SPEED CONTROL OF DC MOTOR USING SELF-TUNED FUZZY PID CONTROLLER

A Thesis presented to
The Academic Faculty
By

Fahim Faisal (102451)
Mahdi Rahman (102427)
Fahad Bin Hashem (102437)

Under The Supervision Of
Prof. Dr. Md. Ashraful Hoque

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
THE ORGANIZATION OF ISLAMIC COOPERATION (OIC)
BOARDBAZAR, GAZIPUR-1704, BANGLADESH

SPEED CONTROL OF DC MOTOR USING SELF-TUNED FUZZY PID CONTROLLER

A Thesis presented to
The Academic Faculty
By

Fahim Faisal (102451)
Mahdi Rahman (102427)
Fahad Bin Hashem (102437)

In Partial Fulfillment
of the Requirements for the Degree
of

Bachelor of Science in Electrical and Electronic Engineering

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)
THE ORGANIZATION OF THE ISLAMIC COOPERATION (OIC)
BOARDBAZAR, GAZIPUR-1704, BANGLADESH

NOVEMBER 2014

SPEED CONTROL OF DC MOTOR USING SELF-TUNED FUZZY PID CONTROLLER

A Thesis presented to
The Academic Faculty
By

Fahim Faisal (102451)
Mahdi Rahman (102427)
Fahad Bin Hashem (102437)

Approved by

Prof. Dr. Md. Ashraful Hoque

Prof. Dr. Md. Ashraful Hoque
Thesis Supervisor
Dept. of Electrical & Electronic Engineering

Prof. Dr. Md. Shahid Ullah
Head of the Department
Dept. of Electrical & Electronic Engineering

DECLARATION

This is to certify that the project entitled “Speed Control of DC Motor using Self-tuned Fuzzy PID Controller” is supervised by Prof. Dr. Md. Ashraful Hoque. This Thesis work has not been submitted anywhere for a degree.

_____

Prof. Dr. Md. Ashraful Hoque
Thesis Supervisor
Dept. of Electrical & Electronic Engineering

_____

Fahim Faisal
Co-author

_____

Mahdi Rahman
Co-author

_____

Fahad Bin Hashem
Co-author

ACKNOWLEDGEMENTS

The undergraduate thesis "***Speed Control of DC Motor using Self-tuned Fuzzy PID Controller"*** has been written for the completion of Bachelor of Science degree at Islamic University of Technology, Board Bazar, Gazipur, Bangladesh. This thesis work and writing has been done during the year 2014 under the supervision of ***Dr. Md. Ashraful Hoque, Professor of EEE Department.***

We would like to pay our special thanks and express our deepest gratitude to ***Prof. Dr. Md. Shahid Ullah,*** Head of the Department and ***Prof. Dr. Md. Ashraful Hoque,*** thesis supervisor for their constant guidance, help and encouragement to our work. Without their support it would have been impossible to complete such a task successfully.

We are also grateful to all our well-wishers, who provided their support towards accomplishing this task successfully. Finally, we beg pardon and apologize for the faults and any unintentional mistakes that might be recurred in this thesis paper even after all the care that was taken.

ABSTRACT

This thesis work describes how the speed of a DC motor can be controlled by a self-tuned fuzzy controller. The speed of a DC motor depends on armature voltage, field current and the torque demand. In this thesis, the armature voltage is changed in order to have the required speed output. The armature voltage is firstly controlled by a simple proportional (P) controller and then controlled by a proportional-integral-derivative (PID) controller and finally the values of the proportional, integral and derivative constants are controlled and tuned by a fuzzy logic controller (FLC), hence the name Self-tuned fuzzy PID controller. There are two inputs to the FLC. One input is the difference between the reference speed (the desired speed) and the actual speed available as the output. This difference is known as the speed error (e). The other input is the rate of change of this speed error (de). Both of these are crisp (binary valued) sets. The FLC takes these two crisp sets, convert them into two separate fuzzy sets, takes decision based on a fuzzy inference system comprising of 25 rules in case of our model, then defuzzify the result into three crisp sets of the proportional gain (K_p), integral gain (K_i) and derivative gain (K_d). Basically the PID controller alone can control the speed of the motor, which is first shown in the paper. However, it is difficult to fathom the values of the parameters of the PID controller (K_p, K_i, K_d) that would give the best output (i.e., an output that has small rise time, settling time and fall time, small overshoot, and small or no steady state error). It is not practically feasible to make a trial and error analysis to figure out the values and the calculations regarding the computations of the accurate values can be quite complex, time consuming and not suitable at all in a practical application where a wide range of motor running speed may be desired. The FLC tunes the values of PID parameters based on the fuzzy rules fed into it and gives a satisfactory as well as controlled output wave shape for wide range of motor speed variation which is shown later. All block diagrams and simulations are done utilizing MATLAB Simulink, and the Fuzzy Logic Toolbox of MATLAB is used to design our proposed model of FLC.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure No.	Title of The Figure	Page No.

LIST OF SYMBOLS

Symbols	Meaning of the Symbols
V_t	Armature Voltage
E_b	Back emf of The Motor
I_a	Armature Current
R_a	Armature Resistance
L_a	Armature Inductance
T_m	Mechanical Torque
J_m	Moment of Inertia
B_m	Coefficient of The Motor
ω	Angular Velocity
K_m	Motor Torque Constant
K_b	Back emf Constant
B_L	Friction Coefficient
$T_{\omega L}$	Load Torque
μ	Degree of Membership
T_r	Rise Time
T_p	Peak Time
Ts	Settling Time
T_f	Fall Time
%OS	Percentage Overshoot

LIST OF ABBREVIATIONS

Abbreviated Form	Description
K_p	Proportional Gain
K_i	Integral Gain
K_d	Derivative Gain
FLC	Fuzzy Logic Controller
SSE	Steady State Error
ANFIS	Adaptive Neuro-Fuzzy Inference Systems
SP	Set Point
PV	Process Variable
MF	Membership Function
COG	Center of Gravity
BOA	Bisector of Area
MOM	Mean of Maximum
NL	Negative Large
NS	Negative Small
ZE	Zero
PS	Positive Small
PL	Positive Large
PVS	Positive Very Small
PMS	Positive Medium Small
PM	Positive Medium
PML	Positive Medium Large
PVL	Positive Very Large
IGBT	Insulated Gate Bipolar Transistor
MOSFET	Metal Oxide Semiconductor Field Effect Transistor
GTO	Gate Turn Off thyristor
FLS	Fuzzy Logic System

CHAPTER 1

INTRODUCTION

1.1: INTRODUCTION

DC drives are used extensively because of their simplicity, high starting torque, ease of application, high reliabilities, flexibilities and favorable cost and it is a backbone of industrial applications like steel rolling mills, and electric trains, electric vehicles, electric cranes, robotic manipulators and home appliances where speed and position control of motor are required [1]. The speed of DC motors can easily be adjusted within wide boundaries so that this provides easy controllability and high performance. In general, a high performance motor drive system must have good dynamic speed command tracking and load regulating response to perform task. In these applications, the motor should be precisely controlled to give the desired performance [2]

DC motors speed controller is carried out by means of voltage control. The regulated voltage sources used for DC motor speed control switching devices of power electronics such as MOSFET, IGBT and GTO [3].

For the speed control of DC motor, various types of controllers are available but most widely used controllers are conventional PID controllers but due to non–linearity of DC motor these controllers face problems. The non-linearity arises due to armature current limitation, change in loads and drive inertia [4]. In addition, the parameters of the PID controller have to be fine-tuned (K_p, K_i, K_d) which is difficult to model in a non-linear system. Hence to obtain better speed control, the conventional PID controllers combined with intelligent techniques such as Fuzzy logic are in widely use. Fuzzy logic controller deals with problems having uncertainties and use membership functions with values lying between 0 and 1, and uses rules to optimize the system performances [5]. In this case, the fuzzy logic controller (FLC) can be used to fine-tune the parameters of A PID controller for a non-linear model like that of a DC motor as stated before.

In our study, the speed response of a DC motor exposed to fixed armature voltage was investigated for both under loaded and unloaded operating conditions. Then, to make performance comparison, the speed of the system was controlled using a PID controller tuned by an FLC. The system designed for operating at fixed speed under different load conditions are simulated at MATLAB/Simulink environment.

1.2: NECESSITY OF PID

PID controllers are known as a family of controllers. PID controllers are sold in large quantities and are often the solution of choice when a controller is required to close the loop. The reason PID controllers are so popular is that using PID gives the designer a larger number of options and those options mean that there are more possibilities for changing the dynamics of the system in a way that helps the designer. If the designer can work it right then s/he can get the advantages of several effects. In particular, starting with a proportional controller, and adding integral and derivative terms to the control the designer can take advantage of the following effects:

- A proportional (p) controller gives less amount of overshoot and steady state error (SSE).
- An integral (i) controller gives zero SSE for a step input.
- A derivative (d) controller often produces faster response.

Therefore, it is desirable to have a single controller that contains the characteristics of all three types of controllers described above. This is implemented in the form of proportional-integral-derivative (PID) controller.

A PID controller operates on the error in a feedback system and does the following:

- It calculates a term proportional to the error - the p term.
- It also calculates a term proportional to the integral of the error - the i term.
- And calculates a term proportional to the derivative of the error - the d term.
- The three terms - the p, i and d terms, are added together to produce a control signal that is applied to the system being controlled.

1.3: WHY USE PID CONTROLLERS?

We need a PID controller in many situations, particularly in the following cases where only a proportional, or only an integral, or only a derivative controller may not provide optimum performance.

- A proportional controller gives a good overshoot performance.
- A proportional controller may not give SSE performance needed in a system.
- An integral controller may give SSE performance, but slow a system down.
- An integral one also increases overshoot and settling time.

- Adding a derivative term may help cure both of those problems [6].
- Derivative action predicts system behavior and thus improves settling time and stability of the system.
- Derivative action is seldom used in practice though because of its variable impact on system stability in real-world applications.

1.4: WHY USE FUZZY LOGIC?

Here is a list of general observations about fuzzy logic:

- Fuzzy logic is conceptually easy to understand. The mathematical concepts behind fuzzy reasoning are very simple. What makes fuzzy nice is the "naturalness" of its approach and not its far-reaching complexity.
- It is flexible. With any given system, it's easy to massage it or layer more functionality on top of it without starting again from scratch.
- This logic is tolerant of imprecise data. Everything is imprecise if you look closely enough, but more than that, most things are imprecise even on careful inspection. Fuzzy reasoning builds this understanding into the process rather than tacking it onto the end.
- Fuzzy logic can model nonlinear functions of arbitrary complexity. You can create a fuzzy system to match any set of input-output data. This process is made particularly easy by adaptive techniques like ANFIS (Adaptive Neuro-Fuzzy Inference Systems), which are available in the Fuzzy Logic Toolbox.
- This logic can be built on top of the experience of experts. In direct contrast to neural networks, which take training data and generate opaque, impenetrable models, fuzzy logic lets you rely on the experience of people who already understand your system.
- It can be blended with conventional control techniques. Fuzzy systems don't necessarily replace conventional control methods. In many cases fuzzy systems augment them and simplify their implementation.
- Fuzzy logic is based on natural language. The basis for fuzzy logic is the basis for human communication. This observation underpins many of the other statements about fuzzy logic.

