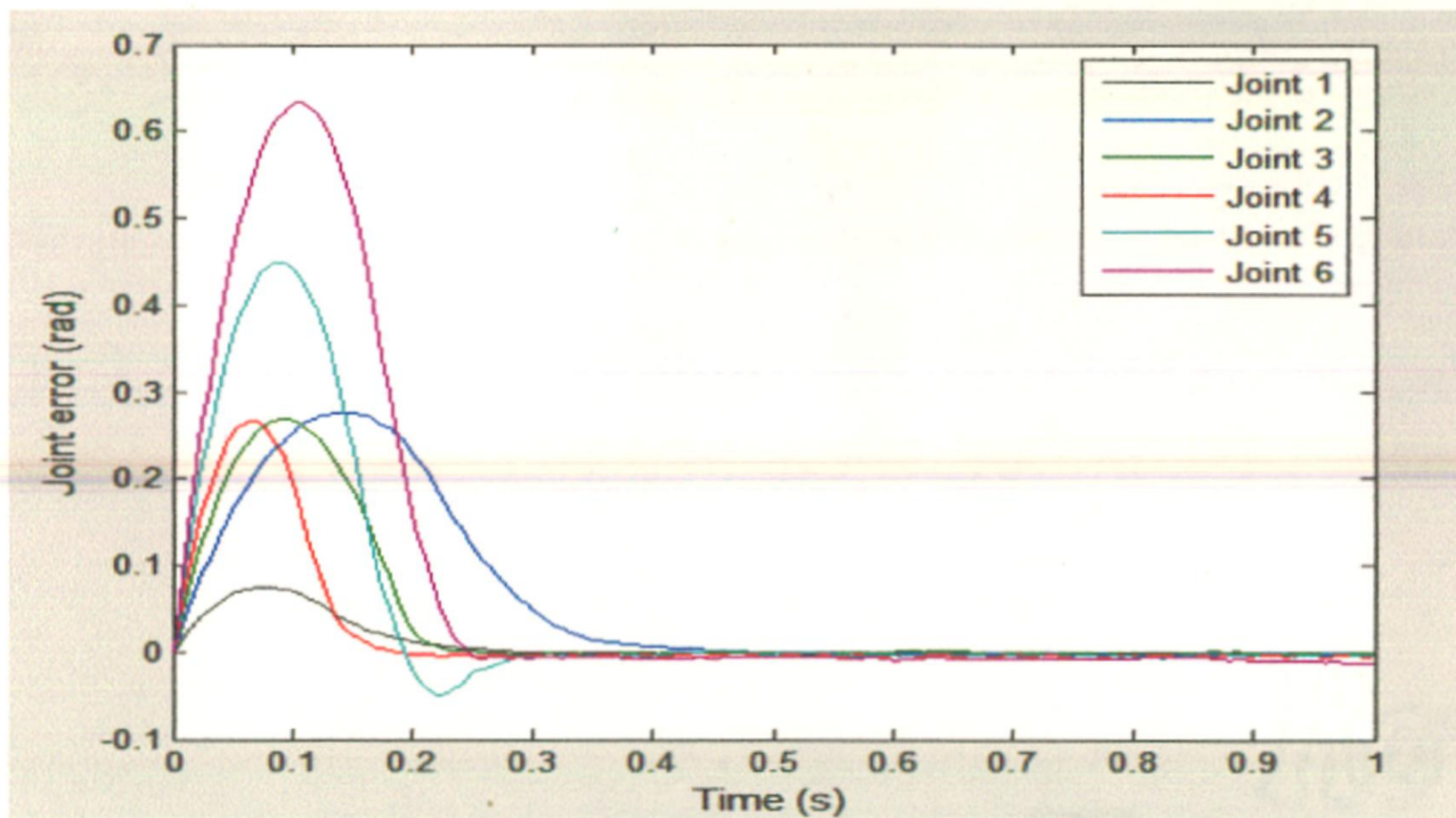


Figure 6.34 Joint errors for optimally minimum rulebase.

After the generation of the optimally minimum rulebase, disturbance analysis is done on this system. The input is now extended till π sec, and a disturbance as shown in Figure 6.35 is applied at 1 sec to the output joint angle. Figure 6.36 shows the joint error plot for this disturbance. The input and output joint trajectory for the optimally minimum system is shown in Figure 6.37. From the plot in Figure 6.36 and Figure 6.37 it is seen that this rulebase can handle small disturbances as well.

