DESIGN OF NEURO-FUZZY CONTROLLER FOR ROBOT MANIPULATOR

A DISSERTATION

Submitted in partial fulfilment 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

B.SRIKANTH REDDY

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

JUNE, 2005

I.D.NO. MT-296/2005-27/BSR-SK

CANDIDATE'S DECLARATION

I hereby declare that the work being presented in the dissertation entitled **"DESIGN OF NEURO-FUZZY CONTROLLER FOR ROBOT MANIPULATOR"** 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 2004 to June 2005, under the guidance of **Dr. Surendra Kumar,** 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: 29-06-2005
Place: Roorkee

B. Srikanth
(B.SRIKANTH REDDY)

CERTIFICATE

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

(Dr. Surendra Kumar)
Asst 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. Their keen interest and constant encouragement gave me the confidence to complete my thesis work successfully. This work is simply the reflection of their thoughts, ideas, and concepts. I am highly indebted to them for their kind and valuable suggestions and of course their valuable time during the period of this work. The huge quantum of knowledge I had gained during their inspiring guidance would be immensely beneficial for my future endeavors.

I am very thankful to Prof. H.O.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 my parents for their blessings and encouragement.

(B.SRIKANTH REDDY)

ABSTRACT

The aim of the thesis is to develop an efficient dynamic control such as point to point and continuous path control strategy for Robot manipulator using fuzzy logic controller (FLC) and Neuro-Fuzzy controller. Owing to the advantage of learning ability & unique characteristics, which enable them to control the Robot manipulators.

Interfacing the neural and fuzzy logic with the Robot manipulator is one of the means of getting the rapid convergence of actual joint angle trajectory to the desired joint angle trajectory.

In this work, we have taken the example of the PUMA560 robot (Programmable Universal Machine for Assembly) and modeled the Fuzzy controller and ANFIS Controller, we have studied also performances obtained from Fuzzy controller,ANFIS and conventional PID controller, with the help of SIMULINK, Fuzzy logic Toolbox of The MATLAB 7.01 Software. Results are compared with the conventional PID controller.

CONTENTS

LIST OF FIGURES

INTRODUCTION

1.1 INTRODUCTION TO ROBOT CONTROL

Up till now, the majority of practical approaches to the industrial robot arm controller design use traditional techniques, such as PD or PID controllers, by treating each joint of the manipulator as a simple linear servomechanism. In designing these kinds of controllers, the non-linear, coupled and time-varying dynamics of the mechanical part of the robot manipulator system are completely ignored, or are dealt with as disturbances. These methods generally give satisfactory perfomance when the robot operates at a low speed. However, when the links are moving simultaneously and at a high speed, the non-linear coupling effects and the interaction forces between the manipulator links, may degrade the performance of the overall system and increase the tracking errors. The disturbances and uncertainties in a task cycle, may also reduce the tracking quality of robot manipulators. Thus, these methods are only suitable for relatively slow manipulator motion and for limited-precision tasks.

The Computed Torque Control (CTC)[2] is commonly used in the research community. The CTC control law has the ability to make the error asymptotically stable if the dynamics of the robot are exactly known however, manipulators are subject to structured and/or unstructured uncertainty. Structured uncertainly is defined as the case of a correct dynamic model but with parameter uncertainty due to tolerance variances in the manipulator link properties, unknown loads, inaccuracies in the torque constants of the actuators, and others. Unstructured uncertainty describes the case of unmodeled dynamics which result from the presence of high-frequency modes in the manipulator, neglected time-delays and non-linear friction. It has been widely recognized that the tracking performance of the CTC method in high-speed operations is severely affected by the structured and unstructured uncertainties. To cope with the problem, some adaptive approaches have been proposed to maintain the tracking performance of the robotic manipulator in the presence of structured uncertainty. Some other researchers have also

tried to incorporate the intelligent controlling techniques into the controller design and good results were reported.

The ability of a machine to emulate human behavior has always been the goal of artificial intelligence. Neural networks and fuzzy logic systems are two of the most important results of research in the area of artificial intelligence. They have been effectively applied to everything from voice and image recognition to toasters and automobile transmissions. Neural networks are best known for their learning capabilities. Fuzzy logic is a method of using human skills and thinking processes in a machine.

While neural networks and fuzzy logic have added a new dimension to many engineering fields of study, their weaknesses have not been overlooked. In many applications, the training of a neural network requires millions of iterative calculations. Sometimes the network can not adequately learn the desired function. Fuzzy logic systems, on the other hand, acquire their knowledge from an expert who encodes his knowledge in a series of IF/THEN rules. Fuzzy logic systems are easy to understand because they mimic human thinking. The problem arises when systems have many inputs and outputs. Obtaining a rule base for large systems is difficult, if not impossible.

These weaknesses inherent in the two technologies and their complementary strengths, prompted the researchers to look at different ways of combining neural networks and fuzzy logic. Due to the relative infancy of this field of study, a consensus on the best way to utilize their strengths and compensate for their shortcomings has not yet been established. Consequently, research into neuro-fuzzy systems branches in many directions. The technique used in this work replaces the rule-base of a traditional fuzzy logic system with a back propagation neural network.

In this thesis it has been demonstrated that independent joint control can be used for the projection and execution of the trajectory tracking. The PID, Fuzzy logic and ANFIS controllers are used for controlling the robot and a comparison of their performances is done.

1.2 LITERATURE REVIEW ON FUZZY CONTROL

L.A.Zadeh, who first introduced the fuzzy logic theory in 1965,combined the multi valued logic, probability theory, artificial intelligence and neural networks to develop this digital control methodology that simulates human thinking by incorporating the impression inherent in all physical systems.[7]

Mamdani and his coworkers applied the Fuzzy control concept to several systems such as steam engines, warm level systems etc, In all cases, the fuzzy controller is located at the error channel and is composed by fuzzy algorithm that relays significant observed variable to control actions. The fuzzy rules employed depend on the type system under control as well as on the heuristic functions used [8]

M.maeda and S.Murakami, works on tuning of FLC.Most of the Practical processes under automatic control are non linear higher order systems and may have considerable dead time FLC always does not produce good approximation to the controller output required for optimum performances. Only statistic or fixed valued SF's and predefined MF's ma not be sufficient to eliminate this drawback. So either the input-output SF's or the definitions of fuzzy sets are to match the current plant characteristics[9]

S.Z.He, and his coworkers proposed a scheme for self tuning of a conventional PID controller using fuzzy rules The proportional sensitivity integral time and derivative time are initially calculated using Ziegler Nichols tuning formula these three parameters are then modified on line by a single parameter which is updated by rule base defined on error and derivative of error shows that there is considerable improvement in the overall performances of the controller over it s conventional type [10]

M.Yoshida, and his coworkers gives gain tuning method He assumes all processes as first order systems with dead time. The input and output SF's are calculated by some empirical relations involving process parameters. Good control performances for higher order systems cannot be ensured by this technique.

H.X.Li and H.B.Gutland are given more emphasis on the tuning of in put and output SF's than that of MF's or rule base. They basically suggested a trail and error

method for tuning of input and output SF's for a fuzzy PID controller developed from two FLC's in parallel one is a PI type and other is PD type.

1.3 LITERATURE REVIEW ON NEURO-FUZZY CONTROL

Berenji was the first to develop FLC that is capable of learning as well as tuning its parameters by using neural network reinforcement learning method[13]

Jang designs a self-learning fuzzy controller based on temporal back propagation. The current state of the system is compared to the desired state and the error is back propagated through the system to adjust individual fuzzy parameters this system is also called as ANFIS. By testing on an inverted pendulum showed that significant adjustments were made on membership function definitions. The trained system exhibited robustness and fault tolerance. Since then great deal of work don neuro-fuzzy control[14]

Mones Iskarous and Kazuhiko Kawamura explained about the modeling of the physical system with neuro-fuzzy system and explained the control procedure with the modeled system[15]

Gurupreet S. Sandhu and Kuldio S.rahan explained about designing of neuro-fuzzy controller with the learning procedure, the architecture of the neuro-controller with an example of simple transfer function they designed the P controller using neuro-fuzzy approach they showed the results that the neuro controller performance is better than the P controller[16]

Wen Yu, Xiaoou Li explained about the neuro fuzz modeling using Stable Learning Algorithm and they showed the result that the this learning algorithm is far better than the conventional algorithms[17]

Oscar Castillo, Patrica Melin are applied the neuro-fuzzy control technique to the robot systems they have taken the example of simple two link robot system and designed the neuro-fuzzy controller and the compared the results with the conventional fuzzy control and showed that response using neuro-fuzzy control is far better than the conventional Fuzzy control[18] .

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 non-linearities 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 an 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'. [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 manpulator. 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[4]. 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

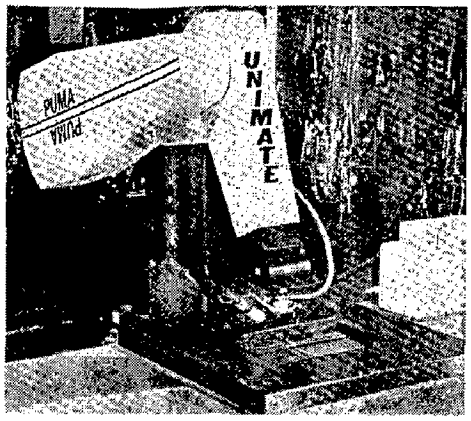


FIGURE 3.17 The Unimation PUMA 560.

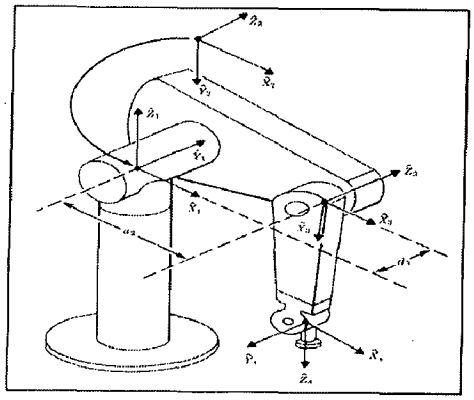


FIGURE 3.18 Some kinematic parameters and frame assignments for the PUMA 560 manipulator.

Fig. 2.1 a. puma 560 robot b. Illustration of Puma 560 robot

2.3 DYNAMICS OF PUMA560 ROBOT

The general form of the robot arm dynamic equation is

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

Where $U_i = P_i e + D_i \frac{de}{dt}$

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

$$
\begin{aligned}
m_{21} &= m_{12} & \qquad m_{62} &= m_{26} \\
m_{31} &= m_{13} & \qquad m_{43} &= m_{34} \\
m_{41} &= m_{14} & \qquad m_{53} &= m_{35} \\
m_{51} &= m_{15} \quad \text{and} & \qquad m_{63} &= m_{36} \\
m_{61} &= m_{16} & \qquad m_{54} &= m_{45} \\
m_{32} &= m_{23} & \qquad m_{64} &= m_{46} \\
m_{42} &= m_{24} & \qquad m_{65} &= m_{56} \\
m_{52} &= m_{25}
\end{aligned}
\tag{2. 2}
$$

All other elements in the mass matrix are zero

CORIOLIS/CENTRIFUGAL TORQUES MATRIX

$$cor_{11} = (-1.38*c_1*s_1*\dot{q}_1) + 0.5*\dot{q}_2*(0.6*s_{23}*c_{23} - 0.744*s_2*s_{23} + 0.744*c_2*c_{23}) + 0.5*\dot{q}_3*(0.6*s_{23}*c_{23} - 0.744*s_2*s_{23} + 0.744*c_2*c_{23})$$

$$cor_{12} = 0.5*\dot{q}_1*(0.6*s_{23}*c_{23} - 0.744*s_2*s_{23} + 0.744*c_2*c_{23}) + 0.5*\dot{q}_2*(1.38*c_2 + 0.268*s_{23} - 0.0476*s_2) + 0.5*\dot{q}_3*(0.268*s_{23} - 0.00397*c_{23})$$

$$cor_{13} = 0.5*\dot{q}_1*(0.6*s_{23}*c_{23} + 0.744*c_2*c_{23}) + 0.5*\dot{q}_2*(0.268*s_{23} - 0.00397*c_{23}) + 0.5*\dot{q}_3*(0.268*s_{23} - 0.00794*c_{23})$$

$$cor_{21} = 0.5*\dot{q}_1*(-0.6*s_{23}*c_{23} + 0.744*s_2*s_{23} - 0.744*c_2*c_{23}) + 0.199*\dot{q}_3*c_{23}$$

$$cor_{22} = 0.372*\dot{q}_3*c_3$$

$$cor_{23} = 0.00199*\dot{q}_1*c_{23} + 0.372*\dot{q}_2*c_3 + 0.5*\dot{q}_3*(0.744*c_3 + 0.022*s_3)$$

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

$$cor_{32} = 0.372*\dot{q}_3*c_3$$

$$cor_{33} = 0.00199*\dot{q}_1*c_{23} + 0.372*\dot{q}_2*c_3 + 0.5*\dot{q}_3*(0.744*c_3 + 0.022*s_3)$$

(2. 3)

All other coriolis matrix elements are zero

GRAVITY MATRIX

$g_1 = 0$, $g_6 = 0$

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

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

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

$$g_5 = -0.028 * (\cos_{23} * \sin_5 + \sin_{23} * \cos_4 * \cos_5) \quad (2.4)$$

All these information and derivation of these equations for PUMA560 given in [4]

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 [5].

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

$$\tau_{mi} = k_{mi} i_{ai} \tag{2.5}$$

Where k_{mi} is known as the motor-torque proportional constant in *N-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:

$$e_{bi} = k_{bi} \dot{\theta}_{mi} \tag{2.6}$$

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

Where k_{bi} is proportionality constant in *V- 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

$$v_{ai} = e_{bi} + i_{ai} R_{ai} + L_{ai} \frac{di_{ai}}{dt} \tag{2.7}$$

By solving we will get

$$i_{ai} = \frac{1}{L} \int_{-\infty}^{t} (v_{ai} - e_{bi} - i_{ai} R_{ai}) dt \tag{2.8}$$

SPECIFICATION FOR THE ACTUATOR

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

$$-97.6N-m \le \tau_1 \le 97.6N-m$$
$$-186.4N-m \le \tau_2 \le 186.4N-m$$
$$-89.4N-m \le \tau_3 \le 89.4N-m$$
$$-24.2N-m \le \tau_4 \le 24.2N-m$$
$$-20.1N-m \le \tau_5 \le 20.1N-m$$
$$-21.3N-m \le \tau_6 \le 21.3N-m$$

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

For all actuators

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

Details are given in[4,5,6]

31 INTRODUCION

Due to the 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 a basis of 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 fuzzy but the controller is processing exact input-data and is producing exact output-data in a deterministic way.

3.2 HISTORICAL BACKGROUND

Since the introduction of the theory of fuzzy sets by L. A. Zadeh in 1965[7], and the industrial application of the first fuzzy controller by E. H. Mamdani in 1974[8], fuzzy systems have obtained a major role in engineering systems and consumer products in the 1980s and 1990s. New theoretical results and new applications are presented continuously.

A reason for this significant role is that fuzzy computing provides a flexible and powerful alternative to construct 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 the maintenance 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 achieved easily, further improvements need a lot of. Especially when the

number of inputs increases, the maintenance of the multi-dimensional rule base is time-consuming.

3.3 FUZZY CONTROL

A block diagram of a fuzzy control system is shown in figure the fuzzy controller is composed of the following four elements:

1. A *rule-base* (a set of If-Then rules), which contains a fuzzy logic quantification of the expert's linguistic description of how to achieve good control.
2. An *inference mechanism* (also called an "inference engine" or "fuzzy inference" module), which emulates the expert's decision making in interpreting and applying knowledge about how best to control the plant.
3. A *fuzzification* interface, which converts controller inputs into information that the inference mechanism can easily use to activate and apply rules.
4. A *defuzzification* interface, which converts the conclusions of the inference mechanism into actual inputs for the process.

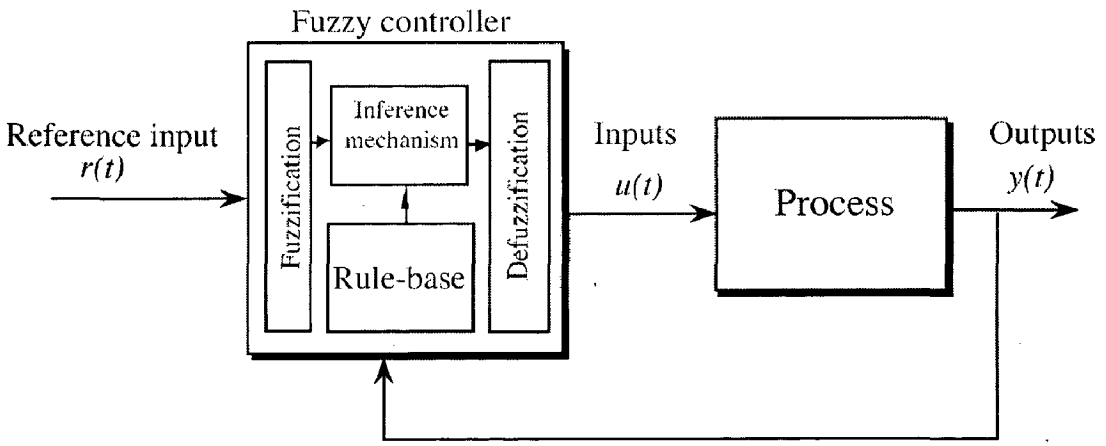


Fig.3.1 Fuzzy controller

We introduce each of the components of the fuzzy controller for a simple problem of balancing an inverted pendulum on a cart, as shown in Figure 3.2. Here, y denotes the angle that the pendulum makes with the vertical (in radians), l is the half-pendulum length (in meters), and u is the force input that moves the cart (in Newton's). We will use r to denote the desired angular position of the pendulum. The goal is to balance the pendulum in the upright position (i.e., $r = 0$) when it initially

starts with some nonzero angle off the vertical (i.e., $y \neq 0$). This is a very simple and academic nonlinear control problem, and many good techniques already exist for its solution. Indeed, for this standard configuration, a simple PD controller works well even in implementation.

In the remainder of this section, we will use the inverted pendulum as a convenient problem to illustrate the design and basic mechanics of the operation of a fuzzy control system.

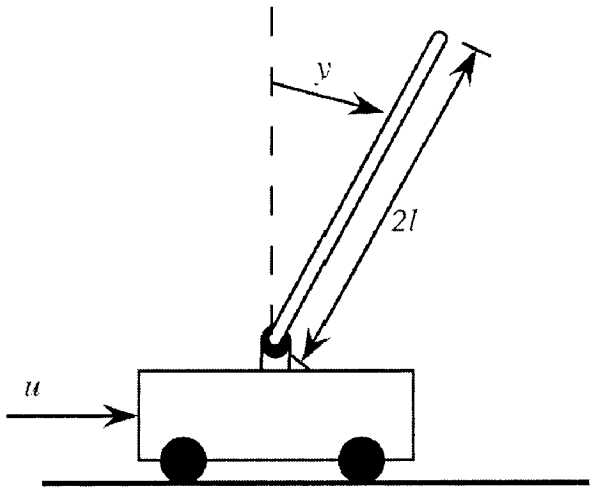


Fig3.2 inverted pendulum

3.4 CHOOSING FUZZY CONTROLLER INPUTS AND OUTPUTS

$\frac{d}{dt}e(t)$ Consider a human-in-the-loop whose responsibility is to control the pendulum, as shown in Figure 3.3. The fuzzy controller is to be designed to automate how a human expert who is successful at this task would control the system. First, the expert tells us (the designers of the fuzzy controller) what information she or he will use as inputs to the decision-making process. Suppose that for the inverted pendulum, the expert says that she or he will use

$$e(t) = r(t) - y(t)$$

And

$$\frac{d}{dt}e(t)$$

as the variables on which to base decisions. Certainly, there are many other choices (e.g., the integral of the error e could also be used) but this choice makes good intuitive sense. Next, we must identify the controlled variable. For the inverted pendulum, we are allowed to control only the force that moves the cart, so the choice here is simple.

