SPEED CONTROL OF DC MOTOR USING SELF TUNED FUZZY CONTROLLER

A DISSERTATION

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

MASTER OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

(With Specialization in Control and Guidance)

By

AJIT KUMAR SINGH

DEPARTMENT OF ELECTRONICS AND COMPUTER ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE -247 667 (INDIA)
JUNE, 2010

CANDIDATE'S DECLARATION

I hereby declare that the work, which is presented in this dissertation report, titled **"Speed Control Of DC Motor Using Self Tuned Fuzzy Controller"**, being submitted in partial fulfillment of the requirements for the award of the degree of **Master of Technology** with specialization in **Control and Guidance**, in the Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee is an authentic record of my own work carried out from July 2009 to June 2010, under guidance and supervision of **Dr. R. MITRA**, Professor, Department of Electronics and Computer Engineering, Indian Institute of Technology, Roorkee.

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

Date : 29/06/10

Place: Roorkee

Ajit Kumar Singh

AJIT KUMAR SINGH

CERTIFICATE

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

[illegible]

(Dr. R. MITRA)
Professor, E&C Department,
Indian Institute of Technology, Roorkee
Roorkee – 247 667, (INDIA)

ACKNOWLEDGEMENTS

It is my privilege and pleasure to express my profound sense of respect, gratitude and indebtedness to my guide, Dr. R. MITRA, Professor, Department of Electronics and Computer Engineering , Indian Institute of Technology, Roorkee, for his inspiration, guidance, constructive criticisms and encouragement throughout this dissertation work.

Thanks are due to the Lab staff Control and Guidance Lab, Department of Electronics and Computer Engineering, IIT Roorkee for providing necessary facilities.

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

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

AJIT KUMAR SINGH

ABSTRACT

Sensitivity and Robustness is the primary issue while designing the controller for non-linear systems. One of the performance objectives for controller design is to keep the error between the controlled output and the set-point as small as possible. Controlling the speed and position of a DC Motor in all the condition is not possible by conventional control technique. The classical control methods have its limitations. Most of these techniques are based on hit and trial method. The response of the system is also not proper. And in the case of disturbance or parameter variation it fails to control the system.

PID controller is one of the most basic and successful controller. It controls the disturbance and parameter variation up to some extent. But the response of the system changes to a large extent. For controlling the response Fuzzy Controller is used. The fuzzy controller helps in controlling the system non-linearity.

A much improved system response can be achieved by Self Tuned Fuzzy Controller, which adjust the gain of controller according to the system variation or disturbance. However this response can be further improved by the systematic analysis of system response. Manually we can adjust the fuzzy rules by analysing the effect of each rule over the system. This analysis can be made through the graphical analysis of the system response.

Adaptive Fuzzy Controller is the most advanced form of controller. It adjusts the fuzzy logic based controller to give better response. We have used a Tabular Based Adaptive Fuzzy Controller to control the system non-linearity. It is a sugeno type controller, which adjust the rule base according to the system variation.

Controlling the position of a DC Motor is a major control issue. A Neuro-Fuzzy Controller helps us to design a controller which is build from the data base of the system. The FIS generated after training is very effective in controlling the position of DC Motor. As it is compared with the PID controller under loaded condition, the response of the controller generated through ANFIS is more robust than PID. This shows that FIS generated have more adaptability than normal PID controller.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Direct current (DC) motors have been widely used in many industrial applications such as electric vehicles, steel rolling mills, electric cranes, and robotic manipulators due to precise, wide, simple, and continuous control characteristics. Traditionally rheostatic armature control method was widely used for the speed control of low power dc motors. However the controllability, cheapness, higher efficiency, and higher current carrying capabilities of static power converters brought a major change in the performance of electrical drives.

The purpose of a motor speed controller is to take a signal representing the demanded speed, and to drive a motor at that speed. The controller may or may not actually measure the speed of the motor. If it does, it is called a Feedback Speed Controller or Closed Loop Speed Controller, if not it is called an Open Loop Speed Controller. Feedback speed control is better, but more complicated, and may not be required for a simple purpose. Motors come in a variety of forms, and the speed controller's motor drive output will be different dependent on these forms.

The desired torque-speed characteristics could be achieved by the use of conventional proportional-integral-derivative (PID) controllers. Speed of a DC Motor can also be controlled by the conventional controller, which are build of power electronic circuit. As PID controllers require exact mathematical modeling, the performance of the system is questionable if there is parameter variation. The speed controling crcuit usualy do not work properly, if the paramiters of the system is changed.

In recent years fuzzy logic and neural network controllers were effectively introduced to improve the performance of nonlinear systems. The application of NNC and Fuzzy Control is very promising in system identification and control due to learning ability, massive parallelism, fast adaptation, inherent approximation capability, and high degree of tolerance.

Fuzzy logic based controller are very effective in controlling the non-linearity of a system. The application of fuzzy logic is an effective alternative for any problem where logical inferences can be derived on the basis of causal relationships. As a mathematical method which encompasses the ideas of vagueness, fuzzy logic attempts to quantify linguistic terms so the variables thus described can be treated as continuous, allowing the system's characteristics and response to be described without the need for exact mathematical formulations.

However the small non-linearity of a system can easily be controlled by a fuzzy controller. But the system with large variation can be controlled effectively by adaptive fuzzy controller. As it is known to us that there are many parameters which effect a fuzzy controller, like membership function, rule base, scaling factor. In the adaptive fuzzy controller these parameters are adjusted according to the system variation. The adaptive controller can tune membership function using performance criteria. It can be a self organizing controller or a model based controller. Adaptive Fuzzy controller used in by us is a tabular based adaptive fuzzy controller. It basicaly consist of two fuzzy rule base. One is static and another one changes its rule base with the change in system, to make combat parameter variation.

One of the most important fuction of a controller is to give controlled output at optimum value. The optimization of of the controller parameter is also vary important for design purpose. The optimization can be done through vareous methods. One of the most effective method is genetic algorithm. The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution[1].

Training the controller with system variation is one of advancing field in control system. This training of controller can be done with Artificial Nural Network. Neural networks are composed of simple elements operating in parallel. These elements are inspired by biological nervous systems. As in nature, the network function is determined largely by the connections between elements. We can train a neural network to perform a particular function by adjusting the values of the connections (weights) between elements.

1.1 History and Background[2]

The first variable speed drives were certainly mechanical and were based on adjustable pitch diameter pulleys. Such systems are still in use but for obvious reasons are not in general uses in industrial applications today. The brushed DC motor was invented in 1856 by Werner

Von Siemens in Germany. Variable speed by armature voltage control was first used in the early 1930s using a system involving a constant speed AC motor driving a D.C. generator. The generator's DC output was varied using a rheostat to vary the field excitation and the resulting variable voltage DC was used to power the armature circuit of another DC machine used as a motor. This system was called a Ward-Leonard system after the two people credited with its development. The Ward-Leonard method of DC variable speed control continued until the late 1960s when Electric Regulator Company brought to market a practical, general purpose, static, solid state controller that converted the AC line directly to rectified DC using SCR (thyristor) devices. That technology was adopted by virtually all manufacturers and still is in use today. In brushless DC Motor, Voltage on the motor determines speed.

1.2 Motivation

DC Motor has great industrial application. Due to the large application its controlling is a major area of research. In control system there are some clasical methods to control the speed of motor. These methods are like speed control of DC Motor by root locus technique, speed control of DC Motor by the frequency domain anlysis, and speed control by the state space analysis. All these methods are are based on trial and error. Also the result of these methods are not very much upto the mark. Although PID controller is able to control the speed of DC Motor with full accurary. But in some cases it not appropriate.

PID controller has its limitations like, it is not suitable for the system which has changing parameters. This limitation urge us to design controller for the speed control of DC Motor. The motor with small non-linearity can be controlled by fuzzy logic controller. But for the large variation we needed adaptive controller to control the motor.

We have initially controlled the speed of motor by a chopper circuit. It was able to control the speed, but the response of the system has little oscillation. This motivated us to design a controller which has minimum oscillation.

1.3 Problem Statement

The classical control methods for the speed control of DC Motor has many problems in controlling the speed of motor. The output of the system is not perfect by these controller. The response of the system is eighter overdamped or underdamped. Also in some cases it fails to meet the design requirement like, settling time not more than 2 seconds, overshoot less than 5%, steady state error less than 1% etc.

To over come these short coming PID controller is designed. It is able to meet all the avobe design criteria. But it fails to control the variation in system parameter. This problem is removed by fuzzy controller. However this controller also has its limitation, like it not able to control the large variation in the system. Then to remove all these problem we have applied adaptive controller. We have used two types of adaptive controller Adaptive Fuzzy controller and Adaptive Neuro Fuzzy Inference System.

1.4 Outline of the dissertation

In this chapter we have introduced DC Motor and its industrial application. History of the speed control of DC Motor drive is introduced in brief. And the motivation of the dissertation. In chapter 2, basic principal of the operation of DC Motor is introduced. Its characteristic is dealed in brief. And the speed control methods of DC Motor is explained. Chapter 3, contains the modeling part of DC Motor. It also deals with the classical control system technique to control the speed of motor. The methods explained are PID speed control technique, Root Locus method of speed control, speed control by Frequency Domain analysis and Digital PID speed control method. Chapter 4 contains the theoretical explanation of Genetic Algorithm, which includes its structure, mechanism and advantage. Chapter 5 deals with the tuning of fuzzy control and modeling of DC Motor drive in fuzzy model. In chapter 6 adaptive fuzzy controller is explained. It also includes the neuro fuzzy conroller explanation. The complite algorithm and equation of ANFIS is explained. Chapter 7 contains the simulation result. A new method of rule tuning is explained in this chapter. It contains the simulation model and results of comperetive study among Adaptive Fuzzy, Simple Fuzzy and PID controller. The comperetive study of ANFIS and a PID controller is also explained with an example in matlab simulation.

2. DC MOTOR

A dc machine is constructed in many forms and for a variety of purposes, from 3-mm stepper motor drawing a few μA at 1.5 V in a quartz crystal watch to the giant 75000-kW or more rolling mill motor. It is a highly versatile and flexible machine. It can satify the load requiring high starting, accelerating and retarding tourques. A dc machine is also easily adaptable for drives with a wide range of speed control and fast reversals.

2.1 History and background [3]

At the most basic level, electric motors exist to convert electrical energy into mechanical energy. The basic principles of electromagnetic induction were discovered in the early 1800's by Oersted, Gauss, and Faraday. By 1820, Hans Christian Oersted and Andre Marie Ampere had discovered that an electric current produces a magnetic field. The next 15 years saw a flurry of cross-Atlantic experimentation and innovation, leading finally to a simple DC rotary motor. **Faraday** set to work devising an experiment to demonstrate whether or not a current-carrying wire produced a circular magnetic field around it, and in October of 1821 succeeded in demonstrating this. **Joseph Henry (U.S.)** praposed by the summer of 1831 Joseph Henry had improved on Faraday's experimental motor. Henry built a simple device whose moving part was a straight electromagnet rocking on a horizontal axis. Its polarity was reversed automatically by its motion as pairs of wires projecting from its ends made connections alternately with two electrochemical cells. Two vertical permanent magnets alternately attracted and repelled the ends of the electromagnet, making it rock back and forth at 75 cycles per minute. **William Sturgeon (U.K.)** Just a year after Henry's motor was demonstrated, William Sturgeon invented the commutator, and with it the first rotary electric motor -- in many ways a rotary analogue of Henry's oscillating motor. Sturgeon's motor, while still simple, was the first to provide continuous rotary motion and contained essentially all the elements of a modern DC motor. Sturgeon used horseshoe electromagnets to produce both the moving and stationary magnetic fields (to be specific, he built a shunt wound DC motor).

2.2 Principles of operation[4]

In any electric motor, operation is based on simple electromagnetism. A current carrying conductor generates a magnetic field; when this is then placed in an external magnetic field, it will experience a force proportional to the current in the conductor, and to the strength of the external magnetic field. As we know, opposite (North and South) polarities attract, while like polarities (North and North, South and South) repel. The internal configuration of a DC motor is designed to harness the magnetic interaction between a current-carrying conductor and an external magnetic field to generate rotational motion. The operating principle can be seen clearly in Fig 2.2.1

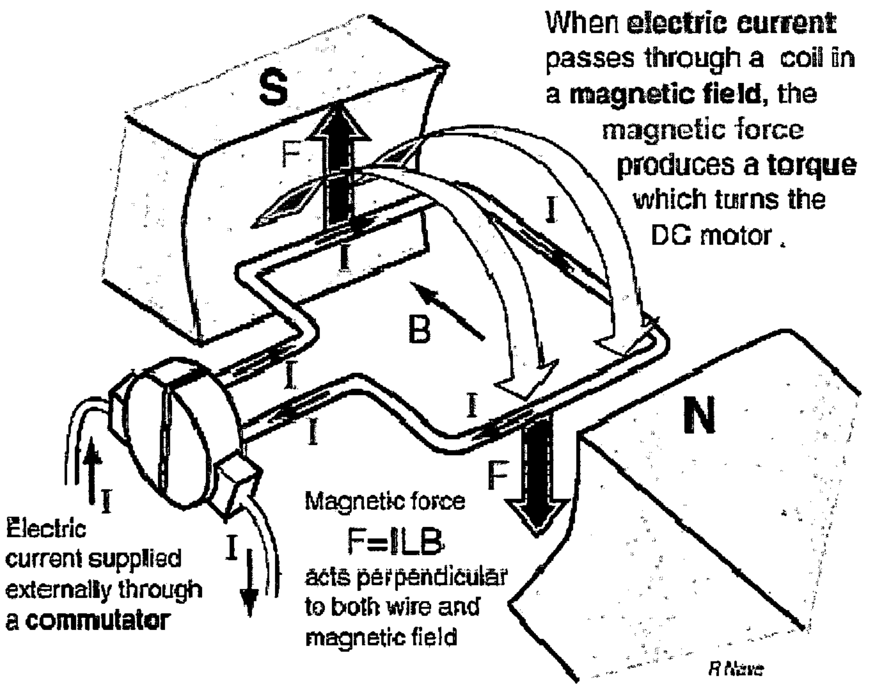
When electric current passes through a coil in a magnetic field, the magnetic force produces a torque which turns the DC motor.
S
N
F
I
B
Electric current supplied externally through a commutator
Magnetic force F=ILB acts perpendicular to both wire and magnetic field
R Nave

Fig 2.2.1 DC Motor Operation

2.3 Some Basic Equations Of DC Motor

a. The magnetic flux Φ_f is generated by the field winding current I_f and Φ_f is proportional to this field current.

   So, $\Phi_f = K_f * I_f$ (Kf is constant)

b. If the magnetic flux Φ_f is generated by permanent magnet, Φ_f can generally be expressed as a constant.

c. When the rotor is rotated, the flux linking those rotor windings will be changed during the rotation. The rate of flux linkage changes will be proportional to the motor speed ω_m.

d. The winding induced voltage can be expressed as:

$Ea = K^* \Phi_f * \omega_m = K * K_f * I_f * \omega_m = K_s * I_f * \omega_m$

e. Neglecting the losses the output power is :

$Pout = Ea * I_a = Ks * I_f * \omega_m * Ia$

f. Hence the torque output of DC motor is :

$Tout = Pout/\omega_m = Ks * I_f * \omega_m * Ia/\omega_m = Ks * I_f * Ia$

2.4 Characteristics of DC Motor

The great power of the dc motor lies in its versatility and ease with which a variety of speed-torque charecteristic can be obtained, and the wide range of speed control which is possible without the need of elaborated control scheme while a high level of operating efficiency is maintained. DC motors are of three types according to how these are excited

a. Shunt Motor

b. Series Motor

c. Compound Motor

c.1 Cumulative Compound Motor

c.2 Differential Compound Motor.

2.5 Speed Control Of DC Motor

The dc motor are in general much more adaptable speed drives than ac motors which are associated with a constant- speed rotating field. Indeed one of the primary reason for the competative position of dc motors in modern industrial drives is the wide range of speeds afforded. As it can be seen from equation given below

$$n = K\left(\frac{V - I_a R_a}{\varphi}\right) \qquad (1)$$

where

n = speed, φ = flux

Since the armature drop is small, it can be neglected

The avobe equation gives the two method of speed control i.e the variation of field excitation and terminal voltage control.

Field Control

Field control in shunt motor is achieved by means of a rehostat in the field circuit. Sreies motor has three different ways of changing field current. These are diverted field control, tapeed field control, and series parallel control.

Armature Control

There are three main types of armature control scheme rehostatic control, shunted armature control, and series parallel control.

3.CLASSICAL METHODS OF DC MOTOR SPEED CONTROL

3.1 Modeling of dc motor

A common actuator in control systems is the DC motor. It directly provides rotary motion and, coupled with wheels or drums and cables, can provide transitional motion. The electric circuit of the armature and the free body diagram of the rotor are shown in the following figure 3.1.1

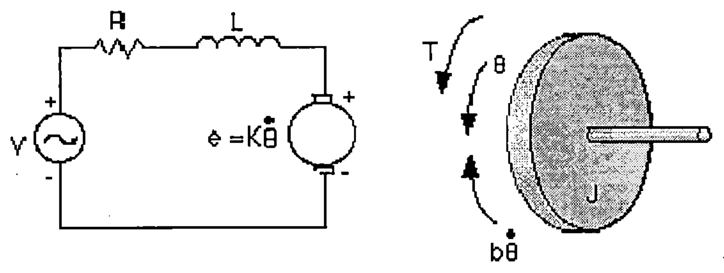

Fig. 3.1.1 Circuit Diagram, and Free body Diagram of Rotor

we will assume the following values for the physical parameters:-

* moment of inertia of the rotor (J) = 0.01 kg.m^2/s^2

* damping ratio of the mechanical system (b) = 0.1 Nms

* electromotive force constant (K=Ke=Kt) = 0.01 Nm/Amp

* electric resistance (R) = 1 ohm

* electric inductance (L) = 0.5 H

* input (V): Source Voltage

* output (theta): position of shaft

The rotor and shaft are assumed to be rigid. The motor torque, **T**, is related to the armature current, **i**, by a constant factor **Kt**. The back emf, **e**, is related to the rotational velocity by the following equations:

$$T = K_t i \qquad 3.1.1$$

$$e = K_e\dot{\theta} \qquad 3.1.2$$

In SI units (which we will use), **Kt** (armature constant) is equal to **Ke** (motor constant).

From the figure 3.3.3 above we can write the following equations based on Newton's law combined with Kirchhoff's law:

$$J\ddot{\theta} + b\,\dot{\theta} = Ki \qquad 3.1.3$$

$$L\frac{di}{dt} + Ri = V - K\,\dot{\theta} \qquad 3.1.4$$

3.1.1 Transfer Function

Using Laplace Transforms, the above modeling equations 3.1.3 can be expressed in terms of s.

$$s(Js + b)\,\theta(s) = K\,I(s) \qquad 3.1.5$$

Simillarly equation 3.1.4 can be expressed as

$$(Ls + R)\,I(s) = V - Ks\,\theta(s) \qquad 3.1.6$$

By eliminating I(s) we can get the following open-loop transfer function, where the rotational speed is the output and the voltage is the input.

$$\frac{\dot{\theta}}{V} = \frac{K}{(Js + b)(Ls + R) + K^2} \qquad 3.1.7$$

3.1.2 State-Space

In the state-space form, the equations above can be expressed by choosing the rotational speed and electric current as the state variables and the voltage as an input. The output is chosen to be the rotational speed.

$$\frac{d}{dt}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} = \begin{bmatrix}-\frac{b}{J} & \frac{K}{J}\\ -\frac{K}{L} & -\frac{R}{L}\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} + \begin{bmatrix}0\\ \frac{1}{L}\end{bmatrix} \qquad 3.1.8$$

$$\dot{\theta} = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\dot{\theta}\\ i\end{bmatrix} \qquad 3.1.9$$

3.2 Matlab representation and open-loop response

We can represent the above transfer function into Matlab by defining the numerator and denominator matrices as follows:

Numerator = K, Denominator = $(Js + b)(Ls + R) + K^2$

Matlab m-file is written to check the open loop response of the system. It is written for both transfer function and state space equation.

Design requirements

First, our uncompensated motor can only rotate at 0.1 rad/sec with an input voltage of 1 Volt which can be seen from fig 3.2.1 given below. Since the most basic requirement of a motor is that it should rotate at the desired speed, the steady-state error of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed as soon as it turns on. In this case, we want it to have a settling time of 2 seconds. Since a speed faster than the reference may damage the equipment, we want to have an overshoot of less than 5%.

If we simulate the reference input (r) by an unit step input, then the motor speed output should have:

- Settling time less than 2 seconds
- Overshoot less than 5%
- Steady-state error less than 1%

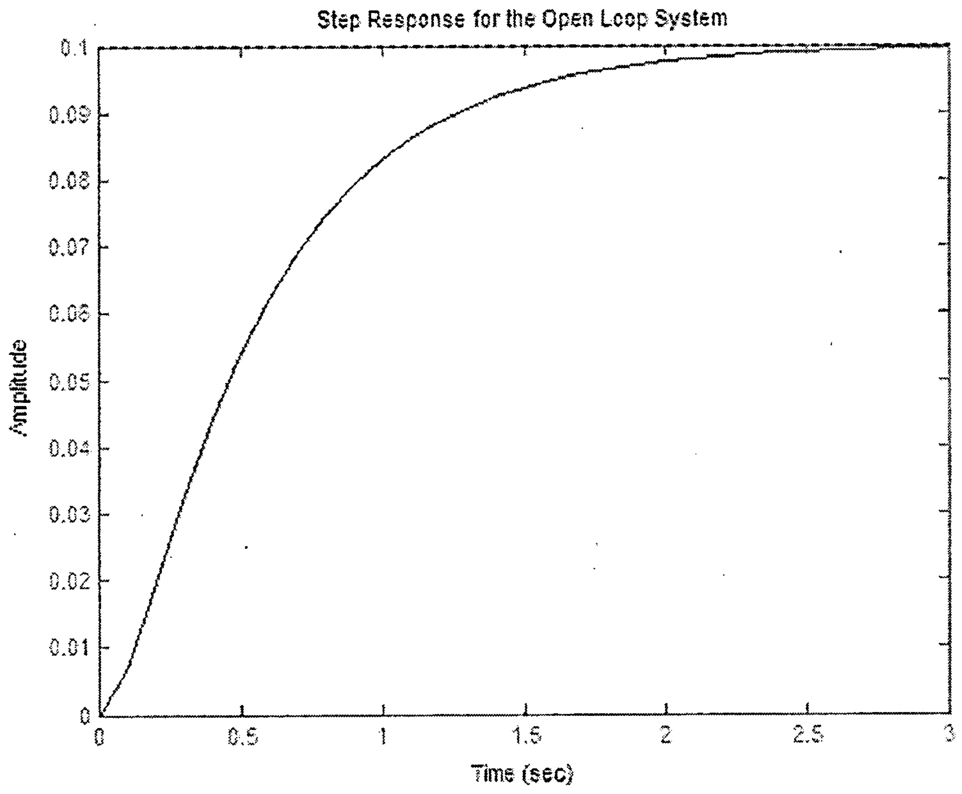

Fig 3.2.1 Step Response of Open Loop System

From the plot we see that when 1 volt is applied to the system, the motor can only achieve a maximum speed of 0.1 rad/sec, ten times smaller than our desired speed. Also, it takes the motor 3 seconds to reach its steady-state speed; this does not satisfy our 2 seconds settling time criterion.

3.3 PID Design Method for DC Motor Speed Control

A PID controller is used for the speed control of dc motor. The transfer function of PID controller is given is:

$$K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s} \qquad 3.3.1$$

The closed loop system used for controlling speed is given in figure 3.3.1

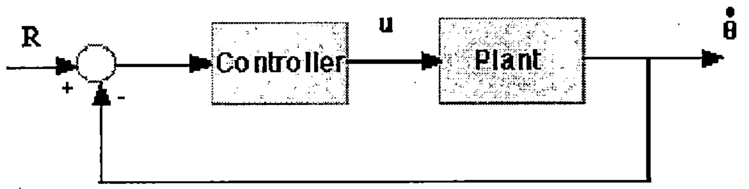

Fig 3.3.1 Closed loop system model for dc motor speed control

R = Reference speed

u = Controller output

$\dot{\theta}$ = Output Speed

We first started our controlling with simple proportional controller (K_p). We taken gain $K_p = 100$. By adding few more codes to the m-file of 3.2.1, for adding the gain K_p and to get the closed loop response of system. The closed loop response of dc motor with proportional controller is shown in fig 3.3.2 from the plot we can see that the system has large steady state error and large overshoot by adding an integral term we can eliminate the steady-state error and a derivative term will reduce the overshoot. We have started PID controller with small K_i and K_d.