Lastly, this logic is perhaps the most important one and it deserves more discussion. Natural language, that which is used by ordinary people on a daily basis, has been shaped by thousands of years of human history to be more convenient and efficient. Sentences written in ordinary language represent a triumph of efficient and convenient communication. We are generally unaware of this because ordinary language is, of course, something we use every day. Since fuzzy logic is built atop the structures of qualitative description used in everyday language, fuzzy logic is easy to use [7].

1.5: OVERVIEW OF THIS THESIS

This thesis is organized as follows: chapter 1 contains the general description and reason for choosing DC motor, and why PID controller and fuzzy logic are used to design the controller in this thesis; chapter 2 covers the details of DC motor including its modelling and simulation; chapter 3 describes in details the theory of PID controller and its applications as well as limitations; chapter 4 contains simulation details of a DC motor controlled by a PID controller; chapter 5 covers details of fuzzy logic theory; chapter 6 explains the general processes involved in a fuzzy logic controller; chapter 7 contains the details used for designing the fuzzy logic controller in this thesis; chapter 8 contains the simulation of DC motor controlled by self-tuned fuzzy PID controller; chapter 9 displays performance analysis of speed-time characteristics from the simulation results from the previous chapter; and finally concludes the thesis with describing the benefits of self-tuned fuzzy PID controller and future scope in this line of research.

CHAPTER 2
DC MOTOR AND SIMULATION

2.1: INTRODUCTION

Over the years works in electric motor control systems have been expanded into a wide range. Various systems have been designed to control the motor parameters such as speed, torque, etc. In this presentation we discuss various techniques of changing the armature voltage to control the speed of a DC motor drive. Due to the ability to provide a variable DC output voltage from a fixed AC voltage, rectifiers are used. DC motors have variable characteristics and are used extensively in variable speed drives.

Motor speed can be varied by controlling-

- Armature voltage ,
- Field current &
- Torque demand.

DC Machines are of three kinds as well-

- Separately excited ,
- Shunt excited &
- Series excited

2.2: EQUIVALENT CIRCUIT OF DC MOTOR

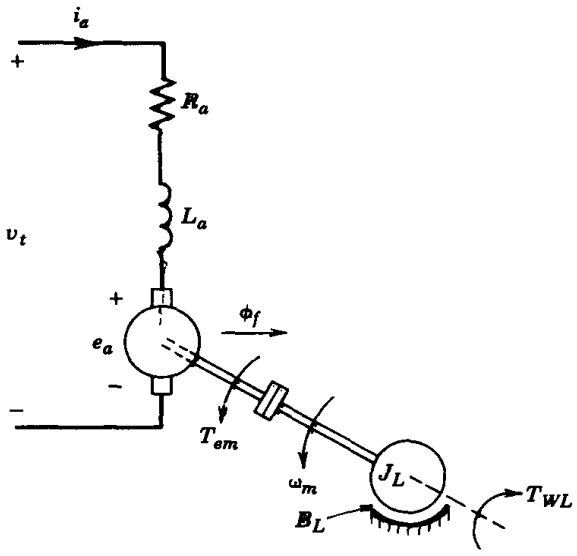

Figure 2.1: Equivalent circuit of a DC motor [8]

- V_t is the armature voltage (In volt)
- E_b is back emf the motor (In volt)

- I_a is the armature current (In ampere)
- R_a is the armature resistance (In ohm)
- L_a is the armature inductance (In henry)
- T_m is the mechanical torque developed (In Nm)
- J_m is moment of inertia (In kg/m²)
- B_m is friction coefficient of the motor (In Nm/ (rad/sec))
- ω is angular velocity (In rad/sec)

2.3: MODELLING OF DC MOTOR

The equations governing a DC motor are shown below.

As there are two parts of our proposed DC model, one is electrical part and the other one is mechanical part. Equations are separated as follows.

For electrical modelling:

$$V_t(t) = R_a i_a(t) + L_a \frac{di_a}{dt}(t) + e_a(t) \quad (1)$$

Or,

$$\frac{di_a}{dt}(t) = \frac{1}{L_a}[V_t(t) - R_a i_a(t) - e_a(t)]$$

Therefore,

$$i_a(t) = \int [\frac{1}{L_a}[V_t(t) - R_a i_a(t) - e_a(t)]dt \quad (2)$$

Where,

$$e_a(t) = K_b w_m(t) \quad (3)$$

For mechanical modelling:

$$T_{em}(t) = K_m i_a(t) \quad (4)$$

$$T_{em}(t) = J_L \frac{dw_m}{dt}(t) + B_L w_m(t) + T_{wL}(t) \quad (5)$$

Or,

$$\frac{dw_m}{dt}(t) = \frac{1}{J_L}[T_{em}(t) - B_L w_m(t) - T_{\omega L}(t)]$$

Therefore,

$$\omega_m(t) = \int [\frac{1}{J_L}[T_{em}(t) - B_L \omega_m(t) - T_{\omega L}(t)]dt \quad (6)$$

2.4: SPECIFICATIONS AND PARAMETRS

The parameters of the DC motor used for the simulations in this paper are as follows:

Armature resistance (R_a) = 0.5Ω

Armature inductance (L_a) = 0.02 H

Armature voltage (V_t) = 200 V

Mechanical inertia (J_L) = 0.1 Kg.m2

Friction coefficient (B_L) = 0.008 N-m/rad/sec

Back emf constant (K_b) = 1.25 V/rad/sec

Rated speed = 3000 r.p.m

Motor torque constant (K_m) = 1 N-m/A

Initially motor runs at no-load. A load torque is added after 15 seconds from start and remains till end of simulation.

Value of load torque $T_{\omega L}$ = 100 N-m

2.5: DC MOTOR IN SIMULINK

Simulink diagrams are synonymous to block diagrams familiar in control systems engineering. The "scopes" and "displays" shown in the Simulink diagram are placed in order to check the variables at each stage they are placed.

The Simulink block diagram based on the above equations is as follows. The values of the parameters are as listed as above.

And on the basis of this Simulink diagram shown below, we then use it in MATLAB Simulink and get our result of speed-time characteristics to find the wave shape and compare this with the other used controllers used in the later chapters.

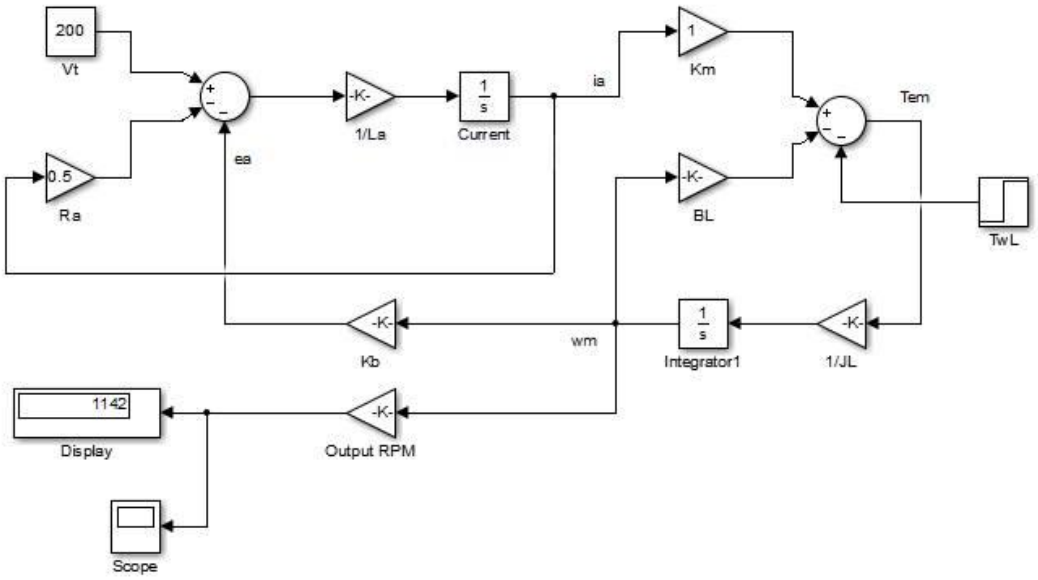

Figure 2.2: Simulink block diagram of a simple DC motor.

Initially, motor is at rest. When an armature voltage of 200V is applied, the speed immediately tries to reach the value set by the voltage. There is high short circuit current flow at the beginning so the speed goes higher than the speed that was supposed to be set for 200V. Because of mechanical inertial and friction, there are overshoot and undershoot. Finally, the speed settles at 1523 r.p.m. After 15 seconds, a load torque of 100 N-m is added. As a result, speed falls to a new value of 1142 r.p.m., after some undershoot and overshoot caused by the same reasons.

Overshoot and undershoot is unwanted, and overshoot particularly is dangerous because high speed may cause the motor to be damaged. It is necessary to reduce these effects so that the motor operates more smoothly.

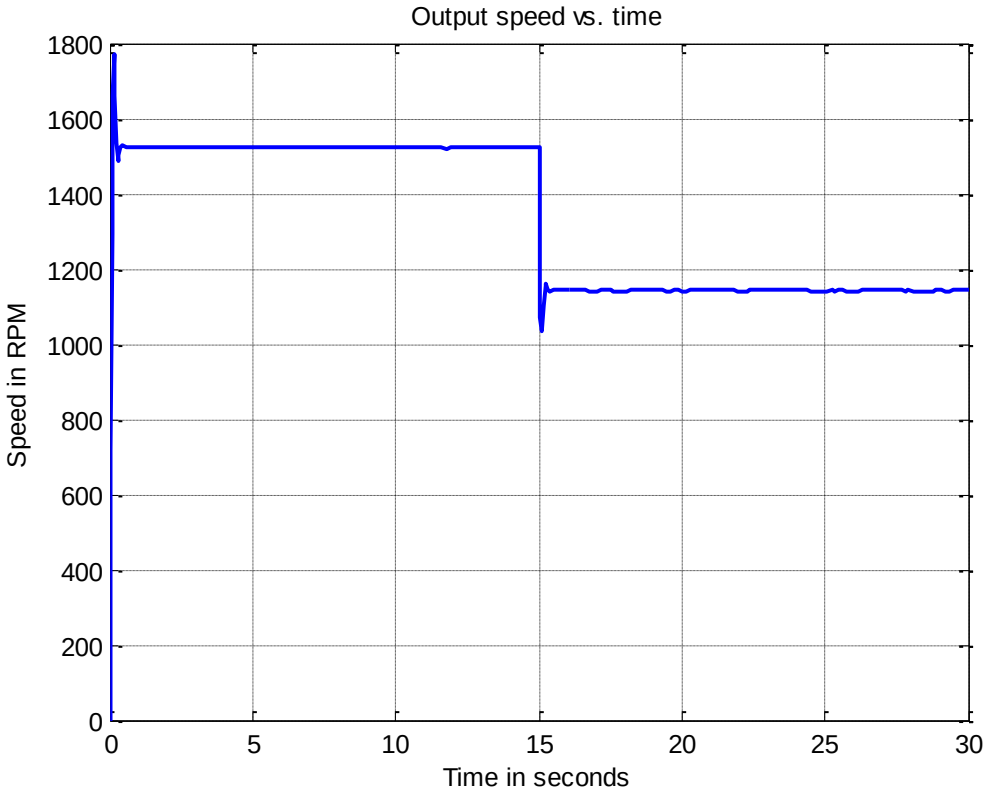

Figure 2.3: Speed-time characteristics of a simple DC motor

Now, we wish to change the speed of the motor according to our desire for the same parameters. This is shown next.