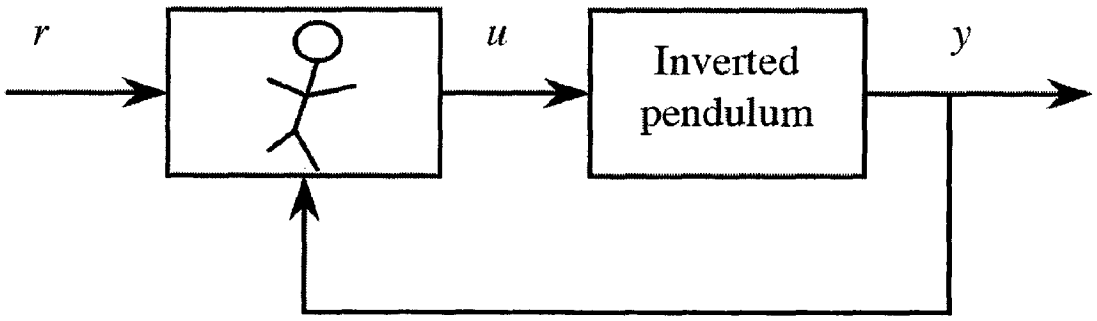


Fig .3.3 Inverted pendulum on a cart

Once the fuzzy controller inputs and outputs are chosen, you must determine what the reference inputs are. For the inverted pendulum, the choice of the reference input $r = 0$ is clear. In some situations, however, you may want to choose r as some nonzero constant to balance the pendulum in the off-vertical position. To do this, the controller must maintain the cart at a constant acceleration so that the pendulum will not fall.

3.5 PUTTING CONTROL KNOWLEDGE INTO RULE-BASES

Suppose that the human expert shown in Figure 3.3 provides a description of how best to control the plant in some natural language (e.g., English). We seek to take this "linguistic" description and load it into the fuzzy controller, as indicated by the arrow in Figure3. 4.

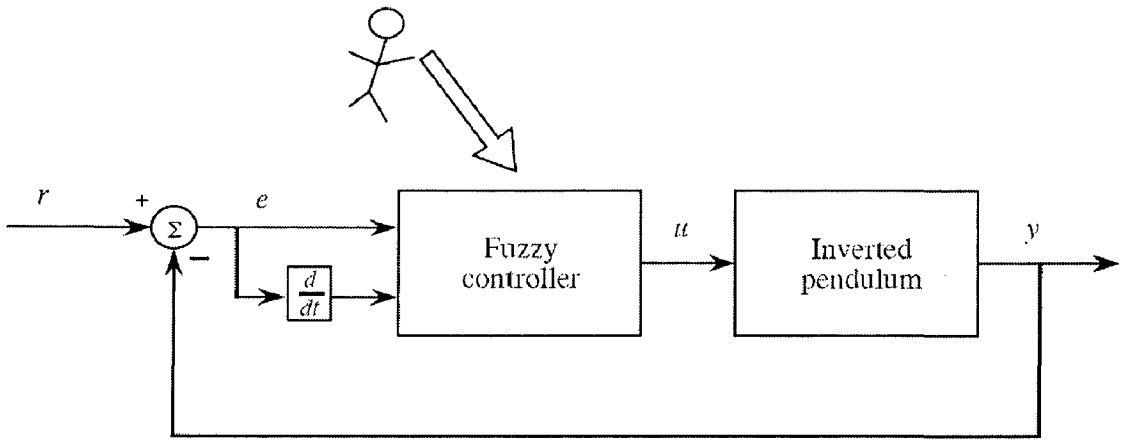


Fig.3.4 Fuzzy controller for an inverted pendulum

Linguistic Descriptions

The linguistic description provided by the expert can generally be broken into several parts. There will be "linguistic variables" that describe each of the time varying fuzzy controller inputs and outputs. For the inverted pendulum,

"error" describes $e(t)$

"change-in-error" describes $\frac{d}{dt}e(t)$

"force" describes $u(t)$

Just as $e(t)$ takes on a value of, for example, 0.1 at $t = 2$ ($e(2) = 0.1$), linguistic variables assume "linguistic values." That is, the values that linguistic variables take on over time change dynamically. Suppose for the pendulum example that "error," "change-in-error," and "force" take on the following values:

"neglarge"

"negsmall"

"zero"

"possmall"

"poslarge"

Note that we are using "negsmall" as an abbreviation for "negative small in size" and so on for the other variables. Such abbreviations help keep the linguistic descriptions short yet precise. For an even shorter description we could use integers:

"−2" to represent "neglarge"

"−1" to represent "negsmall"

"0" to represent "zero"

"1" to represent "possmall"

"2" to represent "poslarge"

The linguistic variables and values provide a language for the expert to express her or his ideas about the control decision-making process in the context of the framework established by our choice of fuzzy controller inputs and outputs. Recall that for the inverted pendulum $r = 0$ and $e = r - y$ so that

$$e = -y$$

and

$$\frac{d}{dt}(e) = -\frac{d}{dt}(y)$$

since $\frac{d}{dt}(r) = 0$. First, we will study how we can quantify certain dynamic behaviors with linguistics. In the next subsection we will study how to quantify knowledge about how to control the pendulum using linguistic descriptions.

For the inverted pendulum each of the following statements quantifies a different configuration of the pendulum

• The statement "error is poslarge" can represent the situation where the pendulum is at a significant angle to the *left* of the vertical.

• The statement "error is negsmall" can represent the situation where the pendulum is just slightly to the right of the vertical, but not too close to the vertical to justify quantifying it as "zero" and not too far away to justify quantifying it as "neglarge."

• The statement "error is zero" can represent the situation where the pendulum is very near the vertical position (a linguistic quantification is not precise, hence we are willing to accept any value of the error around $e(t) = 0$ as being quantified linguistically by "zero" since this can be considered a better quantification than "possmall" or "negsmall").

• The statement "error is poslarge **and** change-in-error is possmall" can represent the situation where the pendulum is to the left of the vertical and, since $\frac{d}{dt}(y)$ the pendulum is moving *away* from the upright position (note that in this case the pendulum is moving counterclockwise).

• The statement "error is negsmall **and** change-in-error is possmall" can represent the situation where the pendulum is slightly to the right of the vertical and, since $\frac{d}{dt}(y) <$

0, the pendulum is moving *toward* the upright position (note that in this case the pendulum is also moving counterclockwise).

It is important to study each of the cases above to understand how the expert's linguistics quantify the dynamics of the pendulum (actually, each partially quantifies the pendulum's state).

Rules

Next, we will use the above linguistic quantification to specify a set of rules (a rule-base) that captures the expert's knowledge about how to control the plant. In particular, for the inverted pendulum in the three positions shown in Figure 2.5, we have the following rules (notice that we drop the quotes since the whole rule is linguistic):

1. **If** error is neglarge **and** change-in-error is neglarge **Then** force is poslarge

This rule quantifies the situation in Figure 3.5(a) where the pendulum has a large positive angle and is moving clockwise; hence it is clear that we should apply a strong positive force (to the right) so that we can try to start the pendulum moving in the proper direction.

2. **If** error is zero **and** change-in-error is possmall **Then** force is negsmall

This rule quantifies the situation in Figure 3.5(b) where the pendulum has nearly a zero angle with the vertical (a linguistic quantification of zero does not imply that $e(t) = 0$ exactly) and is moving counterclockwise; hence we should apply a small negative force (to the left) to counteract the movement so that it moves toward zero (a positive force could result in the pendulum overshooting the desired position).

3. **If** error is poslarge **and** change-in-error is negsmall **Then** force is negsmall

This rule quantifies the situation in Figure 3.5(c) where the pendulum is far to the left of the vertical and is moving clockwise; hence we should apply a small negative force (to the left) to assist the movement, but not a big one since the pendulum is already moving in the proper direction.

The general form of the linguistic rules listed above is

If premise **Then** consequent

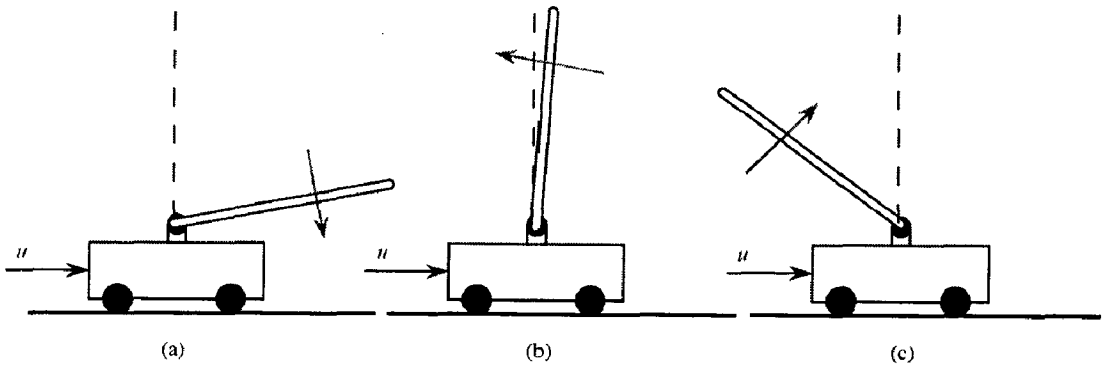


Fig.3.5 Inverted pendulum in various positions.

Rule-Bases

Using the above approach, we could continue to write down rules for the pendulum problem for all possible cases. Note that since we only specify a finite number of linguistic variables and linguistic values, there is only a finite number of possible rules. For the pendulum problem, with two inputs and five linguistic values for each of these, there are at most 52 = 25 possible rules (all possible combinations of premise linguistic values for two inputs).

A tabular representation of one possible set of rules for the inverted pendulum is shown in Table 2.1. Notice that the body of the table lists the linguistic-numeric consequents of the rules, and the left column and top row of the table contain the linguistic-numeric premise terms. Then, for instance, the (2,−1) position (where the "2" represents the row having "2" for a numeric-linguistic value and the "−1" represents the column having "−1" for a numeric-linguistic value) has a −1 ("negsmall") in the body of the table and represents the rule

If error is poslarge **and** change-in-error is negsmall **Then** force is negsmall

which is rule 3 above. Table 3.1 represents abstract knowledge that the expert has about how to control the pendulum given the error and its derivative as inputs.

TABLE 3.1 Rule Table for the Inverted Pendulum

"force" u		"change-in-error" $\dot{e}$				
		-2	-1	0	1	2
"error" e	-2	2	2	2	1	0
	-1	2	2	1	0	-1
	0	2	1	0	-1	-2
	1	1	0	-1	-2	-2
	2	0	-1	-2	-2	-2

3.6 FUZZY QUANTIFICATION OF KNOWLEDGE

Up to this point we have only quantified,we will show how to use fuzzy logic to fully quantify the meaning of linguistic descriptions so that we may automate, in the fuzzy controller, the control rules specified by the expert.

Membership Functions

First, we quantify the meaning of the linguistic values using "membership functions." Consider, for example, Figure 3.6. This is a plot of a function μ versus $e(t)$ that takes on special meaning. The function μ quantifies the *certainty* that $e(t)$ can be classified linguistically as "possmall." To understand the way that a membership function works, it is best to perform a case analysis where we show how to interpret it for various values of $e(t)$:

• If $e(t) = -\pi/2$ then $\mu(-\pi/2) = 0$, indicating that we are certain that $e(t) = -\pi/2$ is *not* "possmall."

• If $e(t) = \pi/8$ then $\mu(\pi/8) = 0.5$, indicating that we are halfway certain that $e(t) = \pi/8$ is "possmall" (we are only halfway certain since it could also be "zero" with some degree of certainty—this value is in a "gray area" in terms of linguistic interpretation).

• If $e(t) = \pi/4$ then $\mu(\pi/4) = 1.0$, indicating that we are absolutely certain that $e(t) = \pi/4$ is what we mean by "possmall."

• If $e(t) = \pi$ then $\mu(\pi) = 0$, indicating that we are certain that $e(t) = \pi$ is not "possmall" (actually, it is "poslarge").

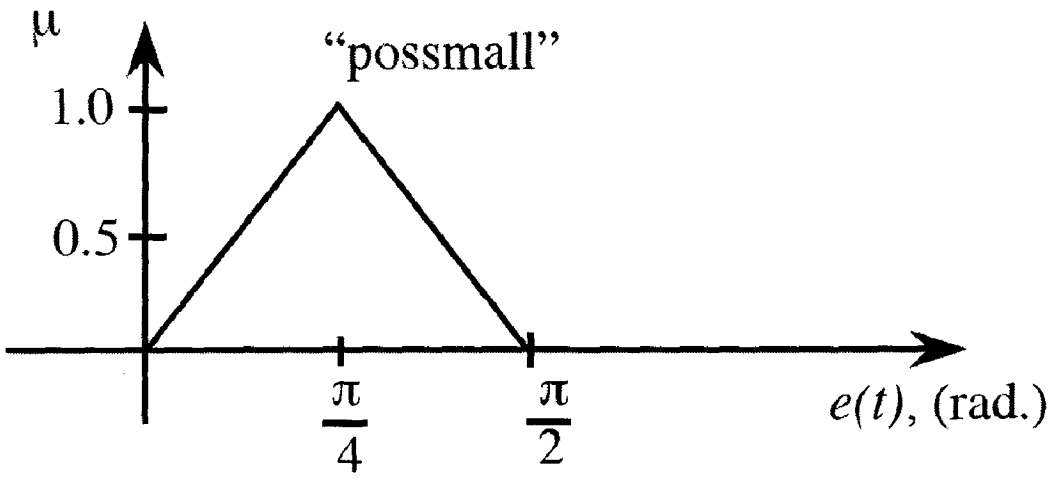


Fig. 3.6 Membership functions for linguistic value "possmall."

Figure 3.6 is only one possible definition of the meaning of "error is possmall"; you could use a bell-shaped function, a trapezoid, or many others.

Now that we know how to specify the meaning of a linguistic value via a membership function (and hence a fuzzy set), we can easily specify the membership functions for all 15 linguistic values (five for each input and five for the output) of our inverted pendulum example.

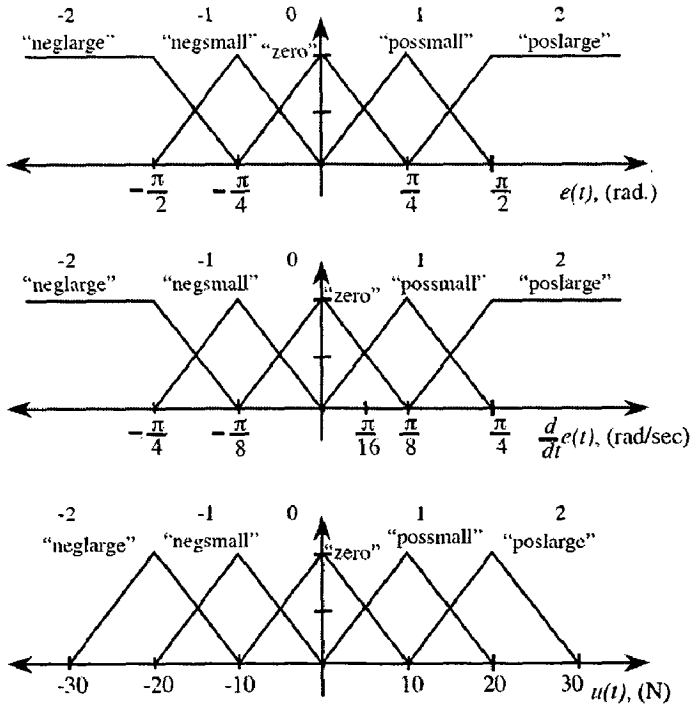


Fig.3.7 Membership functions for an inverted pendulum

3.7 MATCHING: DETERMINING WHICH RULES TO USE

The inference process generally involves two steps:

1. The premises of all the rules are compared to the controller inputs to determine which rules apply to the current situation. This "matching" process involves determining the certainty that each rule applies, and typically we will more strongly take into account the recommendations of rules that we are more certain apply to the current situation.
2. The conclusions (what control actions to take) are determined using the rules that have been determined to apply at the current time. The conclusions are characterized with a fuzzy set (or sets) that represent the certainty that the input to the plant should take on various values.

We will cover step 1 in this subsection and step 2 in the next.

Premise Quantification via Fuzzy Logic

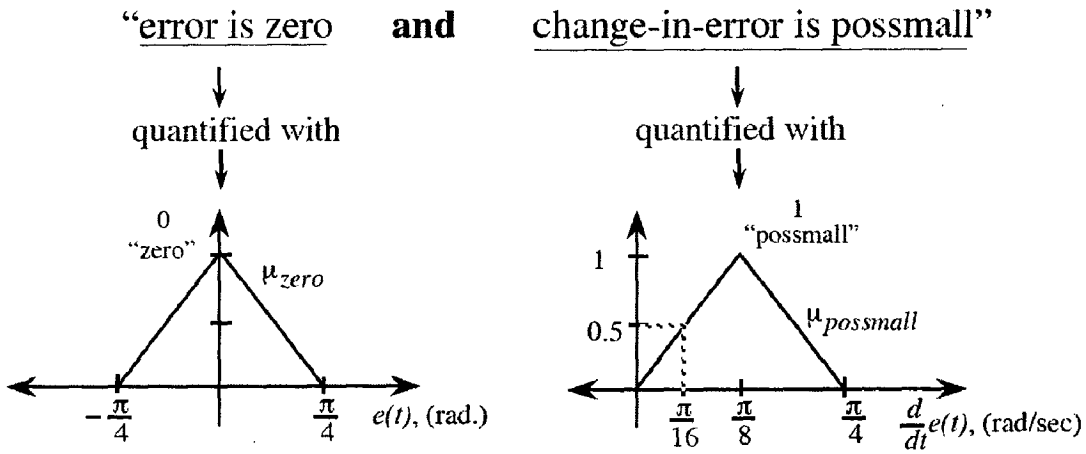


Fig.3.8 Membership functions of premise term

To see how to quantify the "and" operation, begin by supposing that $e(t) = \pi/8$ and d

$\frac{d}{dt}e(t) = \pi/32$, so that using Figure 2.8we see that $\mu_{premise}\left(\frac{d}{dt}e(t)\right)$

$$\mu_{zero}(e(t)) = 0.5\,\mu_{(1)}(u) = \min\{0.25, \mu_{zero}(u)\}$$

and

$$\mu_{possmall}\left(\frac{d}{dt}e(t)\right) = 0.25$$

What, for these values of $e(t)$ and $\frac{d}{dt}e(t)$, is the certainty of the statement

"error is zero **and** change-in-error is possmall"

that is the premise from the above rule? We will denote this certainty by $\mu_{premise}$.

There are actually several ways to define it:

• *Minimum:* Define $\mu_{premise}$ = min {0.5, 0.25} = 0.25, that is, using the minimum of the two membership values.

• *Product:* Define $\mu_{premise}$ = (0.5)(0.25) = 0.125, that is, using the product of the two membership values.

Determining Which Rules Are On

Determining the applicability of each rule is called "matching." We say that a rule is "on at time t" if its premise membership function $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right) > 0$.

Consider, for the inverted pendulum example, how we compute the rules that are on. Suppose that

$e(t)=0$;

and

$$\frac{d}{dt}e(t) = \pi/8 - \pi/32 = 0.294$$

Figure 3.12 shows the membership functions for the inputs and indicates with thick black vertical lines the values above for $e(t)$ and $\frac{d}{dt}e(t)$. Notice that $\mu_{zero}(e(t)) = 1$ but that the other membership functions for the $e(t)$ input are all "off" (i.e., their values are zero). For the $\frac{d}{dt}e(t)$ input we see that $\mu_{zero}\left(\frac{d}{dt}e(t)\right) = 0.25$ and $\mu_{possmall}\left(\frac{d}{dt}e(t)\right) = 0.75$ and that all the other membership functions are off. This implies that rules that have the premise terms

"error is zero"

"change-in-error is zero"

"change-in-error is possmall"

are on (all other rules have $\mu_{premise}\left(e(t), \frac{d}{dt}e(t)\right) = 0$. So, which rules are these? Using Table 3.1 we find that the rules that are on are the following:

1. **If** error is zero **and** change-in-error is zero **Then** force is zero
2. **If** error is zero **and** change-in-error is possmall **Then** force is negsmall

Note that since for the pendulum example we have at most two membership functions overlapping, we will never have more than four rules on at one time. Actually, for this system we will either have one, two, or four rules on at any one time. To get only one rule on choose, for example, $e(t) = 0$ and $\frac{d}{dt}e(t) = \pi$ 8 so that only rule 2 above is on.