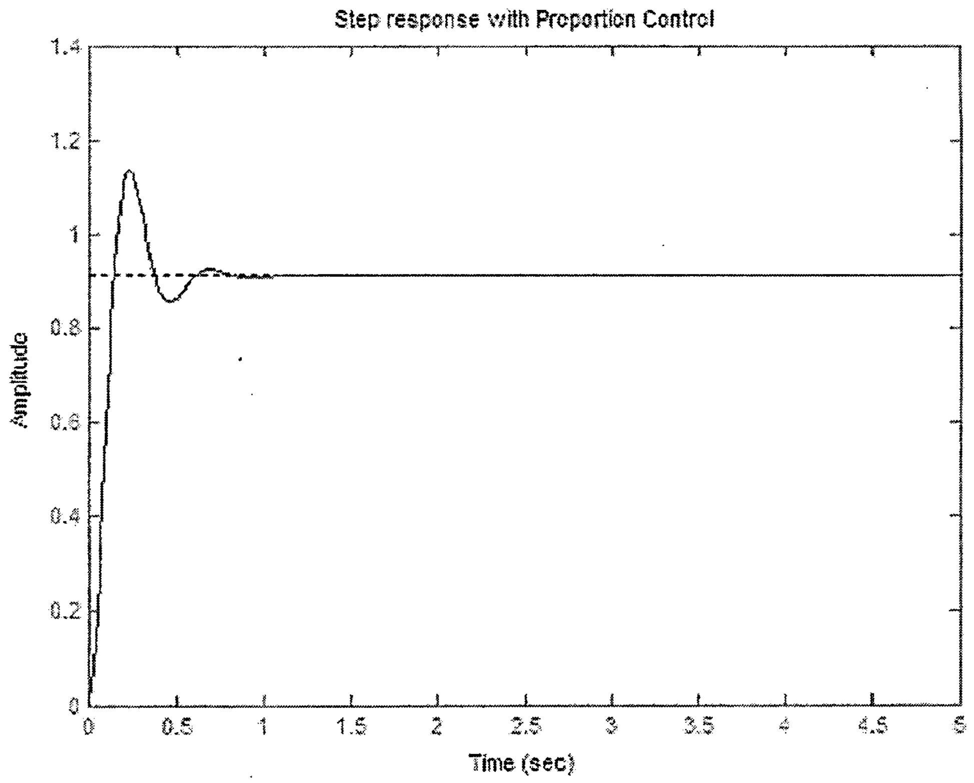

Fig 3.3.2 Closed Loop Step Response with Proportional Controller

By manually tuning the three components of PID controller K_p, K_i, and K_d. This is done trough manually adjusting these gains. Through this hit and trial method we finally got the values of three gains which designed the controller to full fill all the design requirement given in starting.

The gains are

$K_p = 100$

$K_i = 200$

$K_d = 10$

After adding few more codes to the 3.2.1 m-file we got the response given in figure 3.3.3

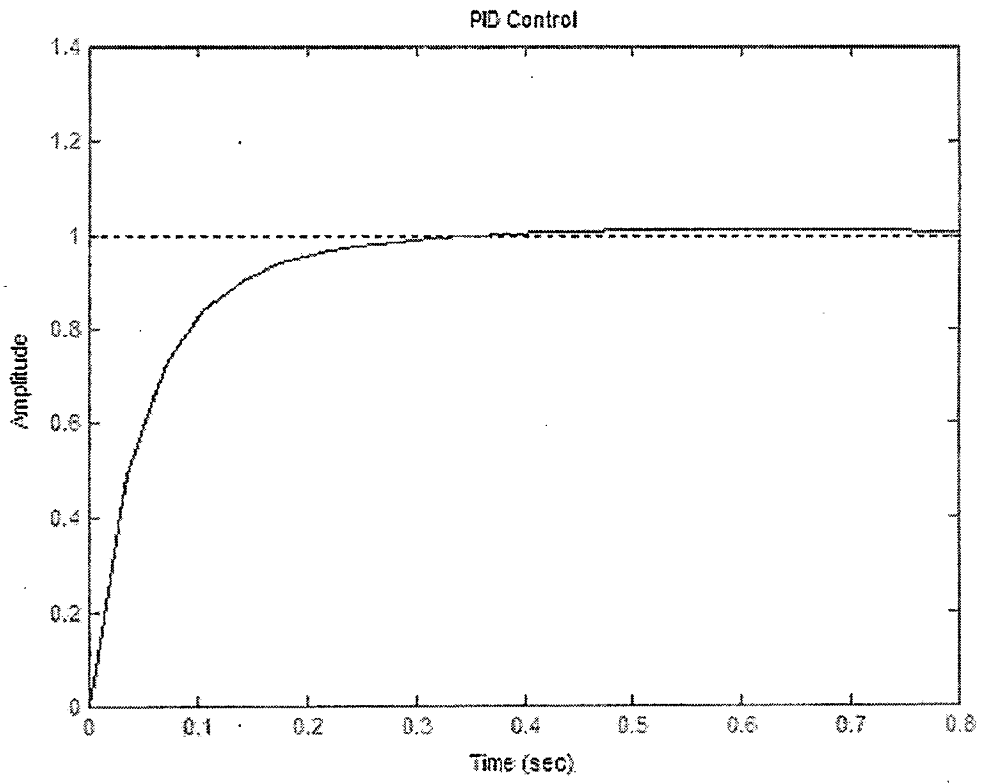

Fig 3.3.3 PID Controlled Output

3.4 Root Locus Design Method for DC Motor Speed Control

A root loci plot is simply a plot of the s zero values and the s poles on a graph with real and imaginary coordinates. The root locus is a curve of the location of the poles of a transfer function as some parameter (generally the gain K) is varied. In addition to determining the stability of the system, the root locus can be used to design for the damping ratio and natural frequency of a feedback system. Lines of constant damping ratio can be drawn radially from the origin and lines of constant natural frequency can be drawn as arcs whose center points coincide with the origin. By selecting a point along the root locus that coincides with a desired damping ratio and natural frequency a gain, K, can be calculated and implemented in the controller. Many controllers like lag, lead, PI, PD and PID controllers can be designed approximately with this technique.

The main idea of root locus design is to find the closed-loop response from the open-loop root locus plot. Then by adding zeros and/or poles to the original plant, the closed-loop response can be modified.

Speed control fo dc motor through root locus design involves following four steps that we have used:

a. Drawing the open-loop root locus
b. Finding the gain using the rlocfind command
c. Adding a lag controller
d. Plotting the closed-loop response

Two arguments in the sgrid command are the damping ratio (zeta) term (0.8 corresponds to a overshoot of 5%), and the natural frequency (Wn) term (= 0 corresponds to no rise time criterion) respectively. The single argument in the sigrid command is the sigma term (4.6/2 seconds = 2.3).

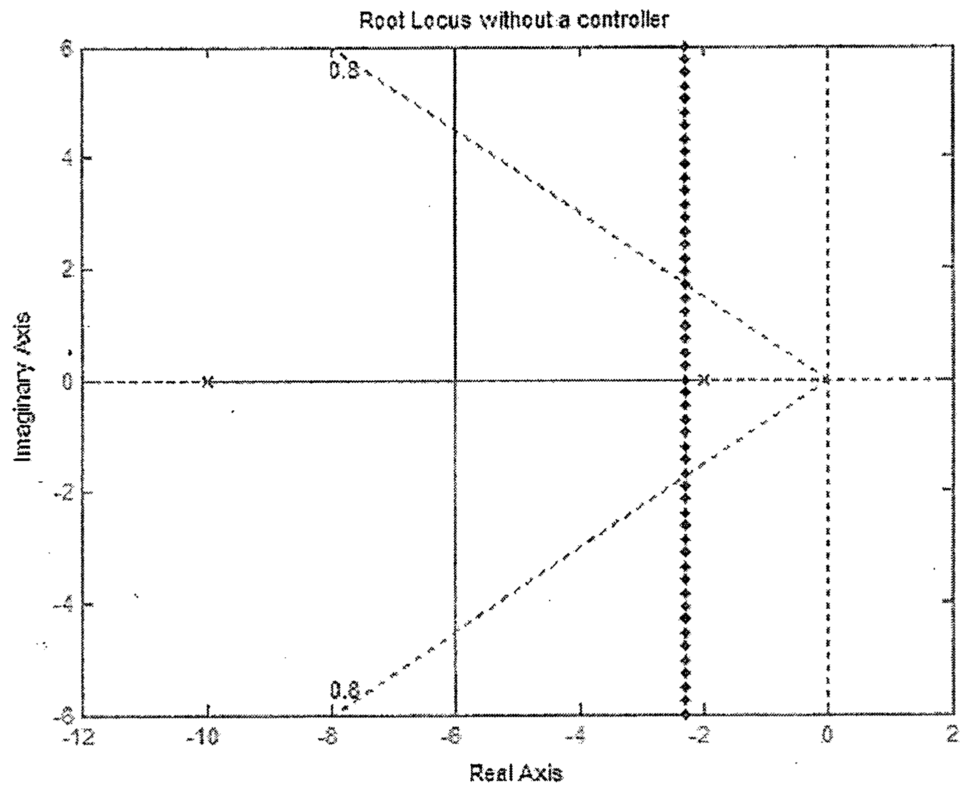

Fig 3.4.1 Root Locus of DC Motor Without Controller

To find gain in root locus plot, we have used rlocfind command

As per our design requirement we need the settling time and the overshoot to be as small as possible. We from the property of rout locus that location of poles determine the behaviour of system. Large damping corresponds to points on the root locus near the real axis. A fast response corresponds to points on the root locus far to the left of the imaginary axis. To find the gain corresponding to a point on the root locus, we can use the rlocfind command. We can find the gain and plot the step response using this gain all at once. This can be done by adding some commands to 3.4.1 m-file. i.e adding proportional gain to the system and the closed loop step response is seen in the figure of output.

```
[k,poles] = rlocfind(num,den)
[numc,denc]=cloop(k*num,den,-1);
t=0:0.01:3;
step(numc,denc,t)
title('Step response with gain')
```

from the plot and we have selected a point on the root locus half-way between the real axis and the damping requirement, say at -6+2.5i. From the camand window of matlab, we got the value of selected point, gain and the location of poles. Selected point = -6.0118 + 2.2547i, k = 10.5317

poles = -6.0000 + 2.2546i, -6.0000 - 2.2546i

The step response for the coresponding gain is shown in the figure 3.4.2 :

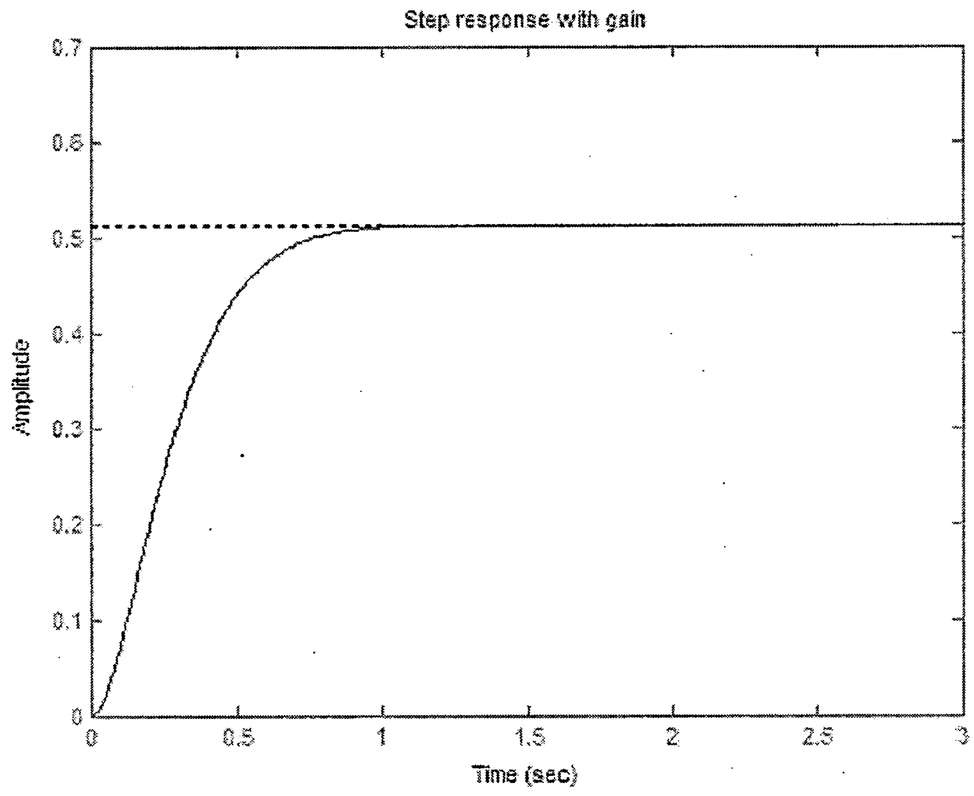

Fig3.4.2 Step Response With Gain 10.53

As we have seen, the system is overdamped and the settling time is about one second, so the overshoot and settling time requirements are satisfied. The only problem we can see from this plot is the steady- state error of about 50%. If we increase the gain to reduce the steady-state error, the overshoot becomes too large. We need to add a lag controller to reduce the steady-state error.

Adding a lag controller

From the plot we have seen that this is a very simple root locus. The damping and settling time criteria were met with the proportional controller. The steady-state error is the only criterion not met with the proportional controller. A lag compensator can reduce the steady-state error. By doing this, we have increased our settling time. We have tried the following lag controller first:

$$\frac{(s+1)}{(s+0.01)} \qquad 3.4.1$$

For adding this lag controller the new m-file 3.4.3 is written. After running this 3.4.3 m-file we got plot very much similar to the figure 3.4.1. the plot is shown below in figure 3.4.3.

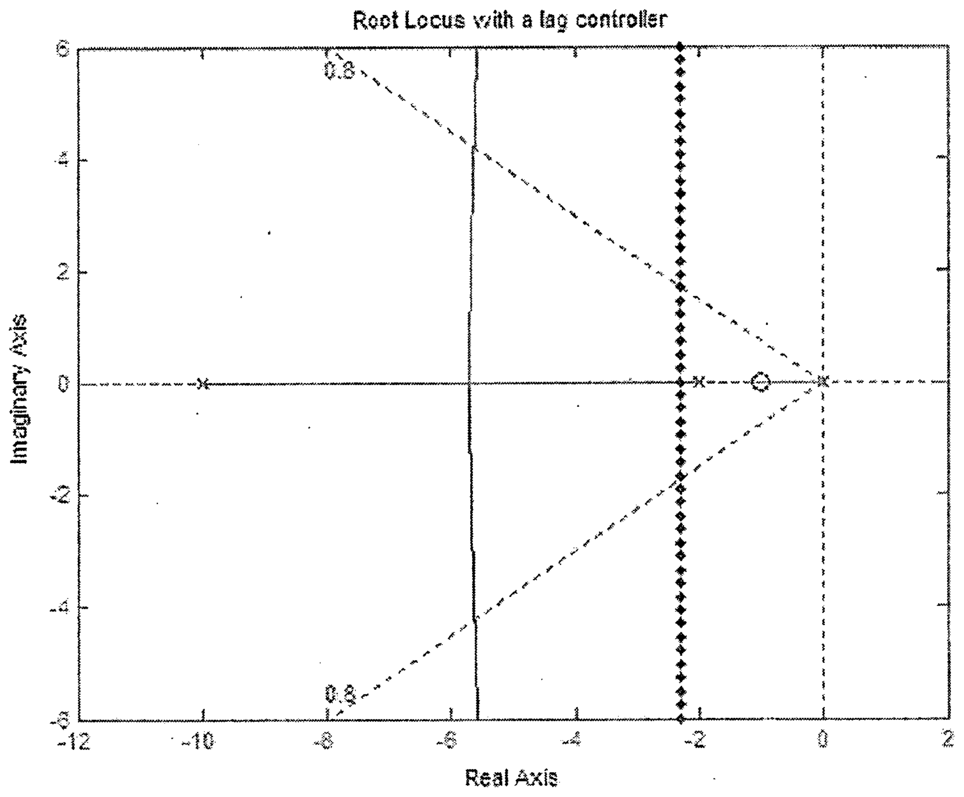

Fig 3.4.3 Root Locus With Lag Controller

Plotting the closed-loop response

We have closed the loop and seen the closed-loop step response. By Entering the following code at the end of 3.4.3 m-file:

```
[k,poles]=rlocfind(numb,denb)
[numc,denc]=cloop(k*numb,denb,-1);
t=0:0.01:3;
step(numc,denc,t)
title('Step response with a lag controller')
```

After reruning this m-file in the Matlab command window. We have than selected a point, near the damping requirement (diagonal dotted line). We got the plot shown in the figure 3.4.4:

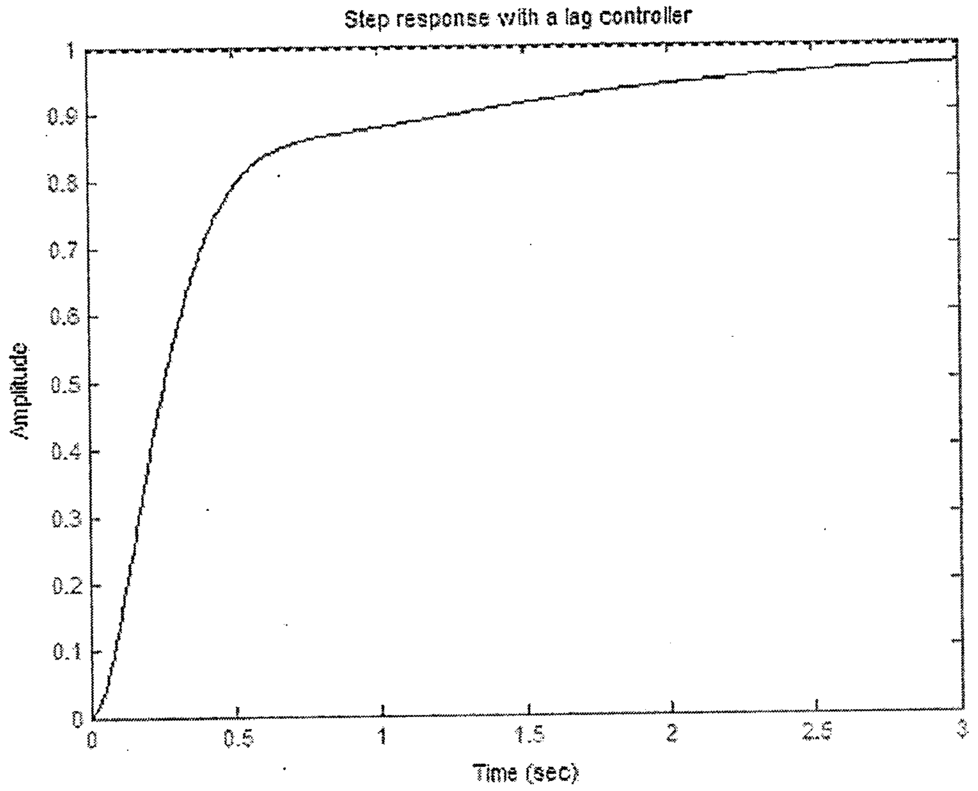

Fig 3.4.4 Step Response With a Lag Controller and Gain 19.4768

Our gain has to be about 20. As we can see the response is not quite satisfactory. We have also noted even though we have selected gain to correlate with a position close to the damping criterion, the overshoot is not even close to five percent. This is due to the effect of the lag controller kicking in at a later time than the plant. (its pole is slower). What this means is that we can go beyond the dotted lines that represent the limit, and get the higher gains without worrying about the overshoot . After reruning the m-file, we have placed the gain just above the white, dotted line. After trying number of times we got a satisfactory response. The response looked as shown in the figure 3.4.5. The gain of the system is arround 50.

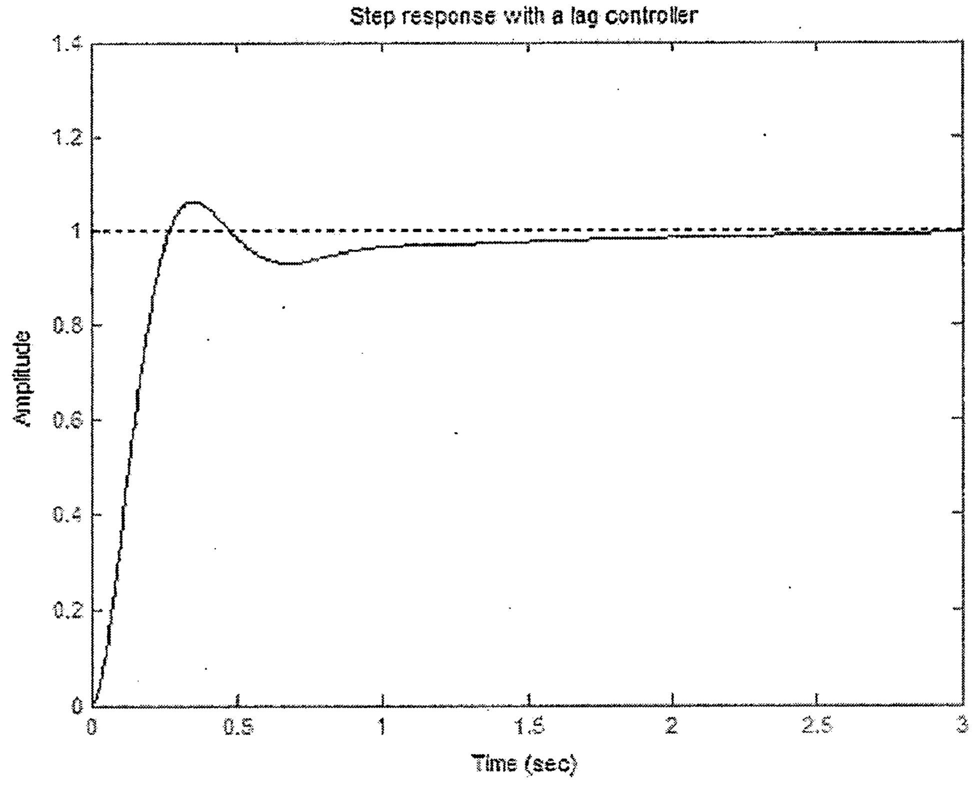

Fig 3.4.5 Step Response With Gain 51.27

The steady-state error is smaller than 1%, and the settling time and overshoot requirements have been met. As we can see, the design process for root locus is very much a trial and error process. If we had not been able to get a satisfactory response by choosing the gains, we could have tried a different lag controller, or even added a lead controller.

3.5 Frequency Design Method for DC Motor Speed Control

The main idea of frequency-based design is to use the Bode plot of the open-loop transfer function to estimate the closed-loop response. Adding a controller to the system changes the open-loop Bode plot, therefore changing the closed-loop response.

A Bode plot is a graph of the logarithm of the transfer function of a linear, time-invariant system versus frequency, plotted with a log-frequency axis, to show the system's frequency response. It is usually a combination of a Bode magnitude plot (usually expressed as dB of gain) and a Bode phase plot (usually expressed as degrees of phase shift).

We have designed the speed controller of dc motor through the steps given below:

a. Drawing the original Bode plot
b. Adding proportional gain
c. Plotting the closed-loop response
d. Adding a lag controller

We have started our designing by drawing the original bode plot of the dc motor.