CHAPTER 3
PID CONTROLLER THEORY

3.1: INTRODUCTION TO PID

A proportional-integral-derivative controller (PID controller) is control loop feedback mechanism also known as controller widely used in industrial control systems. A PID controller calculates an error value as the difference between a measured process variable and a desired set point. It is one of the earlier control strategies. Its early implementation was in pneumatic devices, followed by vacuum and solid state analog electronics, before arriving at today's digital implementation of microprocessors. It has a simple control structure which was understood by plant operators and which they found relatively easy to tune. Since many control systems using PID control have proved satisfactory, it still has a wide range of applications in industrial control [9]. According to a survey for process control systems conducted in 1989, more than 90 of the control loops were of the PID type. PID control has been an active research topic for many years. Since many process plants controlled by PID controllers have similar dynamics it has been found possible to set satisfactory controller parameters from less plant information than a complete mathematical model. These techniques came about because of the desire to adjust controller parameters in situ with a minimum of effort, and also because of the possible difficulty and poor cost benefit of obtaining mathematical models. The two most popular PID techniques were the step reaction curve experiment, and a closed-loop "cycling" experiment under proportional control around the nominal operating point. In the absence of knowledge of the underlying process, a PID controller has historically been considered to be the best controller. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the set point, and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

3.2: WHAT IS PID CONTROLLER?

Proportional-Integral-Derivative (PID) control is the most common control algorithm used in industry and has been universally accepted in industrial control. The popularity of PID controllers can be attributed partly to their robust performance in a wide range

of operating conditions and partly to their functional simplicity, which allows engineers to operate them in a simple, straightforward manner for our given proposed model. As the name suggests, PID algorithm consists of three basic coefficients; proportional, integral and derivative which are varied to get optimal response [10].

3.2.1: Proportional response

The proportional component depends only on the difference between the set point and the process variable. This difference is referred to as the Error term. The proportional gain determines the ratio of output response to the error signal. For instance, if the error term has a magnitude of 10, a proportional gain of 5 would produce a proportional response of 50. In general, increasing the proportional gain will increase the speed of the control system response. However, if the proportional gain is too large, the process variable will begin to oscillate. If K_p is increased further, the oscillations will become larger and the system will become unstable and may even oscillate out of control.

3.2.2: Integral response

The integral component sums the error term over total time. The result is that even a small error term will cause the integral component to increase slowly. The integral response will continually increase over time unless the error is zero, so the effect is to drive the Steady-State error to zero. Steady-State error is the final difference between the process variable and set point. A phenomenon called integral windup results when integral action saturates a controller without the controller driving the error signal toward zero meaning difference between input and output should be as low as possible.

3.2.3. Derivative response

The derivative component causes the output to decrease if the process variable is increasing rapidly. The derivative response is proportional to the rate of change of the process variable. Increasing the derivative time parameter will cause the control system to react more strongly to changes in the error term and will increase the speed of the overall control system response. Most practical control systems use very small derivative time, because the derivative response is highly sensitive to noise in the process variable signal. If the sensor feedback signal is noisy or if the control loop rate is too slow, the derivative response can make the control system unstable. So we need to take a good care of this one**.**

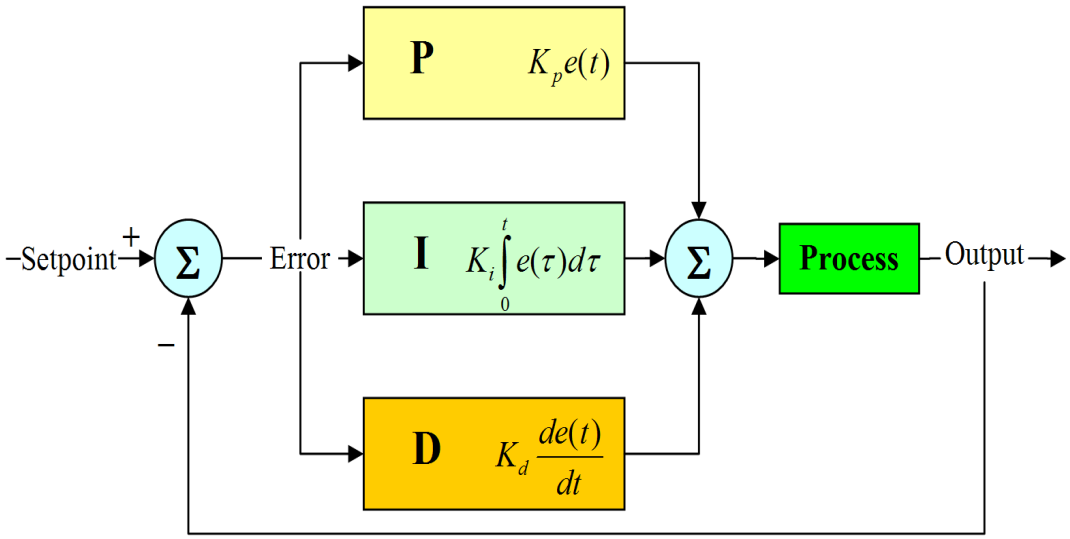

Figure 3.1: Block diagram of PID Controller

3.3: MODELLING OF A PID CONTROLLER

A PID Controller consists of three parts named as proportional gain, integral gain and derivative gain. And it calculates the measured data and desired data at the output. It tries to minimize the difference between them as well as have a good control over speed [11].

3.3.1: Governing equation

Three parameters are named as K_p, K_i and K_d are the gain of proportional, integral and derivate. So the governing equation would be as followed:

$$P_{out} = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t)$$

The "p" stands for proportional control gain, "i" for integral control gain and "d" stands for derivative control. This is also what is called a three term controller.

3.3.2: Effect of changing parameters

- A proportional controller (K_p) will have the effect of reducing the rise time and will reduce, but never eliminate, the Steady state error
- An integral control (K_i) will have the effect of eliminating the steady-state error, but it may make the transient response worse.
- A derivative control (K_d) will have the effect of increasing the stability of the system, reducing the overshoot and improving the transient response.

The following would give us a good idea of the effect of parameters in PID.

Parameters	Rise time	Overshoot	Settling time	Steady state error
K_p	Decrease	Increase	Small Change	Decrease
K_i	Decrease	Increase	Increase	Eliminate
K_d	Small Change	Decrease	Decrease	Small Change

Table 3.1: Effects of changing PID parameters on an output response [12].

3.4: WORKING PRINCIPLE OF PID

The job of a PID controller is to maintain the output at a level so that there is no difference (error) between the process variable (PV) and the set point (SP).

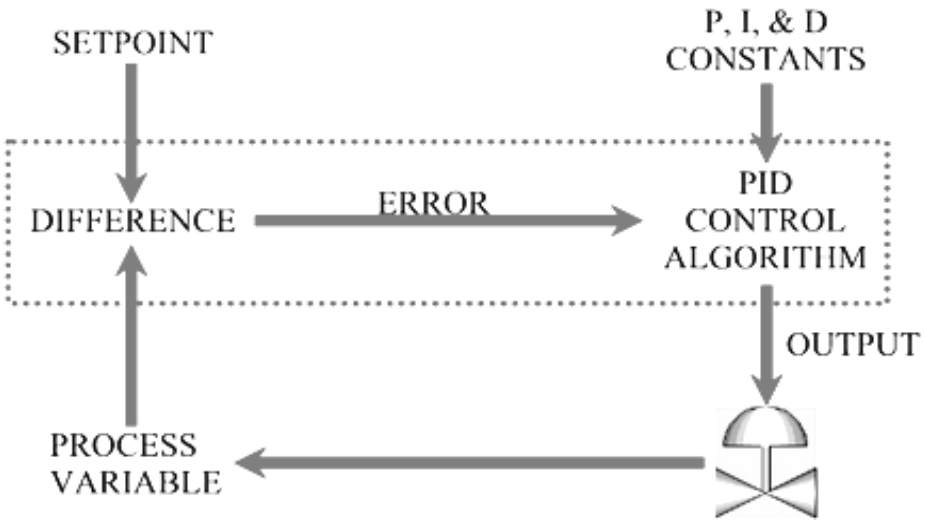

Figure 3.2: Working procedure of PID controller

In the diagram shown above the valve could be controlling the gas going to a heater, the chilling of a cooler, the pressure in a pipe, the flow through a pipe, the level in a tank or any other process control system. PID controller is looking at is the difference (or "error") between the PV and the SP. It looks at the absolute error and the rate of change of error. Absolute error means -- is there a big difference in the PV and SP or a little difference? Rate of change of error means -- is the difference between the PV or SP getting smaller or larger as time goes on. When there is a "process upset", meaning, when the process variable or the set point quickly changes - the PID controller has to quickly change the output to get the process variable back equal to the set point. If you have a walk-in cooler with a PID controller and someone opens the door and walks in, the temperature (process variable) could rise very quickly. Therefore the PID controller has to increase the cooling (output) to compensate for this rise in temperature. Once the PID controller has the process variable equal to the set point, a good PID controller will not vary the output. You want the output to be very steady (not changing). If the valves (motor or other control element) are constantly changing, instead of maintaining a constant value, this could case more wear on the control element. So there are these two contradictory goals. Fast response (fast change in output) when there is a "process upset", but slow response (steady output) when the PV is close to the set point.

3.5: APPLICATIONS OF PID

Some applications may require using only one or two actions to provide the appropriate system for control. This is achieved by setting the other parameters to zero. A PID controller will be called a PI, PD, P or I controller in the absence of the respective control actions. PI controllers are fairly common, since derivative action is sensitive to measurement noise, whereas the absence of an integral term may prevent the system from reaching its target value due to the control action.

3.6: LIMITATIONS OF PID CONTROLLERS

When used alone, PID controllers can give poor performance when the PID loop gains must be decreased so that the control system does not overshoot and/or oscillate about the control set point value. They also have difficulties in the presence of non-linearities,

may trade-off regulation versus response time, do not react to changing process behavior (say, the process changes after it has warmed up), and have lag in responding to large disturbances.

Another limitation is the fact the output response depends entirely on the values of the PID parameters (K_p, K_i, K_d). The exact values of these parameters for which the best output response can be obtained in a few ways (for example the Ziegler–Nichols method), but all of which are extremely tedious and contains complicated mathematics.

Therefore, a method is needed to approximately auto-tune these parameter values to get a good output response. Self-tuned fuzzy PID controller provides a way to tune these parameters to obtain an output response that very closely resembles the ideal response, and this is shown in details in later chapters.

CHAPTER 4

SIMULATION OF DC MOTOR CONTROLLED BY PID CONTROLLER

4.1: PID CONTROLLER IN SIMULINK

The PID controller model shown in chapter 3 is modified for the sake of the simulation in Simulink with a fuzzy logic controller (FLC) which will be shown in chapter 8 (Fig. 4.1). The modified model can also be used for controlling the speed of a DC motor on its own without the FLC. This is done in this chapter in order to show with the help of simulation why PID controller is necessary and on which aspects its performance can be improved by a FLC.