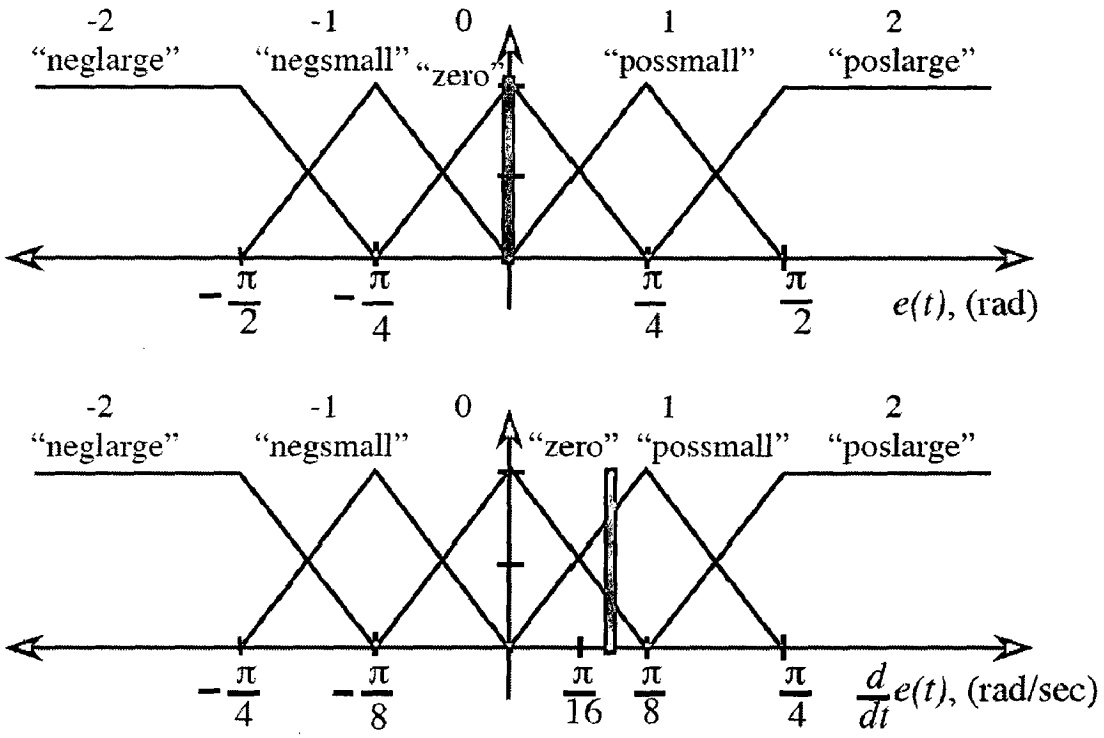


Fig.3.9 Input membership functions with input values.

3.8 INFERENCE STEP: DETERMINING CONCLUSIONS

Next, we consider how to determine which conclusions should be reached when the rules that are on are applied to deciding what the force input to the cart carrying the inverted pendulum should be. To do this, we will first consider the recommendations of each rule independently. Then later we will combine all the recommendations from all the rules to determine the force input to the cart.

Recommendation from One Rule

Consider the conclusion reached by the rule

If error is zero **and** change-in-error is zero **Then** force is zero

Which for convenience we will refer to as "rule (1)." Using the minimum to represent the premise, we have

$$\mu_{\text{premise}_{(1)}} = \min\{0.25,\ 1\} = 0.25$$

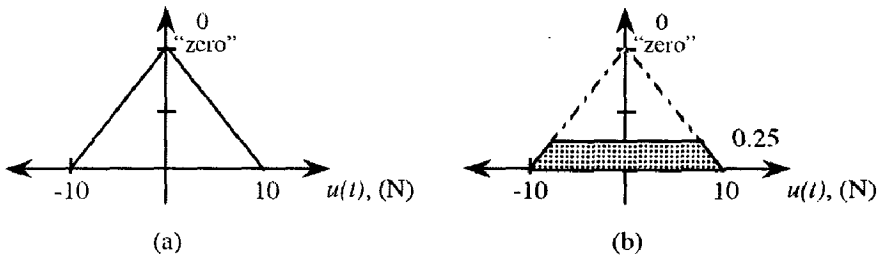


Fig.3.10 (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_{(1)}(u)$ for rule (1)

Recommendation from Another Rule

Next, consider the conclusion reached by the other rule that is on,

If error is zero **and** change-in-error is possmall **Then** force is negsmall

which for convenience we will refer to as "rule (2)." Using the minimum to represent the premise, we have

$$\min\{0.75,\ 1\} = 0.75$$

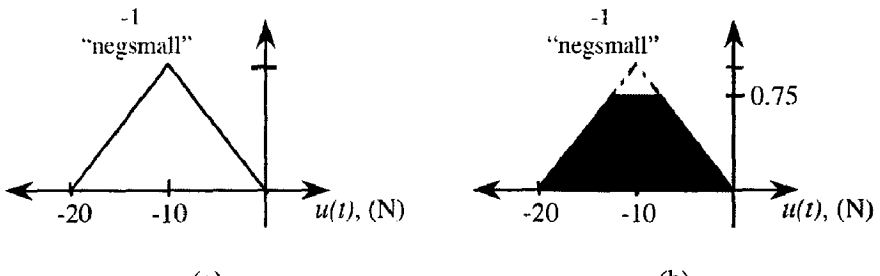


Fig.3.11 (a) Consequent membership function and (b) implied fuzzy set with membership function $\mu_{(2)}(u)$ for rule (2).

This completes the operations of the inference mechanism

3.9 CONVERTING DECISIONS INTO ACTIONS

To understand defuzzification, it is best to first draw all the implied fuzzy sets on one axis as shown in Figure 3.12. We want to find the one output, which we denote by "u^{crisp}" that best represents the conclusions of the fuzzy controller that are represented with the implied fuzzy sets. There are actually many approaches to defuzzification. We will consider two here

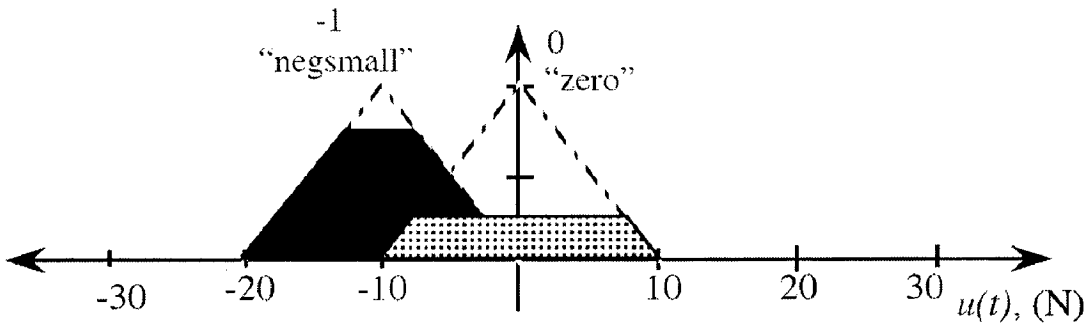


Fig. 3.12 Implied fuzzy sets.

Combining Recommendations

Due to its popularity, we will first consider the "center of gravity" (COG) defuzzification method for combining the recommendations represented by the implied fuzzy sets from all the rules. Let b_i denote the center of the membership function (i.e., where it reaches its peak for our example) of the consequent of rule (*i*). For our example we have

$$b1 = 0.0$$

and

$$b2 = -10$$

as shown in Figure 3.12 Let

$$\int \mu_{(i)}$$

denote the area under the membership function $\mu_{(i)}$.The COG method computes u^{crisp} to be

$$u^{crisp} = \frac{\sum b_i \int \mu_i}{\sum \int \mu_i} \tag{3.1}$$

Using Equation (3.1) with Figure 3.15 we have

$$u^{crisp} = \frac{(0)(4.375) + (-10)(9.375)}{4.375 + 9.375} = -6.81$$

as the input to the pendulum for the given $e(t)$ and $\frac{d}{dt}\, e(t)$

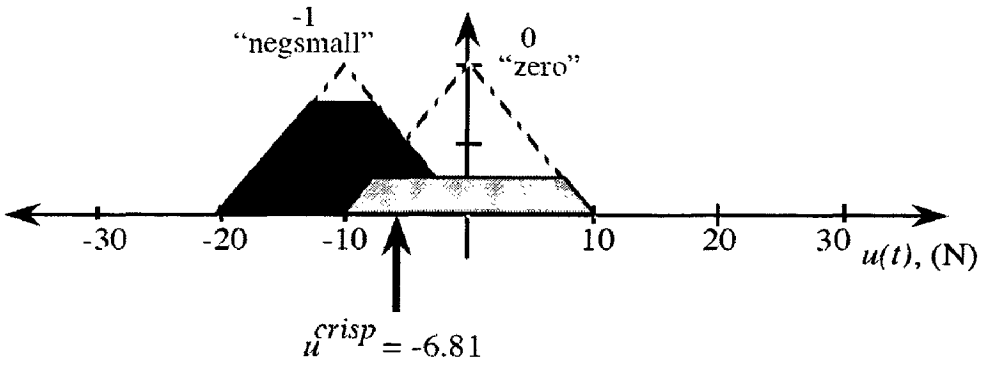


Fig.3.13 Implied fuzzy sets.

It is interesting to note that for our example it will be the case that

$$-20 \le u^{crisp} \le 20$$

To see this, consider Figure 3.14, where we have drawn the output membership functions. Notice that even though we have extended the membership functions at the outermost edges past −20 and +20 (see the shaded regions), the COG method will never compute a value outside this range

.

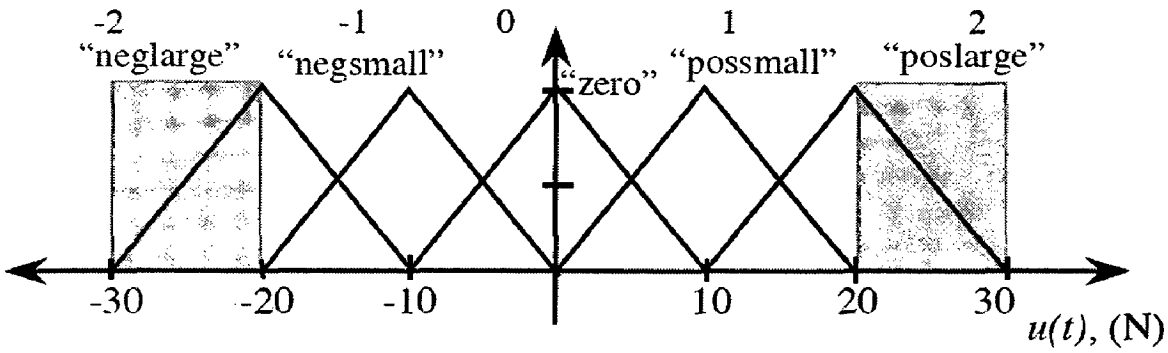


Fig.3.14 Output membership functions.

3.10 GRAPHICAL DEPICTION OF FUZZY DECISION MAKING

Here, we use the minimum operator to represent the "and" in the premise and the implication and COG defuzzification. Product operator is used to represent the "and" in the premise and the implication, and choose values of $e(t)$ and $\frac{d}{dt}e(t)$ that will result in four rules being on. Then, repeat the process when center-average defuzzification is used with either minimum or product used for the premise. Also, learn how to picture in your mind how the parameters of this graphical representation of the fuzzy controller operations change as the fuzzy controller inputs change.Refer [3] for more details on fuzzy control

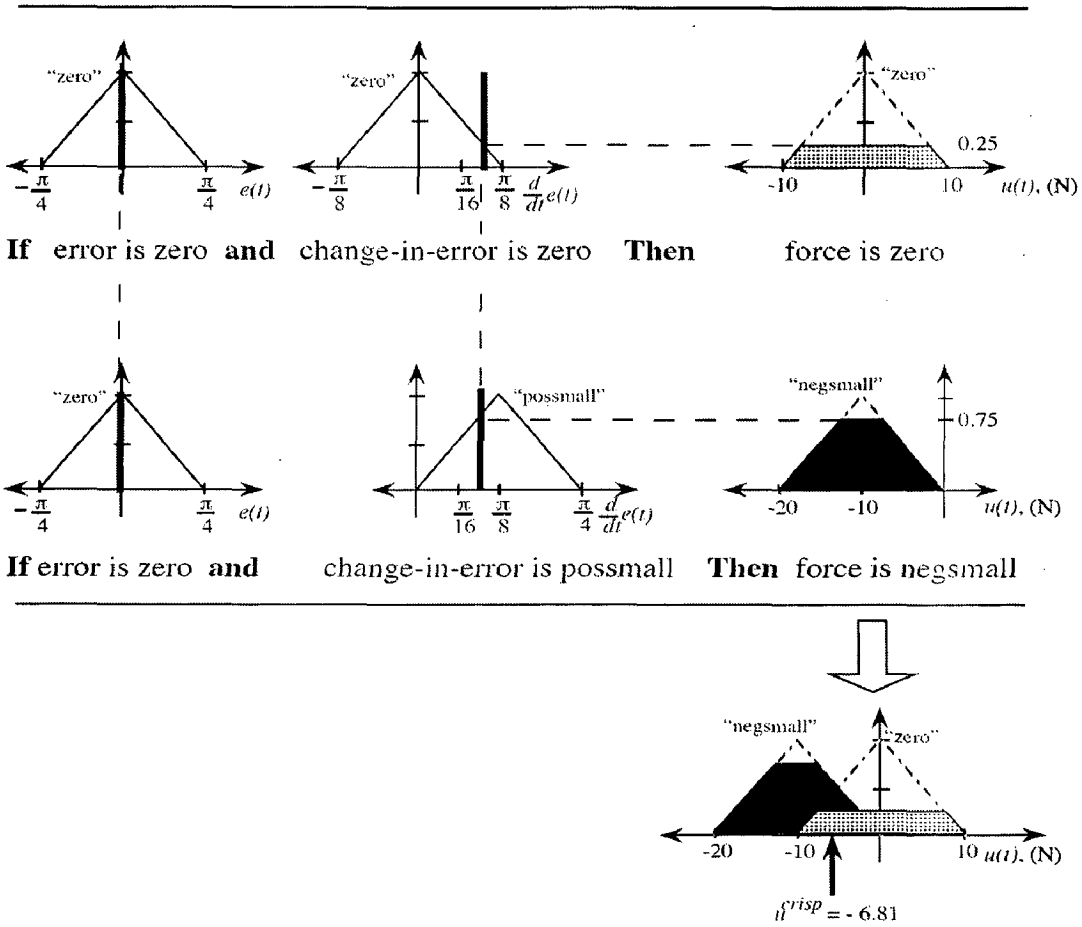


Fig 3.15 Graphical representations of fuzzy controller operations.

4.1 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 PD 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[13]. 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 3-1 is a diagram of such a system.

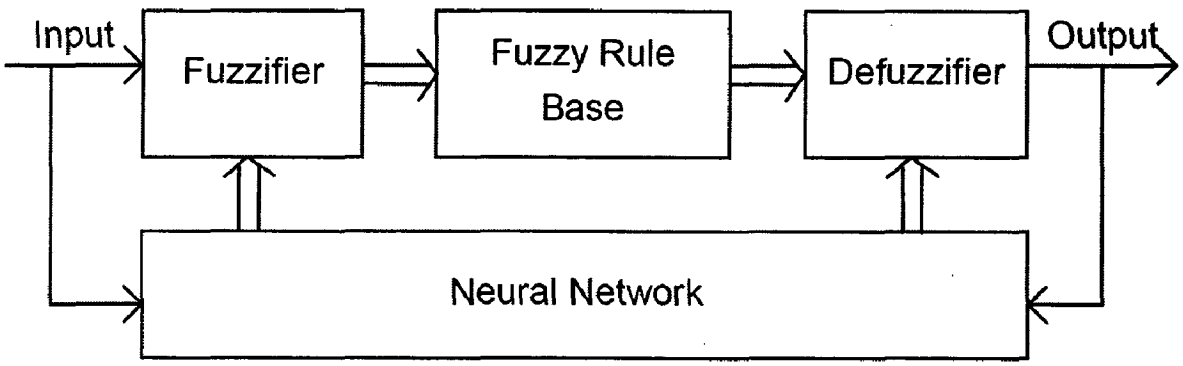


Figure.4.1 A fuzzy system whose membership functions are adjusted by a neural network.

Figure 4.2 shows a more complex integration, the use of neural networks to determine both the fuzzy membership functions and the rule base[14]. 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.

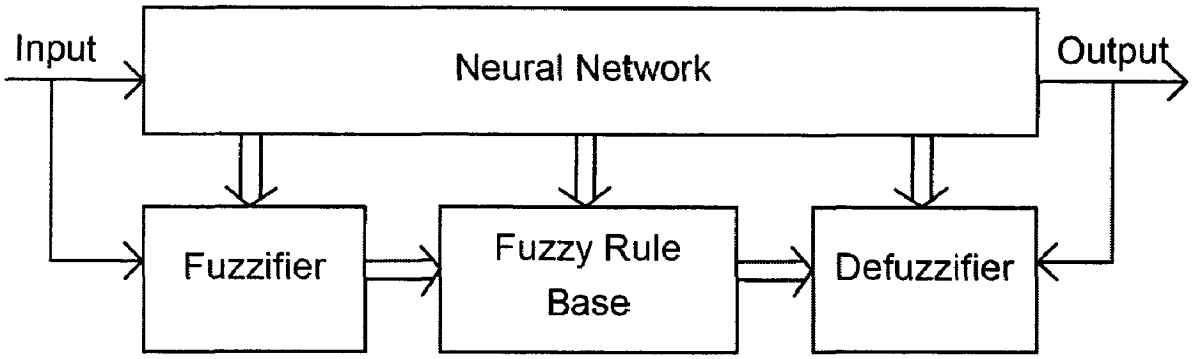


Fig .4.2 A fuzzy system defined by a neural network.

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 4.3.

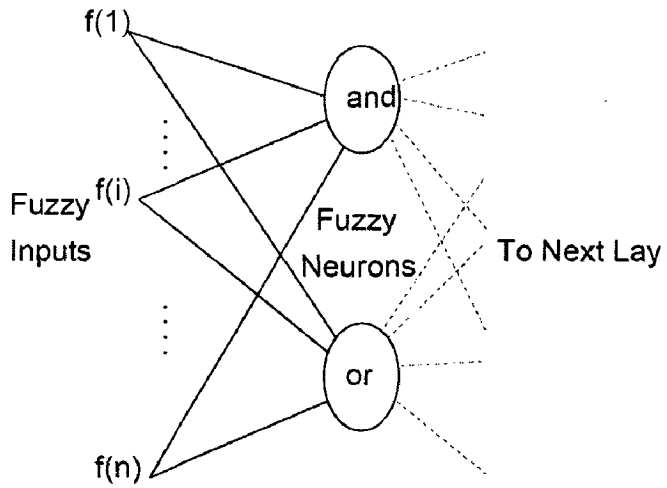


Figure .4.3. A neural network of fuzzy neurons.

Finally, the idea of fuzzification of control variables into degrees of membership in fuzzy sets has been integrated into neural networks. See figure4. 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

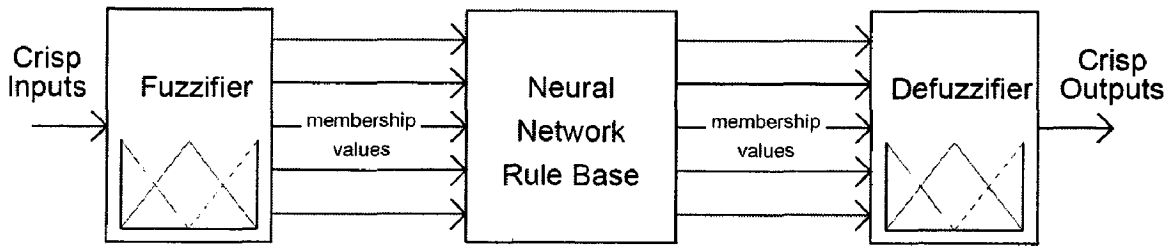


Fig.4.4 A fuzzy system with neural network rule base

4.2 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 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.

A. Architecture and Basic Learning Rule

An adaptive network (Figure 4.5) is a multi-layer feedforward network in which each node performs a particular function (node function) on incoming signals as well as a set of parameters pertaining to this node. The nature of the node functions may vary from node to node, and the choice of each node function depends on the overall input-output function which the adaptive network is required to carry out. Note that the links in an adaptive network only indicate the flow direction of signals between nodes; no weights are associated with the links.