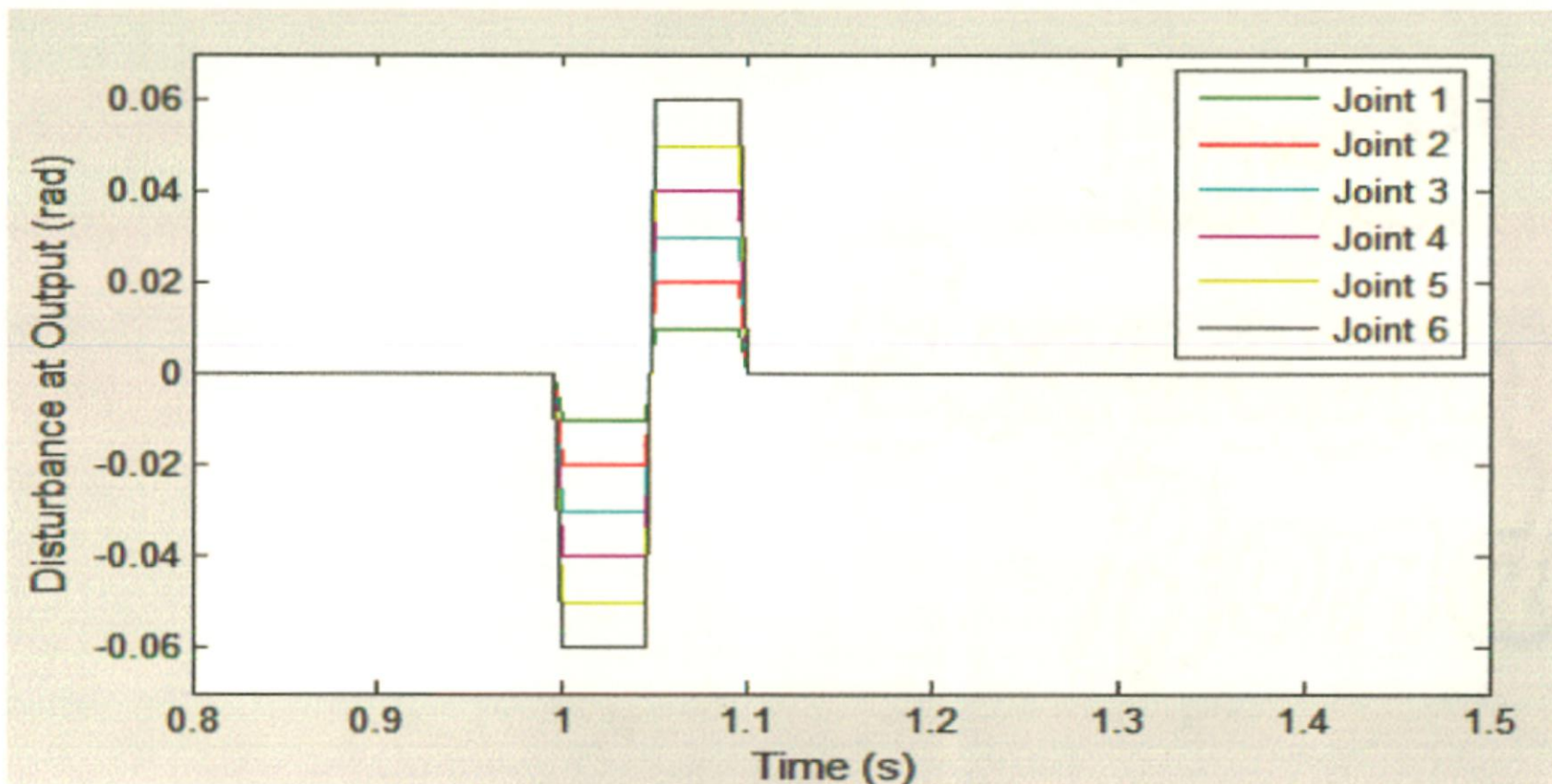


Figure 6.35 Plot of the disturbance signal given to the robot arm at 1 sec

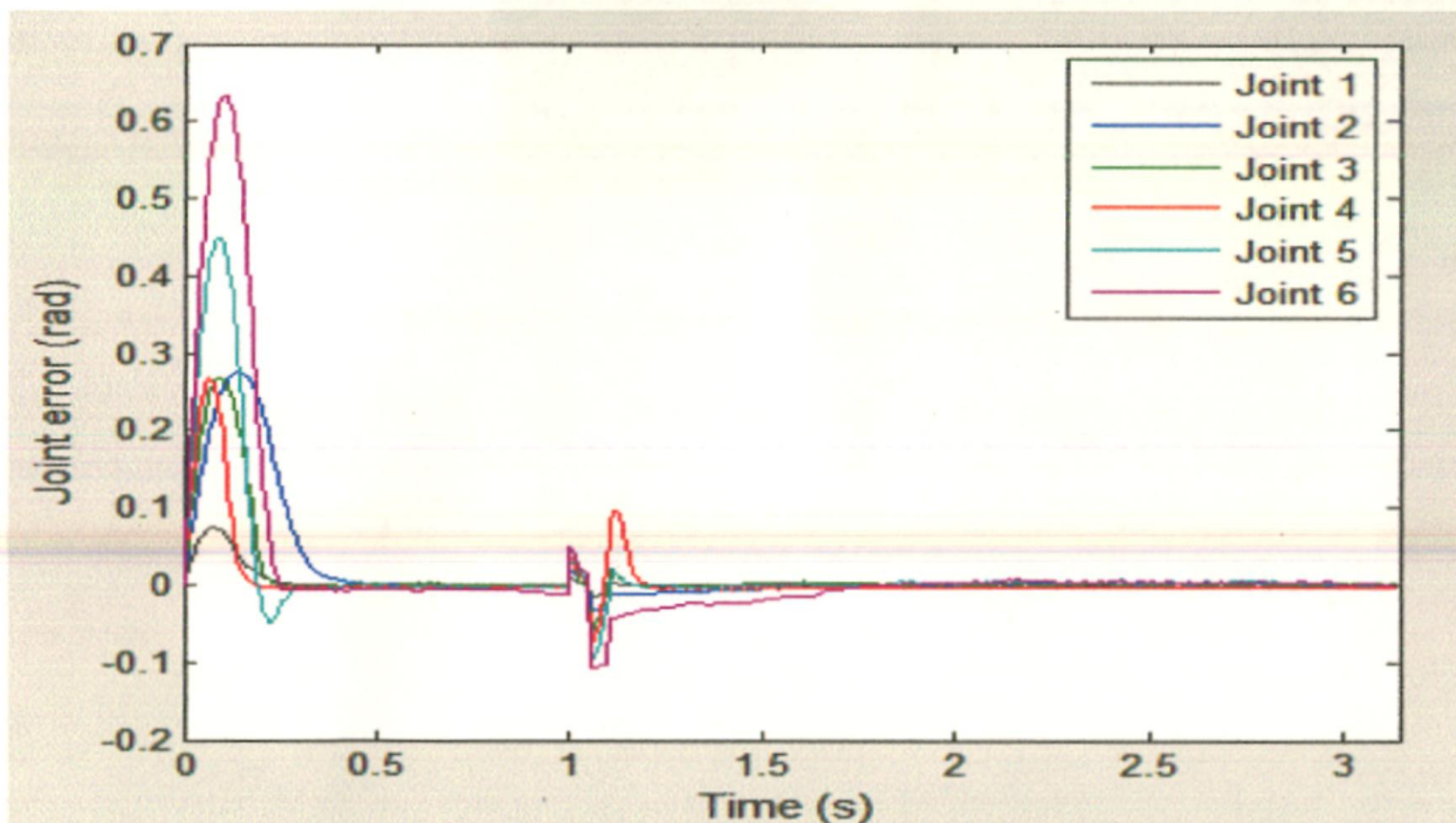


Figure 6.36 Joint errors for optimally minimum rulebase with disturbance signal given at 1 sec

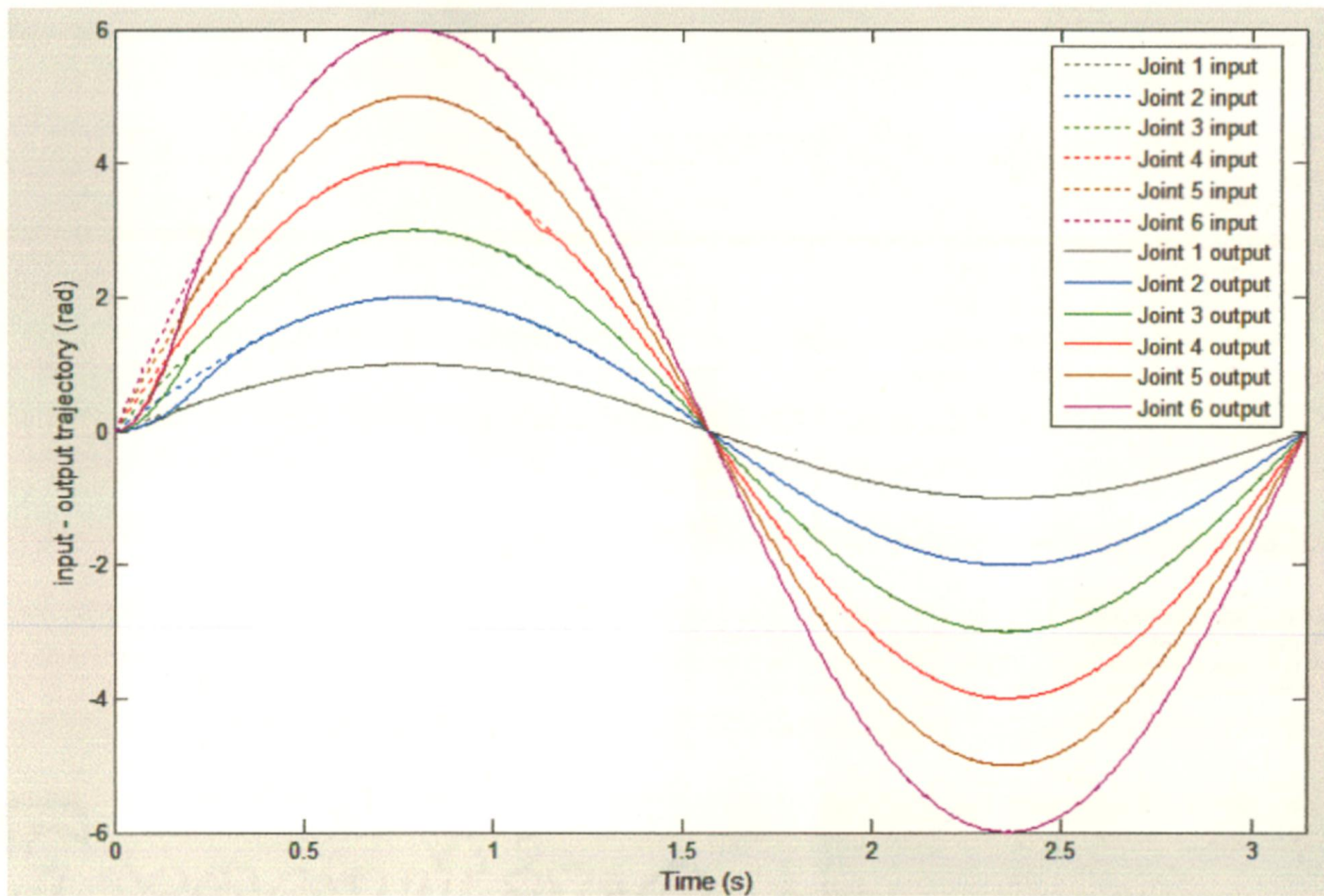


Figure 6.37 Input and output joint trajectory for optimally minimum rulebase.

6.4 Results for proposed Stochastic Algorithm

The results of the proposed algorithm are presented here. To verify the functioning of the proposed algorithm, it is first tested on Rastrigin's function. Rastrigin's function is often used to test the algorithm, because it has many local minima. Once the algorithm works on Rastrigin's function, it can be tested on other functions. For two independent variables, Rastrigin's function is defined as in Equation (6.2)

$$Ras(x) = 20 + x_1^2 + x_2^2 - 10(\cos 2\pi x_1 + \cos 2\pi x_2) \quad (6.2)$$

Rastrigin's function has many local minima but the function has just one global minimum [23], which occurs at the point $[0, 0]$ in the x - y plane, where the value of the function is 0. At any local minimum other than $[0, 0]$, the value of Rastrigin's function is greater than 0. The farther the local minimum is from the origin, the larger the value of the function is at that point. Figure 6.38 shows the plot of Rastrigin's function.

