

ROBOTIC CONTROL BY NEURO-FUZZY APPROACHES

A DISSERTATION

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

of

MASTER OF TECHNOLOGY

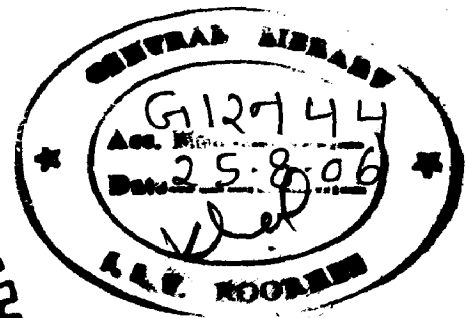
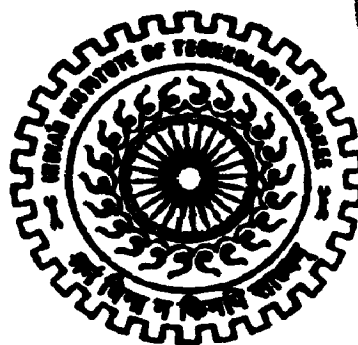
in

ELECTRICAL ENGINEERING

(With Specialization in System Engineering and Operations Research)

By

G.E.NAGANNA



**DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE-247 667 (INDIA)**

JUNE, 2006

CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation entitled "ROBOTIC CONTROL BY NEURO-FUZZY APPROACHES" towards partial fulfillment of the requirements for the award of the degree of **Master of Technology in Electrical Engineering** with specialization in **System Engineering and Operations Research**, submitted to Electrical Engineering Department, Indian Institute of Technology Roorkee, Roorkee, is an authentic record of my own work carried out from July 2005 to June 2006, under the guidance of **Dr. Surendra Kumar**, Assistant Professor, Department of Electrical Engineering, IIT Roorkee.


The matter embodied in this dissertation has not been submitted for the award of any other degree or diploma.

Date: 26/06/06
Place: Roorkee

G. E. Naganna
(G.E.NAGANNA)

CERTIFICATE

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


(Dr. Surendra Kumar) 28/6/2006
Assistant Professor

Electrical Engg Department,
I.I.T. Roorkee
Roorkee- 247667
INDIA

ACKNOWLEDGEMENT

Though the deepest gratitude can only be felt inside my heart, but in words with my deepest esteem I wish to express my deep sense of gratitude and sincere thanks to my beloved guide **Dr.Surendra Kumar** Department of Electrical Engineering, IIT Roorkee, for being helpful and a great source of inspiration. His keen interest and constant encouragement gave me the confidence to complete my thesis work successfully. This work is simply the reflection of his thoughts, ideas, and concepts. I am highly indebted to him for his kind and valuable suggestions and of course his valuable time during the period of this work. The huge quantum of knowledge I had gained during his inspiring guidance would be immensely beneficial for my future endeavors.

I am very thankful to **Prof. S.P.Gupta**, head of the Electrical Engineering Department, for supporting my effort.

I thank all the teaching and non teaching staff members of the department who have contributed directly or indirectly in successful completion of my dissertation work.

I also avail this opportunity to thank all my friends for their continuous support and enthusiastic help and encouragement.

Finally, I would like to say that I am indebted to my parents for everything that they have given to me. I thank them for the sacrifices they made so that I could grow up in a learning environment. They have always stood by me in everything I have done, providing constant support, encouragement and love.

G. E. Naganna
(G.E.NAGANNA)

ABSTRACT

The problem of manipulator control is highly complex problem of controlling a system which is multi-input, multi-output, and non-linear and time variant. A number of different approaches presently followed for the control of manipulator vary from PID to very complex, intelligent, self-learning control algorithms.

This report presents a comparative study of simulated performance of some conventional controllers, like the simple PID, Computed torque control, Feed forward inverse dynamic control and critically damped inverse dynamic control and some Intelligent controllers, like Fuzzy control, Neural control, and Neuro-Fuzzy control. IAE is used for comparison as performance index.

The study concludes that the Critically damped inverse dynamics controller in general performs better than rest of conventional controllers. When the unmodeled term is added to the model, PID and Feed forward inverse dynamic control perform badly. Computed torque control and Critically damped inverse dynamics control performance also effected but they do well. A Neuro-Fuzzy controller combines the advantage of neural networks (learning adaptability) with the advantage of fuzzy logic (use of expert knowledge) to achieve the goal of robust adaptive control of robot dynamics, performs better in intelligent controllers and also shows that intelligent controllers are better even when unmodeled terms are added to the model.

CONTENTS

	<i>Candidates Declaration</i>	i
	<i>Acknowledgement</i>	ii
	<i>Abstract</i>	iii
	<i>Contents</i>	iv
	<i>List of Figures</i>	vi
	<i>List of Tables</i>	viii
Chapter 1	Introduction	1
	1.1 Introduction To Robot Control	1
	1.2 Literature Review on Conventional Controllers	3
	1.3 Literature Review On Fuzzy Control	3
	1.4 Literature Review On Neural Control	4
	1.5 Literature Review On Neuro-Fuzzy Control	4
	1.6 Organization of the Dissertation	5
Chapter 2	Dynamics Of Robotic Manipulator	6
	2.1 Introduction	6
	2.2 Puma Robot	6
	2.3 Dynamics Of Puma560 Robot	7
	2.4 Actuator For The Robot	9
Chapter 3	Conventional Controllers	12
	3.1 Introduction	12
	3.2 PID control	13
	3.3 Feed forward inverse dynamics control	13
	3.4 Computed torque control	14
	3.5 Critically damped inverse dynamics control	15
Chapter 4	Fuzzy Control	16
	4.1 Introduction	16
	4.2 Historical Background	16
	4.3 Structure Of Fuzzy Controller	17
	4.3.1 Preprocessing	17

	4.3.2	Fuzzification	18
	4.3.3	Rule Base	18
	4.3.4	Inference Engine	19
	4.3.5	Defuzzification	22
	4.3.6	Post Processing	23
Chapter 5		Neural Control	24
	5.1	Introduction	24
	5.2	The Back-Propagation Algorithm	25
	5.3	Neural Network Based Control	29
Chapter 6		Neuro-Fuzzy Systems	31
	6.1	Introduction	31
	6.2	Adaptive Networks: Architectures And Learning Algorithms	33
	6.3	ANFIS: Adaptive-Network-Based Fuzzy Inference System	34
Chapter 7		Design and Simulation in Simulink/Matlab7.01	39
	7.1	Introduction	39
	7.2	Puma560 Robot Model	39
	7.3	Actuator Model	42
	7.4	Design of PID Control	43
	7.5	Design of Feed Forward Inverse Dynamics Control	44
	7.6	Design of Computed Torque Control	45
	7.7	Design of Critically Damped Inverse Dynamics Control	46
	7.8	Design of Fuzzy PD+I Control	47
	7.9	Design of Neural Control	49
	7.10	Design of Neuro-Fuzzy Control	51
Chapter 8		Results	53
	8.1	Error Profiles of The Robot for Trajectory Control	53
	8.2	Effect of Unmodeled Term on Performance	56
Chapter 9		Conclusions	60
		References	62

LIST OF FIGURES

Fig. 2.1 (a) puma 560 robot	7
Fig. 2.1 (b). Illustration of Puma 560 robot	7
Fig. 3.1 General structure of robot control system	12
Fig. 3.2 Feed forward inverse dynamics controller	14
Fig. 4.1 Blocks of a fuzzy controller	17
Fig. 4.2 General step response	18
Fig. 4.3 Graphical construction of the control signal in a fuzzy PD controller	20
Fig. 4.4 One input, one output rule base with non-singleton output sets	21
Fig. 5.1 Neural network architecture	27
Fig. 5.2 Neural controller	29
Fig. 5.3 Scheme for learning dynamic model	30
Fig. 6.1 A fuzzy system whose membership functions are adjusted by a neural network.	31
Fig. 6.2 A fuzzy system defined by a neural network	32
Fig. 6.3 Fuzzy neurons of a neural network	32
Fig. 6.4 A fuzzy system with neural network rule base	33
Fig. 6.5(a) fuzzy reasoning ;(b) equivalent ANFIS	34
Fig. 6.6 (a) fuzzy reasoning ;(b) equivalent ANFIS	37
Fig. 6.7 (a) 2-input ANFIS with 9 rules;(b) corresponding fuzzy subspaces	38
Fig. 7.1 PUMA560 simulink model	39
Fig. 7.2 dynamics matrices	40
Fig. 7.3 gravity load	41
Fig. 7.4 Matrix multiplications	42
Fig. 7.5 Actuator simulink model	42
Fig. 7.6 Total system with PID controller	44
Fig. 7.7 Total system with FFID controller	44
Fig. 7.8 Robot Response with PID control and FFID control	45
Fig. 7.9 Input block for CTC method	45
Fig. 7.10 Input block for CDID method	46

Fig. 7.11 Robot Response with CTC control and CDID control	46
Fig. 7.12 Fuzzy controls with gains	47
Fig. 7.13 Total system with fuzzy controller	48
Fig. 7.14 Fuzzy controller	48
Fig. 7.15 Total robot system with neural controller	49
Fig. 7.16 Neural Network1 block	50
Fig. 7.17 Robot response with Fuzzy control and Neural control	50
Fig. 7.18 Total robot system with ANFIS controller	51
Fig. 7.19 ANFIS controller	52
Fig.7.20Robot Response with Neuro-Fuzzy control and membership functions after training	52
Fig. 8.1-8.6 Error profiles of conventional control strategies without unmodeled term at each joint	53-54
Fig. 8.7-8.12 Error profiles of intelligent control strategies without unmodeled term at each joint	55
Fig. 8.13-8.18. Error profiles of PID, FFID, and Neural controllers with unmodeled term at each joint after network is trained to approximate inverse dynamics of robot with unmodeled term	58

LIST OF TABLES

Fuzzy rules	47
Errors (in rad) of Conventional Controllers without unmodeled term	54
Errors (in rad) of Conventional Controllers with unmodeled term	56
Errors (in rad) of Intelligent Controllers without unmodeled term	56
Errors (in rad) of Intelligent Controllers with unmodeled term	57
Errors (in rad) of PID, FFID, and Neural control before network trained to system with unmodeled term	57
Errors (in rad) of PID, FFID, and Neural control after network trained to system with unmodeled term	59

1. INTRODUCTION

1.1 INTRODUCTION TO ROBOT CONTROL

Robotic manipulators have become increasingly important in the field of flexible automation. Robotic manipulators are very complicated nonlinear systems [1, 2]. A robot is typically modeled as a chain of n rigid bodies. In general, one end of the chain is fixed to some reference surface while other end is free, thus forming an open kinematics chain of moving rigid bodies. Dynamics of a manipulator involve nonlinear mapping between applied joint torques and joint positions, velocities and accelerations. These relationships can be described by a set of second-order nonlinear and highly coupled differential equations [3] with uncertainty as a robot work under unknown and changing environments in executing different tasks.

There are many control strategy that can be applied for control of robotic manipulators. These range from conventional to adaptive and intelligent controllers [6]. The inverse dynamic approach is particularly important for control of robots and can be used to compensate for highly coupled and nonlinear arm dynamics. Many strategies have been developed for controlling the motion of a robot. Existing robotic manipulators use simple proportional-(integral)-differential controllers with the gains tuned for critical damping. The advantages of a PID controller include its simple structure along with roust performance in a wide range of operating conditions. A lot of research has been done on PID control scheme and available methods for tuning PID gains are advanced and accurate. This makes the PID as one of the most favored control strategy. However, the design of a PID controller is generally based on the assumption of exact knowledge about the system. This assumption is often not valid since the development of any practical system may not include precise information of factors such as friction, backlash, unmodeled dynamics and uncertainty arising from any of the sources. Advanced modern approaches to the design of controllers for robots includes computed torque control, robust control, model based adaptive control and variable structure control. However, most of these are too complicated and expensive for industrial use. A heavy computational burden prevents them being employed for real-time control applications. Also, some of them need an accurate dynamic model which is not always

available especially when robot is performing under different operating conditions. In order to overcome above problems, intelligent controlling techniques are used.

Intelligent control is a control technology that replaces the human mind in making decisions, planning control strategies, and learning new functions whenever the environment does not allow or does not justify the presence of human operator. Artificial neural networks and fuzzy logic are potential tools for intelligent control engineering. Intelligent controllers are, Fuzzy logic, Neural based control, and Hybrid control (Neuro-Fuzzy). Neural networks are best known for their learning capabilities. Fuzzy logic is a method of using human skills and thinking processes in a machine.

The underlying idea of fuzzy control is to build a model of a human expert who is capable of controlling the plant without thinking in terms of a mathematical model. The control expert specifies the control actions in the form of linguistic rules. The specification of good linguistic rules depends on the knowledge of the control expert, but the translation of these rules into fuzzy set theory framework is not formalized and arbitrary choices concerning, for example, the shape of membership functions have to be made. The quality of a fuzzy logic controller can be drastically affected by the choice of membership functions. Thus, methods for tuning fuzzy logic controllers are necessary. Neural networks offer the possibility of solving the problem of tuning. A combination of neural networks and fuzzy logic offers the possibility of solving tuning problems and design difficulties of fuzzy logic. The resulting network can be easily recognized in the form of fuzzy logic control rules. This new approach combines the well-established advantages of both the methods and avoids the drawbacks of both. The computation of control value from the given measured input value is seen as a feed forward procedure as in layered networks, where the inputs are forwarded through the network resulting in some output value(s). If the actual output value differs from the desired output value, the resulting error is propagated back through the architecture, which in turn results in modification of certain parameters and reduction in error during the next cycle. Interpreting the fuzzy controller as a neural network helps in training the fuzzy controller with learning procedures and the modified structure can still be interpreted as fuzzy logic controller.

1.2 LITERATURE REVIEW ON CONVENTIONAL CONTROLLERS

C.G. Atkeson, J.D.Griffiths, J.M.Hollerbach, C.H.An, proposed the controller's range from PD control applied independently at each joint to feed forward and computed torque methods incorporating full dynamics. Study shows that dynamic compensation by model based controller can improve trajectory accuracy significantly [7].

Sudeept Mohan, Surekha Bhanot presented a comparative study of simulated performance of some conventional algorithms, like simple PID, Feed forward inverse dynamics, computed torque control, and critically damped inverse dynamics. Study shows that the critically damped inverse dynamics controller in generally performs better then the rest of algorithms particularly when the uncertainty of the system increases [8].

D.P.kwok, T.P.Leung, Fang Sheng described the use of Genetic Algorithms (GAS) for optimizing the parameters of PID controllers for a 6-DOF PUMA 560 robot arm. The simulation results obtained are compared with that obtained by traditional optimization techniques, wherever applicable and showed that the GA-based optimal-tuning technique can work effectively and efficiently and has great potential to become a common optimal-tuning approach for the robot arm controllers [9].

1.3 LITERATURE REVIEW ON FUZZY CONTROL

Han-Xiong Li, H.B.Gatland explain systematic analysis and design of the conventional fuzzy control. General robust rule bases is proposed for fuzzy two-term control, and leave the optimum tuning to the scaling gains, which greatly reduces the difficulties of design and tuning. The digital implementation of fuzzy control is also presented for avoiding the influence of the sampling time [10].

T.Brehm, K.S.Rattan proposed a hybrid fuzzy PID controller which takes advantage of the properties of the fuzzy PI and PD controllers and compared Fuzzy PID and Hybrid Fuzzy PID in terms of rule base, design and implementation problems [11].

G.M.Khoury, M.Saad, H.Y.Kanaan, and C.Asmar presented elaboration of fuzzy control laws based on two structures of coupled rules fuzzy PID controllers and compared the Two-input FLC with coupled rule, Three-input FLC with coupled rule, computed torque control, and direct adaptive control method on a five-DOF robot arm in terms position tracking errors [12].

Abdollah Homaifar, Ed McCormick examines the applicability of genetic algorithms (GA's) in the simultaneous design of membership functions and rule sets for fuzzy logic controllers. This new method has been applied to two problems, a cart controller and a truck controller. Beyond the development of these controllers, they also examine the design of a robust controller for the cart problem and its ability to overcome faulty rules [13].

1.4 LITERATURE REVIEW ON NEURAL CONTROL

T.Ozaki, T.Suzuki, T.Furuhashi, S.Okuma, and Y.Uchikawa presents a nonlinear compensator using neural networks for trajectory control of robotic manipulator, proposed a model learning scheme, and the adaptive capability of the neural network controller to compensate unstructured uncertainties is clarified [14].

P.Gupta and N.K. Sinha proposed a Neural-network based control for robotic manipulator and presented a method for improving the learning ability of neural-networks by using a function with changeable shape. Results show that proposed controller has better robustness [15].

M.F.Moller introduced a supervised learning algorithm (Scaled Conjugate Gradient, SCG) with super linear convergence rate. The algorithm is based upon a class of optimization techniques well known in numerical analysis as the Conjugate Gradient Methods. SCG uses second order information from the neural network but requires only $O(N)$ memory usage, where N is the number of weights in the network [16].

1.5 LITERATURE REVIEW ON NEURO-FUZZY CONTROL

J.-S.R.Jang has proposed a novel approach to the design of fuzzy controllers without resorting to domain knowledge of the plant under control. He employed the adaptive networks as building blocks and the back-propagation-type gradient method as a learning procedure to minimize the differences between the actual state and desired state at each time step [17].

Jyh-Shing, Roger Jang have proposed the architecture and learning procedure underlying ANFIS (adaptive-network-based fuzzy inference system) which is a fuzzy inference system implemented in the framework adaptive networks. Hybrid learning is used. The

ANFIS architecture is employed to model nonlinear functions, identify nonlinear components on-line in a control system, and predict chaotic time series, all yielding remarkable results [18].

G.S.Sandhu and K.S.Rattan have presented a general Neuro-Fuzzy controllers which combines the neural networks and Fuzzy logic to solve the problem of tuning fuzzy logic controllers. The Neuro-Fuzzy controller uses the neural-network learning techniques to tune the membership functions while keeping semantics of the fuzzy logic controller intact [19].

Manish Kumar and Devendra P.Garg have proposed intelligent learning of Fuzzy logic controllers via neural network and genetic algorithm. The results show that Genetic-Fuzzy and Neuro-Fuzzy approaches were able to learn rule base and identify membership function parameters accurately [20].

Teo Lian Seng, Marzuki Bin Khalid presented a Neuro-Fuzzy controller where all of its parameters can be tuned simultaneously by Genetic algorithm. The performance of the proposed controller is compared with a conventional fuzzy controller and a PID controller tuned by Genetic algorithm and results show that proposed controller offers encouraging advantages and has better performance [21].

1.6 ORGANIZATION OF THE DISSERTATION

This Dissertation is organized as follows.

Chapter 2 details the Dynamics of 6 DOF PUMA 560 Robot and Actuator of Robot

Chapter 3 Explains Conventional Control strategies.

Chapter 4 Explains the Fuzzy Logic Control.

Chapter 5 Explains the Back Propagation algorithm and Neural Network-based control.

Chapter 6 Explains the Neuro-fuzzy systems architecture and learning algorithms.

Chapter 7 Deals the Design and implementation of Robot, Conventional controllers and intelligent controllers in Simulink/Matlab 7.01.

Chapter 8 Results.

Chapter 9 Conclusions.

2.DYNAMICS OF ROBOTIC MANIPULATOR

2.1 INTRODUCTION

The manipulator system is a classic control problem that is used industries around the world. It is a suitable process to test prototype controllers due to its high nonlinearities and lack of stability. In this chapter, the dynamical equations of the system will be derived, the model will be developed in simulink and basic controllers will be developed. The aim of developing a Robot system in simulink is that the developed model will have the same characteristics as the actual process. It will be possible to test each of the prototype controllers in the simulink environment. Before the robot model can be developed in simulink, the system dynamical equations will be derived using 'Lagrange Equations'. [1, 2] The Lagrangian equations are one of many methods of determining the system equations. Using this method it is possible to derive dynamical system equations for a complicated mechanical system such as the Robot manipulator. The Lagrange equations use the kinetic and potential energy in the system to determine the dynamical equations of the robot system.