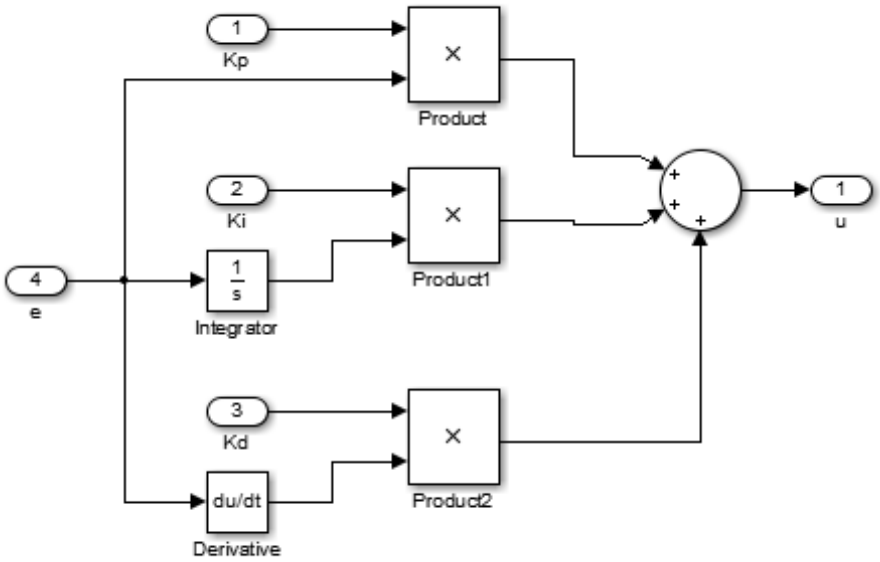

Figure 4.1: Modified block diagram of PID controller

4.2: DC MOTOR CONTROLLED BY PID CONTROLLER

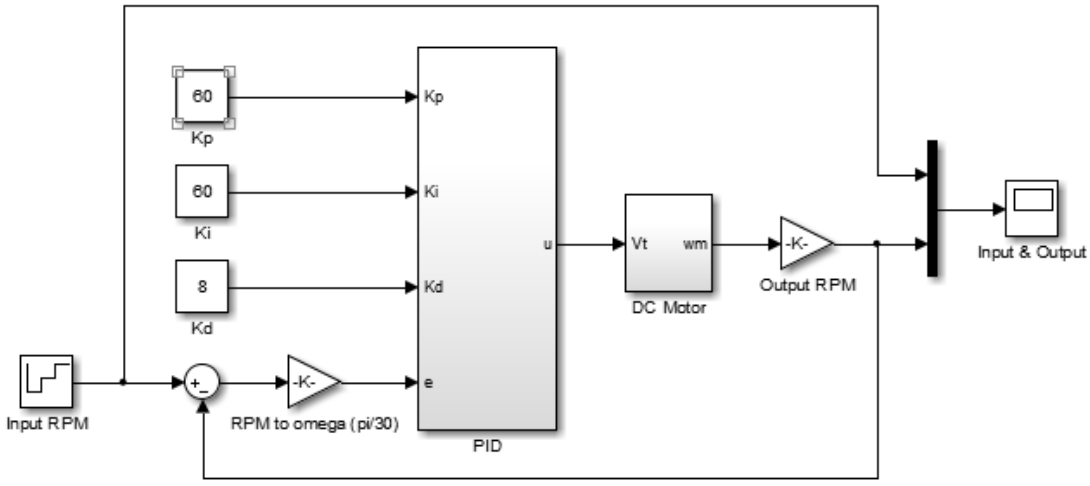

Figure 4.2: Simulink block diagram of DC motor controlled by PID controller

4.3: SIMULATION RESULTS

The simulation is run using different values of K_p, K_i, and K_d to show how they affect the output speed. The input speed is the reference speed that is desired to be run at. Three different speeds are chosen – 1000 r.p.m., 1500 r.p.m. and 2000 r.p.m. These speeds are fed into the controller as a staircase function for simplicity. The simulation is run for 30 seconds, and as before, initially there is no load torque, and a load torque of 100 N-m is added at 15 seconds which is kept till the end of simulation.

The simulation is run for a few random values of K_p, K_i and K_d. As our primary objective is the detailed analysis of a fuzzy PID controller and not a PID controller on its own, we only show simulation results for two such sets of the PID parameters.

For K_p = 60, K_i = 60, K_d = 8

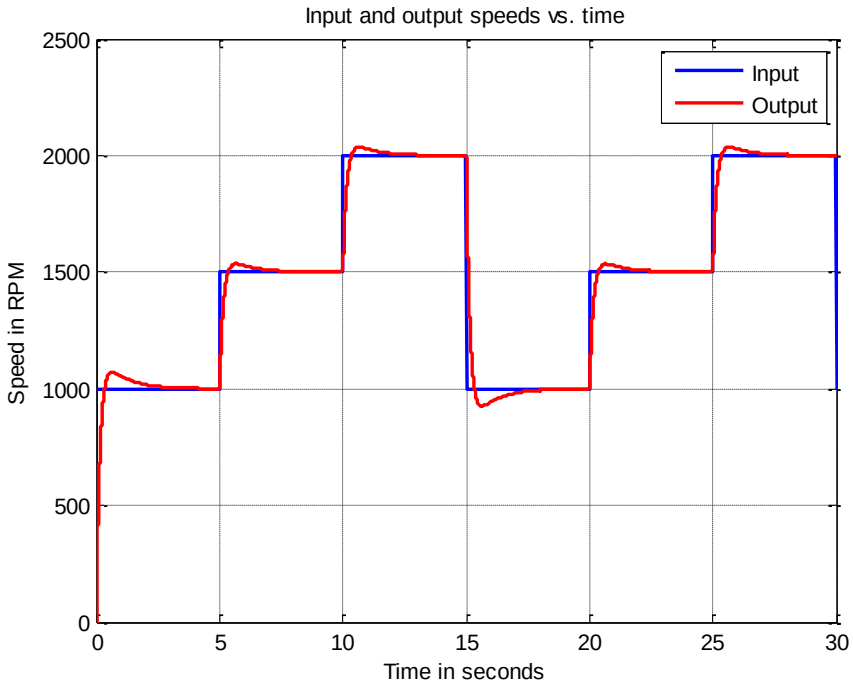

Figure 4.3: Speed-time characteristics of DC motor controlled by PID controller with K_p = 60, K_i = 60, K_d = 8

For K_p = 10, K_i = 10, K_d = 1.6

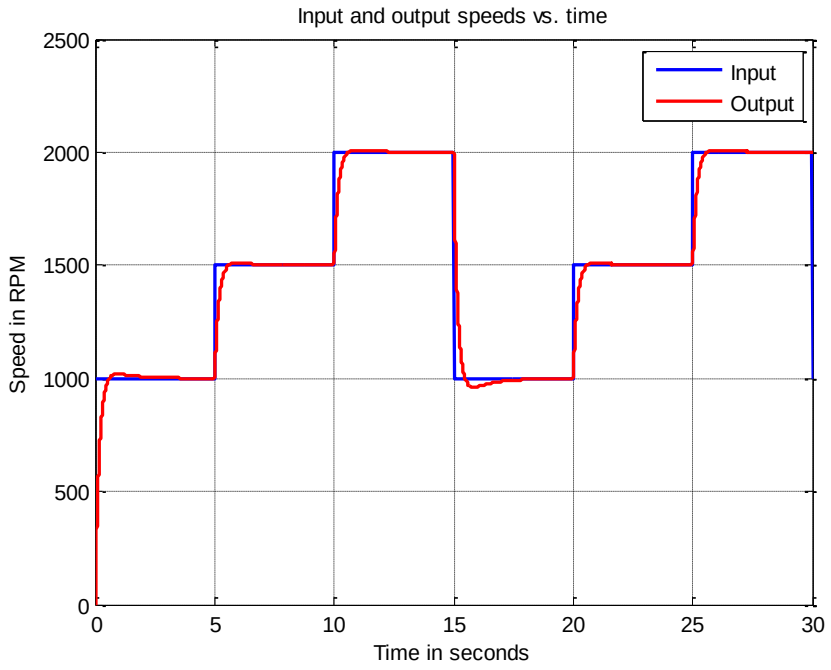

Figure 4.4: Speed-time characteristics of DC motor controlled by PID controller with K_p = 10, K_i = 10, K_d = 1.6

It is seen that adding a PID controller greatly changes the response of the motor. The overshoot and undershoot, though still present, is much softer and does not possess much danger. However, this depends on the values of the PID parameters. Also, there is considerable rise time, fall time and settling time. To make the output speed as close as possible to the input speed, the values of thee PID parameters have to be fine-tuned. This is done by the fuzzy logic controller (FLC) and is shown in details in chapter 8.

CHAPTER 5

THEORY OF FUZZY LOGIC

5.1: INTRODUCTION TO FUZZY THEORY

Fuzzy logic can be defined as the superset of typical logic or Boolean logic that has been extended in order to handle concept of partial truth - truth values between “entirely true" and "entirely false". This can be possible by introducing the concept of degree of membership [13].

Fuzzy logic (FL) is made up of multiple logic levels that enable us to define intermediate values between conventional evaluations like true/false, yes/no, high/low, etc. Concepts like ‘Medium Cold’ or ‘Very Hot’ can be represented mathematically and then processed by computers, which initiates a more human-like way of thinking in the programming of computers [14]. Fuzzy Logic is an efficient tool for the controlling and navigation of systems and complex industrial processes, in addition to household and entertainment electronics, plus other expert systems and applications like the classification of SAR data [15]. Fuzzy Logic was introduced in 1965 [16], [17], [18], by Lotfi A. Zadeh, professor of computer science at the University of California in Berkeley.

5.2: FUZZY SETS AND CRISP SETS

The fundamental idea of fuzzy systems is a fuzzy set (or subset). Crisp set is a familiar term in classical mathematics. For example, consider a number x. A subset S can be defined from the set R of all real numbers between 0 and 1, (e.g. all values for which $0 \leq x \leq 0.4$). The characteristic function of S, (i.e. it assigns a number 1 or 0 to each element in R, depending on whether the element is in the subset S or not) is shown in Fig. 5.1.

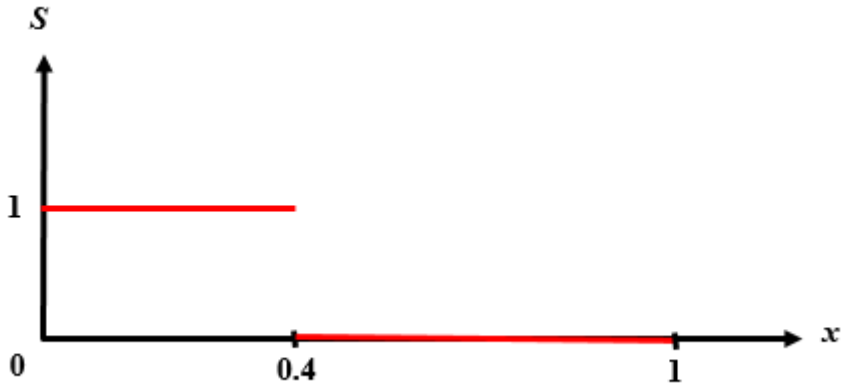

Figure 5.1: Characteristic function of a crisp set

The elements which have been assigned the number 1 can be interpreted as the elements that are in the set *S* and the elements which have assigned the number 0 as the elements that are not in the set *S*. For many areas of applications this idea is sufficient, but as we can see, it has less flexibility for some applications like classification of remotely sensed data analysis. For a much simpler example, consider the temperature of a room. According to Boolean logic, the temperature can either be 'HOT' or 'COLD'. If we define HOT as 30^0C and COLD as 10^0C, then according to Boolean logic, the room cannot have an intermediate temperate e.g. 20^0C, or 25^0C. Fuzzy logic provides a way to take intermediate values between 0 and 1, e.g. 20^0C is 0.5HOT or 0.5COLD, while 25^0C is 0.75HOT or 0.25COLD. This is shown in Fig. 5.2.