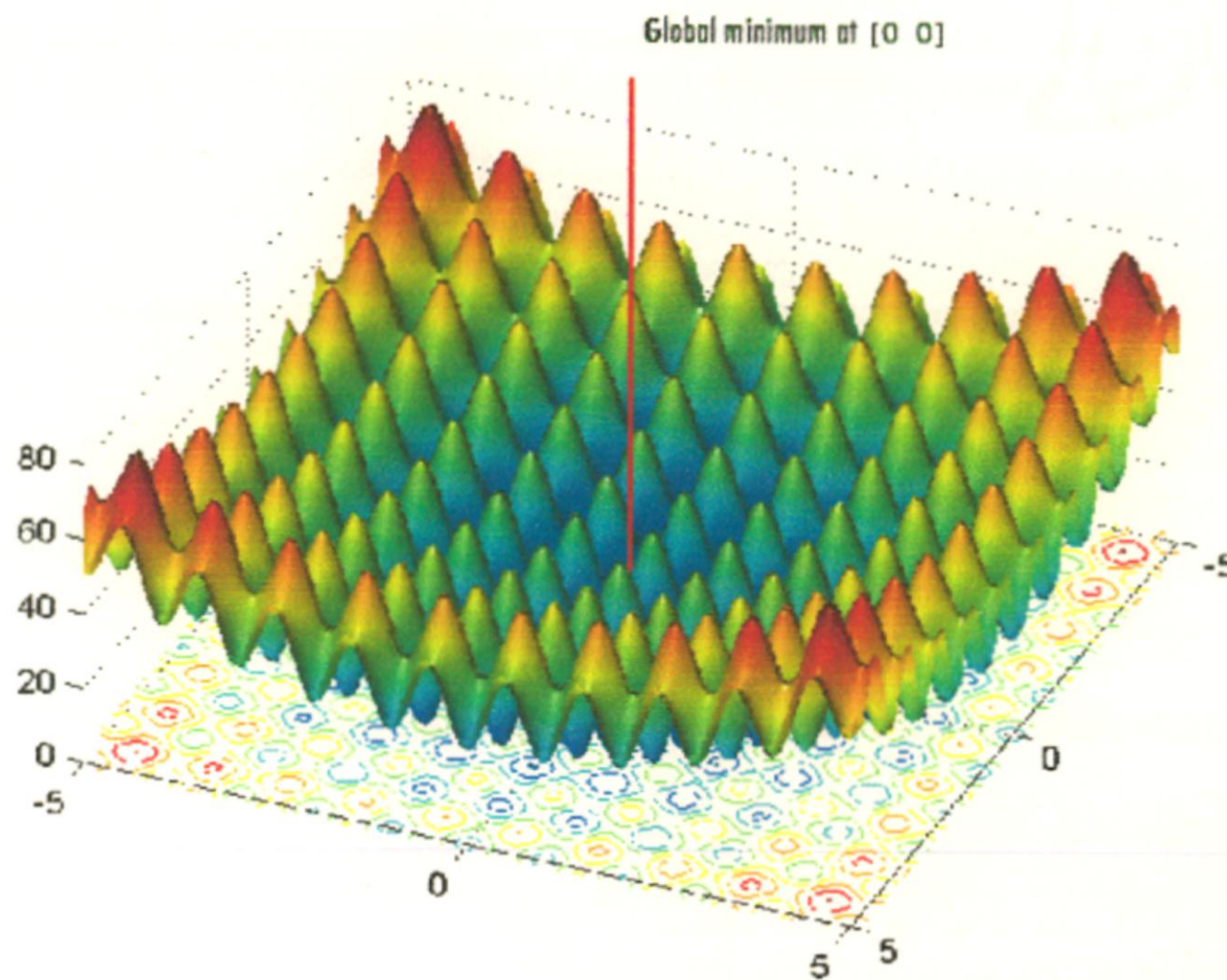


Figure 6.38 Plot of the Rastrigin's Function

Table 6.5 shows the tabulated results for Rastrigin's function by Genetic algorithm and proposed algorithm. The initial range of search space was [9, 10].

Table 6.5 Results for Rastrigin's function by Genetic algorithm and proposed algorithm

| Algorithm used | x1 | x2 | y |
|--------------------|----------|-----------|----------|
| Genetic Algorithm | 0.0011 | 0.99877 | 0.99808 |
| Proposed algorithm | 2.06E-07 | -2.21E-07 | 1.81E-11 |

Figure 6.39 shows the comparative plot for objective function value of Rastrigin's function by Genetic algorithm and proposed algorithm with the x axis denoting the objective function value of Rastrigin's function and y axis denoting the number of iteration. Since we are using genetic algorithm with population size of 20, one generation corresponds to 20 iterations. Genetic algorithm is run for 50 generations, in other words 1000 iterations.

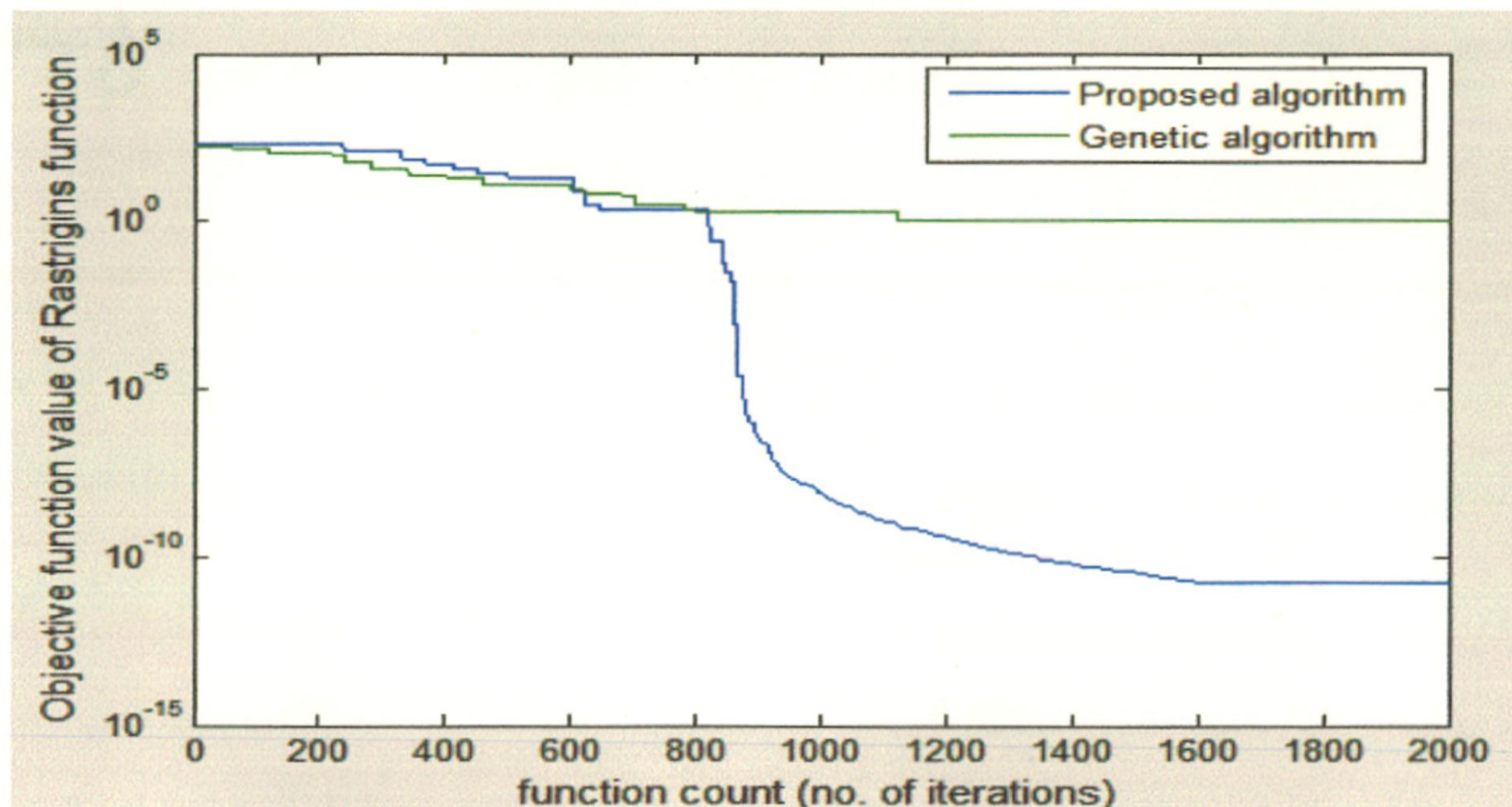


Figure 6.39 Comparative plot for objective function value of Rastrigin's function by Genetic algorithm and proposed algorithm

With the encouraging Rastrigin's function result, the algorithm is used for tuning the gains of the reference Fuzzy PD+I controller described in section 6.1.1.1. Now a Pseudo-random joint angle trajectory is used as input to robot arm as shown in Figure 6.40. Pseudo random joint trajectory generation is discussed in Appendix A.4.