2.2 PUMA ROBOT

The PUMA robot was initially built by the Unimation Inc. (now defunct), to specifications developed by General Motors. PUMA stands for Programmable Universal Machine for Assembly [3]. This robot system was the first commercially available industry robot. It had all electric drives, and a reasonably sophisticated controller. Its controller could be disconnected and replaced by another custom-built controller. For these reasons, PUMA became one of the most popular with robotic researchers around the world. Figure.2.1 shows the PUMA robot

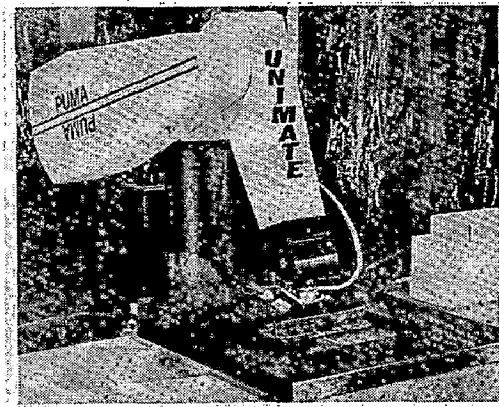


FIGURE 3.17 The Unimate PUMA 560.

Fig. 2.1 a. puma 560 robot

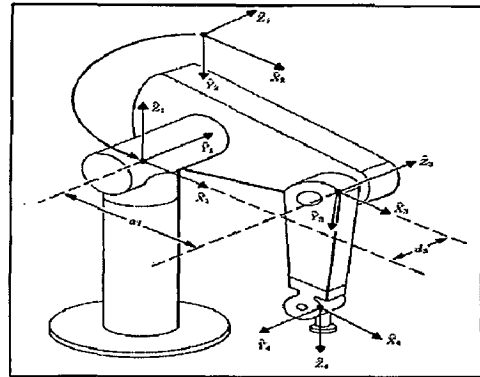


FIGURE 3.18 Some kinematic parameters and frame assignments for the PUMA 560 manipulator.

b. Illustration of Puma 560 robot

2.3 DYNAMICS OF PUMA560 ROBOT

The general form of the robot arm dynamic equation is

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) = \tau \quad \dots 2.1$$

Where q is angle matrix

$M(q)$ is mass matrix

$C(q, \dot{q})$ is coriolis/centrifugal matrix

$g(q)$ is gravity matrix

Puma is six degree of freedom robot so all matrices for puma robot are the order of six by six. For the simplicity these equations are abbreviated as per listed below

$$c_1 = \cos(q(1)), c_2 = \cos(q(2)), c_3 = \cos(q(3)), c_4 = \cos(q(4)), c_5 = \cos(q(5)),$$

$$c_6 = \cos(q(6))$$

$$s_1 = \sin(q(1)), s_2 = \sin(q(2)), s_3 = \sin(q(3)), s_4 = \sin(q(4)), s_5 = \sin(q(5)), s_6 = \sin(q(6))$$

$$s_{23} = \sin(q(2)+q(3))$$

$$c_{23} = \cos(q(2)+q(3))$$

MASS MATRIX

$$m_{11} = 2.57 + (1.38*c_2*c_2) + (0.3*s_{23}*s_{23}) + (0.744*c_2*s_{23})$$

$$m_{12} = (0.69*s_2) + (-0.134*c_{23}) + (0.0238*c_2)$$

$$m_{13} = (-0.134*c_{23}) + (-0.00397*s_{23})$$

$$m_{22} = 6.79 + (0.744*s_3)$$

$$m_{23} = 0.333 + (0.372*s_3) + (-0.011*c_3)$$

$$m_{33} = 1.16$$

$$m_{34} = -0.00125*s_4*s_5$$

$$m_{35} = 0.00125*c_4*c_5$$

$$m_{44} = 0.2$$

$$m_{55} = 0.18$$

$$m_{66} = 0.19$$

The mass matrix is symmetric so other equations are

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

$$m_{62} = m_{26}$$

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

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

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

$$m_{53} = m_{35}$$

$$m_{51} = m_{15} \quad \text{and}$$

$$m_{63} = m_{36}$$

$$m_{61} = m_{16}$$

$$m_{54} = m_{45}$$

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

$$m_{64} = m_{46}$$

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

$$m_{65} = m_{56}$$

$$m_{52} = m_{25}$$

All other elements in the mass matrix are zero

CORIOLIS/CENTRIFUGAL TORQUES MATRIX

$$\begin{aligned} \text{cor}_{11} = & (-1.38*c_1*s_1*\ddot{q}_1) + \\ & 0.5*\ddot{q}_2*(0.6*s_{23}*c_{23} - 0.744*s_2*s_{23} + 0.744*c_2*c_{23}) + \\ & 0.5*\ddot{q}_3*(0.6*s_{23}*c_{23} - 0.744*s_2*s_{23} + 0.744*c_2*c_{23}) \end{aligned}$$

$$\begin{aligned} \text{cor}_{12} = & 0.5*\ddot{q}_1*(0.6*s_{23}*c_{23} - 0.744*s_2*s_{23} + 0.744*c_2*c_{23}) + \\ & 0.5*\ddot{q}_2*(1.38*c_2 + 0.268*s_{23} - 0.0476*s_2) + \\ & 0.5*\ddot{q}_3*(0.268*s_{23} - 0.00397*c_{23}) \end{aligned}$$

$$\begin{aligned} \text{cor}_{13} = & 0.5*\ddot{q}_1*(0.6*s_{23}*c_{23} + 0.744*c_2*c_{23}) + \\ & 0.5*\ddot{q}_2*(0.268*s_{23} - 0.00397*c_{23}) + \\ & 0.5*\ddot{q}_3*(0.268*s_{23} - 0.00794*c_{23}) \end{aligned}$$

$$\begin{aligned} \text{cor}_{21} = & 0.5*\ddot{q}_1*(-0.6*s_{23}*c_{23} + 0.744*s_2*s_{23} - 0.744*c_2*c_{23}) + \\ & 0.199*\ddot{q}_3*c_{23} \end{aligned}$$

$$\text{cor}_{32} = 0.372 * \dot{q}_3 * c_3$$

$$\text{cor}_{23} = 0.00199 * \dot{q}_1 * c_{23} + 0.372 * \dot{q}_3 * c_3 + 0.5 * \dot{q}_3 * (0.744 * c_3 + 0.022 * s_2)$$

$$\text{cor}_{31} = 0.5 * \dot{q}_1 * (-0.6 * s_{23} * c_{23} + 0.744 * s_2 * s_{23} - 0.744 * c_2 * c_{23}) + 0.00199 * \dot{q}_3 * c_{23}$$

$$\text{cor}_{33} = 0.372 * \dot{q}_3 * c_3$$

$$\text{cor}_{22} = 0.00199 * \dot{q}_1 * c_{22} + 0.372 * \dot{q}_3 * c_3 + 0.5 * \dot{q}_3 * (0.744 * c_3 + 0.022 * s_2)$$

All other coriolis matrix elements are zero

GRAVITY MATRIX

$$g_1 = 0, g_6 = 0$$

$$g_2 = -37.196 * c_2 - 8.445 * s_{23} + 1.023 * s_2$$

$$g_3 = -8.445 * \sin_{23} + 1.023 * \cos_{23} + 0.248 * \cos_{23} * \cos_{45} + \cos_5 * \sin_{23}$$

$$g_4 = 0.028 * \sin_{23} * \sin_4 * \sin_5$$

$$g_5 = -0.028 * (\cos_{23} * \sin_5 + \sin_{23} * \cos_4 * \cos_5)$$

All these information and derivation of these equations for PUMA560 given in [3]

2.4 ACTUATOR FOR THE ROBOT

This section describes the method that is used to build the actuator model in a single joint of a robot arm, assuming that the robot is electrically actuated. An analytical description of a DC actuator has been well established in the literature [4].

The torque produced by a DC motor is proportional to the armature current when the motor is operated in its linear range

$$\tau_{mi} = k_{mi} i_{ai} \quad \dots 2.2$$

Where k_{mi} is known as the motor-torque proportional constant in $N\text{-m}/A$.

When the motor is rotating it acts like a generator and a voltage develops across the armature. This voltage is called back electromotive force (emf), which is proportional to a given armature, angular velocity as:

$$e_{bi} = k_{bi} \dot{\theta}_{mi} \quad \dots 2.3$$

$$e_{bi} = k_{bi} \dot{\theta}_{mi} \tau_{mi} = k_{mi} i_{ai} \quad \dots 2.4$$

Where k_{bi} is proportionality constant in $V \cdot S/rad$.

For puma robot $k_{mi} = k_{bi}$

An armature-control led DC motor circuit can be described by a first-order differential equation given by

$$v_{ai} = e_{bi} + i_{ai} R_{ai} + L_{ai} \frac{di_{ai}}{dt} \quad \dots 2.5$$

By solving we will get

$$i_{ai} = \frac{1}{L} \int_{-\infty}^t (v_{ai} - e_{bi} - i_{ai} R_{ai}) dt \quad \dots 2.6$$

SPECIFICATION FOR THE ACTUATOR

$R_{a1} = 2.1 \text{ Ohm}$	$k_{m1} = 0.189 \text{ N-m/A}$
$R_{a1} = 2.1 \text{ Ohm}$	$k_{m2} = 0.219 \text{ N-m/A}$
$R_{a1} = 2.1 \text{ Ohm}$	$k_{m3} = 0.202 \text{ N-m/A}$
$R_{a1} = 6.7 \text{ Ohm}$	$k_{m4} = 0.075 \text{ N-m/A}$
$R_{a1} = 6.7 \text{ Ohm}$	$k_{m5} = 0.066 \text{ N-m/A}$
$R_{a1} = 6.7 \text{ Ohm}$	$k_{m6} = 0.066 \text{ N-m/A}$

Armature inductance for the PUMA 560 robot is very low

L for all joints is approximately equals to 1 mH

Maximum and minimum torque that should given to the robot is

$$\begin{aligned} -97.6N - m &\leq \tau_1 \leq 97.6N - m \\ -186.4N - m &\leq \tau_2 \leq 186.4N - m \\ -89.4N - m &\leq \tau_3 \leq 89.4N - m \\ -24.2N - m &\leq \tau_4 \leq 24.2N - m \\ -20.1N - m &\leq \tau_5 \leq 20.1N - m \\ -21.3N - m &\leq \tau_6 \leq 21.3N - m \end{aligned}$$

Maximum and minimum controlled voltage that should be given to the actuator is

For all actuators

$$v_{\max} = 40 \text{volts}$$

$$v_{\min} = -40 \text{volts}$$

Details are given in [3, 4, and 5]

3. CONVENTIONAL CONTROLLERS

3.1 INTRODUCTION

There are many control strategies that can be applied for control of robot arm. These strategies are conventional, adaptive and intelligent control strategies. The general structure of a robot manipulator with controller is shown in figure 3.1 below. The trajectory generator provides the controller with information about the desired position, velocity and acceleration ($\theta_d, \dot{\theta}_d, \ddot{\theta}_d$) for each joint and keeps updating this information at the path update rate. The controller takes this information and compares it with the present (actual) position and velocity (sometimes acceleration also) of joints ($\theta, \dot{\theta}, \ddot{\theta}$), which are provided as feedback through the sensors.

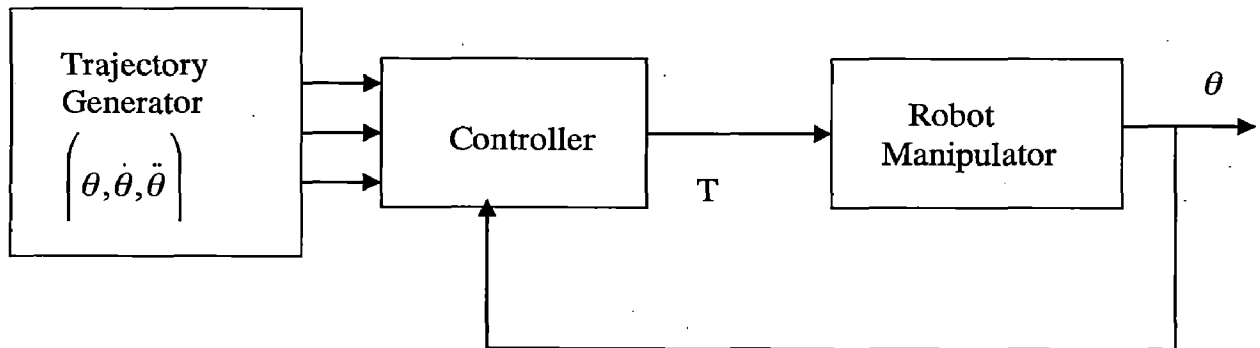


Fig .3.1 General structure of robot control system

Based upon the error between the desired and actual values, the controller calculates a vector of torques (τ), which should be applied at respective joints by the actuators to minimize these errors. The torques is calculated using control law. The goal of the

controller is thus, minimization of error, e and its first derivative \dot{e} . The dynamic model of Robotic manipulator can be described in the form of equation as below

$$\tau(t) = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) \quad \dots 3.1$$

Where, $M(\theta)$ is the inertia matrix, $C(\theta, \dot{\theta})$ is the centripetal-coriolis matrix, and $G(\theta)$ is the Gravity vector, θ is Joint angles, τ is the joint actuator torque. The use of linear control techniques for any system is valid only when the system to be controlled can be modeled by linear differential equations. Thus the linear control of robot manipulators is essentially an approximation, as the manipulator dynamics is described by highly non-linear equations. The linear control strategies for robots give excellent performance for manipulators having highly geared joints. This is the case with most of the industrial robots in use today.

3.2 PID Control

One common linear control strategy is PID (proportional-derivative and integral) control. The control law used for this strategy is given by

$$\tau_{PID} = K_D \dot{e} + K_P e + K_I \int e dt \quad \dots 3.2$$

K_D , K_I and K_P are the controller gain matrices. τ_{PID} is the vector of joint torques. It is possible to get the desired performance from the system by choosing the appropriate values of parameters of PID controller. Hand tuning method is used for selection of PID control gains. A robotic control system cannot be allowed to have an oscillatory response for obvious reasons. For instance, in a pick- n -place operation, an oscillating end-effector may strike against the object before picking it to manipulate. Hence, highest possible speed of response and yet non-oscillatory response, dictates that the controller design parameters shall be chosen to have the damping ratio equal to unity or least close to it but less than unity.

3.3 Feed Forward inverse dynamics control

Feed forward inverse dynamics control is a model based non-linear technique. Scheme for feed forward inverse dynamics control is shown below Fig 3.2. This scheme uses the inverse dynamics equations of robot manipulator in feed forward mode. As can be seen

from this figure, the sum of the outputs of the inverse model and feedback controller (i.e. PID Controller) will be the actual input torque to robot.

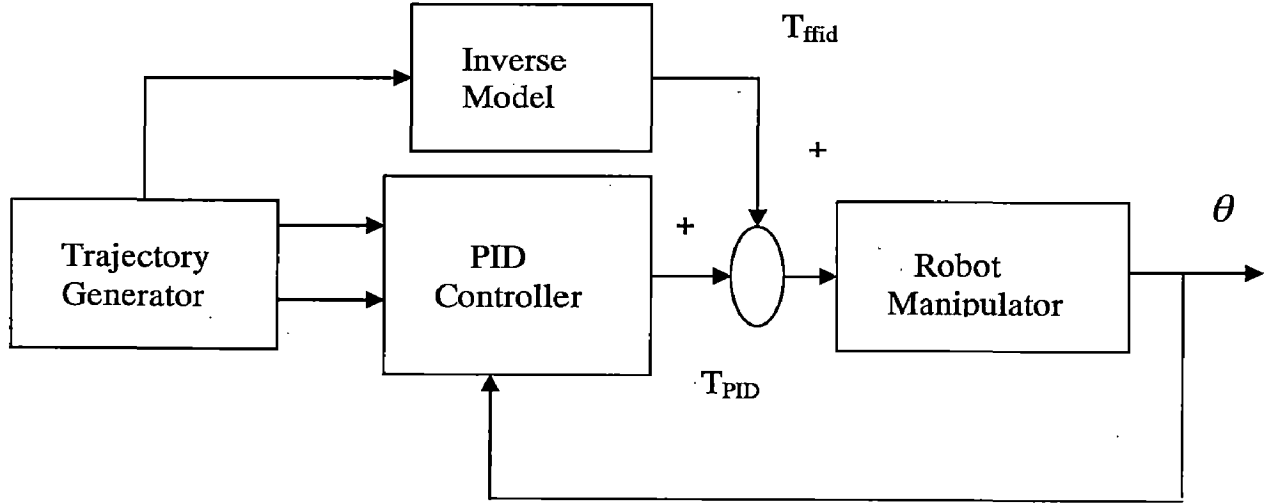


Fig.3.2 Feed forward inverse dynamics controller

In this strategy the torque is calculated as

$$\tau_{ffid} = M(\theta)\ddot{\theta} + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) \quad \dots 3.3$$

$$\tau_{PID} = K_D \dot{e} + K_P e + K_I \int e dt \quad \dots 3.4$$

Total control torque is $\tau = \tau_{PID} + \tau_{ffid}$. The feedback controller plays a role in making the whole system stable.

3.4 Computed Torque Control

The most common non-linear control technique for manipulator control is the Computed torque control. Scheme is similar to feed forward inverse dynamic control. Here the computed torque is given by

$$\tau_{CTC} = \tau_{PID} + M(\theta) \left[\ddot{\theta} + K_D \dot{e} + K_P e \right] + C(\theta, \dot{\theta})\dot{\theta} + G(\theta) \quad \dots 3.5$$

If the manipulator model is known exactly then this scheme results in asymptotically stable and provides asymptotically exact tracking

3.5 Critically damped inverse dynamics control

This control strategy is almost same as inverse dynamics except that the feed forward torque is calculated using reference velocity and reference acceleration instead of the desired values. These reference values are defined as

$$\begin{aligned}\dot{\theta}_R &= \dot{\theta}_d + K_P (\theta_d - \theta) \\ \ddot{\theta}_R &= \ddot{\theta}_d + K_D (\dot{\theta}_d - \dot{\theta})\end{aligned}\quad \dots 3.6$$

In this strategy the torque is calculated as

$$\tau_{CDID} = \tau_{PID} + M(\theta_R)\ddot{\theta}_R + C(\theta_R, \dot{\theta}_R)\dot{\theta}_R + G(\theta_R) \quad \dots 3.7$$

4. FUZZY CONTROL

4.1 INTRODUCTION

Due to continuously developing automation systems and more demanding Control performance requirements, conventional control methods are not always adequate. On the other hand, practical control problems are usually imprecise. The input output relations of the system may be uncertain and they can be changed by unknown external disturbances. New schemes are needed to solve such problems. One such an approach is to utilize fuzzy control.

Fuzzy control is based on fuzzy logic, which provides an efficient method to handle in exact information as basis reasoning. With fuzzy logic it is possible to convert knowledge, which is expressed in an uncertain form, to an exact algorithm. In fuzzy control, the controller can be represented with linguistic if-then rules. The interpretation of the controller is the fuzzy but controller is processing exact input-data and is producing exact output-data in a deterministic way.

4.2 HISTORICAL BACKGROUND

Since the introduction of the theory of fuzzy sets by L. A. Zadeh in 1965, and the industrial application of the first fuzzy controller by E.H. Mamadani in 1974, fuzzy systems have obtained a major role in engineering systems and consumer's products in 1980s and 1990s. New applications are presented continuously.

A reason for this significant role is that fuzzy computing provides a flexible and powerful alternative to contract controllers, supervisory blocks, computing units and compensation systems in different application areas. With fuzzy sets very nonlinear control actions can be formed easily. The transparency of fuzzy rules and the locality of parameters are helpful in the design and maintenances of the systems. Therefore, preliminary results can be obtained within a short development period.

However, fuzzy control does have some weaknesses. One is that fuzzy control is still lacking generally accepted theoretical design tools. Although preliminary results are easily, further improvements need a lot of especially when the number of inputs increases, the maintenances of the multi-dimensional rule base is time consuming

4.3 STRUCTURE OF A FUZZY CONTROLLER

Fuzzy control is a control method based on fuzzy logic. Just as fuzzy logic can be described simply as "computing with words rather than numbers"; fuzzy control can be described simply as "control with sentences rather than equations". There are specific components characteristic of a fuzzy controller to support a design procedure. In the block diagram in Fig 4.1, the controller is between a preprocessing block and a post-processing block. The following explains the diagram block by block.