Original bode plot is shown below fig 3.5.1:

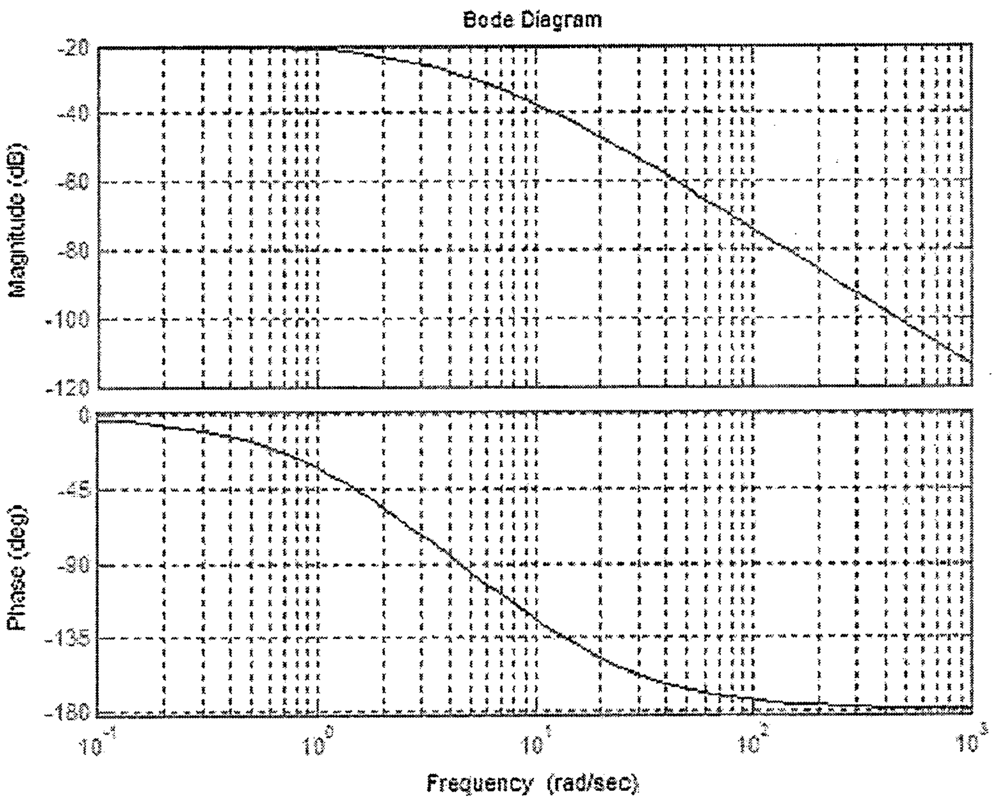

Fig 3.5.1 Original Bode Plot of DC Motor

Adding proportional gain

From the bode plot above, we have seen that the phase margin can be greater than about 60 degrees if w is less than 10 rad/sec. We have added gain to the system so the bandwidth frequency is 10 rad/sec, which will give us a phase margin of about 60 degrees. To find the gain at 10 rad/sec, we read it through Bode plot (it looks to be slightly more than -40 dB, or 0.01 in magnitude). The bode command is used to get the exact magnitute:

```
[mag,phase,w] = bode(num,den,10)

mag = 0.0139
```

To have a gain of 1 at 10 rad/sec, we have multiplyed the numerator by 1/0.0139 or approximately 72.

```
num = 70*num
```

Know again runing the m-file though slite modification we got the bode plot shown below:

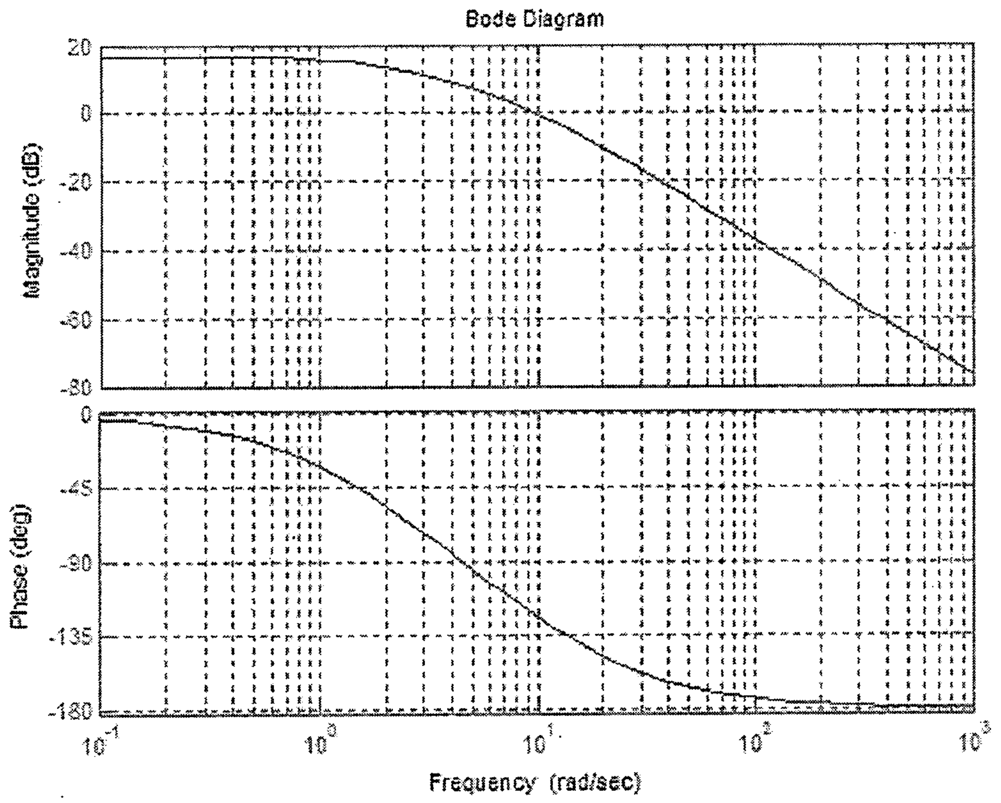

Fig 3.5.2 Bode Plot of DC Motor by Adding Proportional Gain

Plotting the closed-loop response

From the plot above we have seen that the phase margin is now quite large. The closed-loop response look like:

```
[numc,denc]=cloop(num, den, -1);

t=0:0.01:10;

step(numc,denc,t)
```

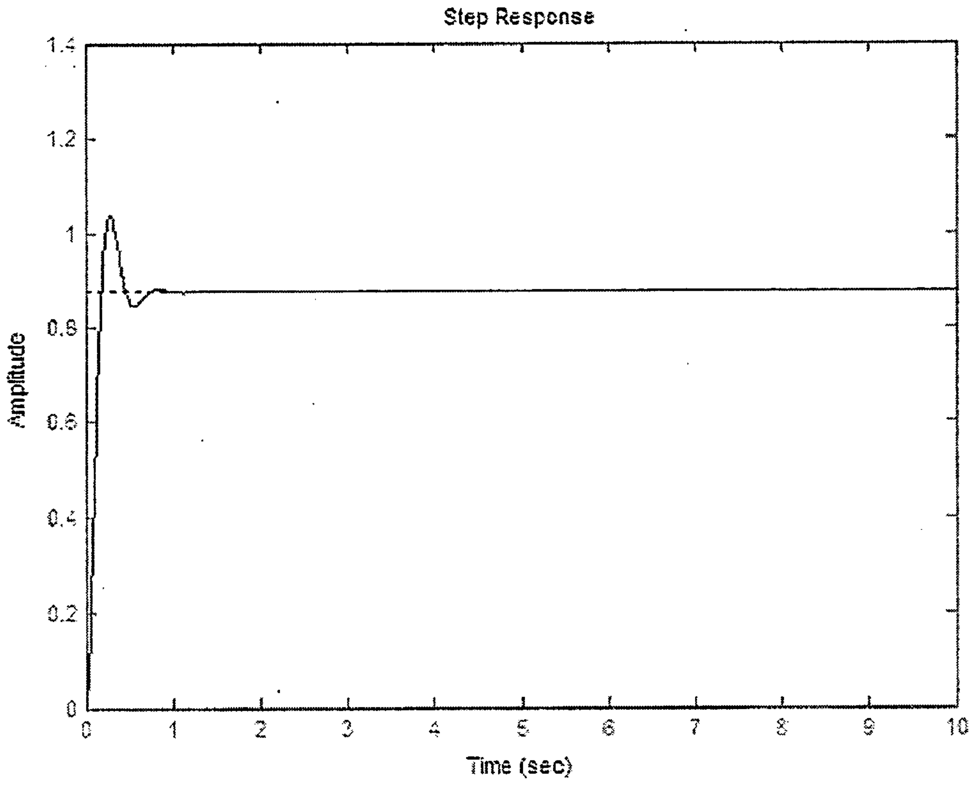

Fig 3.5.3 Closed Loop Step Response

We observed from figure 3.5.3 that the settling time is fast enough, but the overshoot and the steady-state error are too high. The overshoot can be reduced by reducing the gain a bit to get a higher phase margin, but this would cause the steady-state error to increase. We have applied a lag controller.

Adding a lag controller

We have added a lag controller to reduce the steady-state error. At the same time, we have tried to reduce the overshoot by reducing the gain. We have taken gain equal to 50. The lag controller is

$$\frac{(s+1)}{(s+0.1)} \qquad 3.5.1$$

the steady-state error is redused by a factor of 1/0.01 = 100 (but it increase the settling time).

By adding few codes to 3.5.1 m-file, we got the plot shown in figure 3.5.4.

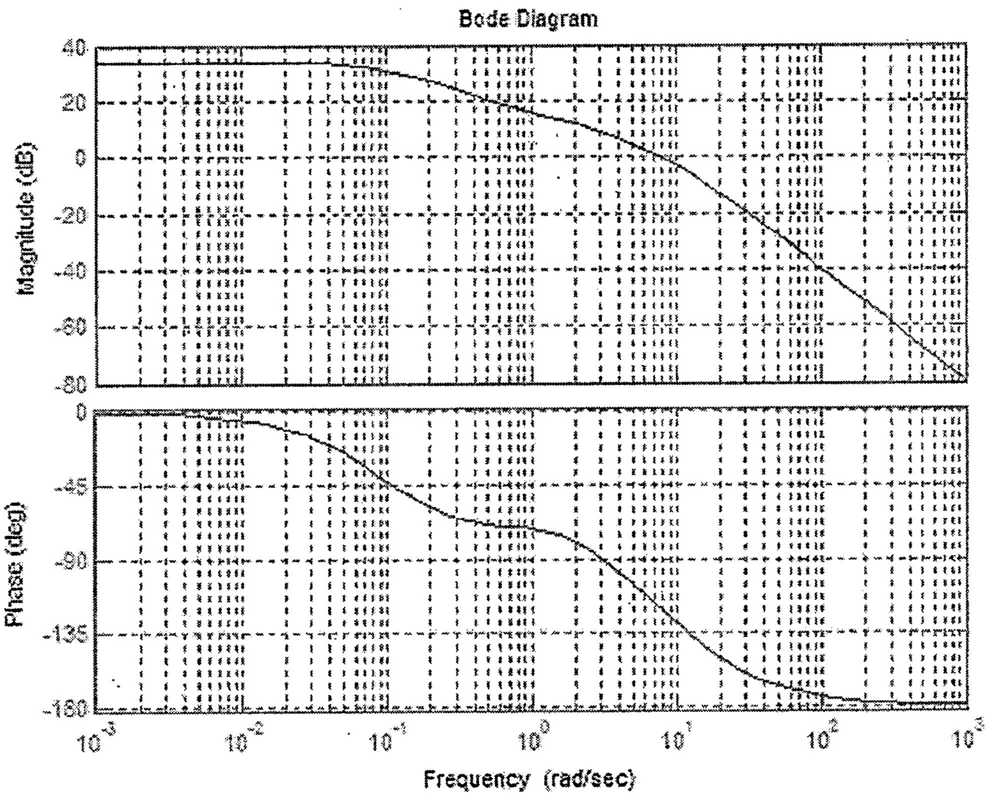

Fig 3.5.4 Bode Plot by Reducing Gain and Adding Lag Controller

The phase margin is quite good. The steady-state error is about 1/40dB or 1%, as desired. We have closed the loop and seen the step response. Added the following lines of code to the end of our m-file and rerun

```
[numc,denc]=cloop(numb, denb, -1);
t=0:0.01:10;
step(numc,denc,t)
```

We have seen the response given in figure 3.5.5. Finaly we had the step response that meets the design requirements.

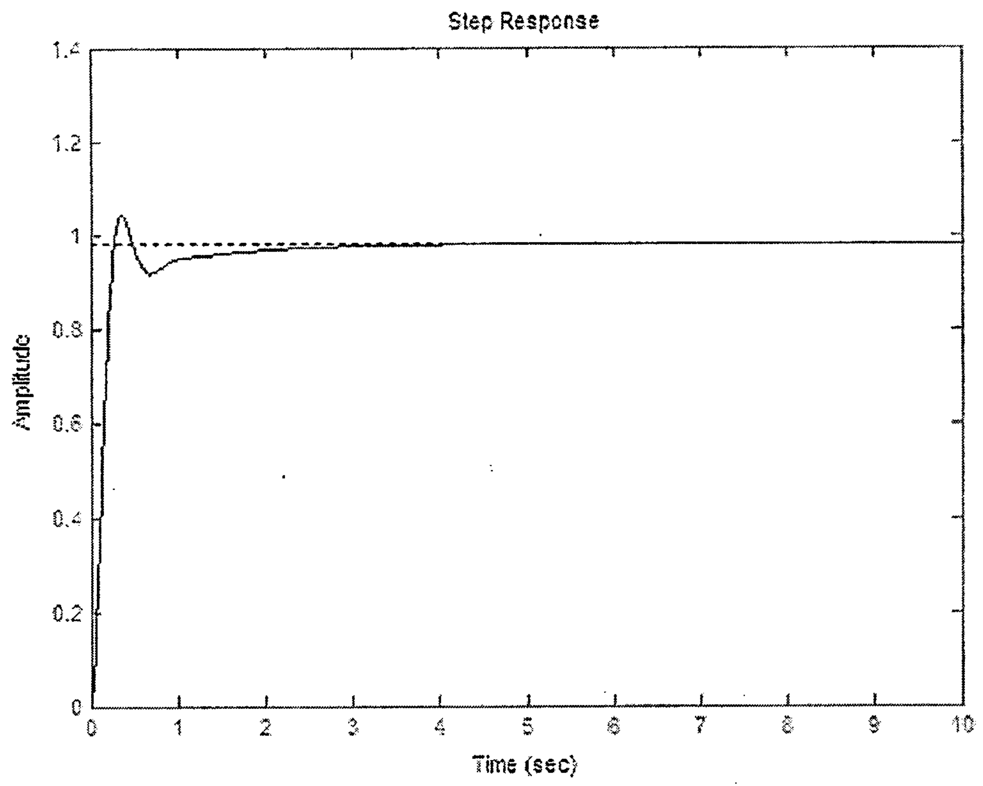

Fig 3.5.5 Closed Loop Step Response of DC Motor

3.6 Digital DC Motor Speed Control with PID Control

In this section, we havel considered the digital control version of DC motor speed problem. A digital DC motor model can be obtained from conversion of the analog model. The controller in this example is designed by PID method.

Prosedure for desining the digital speed controller of dc motor is:

a. Continuous to Discrete Conversion
b. PID Controller

The modeling of dc motor is dealed in section 3.1. The same parameters of motor is considered in this example.

Continuous to Discrete Conversion: The first step in designing a discrete control system is to convert the continuous transfer function to a discrete transfer function. Matlab command c2dm will do this. The c2dm command requires the following four arguments: the numerator polynomial (num), the denominator polynomial (den), the sampling time (Ts) and the type of hold circuit. In this example, the hold we have used is the zero-order hold ('zoh').

From the design requirement, we have taken sampling time, **Ts** equal to 0.12 seconds, which is 1/10 the time constant of a system with a settling time of 2 seconds.

After running the m-file 3.6.1 we got the discrite transfer function:

$$\frac{\dot{\theta}}{V} = \frac{0.0092z + 0.0057}{z^2 - 1.0877z - 0.2369} \qquad 3.6.1$$

First, we have seen the closed-loop response of the system any controller. As we have seen the numz matrices shown above, it has one extra zero in the front; we have to get rid of it before closing the loop with the Matlab cloop command. We haveadded the following code into the end of our m-file 3.6.1:

```
numz = [numz(2) numz(3)];
[numz_cl,denz_cl] = cloop(numz,denz);
```

After we have done this, seen the closed-loop step response looks like. The dstep command will generate the vector of discrete output signals and stairs command will connect these signals. Added the following Matlab code at the end of previous m-file and rerun it.

```
[x1] = dstep(numz_cl,denz_cl,101);
t=0:0.12:12;
stairs(t,x1)
xlabel('Time (seconds)')
ylabel('Velocity (rad/s)')
title('Stairstep Response:Original')
```

we have seen the response shown in the figure 3.6.1:

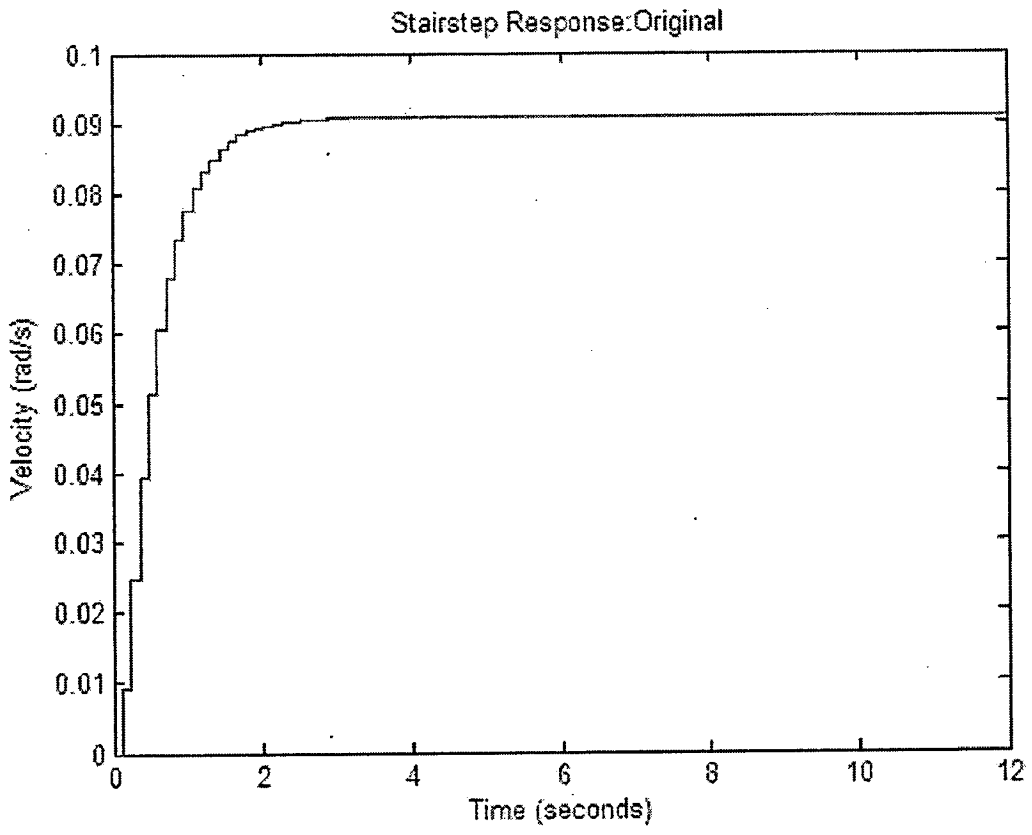

Fig 3.6.1 Stare Step Response of DC Motor

As we can see from the avobe figure the system has large steady state error. Now we have applied the PID controller to get the desierd response.

PID Controller

There are several ways for mapping from the s-plane to z-plane. The most accurate one is

$$z = e^{Ts}$$

We cannot obtain PID transfer function in this way because the discrete-time transfer function would have more zeroes than poles, which is not realizable. Instead we are going to use the bilinear transformation shown as follows:

$$s = \frac{2}{Ts} \cdot \frac{z-1}{z+1}$$

Thus we can derive the discrete PID controller with bilinear transformation mapping. Equivalently, the c2dm command in Matlab will help us to convert the continuous-time PID compensator to discrete-time PID compensator by using the "tustin" method in this case. The "tustin" method will use bilinear approximation to convert to discrete time of the derivative.

According to the PID Design Method for the DC Motor explained in section 3.3, $K_p = 100$, $K_i = 200$ and $K_d = 10$ satisfied the design requirement. We will use all of these gains in this example.

After running the matlab comand with these values of Kp, Ki and Kd, we got the unstable response. As we have seen from the plot, the closed-loop response of the system is unstable. Now we have made a proper analysis of system through root locus.From this root-locus plot, we have seen that the denominator of the PID controller has a pole at -1 in the z-plane. We know that if a pole of a system is outside the unit circle, the system will be unstable. This compensated system will always be unstable for any positive gain because there are an even number of poles and zeroes to the right of the pole at -1. Therefore that pole will always move to the left and outside the unit circle. The pole at -1 comes from the compensator, and we can change its location by changing the compensator design. We choose it to cancel the zero at -0.62. This will make the system stable for at least some gains. Furthermore we have choose an appropriate gain from the root locus plot to satisfy the design requirements using rlocfind. 3.6.2 m-file gives the complete modified m-file.

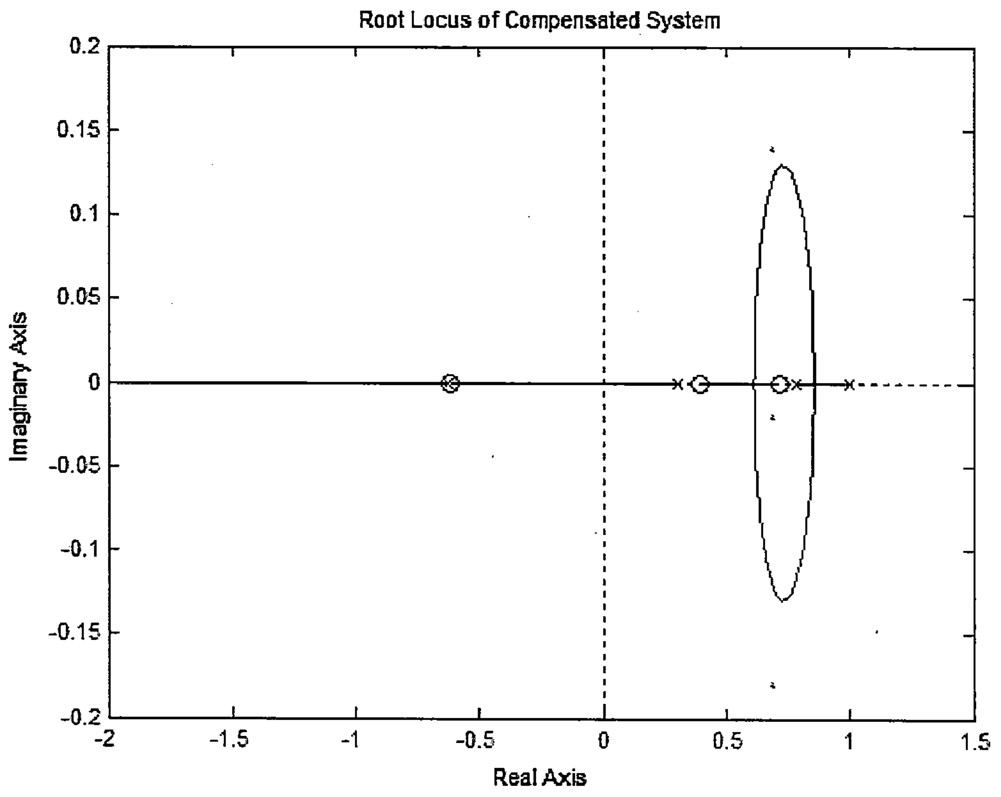

Fig 3.6.2 Root Locus of Compensated System

The closed loop response of digital control dc motor is shown in figure 3.6.2.

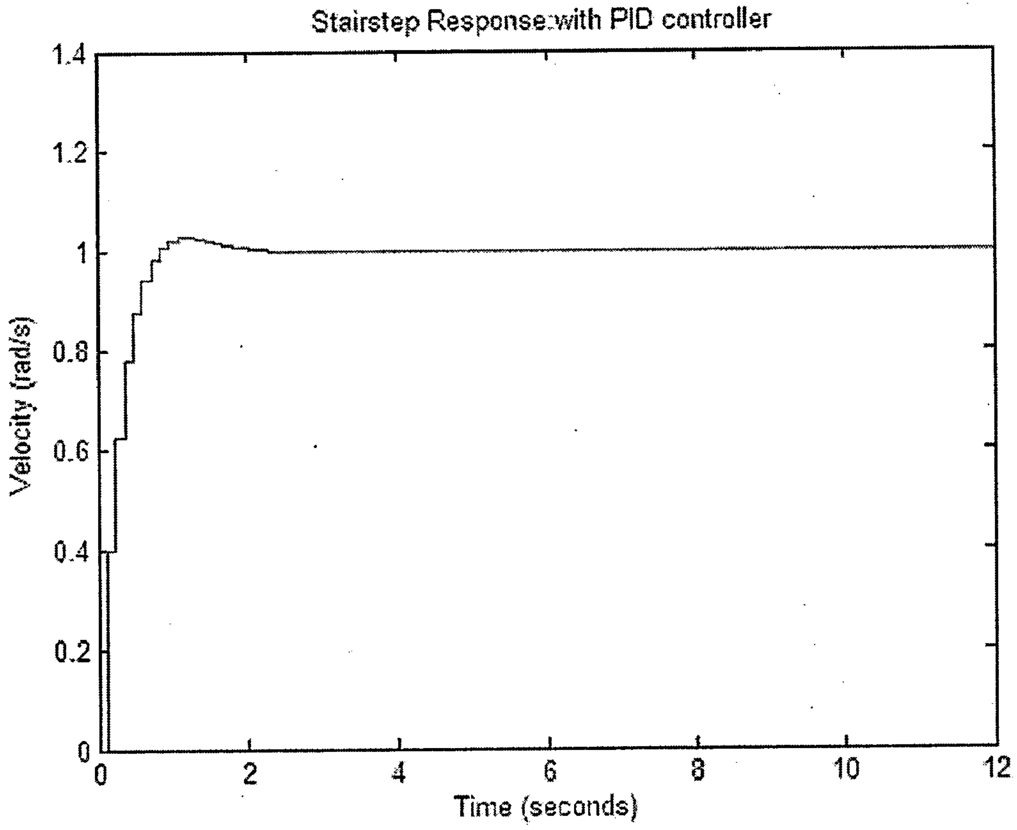

Fig 3.6.3 Closed Loop Step Response of Compensated System

The plot shows that the settling time is less than 2 seconds and the percent overshoot is around 3%. In addition, the steady state error is zero.

4. GENETIC ALGORITHM

4.1 Introduction [5]

This idea appears first in 1967 in J. D. Bagley's thesis "The Behavior of Adaptive Systems Which Employ Genetic and Correlative Algorithms". The theory and applicability was then strongly influenced by J. H. Holland, who can be considered as the pioneer of genetic algorithms.

The world as we see it today, with its variety of different creatures, its individuals highly adapted to their environment, with its ecological balance (under the optimistic assumption that there is still one), is the product of a three billion years experiment we call evolution, a process based on sexual and asexual reproduction, natural selection, mutation, and so on. If we look inside, the complexity and adaptability of today's creatures has been achieved by refining and combining the genetic material over a long period of time.

Genetic Algorithms are simulations of evolution, of what kind ever. In most cases, however, genetic algorithms are nothing else than probabilistic optimization methods which are based on the principles of evolution. It can also be said as genetic algorithms are optimization methods.