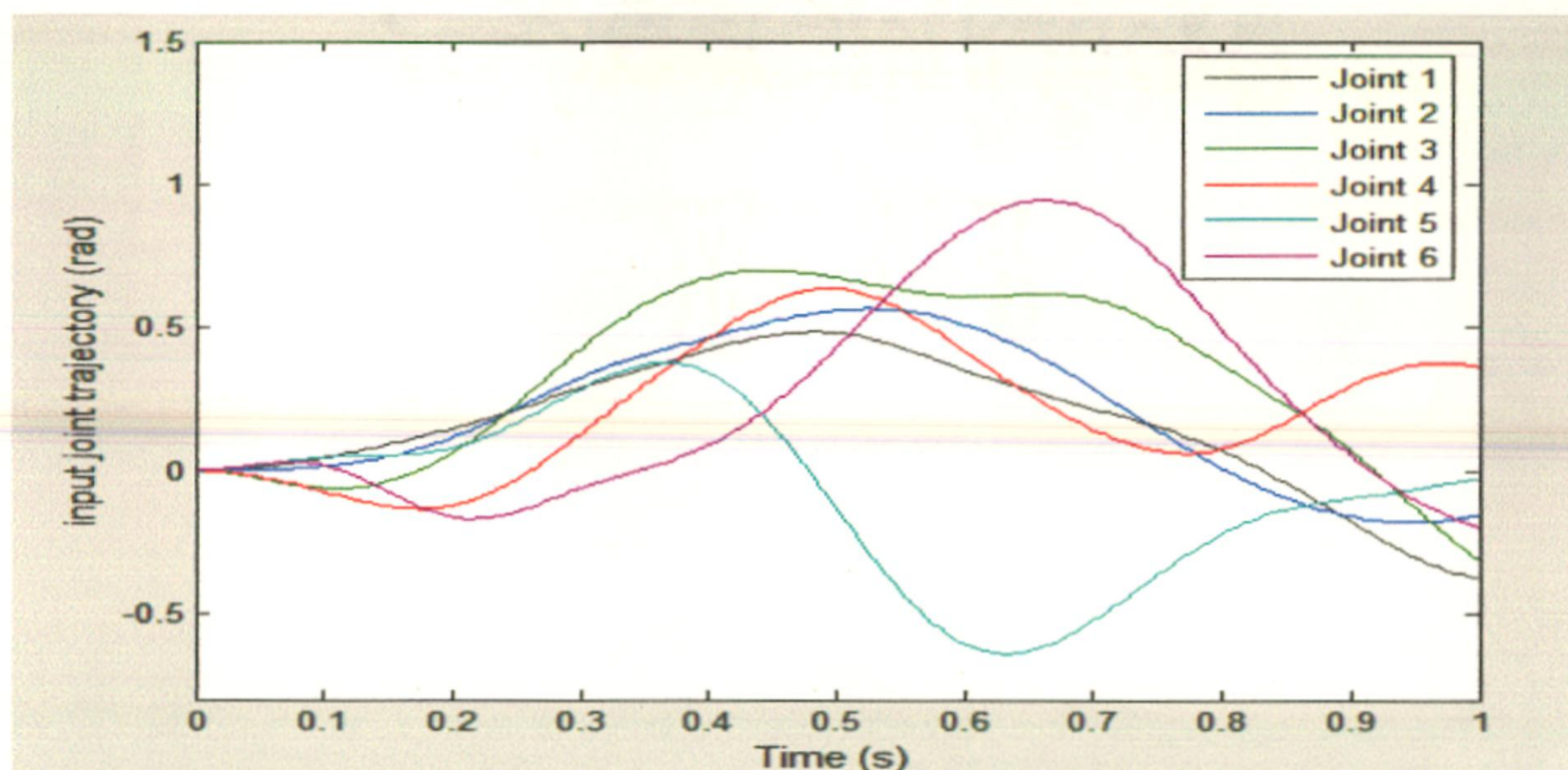


Figure 6.40 Pseudo-random input joint angle trajectories

The objective function used is same as Equation (6.1). Figure 6.41 shows the convergence of both Genetic algorithm and the proposed algorithm. Figure 6.42 shows the joint error plot for gain tuning by Genetic algorithm and Figure 6.43 shows the joint error plot for the gain tuning by the proposed algorithm.

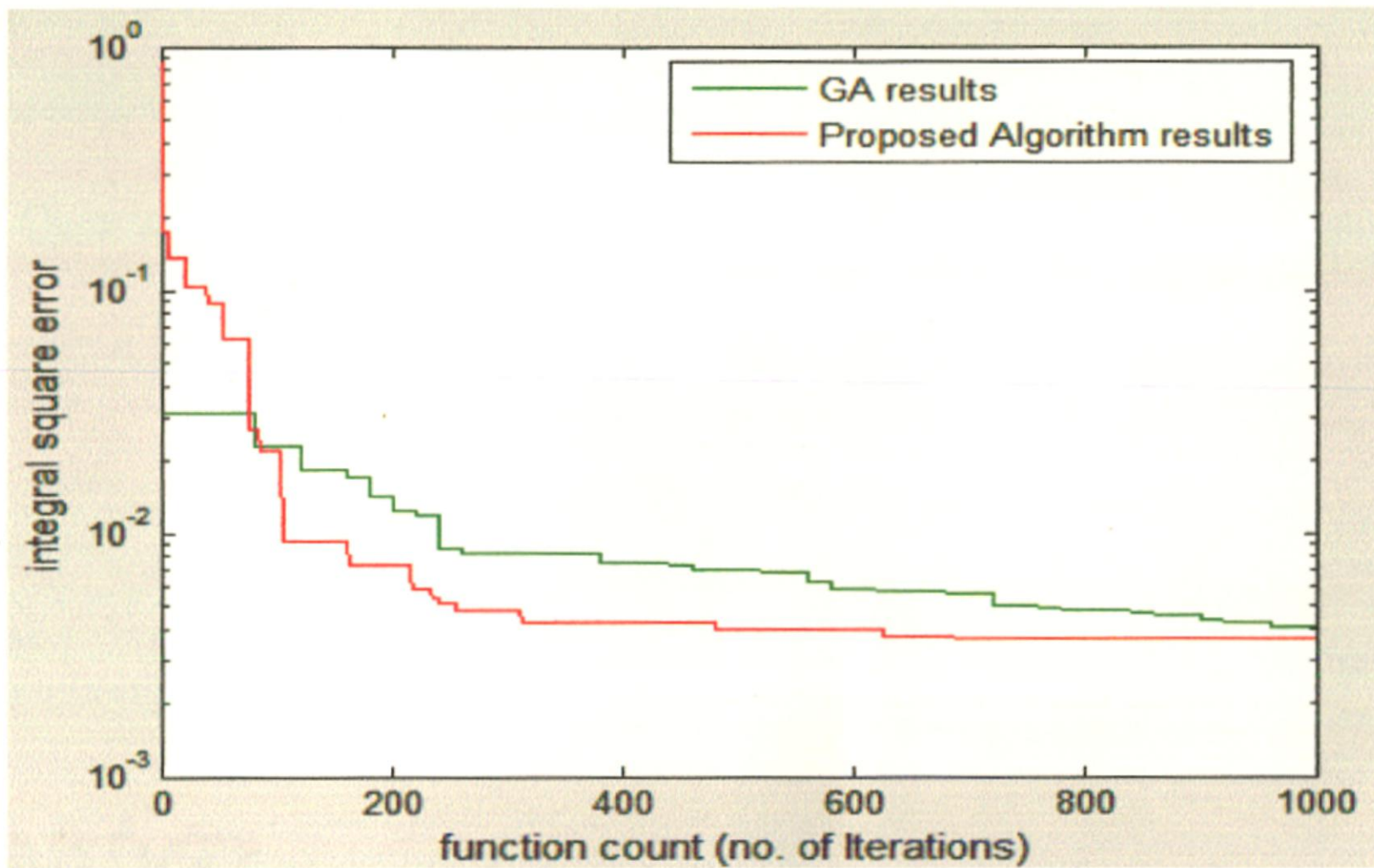


Figure 6.41 Convergence plot of Genetic algorithm and the proposed algorithm.