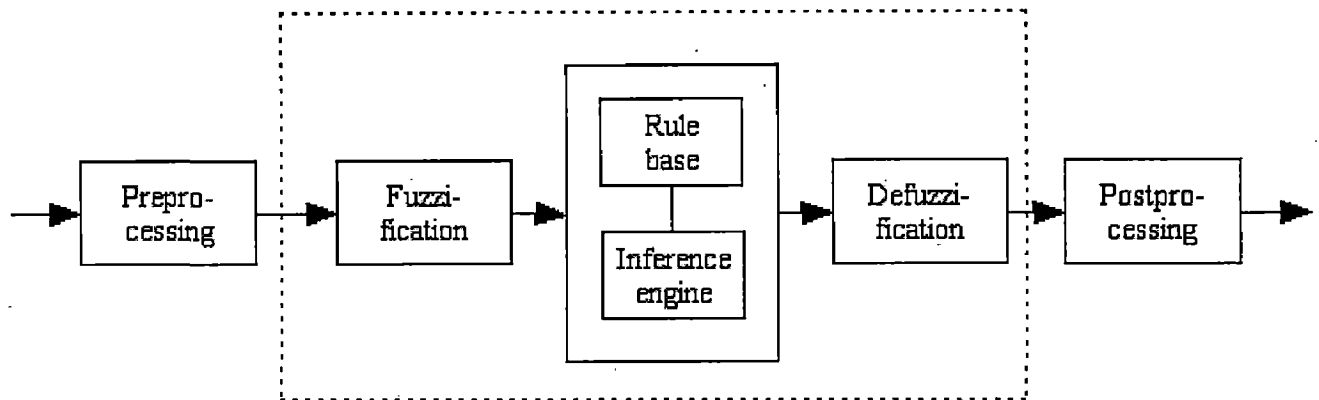


Figure 4.1: Blocks of a fuzzy controller

4.3.1 Preprocessing

The inputs are most often hard or crisp measurements from some measuring equipment, rather than linguistic. A preprocessor, the first block in Fig 4.1, conditions the measurements before they enter the controller. Examples of preprocessing are: Quantization in connection with sampling or rounding to integers; normalization or scaling onto a particular, standard range; filtering in order to remove noise;

A quantiser is necessary to convert the incoming values in order to find the best level in a discrete universe. Assume, for instance, that the variable error has the value 4.5, but the universe is $U = (-5, -4, \dots, 0, \dots, 4, 5)$. The quantiser rounds to 5 to fit it to the nearest level. Quantization is a means to reduce data, but if the quantization is too coarse the controller may oscillate around the reference or even become unstable. When the input to the controller is error, the control strategy is a static mapping between input and control signal. A dynamic controller would have additional inputs, for example derivatives, integrals, or previous values of measurements backwards in time. These are created in the

preprocessor thus making the controller multi-dimensional, which requires many rules and makes it more difficult to design. The preprocessor then passes the data on to the controller.

4.3.2 Fuzzification

The first block inside the controller is fuzzification, which converts each piece of input data to degrees of membership by a lookup in one or several membership functions. The fuzzification block thus matches the input data with the conditions of the rules to determine how well the condition of each rule matches that particular input instance. There is a degree of membership for each linguistic term that applies to that input variable

4.3.3 Rule base The rule base is to do with the fuzzy inference rules. The step response of the system can be roughly divided into four areas $A_1 \sim A_4$ and two sets of points: cross-over $\{b_1, b_2\}$ and peak-valley $\{c_1, c_2\}$ as shown in Fig.4.2. The system equilibrium point is the origin of the phase plane

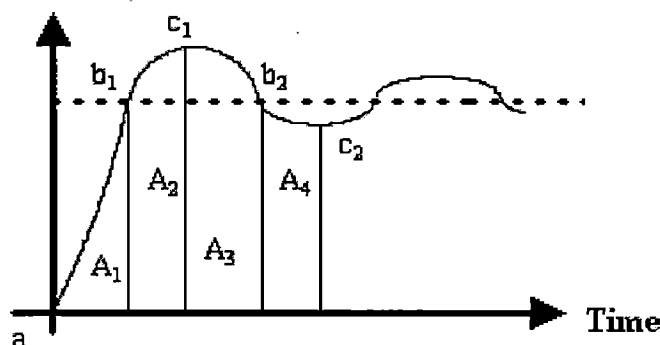


Fig. 4.2 General step response

a) *The sign of rules:* The sign of the rule base can be determined by following meta-rules

- 1) If both e and \dot{e} are zero, then maintain present control setting.
- 2) If conditions are such that e will go to zero at a satisfactory rate, then maintain present control setting.
- 3) If e is not self-correcting, then the sign of the rule can be determined by five sub-criteria.
 - a) Rules for cross-over $\{b_1, b_2\}$ should prevent the overshoot in A_2/A_4 .

- b) Rules for peak-valley $\{c_1, c_2\}$ should speed up the response.
- c) Rules for area A_1/A_3 should speed up the response when e is large and prevent the overshoot in A_2/A_4 when e is close to zero.
- d) Rules for area A_2 should decrease the overshoot around the peak.
- e) Rules for area A_4 should decrease the overshoot around the valley.

The above heuristic method can build general rule base

4.3.4 Inference Engine

Figures 4.3 and 4.4 are both a graphical construction of the algorithm in the core of the controller. In Fig. 4.3, each of the nine rows refers to one rule. For example, the first row says that if the error is negative (row 1, column 1) and the change in error is negative (row 1, column 2) then the output should be negative big (row 1, column 3). The picture corresponds to the rule base in (2). The rules reflect the strategy that the control signal should be a combination of the reference error and the change in error, a fuzzy proportional-derivative controller. We shall refer to that figure in the following. The instances of the error and the change in error are indicated by the vertical lines on the first and second columns of the chart. For each rule, the inference engine looks up the membership values in the condition of the rule.

Aggregation: The aggregation operation is used when calculating the degree of fulfillment or firing strength α_k of the condition of a rule k . A rule, say rule 1, will generate a fuzzy membership value μ_{e1} coming from the error and a membership value μ_{ce1} coming from the change in error measurement. The aggregation is their combination,

$$\mu_{e1} \text{ and } \mu_{ce1} \quad \dots 4.1$$

Similarly for the other rules, Aggregation is equivalent to fuzzification, when there is only one input to the controller. Aggregation is sometimes also called fulfillment of the rule or firing strength.

Activation: The activation of a rule is the deduction of the conclusion, possibly reduced by its firing strength. Thickened lines in the third column indicate the firing strength of each rule. Only the thickened part of the singletons are activated, and **min** or product (*) is used as the activation operator. It makes no difference in this case, since the output membership functions are singletons, but in the general case of s-, II- and z- functions in the third column, the multiplication scales the membership curves, thus preserving the

initial shape, rather than clipping them as the **min** operation does. Both methods work well in general, although the multiplication results in a slightly smoother control signal. In Fig. 4.3, only rules four and five are active.

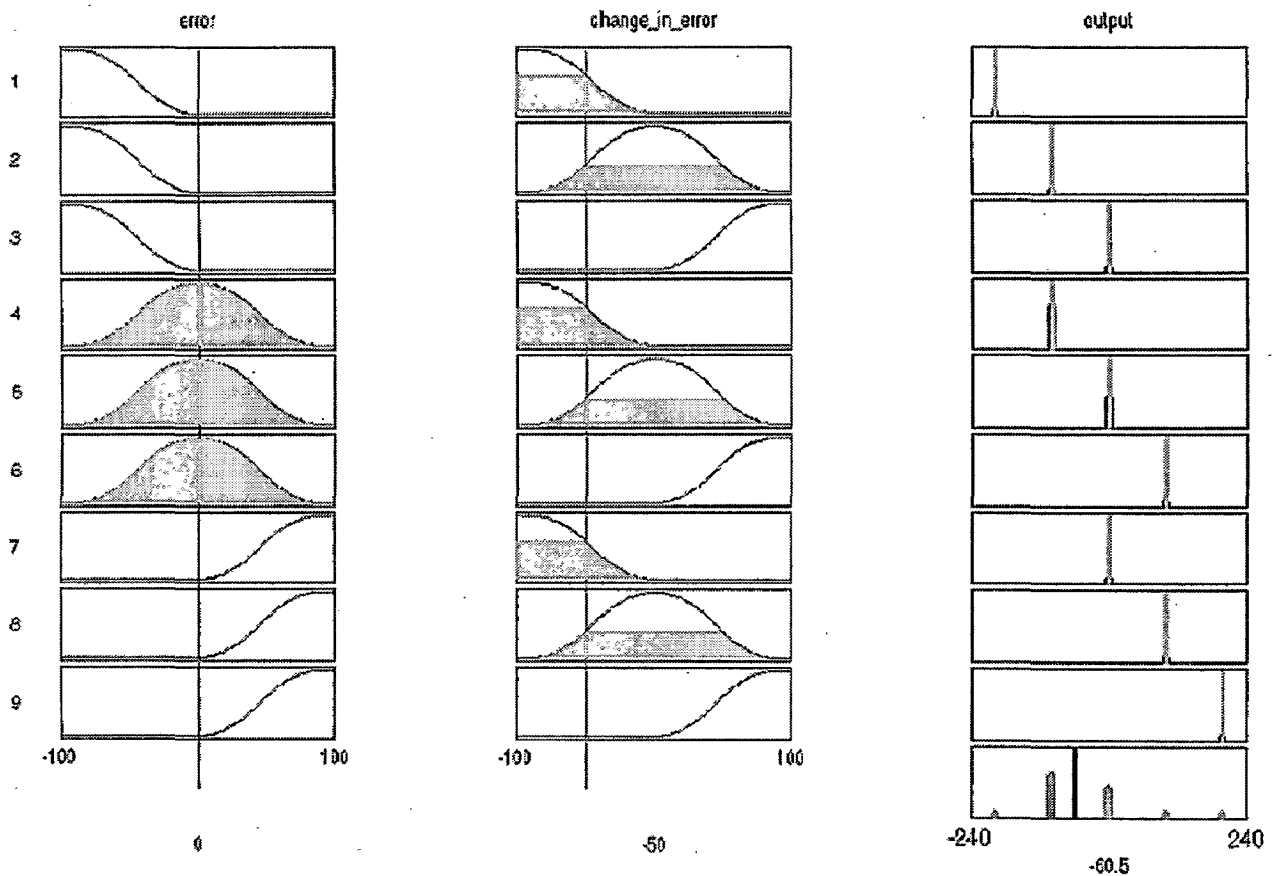


Figure 4.3: Graphical construction of the control signal in a fuzzy PD controller (generated in the Matlab Fuzzy Logic Toolbox).

A rule k can be weighted a priori by a weighting factor $w_k \in [0, 1]$, which is its degree of confidence. In that case the firing strength is modified to

$$\alpha_k^* = w_k * \alpha_k \quad \dots 4.2$$

The degree of confidence is determined by the designer or a learning program trying to adapt the rules to some input-output relationship.

Accumulation: All activated conclusions are accumulated, using the **max** operation, to the final graph on the bottom right (Fig. 4.3). Alternatively, **sum** accumulation counts

overlapping areas more than once (Fig. 4.4). Singleton output (Fig. 4.3) and **sum** accumulation results in the simple output.

$$\alpha_1*s_1+\alpha_2*s_2+\dots+\alpha_n*s_n$$

...4.3

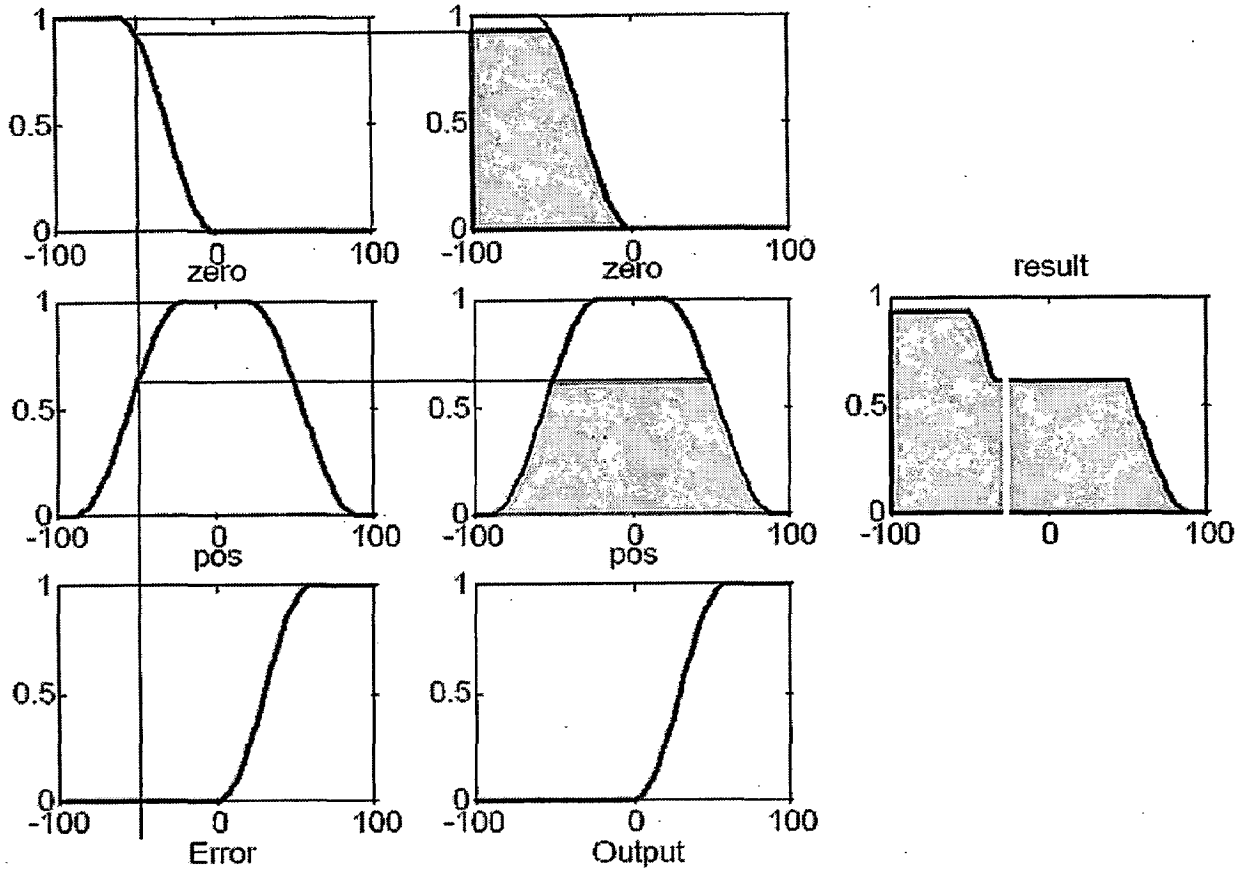


Figure 4.4: One input, one output rule base with non-singleton output sets.

The alphas are the firing strengths from the k rules and $s_1 \dots s_n$ are the output singletons. Since this can be computed as a vector product, this type of inference is relatively fast in a matrix oriented language. There could actually have been several conclusion sets. An example of a one-input two- outputs rule is “If e_a is a then o_1 is x and o_2 is y”. The inference engine can treat two (or several) columns on the conclusion side in parallel by applying the firing strength to both conclusion sets. In practice, one would often implement this situation as two rules rather than one, that is, “If e_a is a then o_1 is x”, “If e_a is a then o_2 is y”.

4.3.5 Defuzzification

The resulting fuzzy set (Fig. 4.3, bottom right; Fig. 4.4, extreme right) must be converted to a number that can be sent to the process as a control signal. This operation is called Defuzzification, and in Fig. 4.4 the x-coordinate marked by a white, vertical dividing line becomes the control signal. The resulting fuzzy set is thus defuzzified into a crisp control signal. There are several Defuzzification methods.

Centre of gravity (COG): The crisp output value u (white line in Fig. 4.4) is the abscissa under the centre of gravity of the fuzzy set,

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \quad \dots 4.4$$

Here x_i is a running point in a discrete universe, and $\mu(x_i)$, is its membership value in the membership function. The expression can be interpreted as the weighted average of the elements in the support set. For the continuous case, replace the summations by integrals. It is a much used method although its computational complexity is relatively high. This method is also called centroid of area.

Centre of gravity method for singleton (COGS): If the membership functions of the conclusions are singletons (Fig.4.3), the output value is

$$u = \frac{\sum_i \mu(s_i) s_i}{\sum_i \mu(s_i)} \quad \dots 4.5$$

Here s_i is the position of singleton i in the universe, and $\mu(s_i)$, is equal to the firing strength α_i of rule i . This method has a relatively good computational complexity, and u is differentiable with respect to the singletons s_i , which is useful in neuro fuzzy systems.

Bisector of area (BOA): This method picks the abscissa of the vertical line that divides the area under the curve in two equal halves. In the continuous case,

$$U = \{x \mid \int_{Min}^x \mu(x) dx = \int_x^{Max} \mu(x) dx\} \quad \dots 4.6$$

Here x is the running point in the universe $\mu(x)$, is its membership, Min is the leftmost value of the universe, and Max is the rightmost value. Its computational complexity is relatively high, and it can be ambiguous. For example, if the fuzzy set consists of two singletons any point between the two would divide the area in two halves; consequently it is safer to say that in the discrete case, BOA is not defined.

Mean of maximum (MOM): An intuitive approach is to choose the point with the strongest possibility, i.e. maximal membership. It may happen, though, that several such points exist, and a common practice is to take the mean of maximum (MOM). This method disregards the shape of the fuzzy set, but the computational complexity is relatively good.

Left maximum (LM), and Right maximum (RM): Another possibility is to choose the leftmost maximum (LM), or the rightmost maximum (RM). In the case of a robot, for instance, it must choose between left and right to avoid an obstacle in front of it. The defuzzifier must then choose one or the other, not something in between. These methods are indifferent to the shape of the fuzzy set, but the computational complexity is relatively small.

4.3.6 Post processing

Output scaling is also relevant. In case the output is defined on a standard universe this must be scaled to engineering units for instance, volts, meters, or tons per hour. An example is the scaling from the standard universe $[-1, 1]$ to the physical units $[-10, 10]$ volts. The post processing block often contains an output gain that can be tuned, and sometimes also an integrator.

5. NEURAL CONTROL

5.1 INTRODUCTION

An Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information. The key element of this paradigm is the novel structure of the information processing system. It is composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems. ANN's, like people, learn by example. An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process. Learning in biological systems involves adjustments to the synaptic connections that exist between the neurons. This is true of ANN's as well. Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. A trained neural network can be thought of as an "expert" in the category of information it has been given to analyze.

Other advantages include:

1. Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience.
2. Self-Organization: An ANN can create its own organization or representation of the information it receives during learning time.
3. Real Time Operation: ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
4. Fault Tolerance via Redundant Information Coding: Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Neural networks take a different approach to problem solving than that of conventional computers. Conventional computers use an algorithmic approach i.e. the computer follows a set of instructions in order to solve a problem. Unless the specific steps that the

computer needs to follow are known the computer cannot solve the problem. That restricts the problem solving capability of conventional computers to problems that we already understand and know how to solve. But computers would be so much more useful if they could do things that we don't exactly know how to do.

Neural networks process information in a similar way the human brain does. The network is composed of a large number of highly interconnected processing elements (neurons) working in parallel to solve a specific problem. Neural networks learn by example. They cannot be programmed to perform a specific task. The examples must be selected carefully otherwise useful time is wasted or even worse the network might be functioning incorrectly. The disadvantage is that because the network finds out how to solve the problem by itself, its operation can be unpredictable. On the other hand, conventional computers use a cognitive approach to problem solving; the way the problem is to be solved must be known and stated in small unambiguous instructions. These instructions are then converted to a high level language program and then into machine code that the computer can understand. These machines are totally predictable; if anything goes wrong is due to a software or hardware fault. Neural networks and conventional algorithmic computers are not in competition but complement each other. There are tasks more suited to an algorithmic approach like arithmetic operations and tasks that are more suited to neural networks. Even more, a large number of tasks, require systems that use a combination of the two approaches (normally a conventional computer is used to supervise the neural network) in order to perform at maximum efficiency. Neural networks do not perform miracles. But if used sensibly they can produce some amazing results.

5.2 THE BACK-PROPAGATION ALGORITHM

In order to train a neural network to perform some task, we must adjust the weights of each unit in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights (dW). In other words, it must calculate how the error changes as each weight is increased or decreased slightly. The back propagation algorithm is the most widely used method for determining the dW .