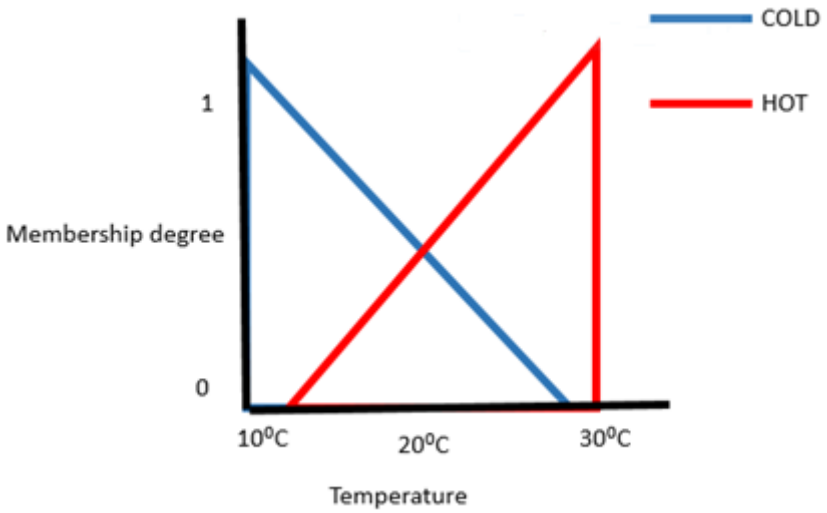

Figure 5.2: Example of membership function of a fuzzy set

Fuzzy sets aims to make computers more 'intelligent', therefore fuzzy sets allow more values between 0 and 1. In fact, infinite number of values can be allowed between the boundaries 0 and 1. For example, in the above room temperature problem the numbers 0.5, 0.25 and 0.75 have been used which are between 0 and 1. Numerous other values in this range like 0.1, 0.01 and 0.55 could be taken to make the system more precise.

The membership function means graphical representation of the magnitude of contribution of each input. Each of the inputs are processed as it is associated with a weighing number. The membership function also define functional overlap between inputs, and in the end establishes an output response. The rules use the input membership values as weighing factors to determine their influence on the fuzzy output sets of the final output conclusion.

The membership function, operating in this case on the fuzzy set of temperature, returns a value between 0.0 and 1.0. For example, a temperature of 20^0C has a degree of membership of 0.5 (see Fig. 2). It is essential to mention the difference between fuzzy logic and probability. Both fuzzy logic and probability operate over the same numeric range, and have similar values: 0.0 representing False (or non-membership), and 1.0 representing True (or full-membership). Nevertheless, there is a difference to be made involving the two statements: The probabilistic approach yields the natural-language statement, "There is a 50% chance that temperature is COLD or HOT," while the fuzzy terminology corresponds to room temperature's degree of membership within the set of temperature which is 0.50 for 25^0C." The semantic difference is substantial: the first interpretation supposes that temperature is HOT or COLD; it is just that we only have a 50% chance of knowing which set (HOT or COLD) it is in. On the contrary, fuzzy language assumes that temperature is "more or less" HOT/COLD, or in some other term corresponding to the value of 0.50.

5.3: OPERATIONS ON FUZZY SET

Basic operations can be introduced on fuzzy sets, as similar to the operations on crisp sets .We also want to intersect, unify and negate fuzzy sets. Minimum operator for the intersection and the maximum operator for the union of two fuzzy sets was suggested by L. A. Zadeh in his very first paper about fuzzy sets [19]. These operators match with

the crisp unification, and intersection if we only consider the membership degrees 0 and 1. For example, consider A is a fuzzy interval between 6 and 9 and B is a fuzzy number about 5 as shown in Fig. 5.3 below

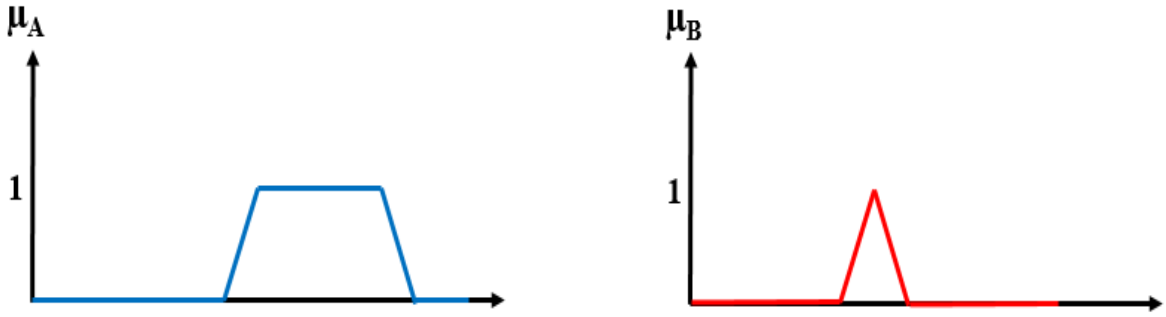

Figure 5.3: Example: fuzzy sets

In this case, the fuzzy set between 6 and 9 AND about 5 (intersection) is as shown in Fig. 5.4.

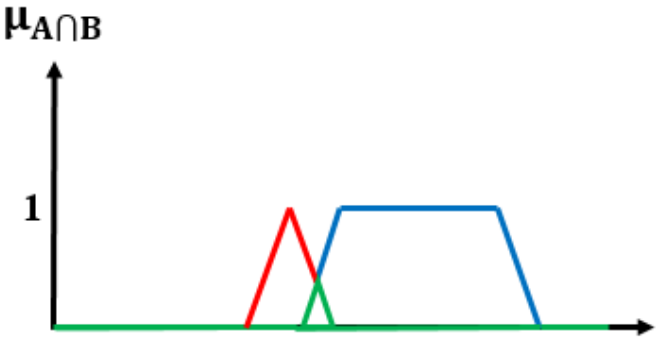

Figure 5.4: Example: fuzzy AND

The **intersection** of two fuzzy sets A and B is the largest fuzzy set within both A and B. The intersection is a logical AND operator written as A∩B or A.B, where the membership function is defined as:

$\mu_{A.B}(x) = \text{Min}\ [\mu_A(x), \mu_B(x)]$

The set of values between 6 and 9 OR about 5 (union) is shown in the Fig. 5.5.

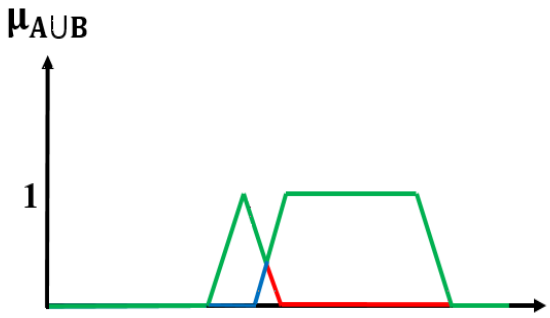

Figure 5.5: Example: fuzzy OR

The **union** of two fuzzy sets A and B is the smallest fuzzy set which include all articles in A or B or A and B. The union is a logical OR operator written as A+B or AUB, where the membership function is defined as:

$\mu_{A+B}(x) = \text{Max}\ [\mu_A(x), \mu_B(x)]$

The NEGATION (complement) of the fuzzy set A is shown in Fig. 5.6 below.

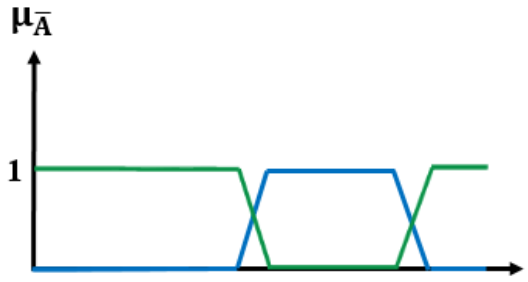

Figure 5.6: Example: fuzzy NEGATION

The **complement** of fuzzy set A is defined as

$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$

Complement of A = {x | x is not near an integer}

5.4: FUZZY CLASSIFICATION

One application of fuzzy theory is the fuzzy classifier. Fuzzy sets are described using linguistic variables to express detailed information in a very natural way.

Table 5.1 shows the general form of an example with 25 rules, The detailed information for these variables can be expressed as a rule like IF feature A is at LEVEL 2 AND feature B is at LEVEL 4 THEN ACTION = ACTION 17.

The rules can be combined in a table like in Table 1. The table in which all the fuzzy rules are combined is called a rule base table.

FEATURE A / **FEATURE B**	*LEVEL 1*	*LEVEL 2*	*LEVEL 3*	*LEVEL 4*	*LEVEL 5*
LEVEL 1	ACTION 1	ACTION 2	ACTION 3	ACTION 4	ACTION 5
LEVEL 2	ACTION 6	ACTION 7	ACTION 8	ACTION 9	ACTION 10
LEVEL 3	ACTION 11	ACTION 12	ACTION 13	ACTION 14	ACTION 15
LEVEL 4	ACTION 16	ACTION 17	ACTION 18	ACTION 19	ACTION 20
LEVEL 5	ACTION 21	ACTION 22	ACTION 23	ACTION 24	ACTION 25

Table 5.1: Example for a fuzzy rule base. Rules read as example: RULE No.1: **IF** A is at level 3 **AND** B is at level 5 **THEN** do action number 23.

The "*levels*" mentioned in the above example are replaced when making actual fuzzy models with words like *close, heavy, light, big, small, smart, fast, slow, hot, cold, tall, short, near, far* etc. depending upon the system. Because of this incredible flexibility and simplicity, the system is said to "think like a human".

Linguistic rules describing the control system involves two parts; an antecedent block (between the IF and THEN) and a consequent block (following THEN). Evaluation of all possible input combinations may not be required because some combinations may never take place depending on the system. By making this type of evaluation, usually done by an experienced operator, fewer rules can be evaluated, thus simplifying the processing logic and perhaps even improving the fuzzy logic system performance.

The inputs in this example are combined logically using the AND operator to produce output response values for all expected inputs. The active decisions are then combined into a logical sum for each membership function.

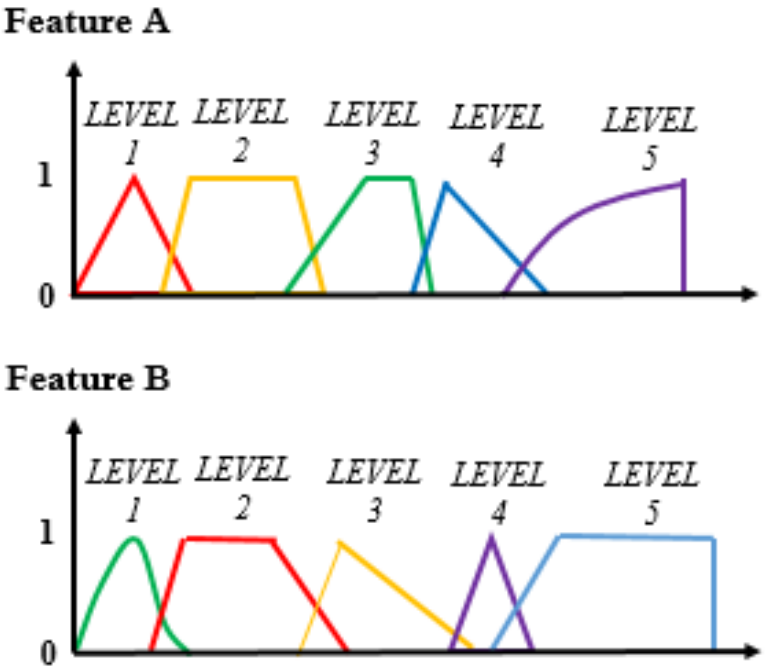

Figure 5.7: Example: linguistic variables and membership functions

The fuzzy outputs for all rules are finally aggregated to one fuzzy set. To obtain a crisp decision from this fuzzy output, the fuzzy set is defuzzified, where one representative value is chosen as the final output. There are several heuristic methods (defuzzification methods), one of them is to take the center of gravity of the fuzzy set as shown in Fig. 5.8, which is very commonly used for fuzzy sets. In case of discrete values with singletons usually the maximum-method is used where the point with the maximum singleton is chosen.