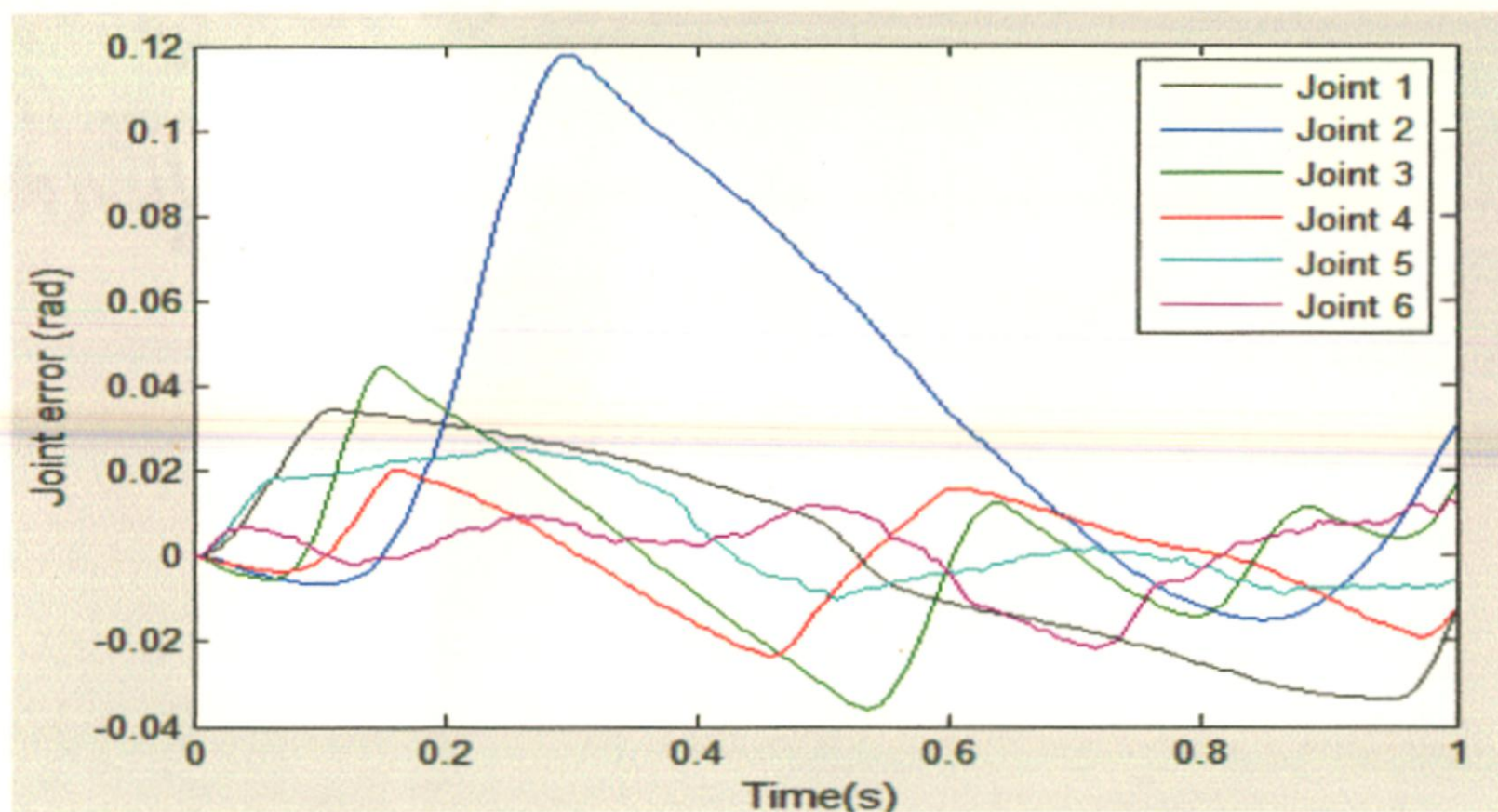


Figure 6.42 Joint errors plot for gain tuning by Genetic algorithm

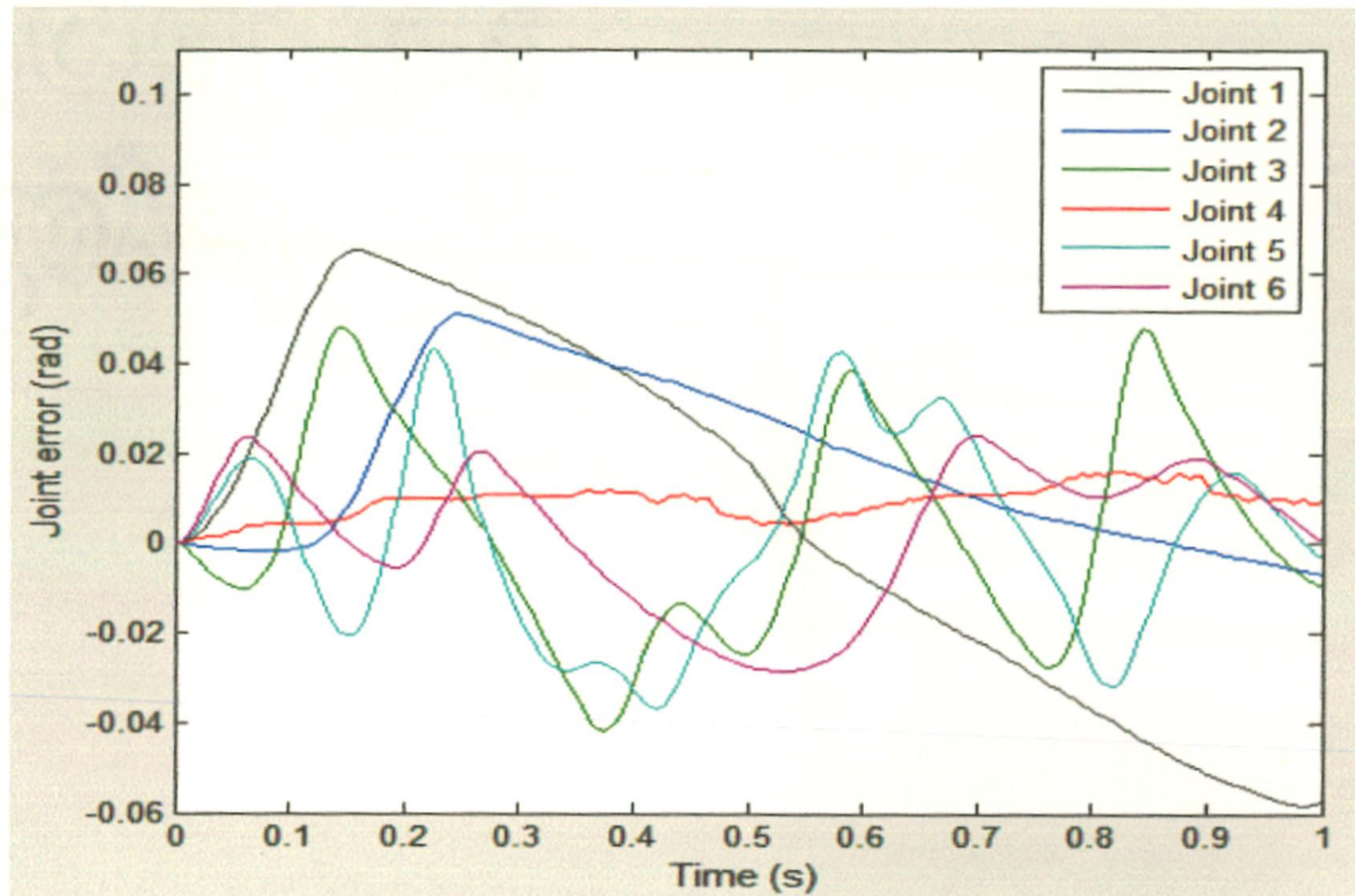


Figure 6.43 Joint errors plot for gain tuning by proposed algorithm

The value of the performance index by the Equation (6.1) is $4.15E-03$ for Genetic algorithm after 1000 iterations (50 generation) and $3.63E-03$ for the propose algorithm after 1000 iterations

Chapter 7: Conclusions and Future Scope

In this dissertation, Genetic fuzzy system for the joint control of PUMA560 is evaluated using various methodologies. As part of the preliminary studies Rule tuning, Rule-weight tuning and Membership function tuning were successfully carried out using Genetic algorithm and it has been established that Rule-weight tuning gives a better performance index when compared with the other two. Later two stage tuning method of Rule followed by Rule-weight tuning is evaluated and it was seen that there is performance improvement. Next, three stage tuning method consisting of two stage tuning followed by Membership function tuning is evaluated and it was established that there is still better performance improvement. Simultaneously tuning of all these parameters were carried out and based on the results, it was found that its performance is inferior to the three stage tuning method.