The back-propagation algorithm is easiest to understand if all the units in the network are linear. The algorithm computes each dW by first computing the error derivative (EA), the rate at which the error changes as the activity level of a unit is changed. For output units, the EA is simply the difference between the actual and the desired output. To compute the EA for a hidden unit in the layer just before the output layer, we first identify all the weights between that hidden unit and the output units to which it is connected. We then multiply those weights by the EAs of those output units and add the products. This sum equals the EA for the chosen hidden unit. After calculating all the EAs in the hidden layer just before the output layer, we can compute in like fashion the EAs for other layers, moving from layer to layer in a direction opposite to the way activities propagate through the network. This is what gives back propagation its name. Once the EA has been computed for a unit, it is straight forward to compute the dW for each incoming connection of the unit. The dW is the product of the EA and the activity through the incoming connection. Note that for non-linear units, the back-propagation algorithm includes an extra step. Before back-propagating, the EA must be converted into the EI, the rate at which the error changes as the total input received by a unit is changed.

The back-propagation Algorithm - a mathematical approach

Units are connected to one another. Connections correspond to the edges of the underlying directed graph. There is a real number associated with each connection, which is called the weight of the connection. We denote by W_{ij} the weight of the connection from unit u_i to unit u_j . It is then convenient to represent the pattern of connectivity in the network by a weight matrix W whose elements are the weights W_{ij} . Two types of connection are usually distinguished: excitatory and inhibitory. A positive weight represents an excitatory connection whereas a negative weight represents an inhibitory connection. The pattern of connectivity characterizes the architecture of the network which is shown in Fig 5.1.

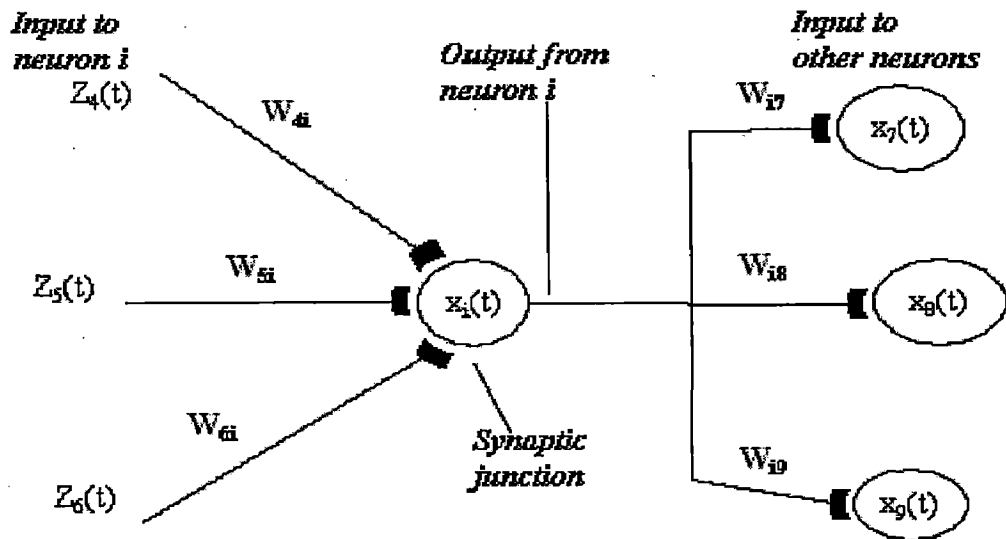


Fig 5.1. Neural network architecture

A unit in the output layer determines its activity by following a two step procedure.

- First, it computes the total weighted input x_j , using the formula:

$$X_i = \sum_i y_i W_{ij} \quad \dots 5.1$$

where y_i is the activity level of the j^{th} unit in the previous layer and W_{ij} is the weight of the connection between the i^{th} and the j^{th} unit.

- Next, the unit calculates the activity y_j using some function of the total weighted input. Typically we use the sigmoid function:

$$y_i = \frac{1}{1 + e^{-x_j}} \quad \dots 5.2$$

Once the activities of all output units have been determined, the network computes the error E , which is defined by the expression:

$$E = \frac{1}{2} \sum_j (y_j - d_j)^2 \quad \dots 5.3$$

where y_j is the activity level of the j^{th} unit in the top layer and d_j is the desired output of the j^{th} unit.

The back-propagation algorithm consists of four steps:

1. Compute how fast the error changes as the activity of an output unit is changed. This error derivative (EA) is the difference between the actual and the desired activity.

$$EA_j = \frac{\partial E}{\partial y_j} = y_j - d_j \quad \dots 5.4$$

2. Compute how fast the error changes as the total input received by an output unit is changed. This quantity (EI) is the answer from step 1 multiplied by the rate at which the output of a unit changes as its total input is changed.

$$EI_j = \frac{\partial E}{\partial x_j} = \frac{\partial E}{\partial y_j} \times \frac{\partial y}{\partial x_j} = EA_j y_j (1 - y_j) \quad \dots 5.5$$

3. Compute how fast the error changes as a weight on the connection into an output unit is changed. This quantity (EW) is the answer from step 2 multiplied by the activity level of the unit from which the connection emanates.

$$EW_{ij} = \frac{\partial E}{\partial W_{ij}} = \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial W_{ij}} = EI_j y_i \quad \dots 5.6$$

4. Compute how fast the error changes as the activity of a unit in the previous layer is changed. This crucial step allows back propagation to be applied to multilayer networks. When the activity of a unit in the previous layer changes, it affects the activities of all the output units to which it is connected. So to compute the overall effect on the error, we add together all these separate effects on output units. But each effect is simple to

calculate. It is the answer in step 2 multiplied by the weight on the connection to that output unit.

$$EA_i = \frac{\partial E}{\partial y_i} = \sum_j \frac{\partial E}{\partial x_j} \times \frac{\partial x_j}{\partial y_i} = \sum_j EI_j W_{ij} \quad \dots 5.7$$

By using steps 2 and 4, we can convert the EAs of one layer of units into EAs for the previous layer. This procedure can be repeated to get the EAs for as many previous layers as desired. Once we know the EA of a unit, we can use steps 2 and 3 to compute the dWs on its incoming connections.

5.3 NEURAL- NETWORK BASED CONTROL:

The block diagram of the control system is shown in Fig5.2. As can be seen from this figure, the sum of the outputs of the neural network and feedback controller (i.e. PID Controller) will be the actual input torque to robot.

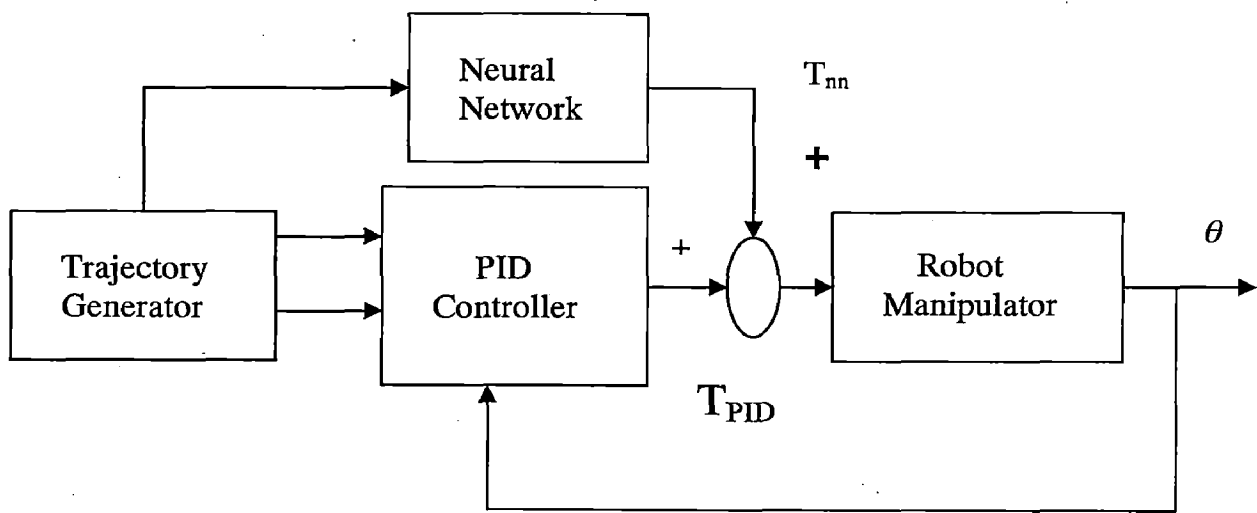


Fig.5.2 Neural controller

This can be expressed as

$$\tau = \tau_{PID} + \tau_{NN} \quad \dots 5.8$$

where τ_{PID} is the output of feedback PID controller and τ_{NN} is the output of the neural network. The feedback controller plays a role in making the whole system stable. The neural network has been trained off line to approximate the inverse dynamic model of the robot manipulator. The learning scheme is shown in Fig.5.3.

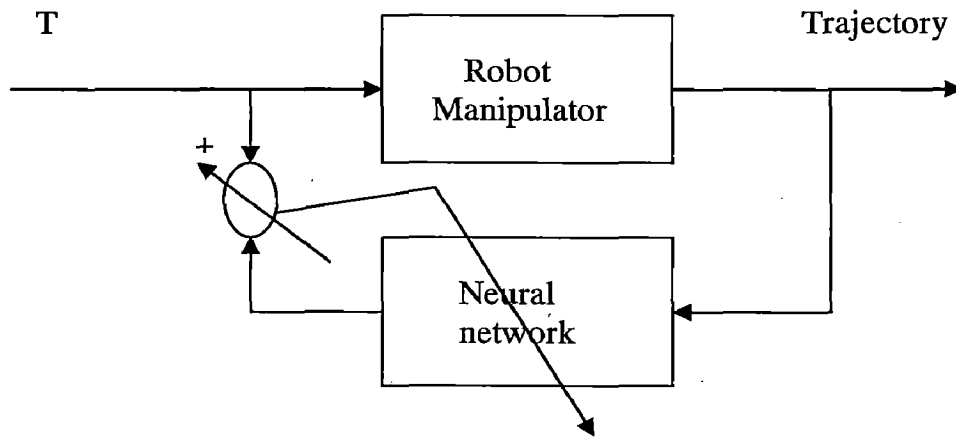


Fig.5.3 Scheme for learning dynamic model

The manipulator receives the torque τ and outputs the resulting trajectory θ . Inverse dynamic model is set in the opposite input-output direction to that of the manipulator. That is, it receives the trajectory as an input and produces the torque τ_{NN} as its output. The error signal is the difference between the actual torque and estimated torque. It is expected that this difference tends to zero as learning proceeds. Once the neural network finishes learning, it produces an approximate inverse dynamic model.

6. NEURO-FUZZY SYSTEMS

6.1 INTRODUCTION

Neuro-fuzzy control is one of the intelligent control methods since knowledge engineering is used in neuro fuzzy control. Neuro fuzzy control is usually utilised for two purposes. One is for non-linear applications and the other is for adding human intelligence to controllers. A linear PID Controller structure is changed by fuzzy logic, such that the controller makes the system respond quickly if the error is large.

Recently, the combination of neural networks and fuzzy logic has received attention. The idea is to lose the disadvantages of the two and gain the advantages of both. Neural networks bring into this union the ability to learn. Fuzzy logic brings into this union a model of the system based on membership functions and a rule base.

Determining the fuzzy membership functions from sample data using a neural network is the most obvious method of using the two together. The definition of the membership function has a huge impact on the system response. Often, the programmer must use trial and error to find acceptable values. Assuming a certain shape and finding the beginning and endpoints for the fuzzy values in a fuzzy set is a neural network optimization problem. Figure 6.1 is a diagram of such a system.

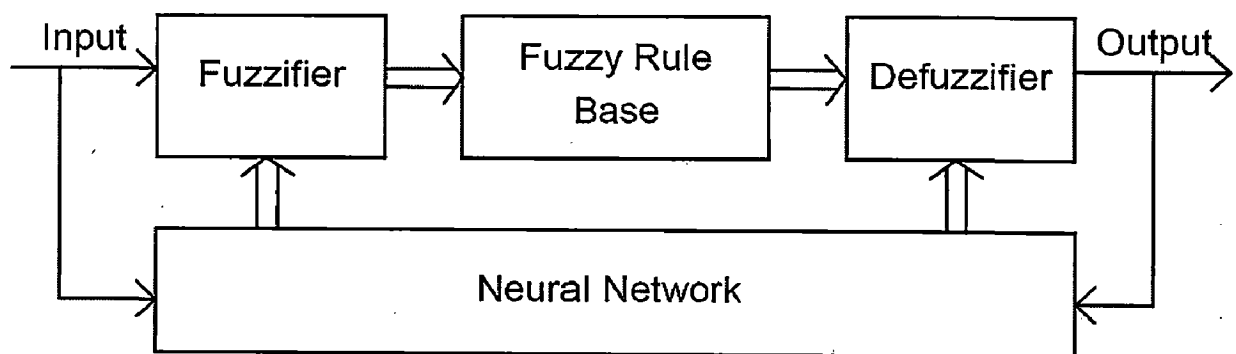


Figure.6.1 A fuzzy system whose membership functions are adjusted by a neural network.

Figure 6.2 shows a more complex integration, the use of neural networks to determine both the fuzzy membership functions and the rule base. Scientists have developed a system that converts input and output data into nonlinear membership functions and a rule base. The nonlinearity of the membership functions is unique to membership functions derived by neural networks. They help minimize the number of rules.

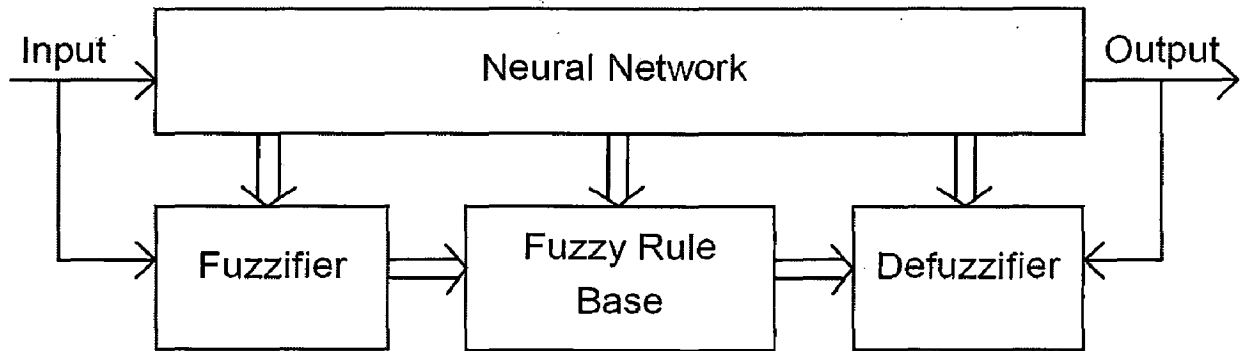


Fig .6.2 A fuzzy system defined by a neural network.

Another approach is to incorporate fuzzy logic into the neurons of the neural networks.. It was quickly realized that neurons with output in the range of $[0, 1]$ produced much better results. The concept of a fuzzy neuron, however, has advanced beyond simply expanding the range of outputs on a crisp neuron. Some researchers have incorporated membership functions and rule bases into the individual neurons, as shown in figure 6.3.

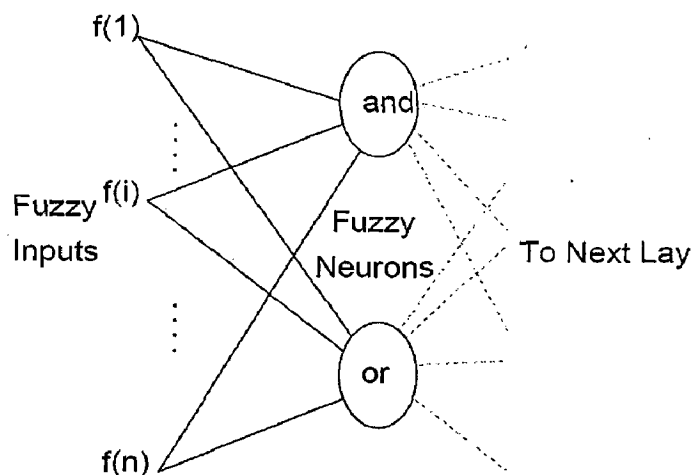


Figure .6.3. A neural network of fuzzy neurons.

Finally, the idea of fuzzification of control variables into degrees of membership in fuzzy sets has been integrated into neural networks. See figure 6.4. If the inputs and outputs of a

neural network are fuzzified and defuzzified, significant improvements in the training time, in the ability to generalize, and in the ability to find minimizing weights can be realized. Also, the membership function definition gives the designer more control over the neural network inputs and outputs. It is this technique that is implemented in this thesis for the control of a robotic arm

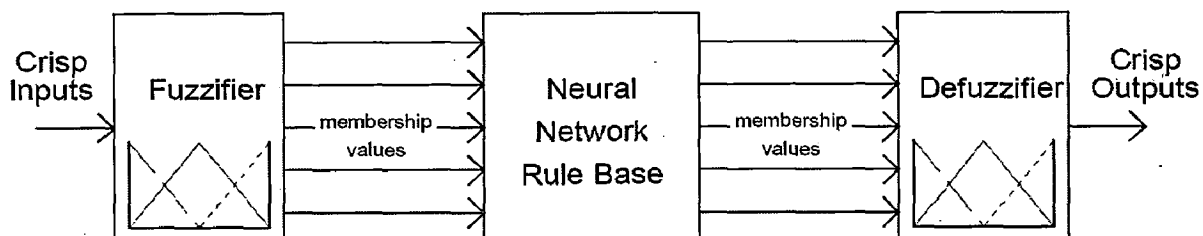


Fig.6.4 A fuzzy system with neural network rule base

6.2 ADAPTIVE NETWORKS: ARCHITECTURES AND LEARNING ALGORITHMS

This section introduces the architecture and learning procedure of the adaptive network which is in fact a superset of all kinds of feedforward neural networks with supervised learning capability. An adaptive network, as its name implies, is a network structure consisting of nodes and directional links through which the nodes are connected. Moreover, part or all of the nodes are adaptive, which means each output of these nodes depends on the parameter(s) pertaining to this node, and the learning rule specifies how these parameters should be changed to minimize a prescribed error measure.

The basic learning rule of adaptive networks is based on the gradient descent and the chain rule, which was proposed by Werbos in the 1970's. However, due to the state of artificial neural network research at that time, Werbos' early work failed to receive the attention it deserved.

Since the basic learning rule is based the gradient method which is notorious for its slowness and tendency to become trapped in local minima, here we propose a hybrid

learning rule which can speed up the learning process substantially. Both the batch learning and the pattern learning of the proposed hybrid learning rule is discussed below, though our simulations are mostly based on the batch learning.

6.3 ANFIS: ADAPTIVE-NETWORK-BASED FUZZY INFERENCE SYSTEM

In this section, we propose a class of adaptive networks which are functionally equivalent to fuzzy inference systems. The proposed architecture is referred to as ANFIS, standing for Adaptive-Network-based Fuzzy Inference System. We describe how to decompose the parameter set in order to apply the hybrid learning rule. Besides, we demonstrate how to apply the Stone-Weierstrass theorem to ANFIS with simplified fuzzy if-then rules and how the radial basis function network relate to this kind of simplified ANFIS.