Genetics is very much related to genetic algoritms. The table shown below gives a list of different expressions, which are common in genetics, along with their equivalent in the framework of GAs:

Natural Evolution	Genetic Algorithm
genotype	coded string
phenotype	uncoded point
chromosome	string
gene	string position
allele	value at a certain position
fitness	objective function value

4.2 Algorithm

```
    t: = 0;
    Compute initial population Bo;
  WHILE stopping condition not fulfilled DO
  BEGIN
    select individuals for reproduction;
    create offsprings by crossing individuals;
    eventually mutate some individuals;
    compute new generation
  END
```

As obvious from the above algorithm, the transition from one generation to the next consists of four basic components:

Selection: Mechanism for selecting individuals (strings) for reproduction according to their fitness (objective function value).

Crossover: Method of merging the genetic information of two individuals; if the coding is chosen properly, two good parents produce good children.

Mutation: In real evolution, the genetic material can by changed randomly by erroneous reproduction or other deformations of genes, e.g. by gamma radiation. In genetic algorithms, mutation can be realized as a random deformation of the strings with a certain probability. The positive effect is preservation of genetic diversity and, as an effect, that local maxima can be avoided.

Sampling: Procedure which computes a new generation from the previous one and its offsprings.

The flow chart of Genetic Algorithms is shown below [6]. The flow chart clearly explains the procedure of optimization. In the beginning suppose the nth generation is intialized it goes under evaluation, if it satisfy the desired value the algorithm stops at that point. But if it do not find the apropriate value it goes under various process like reproduction, cross over, mutation. And finaly it gives (n + 1)th generation. Simillarly algoritm repeats it self.

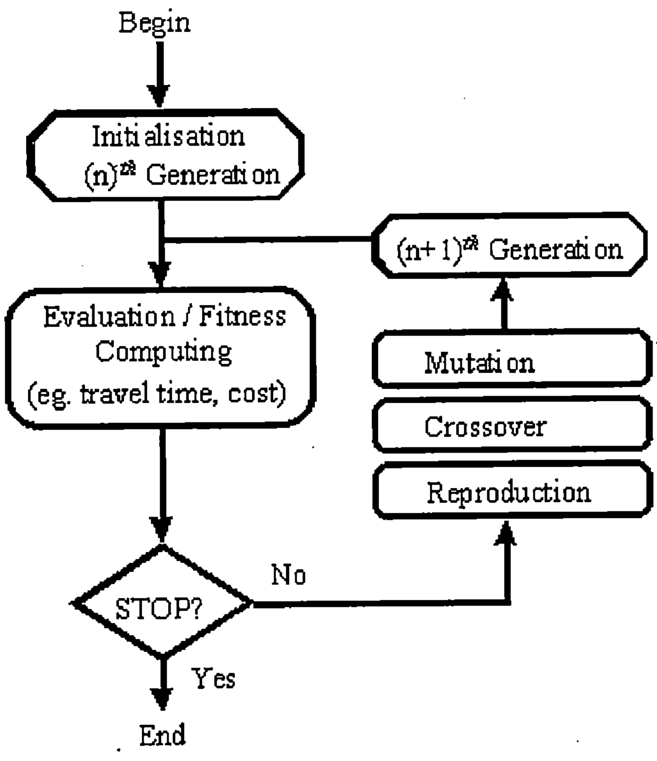

Fig 4.2.1 Genetic Algorithm Structure

Compared with traditional continuous optimization methods, such as Newton or gradient descent methods, we can state the following significant differences:

1. GAs manipulate coded versions of the problem parameters instead of the parameters themselves.

2. While almost all conventional methods search from a single point, GAs always operate on a whole population of points (strings). This contributes much to the robustness of genetic algorithms. It improves the chance of reaching the global optimum and, vice versa, reduces the risk of becoming trapped in a local stationary point.

3. Normal genetic algorithms do not use any auxiliary information about the objective function value such as derivatives. Therefore, they can be applied to any kind of continuous or discrete optimization problem. The only thing to be done is to specify a meaningful decoding function.

4. GAs use probabilistic transition operators while conventional methods for continuous optimization apply deterministic transition operators. More specifically, the way a new generation is computed from the actual one has some random components.

4.3 Adaptive Genetic Algorithms

Adaptive genetic algorithms are GAs whose parameters, such as the population size, the crossing over probability, or the mutation probability are varied while the GA is running. A simple variant could be the following: The mutation rate is changed according to changes in the population; the longer the population does not improve, the higher the mutation rate is chosen. Vice versa, it is decreased again as soon as an improvement of the population occurs.

4.4 Hybrid Genetic Algorithms

As they use the fitness function only in the selection step, genetic algorithms are blind optimizers which do not use any auxiliary information such as derivatives or other specific knowledge about the special structure of the objective function. If there is such knowledge, however, it is unwise and inefficient not to make use of it. Several investigations have shown that a lot of synergism lies in the combination of genetic algorithms and conventional methods.

The basic idea is to divide the optimization task into two complementary parts. The coarse, global optimization is done by the GA while local refinement is done by the conventional method (e.g. gradient-based, hill climbing, greedy algorithm, simulated annealing, etc.). A number of variants is reasonable:

1. The GA performs coarse search first. After the GA is completed, local refinement is done.

2. The local method is integrated in the GA. For instance, every K generations, the population is doped with a locally optimal individual.

3. Both methods run in parallel: All individuals are continuously used as initial values for the local method. The locally optimized individuals are re-implanted into the current generation.

4.5 Self-Organizing Genetic Algorithms

As already mentioned, the reproduction methods and the representations of the genetic material were adapted through the billions of years of evolution. Many of these adaptations were able to increase the speed of adaptation of the individuals. We have seen several times that the choice of the coding method and the genetic operators is crucial for the convergence of a GA. Therefore, it is promising not to encode only the raw genetic information, but also some additional information, for example, parameters of the coding function or the genetic operators. If this is done properly, the GA could find its own optimal way for representing and manipulating data automatically.

4.6 Tuning of Fuzzy Systems Using Genetic Algorithms

There are two concepts within fuzzy logic which play a central role in its applications. The first is that of a linguistic variable, that is, a variable whose values are words or sentences in a natural or synthetic language. The other is that of a fuzzy if-then rule in which the antecedent and consequent are propositions containing linguistic variables. The essential function served by linguistic variables is that of granulation of variables and their dependencies. In effect, the use of linguistic variables and fuzzy if-then rules results through granulation—in soft data compression which exploits the tolerance for imprecision and uncertainty. In this respect, fuzzy logic mimics the crucial ability of the human mind to summarize data and focus on decision-relevant information.

Two important components which have to be specified in order to make a fuzzy system work, the rules and the fuzzy sets. The scaling factor of fuzzy control can tuned through genetic algorithms.

5. TUNING OF FUZZY CONTROL AND DESIGNING FUZZY MODEL OF DC MOTOR DRIVE

Conventional control system design depends upon the development of a mathematical description of the system's behavior. This usually involves assumptions being made in relation to the system dynamics and any non-linear behavior that may occur. In cases where assumptions in respect of non-linear behavior cannot be made, the need to describe mathematically, ever increasing complexity becomes difficult and perhaps infeasible.

Fuzzy logic is the application of logic to imprecision and has found application in control system design in the form of Fuzzy Logic Controllers (FLCs). Fuzzy logic controllers facilitate the application of human expert knowledge, gained through experience, intuition or experimentation, to a control problem. Such expert knowledge of a system's behavior and the necessary intervention required to adequately control that behavior is described using imprecise term known as "linguistic variables". The imprecision of linguistic variables reflects the nature of human observation and judgment of objects and events within our environment, and there use in FLCs thus allows the mapping of heuristic, system-related information to actions observed to provide adequate system control. In this way, FLCs obviate the need for complex mathematical descriptions of non-linear behavior to the nth degree and thus offer an alternative method of system control.

5.1 Structure of a fuzzy controller [7]

The principal structure of a fuzzy controller is shown in figure 5.1. It consists of following components:

5.1.1 Fuzzification Module

The fuzzification module performe the following functions:

1. FM - F1: Performs a scale transformation (i.e., an input normalization) which maps the physical values of the curren process state variables into a normalized universe of discourse.
2. FM – F2: Perform the so-called fuzzification which converts a point-wise (crisp), current value of a process state variable into a fuzzy set, in order to make it

compatible with the fuzzy set representation of the process state variable in the rule-antecedent.

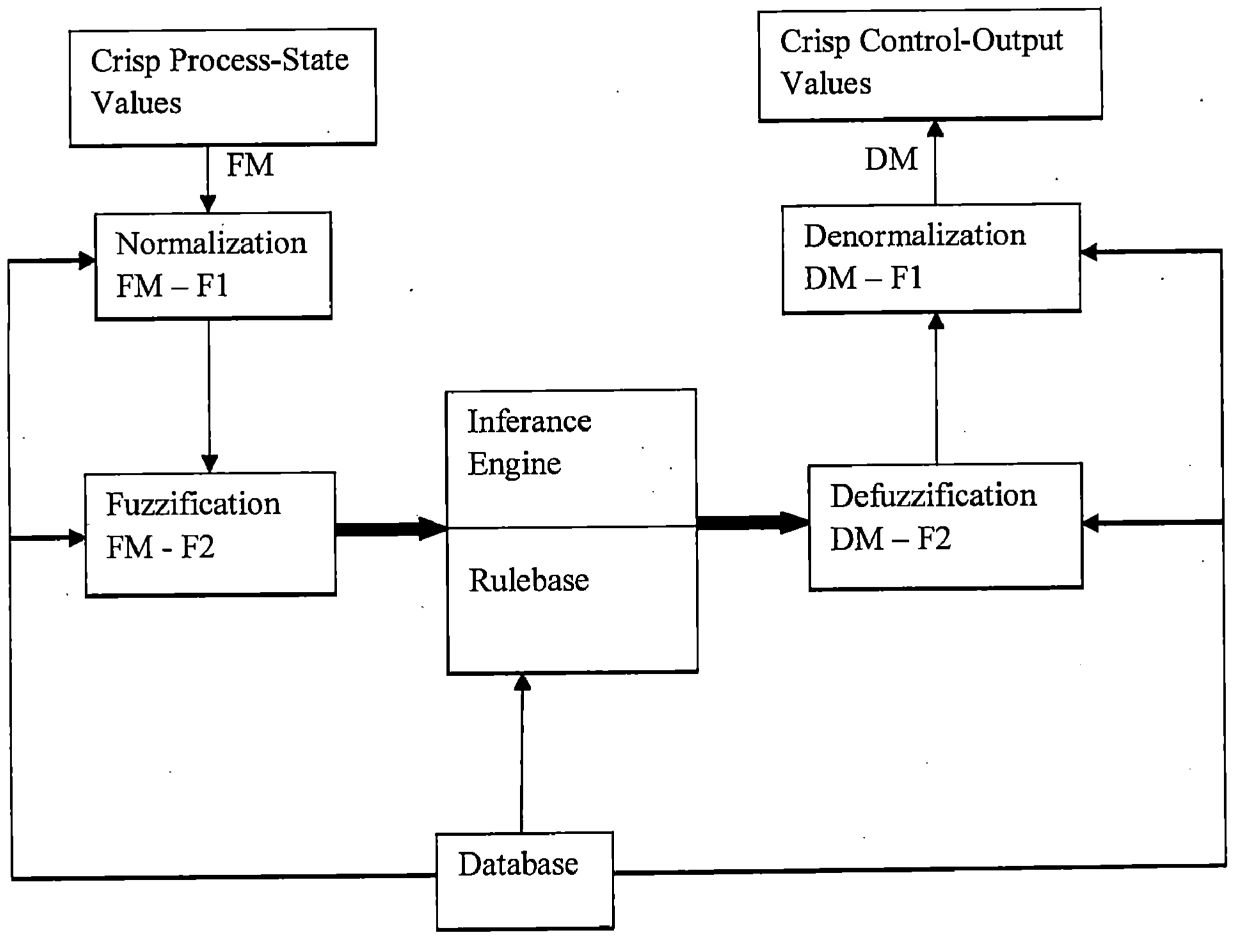

Fig 5.1 Structure of Fuzzy Control

The design parameter of fuzzification module is choice of fuzzification strategy.

5.1.2 Knowledge Base:

The knowledge base of a fuzzy controller consist of data base and rule base.

a. **Data Base**

The basic function of the data base is to provide the necessary information for the proper functioning of the fuzzification module, the rule base, and the defuzzification module. This information includes:

1. Fuzzy sets (membership functions) representing the meaning of the linguistic values of the process state and contorl output variables.
2. Physical domains and their normalized counterparts together with the normalization/denomalization (scaling) factors.

Design parameters of the data base include, choise of the membership function(M.F) and the choice of scaling factors.

b. Rule Base

The basic function of the rule base is to represent in a structured way the control policy of an experienced process operator in the form of a set of production rules such as

if (process) then (control output)

5.1.3 Inference Engine

The inference mechanism has two basic tasks: (1) determining the extent to which each rule is relevant to the current situation as characterized by the inputs u_i, where i = 1, 2, ...n (we call this task "matching"); and (2) drawing conclusions using the current inputs *ui* and the information in the rule-base (we call this task an "inference step"). The inference engine or rule firing can be of two basic types:

- Composition based inference: In this case, the fuzzy relations representing the meaning of each individual rule are aggregated into one fuzzy relation describing the meaning of the overall set of rules. The inference or firing with this fuzzy relation is performed via the operation composition between the fuzzified crisp input and the fuzzy relation representing the meaning of the overall set of value. As a result of the composition one obtains the fuzzy set describing the fuzzy value of the overall control output.
- Individual rule based inference: In this case, first each single rule is fired. This firing can be simply described by: (1) computing the degree of match between the crisp

input and the fuzzy sets describing the meaning of the antecedent and (2) "cliping" the fuzzy sets describing the meaning of rule consequent to the degree to the rule antecedent has been matched by the crisp input. Finally the clipped values for the control output of each rule are aggregated, thus forming the value of the overall control output.

5.1.4 Defuzzification

The final process of the FLC is to aggregate the fuzzy sets resulting from the inference mechanism to produce a decision (i.e. crisp output), which is the "most certain" in respect of the current system behavior.

A number of methods can be used for defuzzification (e.g. center-average, mean-of maxima), however the most commonly used method is the equation for computation of center-of-gravity (COG), or centroid, which ensures a smooth control action but which requires more complex calculations particularly for non-linear MFs.

5.2 Assumptions and Constraints in Designing a Fuzzy Logic Controller [8]

To apply the Fuzzy Logic Controller to various control engineering problems, certain properties of the system are exploited so that the design of the controller can be made easier. As the systems used are symmetrical, it is assumed that symmetrical membership functions about the y-axis will provide a valid controller. A symmetrical rule-base is also assumed.

Other constraints are also introduced to the design of the FLC:

- All universes of discourses are normalized to lie between –1 and 1 with scaling factors external to the FLC used to give appropriate values to the variables.
- It is assumed that the first and last membership functions have their apexes at –1 and 1 respectively. This can be justified by the fact that changing the external scaling would have similar effect to changing these positions.
- Mostly triangular membership functions are to be used.

- The number of fuzzy sets is constrained to be an odd integer greater than unity. In combination with the symmetry requirement, this means that the central membership function for all variables will have its apex at zero.
- The base vertices of membership functions are coincident with the apex of the adjacent membership functions. This ensures that the value of any input variable is a member of at most two fuzzy sets, which is an intuitively sensible situation. It also ensures that when a variable's membership of any set is certain, i.e. unity, it is a member of no other sets.

5.3 Tuning of Fuzzy Controller via Scaling Universes of Discourse [8]

As we have explained in our privious section that the scaling factor in fuzzy control behaves same as the gains in normal controller. In this section we will exaplain the tuning of fuzzy scalling factor and it effect on the syatem.

We have taken an example of inverted pendulam. As the of fuzzy controller for inverted pendulam is designed for balancing it over moving cart.

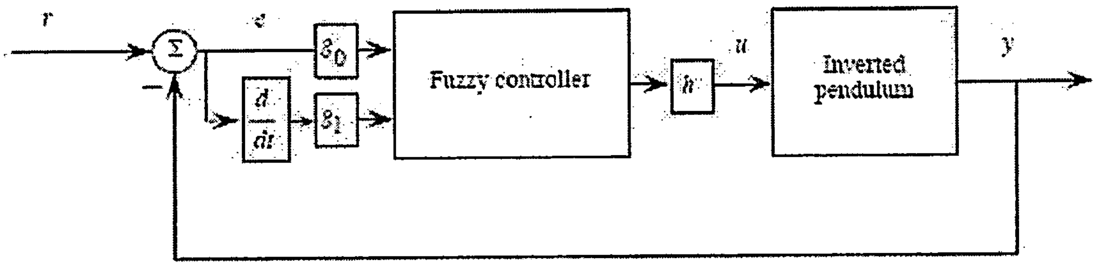

Fig 5.3.1 Fuzzy Controller for Inverted Pendulum with Scaling Gains g_0, g_1, and h.

In the figure 5.3.1, r is the refrence angle and y is the output, u is the controlled output g_0, g_1 and h is the scaling factors of the fuzzy controller. We have tried to explain the effect of scaling factor over fuzzy control.

5.3.1 Input Scaling Gains

First, consider the effect of the input scaling gains g_0 and g_1. We can actually achieve the same effect as scaling via g_1 by simply changing the labeling of the $\frac{d}{dt}e(t)$ axis for the membership functions of that input. The case where $g_0 = g_1 = h = 1.0$ corresponds to our original choice for the membership functions in Figure 5.3.2.

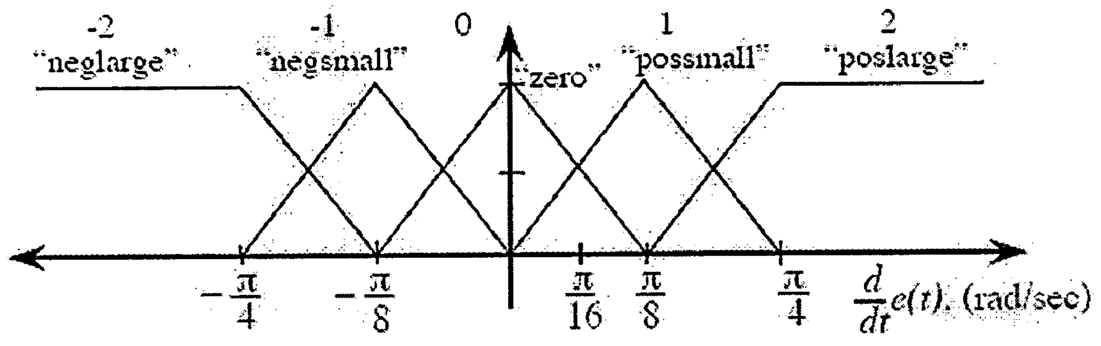

Fig 5.3.2 Membership functions for an inverted pendulum for $\frac{d}{dt}e(t)$

The choice of $g_1 = 0.1$ as a scaling gain for the fuzzy controller with these membership functions is equivalent to having the membership functions shown in Figure 5.3.3 with a scaling gain of $g_1 = 1$. We see that the choice of a scaling gain g_1 results in scaling the horizontal axis of the membership functions by $1/g_1$.

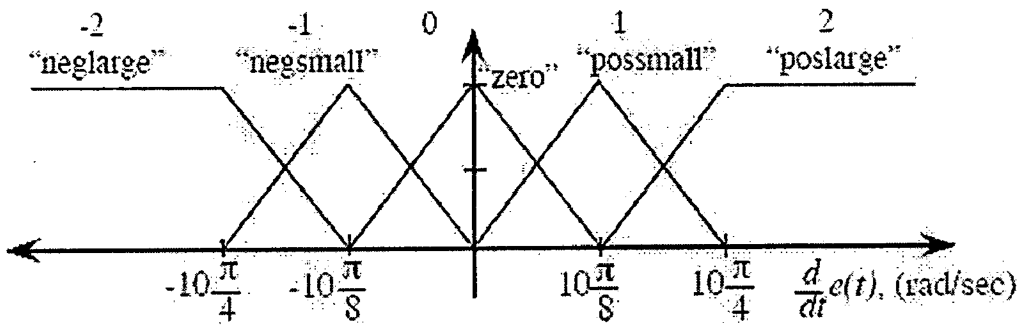

Fig 5.3.3 Scaled Membership Function for $\frac{d}{dt}e(t)$

Generally, the scaling gain g_1 has the following effects:

- If $g_1 = 1$, there is no effect on the membership functions.

- If $g_1 < 1$, the membership functions are uniformly "spread out" by a factor of $1/ g_1$ (we can see each number on the horizontal axis of Figure 5.3.2 is multiplyed by 10 produces a Figure 5.3.3).
- If $g_1 > 1$, the membership functions are uniformly "contracted".

The expansion and contraction of the horizontal axes by the input scaling gains is sometimes described as similar to how an accordion operates, especially for triangular membership functions. Notice that the membership functions for the other input to the fuzzy controller will be affected in a similar way by the gain g_0. Similar statements can be made about all the other membership functions and their associated linguistic values. Overall, we see that the input scaling factors have an inverse relationship in terms of their ultimate effect on scaling (larger g_1 that is greater than 1 corresponds to changing the meaning of the linguistics so that they quantify smaller numbers). While such an inverse relationship exists for the input scaling gains, just the opposite effect is seen for the output scaling gains.

5.3.2 Output Scaling Gain

Similarly the effect of scaling factor can be explained as:

- If $h = 1$, there is no effect on the output membership functions.
- If $h < 1$, there is the effect of contracting the output membership functions and hence making the meaning of their associated linguistics quantify smaller numbers.
- If $h > 1$, there is the effect of spreading out the output membership functions and hence making the meaning of their associated linguistics quantify larger numbers.

There is a proportional effect between the scaling gain h and the output membership functions. This can be explained form the two figures shown below. Figure 5.3.4 shows the normal membership function. And the figure 5.3.5 shows the scaled output membership function. The figure clearly shows the effect of scaling gain h on the spacing of the output membership functions.

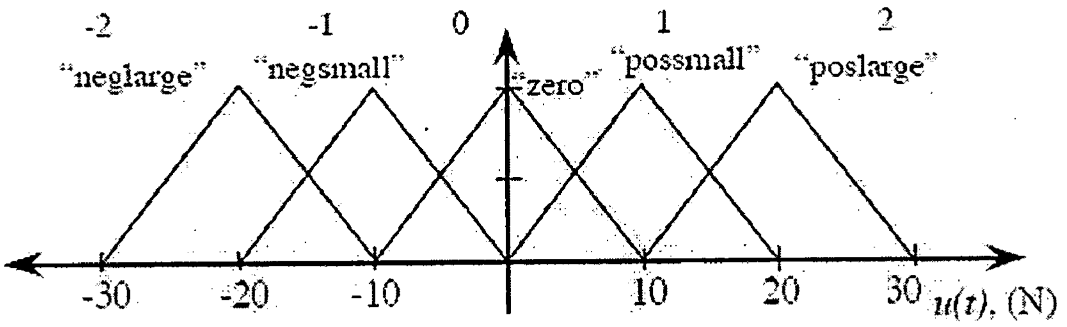

Fig 5.3.4 Normal Membership Function

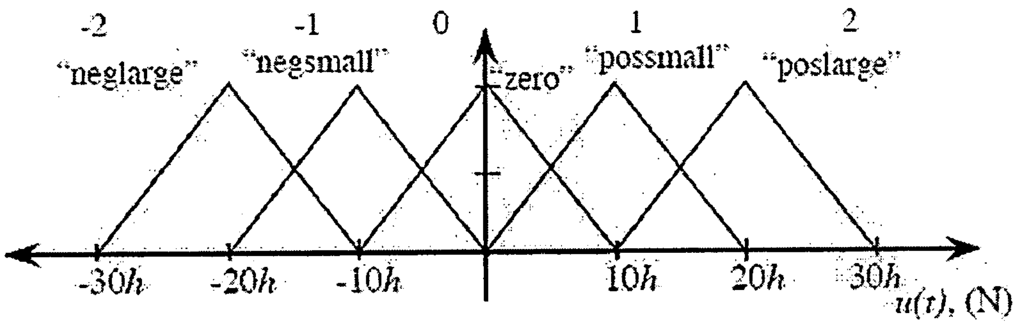

Fig 5.3 Output Membership Functions Scaling Gain *h*

5.4 Self Tuning of Fuzzy Control

Most of the real world processes that require automatic control are non linear in nature. That is , their parameter values alter as the operating point changes, over time, or both. As conventional control schemes are linear , a controller can only be tunned to give good performance at a particular point or for a limited period of time . The controller needs to tunned if the operating point Changes ,or retuned periodically if the process changes with time. This necessity to retune has driven the need for adaptive controllers that can automatically retune themselves to match the current process characteristics. An excellent introduction to "conventional" adaptive control systems is by Astrom.

There is still contention as to what exactly constitutes an adaptive controller, and there is no consensus on the terminology to use in describing adaptive controllers. Adaptive controllers generally contain two extra components on top of the standard controller itself. The first is a "process monitor" that detects changes in the process characteristics. It is usually in one of two forms:

- A performance measure that assesses how well the controller is controlling,
- A parameter estimator that constantly updates a model of the process.

The second component is the adaptation mechanism itself. It uses information passed to it by the process monitor to update the controller parameters and so adapts the controller to the changing process characteristics. Adaptive controllers can be classified as performance-adaptive or parameter-adaptive depending on which type of process monitor they employ. Fuzzy controller contains a number of sets of parameters that can be altered to modify the controller performance. These are:

1. The scaling factor for each variable,
2. The fuzzy set representing the meaning of linguistic values,
3. The if-then rules.