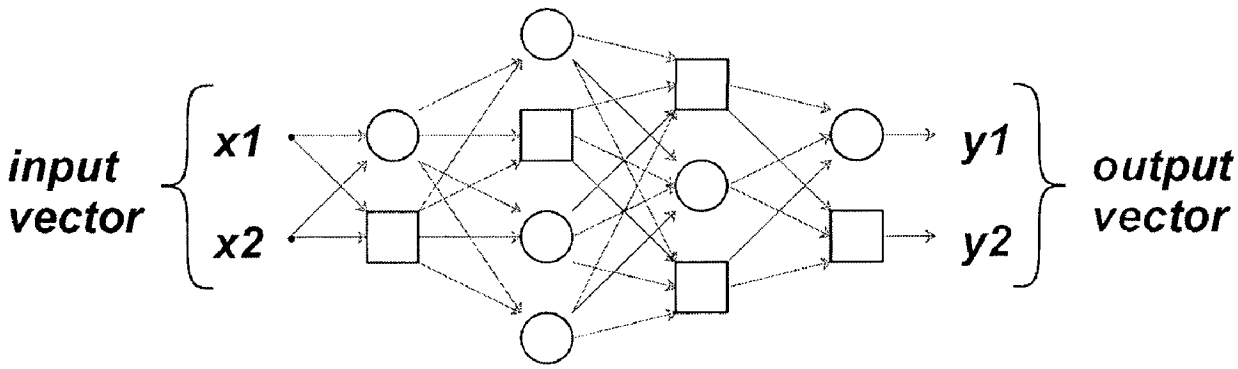


Figure 4.5: An adaptive network

To reflect different adaptive capabilities, we use both circle and square nodes in an adaptive network. A square node (adaptive node) has parameters while a circle node (fixed node) has none. The parameter set of an adaptive network is the union of the parameter sets of each adaptive node In order to achieve a desired input-output mapping, these parameters are updated according to given training data and a gradient-based learning procedure described below

Suppose that a given adaptive network has L layers and the k-th layer has $\neq(k)$ nodes. We can denote the node in the i-th position of the *k-th* layer by *(k, i)*, and its node function (or node output) by O_i^k. Since a node output depends on its incoming signals and its parameter set, we have

$$O_i^k = O_i^k\left(O_i^{k-1},\ldots O_{\neq(k-1)}^{k-1}, a, b, c, \ldots\right), \tag{4.1}$$

where *a, b, c,* etc. are the parameters pertaining to this node. (Note that we use O_i^k as both the node output and node function.)

Assuming the given training data set has *P* entries, we can define the *error measure* (or *energy function*) for the *p-th* $(1 < p < P)$ entry of training data entry as the sum of squared errors

$$E_p = \sum_{m=1}^{\neq(L)}\left(T_{m,p} - O_{m,p}^L\right)^2, \tag{4.2}$$

Where Tm,p is the m-th component of p-th target output vector, and $O_{m,p}^L$ is the m-th component of actual output vector produced by the presentation of the p-th input vector.

Hence the overall error measure is $E = \sum_{p=1}^{p} E_p$

In order to develop a learning procedure that implements gradient descent in E over the parameter space, first we have to calculate the error rate $\frac{\partial E_p}{\partial O}$ for p-th training data and for each node output O. The error rate for the output node at (L,i) can be calculated readily from equation (4.2)

$$\frac{\partial E_p}{\partial O_{i,p}^k} = -2(T_{i,p} - O_{i,p}^L) \tag{4.3}$$

For the internal node at (k, i), the error rate can be derived by the chain rule:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{\neq(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^k}, \tag{4.4}$$

learning (or on-line learning). In the following we will derive a faster hybrid learning rule and both of its learning paradigms

B. Hybrid Learning Rule: Batch Learning

Though we can apply the gradient method to identify the parameters in an adaptive network, the method is generally slow and likely to become trapped in local minima. Here we propose a hybrid learning rule which combines the gradient method and the least squares estimate (LSE) to identify parameters.

For simplicity, assume that the adaptive network under consideration has only one output

$$\text{output} = F(\bar{I}, S), \tag{4.9}$$

where $\bar{I}$ is the set of input variables and S is the set of parameters. If there exists a function H such that the composite function H o F is, linear in some of the elements of S, then these elements can be identified by the least squares method. More formally, if the parameter set 5 can be decomposed into two sets

$$S = S_1 \oplus S_2 \tag{4.10}$$

(where $\oplus$ represents direct sum) such that HoF is linear in the elements of S_2, then upon applying H to equation (4.9), we have

$$H(output) = HoF(\bar{I}, S), \tag{4.11}$$

Which is linear in the elements of S_2. Now given values of elements of S_1, we can plug P training data into equation (4.11) and obtain a matrix equation

$$AX = B \tag{4.12}$$

Where X is an unknown vector whose elements are parameters in S_2 Let $|S_2| = M$, then the dimensions of A, X and B are P x M, M x 1 and P x 1, respectively. Since P (number of training data pairs) is usually greater than M (number of linear parameters), this is an over determined problem and generally there is no exact solution to equation (4.12).

Instead, a least squares estimate (LSE) of X, X*, is sought to minimize the squared error $\|AX - B\|^2$. This is a standard problem that forms the grounds for linear regression, adaptive filtering and signal processing. The most well-known formula for X* uses the pseudo-inverse of X:

$$X^* = \left(A^T A\right)^{-1} A^T B, \tag{4.13}$$

Where A^T is the transpose of A, and $\left(A^T A\right)^{-1} A^T$ is the pseudo-inverse of A if $A^T A$ is non-singular. While equation (4.13) is concise in notation, it is expensive in computation when dealing with the matrix inverse and, moreover, it becomes ill-defined if $A^T A$ is singular. As a result, we employ sequential formulas to compute the LSE of X. This sequential method of LSE is more efficient (especially when M is small) and can be easily modified to an on-line version (see below) for systems with changing characteristics. Specifically, let the ith row vector of matrix A defined in equation (4.12) be a_i^T and the ith element of B be b_i^T, then X can be calculated

$$X_{i+1} = X_i + S_{i+1} a_{i+1} \left(b_{i+1}^T - a_{i+1}^T X_i\right)$$

$$S_{i+1} = S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}}, \qquad \text{i=0,1.....p-1} \tag{4.14}$$

where S_i is often called the covariance matrix and the least squares estimate X* is equal to X_p. The initial conditions to bootstrap equation (3.14) are $X_0 = 0$ and $S_0 = \gamma I$, where 7 is a positive large number and I is the identity matrix of dimension M x M. When dealing with multi-output adaptive networks (output in equation (4.9) is a column vector), equation (4.14) still applies except that b_i^T is the i-th rows of matrix B.

Now we can combine the gradient method and the least squares estimate to update the parameters in an adaptive network. Each epoch of this hybrid learning procedure is composed of a forward pass and a backward pass. In the forward pass, we supply input data and functional signals go forward to calculate each node output until the matrices A and B in equation (4.12) are obtained, and the parameters in S_2 are identified by the sequential least squares formulas in equation (4.14). After identifying parameters in S_2, the functional signals keep going forward till the error measure is calculated. In the backward pass, the error rates (the derivative of the error measure w.r.t. each node output,

see equation (4.3) and (4.4)) propagate from the output end toward the input end, and the parameters in S_1 are updated by the gradient method in equation (4.7).

For given fixed values of parameters in S_1, the parameters in S_2 thus found are guaranteed to be the global optimum point in the S_2 parameter space due to the choice of the squared error measure. Not only can this hybrid learning rule decrease the dimension of the search space in the gradient method, but, in general, it will also cut down substantially the convergence time.

Take for example an one-hidden-layer back-propagation neural network with sigmoid activation functions. If this neural network has p output units, then the output in equation (4.9) is a column vector. Let $H(.)$ be the inverse sigmoid function

$$H(x) = \ln\left(\frac{x}{1-x}\right) \tag{4.15}$$

Then equation (4.11) becomes a linear (vector) function such that each element of H(output) is a linear combination of the parameters (weights and thresholds) pertaining to layer 2. In other words,

S_1 = weights and thresholds of hidden layer,
S_2 = weights and thresholds of output layer

Therefore we can apply the back-propagation learning rule to tune the parameters in the hidden layer, and the parameters in the output layer can be identified by the least squares method. However, it should be keep in mind that by using the least squares method on the data transformed by H(.) ,the obtained parameters are optimal in terms of the transformed squared error measure instead of the original one. Usually this will not cause practical problem as long as H(.) is monotonically increasing.

C. Hybrid Learning Rule: Pattern Learning

If the parameters are updated after each data presentation, we have the pattern learning or online learning paradigm. This learning paradigm is vital to the on-line parameter identification for systems with changing characteristics. To modify the batch

learning rule to its on-line version, it is obvious that the gradient descent should be based on E_P(see equation (4.5)) instead of E. Strictly speaking, this is not a truly gradient search procedure to minimize E, yet it will approximate to one if the learning rate is small.

For the sequential least squares formulas to account for the time-varying characteristics of the incoming data, we need to decay the effects of old data pairs as new data pairs become available. One simple method is to formulate the squared error measure as a weighted version that gives higher weighting factors to more recent data pairs. This amounts to the addition of a forgetting factor λ A to the original sequential formula

$$X_{i+1} = X_i + S_{i+1} a_{i+1} \left(b_{i+1}^T - a_{i+1}^T X_i \right)$$

$$S_{i+1} = \frac{1}{\lambda} \left[S_i - \frac{S_i a_{i+1} a_{i+1}^T S_i}{1 + a_{i+1}^T S_i a_{i+1}} \right] \tag{4.16}$$

4.3 ANFIS: ADAPTIVE-NETWORK-BASED FUZZY INFERENCE SYSTEM

The architecture and learning rules of adaptive networks have been described in the previous section. Functionally, there are almost no constraints on the node functions of an adaptive network except piecewise differentiability. Structurally, the only limitation of network configuration is that it should be of feedforward type. Due to these minimal restrictions, the adaptive network's applications are immediate and immense in various areas. 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.

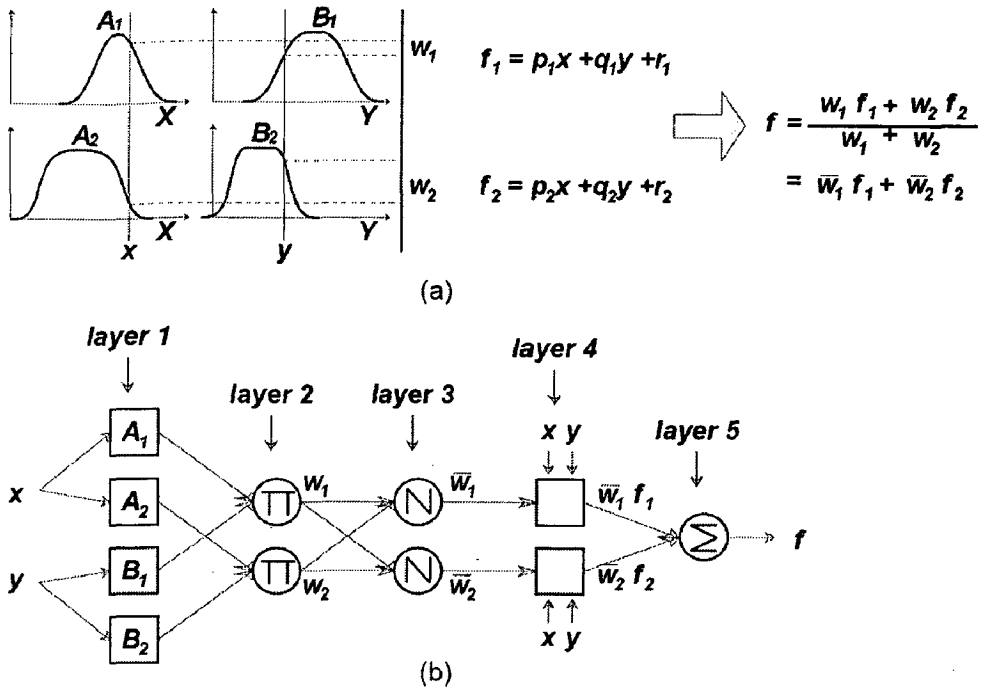


Fig 4.6 (a)fuzzy reasoning (b) equivalent ANFIS.

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 $f_1 = p_1 x + q_1 y + r_1$, Rule 2: If a; is A_2 and y is B_2, then $j_2 = p_2 x + q_2 y + r_2$

Then the fuzzy reasoning is illustrated in Figure 4.4(a), and the corresponding equivalent ANFIS architecture is shown in Figure 4.4(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

$$O_i^1 = \mu_{A_i}(x), \tag{4.17}$$

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 $\mu_{A_i}(x)$ t° be bell-shaped with maximum equal to 1 and minimum equal to 0, such as

$$\mu_{A_i}(x) = \frac{1}{1+\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}}, \tag{4.18}$$

Or

$$\mu_{A_i}(x) = \exp\left\{-\left[\left(\frac{x-c_i}{a_i}\right)^2\right]^{b_i}\right\}, \tag{4.19}$$

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 Ai. 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 II which multiplies the incoming signals and sends the product out. For instance

$$w_i = \mu_{A_i}(x) \times \mu_{B_i}(y), \quad \text{i=1,2} \tag{4.20}$$

Each node output represents the firing strength of a rule. (In fact, other T-norm operators that 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

$$\overline{w}_i = \frac{w_i}{w_1 + w_2}, \tag{4.21}$$

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

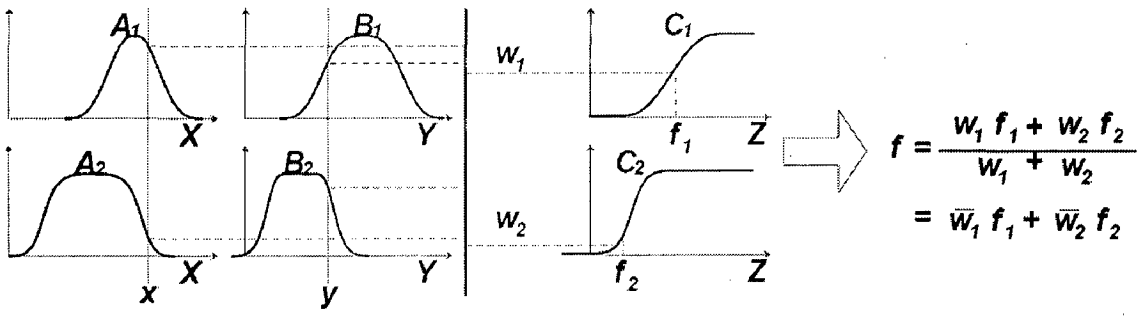
Layer 4 Every node i in this layer is a square node with a node function

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i \left(p_i x + q_i y + r_i\right), \tag{4.22}$$

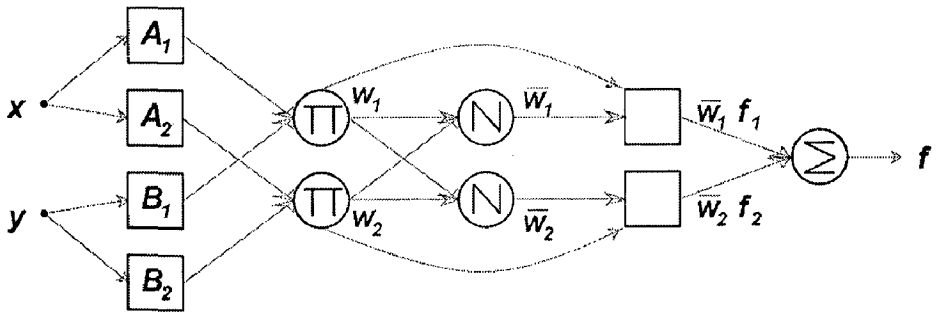
where $\overline{w}_i$ is the output of layer 3, and $\{p_i, q_i, r_i\}$ is the parameter set. Parameters in this layer will be referred to as consequent parameters.

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

$$O_1^5 = \text{overall output} = \sum_i \overline{w}_i f_i = \frac{\sum_i \overline{w}_i f_i}{\sum_i w_i} \tag{4.23}$$



(a)



(b)

Figure4. 7: (a) fuzzy reasoning;
(b) equivalent ANFIS.

Thus we have constructed an adaptive network which is functionally equivalent to a fuzzy inference system shown in fig.3.8. For fuzzy inference systems shown in fig4.7, the extension is quite straightforward and the ANFIS is shown in Figure

4.8 where the output of each rule is induced jointly by the output membership funcion and the firing strength. For fuzzy inference systems shown in fig.4.7, 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 4.8 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 (Figure4.8), 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 8 can be rewritten as

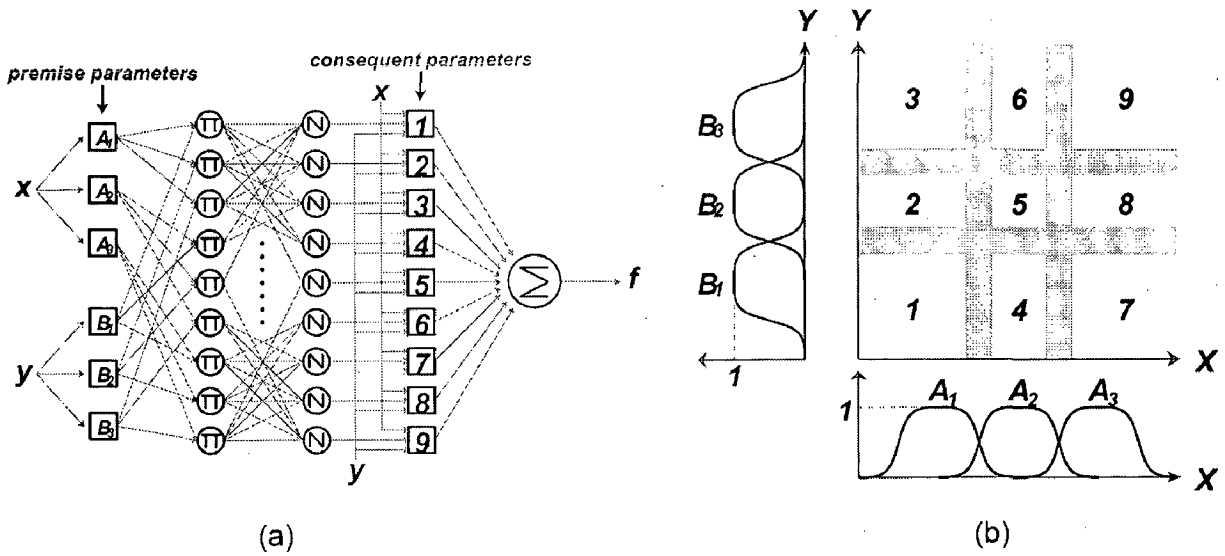


Figure 4.8: (a) 2-input ANFIS with 9 rules; (b) corresponding fuzzy subspaces

$$
\begin{aligned}
f &= \frac{w_1}{w_1 + w_2} f_1 + \frac{w_1}{w_1 + w_2} f_2 \\
&= \overline{w}_1 f_1 + \overline{w}_2 f_2 \\
&= (\overline{w}_1 x) p_1 + (\overline{w}_1 y) q_1 + (\overline{w}_1) r_1 + (\overline{w}_2 x) p_2 + (\overline{w}_2 y) q_2 + (\overline{w}_2) r_2,
\end{aligned}
\tag{4.24}
$$

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

in equation (4.10); $H(\bullet)$ and $F(\bullet, \bullet)$ are the identity function and the function of the fuzzy inference system, respectively. Therefore the hybrid learning algorithm developed in the previous chapter 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[14]

5.1. 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 PD, Fuzzy and Neuro-Fuzzy. In this chapter design and development of simulink model for robot manipulator, Actuator, PD controller, Fuzzy controller and Neuro-Fuzzy controller is explained.