Approximation of the fuzzy function is carried out from its input output data. Of the two methods used for approximation, namely Bivariate polynomial approximation and Weighted-Rule Fuzzy approximation, The Weighted-Rule Fuzzy approximation is better. The bivariate Polynomial approximation can be easily extendable to multi-variate scenario, but will have a large approximation error on the other hand extending Weighted-Rule Fuzzy approximation for multiple inputs can get complicated.

Optimally minimum rule base was generated successfully by genetic algorithm and employed for control of joint trajectory of PUMA560. It was seen that this rule base can successfully handle small disturbances as well.

Evaluation of the proposed stochastic algorithm based on the empirical and graphical data are encouraging. It proves its worth by being able to converge at a faster rate when compared to Genetic algorithm for the Rastrigin's function and gain tuning of Fuzzy PD+I controller of PUMA560.

As part of future work, improvement of the simultaneous tuning method can be undertaken by studying the effects of genetic fitness, scaling and other option on it. The Weighted-Rule Fuzzy approximation can still be improved by tuning other parameters along with the weights. Improving the proposed algorithm can be a worthy undertaking.

References

- [1]. Bruno Siciliano , Lorenzo Sciavicco, Luigi Villani and Giuseppe Oriolo, “*Robotics: Modelling, Planning and Control*”, Advanced Textbooks in Control and Signal Processing series, ISSN 1439-2232, ISBN 978-1-84628-641-4, Springer-Verlag London Limited, 2009.
- [2]. Lakhmi C. Jain and N.M. Martin, “*Fusion of Neural Networks, Fuzzy Systems and Genetic Algorithms: Industrial Applications*”, CRC Press, CRC Press LLC, ISBN: 0849398045, 1998.
- [3]. Jan Jantzen, “*Foundations of Fuzzy Control*”, ISBN 978-0-470-02963-3, John Wiley & Sons Ltd, 2007.
- [4]. S.N.Sivanandam and S.N.Deepa, “*Introduction to Genetic Algorithms*”, ISBN 978-3-540-73189-4, Springer-Verlag Berlin Heidelberg 2008.
- [5]. Bong Joo Kim and Chung Choo Chung, “*Design of Fuzzy PD + I Controller for Tracking Control*”, Proceedings of the American Control Conference Anchorage, AK May 8-1, AACC ,p: 2124-2129, 2002.
- [6]. Srinivasan Alavandar and M.J. Nigam, “*Fuzzy PD + I control of a six DOF robot manipulator*”, Industrial Robot: An International Journal, Volume 35, Number 2, p:125–132, 2008.
- [7]. Sufian Ashraf Mazhari and Surendra Kumar, “*PUMA 560 Optimal Trajectory Control using Genetic Algorithm, Simulated Annealing and Generalized Pattern Search Techniques*”, International journal of electrical, computer and systems engineering 2;1, p:71-80, winter 2008.

- [8]. Sufian Ashraf Mazhari and Surendra Kumar, "*Heuristic Search Algorithms for Tuning PUMA 560 Fuzzy PID Controller*", International journal of computer science 3;4 , fall 2008.
- [9]. O.Cordon, F.Herrera, E.Herrera-Viedma and M.Lozano, "*Genetic Algorithms and Fuzzy logic in control processes*", Department of computer science and artificial intelligence (DECSAI), Technical report # DECSAI-95109, March, 1995.
- [10]. Oscar Cordon, Francisco Herrera, Frank Hoffmann and Luis Magdalena, "*Genetic Fuzzy Systems: Evolutionary Tuning And Learning Of Fuzzy Knowledge Bases*", Advances in Fuzzy Systems — Applications and Theory, World Scientific Publishing Co. Pte. Ltd. Vol. 19, ISBN 981-02-4016-3, 2001.
- [11]. Rafael Alcalá, Oscar Cordon, and Francisco Herrera, "*Combining Rule Weight Learning and Rule Selection to Obtain Simpler and More Accurate Linguistic Fuzzy Models*", Modelling with Words, LNAI 2873, pp. 44–64, 2003.Springer-Verlag Berlin Heidelberg 2003.
- [12]. Hisao Ishibuchi, Yutaka Kaisho, and Yusuke Nojima, "*Complexity, Interpretability and Explanation Capability of Fuzzy Rule-Based Classifiers*", FUZZ-IEEE 2009, Korea, August 20-24, 2009.
- [13]. D. Nauck, "*Adaptive Rule Weights in Neuro-Fuzzy Systems*", Neural Computing & Application, Springer-Verlag London Limited, 2000 vol.9 p: 60-70, 2000.
- [14]. Mariano Gasca and Thomas Sauer "*On the history of multivariate polynomial interpolation*", Journal of Computational and Applied Mathematics 122, p: 23-35, 2000
- [15]. Mariano Gasca and Thomas Sauer, "*Polynomial interpolation in several variables*", Advances in Computational Mathematics, vol 12; part 4, p: 377-410, 2000

- [16]. K. Belarbi, F. Titel, W. Bourebia and K. Benmahammed, "*Design of Mamdani fuzzy logic controllers with rule base minimisation using genetic algorithm*", Engineering Applications of Artificial Intelligence vol.18 p: 875–880, 2005
- [17]. Srinivasan Alavandar, "*Intelligent control of Robot manipulators using soft computing techniques*", PhD. Thesis report, Indian Institute of Technology Roorkee, December 2008
- [18] Structure of PUMA 560 robot manipulator. Web Resource available at, <http://www.emeraldinsight.com/fig/0490250605009.png>
- [19]. D.K. Chaturvedi, "*Soft Computing: Techniques and its Applications in Electrical Engineering*", Studies in Computational Intelligence, Volume 103, ISBN 978-3-540-77480-8, Springer-Verlag Berlin Heidelberg 2008.
- [20]. Kevin M. Passino and Stephen Yurkovich, "*Fuzzy Control*", Addison Wesley Longman, Inc., ISBN 0–201–18074–X, 1998
- [21]. Rafael Alcalá, Jose Ramon Cano, Oscar Cordon ,Francisco Herrera, Pedro Villar and Igor Zwir, "*Linguistic modeling with hierarchical systems of weighted linguistic rules*", International Journal of Approximate Reasoning 32, p:187–215, 2003
- [22]. M. Mucientes, R. Alcalá, J. Alcalá-Fdez, and J. Casillas, "*Learning Weighted Linguistic Rules to Control an Autonomous Robot*", International Journal of Intelligent Systems, Vol. 24, Issue 3, p: 226-251, 2009
- [23]. MATLAB Help file, MATLAB Version 7.0.1.24704 (R14 with Genetic algorithm & Direct search toolbox and Fuzzy Logic Toolbox), The MathWorks Inc, 2004

- [24]. F. Herrera and L. Magdalena. "*Genetic Fuzzy Systems: A Tutorial*", R.Mesiar, B.Riecan (Eds) Fuzzy structures, Current trends. Lecture Notes of the Tutorial: Genetic Fuzzy Systems. Seventh IFSA world congress (IFSA97), Prage, June 1997, Tatra Mountains Mathematical Publications Vol. 13, p: 93-121, 1997.
- [25]. F. Hoffmann, "*Evolutionary algorithms for fuzzy control system design*," Proc. IEEE, vol. 89, p: 1318–1333, 2001
- [26]. D. Drainkov, H. Hellendoorn and M. Rienfrank "*An Introduction to fuzzy control*", ISBN 81-7319-069-0, Springer-Verlag Berlin Heidelberg, 1993.
- [27]. David Terr. "*Bivariate Polynomial*", MathWorld, Web Resource, <http://mathworld.wolfram.com/BivariatePolynomial.html>
- [28]. Pierre Collet and Jean-Philippe Rennard, "*Stochastic Optimization Algorithms*", Handbook of Research on Nature Inspired Computing for Economics and Management, ISBN: 1-59140-984-5, 2006