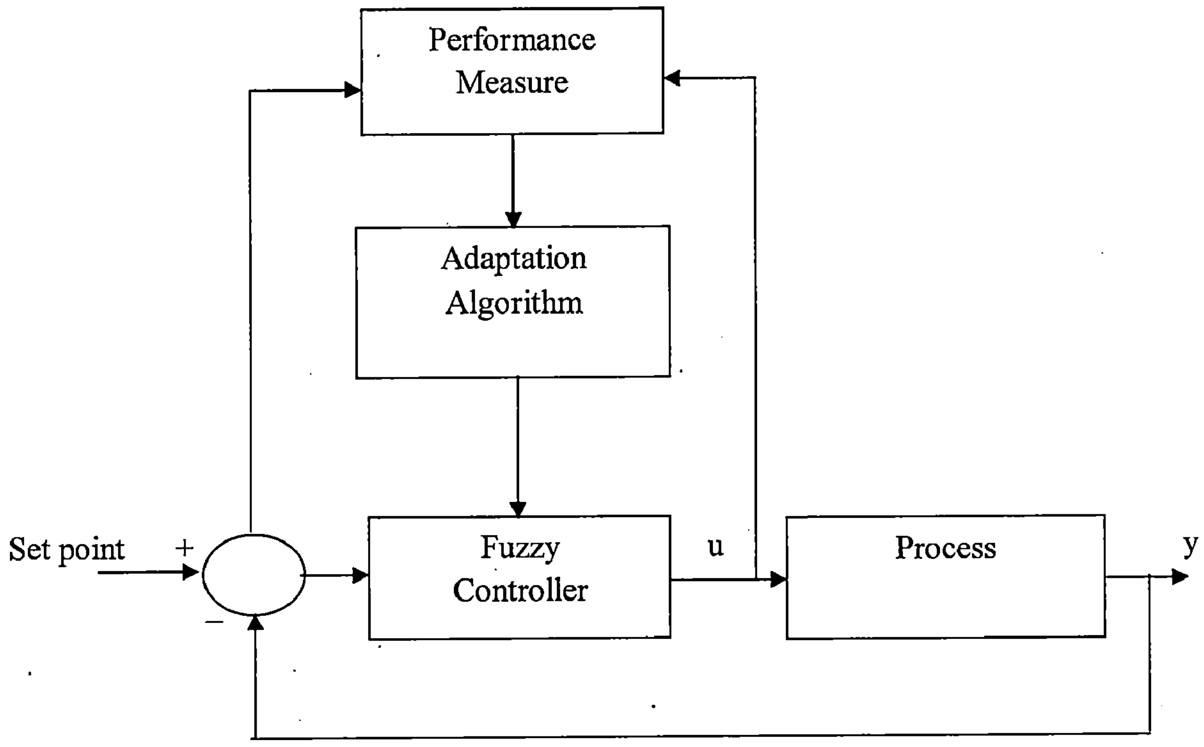

Fig 5.4.1 Performance Adaptive Fuzzy Controller

The introduction of the Takagi-Sugeno fuzzy model initiated research on the structure and parameter identification for fuzzy systems. Fuzzy systems have been utilized to generate nonlinear input-output functions of complex plants. Evolutionary steps that have shaped fuzzy control theory in forms similar to those of conventional control theory. This latter approach to fuzzy modelling and control involves off-line parametric estimation and stabilization of the identified model within the context of classical control tools. The control problem in the fuzzy model based methodology is associated with the requirement of well-known system parameters. The case of changing system parameters due to operational variations in the controlled process raises the question of adaptive design schemes. Fuzzy adaptation in tuning for some types of control systems has been investigated. The treatment of the concurrent problem of identification and fuzzy control has been explored. In this article, we approach the solution of the fuzzy modelling and control problem simultaneously. A fuzzy model identifier is employed to approximate the input-output description of the physical plant online. A fuzzy controller is utilized to implement the control action, which is calculated on the basis of the parameter estimates and according to the certainty equivalence principle. Since the fuzzy model is a weighted superposition of linear systems, the design specifications of the closed-loop can be met through the imposition of analogous criteria on the component systems. Hence, linear control techniques are reckoned suitable to achieve the design objective. The architecture of the overall scheme emanates from the self-tuning control structure.

5.5 DC Drive Fuzzy Model[9]

This section deals with the methodology of designing a complete fuzzy model of a DC drive based on a suitable database of measured input–output values. This methodology covers the entire range of possible drive inputs, without requiring any information about the drive structure and parameters.

Methods of applying fuzzy sets in various applications in the field of electrical drives have recently become a frequently appearing issue in specialized literature. One of the tasks involved is the development of corresponding models of the particular drives. The solution of this task can be based on analytical knowledge of the given drive type, however this approach does not introduce any advantages against conventional analytical models, neither does it exploit fuzzy system properties. The other option is to attempt setting up a model of the drive based only on the knowledge of the relevant drive inputs and outputs, without prior knowledge of the drive structure or parameters. This method deals with the latter approach; it provides a description of a DC drive fuzzy model design procedure that is based on a suitable database of measured data, without anticipation of any further information about the given drive. This algorithm is verified by a DC motor drive simulation.

An electrical drive presents a dynamic system that can be generally described in state space by the following equations:

$$\dot{x} = A(x,t)X + B(x,t)u \qquad 5.5.1$$

$$y = Cx \qquad 5.5.2$$

The aim of the investigation is to set up a fuzzy model of the drive on basis of the measured database of inputs and their corresponding output values.

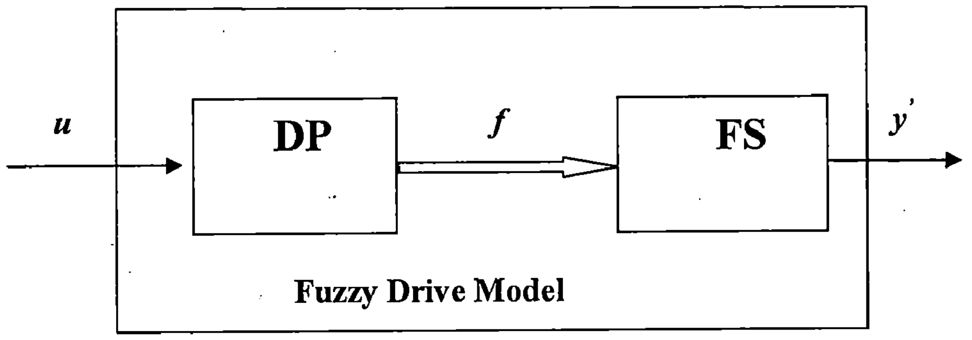

Fig 5.5.1 Electrical Drive Fuzzy Model Structure

In order to enable modelling of the drive in according to the equation (1) and (2), it is necessary to complement the fuzzy system (FS) by a dynamic part (DP). A complete fuzzy model of the drive will then look as shown in Fig. 5.5.1.

The principal requirements that must be observed when designing a fuzzy model of a drive in accordance with Fig. 5.5.1 are as follows:

1. For the measured input points of the database (and hence also input vectors f for the FS), the y outputs must be equally accurate.

2. There must be full coverage of the entire scope of possible drive inputs by rules accord. Otherwise it is possible that with an uncovered input state f, the FS output would present a non-defined or incorrect value.

3. Consistency of the measured values database, or of the progression of vectors f, corresponding to the measured values, must be ensured. With an inconsistent database, there would be various outputs y corresponding to "approximate" vectors f. As vectors f are in fact

inputs into the FS, this would result in inconsistent rules and hence in the principal non-feasibility of the FS.

The quality of meeting the above requirements depends mainly upon:

- Proper selection of the structure of the dynamic part (DP) of the electrical drive model
- Properly selected measurements regarding the drive input signal u
- Proper forms of membership functions at the fuzzification of the FS inputs.

5.6 Procedure of Designing an Electrical Drive Fuzzy Model

For a constant input signal value *Ui*, the drive output always stabilizes at the same value *Yi* during a period of time shorter than *T*max. For example, at a specific voltage of the DC rotor the angular speed always stabilizes at the same value; at a specific asynchronous motor stator frequency the angular speed always stabilizes at the same value, etc.

A complete fuzzy model of the drive can be obtained by integration of all the transition trajectories that originate from all the transitions between all possible drive input values. If we divide the range of the input variable u into n number of levels, we will need to measure and then model n (n - 1) trajectories.

Modelling an Individual Trajectory

The block scheme for creating a database of values for a single trajectory of a drive fuzzy model is shown in figure 5.6.1.

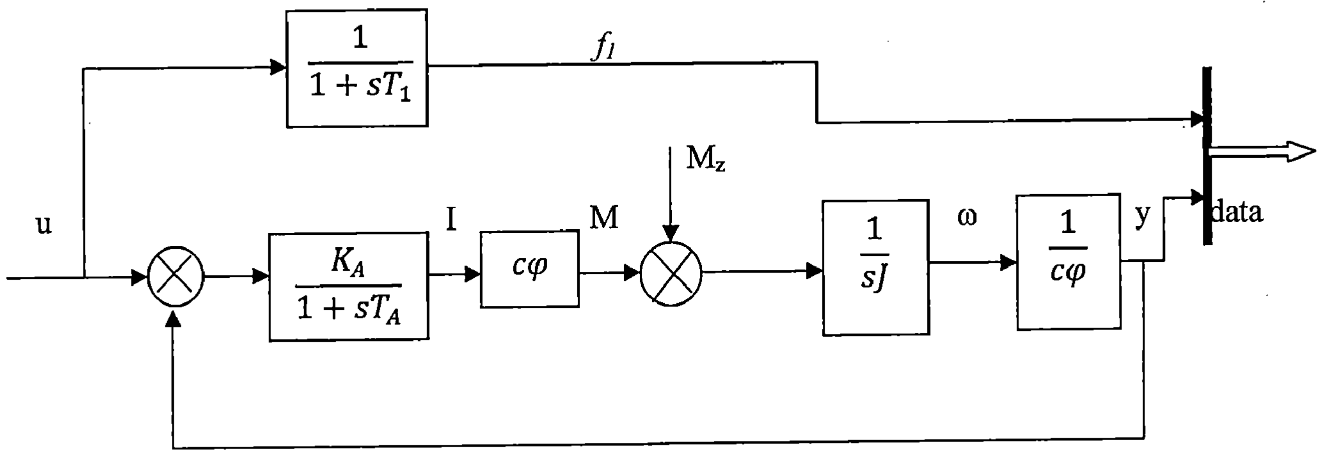

Fig 5.6.1 Block Diagram for Creating Database

Figure 5.6.1 shows the block diagram for the creation of data. The parameters shown in the blocks are:

K_A – armature gain, "4.55"

T_A – armature time constant, "0.05"

$c\varphi$ - motor constant, "0.333"

J – moment of inertia, "0.382"

T_1 – time constant of first order inertia system, "0.2"

M – motor torque,

Mz – load torque,

ω - angular speed,

u – system input,

y – system output.

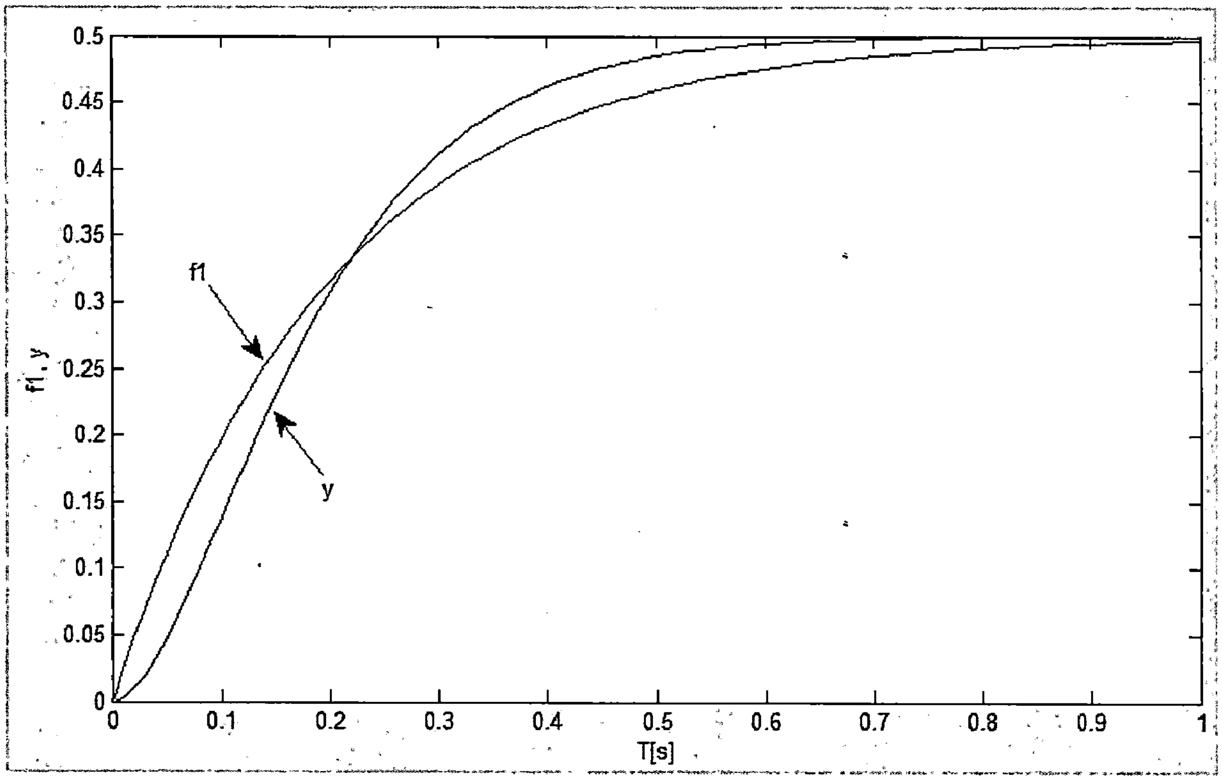

Fig 5.6.2 Time Response of Both with Step Input

The trajectory that denotes the transition from one steady state to another steady state at step change of input u represents a set of successive output y values in time. In order to explicitly distinguish the time intervals and to avoid measurement of absolute time, input u will be transformed through a first order block of inertia with unit amplification to function f_1.

The database will then be made up of points $<f_{1i}, y_i>$ measured in ti time intervals. In order to reduce the number of fuzzy rules necessary for the FS, only several central points (centres) will be selected from the database. This can be done either on basis of the graphical time response of the trajectory.

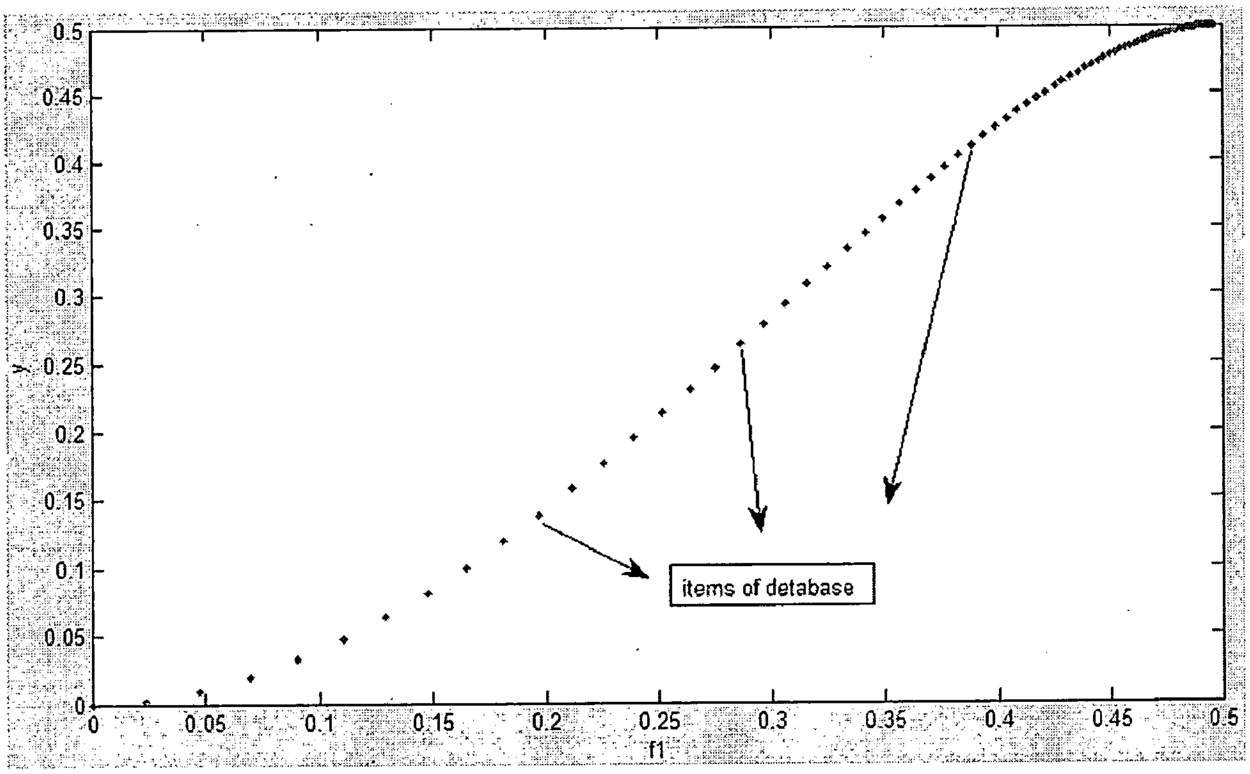

Fig 5.6.3 Selected pairs of points describing the static part of the FS

The mentioned points will represent the centres of individual membership functions, which will be applied in the fuzzification of the input f_1 function for the FS. The width of a particular membership function must not extend further than the immediate neighbour (centre) on either side of the relevant centre. Otherwise the output value y in the given centre would not necessarily be identical with the value measured in the centre, which is in conflict with requirement No. 1 quoted in the preceding Section. Membership functions for neighbouring centres must overlap, as otherwise they would not cover the entire f_1 range, and this would be in conflict with requirement No. 2 quoted in the preceding Section.

We have Consider a model of a separately excited DC motor with step change of input. The database will be created by measurements of values f_1 s and y at constant time intervals $T1$ = 0.1s, as per Figure 5.6.4. The value of $T1$ will be selected according to the transient function of the drive in such a way that the dynamics of f_1 would approximately correspond with the dynamics of the transient function. The time response of both f_1 and y is shown in Figure 5.6.2. Selected pairs of points $[f_1, y]$ describing the static part of the FS model are shown in Figure 5.6.3.

The fuzzy model of the presented drive is shown in Figure 5.5.1. Its dynamic part is represented by first order inertia. The FS static part is created according to Figure 5.6.4, which presents the graphical representation of rules and membership functions for considered FS.

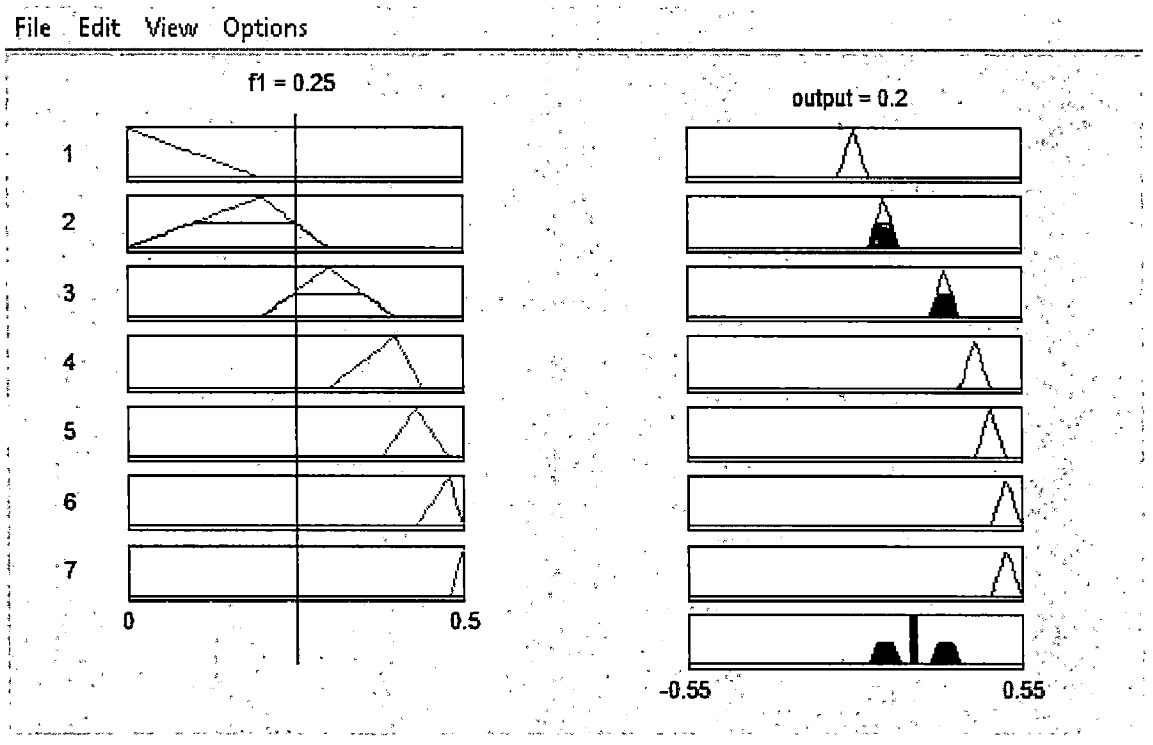

Fig 5.6.4 Graphical Representation of Rules and Membership Functions

Figure 6.5.5 shows the comparison of the real trajectory with its modelled fuzzy substitute: the two are identical in those points, from which the FS part of the fuzzy model was built.

Hence it is possible to select a different f_1 function:

First order inertia system is, however, very simple and easily implemented. For the fuzzification of any f_1, all of the above requirements must be met. Furthermore, it is clear that the more precise fuzzy approximation of the y trajectory is requested, the more fuzzification centres over f_1 have to be selected, which, however, results in a larger number of rules within the FS (one centre = one rule).

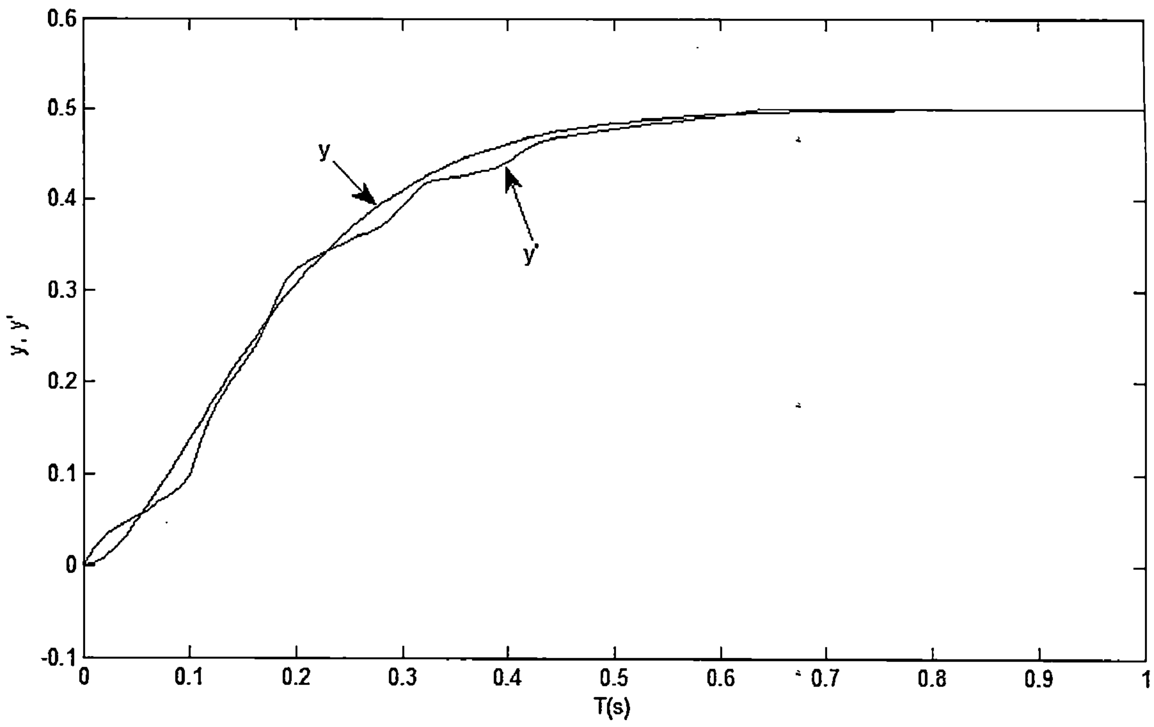

Fig 5.6.5 Comparison of the Real and Modelled Trajectory of a DC Drive Output

The principal properties of a fuzzy model designed in the described manner are as follows:

- It covers the entire state space of the drive.
- Depending on the required degree of precision, the density of dividing u into individual states can be selected.
- For a number of selected input levels n, the number of FS rules will always be pn^2, where p is the number of points selected for an individual trajectory (at complemented trajectories for steady states).
- The dynamic part of the model will be identical for each drive.
- The fuzzification of the nonlinear static part of the FS model is very simple and it ensures coverage of the entire state space of the drive.
- Principal consistence of the measured database values is ensured.
- No knowledge of the drive type, structure or parameters is required.

With regard to the properties outlined above it can be assumed that the presented manner of modelling can be applied to any type of drive.