5.2 PUMA560 ROBOT MODEL

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

$$\Rightarrow$$

$$\ddot{q} = M(q)^{-1}\left[\tau - C(q,\dot{q})\dot{q} - g(q)\right] \qquad (5.1)$$

By equation (5.1) we can develop,the simulink model

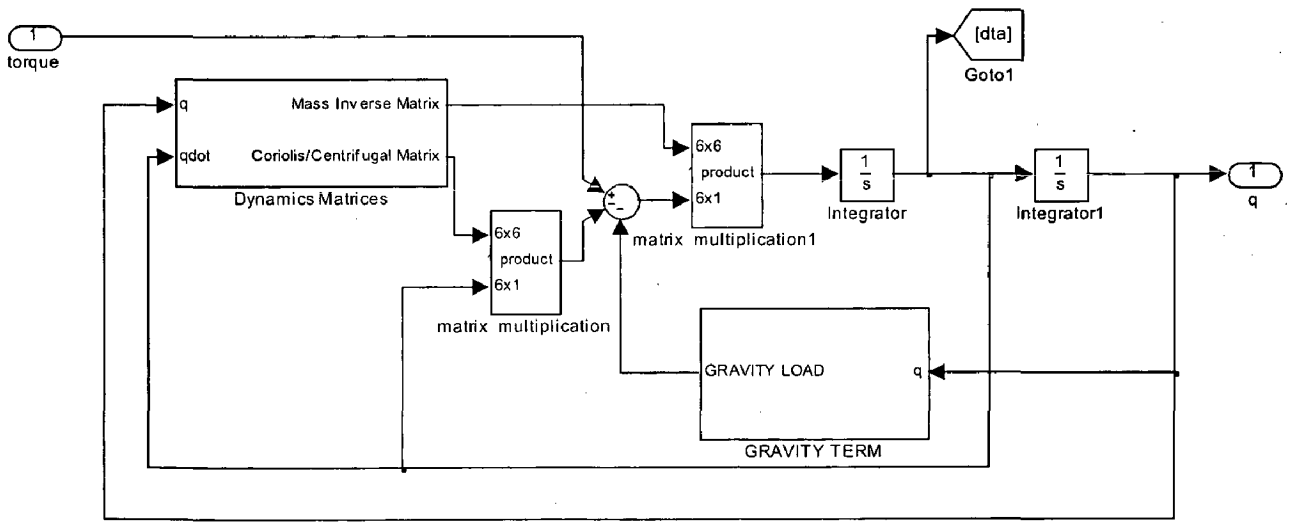


Fig .5.1 PUMA560 simulink model

Fig 5.1 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

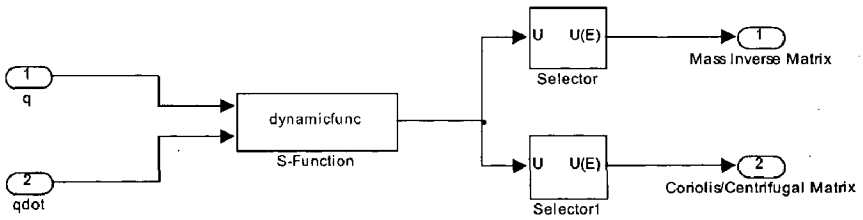


Fig 5.2 dynamics matrices

This block have the two input matrices angles at respective joints and derivatives of the angles of the order 6 byl and the two out puts mass inverse matrix and coriolis/centrifugal matrix of the order 36 byl (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. Code for S-Function is given in the appendix1. 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

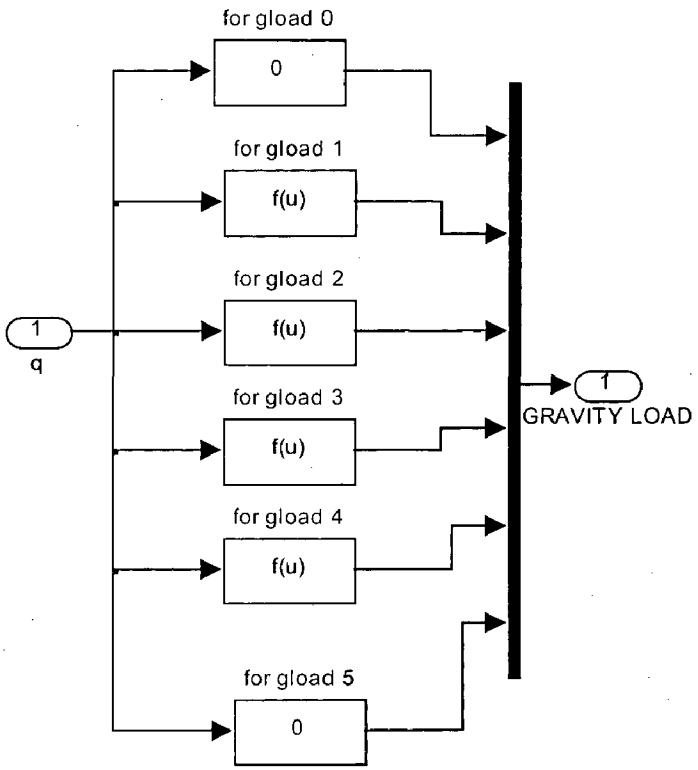


Fig 5.3 gravity load

Figure 5.3 shows the gravity load for the manipulator. Expressions for the gravity load are explained and given in chapter 4 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

MATRIX MULTIPLICATION

This matrix multiplication block shown in fig 5.4 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

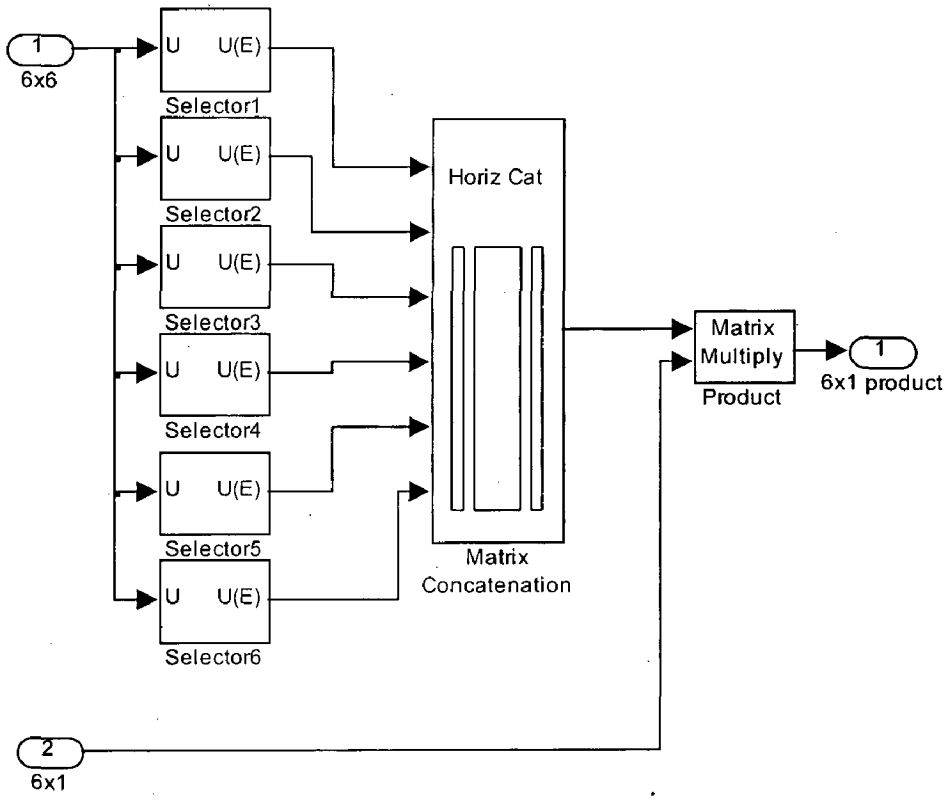


Fig5. 4 Matrix multiplication

By this discussion on design of PUMA560 model is complete

5.3 ACTUATOR MODEL

Fig 5.5 shows the simulink model for the actuator. It is designed as per the equations and specifications given in the chapter4

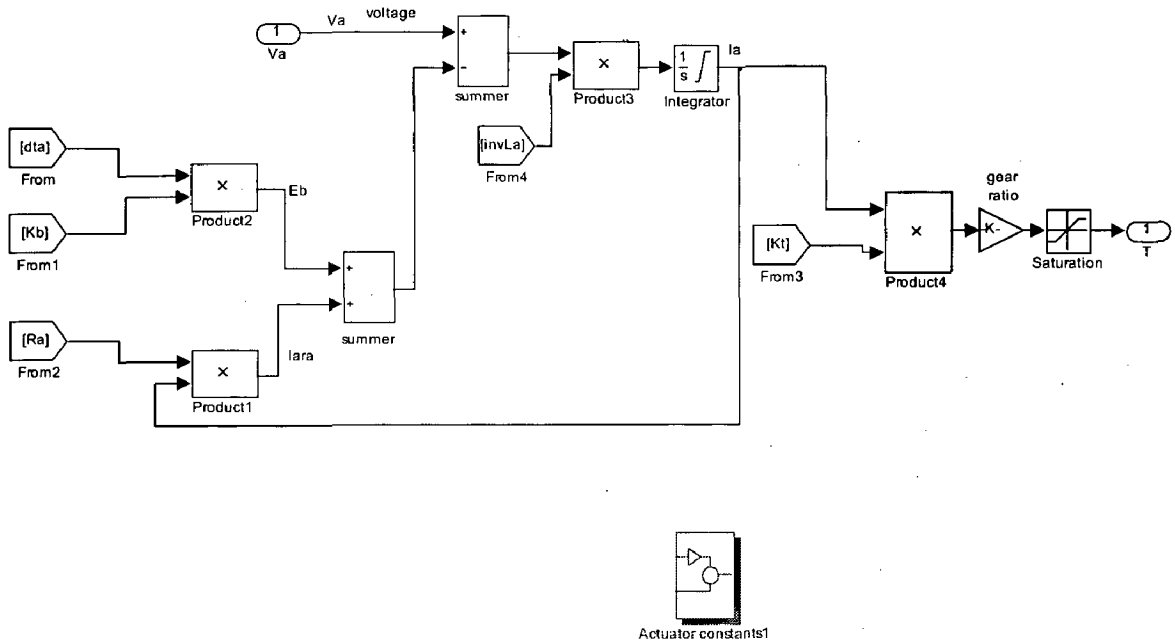


Fig. 5.5 Actuator simulink model

5.4DESIGN OF PID CONTROLLER

Equation for the PD controller is

$$U_i = P_i e + D_i \frac{de}{dt} + I_i \int e \qquad \text{(i=1,2...6)} \qquad (5.2)$$

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

By doing this gains are found to be

P_i=[40 20 13.333 10 8 6.6777]

D_i=[10 6 3.5 2.5 2.3 2.1]

I_i =[10 6 3.5 2.5 2.3 2.1]

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

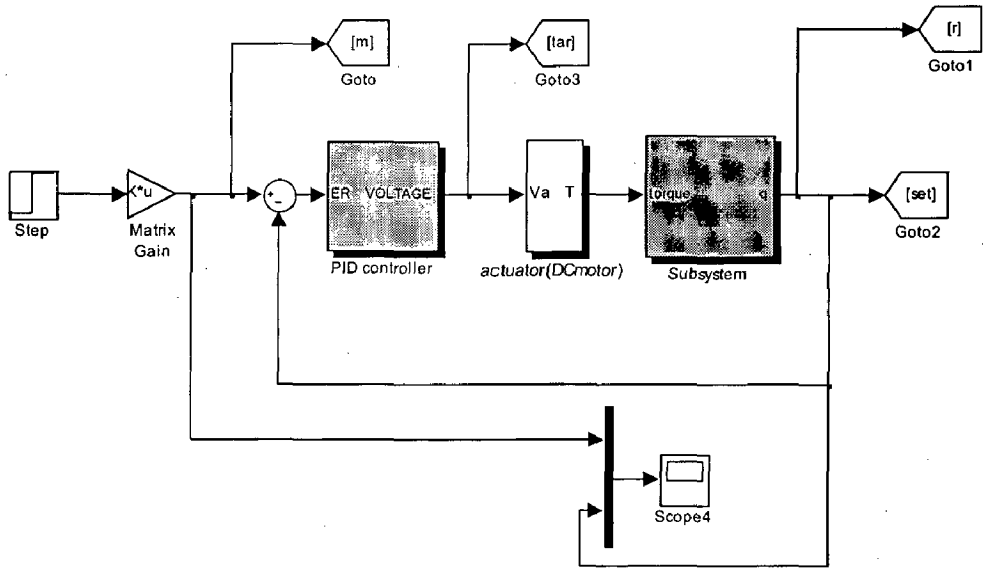


Fig 5.6 Total system with PID controller

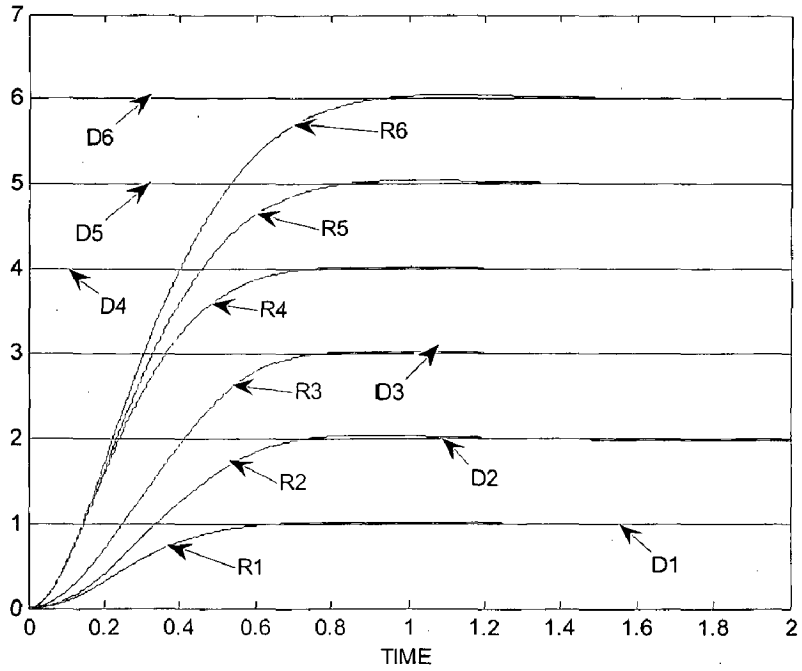


Figure 5.7 (a) Response of the robot (point to point control)

'Di 'corresponds to desired trajectory where 'i' correspond to joint no

'Ri 'corresponds to response where 'i' correspond to joint no

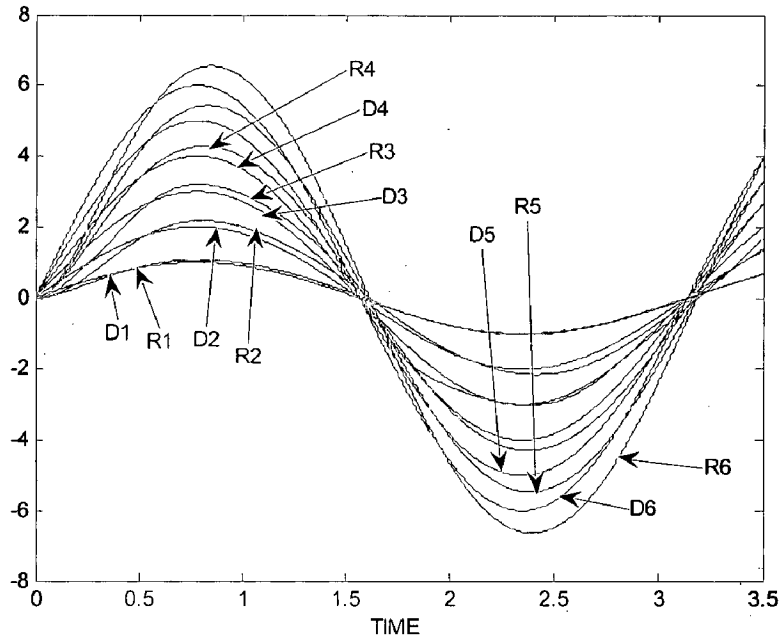


Figure 5.7.(b) Response of the robot (trajectory control)

'Di 'corresponds to desired trajectory where 'i' correspond to joint no

'Ri 'corresponds to response where 'i' correspond to joint no

5.5 DESIGN OF FUZZYPD+I CONTROLLER

Designing the Fuzzy controller in simulink consists of two steps

1. Designing the rule base
2. gain scheduling

TABLE 5.1 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 in the[3] these rules generally used for the Fuzzy PD controller

Gain scheduling

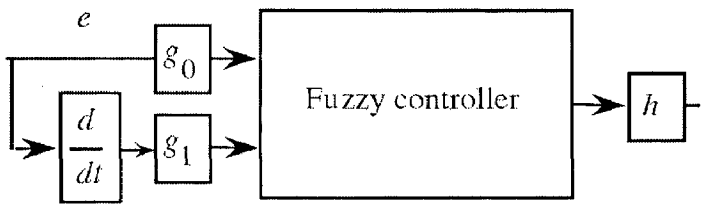


Fig5. 8 Fuzzy controller with gains

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 the there will not be any rule to processes 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 gives the maximum output ,that will be the maximum controller output
3. then increase g_0 until overshoots under the allowable range

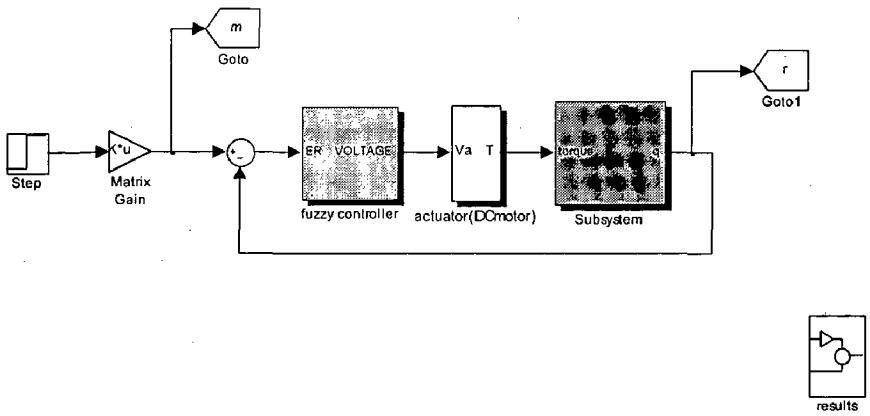


Fig 5.9 System with fuzzy controller

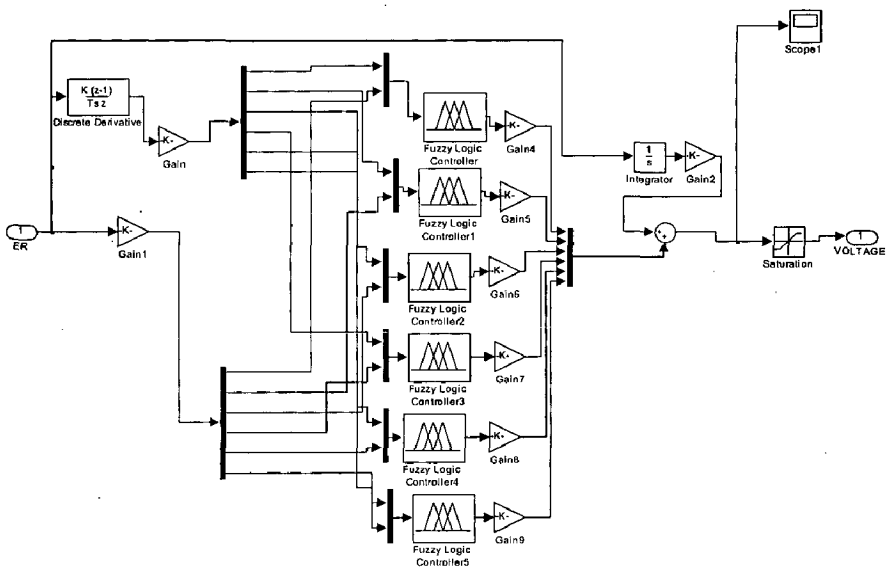


Fig 5.10 Fuzzy controller

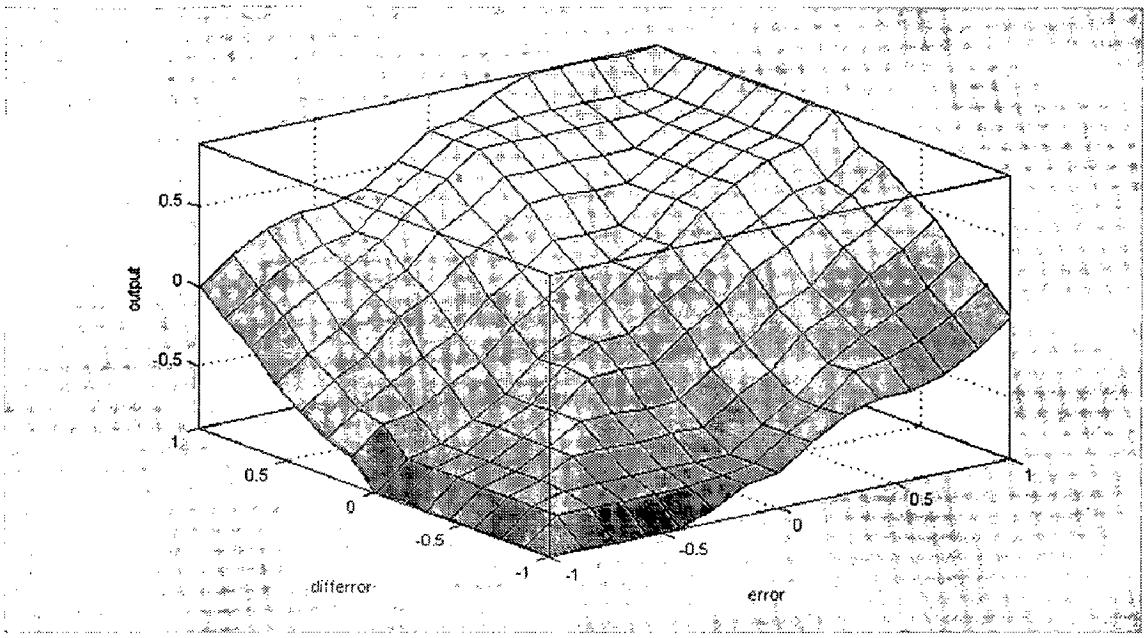


Fig5.11 nonlinear output for the Fuzzy PD controller

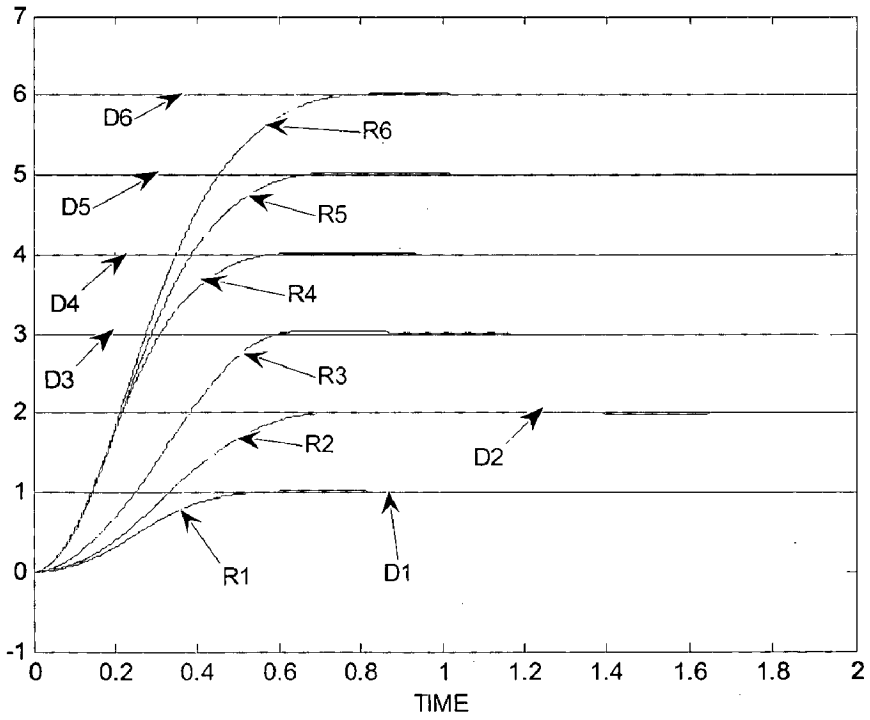


Fig 5.12. (a) Robot response with the FuzzyPD+I controller (point to point)