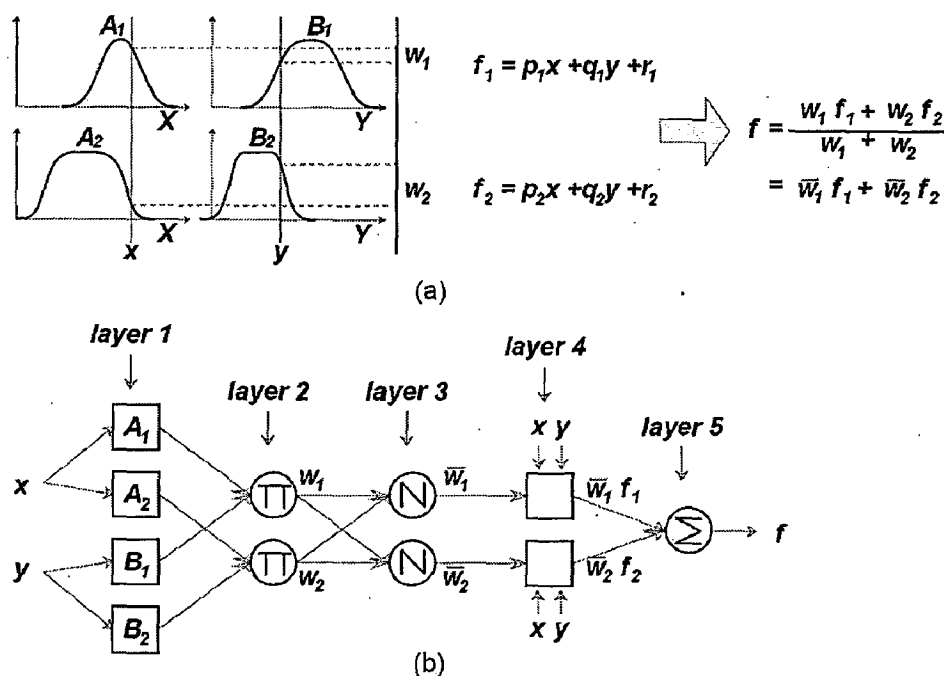


Fig 6.5 (a) fuzzy reasoning (b) equivalent ANFIS.

A. ANFIS architecture

For simplicity, we assume the fuzzy inference system under consideration has two inputs x and y and one output z . Suppose that the rule base contains two fuzzy if-then rules of Takagi and Sugeno's type

Rule 1: If x is A_1 and y is B_1 , then $f_1 = p_1x + q_1y + r_1$, Rule
 2: If x is A_2 and y is B_2 , then $f_2 = p_2x + q_2y + r_2$

Then the fuzzy reasoning is illustrated in Figure 6.5(a), and the corresponding equivalent ANFIS architecture is shown in Figure 6.5(b). The node functions in the same layer are of the same function family as described below:

Layer 1: Every node i in this layer is a square node with a node function

$$O_i^1 = \mu_{A_i}(x), \quad \dots 6.1$$

Where x is the input to node i , and A_i is the linguistic label (*small, large, etc.*) associated with this node function. In other words, O_i^1 is the membership function of A_i and it specifies the degree to which the given x satisfies the quantifier A_i . Usually we choose $\mu_{A_i}(x)$ to be bell-shaped with maximum equal to 1 and minimum equal to 0, such as

$$\mu_{A_i}(x) = \frac{1}{1 + \left[\left(\frac{x - c_i}{a_i} \right)^2 \right]^{b_i}}, \quad \dots 6.2$$

Or

$$\mu_{A_i}(x) = \exp \left\{ - \left[\left(\frac{x - c_i}{a_i} \right)^2 \right]^{b_i} \right\}, \quad \dots 6.3$$

Where $\{a_i, b_i, c_i\}$ is the parameter set? As the values of these parameters change, the bell-shaped functions vary accordingly, thus exhibiting various forms of membership

functions on linguistic label A_i . In fact, any continuous and piecewise differentiable functions, such as commonly used trapezoidal or triangular-shaped membership functions, are also qualified candidates for node functions in this layer. Parameters in this layer are referred to as premise parameters.

Layer 2 : Every node in this layer is a circle node labeled Π which multiplies the incoming signals and sends the product out. For instance

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), \quad i=1,2 \quad \dots 6.4$$

Each node output represents the firing strength of a rule. (In fact, other T-norm operators those perform generalized AND can be used as the node function in this layer.)

Layer 3: Every node in this layer is a circle node labeled N . The i -th node calculates the ratio of the i -th rule's firing strength to the sum of all rules' firing strengths

$$\bar{w}_i = \frac{w_i}{w_1 + w_2}, \quad \dots 6.5$$

For convenience, outputs of this layer will be called called normalized firing strengths.

Layer 4: Every node i in this layer is a square node with a node function

$$O_i^4 = \bar{w}_i f_i = \bar{w}_i (p_i x + q_i y + r_i), \quad \dots 6.6$$

where \bar{w}_i is the output of layer 3, and $\{p_i, q_i, r_i\}$ is the parameter set. Parameters in this layer will be referred to as consequent parameters.

Layer 5: The single node in this layer is a circle node labeled Σ that computes the overall output as the summation of all incoming signals, i.e.,

$$O_1^5 = \text{overall output} = \sum_i \bar{w}_i f_i = \frac{\sum_i \bar{w}_i f_i}{\sum_i w_i} \quad \dots 6.7$$

Thus we have constructed an adaptive network which is functionally equivalent to a fuzzy inference system shown in fig.6.7. For fuzzy inference systems shown in fig6.6, the extension is quite straightforward and the ANFIS is shown in Figure 6.7 where the output of each rule is induced jointly by the output membership function and the firing strength.

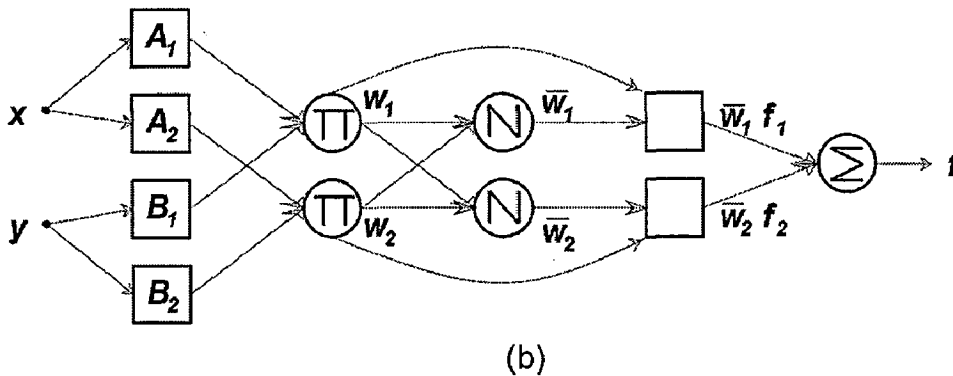
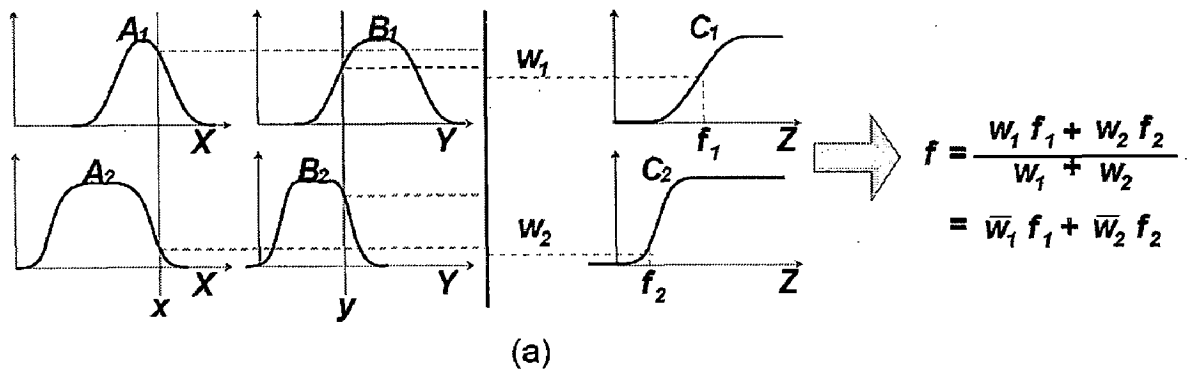


Figure 6.6: (a) fuzzy reasoning;
(b) Equivalent ANFIS.

For fuzzy inference systems shown in fig.6.6, if we replace the centroid defuzzification operator with a discrete version which calculates the approximate centroid of area, then ANFIS can still be constructed accordingly. However, it will be more complicated than its versions and thus not worth the efforts to do so.

Figure 6.7 shows a 2-input, ANFIS with 9 rules. Three membership functions are associated with each input, so the input space is partitioned into 9 fuzzy subspaces, each of which is governed by a fuzzy if-then rule. The premise part of a rule delineates a fuzzy subspace, while the consequent part specifies the output within this fuzzy subspace.

B. Hybrid Learning Algorithm

From the proposed ANFIS architecture (Figure 6.7), it is observed that given the values of premise parameters, the overall output can be expressed as linear combinations of the consequent parameters. More precisely, the output f in Figure 6.7 can be rewritten as

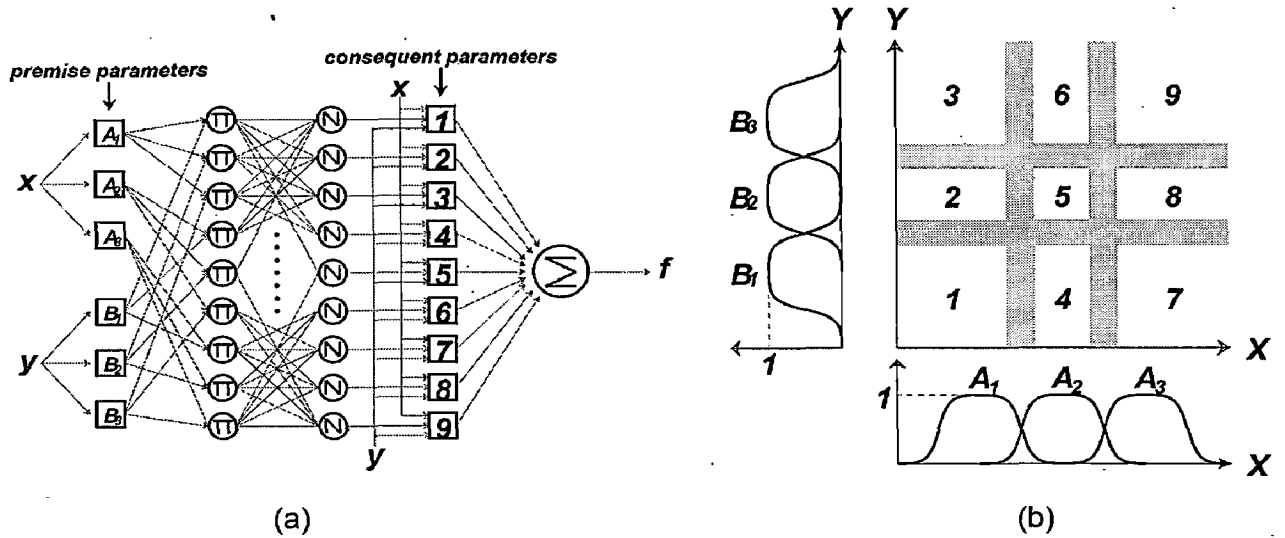


Figure 6.7: (a) 2-input ANFIS with 9 rules; (b) corresponding fuzzy subspaces

$$\begin{aligned}
 f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_2}{w_1 + w_2} f_2 \\
 &= \bar{w}_1 f_1 + \bar{w}_2 f_2 \quad \dots 6.8 \\
 &= (\bar{w}_1 x) p_1 + (\bar{w}_1 y) q_1 + (\bar{w}_1) r_1 + (\bar{w}_2 x) p_2 + (\bar{w}_2 y) q_2 + (\bar{w}_2) r_2,
 \end{aligned}$$

Which is linear in the consequent parameters $(p_1, q_1, r_1, p_2, q_2$ and $r_2)$ As a result, we have

S = set of total parameters,

S_1 = set of premise parameters,

S_2 = set of consequent parameters

The hybrid learning algorithm can be applied directly. More specifically, in the forward pass of the hybrid learning algorithm, functional signals go forward till layer 4 and the consequent parameters are identified by the least squares estimate. In the backward pass, the error rates propagate backward and the premise parameters are updated by the gradient descent. More details are in [18]

7. DESIGN AND SIMULATION IN SIMULINK/MATLAB7.01

7.1 INTRODUCTION

The aim of simulation is to develop complete model of the physical system and to analyze the system in different ways before going to implement it practically. In my dissertation control of puma 560 robot is analyzed with different controllers such as Conventional and Intelligent controllers. In this chapter design and development of simulink model for robot manipulator, Actuator, Conventional controllers and intelligent controllers are explained.

7.2 PUMA560 ROBOT MODEL

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

⇒

$$\ddot{q} = M(q)^{-1} [\tau - C(q, \dot{q})\dot{q} - g(q)] \quad \dots 7.1$$

By equation 7.1, we can develop the simulink model

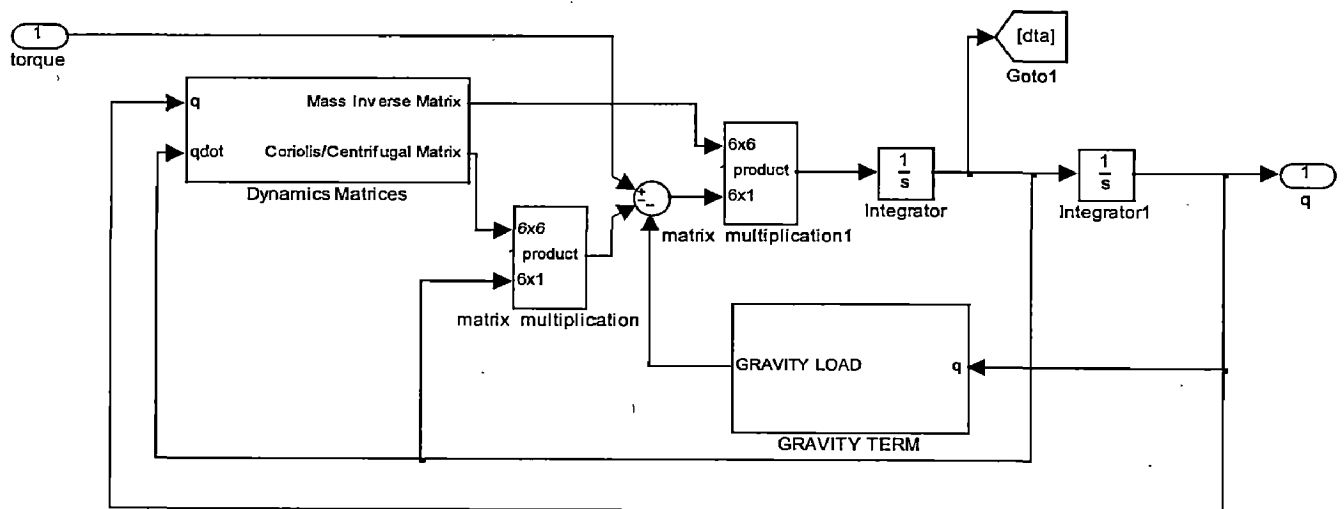


Fig .7.1 PUMA560 simulink model

Fig7.1 shows the simulink model of the PUMA 560 robot it contains the following blocks

1. dynamics matrices
2. gravity term
3. matrix multiplication

1. DYNAMICS MATRICES

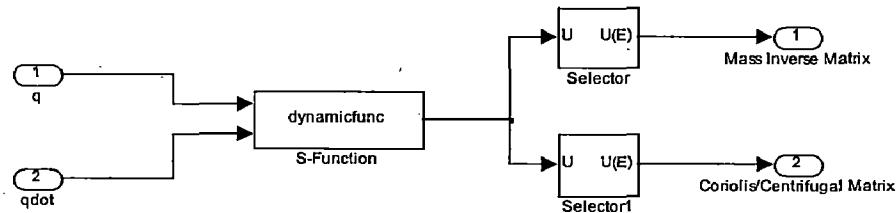


Fig7.2 dynamics matrices

This block have the two input matrices angles at respective joints and derivatives of the angles of the order 6 by1 and the two out puts mass inverse matrix and coriolis/centrifugal matrix of the order 36 by1 (actually these matrices are the order of 6 by 6 but they are arranged column wise for the simplicity)

For this particular block a S-Function is written in C-Language which have two inputs of the order 6 by 1 and one out put of the order 72 by 1. S-Function is used because design of two in puts of the order 6 by 1 two out puts of the order 6 by 6 is extremely difficult in simulink for this purpose S-Function is used. After written the code in C we have to compile the Dot C file by the command 'MEX' then '.mdl' file will be created in the current directory, this '.mdl' file will be useful for the simulink to run the simulation

2 GRAVITY TERM

Figure 7.3 shows the gravity load for the manipulator. Expressions for the gravity load are explained and given in chapter 2 in this block functional blocks are used we can write function in that block since gravity load is function of the angles so we can write any function in the functional block

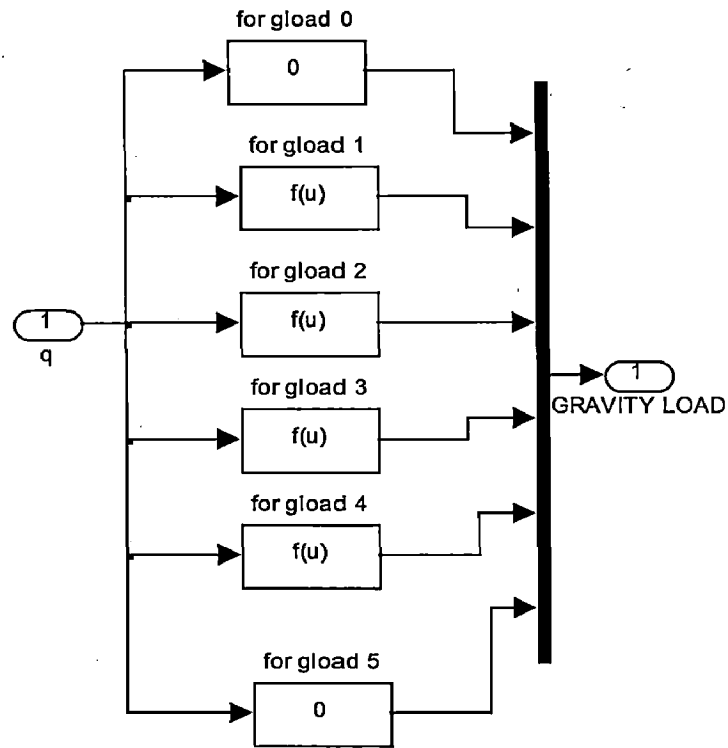


Fig 7.3 gravity load

MATRIX MULTIPLICATION

This matrix multiplication block shown in fig 7.4 is having two inputs of the order 36 by 1 and 6 by 1 one output of the order 6 by 1 since one input of the order 36 by 1 six selectors selects 6 elements column wise one after other and these are concatenated horizontally by matrix concatenation block after this process elements become matrix of the order 6 by 6 then product block is used to multiply this matrix with second input 2

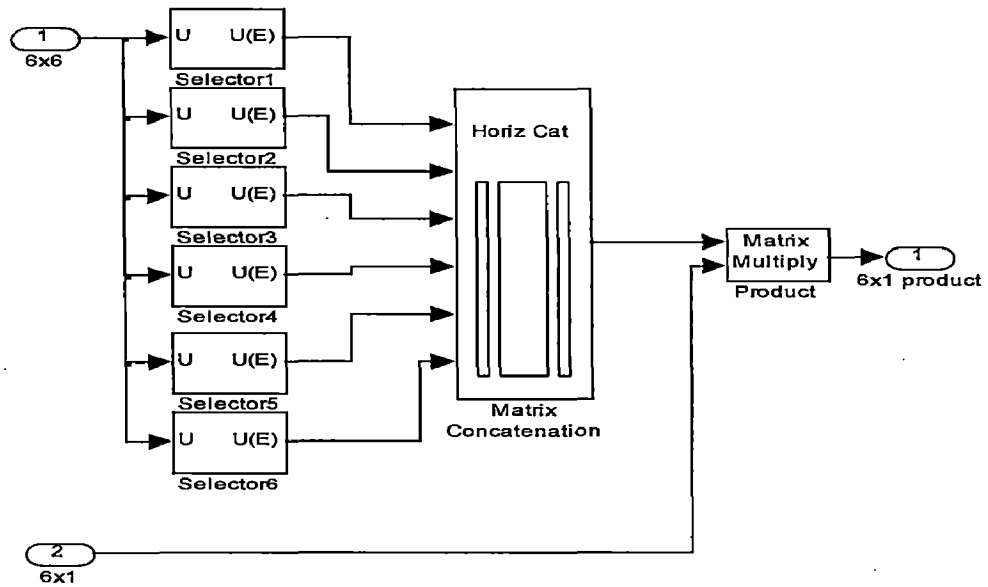


Fig 7.4 Matrix multiplications

By this discussion on design of PUMA560 model is complete

7.3 ACTUATOR MODEL

Fig 7.5 shows the simulink model for the actuator. It is designed as per the equations and specifications given in the chapter 2