6. ADAPTIVE FUZZY AND ADAPTIVE NEURO-FUZZY INFERENCE SYSTEM

6.1 Adaptive Fuzzy Controller[7]

Adaptive controller consists of two parts:

1. The process monitor
2. The adaption mechanism

Change in process characteristic can eighter be detected through on line identification of process model, or by assessment of the controlled response of the process.

The most commonly used model in controller design is the single input and single output , linear , first order plus dead time model described by transfer function.

$$\frac{y(s)}{u(s)} = \frac{Kp_e{}^{-t_d s}}{\tau s+1}$$

Where $y(s)$ is the Laplace transform of the process-output,

$u(s)$ is the Laplace transform of the process-input,

Kp is the gain,

t_d is the dead-time, and

τ is the time constant.

Identification of fuzzy process model involves estimation of fuzzy relation, R, from process input output data. R is also called fuzzy relation matrix. Adaptive controllers that use on line identification of a process model as their performance monitor are known as parameter adaptive controllers.

The alternative type of process monitor forms an assessment of controller performance based on readily measured variables. For the regulatory control problem, were the aim is to keep a process state variable at its specified set point. Adaptive controllers that use the measure of controller performance as their performance monitor are known as performance adaptive controllers. Number of performance related variables is overshoot, rise time, settling time, decay, ratio, frequency of oscillating of the transient, integral of square error, gain and phase margin.

6.2 Self Organizing Controllers

This type of controller is developed by Mamdani. Their idea is to identify which rule is responsible for poor performance of the controller, and then to replace these rules with better rules. The performance monitor accesses the controller performance on the basis of the error and change of error of process output variable compared to that of the desired one. This gives the idea of changing the process output variable to achieve the good control. This can be done using simple incremental control of the process that relates changes in process inputs to change the process output. This controller can modify a predefined set of rules, or it can start with no rules at all and learn its control policy as it goes. It is also performance adaptive controller. In this case it is the rules that adjust itself, not the fuzzy set definition, or scaling factor.

The controller is double input, single output type it can be seen from the figure 6.2.1. The error (e) and change in error (Δe) are input, the process input or control output (u).

Performance monitor consist of measuring the performance and regulate the control requirement. That is the sufficient fast approach to return to set point, good damping when close to set point and the measure of tolerance when close to set point. The output of performance monitor is not given as the value of performance, but as a value for correction required at the process output to obtain good performance.

6.2.1Adaptation Algorithm

This input reinforcement is the amount that must be added to the process input, i.e., control output, to compensate for the current poor performance. Dynamics of control output processes are responsible for poor performance. High order process with large time lag will

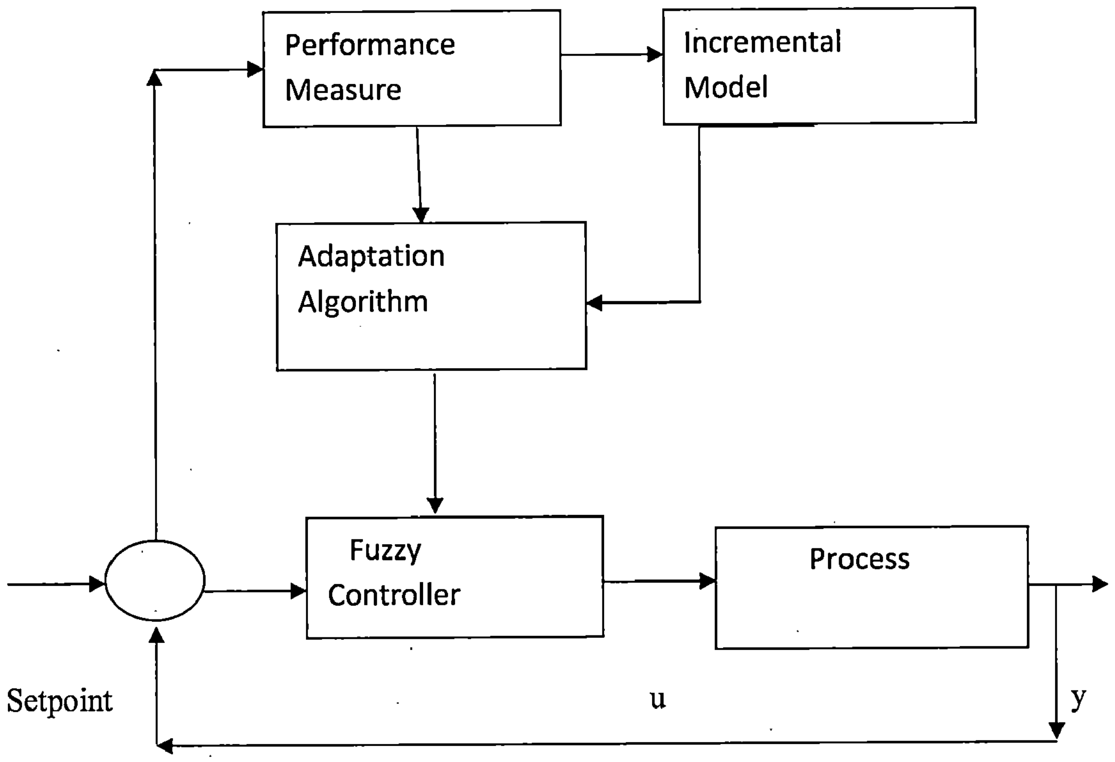

Fig 6:2.1 Self Organized Fuzzy Controller

require control output for long time in the past to be adjusted. Low order process for short time lags will require control output much nearer the present to be adjusted. Correcting the control output in the fuzzy controller means altering the rule consequents of the appropriate rules. Their adaptation mechanism requires the parameter that specifies which past control outputs should be corrected. If delay is small the control action is too close to the present is corrected.

The controller used by us is self organising fuzzy controller, which changes the rules according to the situation. It is a tabular fuzzy which is of sugeno type. The controller used by us is like two fuzzy controllers working together. The values or a rule of one fuzzy controller is adjusted by other. It is a on line tuning technique.

6.3 Adaptive Neuro-Fuzzy Inference System(ANFIS)[10]

ANFIS uses a hybrid learning algorithm to identify the membership function parameters of single-output, Sugeno type fuzzy inference systems (FIS). A combination of least-squares and backpropagation gradient descent methods are used for training FIS membership function parameters to model a given set of input/output data. As we have already seen, fuzzy systems present particular problems to a developer:

- Rules. The if-then rules have to be determined somehow. This is usually done by 'knowledge acquisition' from an expert. It is a time consuming process that is fraught with problems.

- Membership functions. A fuzzy set is fully determined by its membership function. This has to be determined. For example if it's gaussian then what are the parameters?

The ANFIS approach learns the rules and membership functions from data. ANFIS is an *adaptive network*. An adaptive network is network of nodes and directional links. Associated with the network is a learning rule, for example back propagation. It's called adaptive because some, or all, of the nodes have parameters which affect the output of the node. These networks are learning a relationship between inputs and outputs.

Adaptive networks covers a number of different approaches but for our purposes we will investigate in some detail the method proposed by Jang known as ANFIS. The ANFIS architecture is shown in figure 6.3.1. The circular nodes represent nodes that are fixed whereas the square nodes are nodes that have parameters to be learnt.

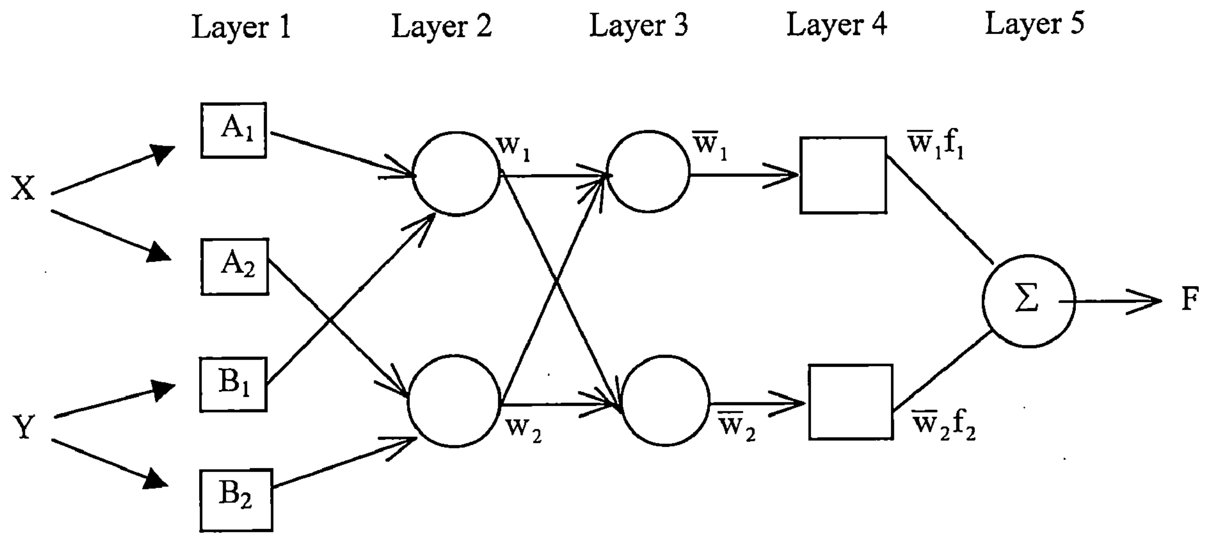

Fig 6.3.1 An ANFIS architecture for a two rule Sugeno system

A Two Rule Sugeno ANFIS has rules of the form:

$$\textit{If } x \textit{ is } A_1 \textit{ and } y \textit{ is } B_1 \quad \textit{THEN } f_1 = p_1 x + q_1 y + r_1$$

$$\textit{If } x \textit{ is } A_2 \textit{ and } y \textit{ is } B_2 \quad \textit{THEN } f_2 = p_2 x + q_2 y + r_2$$

For the training of the network, there is a forward pass and a backward pass. We now look at each layer in turn for the forward pass. The forward pass propagates the input vector through the network layer by layer. In the backward pass, the error is sent back through the network in a similar manner to backpropagation.

Layer 1

The output of each node is:

$$O_{1,i} = \mu_{A_i}(x) \qquad \textit{for } i = 1,2$$

$$O_{1,i} = \mu_{B_{i-2}}(y) \qquad \textit{for } i = 3,4$$

So, the $O_{1,i}(x)$ is essentially the membership grade for x and y.

The membership functions could be anything but for illustration purposes we will use the bell shaped function given by:

$$\mu_A(x) = \frac{1}{1 + \left| \frac{x - c_i}{a_i} \right|^{2b_i}}$$

Where a_i, b_i, c_i are parameters to be learnt. These are the premise parameters.

Layer 2

Every node in this layer is fixed. This is where the t-norm is used to 'AND' the membership grades - for example the product:

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

Layer 3

Layer 3 contains fixed nodes which calculates the ratio of the firing strengths of the rules:

$$O_{3,i} = \overline{w_i} = \frac{w_i}{w_1 + w_2}$$

Layer 4

The nodes in this layer are adaptive and perform the consequent of the rules:

$$O_{4,i} = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i)$$

The parameters in this layer (p_i, q_i, r_i) are to be determined and are referred to as the consequent parameters.

Layer 5

There is a single node here that computes the overall output:

$$O_{5,i} = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

This then is how, typically, the input vector is fed through the network layer by layer. We now consider how the ANFIS learns the premise and consequent parameters for the membership functions and the rules.

There are a number of possible approaches but we will discuss the hybrid learning algorithm proposed by Jang, Sun and Mizutani (Neuro-Fuzzy and Soft Computing, Prentice Hall, 1997) which uses a combination of Steepest Descent and Least Squares Estimation (LSE).

It can be shown that for the network described if the premise parameters are fixed the output is linear in the consequent parameters.

We split the total parameter set into three:

S = set of total parameters

S_1= set of premise (nonlinear) parameters

S_2= set of consequent (linear) parameters

So, ANFIS uses a two pass learning algorithm:

- Forward Pass. Here S_1 is unmodified and S_2 is computed using a LSE algorithm.
- Backward Pass. Here S_2 is unmodified and S_1 is computed using a gradient descent algorithm such as back propagation.

So, the hybrid learning algorithm uses a combination of steepest descent and least squares to adapt the parameters in the adaptive network.

The summary of the process is given below:

The Forward Pass

Present the input vector

Calculate the node outputs layer by layer

Repeat for all data $\rightarrow A$ and y formed

Identify parameters in S_2 using Least Squares

Compute the error measure for each training pair

Backward Pass

Use steepest descent algorithm to update parameters in S_1 (backpropagation)

For given fixed values of S_1 the parameters in S_2 found by this approach are guaranteed to be the global optimum point.

7. SIMULATION RESULTS

7.1 Tuning of Fuzzy Ruler by Graphical Analysis of System Response

7.1.1 Introduction:

In this method, tuning of Fuzzy Rules based on graphical analysis of the system response is proposed. Two fuzzy contollers with triangular membership functions of equal scaling parameters and rule bases but with dissimilar crossover points are considered.First controller has cross points greater than 0.5 and other equal to 0.5.Based on the observed superior response of the first controller,the second one is tuned to yield better results than the earlier one.Comparitive study of firing of rule base of both the controllers are done,which prompted us in designing the simpler controller with reduced rule base to yield better response.In order to study the behaviour of the tuned Fuzzy controller it is employed to a DC Motor.

With refrence to the technique given in the book[11]. The membership functions(MF) with larger support was used. Figure 7.1.1 shown below has membeship functions with cross point more than 0.5. The type of membership function was used in book[11]. As when these kind of membership functions are compered with membership functions with cross point equal to 0.5 as shown in Figure 7.1.2. A better result is given by first one.

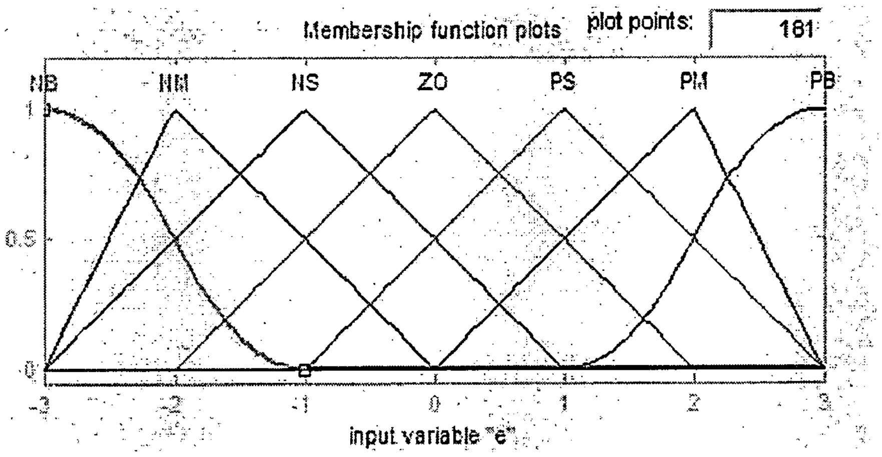

Fig 7.1.1 Membership Function With Cross Point Greater Than 0.5

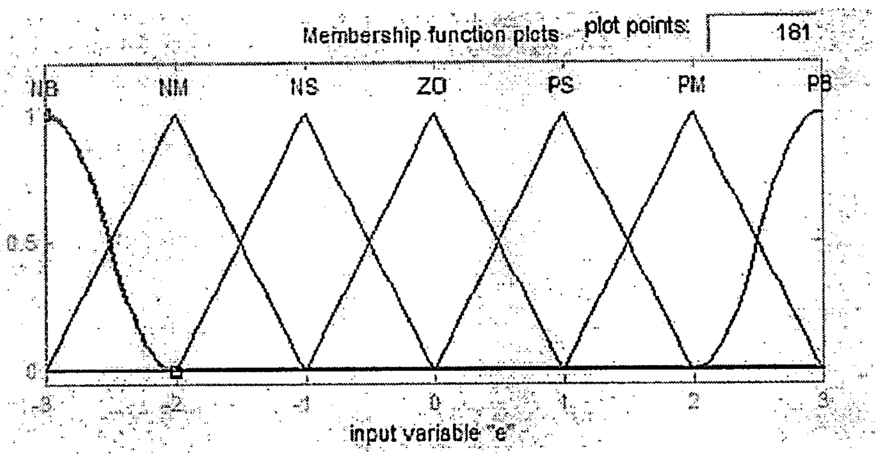

Fig 7.1.2 Membership Function With Cross Point Equal To 0.5.

To study the behavior of these kind of M.F , analysis on these was performed using matlab. A DC shunt motor is used to study the behavior. The motor is armature controlled motor. With the M.F given in this technique, at a time upto 8 to 9 rules get fired. To analyse how these M.F are better than M.F with normal support. A comparission is made in matlab.

This has been compared with fuzzy controller having normal M.F with cross point equal to 0.5 and same rule base. By using these kind of M.F only 3 to 4 rule fire at a time.

Both type of controller is implemented on the transfer function of DC Motor.

$$G_v(s) = \frac{\omega(s)}{V(s)} = \frac{K}{[(R + Ls)(Js + b) + K^2]}$$

The simulation is done on matlab. On compereing these two kind of M.F in matlab simulation we observed, that M.F with spread support(MF with cross point > 0.5) has better response than normal M.F(with cross point = 0.5). It can be seen in figure 7.1.3 given below.

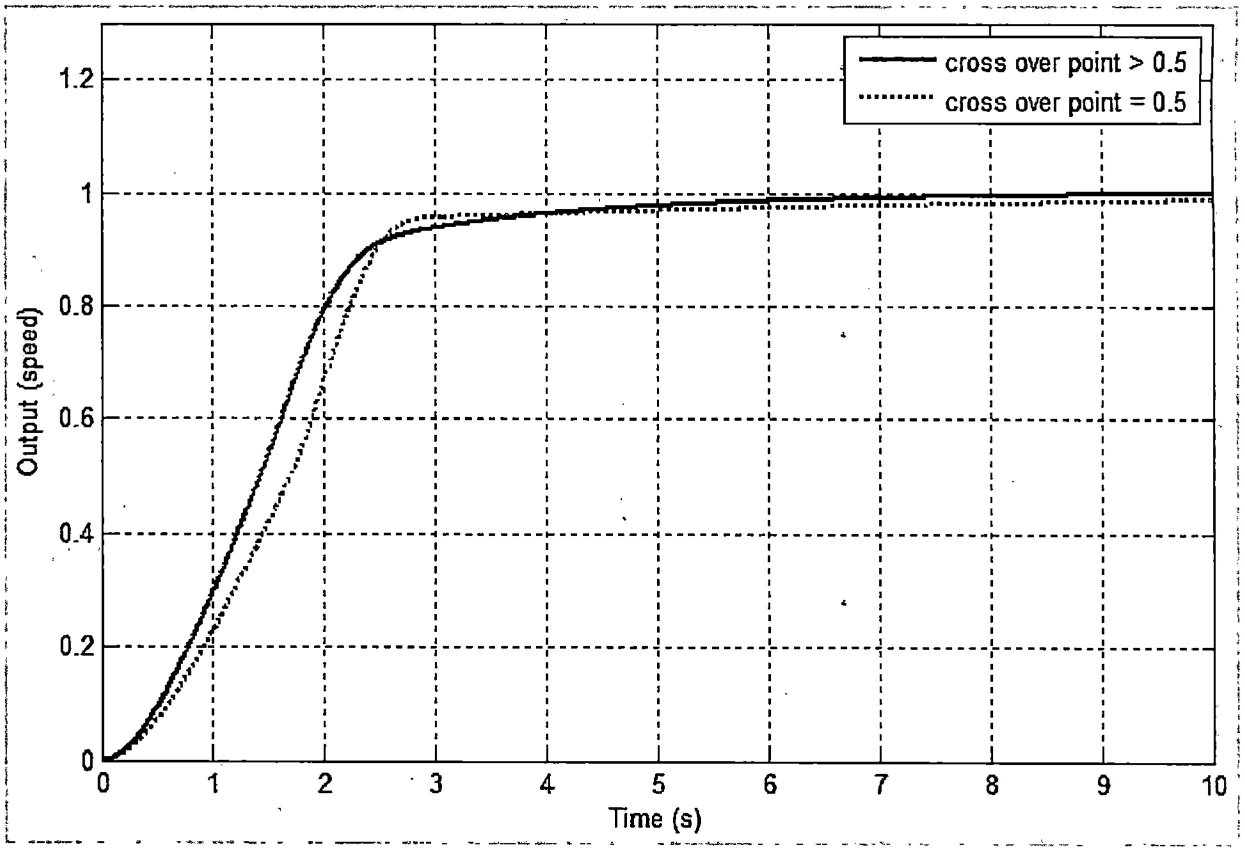

Fig 7.1.3 Characteristic System Response Using Two Type Of Controller.

Allthough this method gives better response, than normal fuzzy M.F. But we can modify the system response by altering some of the rules. Analysis of system response is made. The output of system response is observed closely in excell sheet (Appendix Table 1). This analysis gives us the clear idea about the firing of rules. By analysing system response of error and change of error, and coresponding values of K_p, K_i, K_d. We have come to know exactly which rules are responsible for the system response. By simple modification in these rules we got a controller which is superior than the controller with spread M.F .

Thus we concluded that by keeping normal membership function and by systematic study of system response we can get the desired response by altering some rules.

7.1.2 System Analysis and Alteration of Rules

The whole process of the alteration of rules are described below. The whole analysis is done through excell sheet (Appendix Table 1) and matlab system response.

Alteration of rules are done as

By seing the initial graph of the system in the above Figure 7.1.3 we concluded that the system has low rise time. For this K_p should be made proper.

Initialy the error and derivative of error are in the range of zero. Which can only be observed though the analysis of excell sheet value. So the rules of this region is altered.

From, *e(z) ec(z)* K_p *(z)*

To, *e(Z) ec(Z)* K_p *(PB)*

By increasing K_p from Z to PB the rise time increases, but it has large overshoot and steady state error which can be seen in Figure 7.1.4.

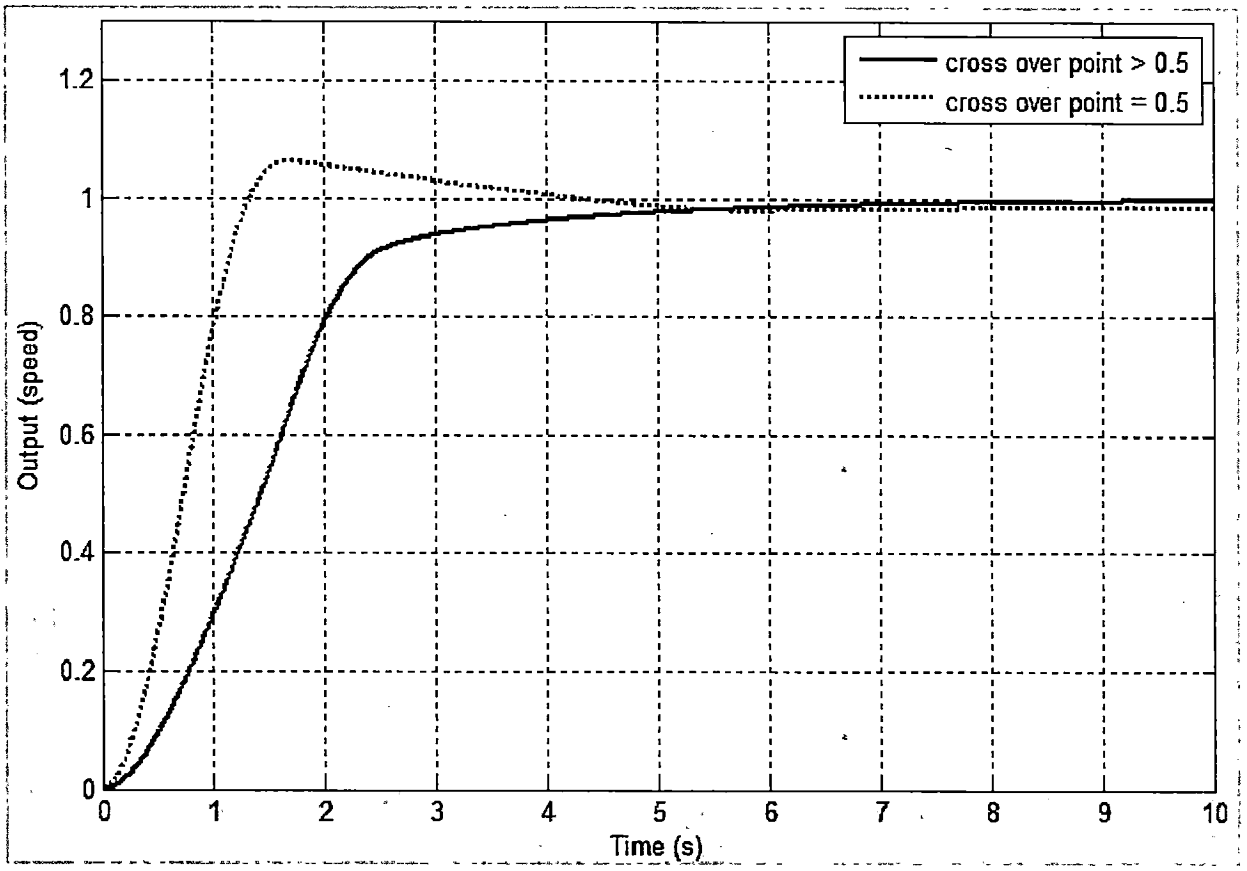

Fig 7.1.4 System Response by Changing K_p

So accordingly K_i and K_d is changed to modify the system response. These two valued are also observed though excell sheet values. As we know that K_d is responsible for the overshoot of system, and K_i is responsible for the steady state error. Hence the rules for K_i and K_d is changed as

From , *e (Z)* *ec (Z)* *K_i (Z)* *K_d (NS)*

To, *e (Z)* *ec (Z)* *K_i (PS)* *K_d (PB)*

by changing these rules, rise time of system remaines same, but oveshoot decreases and steady state error become zero. As it can be clearly seen in the graphical response of the system shown below in Figure 7.1.5.