'Di 'corresponds to desired trajectory where 'i' correspond to joint no

'Ri 'corresponds to response where 'i' correspond to joint no

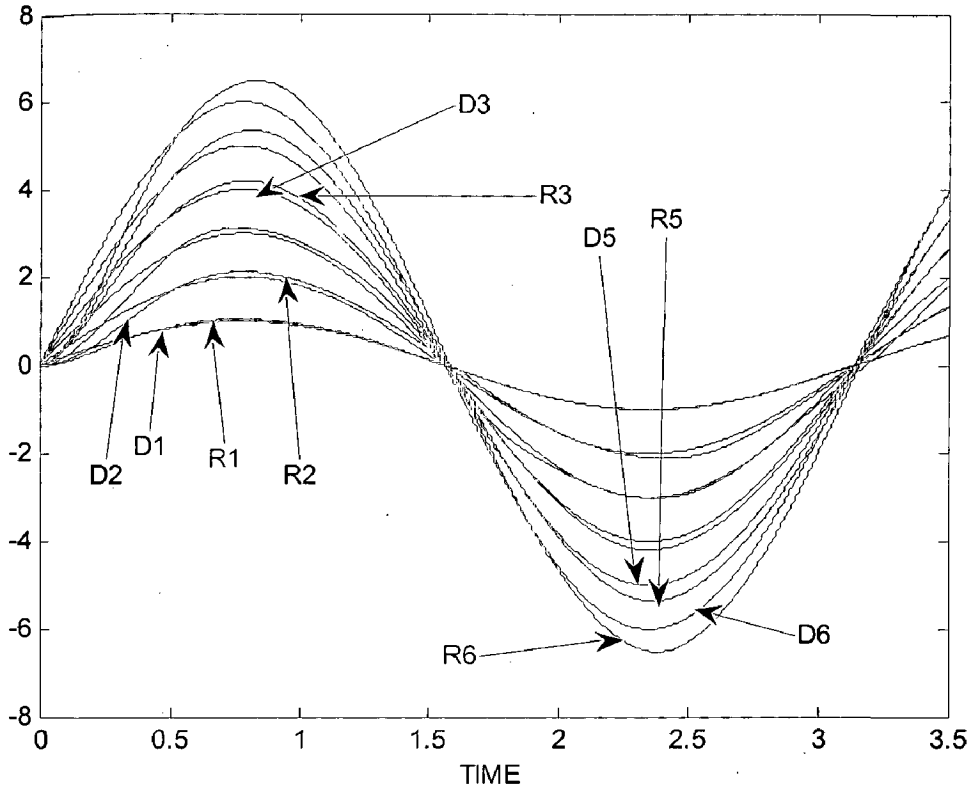


Fig 5.12.(b) Robot response with the FuzzyPD+I controller (trajectory)

'Di 'corresponds to desired trajectory where 'i' correspond to joint no

'Ri 'corresponds to response where 'i' correspond to joint no

5.6 DESIGN OF NEURO-FUZZY CONTROLLER

This Neuro-Fuzzy controller can be designed in simulink using the fuzzy logic toolbox. About ANFIS already explained in chapter4 .In matlab design of ANFIS consists of Training

Model Learning and Inference through ANFIS

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. 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. This is explained in[19].

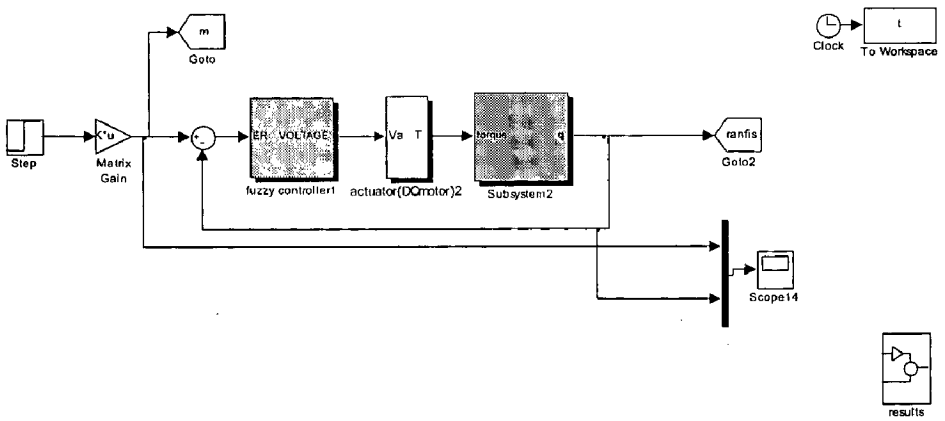


Fig.5.13 Total robot system with Anifs controller

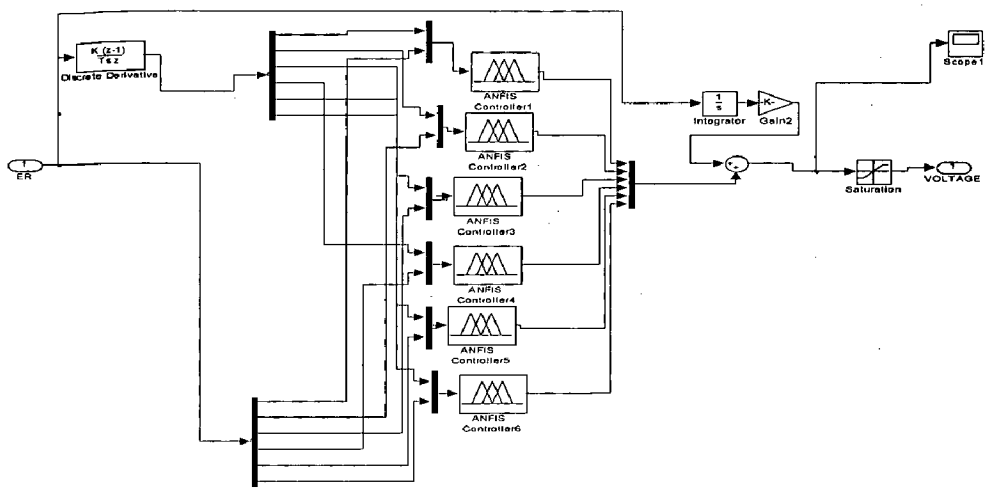


Fig.5.14 ANFIS controller

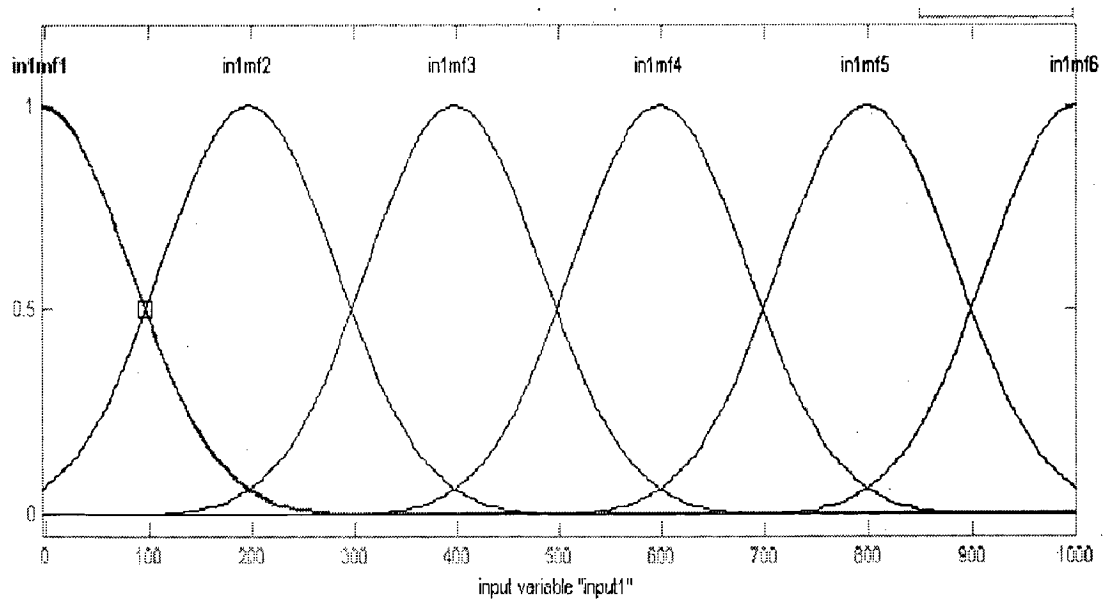


Fig 15 membership functions after training

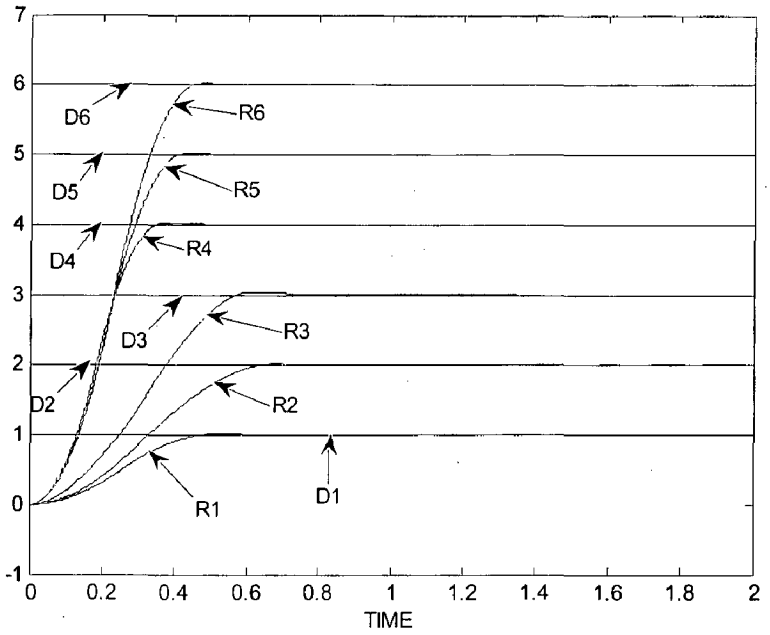


Fig.5.16 (a) system response with the Anfis controller (point to point)
'Di 'corresponds to desired trajectory where 'i' correspond to joint no
'Ri 'corresponds to response where 'i' correspond to joint no

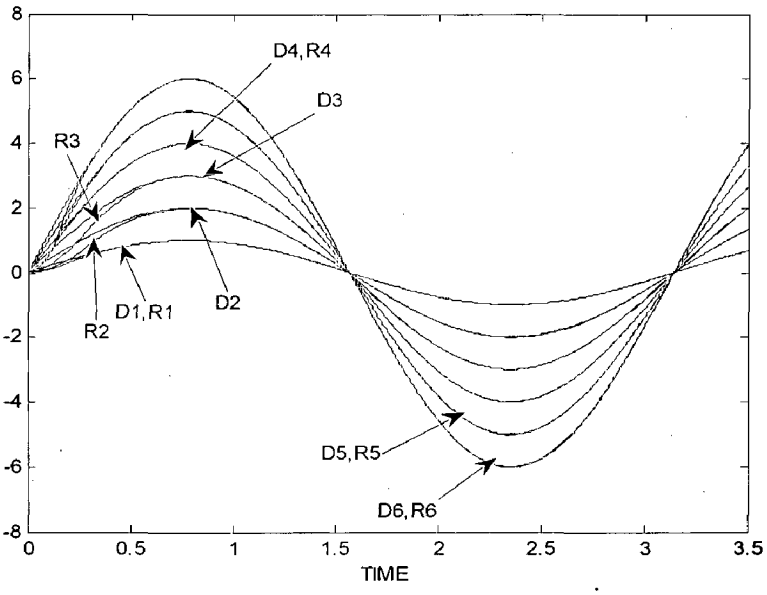


Fig.5.16 (b) System response with the Anfis controller (trajectory)
'Di 'corresponds to desired trajectory where 'i' correspond to joint no
'Ri 'corresponds to response where 'i' correspond to joint no

6.1 RESPONSES OF THE ROBOT FOR POINT TO POINT CONTROL

There are six joints to be controlled in the ROBOT. 1, 2, 3,4,5,6 radians are taken as the desired point at each join respectively, the responses at each joint are shown in the figures 6.1-6.6 below.