Research and publications by the author

1. Singh Vivekkumar Radhamohan, Mona Subramaniam A and M.J.Nigam, "Fuzzy Swing-up and Stabilization of Real Inverted Pendulum using single Rulebase", Journal of Theoretical and Applied Information Technology, vol 14. no. 1 , pp. 43-50, April-2010.
2. Mona Subramaniam A, Manju A and M.J.Nigam, "Bi-variate Polynomial approximation of Fuzzy controller using Genetic Algorithm for Trajectory control of Puma560", International conference on advances in information and communication technologies 2010, Proceedings will be published by Springer LNCS-CCIS (*accepted for conference on 07-09 Sep 2010*)
3. Mona Subramaniam A, Manju A and M.J.Nigam, "Two-stage weighted Fuzzy Rulebase tuning using Genetic Algorithm for Trajectory control of Puma560", International Journal of Computational Cognition (*submitted for review*)
4. Mona Subramaniam A, Manju A and M.J.Nigam, "Comparative Study on parameter tuning of weighted Fuzzy Rulebase using Genetic Algorithm for Trajectory control of Puma560", Journal of Intelligent and Fuzzy Systems. (*Submitted for review*)
5. Mona Subramaniam A, Manju A and M.J.Nigam, "A study on parameter tuning of weighted Fuzzy Rulebase using Genetic Algorithm for Trajectory control of Puma560", (*under writing*)
6. Mona Subramaniam A, Manju A and M.J.Nigam, "Optimally minimum fuzzy rulebase generation using Genetic Algorithm for Trajectory control of Puma560", (*under writing*)
7. Mona Subramaniam A, Manju A and M.J.Nigam, "A Novel Stochastic Algorithm using Pythagorean mean for minimization", (*under writing*)

Appendix A.1: PUMA560 Dynamic parameters

For the dynamic equation $M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau$, the values of the mass inertia matrix, Coriolis matrix, gravitational matrix and the actuator torque limitations of PUMA560 are given in this section.

Mass inertia matrix

$$m_{11} = 2.57 + (1.38*c2*c2) + (0.3*s23*s23) + (0.744*c2*s23);$$

$$m_{12} = (0.69*s2) + (-0.134*c23) + (0.0238*c2);$$

$$m_{13} = (-0.134*c23) + (-0.00397*s23);$$

$$m_{14} = 0;$$

$$m_{15} = 0;$$

$$m_{16} = 0;$$

$$m_{21} = m_{12};$$

$$m_{22} = 6.79 + (0.744*s3);$$

$$m_{23} = 0.333 + (0.372*s3) + (-0.011*c3);$$

$$m_{24} = 0;$$

$$m_{25} = 0;$$

$$m_{26} = 0;$$

$$m_{31} = m_{13};$$

$$m_{32} = m_{23};$$

$$m_{33} = 1.16,$$

$$m_{34} = -0.00125*s4*s5;$$

$$m_{35} = 0.00125*c4*c5;$$

$$m_{36} = 0;$$

$$m_{41} = m_{14};$$

$$m_{42} = m_{24};$$

$$m_{43} = m_{34};$$

$$m44 = 0.2;$$

$$m45 = 0;$$

$$m46 = 0;$$

$$m51 = m15;$$

$$m52 = m25;$$

$$m53 = m35;$$

$$m54 = m45;$$

$$m55 = 0.18;$$

$$m56 = 0;$$

$$m61 = m16;$$

$$m62 = m26;$$

$$m63 = m36;$$

$$m64 = m46;$$

$$m65 = m56;$$

$$m66 = 0.19;$$

Coriolis matrix

$$\begin{aligned} \text{cor11} = & (-1.38*c1*s1* \text{qd1}) + 0.5* \text{qd2}*(0.6*s23*c23 - 0.744*s2*s23 + 0.744*c2*c23) \\ & + 0.5* \text{qd3}*(0.6*s23*c23 - 0.744*s2*s23 + 0.744*c2*c23); \end{aligned}$$

$$\begin{aligned} \text{cor12} = & 0.5* \text{qd1}*(0.6*s23*c23 - 0.744*s2*s23 + 0.744*c2*c23) + 0.5* \text{qd2}*(1.38*c2 + \\ & 0.268*s23 - 0.0476*s2) + 0.5* \text{qd3}*(0.268*s23 - 0.00397*c23); \end{aligned}$$

$$\begin{aligned} \text{cor13} = & 0.5* \text{qd1}*(0.6*s23*c23 + 0.744*c2*c23) + 0.5* \text{qd2}*(0.268*s23 - 0.00397*c23) + \\ & 0.5* \text{qd3}*(0.268*s23 - 0.00794*c23); \end{aligned}$$

$$\text{cor21} = 0.5* \text{qd1}*(-0.6*s23*c23 + 0.744*s2*s23 - 0.744*c2*c23) + 0.199* \text{qd3}*c23;$$

$$\text{cor22} = 0.372 * \text{qd3} * \text{c3};$$

$$\text{cor23} = 0.00199 * \text{qd1} * \text{c23} + 0.372 * \text{qd2} * \text{c3} + 0.5 * \text{qd3} * (0.744 * \text{c3} + 0.022 * \text{s3});$$

$$\text{cor31} = 0.5 * \text{qd1} * (-0.6 * \text{s23} * \text{c33} + 0.744 * \text{s2} * \text{s23} - 0.744 * \text{c2} * \text{c23}) + 0.00199 * \text{qd3} * \text{c33};$$

$$\text{cor32} = 0.372 * \text{qd3} * \text{c3};$$

$$\text{cor33} = 0.00199 * \text{qd1} * \text{c23} + 0.372 * \text{qd2} * \text{c3} + 0.5 * \text{qd3} * (0.744 * \text{c3} + 0.022 * \text{s3});$$

All other Coriolis matrix elements are zeros.

Gravity matrix

$$\text{g1} = 0;$$

$$\text{g6} = 0;$$

$$\text{g2} = -37.196 * \text{c2} - 8.445 * \text{s23} + 1.023 * \text{s2};$$

$$\text{g3} = -8.445 * \text{s23} + 1.023 * \text{c23} + 0.248 * \text{c23} * \text{c45} + \text{c5} * \text{s23};$$

$$\text{g4} = 0.028 * \text{s23} * \text{s4} * \text{s5};$$

$$\text{g5} = -0.028 * (\text{c23} * \text{s5} + \text{s23} * \text{c4} * \text{c5});$$

Actuator torque limitations

$$-97.6\text{Nm} \leq \tau_1 \leq 97.6\text{Nm}$$

$$-186.4\text{Nm} \leq \tau_2 \leq 186.4\text{Nm}$$

$$-89.4\text{Nm} \leq \tau_3 \leq 89.4\text{Nm}$$

$$-24.2\text{Nm} \leq \tau_4 \leq 24.2\text{Nm}$$

$$-20.1\text{Nm} \leq \tau_4 \leq 20.1\text{Nm}$$

$$-21.3\text{Nm} \leq \tau_5 \leq 21.3\text{Nm}$$

Appendix A.2: Default Genetic Algorithm settings

Genetic Algorithm settings used for simulation are as follows.

| | |
|---------------------|--------------------|
| Population size | 20 |
| Creation function | uniform |
| Scaling function | Rank |
| Selection function | stochastic uniform |
| Elite count | 2 |
| Crossover fraction | 0.8 |
| Mutation function | Gaussian |
| Crossover | scattered |
| Migration direction | forward |
| Migration fraction | 0.2 |
| Migration interval | 20 |

Appendix A.3: Generated Bivariate Polynomial coefficients

The Bivariate polynomial equation used in this thesis is given below, where 'x' corresponds to the error signal and 'y' corresponding to differential of error signal of the input signal. 'z' is the output of the bivariate polynomial equation.

$$\begin{aligned} z = & k_{01} + k_{02}x + k_{03}y + k_{04}x^2 + k_{05}x^3 + k_{06}x^4 + k_{07}x^5 + k_{08}x^6 + k_{09}x^7 \\ & + k_{10}x^8 + k_{11}x^9 + k_{12}x^{10} + k_{13}x^9y + k_{14}y^2 + k_{15}x^8y^2 + k_{16}x^8y \\ & + k_{17}x^7y^3 + k_{19}x^7y^2 + k_{20}x^7y + k_{21}x^6y^4 + k_{22}x^6y^3 + k_{23}x^6y^2 \\ & + k_{24}x^6y + k_{25}x^5y^5 + k_{26}y^3 + k_{27}y^4 + k_{28}y^5 + k_{29}y^6 + k_{30}y^7 \\ & + k_{31}y^8 + k_{32}y^9 + k_{33}x^5y^4 + k_{34}x^5y^3 + k_{35}x^5y^2 + k_{36}x^5y \\ & + k_{37}x^4y^6 + k_{38}x^4y^5 + k_{39}x^4y^4 + k_{40}x^4y^3 + k_{41}x^4y^2 + k_{42}x^4y \\ & + k_{43}x^3y^7 + k_{44}x^3y^6 + k_{45}x^3y^5 + k_{46}x^3y^4 + k_{47}x^3y^3 \\ & + k_{48}x^3y^2 + k_{49}x^3y + k_{50}x^2y^8 + k_{51}x^2y^7 + k_{52}x^2y^6 + k_{53}x^2y^5 \\ & + k_{54}x^2y^4 + k_{55}x^2y^3 + k_{56}x^2y^2 + k_{57}x^2y + k_{58}xy^9 + k_{59}xy^8 \\ & + k_{60}xy^7 + k_{61}xy^6 + k_{62}xy^5 + k_{63}xy^4 + k_{64}xy^3 + k_{65}xy^2 + k_{66}xy \\ & + k_{67}y^{10}; \end{aligned}$$

The coefficients corresponding to the individual joints are given below.