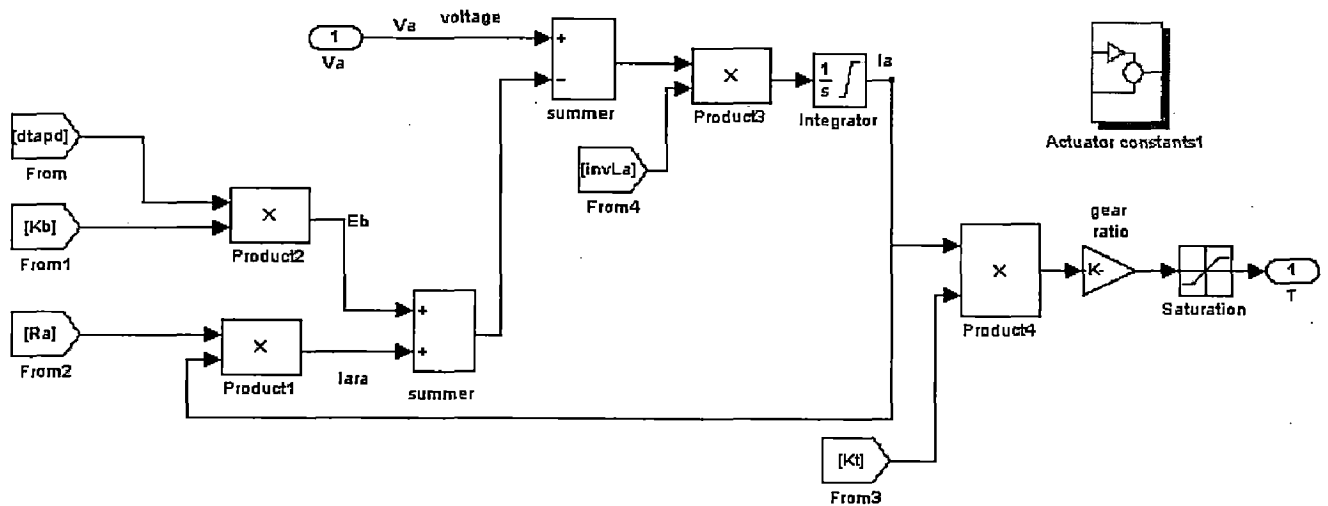


Fig.7.5 Actuator simulink model

7.4 DESIGN OF PID CONTROLLER

Equation for the PID controller is

$$U_i = P_i e + D_i \frac{de}{dt} + I_i \int e \quad (i=1, 2, \dots, 6) \quad \dots 7.2$$

Where e is the error

P_i is the proportional gain

D_i is the differential gain

I_i is the integral gain

U_i is the controller output

The objective of designing PID controller is to find the P_i, D_i, I_i for the optimum response of the system

Hand tuning procedure for the tuning of the PID controller

- a) remove all integral and differential action
- b) tune the proportional gain or increase the proportional P_i to give the desired response ignoring any offset or peak over shoots
- c) then tune the differential gain D_i (increase) until the oscillations are under the allowable range
- d) tune the integral gain I_i (increase) until the until offset is in the allowable range
- e) repeat this until P_i as large as possible

In designing considered that controller output should not more than 40 volts

Total system with PID controller is shown in Fig 7.6 and Response of system is shown in Fig 7.8.

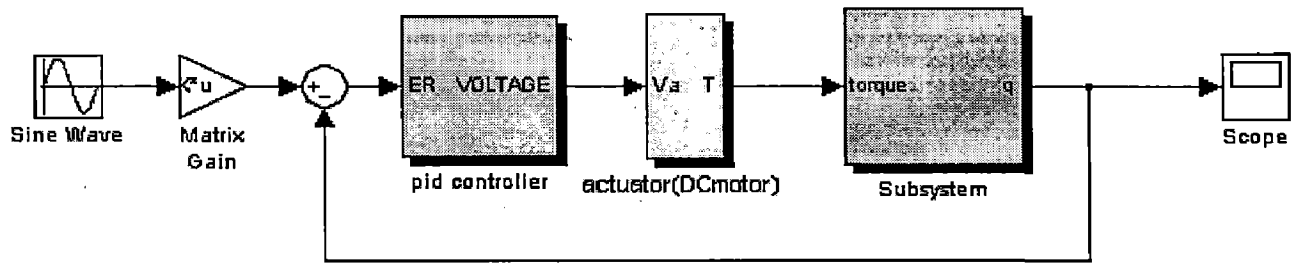


Fig 7.6.Total system with PID controller

7.5 DESIGN OF FEED FORWARD INVERSE DYNAMICS CONTROLLER

Total system with FFID controller is shown in Fig 7.7 and Response of system is shown in Fig 7.8. Designing the Feed forward inverse dynamics controller in simulink consists of three steps

1. Feed forward inverse model
2. Feed back controller
3. Robot manipulator

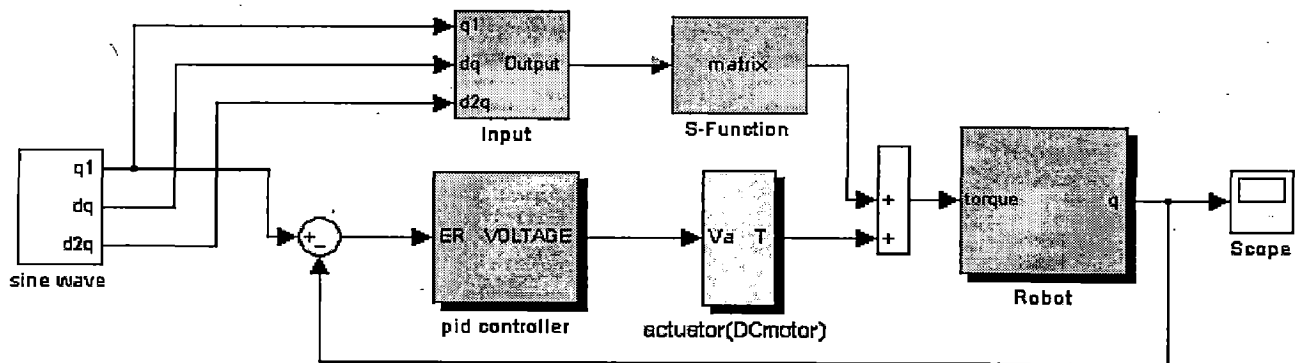


Fig 7.7 Total system with FFID controller

Feed forward inverse model consists of two blocks those are input block and S-function block. In Feed forward inverse model block, we have to calculate the torques required for each link for given position, velocity and acceleration. 6 degree of freedom robot

manipulator model equations (given in chapter 2) are used to calculate torques required for each link for given position, velocity and acceleration. In the input block,

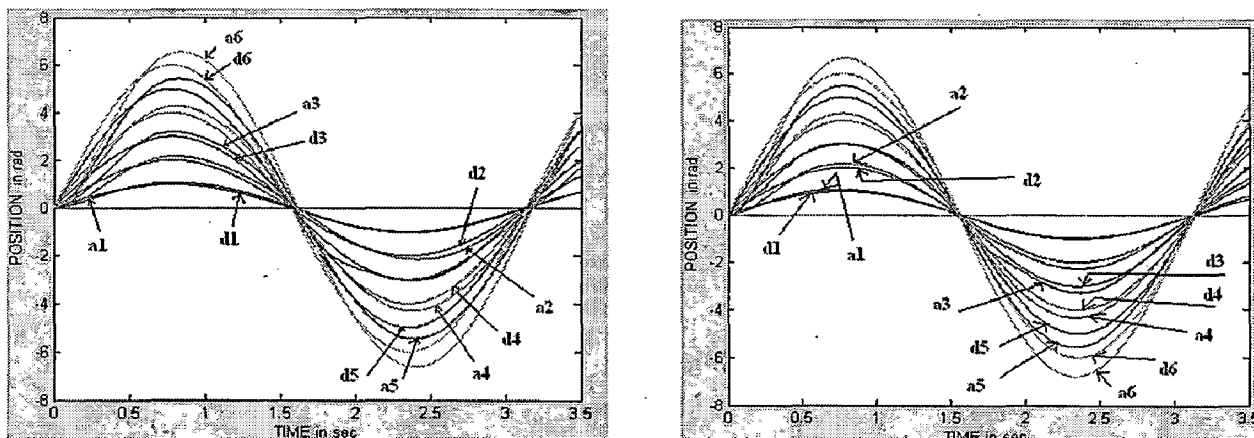


Figure 7.8 Robot Response with PID control and FFID control

inputs are provided in order to S-function block. M-file is used to calculate torques. Feed back controller is simply PID controller.

7.6 DESIGN OF COMPUTED TORQUE CONTROLLER

Designing of Computed torque control is similar to the Feed forward inverse dynamics controller. The only difference is in input block i.e. how the inputs are provided. Input block of CTC controller is shown in Fig 7.9 and response of system is shown in Fig 7.11

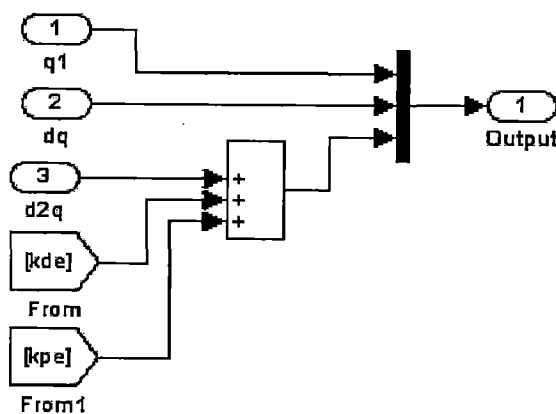


Fig 7.9 Input block for CTC method

7.7 DESIGN OF CRITICALLY DAMPED INVERSE DYNAMICS CONTROLLER

Designing of Critically damped inverse dynamic control is also similar to the Feed forward inverse dynamics controller. The only difference is in input block i.e. how the inputs are provided. Input block of CDID controller is shown in Fig 7.10 and response of system is shown in Fig 7.11

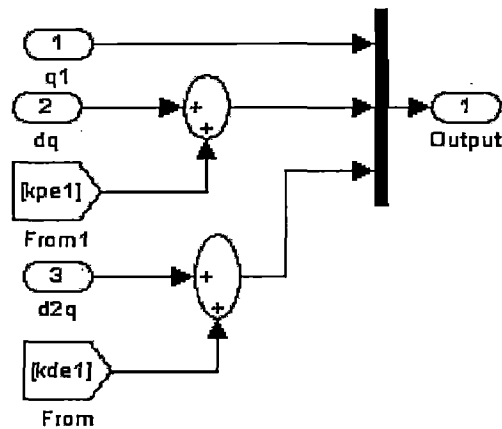


Fig 7.10 Input block for CDID method

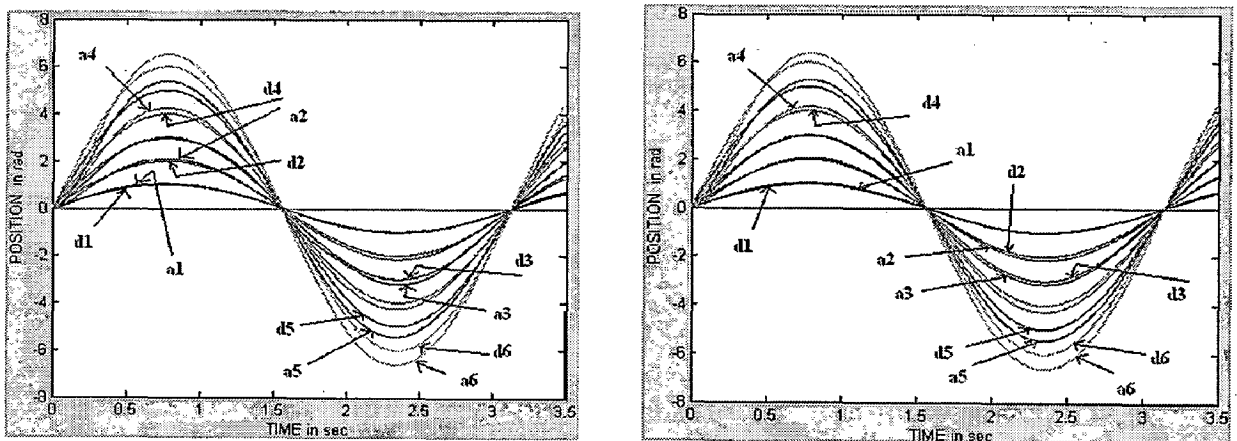


Figure 7.11 Robot Response with CTC control and CDID control

7.8 DESIGN OF FUZZY PD+I CONTROL

Total system with Fuzzy controller is shown in Fig 7.13 and Response of system is shown in Fig 7.17. Designing the Fuzzy controller in simulink consists of two steps

1. Designing the rule base
2. gain scheduling

TABLE 1 Fuzzy rules

$\Delta e/e$	NB	NM	NS	ZE	PS	PM	PB
NB	NB	NB	NB	NM	NS	NS	ZE
NM	NB	NM	NM	NM	NS	ZE	PS
NS	NB	NM	NS	NS	ZE	PS	PM
ZE	NB	NM	NS	ZE	PS	PM	PB
PS	NM	NS	ZE	PS	PS	PM	PB
PM	NS	ZE	PS	PM	PM	PM	PB
PB	ZE	PS	PS	PM	PB	PB	PB

Design of rule base

Table 1 shows the rule base for the Fuzzy PD controller the rule base is to design as explained in the second chapter, but complex systems such as robot understanding the system behavior is very difficult so set of PD rules were proposed [10]. These rules generally used for the Fuzzy PD controller

Gain scheduling

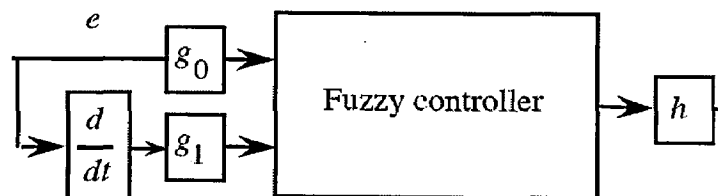


Fig7.12 Fuzzy control with gains

Gain scheduling means designing of g_0 , g_1 and h for the optimum response of the system

Gain scheduling procedure for the Fuzzy controller

1. Initially put $g_0=0$ and increase g_1 until the controller gives the output normally, when the signal after the gain g_1 crosses the universe of discourse there will not be any rule to process then controller then the output will be zero before this happens previously designed gain will be the optimum gain for the g_1
2. increase h until the controller will give the maximum output, that will be the maximum controller output
3. then increase g_0 until overshoots under the allowable range

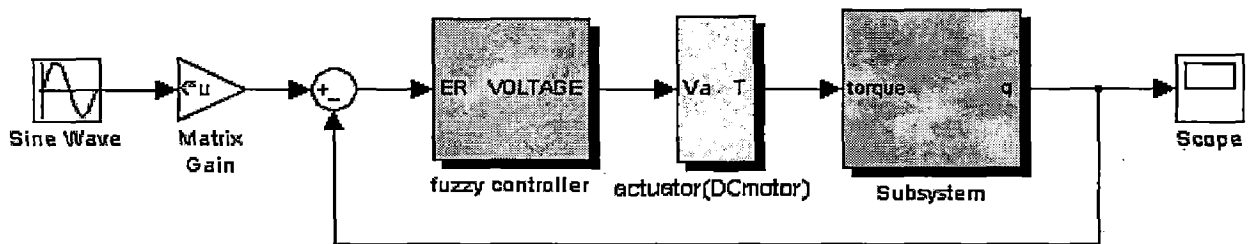


Fig7.13 System with fuzzy controller

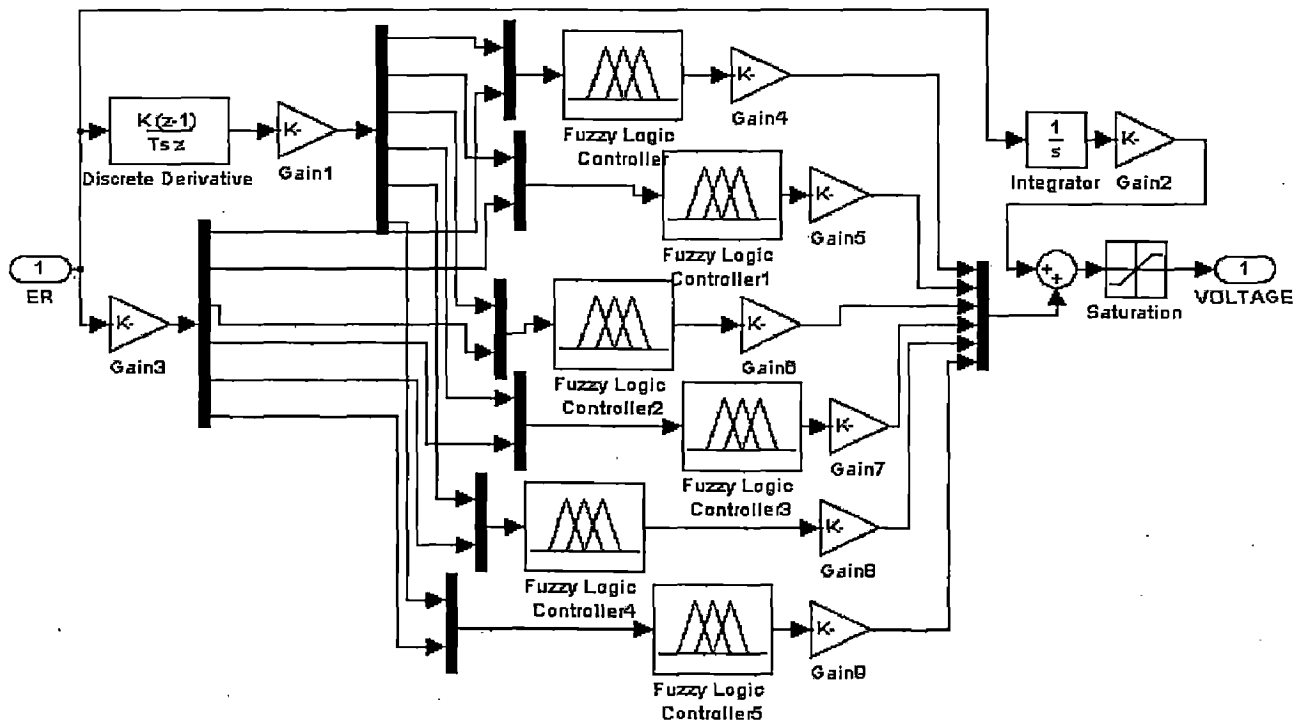


Fig7.14 Fuzzy controller

7.9 DESIGN OF NEURAL NETWORK BASED CONTROL

Total system with Neural based controller is shown in Fig 7.15 and Response of system is shown in Fig 7.17. Designing the Neural controller in simulink consists of three steps

1. Feed forward controller
2. Feed back controller
3. Robot manipulator

This Neural controller can be designed in simulink using the neural network toolbox and simulink.

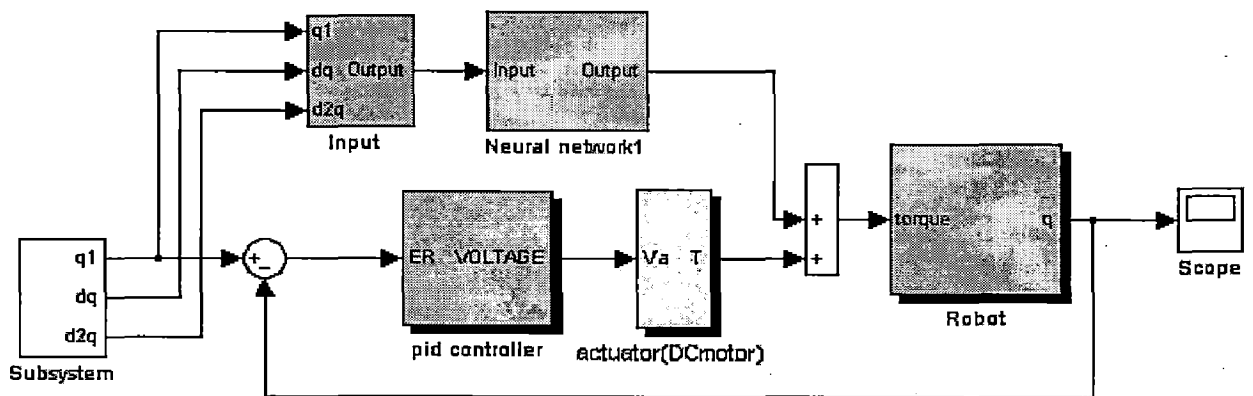


Fig.7.15 Total robot system with neural controller

Feed back controller and Robot manipulator blocks are same as in Feed forward inverse dynamics controller. Feed forward controller contains two blocks, which are Input block and Neural Network1 block. Input block provides inputs to Neural Network1 block in which order Neural network requires inputs.