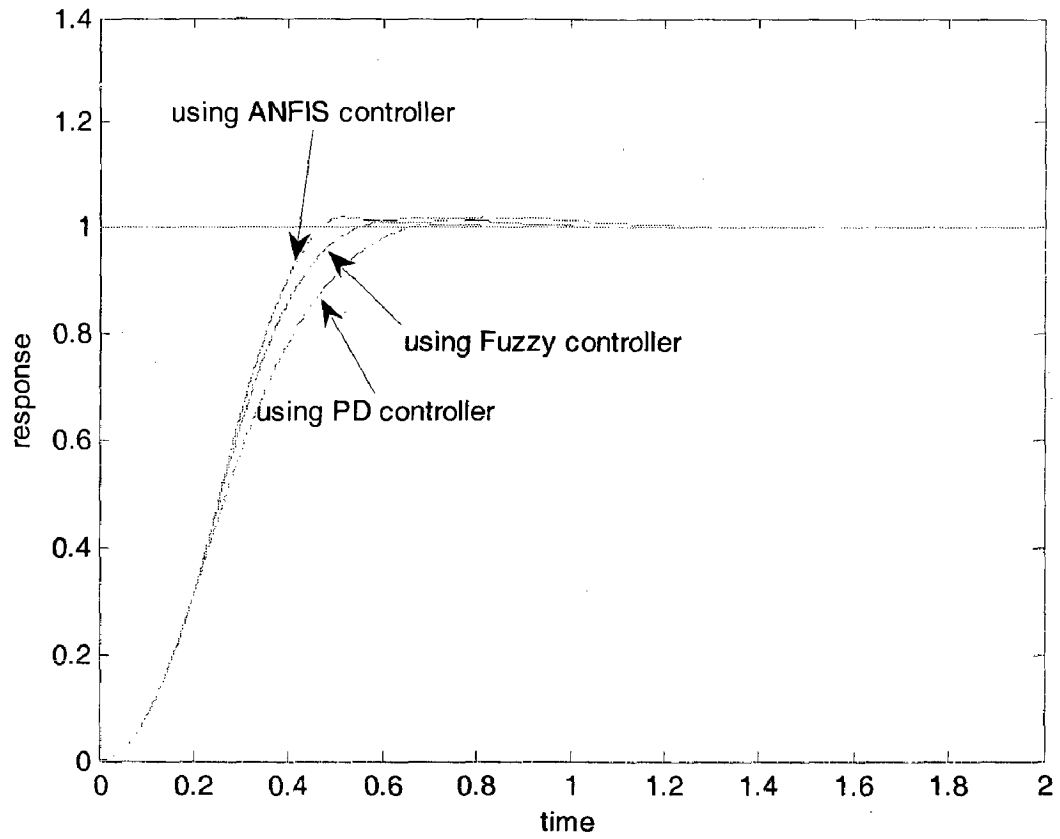


Fig.6.1 Response at joint1 of the Robot for point to point control

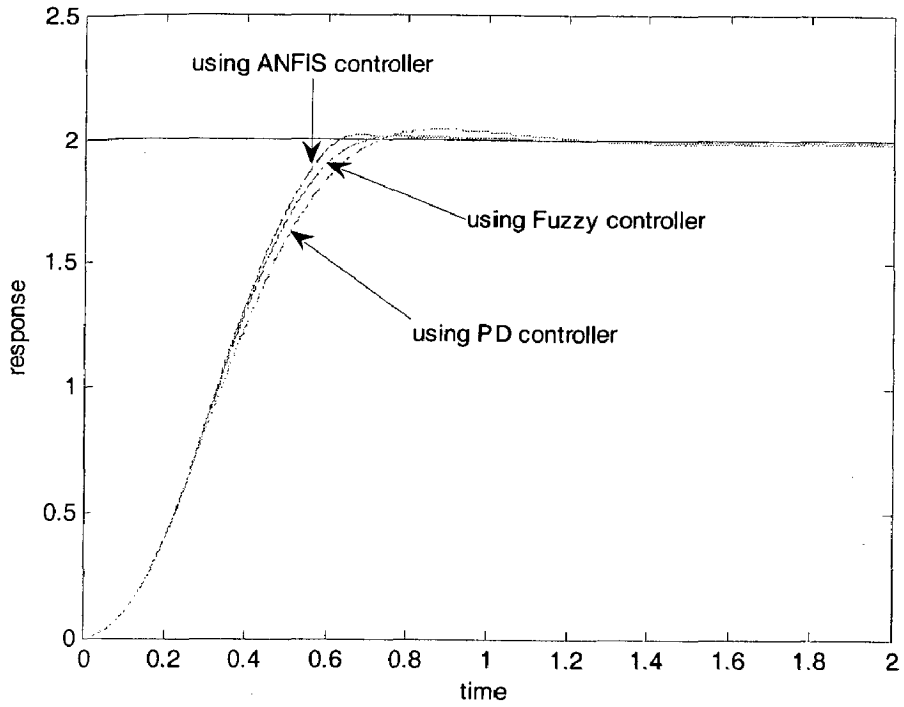


Fig.6.2 Response at joint2 of the Robot for point to point control

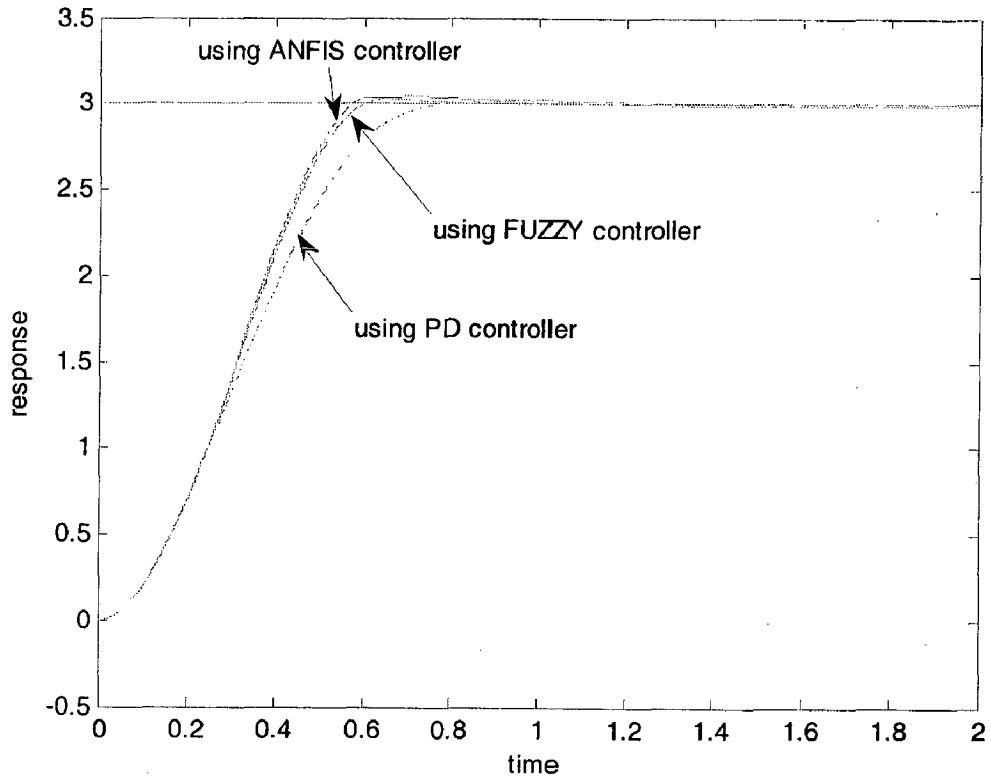


Fig.6.3 Response at joint3 of the Robot for point to point control

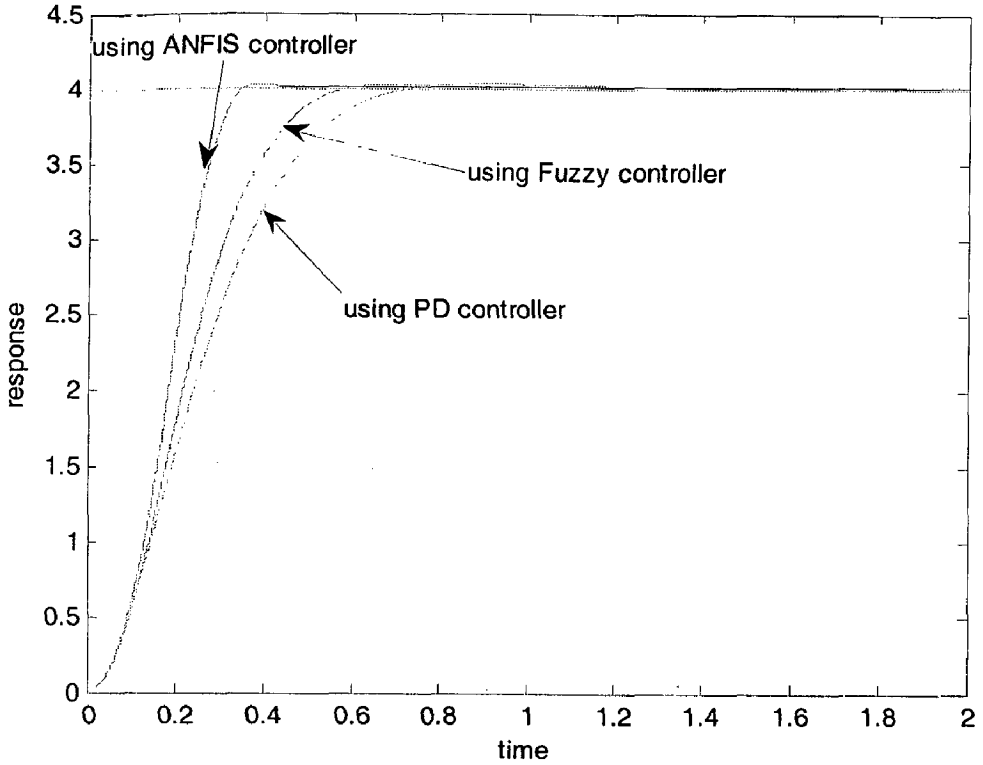


Fig.6.4 Response at joint4 of the Robot for point to point control

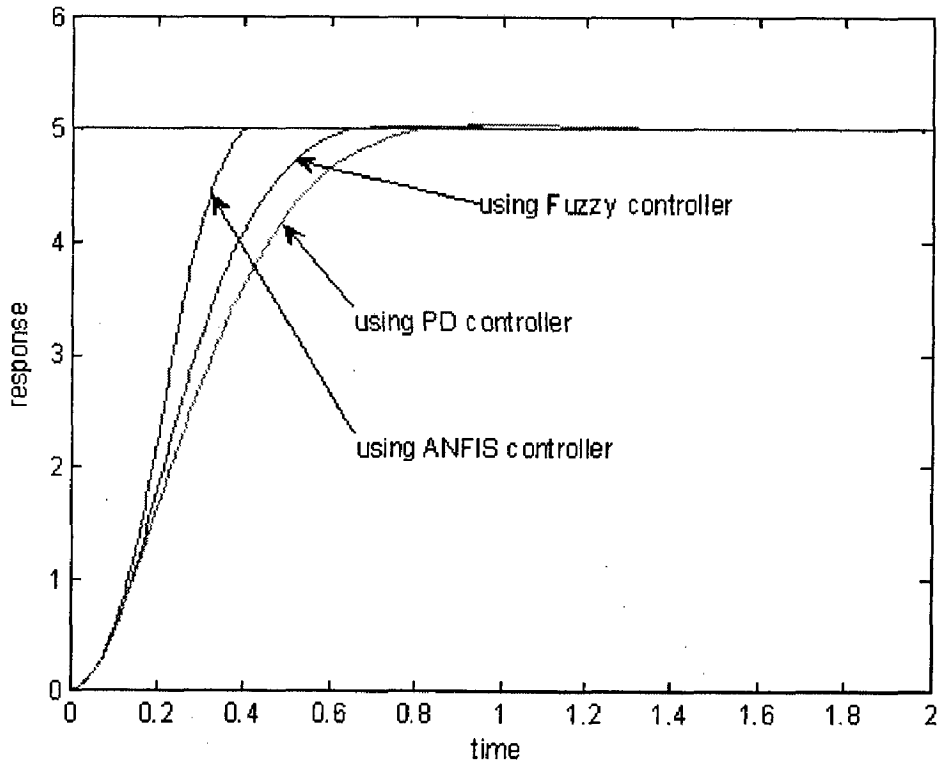


Fig.6.5 Response at joint5 of the Robot for point to point control

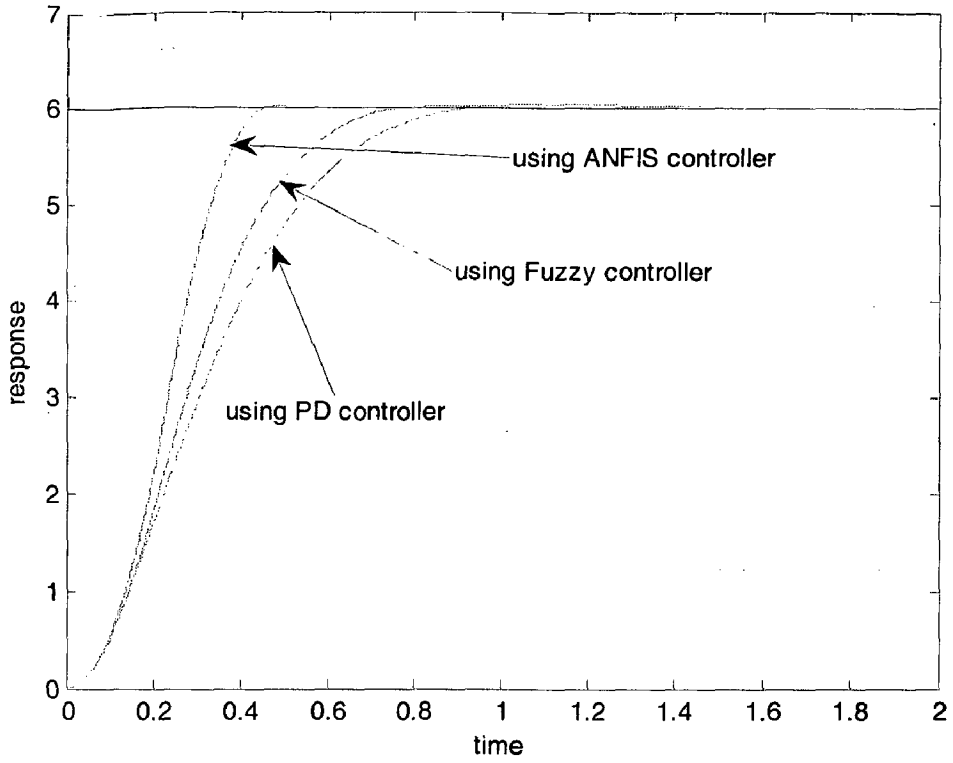


Fig.6.6 Response at joint6 of the Robot for point to point control

All the responses of the robot (all joints) for the point to point control with all the controllers are shown comparatively in the above figures, from these figures we can clearly see that the response using PD controller is normal, with Fuzzy controller response is better than the PD because of nonlinear controller output with expert knowledge base it gives the better response, and with the ANFIS controller response is better than both Fuzzy and the PD controllers,.

6.2 RESPONSES 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 2Hz, maximum values of the desired trajectories are 1,2,3,4,5,6 radians respectively, the responses at each joint are shown in the figures from 6.7-6.12.