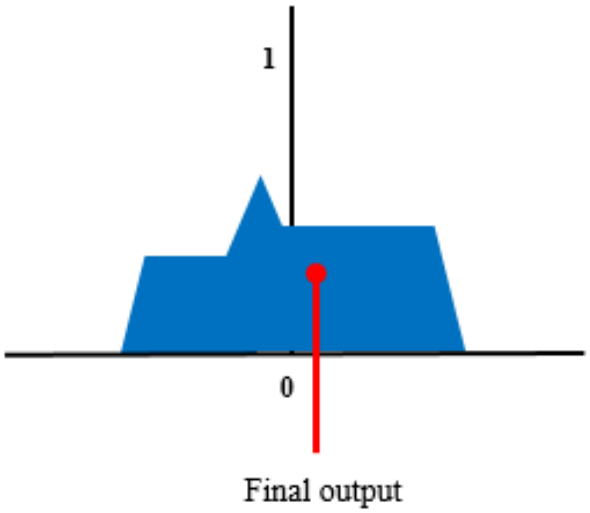

Figure 5.8: Defuzzification using the center of gravity approach

5.5: BASIC TERMINOLOGY IN FUZZY LOGIC

5.5.1: Degree of membership (μ)

The degree to which a crisp variable belongs to a fuzzy set is called degree of membership. It can be represented either as a fractional value ranging from 0.0 to 1.0 or as percentage ranging from 0% to 100%.

5.5.2: Membership function

A membership function (MF) is usually expressed graphically and tends to demonstrate how completely a crisp variable belongs to a fuzzy set.

Membership functions are determined by utilizing the knowledge of human expert data gathered from various sensors.

With the aim of describing fuzzy membership function, designers choose many different shapes based on their preference and experience. Mainly four types of membership functions are used:

1: Trapezoidal MF

2: Triangular MF

3: Gaussian MF

4: Generalized bell MF

Among these functions the most popular shapes are triangular and trapezoidal because these shapes represent the designer's idea easily and require low computation time.

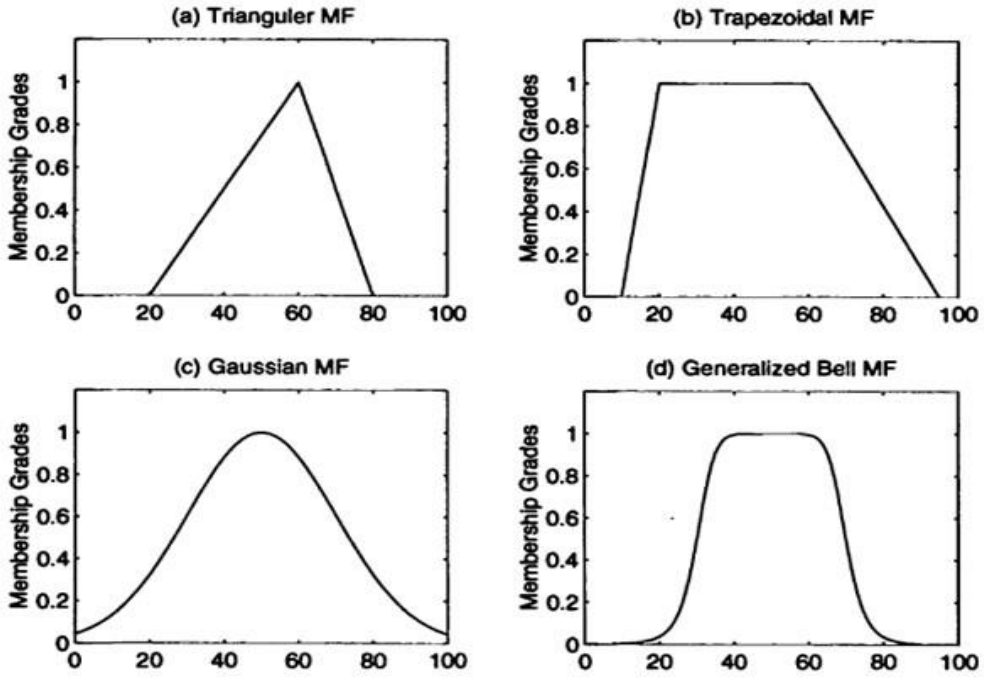

Figure 5.9: Shapes of different membership functions [20]

5.5.3: Crisp variable

A physical variable that can be measured using instruments and can be assigned a crisp value is called a crisp variable. For example, a distance of 30m, or a current of 1A etc.

5.5.4: Linguistic variable

If a variable can take words in natural languages as its values, it is called a linguistic variable, where the words are characterized by fuzzy sets defined in universe of discourse in which the variable is defined.

5.6: HOW FUZZY LOGIC WORKS

Fuzzy logic need some numerical parameters so that it can detect and operate changes in input such as significant error and significant rate-of-change-of-error, but precise values of these numerical parameters are generally not critical unless very responsive performance is required in which case experimental tuning would determine them. For instance, think of a simple temperature control system using a single temperature feedback sensor whose data is subtracted from the reference signal to compute "error" and then time-differentiated yielding the error slope or rate-of-change-of-error, henceforth called "error-dot". Error might have units of degrees F and a small error considered to be 3F while a large error is 8F. The "error-dot" might then have units of degrees/min with a small error-dot being 8F/min and a large one being 18F/min. These values need not to be symmetrical and can be "tweaked" once the system is in operation to facilitate optimize performance.

5.7: APPLICATION OF FUZZY LOGIC IN TECHNICAL SYSTEMS

A small number of areas where fuzzy sets have already been functional to a variety of degrees are: communication engineering, production engineering, image and speech understanding, reliability analysis of large-scale software systems, transportation, earthquake studies, controller applications, system identification, consumer electronics and robotics.

CHAPTER 6

THE FUZZY LOGIC CONTROLLER

6.1: FUZZY LOGIC CONTROLLER

Fuzzy logic control (FLC) as defined earlier is a control algorithm based on a linguistic control strategy, which is obtained from detailed information into an auto-control scheme. The operation of a FLC is based on qualitative knowledge about the system being controlled. Any complicated mathematical calculation like those in other control systems are not necessary for a FLC. It only uses simple mathematical calculation to simulate the detailed information whereas other control systems use complicated mathematical calculation to build a model of the controlled plant,

The requirement for the application of a FLC arises mainly in situations where precise definitions are not possible like:

- Where the technological process is described only in word form, not in analytical form.
- Precise identification of the process parameters is not possible
- The description of the process is so complex that it is more practical to express its description in simple language words.
- The controlled technological process has a "fuzzy" character.

A fuzzy logic controller consists of four main components as shown in Fig. 6.1:

a) Fuzzification
b) Rule base
c) Inference engine
d) Defuzzification

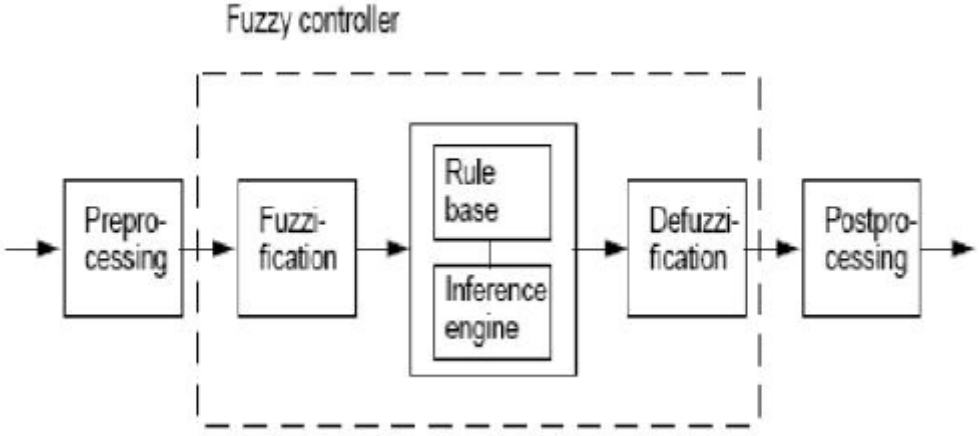

Figure 6.1: Structure of fuzzy logic controller [21]

The fuzzy logic control method is explained in the following algorithm:

1. Define the linguistic variables and terms (initialization)
2. Construct the membership functions (initialization)
3. Construct the rule base (initialization)
4. Convert crisp input data to fuzzy values using the membership functions (fuzzification)
5. Evaluate the rules in the rule base (inference)
6. Combine the results of each rule (inference)
7. Convert the output data to non-fuzzy values (defuzzification)

At first, a crisp set of input data are collected and transformed to a fuzzy set by means of fuzzy linguistic variables, fuzzy linguistic terms and membership functions. This is known as fuzzification. Subsequently, based on a set of rules an inference is made. Finally, the resulting fuzzy output is converted to a crisp output using the membership functions, in the defuzzification step.

6.2: FUZZIFICATION

The first step in designing a fuzzy logic controller is to settle on which state variables representing the system dynamic performance must be taken as the input signal to the controller. Instead of numerical variables, fuzzy logic utilizes linguistic variables. Fuzzification is the procedure of converting a numerical variable (real number or crisp variables) into a linguistic variable (fuzzy number). This is accomplished with the help of different types of fuzzifiers. There are normally three types of fuzzifiers used for the fuzzification process:

1. Singleton fuzzifier
2. Gaussian fuzzifier
3. Trapezoidal or triangular fuzzifier

6.3: RULE BASE

Based on the knowledge obtained from the control rules and designation of linguistic variables, a decision making logic carries out fuzzy control action, simulating a human decision procedure [22]. All the rules are in “If Then” format and the ‘If’ side is called the condition or antecedent and the ‘Then’ side is called the conclusion or consequent.

The computer is capable of executing the rules and computes a control signal depending on the measured inputs error (e) and change in error (de). In a rule based controller the control strategy is stored more or less in the form of natural language. For a non-specialist end user rule base controller is easy to understand and easy to maintain. Conventional techniques could be implemented for an equivalent controller [23].

6.4: INFERENCE ENGINE

To process the rules, cases, objects or other type of knowledge and knowledge based on the facts of a given situation a software code called the inference engine is used. When a problem occurs that needs to be solved by logic rather than mathematical know-how, a series of inference steps is taken which may include deduction, association, recognition, and decision making. An inference engine is an information processing system (such as a computer program) that systematically employs these inference steps similar to that of a human brain.