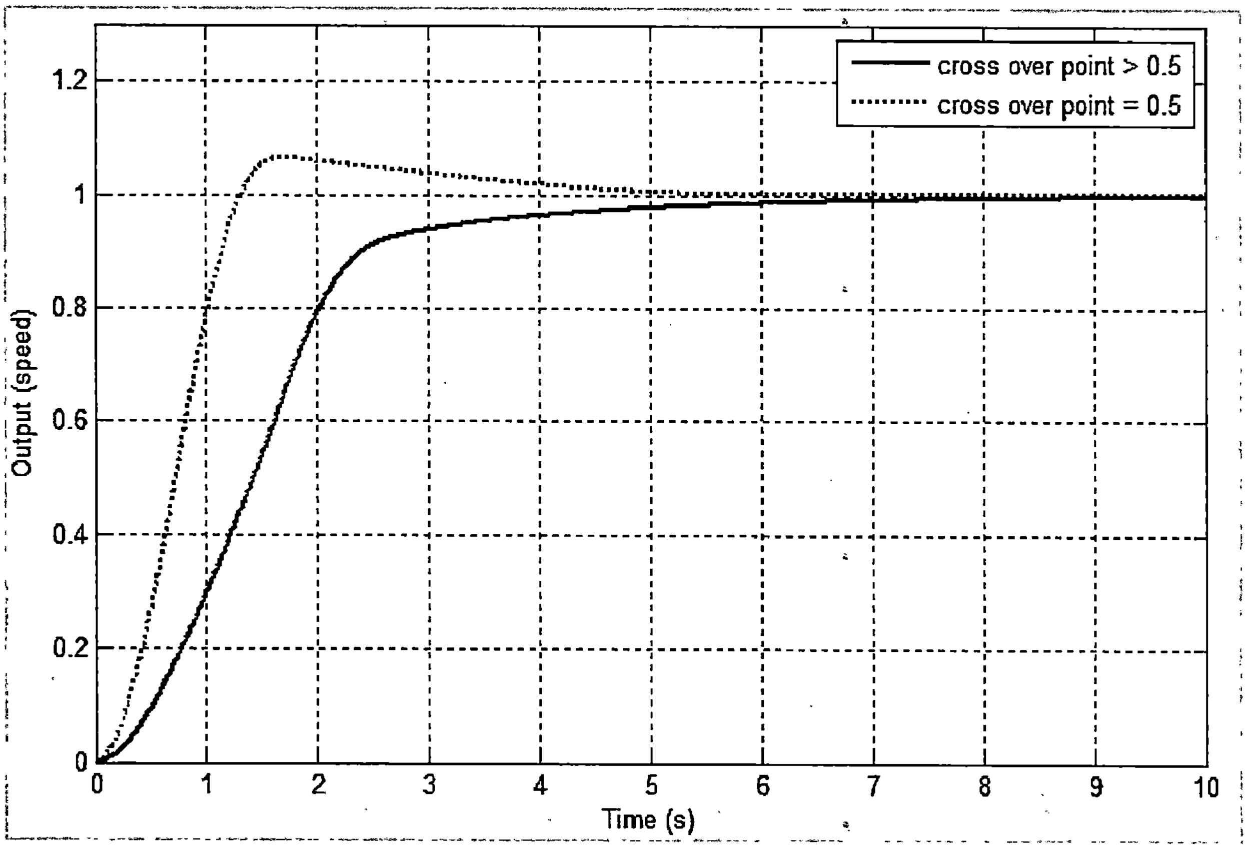

Fig 7.1.5 System Response By Changing K_i And K_d

Becouse of change of rule specialy K_p, the system has oveshoot. It is seen from the graph that, at a point e(Z) and ec(NB) the K_p has to be changed. By changing this rule we got the better response.

It is changed

From, *e(Z)* *ec(NB)* K_p *(PM)*

To, *e(Z)* *ec(NB)* K_p *(PS)*

By changing this rule large overshoot is controlled. As it can be seen from the Figure 6.1.6.

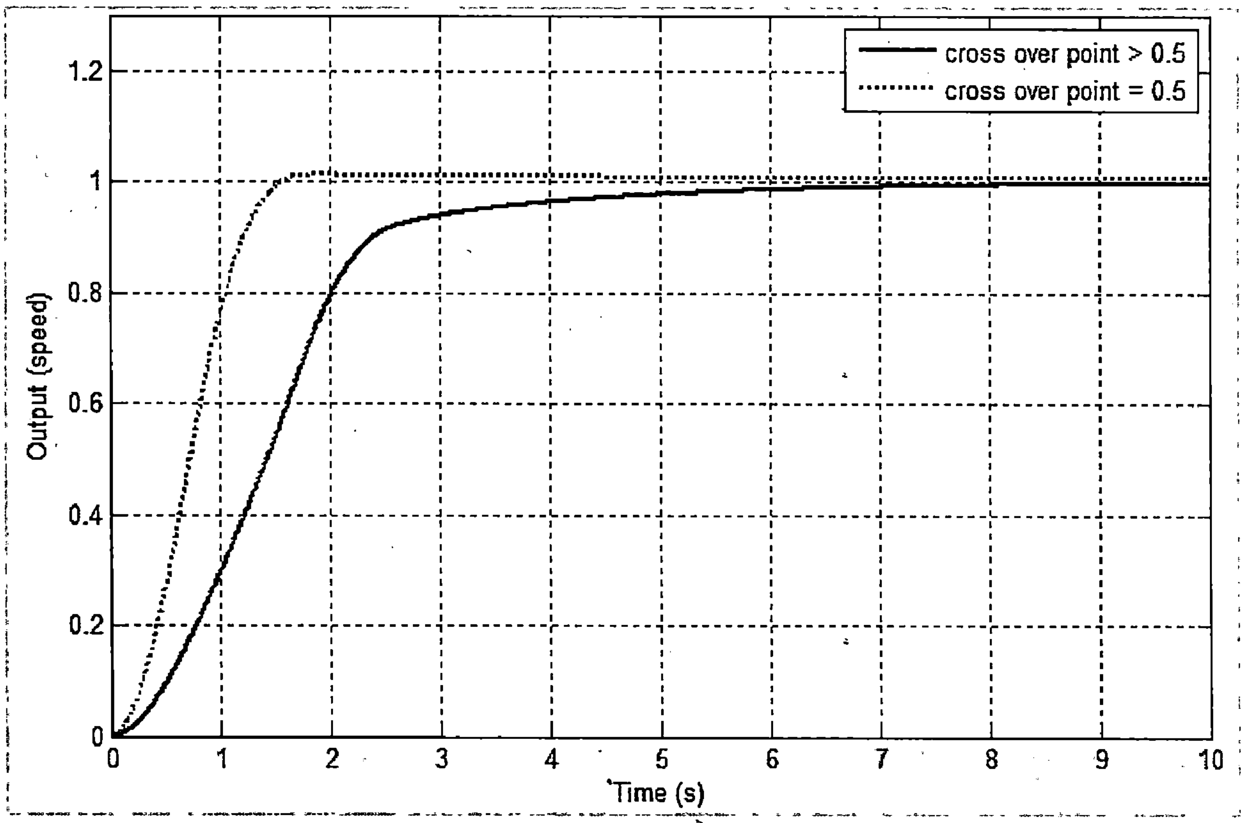

Fig 7.1.6 Final System Response By Changing Rules.

Thus by the systematic analysis of system response we can modify system response by changing some rules which are responsible for system behaviour.

DC Motor is used for the analysis. Modeling of DC Motor is done as under[12].

To perform the simulation of the system, an appropriate model needed to be established. Therefore, a model based on the motor specifications needs to be obtained.

7.1.3 System Equation and Transfer Function

The motor torque T is related to the armature current, **i**, by a torque constant **K**;

$T = Ki$ (7.1)

The generated voltage e_a, is relative to angular velocity by;

$e_a = K\omega_{m=}K\frac{d\theta}{dt}$ (7.2)

we can write the following equations based on the Newton's law combined with the Kirchoff's law:

$$J\frac{d^2\theta}{dt^2} + b\frac{d\theta}{dt} = Ki \qquad (7.3)$$

$$L\frac{di}{dt} + Ri = V - K\frac{d\theta}{dt} \qquad (7.4)$$

Transfer Function

Using the Laplace transform, equations (7.3) and (7.4) can be written as:

$$Js^2\theta(s) + bs\theta(s) = KI(s) \qquad (7.5)$$

$$LsI(s) + RI(s) = V(s) - Ks\theta(s) \qquad (7.6)$$

From (7.6) we can express:

$$I(s) = \frac{V(s) - Ks\theta(s)}{R + Ls} \qquad (7.7)$$

and substitute it in (7.5) to obtain:

$$Js^2\theta(s) + bs\theta(s) = \frac{K(V(s) - Ks\theta(s))}{R + Ls} \qquad (7.8)$$

It is easy to see that the transfer function from the input voltage, **V** (*s*), to the angular velocity, **ω**, is:

$$G_v(s) = \frac{\omega(s)}{V(s)} = \frac{K}{[(R + Ls)(Js + b) + K^2]} \qquad (7.9)$$

Let the motor simulation constant are:

R= 1 ohm,

L= 0.5 H,

K= 0.01 Nm/A,

b= 0.1,

J= 1.

So we get the transfer fuction of DC Motor:

$$G_v(s) = \frac{0.01}{0.5s^2 + 1.05s + 0.1001} \qquad (7.10)$$

Simulation model of system is shown in Fig 7.1.7.

The self tuned simulation model of matlab shows that response of the system is controlled by the fuzzy controller. The scaling factor for error and change in error is calculated through Genetic Algoritham. The Genetic Algoritham gives us the best possible gain of K_p, K_i, K_d. The same gain is used in both the model. Initialy the spreaded M.F shows better result which can be seen from figure 7.1.3. By keeping the same gain we have modified some rules through analysis of system response which gives much better result, it can be seen from figure 7.1.6.

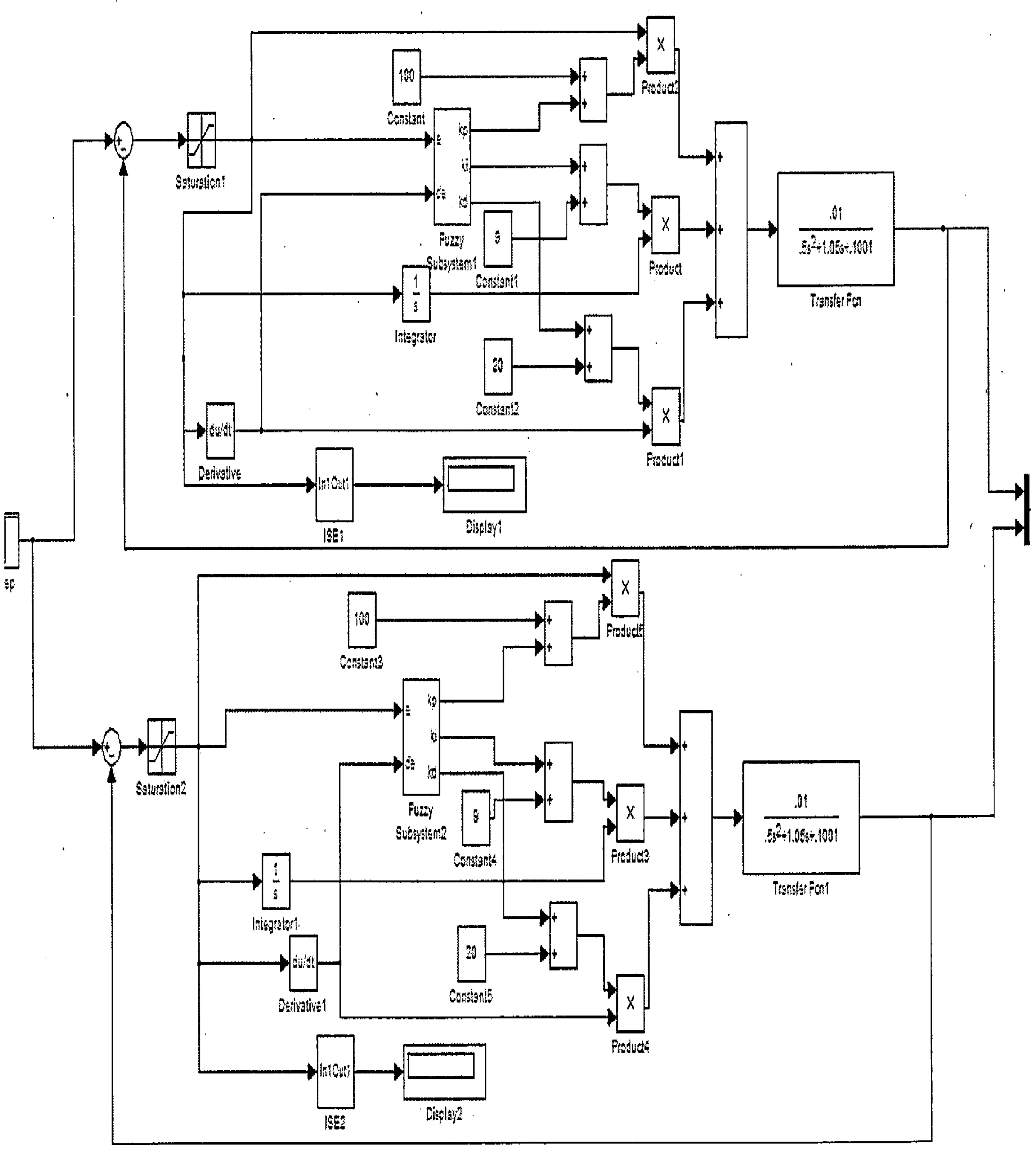

Fig 7.1.7 Matlab Simulation Model of Tuning Technique

7.2 Speed Control of DC Motor by Adaptive Tabular Fuzzy Control

In this controller a adaptive fuzzy controller is created to control the non-linearity of the system. The controller has the sugeno type rule base. Initialy a static rule base is created. The rules base of first fuzzy controller is changed or regulated by another fuzzy controller. When ever the system parameters (DC Motor Transfer Function) is changed the adaptive controller has the best perfonse.

The comperision is made among Adaptive fuzzy, Simple Fuzzy and a PID controller. The simulation is done on Matlab, the block diagram is shown in figure 7.2.3. Equation 7.10 of the DC Motor is used for simulation. Initialy we have tried to adjusted the response of all the three controller to match each other. It can be seen from the figure 7.2.1.

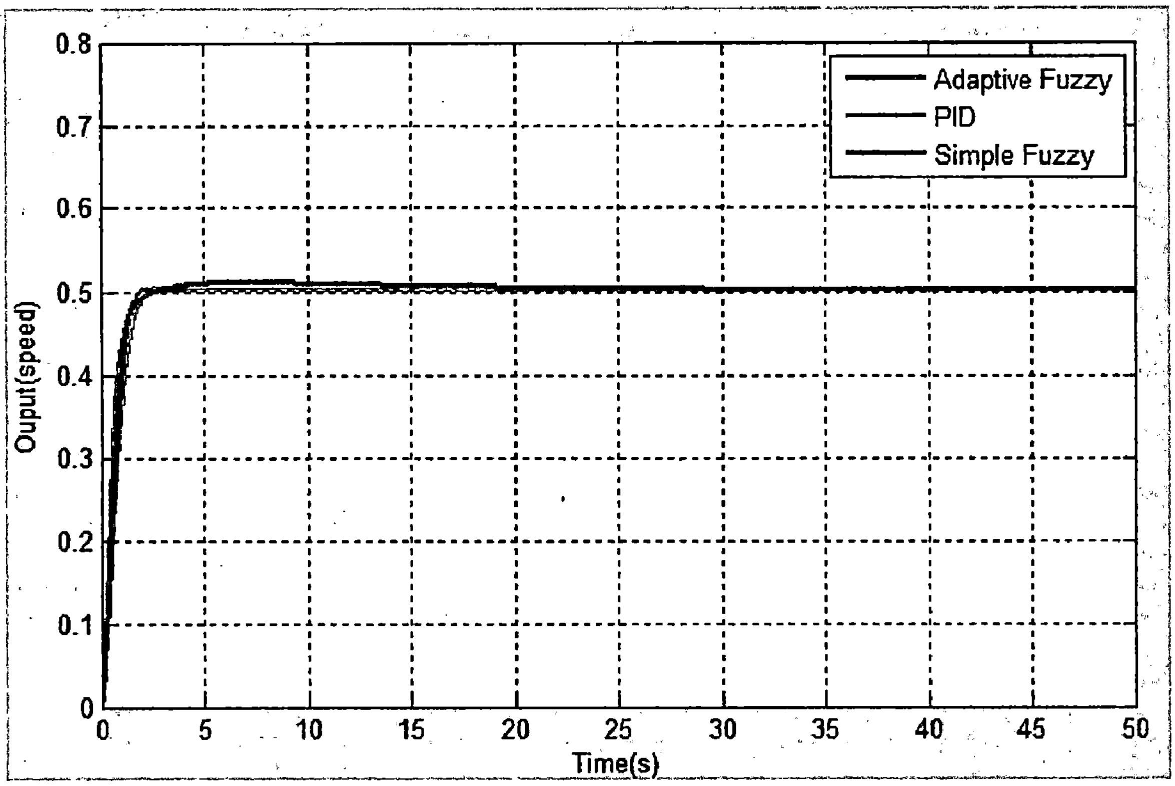

Fig 7.2.1 Response of Three Controller Without System Variation

The ISE of the three controllers are:

Adaptive Fuzzy = 0.099, Simple Fuzzy = 0.126, PID = 0.132

It can be seen from the figure 7.2.1 that all the controllers gives allmost same response for a step input of magnitude 0.5. Now to study the robust ness of the controller, we have again simulated the same simuling block. But this time we have changed the parameter of the system. The coefficient of s^2 is changed from 0.5 to 5.5. The response of the system can be seen from the figure 7.2.2.

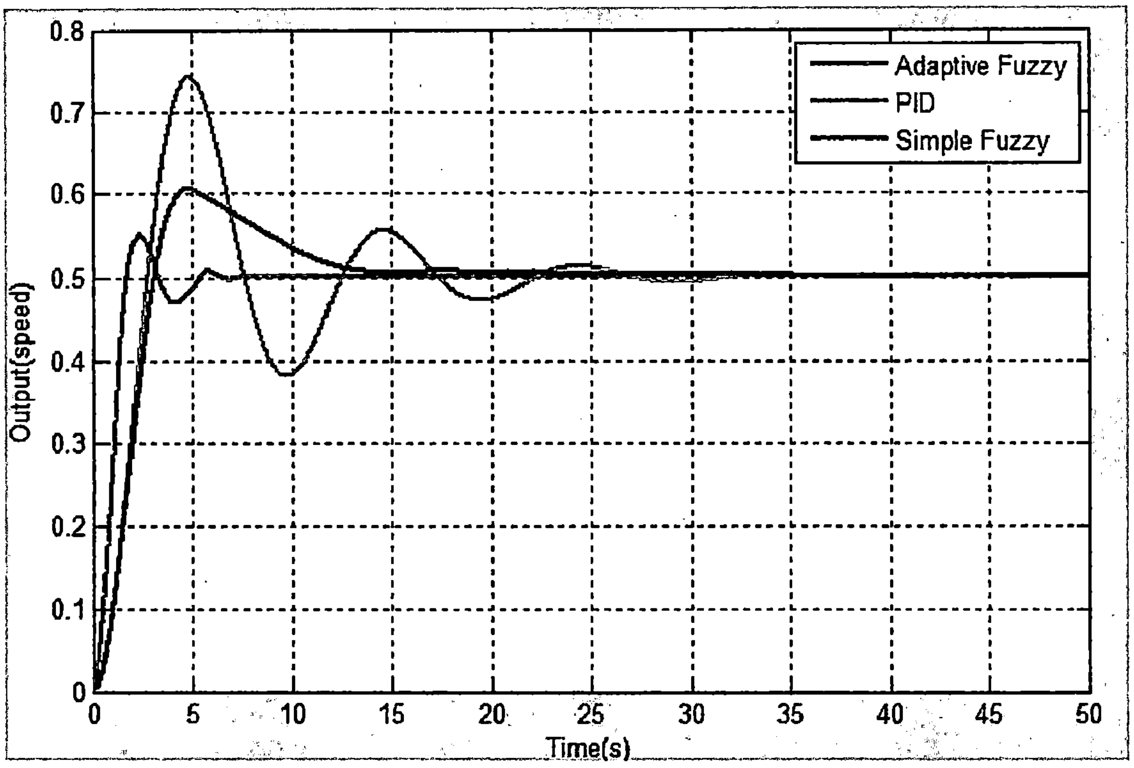

Fig 7.2.2 Response of Three Controller With Parameter Variation

The ISE of three controller after parameter variation:

Adaptive Fuzzy = 0.179, Simple Fuzzy = 0.372, PID = 0.504

Observations from figure 7.2.2:

1. Figure shows Adaptive Fuzzy controller is the most robust controller. It can be seen from the figure, it gives only slight over shoot.
2. Simple fuzzy controller shows better response than PID. It has low over shoot and less settling time than PID.
3. PID was disturbed most. It has the oscillation due to system non-linearity.

Adaptive Controller has the rule base in tabular form. It has two tabular based rule base. One is static and another one is the inverse rule base. The inverse rule base changes the rules of static rule rule base to adapt the non-linearity.