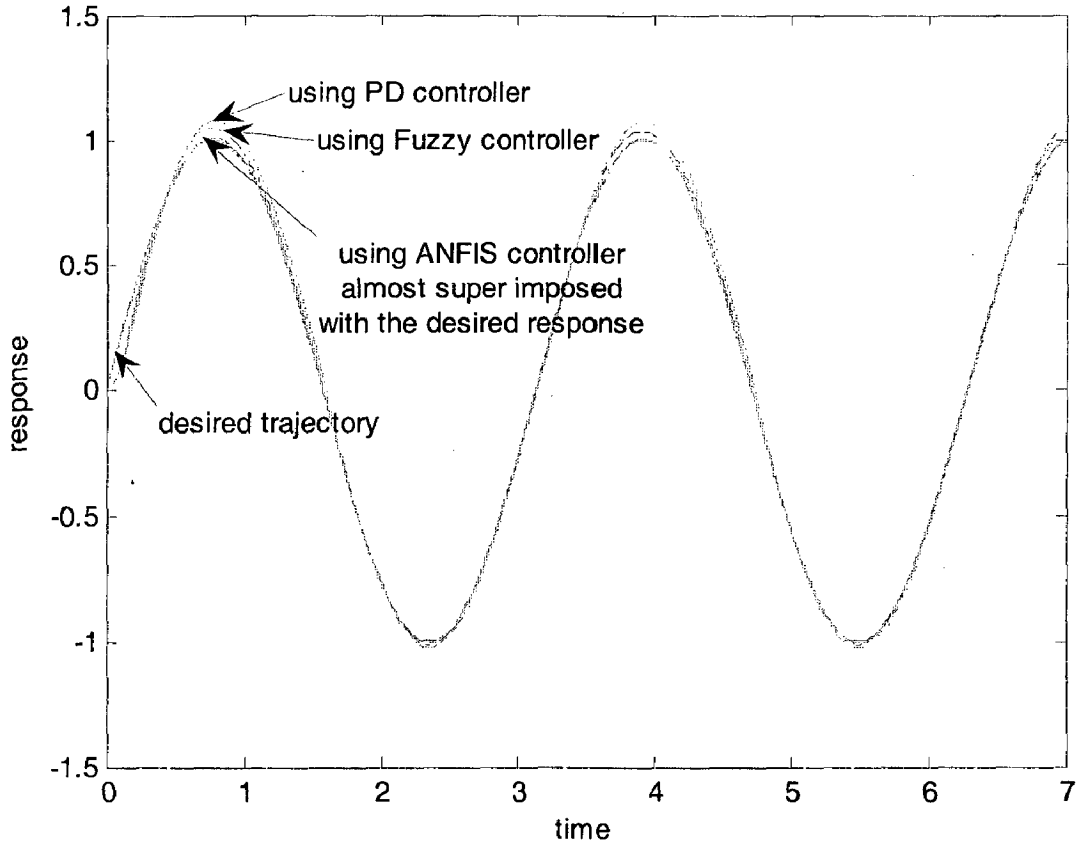


Fig.6.7 Response at joint1 of the Robot for trajectory control

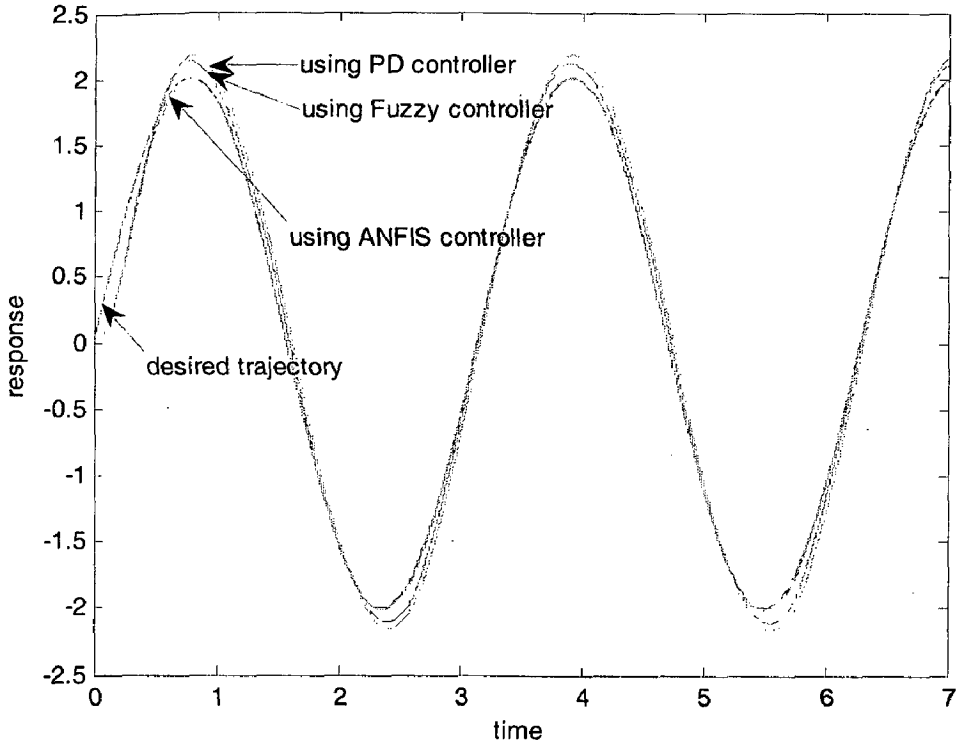


Fig.6.8 Response at joint2 of the Robot for trajectory control

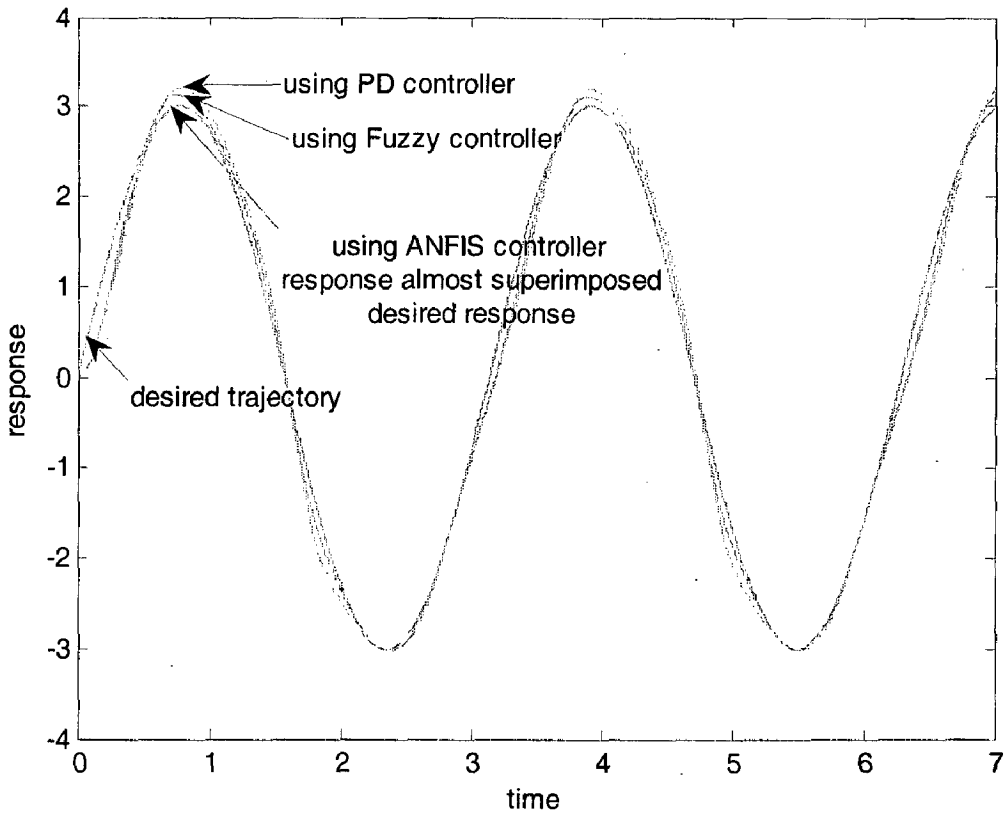


Fig.6.9 Response at joint3 of the Robot for trajectory control

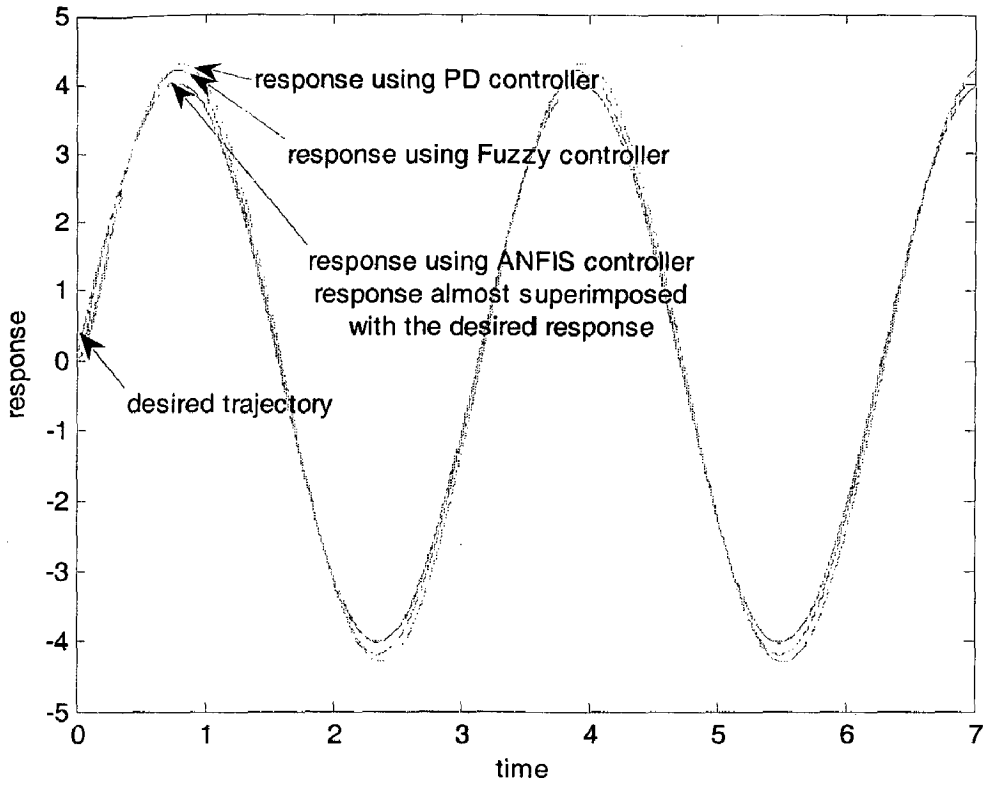


Fig.6.10 Response at joint4 of the Robot for trajectory control

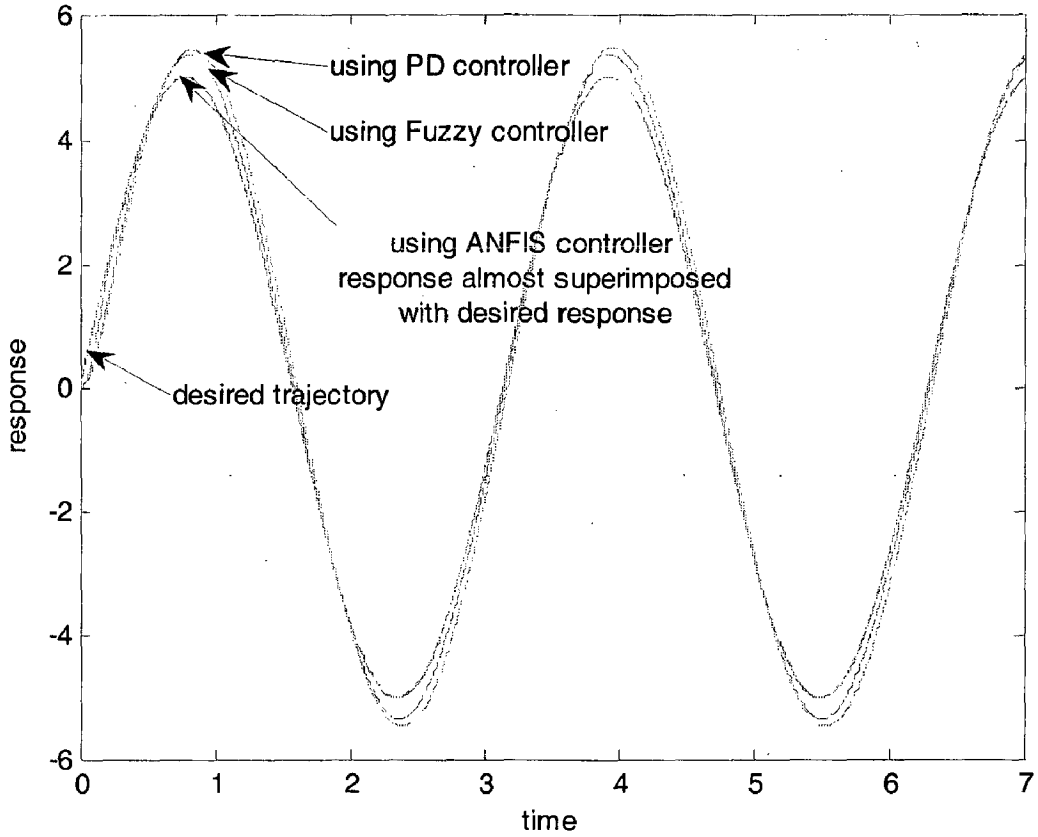


Fig.6.11 Response at joint5 of the Robot for trajectory control

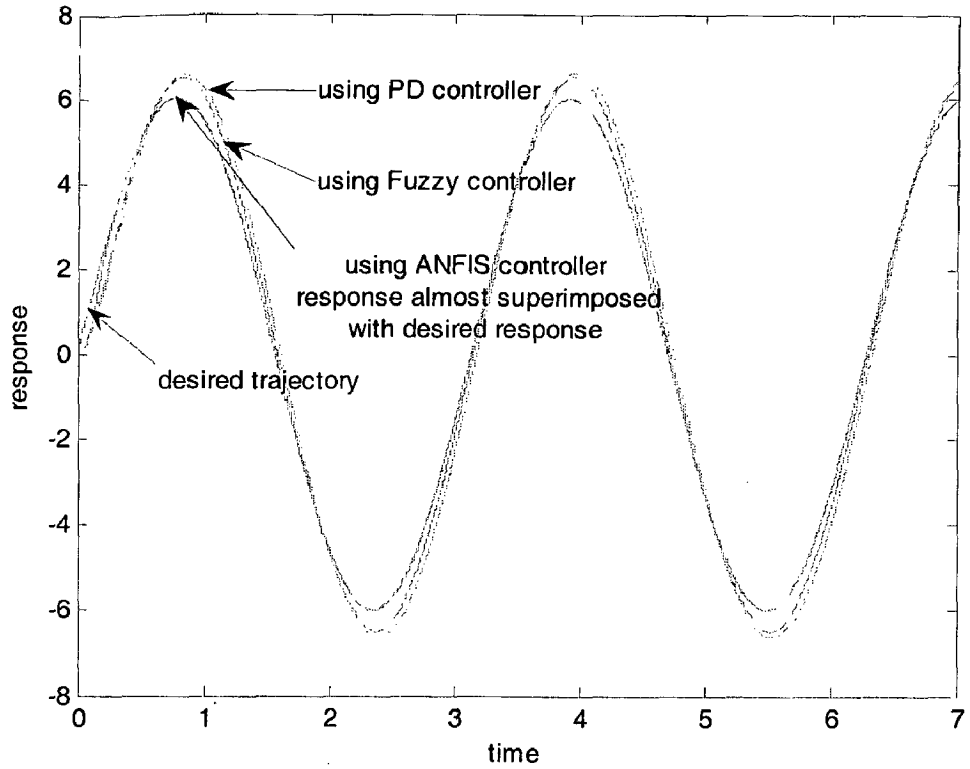


Fig.6.12 Response at joint6 of the Robot for trajectory control

All the responses of the robot (all joints) for the trajectory control with all the controllers are shown comparatively in the above figures, from these figures we can clearly see that the response using PD controller is normal, with Fuzzy controller response is better than the PD because of nonlinear controller output with expert knowledge base it gives the better response, and with the ANFIS controller response is better than both Fuzzy and the PD controllers

CONCLUSIONS

The study compares the conventional PID control scheme, Fuzzy control scheme and Neuro-Fuzzy control scheme. From the results of the first part of the work it has been found that conventional control scheme using PID controller is simplest to be implemented but it cannot cope with uncertainties in the robot dynamics.

From the second part of the work incase of Fuzzy based controller has good tracking performance only difficulties are designing with the rulebase.Generalized rulebase is taken for the fuzzy controller

Third part of the work in which ANFIS (adaptive neuro fuzzy inference system) is designed it has been found good tracking performances. 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.

FUTURE SCOPE OF WORK

The Proposed ANFIS structure uses Temporal back propagation hybrid algorithm. The convergence time depends on the number of input membership functions, if the number of input membership functions increases then the learning process becomes slow and if the no of member ship 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] .The convergence time using Genetic Algorithm is far better the Temporal Back Propagation Algorithm or any other conventional algorithms All these well explained in [21].

REFERENCES

1.'Introduction to robotics Mechanics and Control' by John J.Craig, second edition Addison Wesley publications.

2.'Robot Dynamics and Control' by Mark W.Spong, M.Vidyasagar, John Wiley & Sons publications.

3.'Fuzzy Control'Kevin M. Passino, Stephen Yurkovich, Addison Wesley Longman publications.

4. 'The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm' Brian Armstrong, Oussama Khatib, Joel Burdick, Stanford Artificial Intelligence Laboratory Stanford University, IEEE Transactions and Systems 1986.

5.'The Unimation Puma servo system' Peter I.corke CSIRO Division of Manufacturing Technology, AUSTRALIA July 1994.

6. 'A Search for Consensus Among Model Parameters Reported for the PUMA 560 Robot' Peter I. Corke ,CSIRO Division of Manufacturing Technology, AUSTRALIA. Brain Armstrong-Helouvry University of Wisconsin Milwaukee ,USA. July 1994

7.L.A.Zadeh,'Fuzzy sets' informat contr Vol.-8,pp.338-353,1965.

8.S.Assilian,E.H.Mamdani,'An Experiment With Linguistic Synthesis With Fuzzy Logic Controller' Int Journal on Man Machine studies,Vol.7,pp.1-13,1975.

9. M.maeda ,S.Murakami,'A self Tuning Fuzzy Controller' Fuzzy Sets and Sytems,Vol.-56,pp.53-65,1979.

10.S.Z.He,S.Tan,' Fuzzy Self Learning of PID Controllers',Fuzzy Sets Systems, Vol.-56,pp.37-46,1993

11.M.Yoshida,Y.Ishida,'Gain Tuning Method for Designing of Fuzzy Control System',IEEE Trans.Fuzzy Systems,Vol.52,pp.405-408,1990.

12. H.X.Li , H.B.Gutland,'Conventional Fuzzy Control and its enhancement', IEEE Trans,Syst.Man,Cyber.,Vol.-26,pp791-797,1996

13.H.R.Berenji,'Learning and Tuning Fuzzy Logic Controllers Through Reinforcements', IEEE Transactions on Neural Networks,Vol3,1992,pp724-740

14.J.R.Jang 'Self-Learning Fuzzy Controllers Based on the Temporal Back Propagation' IEEE transactions on Neural Networks,Vol.3, 1992,pp714-721

15.Moens Iskarous and Kazuhiko Kawamura 'Intelligent Control Using a Neuro-Fuzzy Network' IEEE transactions on Neural Networks ,Vol.4,1995,pp350-355

16. Gurupreet S. Sandhu and Kuldio S.rahan,'Design of Neuro Fuzzy Controller' IEEE transactions on Neural Networks, Vol.1,1997,pp3170-3175

17.Wen Yu,Xiaoou Li,'Fuzzy Neuro Modeling Using Stable Learning Algorithm' IEEE transactions on Neural Networks,Vol.2,2003,pp4542-4548

18. Oscar Castillo, Patrica Melin'Inteligent adaptive model-based of robotic dynamic systems with a hybrid fuzzy-neural approach'ELSEVIER transactions on Applied Soft Computing,Vol.3,2003,pp367-378

19. 'Fuzzy Logic Toolbox User's Guide' ,Version 2, www.mathworks.com

20.' Writing S-Functions User's Guide', Version 5, www.mathworks.com

21.Ashok Kumar Goel, Suresh Chandra Saxena, Surekha Bhanot, 'A Genetic Based Neuro-Fuzzy Controller for Tharmal Processes'.JCS&T,Vol.5,No.1 2005,37-43

1.Main file for S-Function

```
 * File : dynamicfunc
 * Abstract:
 *      All this thing does is calculate the Coriolis/centrifugal torque matrix
 *      and the kinetic energy (mass) matrix based on q and qdot inputs

 #define S_FUNCTION_NAME  dynamicfunc
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"
#include "dynamicfunc.h"

/* Function: mdlInitializeSizes
=============================================
 * Abstract:
 *   Setup sizes of the various vectors.
 */
static void mdlInitializeSizes(SimStruct *S)
{
   ssSetNumSFcnParams(S, 0);
   if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S))
         {
      return; /* Parameter mismatch will be reported by Simulink */
   }

   if (!ssSetNumInputPorts(S, 2)) return;
   ssSetInputPortWidth(S, 0, 6);
         ssSetInputPortWidth(S, 1, 6);

   ssSetInputPortDirectFeedThrough(S, 0, 1);
   ssSetInputPortDirectFeedThrough(S, 1, 1);

   if (!ssSetNumOutputPorts(S, 1)) return;
   ssSetOutputPortWidth(S, 0, 72);

   ssSetNumSampleTimes(S, 1);

   /* Take care when specifying exception free code - see sfuntmpl.doc */
   ssSetOptions(S, SS_OPTION_EXCEPTION_FREE_CODE);

}
```

```
/* Function: mdlInitializeSampleTimes
=============================================
 * Abstract:
 *    Specifiy that we inherit our sample time from the driving block.
 */
static void mdlInitializeSampleTimes(SimStruct *S)
{
   ssSetSampleTime(S, 0, INHERITED_SAMPLE_TIME);
   ssSetOffsetTime(S, 0, 0.0);


}


/* Function: mdlOutputs
=======================================================
 */
static void mdlOutputs(SimStruct *S, int_T tid)
{
        InputRealPtrsType uPtrs0 = ssGetInputPortRealSignalPtrs(S,0);
        InputRealPtrsType uPtrs1 = ssGetInputPortRealSignalPtrs(S,1);

        real_T          *y0   = ssGetOutputPortRealSignal(S,0);
        int_T                           i;
        real_T                          q[6];
        real_T                          qdot[6];
        real_T                          massinv[6][6];
        real_T                          coricen[6][6];


        for(i=0;i<6;i++)
        {
                q[i] = *uPtrs0[i];
                qdot[i] = *uPtrs1[i];
        }


        calculate_mass(q);
        calculate_massinv(massinv);
        calculate_coricen(qdot, coricen);


        for(i=0;i<6;i++)
        {
                y0[i] = massinv[0][i];
                y0[i+6] = massinv[1][i];
```

```
            y0[i+12] = massinv[2][i];
            y0[i+18] = massinv[3][i];
            y0[i+24] = massinv[4][i];
            y0[i+30] = massinv[5][i];
            y0[i+36] = coricen[0][i];
            y0[i+42] = coricen[1][i];
            y0[i+48] = coricen[2][i];
            y0[i+54] = coricen[3][i];
            y0[i+60] = coricen[4][i];
            y0[i+66] = coricen[5][i];

        }

}


/* Function: mdlTerminate
=============================================================
 * Abstract:
 *    No termination needed, but we are required to have this routine.
 */
static void mdlTerminate(SimStruct *S)
{
}


#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

---

2. Header file for the main file

```
//Header for dynamicfunc.c

#include <math.h>

real_T c1,c2,c3,c4,c5,c6,c23,s1,s2,s3,s4,s5,s6,s23;

real_T m11,m12,m13,m14,m15,m16,m21,m22,m23,m24,m25,m26,
            m31,m32,m33,m34,m35,m36,m41,m42,m43,m44,m45,m46,m51,
            m52,m53,m54,m55,m56,m61,m62,m63,m64,m65,m66;

void sincos(real_T q[6])
{
```

```
c1 = cos(q[0]);
c2 = cos(q[1]);
c3 = cos(q[2]);
c4 = cos(q[3]);
c5 = cos(q[4]);
c6 = cos(q[5]);
s1 = sin(q[0]);
s2 = sin(q[1]);
s3 = sin(q[2]);
s4 = sin(q[3]);
s5 = sin(q[4]);
s6 = sin(q[5]);
s23 = sin(q[1]+q[2]);
c23 = cos(q[1]+q[2]);

}


//Function to compute the kinetic energy (mass) matrix
void calculate_mass(real_T q[6])
{
        sincos(q);

        m11 = 2.57 + (1.38*c2*c2) + (0.3*s23*s23) + (0.744*c2*s23);
        m12 = (0.69*s2) + (-0.134*c23) + (0.0238*c2);
        m13 = (-0.134*c23) + (-0.00397*s23);
        m14 = 0;
        m15 = 0;
        m16 = 0;
        m22 = 6.79 + (0.744*s3);
        m23 = 0.333 + (0.372*s3) + (-0.011*c3);
        m24 = 0;
        m25 = 0;
        m26 = 0;
        m33 = 1.16;
        m34 = -0.00125*s4*s5;
        m35 = 0.00125*c4*c5;
        m36 = 0;
        m44 = 0.2;
        m45 = 0;
        m46 = 0;
        m55 = 0.18;
        m56 = 0;
        m66 = 0.19;
```

```
        //The mass matrix is symmetric, so now we take care of elements below
        //the diagonal

        m21 = m12;
        m31 = m13;
        m41 = m14;
        m51 = m15;
        m61 = m16;
        m32 = m23;
        m42 = m24;
        m52 = m25;
        m62 = m26;
        m43 = m34;
        m53 = m35;
        m63 = m36;
        m54 = m45;
        m64 = m46;
        m65 = m56;

}


//Function to compute the kinetic energy (mass) matrix inverse
calculate_massinv(real_T mi[6][6])
{

        real_T factor1;

        factor1 = m21*m12*m34*m55*m43 - m21*m12*m33*m44*m55 +
m21*m12*m53*m44*m35 +
                                m21*m32*m13*m44*m55 + m31*m23*m44*m55*m12 -
m34*m22*m11*m55*m43 -
                                m53*m44*m22*m11*m35 + m33*m44*m22*m11*m55 -
m22*m31*m13*m44*m55 -
                                m11*m32*m23*m44*m55;

        mi[0][0] = (-m44*m22*m35*m53 + m44*m22*m33*m55 - m44*m23*m55*m32
- m22*m34*m55*m43)/factor1;
        mi[0][1] = (m12*m34*m55*m43-
m12*m33*m44*m55+m12*m53*m44*m35+m32*m13*m44*m55)/factor1;
        mi[0][2] = -(m55*m44*(m13*m22-m12*m23)/factor1);
        mi[0][3] = m34*m55*(m13*m22-m12*m23)/factor1;
        mi[0][4] = m35*m44*(m13*m22-m12*m23)/factor1;
        mi[0][5] = 0;
```

```
        mi[1][0] = -(-m21*m34*m55*m43+m21*m33*m44*m55-m21*m53*m44*m35-
m31*m23*m44*m55)/factor1;
        mi[1][1] = -(m44*m11*m35*m53-
m44*m11*m33*m55+m44*m13*m55*m31+m11*m34*m55*m43)/factor1;
        mi[1][2] = m55*m44*(m21*m13-m11*m23)/factor1;
        mi[1][3] = -m34*m55*(m21*m13-m11*m23)/factor1;
        mi[1][4] = -m35*m44*(m21*m13-m11*m23)/factor1;
        mi[1][5] = 0;

        mi[2][0] = m55*m44*(m32*m21-m31*m22)/factor1;
        mi[2][1] = m55*m44*(-m11*m32+m31*m12)/factor1;
        mi[2][2] = -m55*m44*(-m11*m22+m21*m12)/factor1;
        mi[2][3] = m34*m55*(-m11*m22+m21*m12)/factor1;
        mi[2][4] = m35*m44*(-m11*m22+m21*m12)/factor1;
        mi[2][5] = 0;

        mi[3][0] = -m43*m55*(m32*m21-m31*m22)/factor1;
        mi[3][1] = -m43*m55*(-m11*m32+m31*m12)/factor1;
        mi[3][2] = m43*m55*(-m11*m22+m21*m12)/factor1;
        mi[3][3] = (m21*m12*m35*m53-
m21*m12*m33*m55+m21*m32*m13*m55+m31*m23*m55*m12 -
                m53*m35*m22*m11+m33*m55*m22*m11-m22*m31*m13*m55-
m11*m32*m23*m55)/factor1;
        mi[3][4] = -m43*m35*(-m11*m22+m21*m12)/factor1;
        mi[3][5] = 0;

        mi[4][0] = -m53*m44*(m32*m21-m31*m22)/factor1;
        mi[4][1] = -m53*m44*(-m11*m32+m31*m12)/factor1;
        mi[4][2] = m53*m44*(-m11*m22+m21*m12)/factor1;
        mi[4][3] = -m34*m53*(-m11*m22+m21*m12)/factor1;
        mi[4][4] = (m21*m44*m32*m13-m21*m12*m33*m44+m21*m12*m43*m34-
m31*m22*m44*m13 -
                m11*m23*m44*m32+m44*m31*m23*m12+m44*m33*m22*m11-
m34*m43*m22*m11)/factor1;
        mi[4][5] = 0;

        mi[5][0] = 0;
        mi[5][1] = 0;
        mi[5][2] = 0;
        mi[5][3] = 0;
        mi[5][4] = 0;
        mi[5][5] = 1/m66;

}
```

```
//Function to compute the coriolis/centrifugal torques matrix
calculate_coricen(real_T qdot[6], real_T cor[6][6])
{

cor[0][0] = (-1.38*c1*s1*qdot[0]) +
            0.5*qdot[1]*(0.6*s23*c23 - 0.744*s2*s23 + 0.744*c2*c23) +
            0.5*qdot[2]*(0.6*s23*c23 - 0.744*s2*s23 + 0.744*c2*c23);

cor[0][1] = 0.5*qdot[0]*(0.6*s23*c23 - 0.744*s2*s23 + 0.744*c2*c23) +
            0.5*qdot[1]*(1.38*c2 + 0.268*s23 - 0.0476*s2) +
            0.5*qdot[2]*(0.268*s23 - 0.00397*c23);

cor[0][2] = 0.5*qdot[0]*(0.6*s23*c23 + 0.744*c2*c23) +
            0.5*qdot[1]*(0.268*s23 - 0.00397*c23) +
            0.5*qdot[2]*(0.268*s23 - 0.00794*c23);

cor[1][0] = 0.5*qdot[0]*(-0.6*s23*c23 + 0.744*s2*s23 -0.744*c2*c23) +
            0.199*qdot[2]*c23;

cor[1][1] = 0.372*qdot[2]*c3;

cor[1][2] = 0.00199*qdot[0]*c23 + 0.372*qdot[1]*c3 +
             0.5*qdot[2]*(0.744*c3 + 0.022*s3);

cor[2][0] = 0.5*qdot[0]*(-0.6*s23*c23 + 0.744*s2*s23 -0.744*c2*c23) +
            0.00199*qdot[2]*c23;

cor[2][1] = 0.372*qdot[2]*c3;

cor[2][2] = 0.00199*qdot[0]*c23 + 0.372*qdot[1]*c3 +
             0.5*qdot[2]*(0.744*c3 + 0.022*s3);

cor[0][3] = 0;
cor[0][4] = 0;
cor[0][5] = 0;
cor[1][3] = 0;
cor[1][4] = 0;
cor[1][5] = 0;
cor[2][3] = 0;
cor[2][4] = 0;
cor[2][5] = 0;
cor[3][0] = 0;
cor[3][1] = 0;
cor[3][2] = 0;
cor[3][3] = 0;
cor[3][4] = 0;
```

```
cor[3][5] = 0;
cor[4][0] = 0;
cor[4][1] = 0;
cor[4][2] = 0;
cor[4][3] = 0;
cor[4][4] = 0;
cor[4][5] = 0;
cor[5][0] = 0;
cor[5][1] = 0;
cor[5][2] = 0;
cor[5][3] = 0;
cor[5][4] = 0;
cor[5][5] = 0;


}
```