Neural Network1 block is shown in Fig 7.16. It consists of three blocks, which are

1. Pre processing block
2. Neural network
3. Post processing block

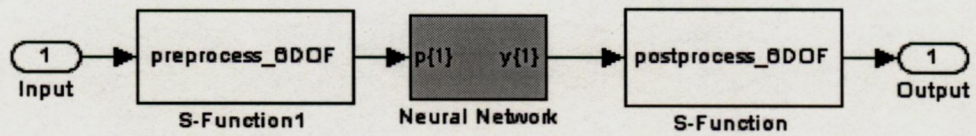


Fig 7.16 Neural Network1 block

Pre processing and Post processing blocks normalizes the inputs and outputs respectively. M-file programs have written for Pre processing and Post processing blocks. Neural network toolbox and **gensim** command is used to create neural network block. M-file program is written for this purpose.

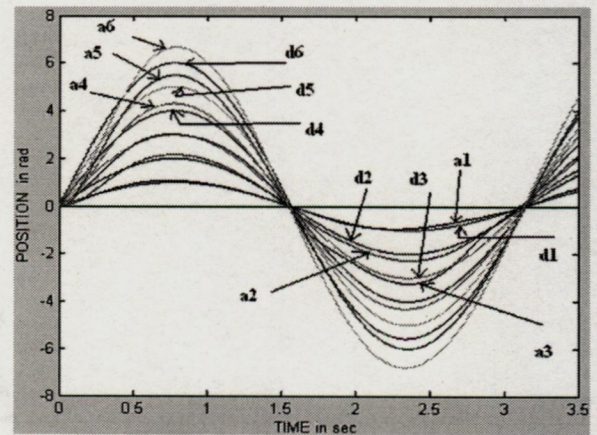
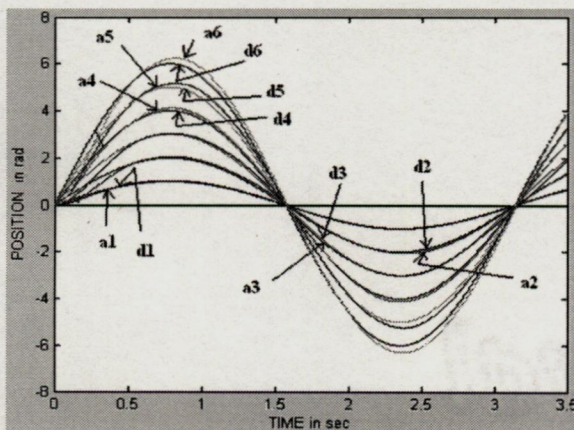
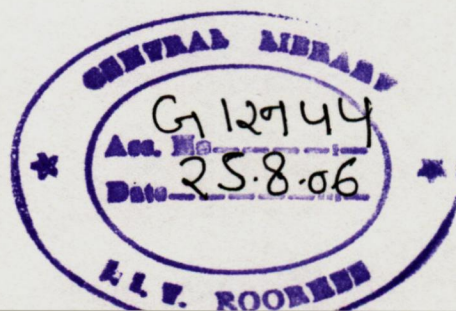


Fig7.17 Robot response with Fuzzy control and Neural control

Algorithm for creation of neural network block

1. Generate inputs and targets for a given trajectory by using dynamic model equations of two-link robot manipulator.
2. Create the network with required structure by using **newff** command.
3. Train the Neural network by **train** command with specifications.
4. Use **gensim** command to generate simulink block of trained network.

The numbers of input and output nodes of a neural network are determined from the problem at hand whereas the numbers of nodes in hidden layers are flexible. The neural network employed in simulation consists of an input layer with 18 nodes. The first hidden layer with 60 nodes, the second hidden layer with 30 nodes and an output layer with six nodes. Using an optimization approach an alternative and more effective learning algorithm, Scaled Conjugate Gradient (SCG) than the standard back propagation (BP) has



been used. SCG belongs to the class of Conjugate Gradient Methods, which shows super linear convergence on most problems.

7.10 DESIGN OF NEURO-FUZZY CONTROL

This Neuro-Fuzzy controller can be designed in simulink using the fuzzy logic toolbox. About ANFIS already explained in chapter 6 .In matlab design of ANFIS consists of Training

Model Learning and Inference through ANFIS

Total system with ANFIS controller is shown in Fig 7.18. The basic idea behind these neuro-adaptive learning techniques is very simple. These techniques provide a method for the fuzzy modeling procedure to learn information about a data set, in order to compute the membership function parameters that best allow the associated fuzzy inference system to track the given input/output data. This learning method works similarly to that of neural networks.

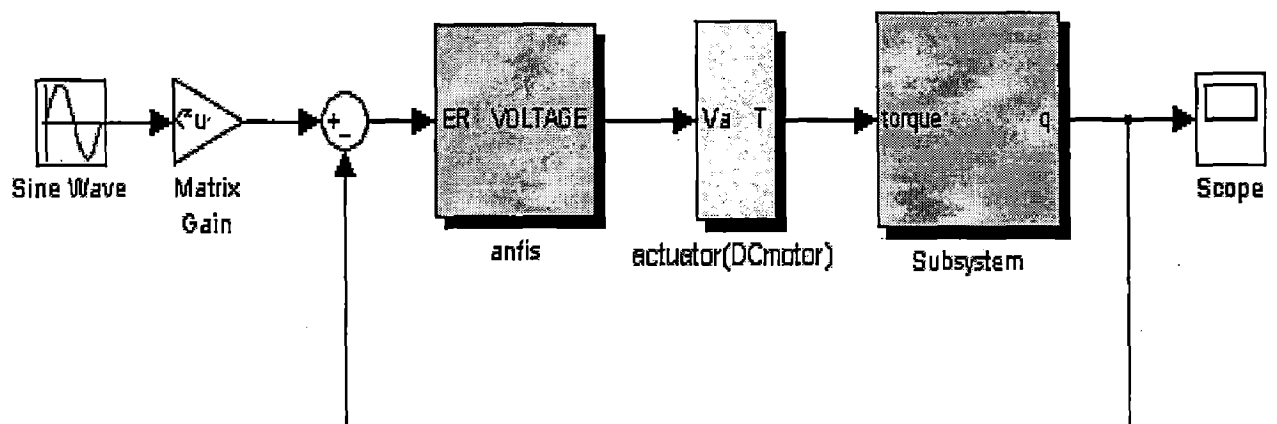


Fig.7.18 Total robot system with ANFIS controller

The Fuzzy Logic Toolbox function that accomplishes this membership function parameter adjustment is called ANFIS. ANFIS can be accessed either from the command line, or through the ANFIS Editor GUI. After training parameters of member function are set such that that will perform best. Response of system and membership functions after training shown in Fig 7.20

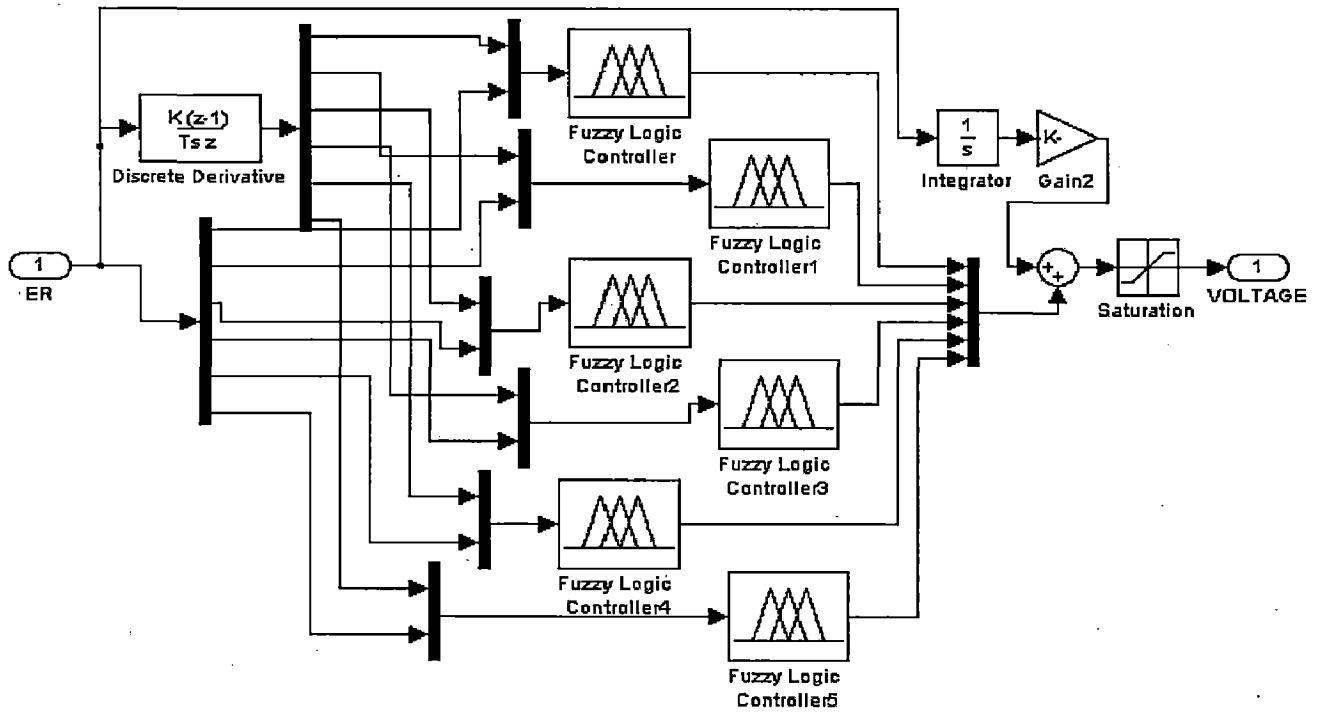


Fig.7.19 ANFIS controller

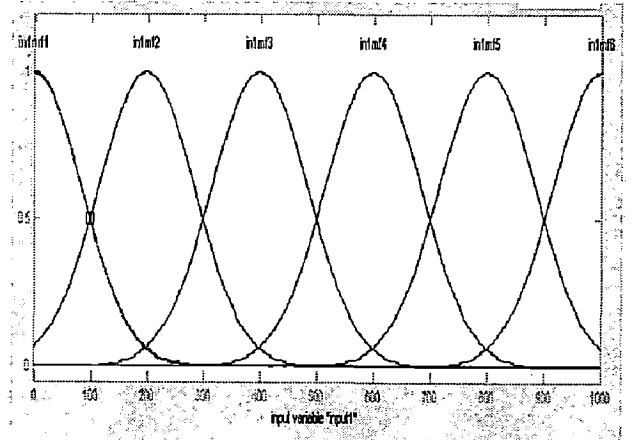
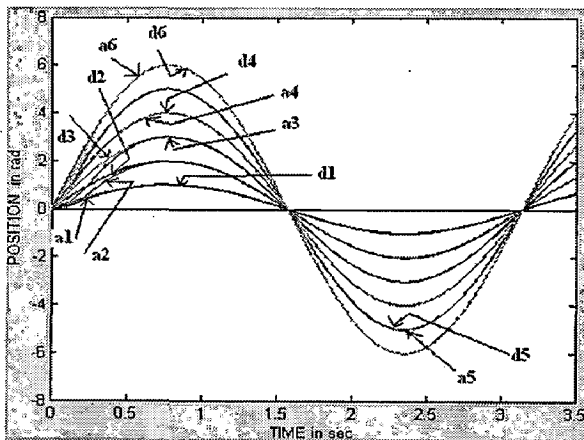


Fig.7.20 Robot Response with Neuro-Fuzzy control and membership functions after training

8. RESULTS

8.1 ERROR PROFILES OF THE ROBOT FOR TRAJECTORY CONTROL

There are six joints to be controlled in the ROBOT. Sine waves are chosen as desired trajectories with frequency of 2 rad/sec, maximum values of the desired trajectories are 1, 2, 3, 4, 5, 6 radians respectively, the error profiles of conventional control strategies without unmodeled term at each joint are shown in the figures from 8.1-8.6. Values of Integral absolute errors of conventional control strategies without unmodeled term are tabulated in Table I. From the Table I, we can clearly see that critically damped inverse dynamic (CDID) control performs well.

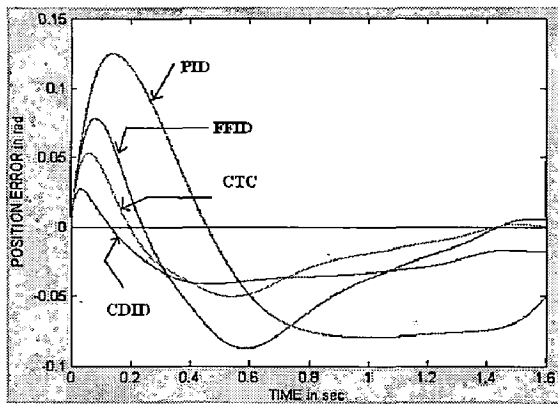


Fig.8.1 Error profiles at joint1

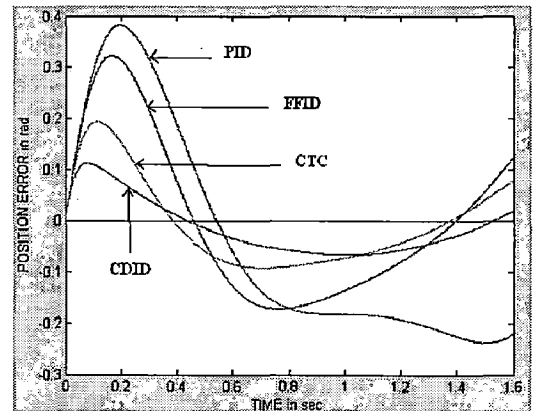


Fig.8.2 Error profiles at joint2

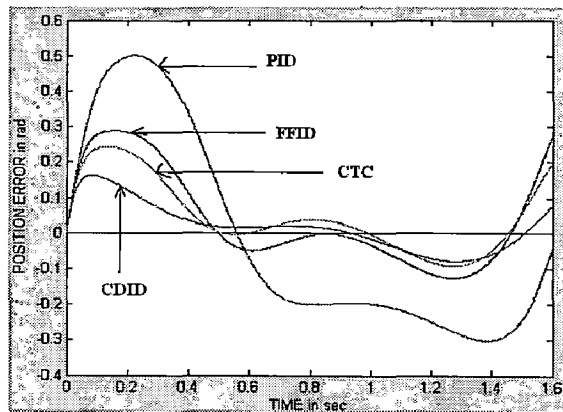


Fig.8.3 Error profiles at joint3

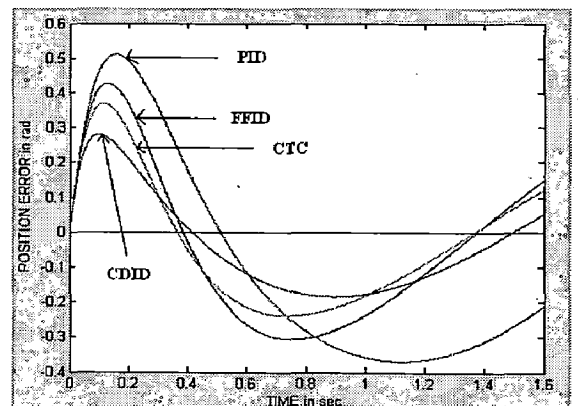


Fig.8.4 Error profiles at joint4

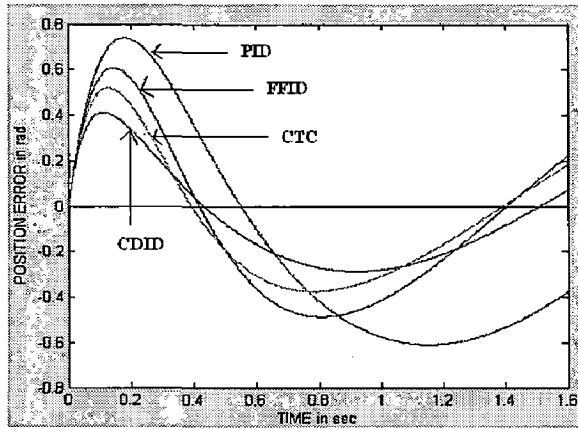


Fig.8.5 Error profiles at joint5

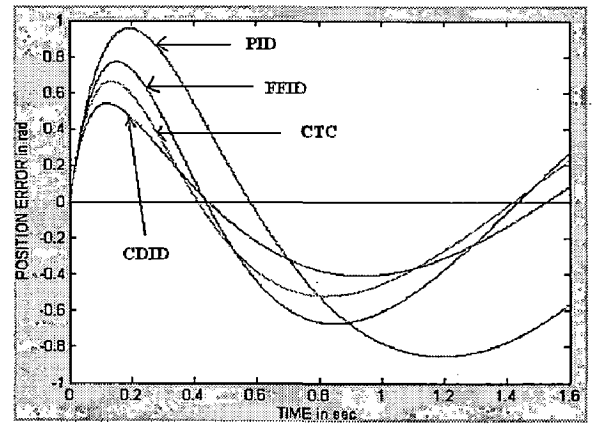


Fig.8.6 Error profiles at joint6

TABLE I
ERRORS (IN RAD) OF CONVENTIONAL CONTROLLERS WITHOUT UNMODELED TERM

CONTROL STRATEGY	Link1	Link2	Link3	Link4	Link5	Link6
PID	0.3565	1.0036	1.2932	1.4766	2.3330	3.1726
FFID	0.2087	0.6685	0.5529	1.0039	1.5588	2.1114
CTC	0.1310	0.3754	0.4254	0.8107	1.2427	1.6909
CDID	0.1201	0.2474	0.2805	0.6458	1.0014	1.3978

The error profiles of intelligent control strategies without unmodeled term at each joint are shown in the figures from 8.7-8.12. Values of Integral absolute errors of intelligent control strategies without unmodeled term are tabulated in Table II. From the Table II, we can clearly see that Neuro-Fuzzy control performs well. Integral absolute error of Feed forward inverse dynamic control and neural network based control are same because inverse model in FFID control strategy is replaced by neural network which has been trained off line to approximate the inverse dynamic model of the robot manipulator.