Persistent rules are

```
rules=[1   1   1   1   1   1  0.8  0.6  0.3  0.1   0;
   1   1   1   1   1  0.8  0.6  0.3  0.1  0   -0.1;
   1   1   1   1  0.8 0.6  0.3  0.1  0   -0.1 -0.3;
   1   1   1  0.8  0.6 0.3  0.1  0  -0.1 -0.3 -0.6;
   1   1  0.8  0.6  0.3 0.1  0  -0.1  -0.3  -0.6 -0.8;
   1 0.8  0.6  0.3  0.1  0  -0.1 -0.3  -0.6  -0.8  -1;
  0.8 0.6  0.3  0.1   0  -0.1 -0.3 -0.6  -0.8  -1   -1;
  0.6 0.3  0.1   0  -0.1 -0.3 -0.6 -0.8  -1   -1   -1;
  0.3 0.1   0  -0.1 -0.3 -0.6 -0.8  -1   -1   -1   -1;
  0.1  0  -0.1 -0.3 -0.6 -0.8  -1   -1   -1   -1   -1;
   0 -0.1 -0.3 -0.6 -0.8 -1   -1   -1   -1   -1   -1]*gf;
```

The adaptive nature of the rules can be observed by the change in rules in matlab workspace. One of the changed matrix of rules is shown below:

```
simout(:,:,1400) =


  Columns 1 through 9

  -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -40.0000 -30.0000 -15.0000
  -50.0000 -50.0000 -50.0000 -50.0000 -50.0000 -40.0000 -30.0000 -15.0000  -5.0000
  -50.0000 -50.0000 -50.0000 -50.0000 -40.0000 -30.0000 -15.0000  -5.0000        0
  -50.0000 -50.0000 -50.0000 -40.0000 -30.0000 -15.0000  -5.0000        0   5.0000
  -50.0000 -50.0183 -40.1338 -30.3426 -15.7531  -6.8165  -1.2905   5.0000  15.0000
  -49.9419 -40.0105 -30.1455 -15.3426  -4.2835   0.1192   4.1756  15.0000  30.0000
```

```
 -39.6846 -29.9921 -15.0117  -5.0000   1.4696   6.9356  15.0026  30.0000  40.0000
 -29.3842 -15.0000  -5.0000        0   5.0000  15.0000  30.0000  40.0000  50.0000
 -14.0552  -5.0000        0   5.0000  15.0000  30.0000  40.0000  50.0000  50.0000
  -3.6978        0   5.0000  15.0000  30.0000  40.0000  50.0000  50.0000  50.0000
  15.0092  15.0908  15.0972  30.0000  40.0000  50.0000  50.0000  50.0000  50.0000

Columns 10 through 11

  -5.0000        0
        0   5.0000
   5.0000  15.0000
  15.0000  30.0000
  30.0000  40.0000
  40.0000  50.0000
  50.0000  50.0000
  50.0000  50.0000
  50.0000  50.0000
  50.0000  50.0000
  50.0000  50.0000
```

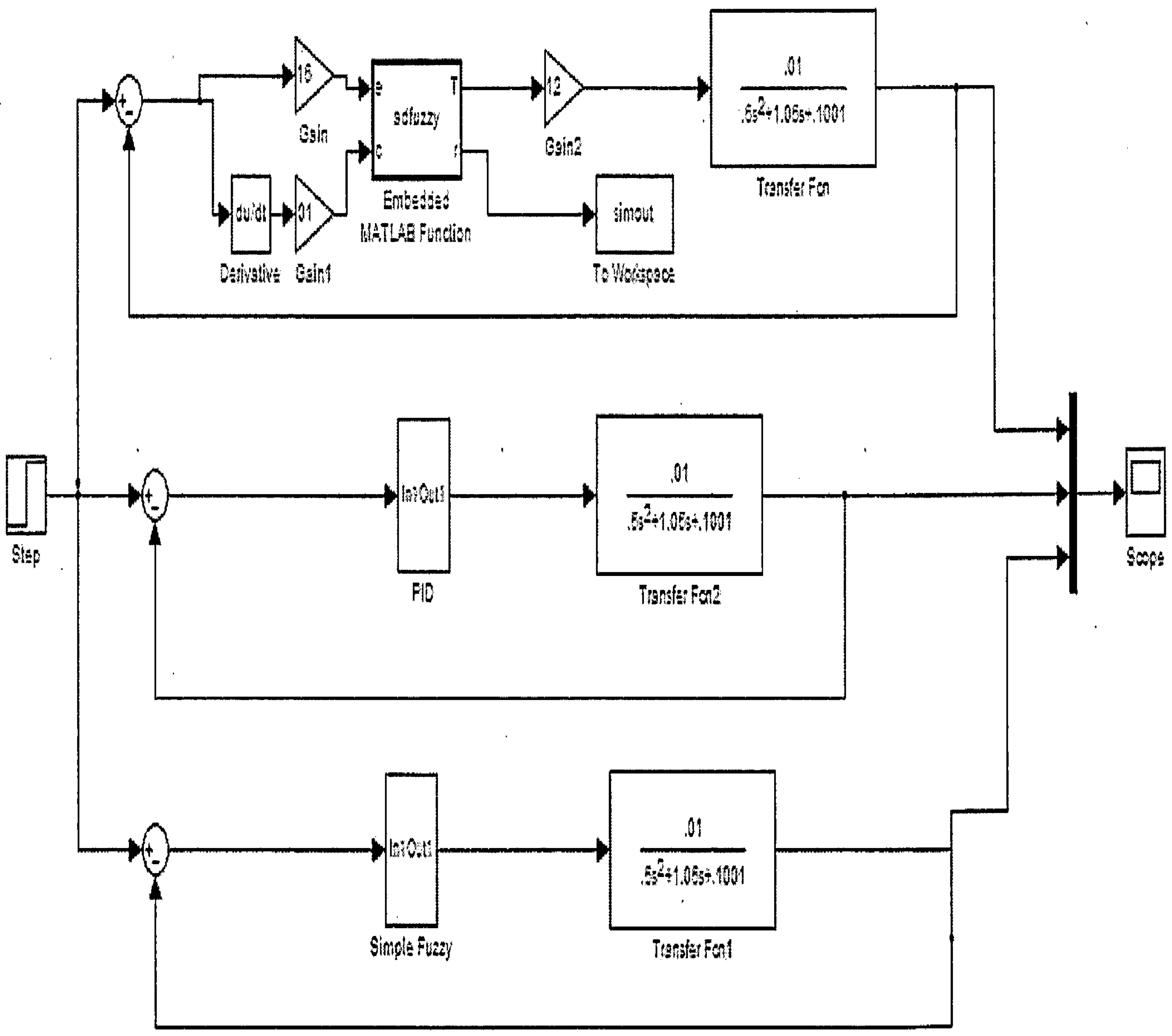

Fig 7.2.3 Matlab Simulation Model of Speed Controller

7.3 Position Control of DC Motor by Adaptive Neuro-Fuzzy Inference System (ANFIS)

The tool box of ANFIS is available in matlab. For creating the FIS of any system, the data are taken into the work space. These datas are used by the tool box to create an adaptive FIS. The FIS created by this tool box is of sugeno type.

The ANFIS for a DC Motor is created to control the angular position of the motor. Initialy we were using PID controller to control its position. But this is not effective. Due this short coming the PID is removed by an FIS created from from ANFIS tool box.

The data points are taken from the PID controller. Three simout is connected for calculating input data base and one simout is connected for output data base. After getting the data base in array form, we open ANFIS tool box. The datas are given into the tool box, the error tolerance is selected and finaly by giving the number epochs, training is started.

After getting the complite FIS the comperetive study of PID and ANFIS are done on matlab simulation. The complete simulation model is shown in figure 7.3.3. In the no-load condition both the controller has same response. It can be seen from figure 7.3.1.

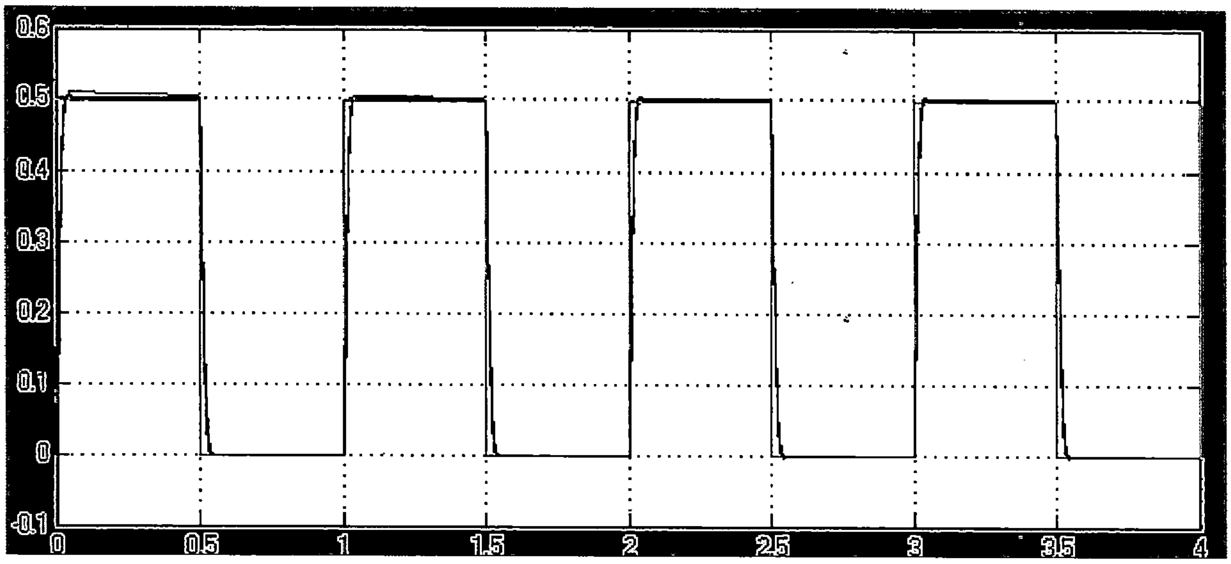

Fig 7.3.1 Step Response of PID and ANFIS at No-Load

It can be seen from the figure 7.3.1, at no-load condition both the controller track the refrense input signal, which is in the form of pulse. The green line in the figure is refrence position. The blue line is the response of ANFIS controller. And the red line is the controller response of PID.

Now the system robustness is tested by loading the simulation model by a load. The response of the system can be seen from the figure 7.3.2.

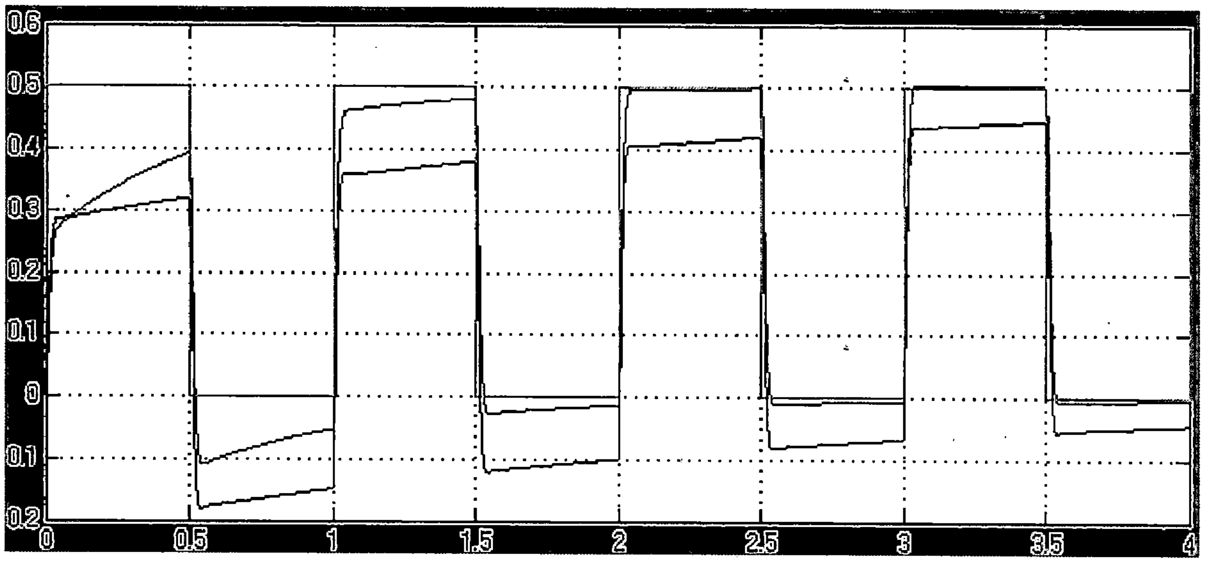

Fig 7.3.2 Step Response of PID and ANFIS on-load

The green line is refrence position. The red line is PID controller output. And blue line is the ANFIS system response.

It can be observed from the figure 7.3.2, the ANFIS gives best response. When the system is loaded both the controller get disturbed. But after some time ANFIS controlled the DC Motor position faster than PID.

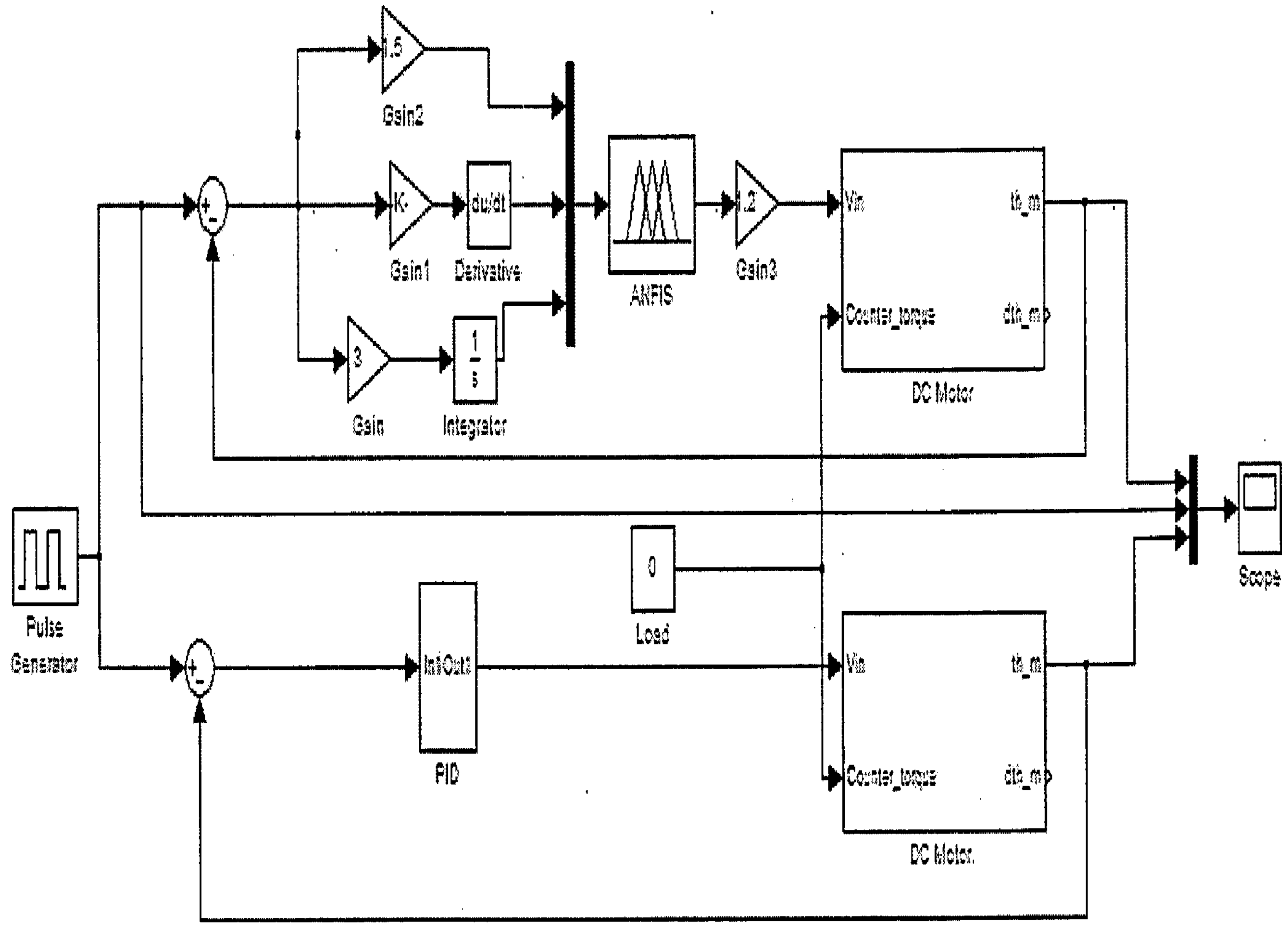

Fig 7.3.3 Matlab Simulation Model of Position Control

CONCLUSION AND FUTURE SCOPE

In this dissertation various speed control technique for DC Motor is discussed in detail. The classical control technique includes PID controller, Root Locus Technique for the speed control, speed control by the Frequency Domain analysis, and Digital PID controller. All these methods are based on manual adjustment and estimation. Which is very time taking and also these methods are not effective in controlling the non-linearity of the DC Motor.

To control the non-linearity of the system a fuzzy logic based controller is used. The controller is very effective in controlling the speed of DC Motor. A much improved response is obtained by the systematic analysis of the graphical response. By altering the rules of fuzzy controller we got the much improved respose.

A comparetive study is made among Adaptive Fuzzy Controller, Simple Fuzzy and PID. All the three controllers are simulated in matlab for the speed control of DC Motor. We concluded that with the variation in system parameter the Adaptive Fuzzy is most robust. And Simple Fuzzy gives better result than PID controller.

Adaptive Nuro-Fuzzy Inference System is used for the position control of DC Motor. It is compared with PID controller. The comparison of the two controllers are observed in matlab simulation. Under the loaded condition of DC Motor the ANFIS gives better response than PID. This shows that ANFIS is a much better controller than normal PID.

Graphical Analysis of the system response can be made generalized for all types of system. So that, this can be used for the designing of more improved fuzzy controller. The Adaptive Fuzzy controller used in this dissertation is sugeno type. This algorithm can be used to construct a mamdani based fuzzy controller.

REFERENCES

[1]. Matlab 7.4 help file, "*Genetic Algorithm*".

[2]. Edward C. Lee, "*Review of Variable Speed Drive Technology*".

[3]. History of dc motor is available on the web site, "*http://www.solarbotics.net/starting/200111_dcmotor/200111_dcmotor.html*".

[4]. I. J. Nagrath, D.P. KHothari, "*Electric Machines*", 2nd edition, TMH publisher.

[5]. Ulrich Bodenhofer; "*Genetic Algorithms: Theory and Applications*", Lecture Notes , Second Edition, WS 2001/2002.

[6]. Available on wesite "*www.its.leeds.ac.uk/projects/smartest/d3f5p9.gif*".

[7]. D Driankov, H Hellendoorn, M Reinfrank; "*An Introduction to Fuzzy Control*", First Narosa Publishing House, Springer International Student Edition, 1993.

[8]. Kevin M. Passino, Stephen Yurkovich, "*Fuzzy Control*" Addison Werley Longman, First Edition, 1998.

[9]. Fedor, P. - Perduková, D.: "*A DC Drive Fuzzy Model*". IJECE – Iranian Journal of Electrical and Computer Engineering, Vol.2, No.1, Winter-Spring 2003, pp.11-16, ISSN 1682 -0053.

[10]. Srinivasan Alavandar, M. J. Nigam, "*Neuro-Fuzzy based Approach for Inverse Kinematics Solution of Industrial Robot Manipulators*", Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844 Vol. III (2008), No. 3, pp. 224-234.

[11]. B. Cao, T.-F. Li, C.-Y. Zhang (Eds.): "*Fuzzy Information and Engineering, Volume 2*" , AISC 62, pp. 967–975.springerlink.com © Springer-Verlag Berlin Heidelberg 2009.

[12]. Wai Phyo Aung, "*Analysis on Modeling and Simulink of DC Motor and its Driving System Used for Wheeled Mobile Robot*", World Academy of Science, Engineering and Technology 32 2007.

Appendix A. TABLE

sl no.	error 1	c error 1	Kp 1	Ki 1	Kd 1	error 2	c error 2	Kp 2	Ki 2	Kd
1	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
2	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
3	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
4	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
5	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
6	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
7	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
8	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
9	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
10	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
11	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
12	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
13	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
14	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
15	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
16	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
17	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
18	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
19	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
20	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
21	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
22	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
23	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
24	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
25	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
26	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
27	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
28	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
29	0.1214	0	0.0049	0.0006	-0.5771	0.1214	0	-0.0158	0.0032	-0.
30	0.1176	-9.4369	0.1228	-0.0313	0.6071	0.1214	0	-0.0158	0.0032	-0.
31	0.1114	-15.2013	0.1229	-0.0315	0.6048	0.1214	0	-0.0158	0.0032	-0.
32	0.105	-15.8977	0.123	-0.0317	0.6025	0.1214	0	-0.0158	0.0032	-0.
33	0.0984	-16.2902	0.1232	-0.0319	0.6	0.1214	0	-0.0158	0.0032	-0.
34	0.0918	-16.4136	0.1233	-0.0321	0.5975	0.1214	0	-0.0158	0.0032	-0.
35	0.0852	-16.3014	0.1234	-0.0324	0.595	0.1202	-2.8749	0.1843	-0.0369	0.
36	0.0787	-15.9861	0.1235	-0.0326	0.5926	0.1151	-12.7823	0.1849	-0.04	0.
37	0.0724	-15.4981	0.1236	-0.0328	0.5902	0.1095	-13.708	0.1856	-0.04	0.
38	0.0664	-14.8661	0.1237	-0.033	0.5878	0.1033	-15.3269	0.1863	-0.04	0.

39	0.0607	-14.1171	0.1238	-0.0332	0.5855	0.0966	-16.4953	0.1871	-0.04	0.
40	0.0553	-13.2763	0.1239	-0.0334	0.5835	0.0897	-17.251	0.188	-0.04	0.
41	0.0503	-12.3666	0.124	-0.0336	0.5816	0.0825	-17.6329	0.1889	-0.04	0.
42	0.0457	-11.4089	0.1241	-0.0337	0.5798	0.0754	-17.6807	0.1897	-0.04	0.
43	0.0415	-10.4221	0.1242	-0.0339	0.5782	0.0683	-17.4337	0.1906	-0.04	0.
44	0.0377	-9.4229	0.1243	-0.034	0.5767	0.0615	-16.9311	0.1914	-0.04	0
45	0.0343	-8.4261	0.1243	-0.0341	0.5754	0.0549	-16.211	0.1923	-0.04	0
46	0.0313	-7.4443	0.1244	-0.0342	0.5742	0.0487	-15.3108	0.1931	-0.04	0.
47	0.0286	-6.4887	0.1244	-0.0343	0.5732	0.0429	-14.2661	0.1939	-0.04	0.
48	0.0264	-5.5684	0.1245	-0.0344	0.5724	0.0376	-13.1103	0.1947	-0.04	0.
49	0.0245	-4.6909	0.1245	-0.0345	0.5716	0.0328	-11.8747	0.1953	-0.04	0.
50	0.0229	-3.8623	0.1245	-0.0345	0.571	0.0285	-10.5882	0.1959	-0.04	0.
51	0.0217	-3.0871	0.1246	-0.0346	0.5705	0.0248	-9.2773	0.1965	-0.04	0.
52	0.0207	-2.3687	0.0874	-0.0197	0.0555	0.0216	-7.9662	0.1969	-0.04	0.
53	0.0199	-1.8906	0.072	-0.0159	-0.1937	0.0189	-6.676	0.1973	-0.04	0.
54	0.0193	-1.6734	0.0588	-0.0123	-0.3724	0.0167	-5.4255	0.1976	-0.04	0.
55	0.0186	-1.5293	0.0509	-0.0104	-0.4793	0.015	-4.2306	0.1979	-0.04	0.
56	0.0181	-1.4202	0.0454	-0.0092	-0.555	0.0137	-3.1045	0.198	-0.04	0.
57	0.0175	-1.3325	0.0411	-0.0083	-0.6048	0.0129	-2.0579	0.1981	-0.0396	-0.
58	0.017	-1.2596	0.0376	-0.0075	-0.6366	0.0124	-1.1159	0.1133	-0.0227	-0.
59	0.0165	-1.1972	0.0347	-0.007	-0.657	0.0122	-0.6361	0.059	-0.0118	-0.
60	0.0161	-1.1429	0.0322	-0.0064	-0.6703	0.0119	-0.5867	0.055	-0.011	-0
61	0.0156	-1.0951	0.03	-0.006	-0.6744	0.0117	-0.4405	0.0434	-0.0087	-0
62	0.0152	-1.0524	0.0281	-0.0056	-0.6755	0.0116	-0.3546	0.0363	-0.0073	-0.
63	0.0148	-1.014	0.0264	-0.0053	-0.6762	0.0115	-0.3045	0.0321	-0.0064	-0.
64	0.0144	-0.9791	0.0251	-0.005	-0.6765	0.0114	-0.2756	0.0296	-0.0059	-0
65	0.014	-0.9468	0.0247	-0.0048	-0.6767	0.0113	-0.2587	0.0281	-0.0056	-0.

Appendix B. M-FILES

3.2.1 m-file of transfer function:-

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
step(num,den,0:0.1:3)
title('Step Response for the Open Loop System')
```

3.4.1 m-file of root locus

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
rlocus(num,den)
sgrid(.8,0)
sigrid(2.3)
title('Root Locus without a controller')
```

The command sigrid is the user-defined function.

3.4.2 m-file sigrid

```
  function[ ] = sigrid(sig)
  error(nargchk(1,1,nargin));
  hold on
  limits = axis;
  mx=limits(1,4);
  mn=limits(1,3);
  stz=abs(mx)+abs(mn);
  st=stz/50;
  im=mn:st:mx;
  lim=length(im);
  for i=1:lim
  re(i)=-sig;
  end
  re(:);
  plot(re,im,'.')
 hold off
return
```

3.4.3 m-file

```
 J=0.01;
 b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
z1=1;
p1=0.01;
numa = [1 z1];
dena = [1 p1];
numb=conv(num,numa);
denb=conv(den,dena);
rlocus(numb,denb)
sgrid(.8,0)
sigrid(2.3)
title('Root Locus with a lag controller')
[k,poles]=rlocfind(numb,denb)
[numc,denc]=cloop(k*numb,denb,-1);
            t=0:0.01:3;
step(numc,denc,t)
title('Step response with a lag controller')
```

3.5.1 m-file bode plot

```
J=0.01;
b=0.1;
K=0.01;
R=1;
L=0.5;
num=K;
den=[(J*L) ((J*R)+(L*b)) ((b*R)+K^2)];
bode(num,den)
```

3.6.1 m-file

```
R=1;
L=0.5;
Kt=0.01;
J=0.01;
b=0.1;
num = Kt;
den = [(J*L) (J*R)+(L*b) (R*b)+(Kt^2)];
Ts = 0.12;
[numz,denz] = c2dm(num,den,Ts,'zoh')
```

3.6.2 m-file

```
R=1;
L=0.5;
Kt=0.01;
J=0.01;
b=0.1;
num = Kt;
den = [(J*L) (J*R)+(L*b) (R*b)+(Kt^2)];
Ts = 0.12;
[numz,denz] = c2dm(num,den,Ts,'zoh')
numz = [numz(2) numz(3)];
[numz_cl,denz_cl] = cloop(numz,denz);
[x1] = dstep(numz_cl,denz_cl,101);
t=0:0.12:12;
stairs(t,x1)
xlabel('Time (seconds)')
ylabel('Velocity (rad/s)')
title('Stairstep Response:Original')
Kp = 100;
Ki = 200;
Kd = 10;
[dencz,numcz]=c2dm([1 0],[Kd Kp Ki],Ts,'tustin');
numaz = conv(numz,numcz);
denaz = conv(denz,dencz);
[numaz_cl,denaz_cl] = cloop(numaz,denaz);
[x2] = dstep(numaz_cl,denaz_cl,101);
t=0:0.12:12;
stairs(t,x2)
xlabel('Time (seconds)')
ylabel('Velocity (rad/s)')
title('Stairstep Response:with PID controller')
rlocus(numaz,denaz)
title('Root Locus of Compensated System')
dencz = conv([1 -1],[1.6 1])
numaz = conv(numz,numcz);
denaz = conv(denz,dencz);
rlocus(numaz,denaz)
title('Root Locus of Compensated System');
[K,poles] = rlocfind(numaz,denaz)
[numaz_cl,denaz_cl] = cloop(K*numaz,denaz);
[x3] = dstep(numaz_cl,denaz_cl,101);
t=0:0.12:12;
stairs(t,x3)
xlabel('Time (seconds)')
ylabel('Velocity (rad/s)')
title('Stairstep Response:with PID controller')
```