6.5: DEFUZZIFICATION

Defuzzification is the reverse of fuzzification. Fuzzy logic controller (FLC) produces required fuzzy output in a linguistic variable form (fuzzy number). These linguistic variables have to be converted to crisp output according to real world requirements. The most common methods of defuzzification are as follows [24]:

1. Center of gravity (COG)
2. Bisector of area (BOA)
3. Mean of maximum (MOM)

6.5.1: Center of gravity (COG)

In case of discrete sets COG is called center of gravity for singletons (COGS) where the crisp control value is the abscissa of the center of gravity of the fuzzy set calculated as follows:

$$u_{COGS} = \frac{\Sigma_i \mu_c(x_i) x_i}{\Sigma_i \mu_c(x_i)}$$

Where x_i is a point in the universe of the conclusion (I = 1, 2, 3…) and $\mu_c(x_i)$ is the membership degree of the resulting conclusion set. In case of continuous sets summations are replaced by integrals.

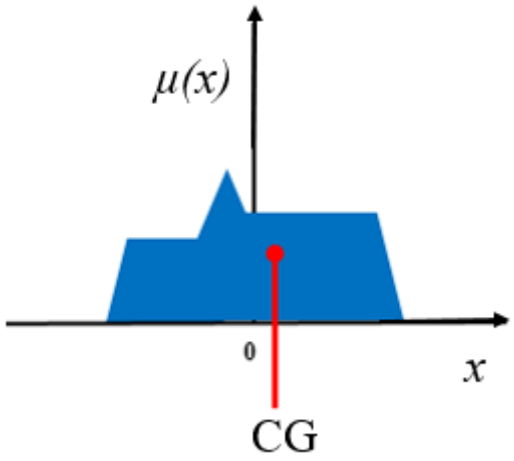

Figure 6.2: Illustration of center of gravity method

6.5.2: Bisector of area (BOA)

In the bisector of area (BOA) defuzzification method, the abscissa of the vertical line that divides the area of the resulting membership function into two equal areas is calculated. For discrete sets, u_{BOA} is the abscissa x_j that minimizes

$$\left| \sum_{i=1}^{j} \mu_c(x_i) - \sum_{i=j+1}^{i_{max}} \mu_c(x_i) \right|, \qquad i < j < i_{max}$$

Here i_{max} is the index of the largest abscissa x_{imax}. BOA is a computationally complex method.

6.5.3: Mean of maximum (MOM)

The crisp value is chosen from the point with the highest membership in this method. There may be many points in the overall implied fuzzy set which have maximum membership degree value. The mean value of these points is calculated in common cases. This method is called mean of maximum (MOM) and the crisp value is calculated as follows:

$$u_{MOM} = \frac{\sum_{i \in I} x_i}{|I|}, \qquad I = \{i | \mu_c(x_i) = \mu_{max}\}$$

Here I is the (crisp) set of indices i where $\mu_c(x_i)$ reaches its maximum μ_{max}, and $| I |$ is its cardinality (the number of members).

It is important to decide on four major issues in order to implement a fuzzy logic controller:

1. Number of fuzzy sets that constitute linguistic variables.
2. Mapping of the measurements onto the support sets.
3. Control protocol that determines the controller behavior.
4. Shape of membership functions.

CHAPTER 7

DESIGN OF FUZZY LOGIC CONTROLLER

7.1: FUZZY LOGIC CONTROLLER

Inputs which are fed to the Self-tuning Fuzzy PID Controller are speed error "$e(t)$" and change in speed error or rate "$de(t)$". Inputs are illustrated in figure are expressed by

$e(t) = \omega_r(t) - \omega_a(t)$

$de(t) = e(t) - e(t - 1)$

PID parameters "K_p"," K_i"," K_d" are auto-tuned using fuzzy control rules on-line, which constitute a self-tuning fuzzy PID controller as shown in Fig. 7.1.

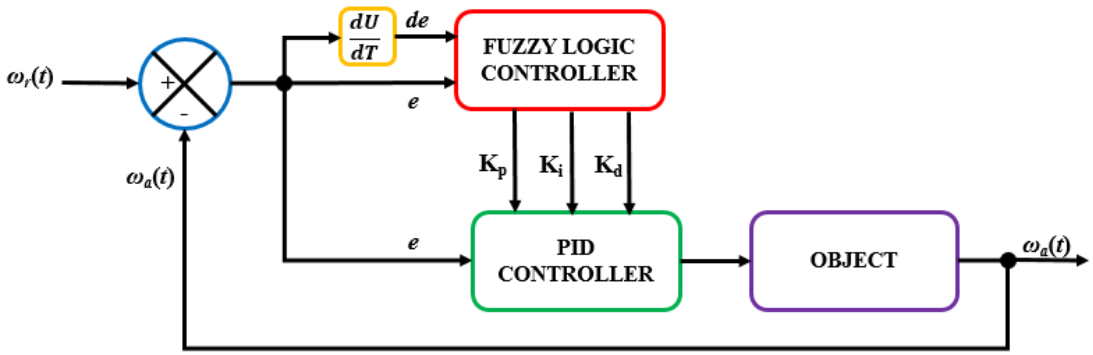

Figure 7.1: The structure of self-tuning fuzzy PID controller

The purpose of fuzzy self-tuning of the PID parameters is to find the fuzzy relationship between the three parameters of PID and "*e*" and "*de*", and in accordance to the principle of fuzzy control, to adjust the three parameters so as to meet different requirements for control parameters when "*e*" and "*de*" are different, and also to enable the control unit have a good dynamic and steady-state performance [25].

7.1.1: Adjusting fuzzy membership functions and rules

The rules and membership functions are adjusted with the purpose of improving the performance of fuzzy logic controller (FLC). The area of membership functions near zero (ZE) region is made narrower to produce finer control resolution during the adjustment of the membership functions. On the contrary, faster control response is achieved by making the area far from ZE region wider. Also by changing the stiffness of rules, performance can be improved [26]. The rule bases comprising of the 25 rules for each of the PID parameter is made by studying the effect on rise time and fall time,

maximum percentage overshoot and undershoot, settling time and steady state error while varying K_p, K_i and K_d.

7.2: DESIGN OF MEMBERSHIP FUNCTION (MF)

The parameters are expressed in the same format as they are written in the Membership Function Editor of the Fuzzy Logic Toolbox in MATLAB.

7.2.1: Input variables

All the membership functions of the input variables (speed error and change in speed error) are triangular in shape.

Fuzzy sets of speed error (*e*) variable

Fuzzy set (Label)	*Description*	*Parameters*
Negative Large (NL)	Large speed difference in negative direction	[-20 -20 40]
Negative Small (NS)	Small speed difference in negative direction	[10 40 100]
Zero (ZE)	Speed difference is zero	[40 70 100]
Positive Small (PS)	Small speed difference in positive direction	[40 100 130]
Positive Large (PL)	Large speed difference in positive direction	[100 160 160]

Table 7.1: Membership function of speed error

Fuzzy sets of change in speed error (*de*) variable

Fuzzy set (Label)	*Description*	*Parameters*
Negative Large (NL)	Large error difference in negative direction	[-1300 -1300 -800]
Negative Small (NS)	Small error difference in negative direction	[-1050 -800 -300]
Zero (ZE)	Error difference is zero	[-800 -550 -300]
Positive Small (PS)	Small error difference in positive direction	[-800 -300 -50]
Positive Large (PL)	Large error difference in positive direction	[-300 200 200]

Table 7.2: Membership function of change in speed error

7.2.2: Output variables

All the membership functions of the output variables (K_p, K_i, K_d) are trapezoidal in shape.

Fuzzy sets for K_p

Fuzzy set (Label)	*Parameters*
Positive Very small (PVS)	[0 0 3.614 16.47]
Positive Small (PS)	[1.957 8.352 13.77 26.64]
Positive Medium small (PMS)	[11.81 18.2 23.56 36.2]
Positive Medium (PM)	[21.8 28.2 31.8 38.2]
Positive Medium Large (PML)	[23.52 36.4 41.8 48.2]
Positive Large (PL)	[33.8 46.44 51.8 58.2]
Positive Very Large (PVL)	[43.68 56.4 60 60]

Table 7.3: Membership function proportional gain K_p

Fuzzy sets for K_i

Fuzzy set (Label)	*Parameters*
Positive Very Small (PVS)	[0 0 3.644 16.6]
Positive Small (PS)	[1.799 8.196 13.61 26.48]
Positive Medium small (PMS)	[11.8 18.2 23.6 36.48]
Positive Medium (PM)	[21.8 28.2 31.8 38.2]
Positive Medium Large (PML)	[23.8 36.44 41.8 48.24]
Positive Large (PL)	[33.68 46.44 51.84 58.2]
Positive Very Large (PVL)	[43.68 56.4 60 60]

Table 7.4: Membership function integral gain K_i

Fuzzy sets for K_d

Fuzzy set (Label)	*Parameters*
Positive Very Small (PVS)	[0 0 0.09792 0.446]
Positive Small (PS)	[0.04804 0.2188 0.3632 0.7058]
Positive Medium small (PMS)	[0.3148 0.4854 0.629 0.969]
Positive Medium (PM)	[0.5814 0.752 0.8484 1.02]
Positive Medium Large (PML)	[0.6346 0.9718 1.115 1.285]
Positive Large (PL)	[0.9012 1.238 1.381 1.552]
Positive Very Large (PVL)	[1.164 1.504 1.6 1.6]

Table 7.5: Membership function derivative gain K_d

7.3: DESIGN OF FUZZY RULES

Rule base for tuning K_p

de / **e**	*NL*	*NS*	*ZE*	*PS*	*PL*
NL	PVL	PVL	PVL	PVL	PVL
NS	PML	PML	PML	PL	PVL
ZE	PVS	PVS	PS	PMS	PMS
PS	PML	PML	PML	PL	PVL
PL	PVL	PVL	PVL	PVL	PVL

Table 7.6: Fuzzy rule base table for K_p

Rule base for tuning K_i

de / e	*NL*	*NS*	*ZE*	*PS*	*PL*
NL	PVL	PVL	PVL	PVL	PVL
NS	PML	PML	PML	PL	PVL
ZE	PVS	PVS	PS	PMS	PMS
PS	PML	PML	PML	PL	PVL
PL	PVL	PVL	PVL	PVL	PVL

Table 7.7: Fuzzy rule base table for K_i

Rule base for tuning K_d

de / e	*NL*	*NS*	*ZE*	*PS*	*PL*
NL	PVL	PVL	PVL	PVL	PVL
NS	PML	PML	PML	PL	PVL
ZE	PVS	PVS	PS	PMS	PMS
PS	PML	PML	PML	PL	PVL
PL	PVL	PVL	PVL	PVL	PVL

Table 7.8: Fuzzy rule base table for K_d

CHAPTER 8

SIMULATION OF SELF-TUNED FUZZY PID CONTROLLED DC MOTOR IN SIMULINK

8.1: FUZZY PID CONTROLLER IN SIMULINK

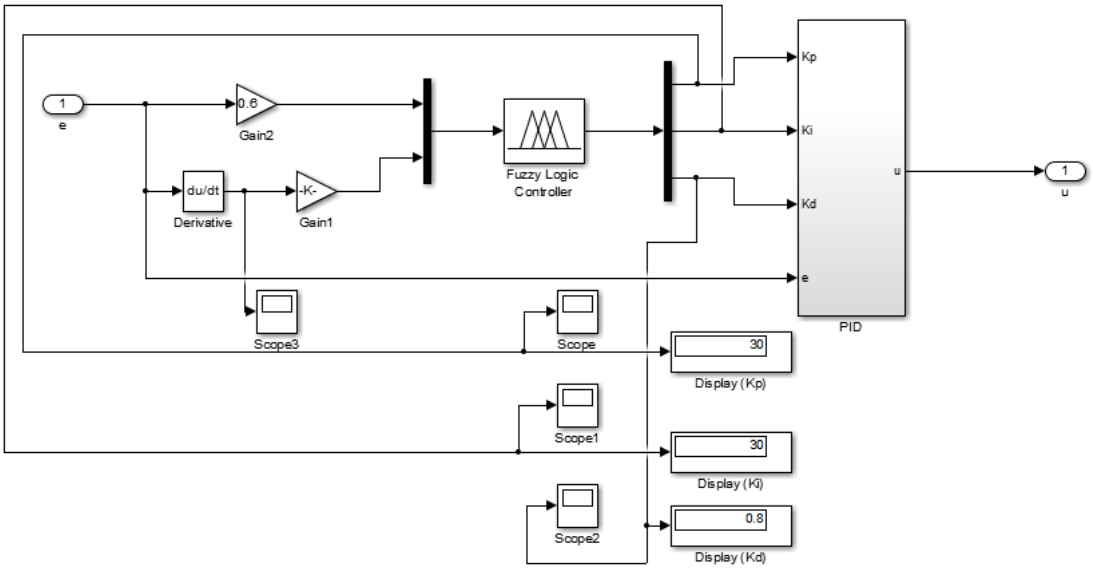

Figure 8.1: Simulink block diagram of fuzzy PID controller

The gains are used to scale down the wide range of speed error and change in speed error for the fuzzy logic controller. Gain1=0.75, Gain2=0.6.