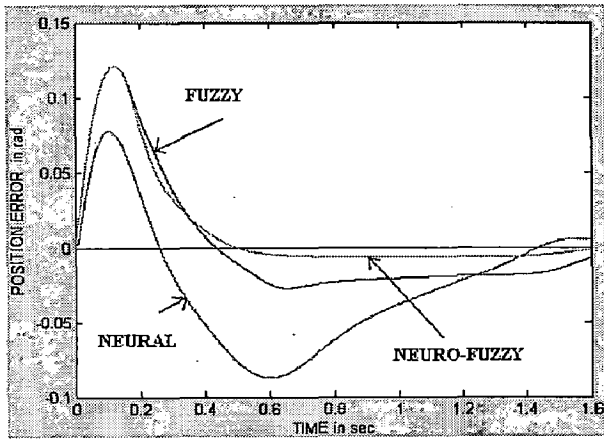


Fig.8.7 Error Profiles at joint1

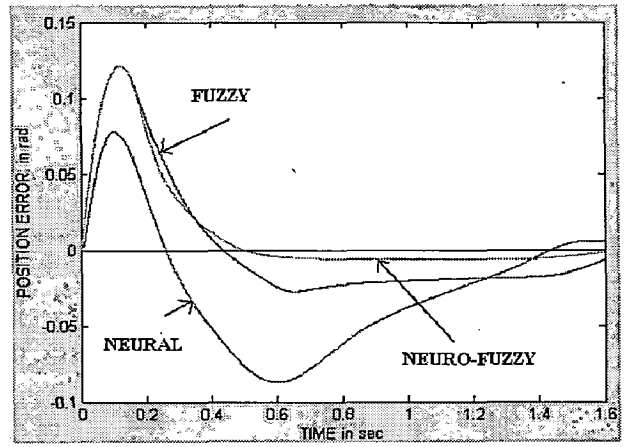


Fig.8.8 Error Profiles at joint2

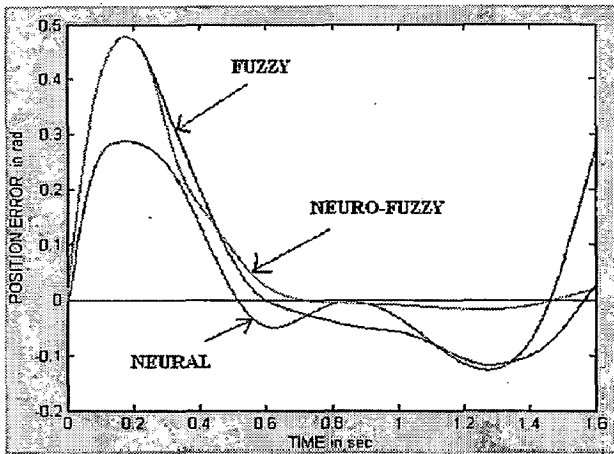


Fig.8.9 Error Profiles at joint3

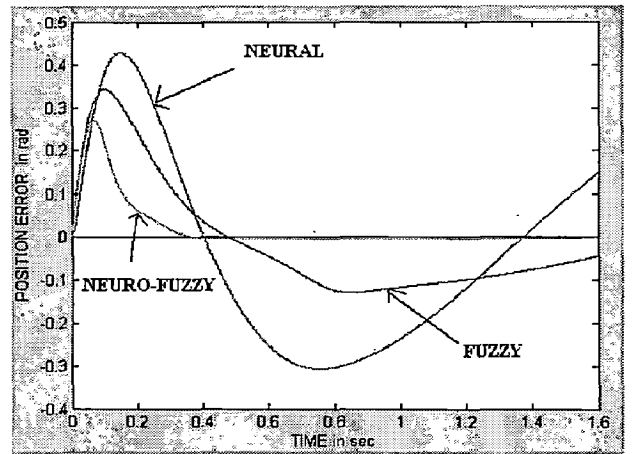


Fig.8.10 Error Profiles at joint4

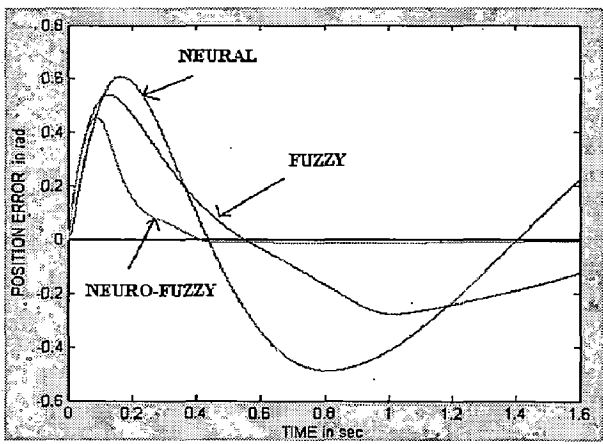


Fig.8.11 Error Profiles at joint5

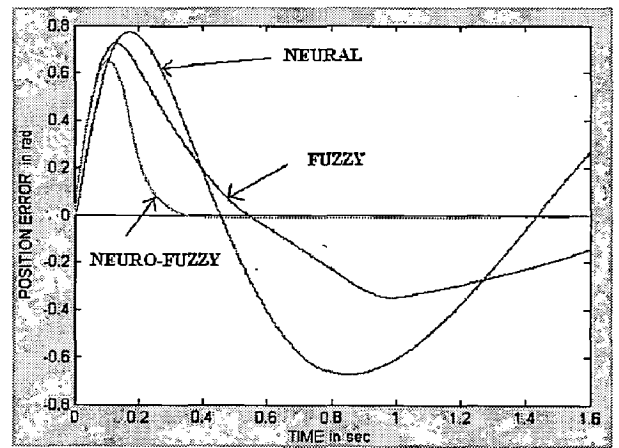


Fig.8.12 Error Profiles at joint6

TABLE II
ERRORS (IN RAD) OF INTELLIGENT CONTROLLERS WITHOUT UNMODELED TERM

CONTROL STRATEGY	Link1	Link2	Link3	Link4	Link5	Link6
FUZZY	0.1656	0.6212	0.7340	0.5934	1.1481	1.4705
NEURAL	0.2087	0.6685	0.5529	1.0039	1.5588	2.1114
NEURO-FUZZY	0.1165	0.4516	0.5123	0.1678	0.3303	0.4525

8.2. EFFECT OF UNMODELED TERM ON PERFORMANCE

Both conventional and intelligent controllers were tested for model with unmodeled term, i.e. A constant term is added to torque equation. Integral absolute error for conventional controllers with unmodeled term is tabulated in Table III. From Table we can clearly see that performances of conventional controllers are affected by unmodeled term is significant. Integral absolute error for intelligent controllers with unmodeled term is tabulated in Table IV. From Table we can clearly see that effect of unmodeled term is very less i.e. performance of intelligent controllers are remain same in both cases.

TABLE III
ERRORS (IN RAD) OF CONVENTIONAL CONTROLLERS WITH UNMODELED TERM

CONTROL STRATEGY	Link1	Link2	Link3	Link4	Link5	Link6
PID	0.2697	0.9664	1.2589	4.5100	6.4990	2.8316
FFID	0.1617	0.6479	0.5417	4.7064	6.7899	1.8673
CTC	0.0953	0.3497	0.4470	3.8280	5.4486	1.4960
CDID	0.0662	0.1963	0.3327	3.7081	5.2799	1.1308

TABLE IV
ERRORS (IN RAD) OF INTELLIGENT CONTROLLERS WITH UNMODELED TERM

CONTROL STRATEGY	Link1	Link2	Link3	Link4	Link5	Link6
FUZZY	0.1618	0.6496	0.7440	1.7906	1.5937	1.3272
NEURAL	0.1617	0.6479	0.5417	4.7064	6.7899	1.8673
NEURO-FUZZY	0.1078	0.4748	0.5284	0.1647	0.2820	0.4414

But the performance of neural network based controller is affected by unmodeled term because neural network has been trained off line to approximate the inverse dynamic model of the robot manipulator without unmodeled term. Integral absolute error of PID, FFID and Neural control before neural network trained to approximate inverse dynamics of robot with unmodeled term are shown in Table V.

TABLE V
ERRORS (IN RADIANS) OF PID, FFID, AND NEURAL CONTROL BEFORE NETWORK TRAINED TO SYSTEM WITH UNMODELED TERM

CONTROL STRATEGY	Link1	Link2	Link3	Link4	Link5	Link6
PID	0.2507	0.8220	1.4141	4.2827	6.1804	6.6187
FFID	0.1502	0.5695	1.4959	4.5077	6.5122	6.9878
NEURAL	0.1502	0.5695	1.4959	4.5077	6.5122	6.9878

In order to improve performance of neural network based controller, neural network has been trained offline to approximate inverse dynamics of robot with unmodeled term. IAE and error profiles of PID, FFID and Neural control after neural network trained to approximate inverse dynamics of robot with unmodeled term are shown in Table VI and Fig.8.13-8.18.

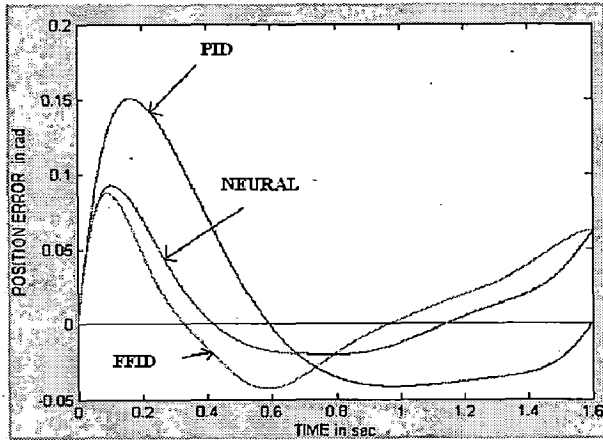


Fig.8.13 Error Profiles at joint1

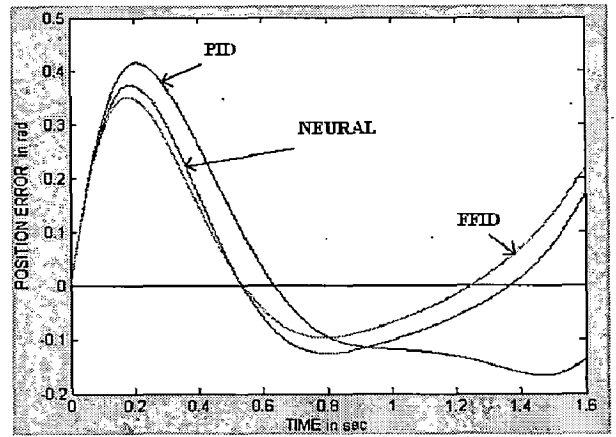


Fig.8.14 Error Profiles at joint 2

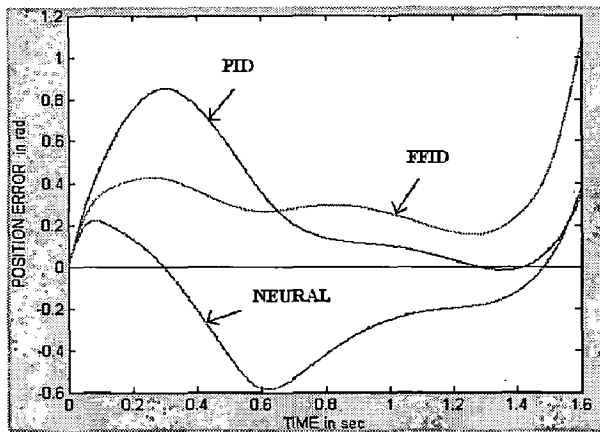


Fig.8.15 Error Profiles at joint3

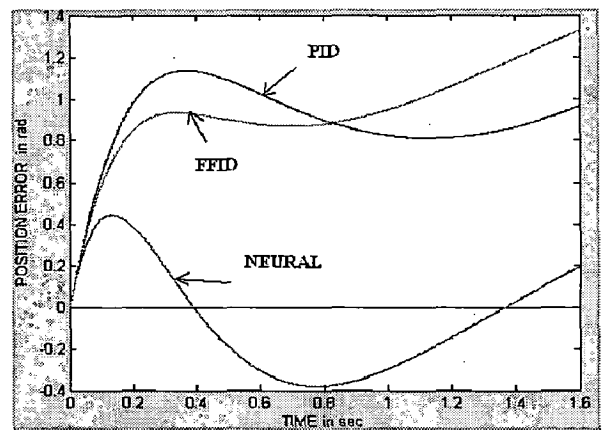


Fig.8.16 Error Profiles at joint 4

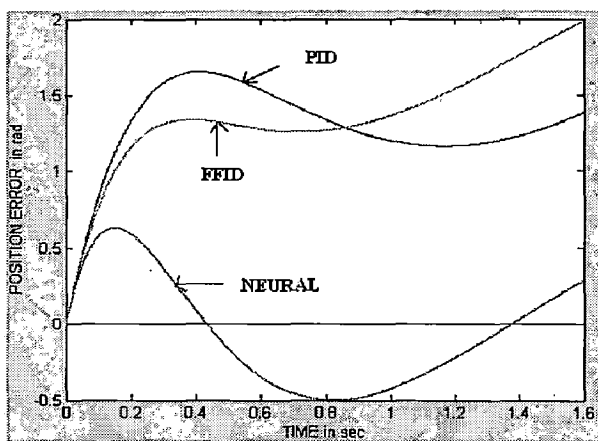


Fig.8.17 Error Profiles at joint5

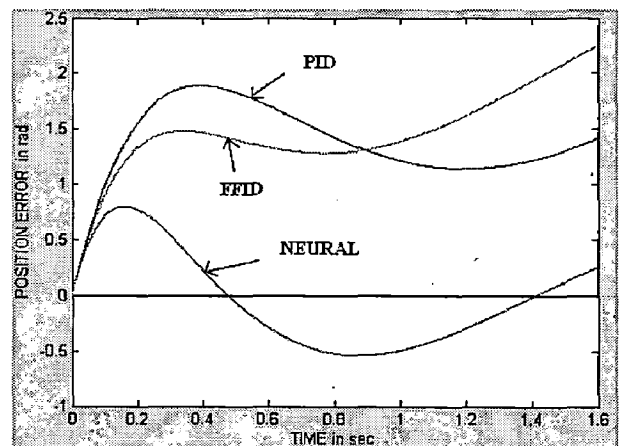


Fig.8.18 Error Profiles at joint 6

TABLE VI
 ERRORS (IN RADIANS) OF PID, FFID, AND NEURAL CONTROL AFTER
 NETWORK TRAINED TO SYSTEM WITH UNMODELED TERM

CONTROL STRATEGY	Link1	Link2	Link3	Link4	Link5	Link6
PID	0.2507	0.8220	1.4141	4.2827	6.1804	6.6187
FFID	0.1502	0.5695	1.4959	4.5077	6.5122	6.9878
NEURAL	0.1274	0.6187	1.2272	1.0979	1.5124	1.7479

CONCLUSIONS

From the simulated results, we conclude the following things.

1. Neruo-Fuzzy controller performs best in all conditions. This new method for control combines the advantage of neural networks (learning adaptability) with the advantage of fuzzy logic (use of expert knowledge) to achieve the goal of robust adaptive control of robot dynamics.

2. The CDID controller is the best performer in the category of conventional controllers. It is observed that the IAE for all links are considerably reduced in magnitude and also observed that when the unmodeled term is added to the model, PID and FFID perform badly. CTC and CDID performance also effected but they do well. It is conclude that CDID perform well in all conditions

3. Neuro-Fuzzy controller is the best performer in the category of intelligent controllers. It is observed that the IAE for all links are considerably reduced in magnitude and also observed that when the unmodeled term is added to the model, performance of intelligent controllers remains same except neural control.

4. Actually Neural control is similar to FFID and performances of both are almost similar in our simulation results. Performance of both controllers is still similar even when the unmodeled term is added to the model. Performance of Neural control is improved by training; training data is collected from model with the unmodeled term.

FUTURE SCOPE OF WORK

Main drawback of hand tuning of PID controller may not give good response. Genetic algorithm [9] can be used to get gains of PID which gives optimal performance.

The quality of a fuzzy logic controller can be drastically affected by the choice of membership functions and gains. Thus, methods for tuning fuzzy logic controllers are necessary. Here we have used hand tuning to select gains and general triangle membership are used which may not yield good performance. By using Genetic algorithm [13], we can tune both gains and parameters of membership functions in order to give optimal performance.

The Proposed ANFIS structure uses Temporal back propagation hybrid algorithm for the Learning of the ANFIS controller. The convergence time depends on the no of input membership functions, if the no of input membership functions increases then the learning process becomes slow and if the no of membership functions decreases then the Performance of the ANFIS controller will become poorer. There is a contradiction between convergence time and the performance. This is the main disadvantage of the Temporal back propagation hybrid algorithm which used for ANFIS. The convergence time can be improved by the Genetic based Neuro-Fuzzy approach All the parameters of the neuro fuzzy structure can be tuned simultaneously using Genetic Algorithm [21]

REFERENCES

1. R.K.Mittel, I.J.Nagrath, "**Robotics and Control**", Tata McGraw Hill publishing Company Limited, NEW DELHI.
2. John J.Craig, "**Introduction to Robotics Mechanics and Control**", Addison Wesley publications, second edition.
3. Brian Armstrong, Oussama Khatib, Joel Burdick, "**The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm**", Stanford Artificial Intelligence Laboratory Stanford University, IEEE Transactions and Systems 1986.
4. Peter I.corke, "**The Unimation Puma Servo System**", CSIRO Division of Manufacturing Technology, AUSTRALIA July 1994.
5. Peter I. Corke, "**A Search for Consensus among Model Parameters Reported for the PUMA 560 Robot**", CSIRO Division of Manufacturing Technology, Australia. Brain Armstrong-Helouvry University of Wisconsin Milwaukee, USA. July 1994.
6. P.K.Dash, S.K.Panda, T.H.Lee, J.X.Xu, A.Routray, "**Fuzzy and Neural Controllers for Dynamic Systems: an Overview**", Power Electronics and Drive Systems, 1997. Proceedings, 1997 International Conference on Vol 2, 26-29 May 1997, Page(s):810 - 816.
7. C.H.Atkeson, J.D.Griffiths, J.M.Hollerbach, C.H.An, "**Experimental Evaluation of feed forward and computed torque control**", Robotics and Automation, IEEE Transactions on Volume 5, Issue 3, June 1989 Page(s):368 – 373.
8. Sudeept Mohan, Surekha Bhanot, "**Conventional Control Strategies for Robot Manipulator: A Simulation Study**", International Conference on Computer Applications in Electrical Engineering Recent Advances, Roorkee, Sep.29-Oct.1, 2005.
9. D.P.kwok, T.P.Leung, Fang Sheng, "**Genetic Algorithms for Optimal Dynamic Control of Robot Arms**", International Conference on Industrial Electronics, Control, and Instrumentation, 1993. Proceedings of the IECON '93, vol.1, 15-19 Nov. 1993 Page(s):380 – 385.

10. Han-Xiong Li; H.B.Gatland, **“Conventional Fuzzy Control and its Enhancement”**, Systems, Man and Cybernetics, Part B, IEEE_Transactions on Volume 26, Issue 5, Oct. 1996 Page(s):791-797.
11. T.Brehm, K.S.Rattan, **“Hybrid fuzzy logic PID controller”**, Fuzzy Systems, 1994. IEEE World Congress on Computational Intelligence, Proceedings of the Third IEEE Conference, vol.3, 26-29 June 1994 Page(s):1682 – 1687.
12. G.M.Khoury, M.Saad, H.Y.Kanaan ,and C.Asmar, **“Fuzzy PID Control of a Five DOF Robot Arm”**, Journal of Intelligent and Robotic systems, Vol. 40 , Issue 3, July 2004, Pages: 299 - 320.
13. Abdollah Homaifar, Ed McCormick, **“Simultaneous Design of Membership Functions and Rule sets for Fuzzy Controllers Using Genetic Algorithms”**, IEEE Transactions on Fuzzy Systems, Vol. 3, No. 2, May 1995.
14. T.Ozaki, T.Suzuki, T.Furuhashi, S.Okuma, Y.Uchikawa, **“Trajectory Control of Robotic Manipulators Using Neural Networks”**, Industrial Electronics, IEEE Transactions on Volume 38, Issue 3, June 1991 Page(s):195 – 202.
15. P.Gupta and N.K.Sinha, **“Control of Robotic Manipulator - a Neural Network Approach”**, International Journal of System Science, vol.29, no.7, pp.723-730, 1998.
16. M. F.Moller, **“A Scaled Conjugate Gradient Algorithm for Fast Supervised Learning”**, Neural networks, vol. 6, pp.525-533, 1993.
17. J.-S.R.Jang, **“Fuzzy Controller Design Without Domain Experts”**, IEEE International Conference on Fuzzy Systems, 8-12 March 1992 Page(s):289 – 296.
18. Jyh-Shing, Roger Jang, **“ANFIS: Adaptive-Network-Based Fuzzy Inference System”**, IEEE Transactions on Systems, Man, and Cybernetics, Vol. 23, No.3, May/June 1993.
19. G.S.Sandhu, K.S.Rattan, **“Design of a Neuro-Fuzzy Controller”**, Systems, Man, and Cybernetics, 1997. Computational Cybernetics and Simulation, 1997 IEEE International Conference on Volume 4, 12-15 Oct. 1997 Page(s):3170 – 3175.
20. Manish Kumar, Devendra P.Garg, **“Intelligent Learning of Fuzzy Logic Controllers via Neural Network and Genetic Algorithm”**, Proceeding of 2004

JUSFA, 2004 Japan-USA Symposium on Flexible Automation, Denver, Colorado, July 19-21, 2004.

21. Teo Lian Seng, Marzuki Bin Khalid, "**Tuning of a Neuro-Fuzzy Controller by Genetic Algorithm**", IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics, Vol. 29, No. 2, April 1999.