Joint 1:

$$[k_{01}, k_{02}, k_{03} \dots k_{67}] =$$

$$\begin{aligned} & [0.010273, 8.7365, 0.41901, -2.85871, 4.241, 1.067, 7.0696, 0.59125, -12.315, 3.6649, - \\ & 2.2925, -1.8594, 6.0014, 1.9456, -2.1864, 0.78301, -0.87877, 3.2321, -4.7972, 2.625, \\ & 0.58673, -6.5208, 6.5712, 5.5778, -2.3628, -3.3356, -0.58435, 2.889, 3.6344, 3.3768, \\ & 10.866, -2.2693, -1.8696, -0.90739, 0.9869, 6.7456, -4.887, 11.52, 1.5316, -5.7521, \\ & 0.1186, -2.5928, -1.5918, 3.5707, 7.0395, 4.418, -1.4488, -13.936, 0.97954, -1.3975, \\ & 6.1484, -0.41037, 1.5403, -3.019, -0.67027, 11.886, 3.2339, -0.94888, -7.207, -11.424, \\ & 2.0959, 3.9007, -0.35742, 0.99464, 6.6465, -2.7345, -1.4256] \end{aligned}$$

Joint 2:

[k01, k02, k03... k67] =

[0.12649, 2.6033, -0.262, -3.5966, 0.36729, -2.2843, 0.56984, -1.8815, 2.7008, -1.6964, 1.4381, 2.9295, -2.5835, -0.52444, -4.2097, 0.62167, -0.81735, -3.2166, -0.45181, 0.62772, 2.9052, 2.8338, 4.2895, 0.95544, -2.332, 1.7665, 1.5009, -2.1396, 0.036002, -0.11559, 0.068517, -1.0346, -3.3376, 1.6122, -0.71706, -2.5954, 2.8828, 0.10273, -0.99928, -1.7763, 2.2104, 0.43455, -2.6442, 1.7593, -2.1607, 1.5638, 4.4682, 3.302, 3.2059, -0.31763, 3.962, 1.6336, -5.2005, 0.58978, 2.9963, -6.1126, 3.0573, -0.57701, -5.1248, 0.025224, -3.1621, -1.3979, -0.44632, 0.011013, -0.93754, -2.0152, -0.23502]

Joint 3:

[k01, k02, k03... k67] =

[0.024088, 1.6854, 0.32903, -0.52849, 3.3456, 2.1944, 0.1832, -3.7514, -3.4691, 3.3083, 2.27, 5.7663, -5.765, 1.623, -5.7849, -3.8574, 7.7911, -0.51643, -1.5596, 0.34314, 1.804, -5.9742, -0.34928, -7.8166, 1.0588, 0.71373, -5.2484, 0.38928, -3.104, 7.1072, -5.6535, 0.024452, -6.7972, -0.46925, -3.0937, -1.3376, 1.3929, -4.7838, -4.4375, -2.4539, -3.7927, 11.327, 10.294, 2.6641, -7.6788, 1.3956, 4.7812, 3.0276, -7.369, -12.96, 3.1703, -0.33661, 3.5123, -7.5074, 3.1822, -3.5109, 0.82953, 3.336, -0.2182, -0.43886, 0.45385, -1.6026, 8.6831, -2.8549, 1.5969, -2.24, 2.018]

Joint 4:

[k01, k02, k03... k67] =

[0.054477, 0.43617, 1.1639, 0.073118, -4.1871, 1.2638, 2.5211, -0.57512, 0.054364, -0.67051, -0.14753, -0.40329, -4.1555, -0.50988, -1.3793, 3.9382, 0.24933, -1.428, 0.17447, -0.093334, 3.5392, -0.34272, 2.0568, 0.317, 1.6275, 1.8276, -0.033226, -0.63566, -1.3519, -0.47436, -3.3963, -1.4397, -1.7317, 1.7834, -3.5767, -3.8629, 6.5307, 3.9697, 1.3647, 1.3671, 4.842, 1.4482, -0.09259, -2.9185, 0.64998, 1.0115, -0.73422, 0.33906, 1.0492, -1.1322, -1.6762, 3.3944, 4.0375, -0.25804, 0.97167, 3.8698, -0.38227, -0.73333, 1.527, 0.22165, 1.88, 0.38171, -3.1831, 1.5559, 0.3509, 0.54328, -0.68954]

Joint 5:

[k01, k02, k03... k67] =

[0.0062423, 3.4804, -0.50602, 9.8647, -8.6976, 1.8486, 0.66285, 9.4293, 6.4105, -0.35998, -1.2956, -4.8808, -2.8242, 0.85335, 1.6532, -6.6514, 0.47995, 11.201, -6.8947, 6.4095, 3.554, 12.809, 6.3687, -18.201, -12.349, 8.3508, -9.299, 11.978, 3.7647, 0.27453, -4.367, -19.379, -1.8882, -8.0287, -4.9577, -2.711, 11.632, -0.37066, -1.7626, 4.5, -7.7601, 21.454, 4.775, 4.6865, 8.24, 4.1203, 16.918, -0.61899, 0.73133, -0.77481, -1.3856, -6.8247, -0.86129, -7.4409, 6.4234, 6.7247, 22.572, -3.4313, -6.9884, 4.4085, 4.4158, 4.7642, -0.41725, -10.915, -3.1059, 4.3855, -1.0823]

Joint 6:

[k01, k02, k03... k67] =

[0.018756, 6.5793, 0.81454, -6.6895, 2.056, 4.5492, -1.9612, -6.6103, -2.72, 1.3182, -2.4076, 1.2005, -8.2849, -0.1919, -5.6496, -3.6659, 2.4988, 2.2695, 0.64721, 1.189, 11.908, -9.0725, 8.0553, -7.1208, -4.2526, 2.1453, -4.7682, -3.3904, 2.8624, -8.3699, 0.95776, 2.9075, -4.5973, 9.0179, -0.32626, -8.3611, 12.642, 2.7972, -1.9189, -10.353, 3.9204, 4.4522, -0.82404, 6.1365, -1.9169, -2.2623, 9.5643, 0.43796, 3.0672, 3.333, 7.8186, -0.026312, 7.9934, 4.1317, -11.759, 5.7567, 6.3169, 6.9623, 10.968, -2.4448, -1.9186, 5.3805, -5.1223, -2.7587, -3.257, -8.4084, 2.9673]

Appendix A.4 Pseudo-random joint angle generation

A sequence of Pseudo random numbers is passed through a system with transfer function given chosen by trial and error. Transfer function chosen is $\frac{100}{s^2 + s + 100}$. When the pseudo random numbers are passed through this system a continuous pseudo-random signal is obtained which is given to the joint angles.

Time intervals at which the numbers in the sequence appears are [0 0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1], that is at time = 0 the first element in the sequence appears, at time = 0.1 the second element and so on.

Joint 1 Pseudo-random sequence: [0.1 0.1 0.2 0.3 0.4 0.2 0.4 0.2 -0.1 -0.2 0]

Joint 2 Pseudo-random sequence: [0 0.1 0.3 0.2 0.5 0.4 0.3 0.2 0.1 0 0]

Joint 3 Pseudo-random sequence: [-0.3 0.2 0.4 0.3 0.4 0.6 0.8 0.2 0.4 -0.2 0]

Joint 4 Pseudo-random sequence: [-0.2 -0.1 0.3 0.2 0.4 0.1 0.4 0.3 0.5 0.08 0]

Joint 5 Pseudo-random sequence: [0.2 -0.1 0.4 0.2 -0.4 -0.2 0.1 -0.3 -0.5 -0.08 0]

Joint 6 Pseudo-random sequence: [0.3 -0.5 0.4 -0.2 0.4 0.6 0.5 0.3 0.5 0.2 0]