8.2: DC MOTOR CONTROLLED BY FUZZY PID CONTROLLER IN SIMULINK

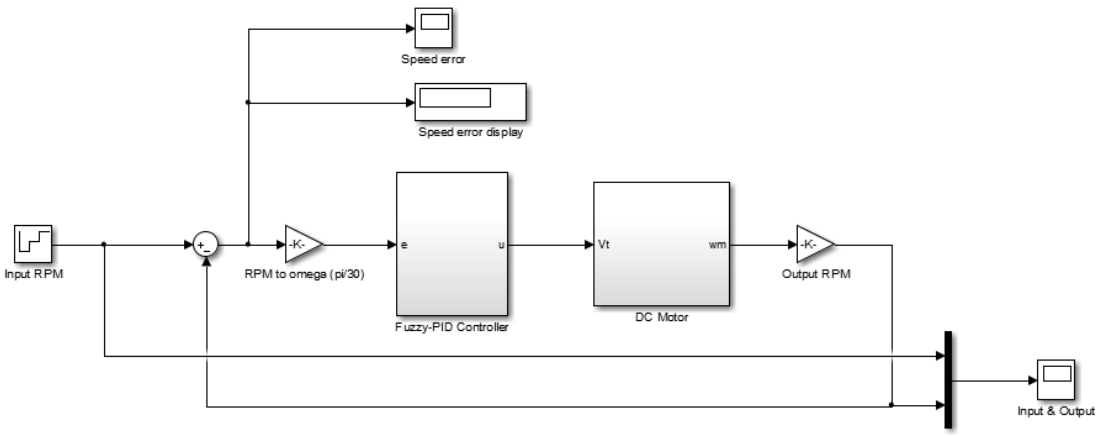

Figure 8.2: Simulink block diagram of DC motor controlled by fuzzy PID controller

8.3: SIMULATION RESULTS

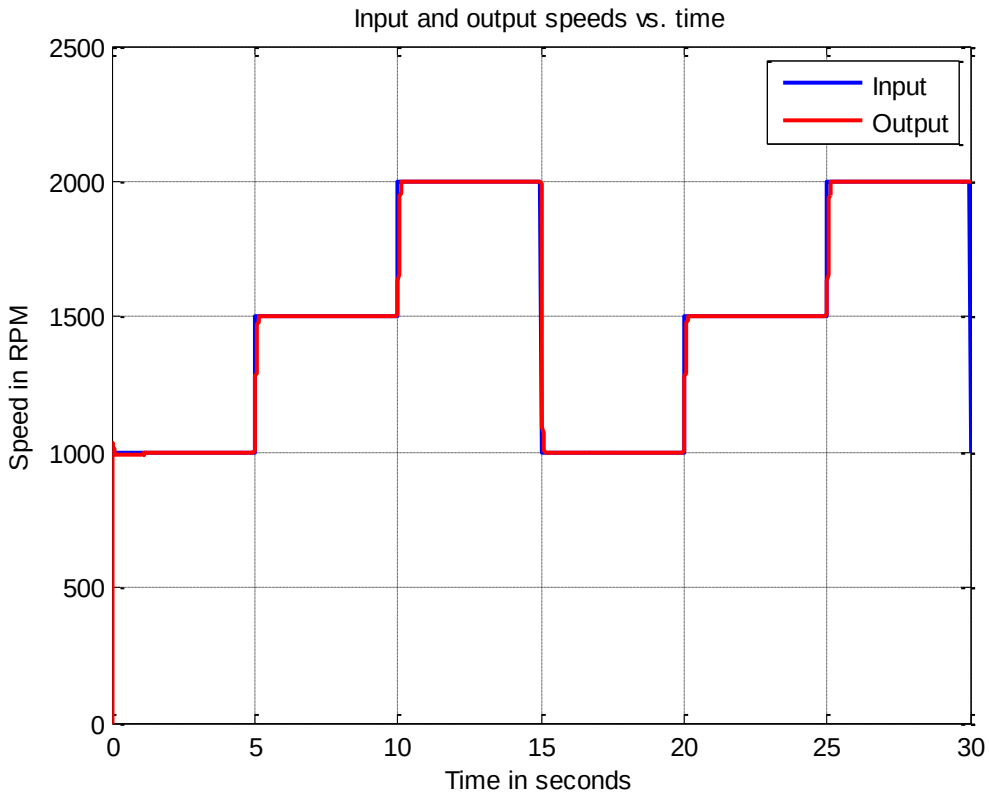

Figure 8.3: Speed-time characteristics of DC motor controlled by self-tuned fuzzy PID controller

The same speeds that were fed into the controller during simulation of PID controlled DC motor as input are used here, i.e., 1000 r.p.m., 1500 r.p.m. and 2000 r.p.m. as a staircase function with sample time of 5 seconds. The simulation is run for 30 seconds, where the motor runs at no-load condition for the first 15 seconds, and at a load of 100 N-m from 15 seconds to the end of simulation. It is observed that the output speed follows the input (reference) speed very closely with very small overshoot and undershoot, both at unloaded and loaded conditions. Detailed findings of rise time, fall time, percentage maximum overshoot and undershoot, settling time and steady state error are provided in chapter 9.

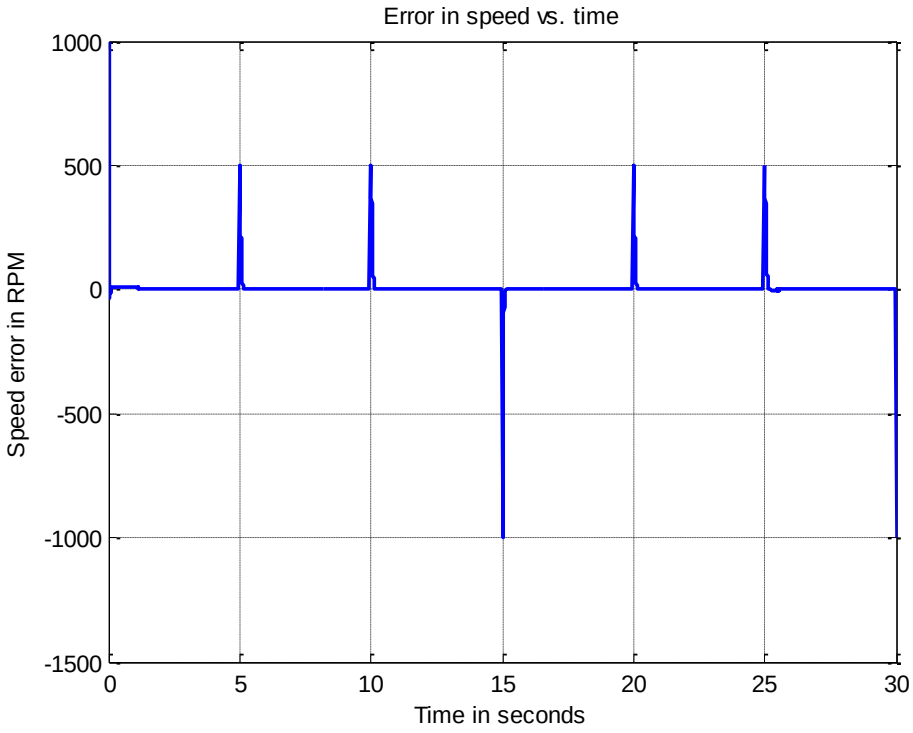

Figure 8.4: Variation of speed error with time

Speed error is defined as the difference between input (reference) and output speeds. The difference is significant only when the input is changed, which causes the spikes in Fig 8.4. At all other times, since the output speed is almost equal to the input speed, the speed error is zero.

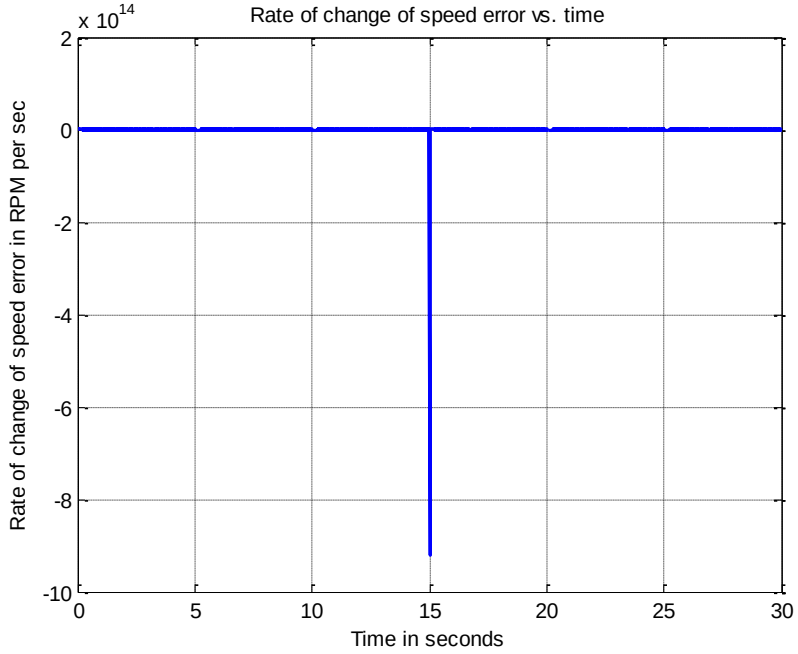

Figure 8.5: Variation of rate of change of speed error with time

The system is modelled for a case where input speed changes as a staircase function. As such, the changes in case of speed error when the input speed changes from 1000 r.p.m. to 1500 r.pm. and from 1500 r.p.m. to 2000 r.p.m. are not large enough to appear in the simulation result. However, the sudden decrease in speed from 2000 r.p.m. to 1000 r.p.m. causes quite a large rate of change in speed error, and this is sown by the spike in Fig. 8.5.

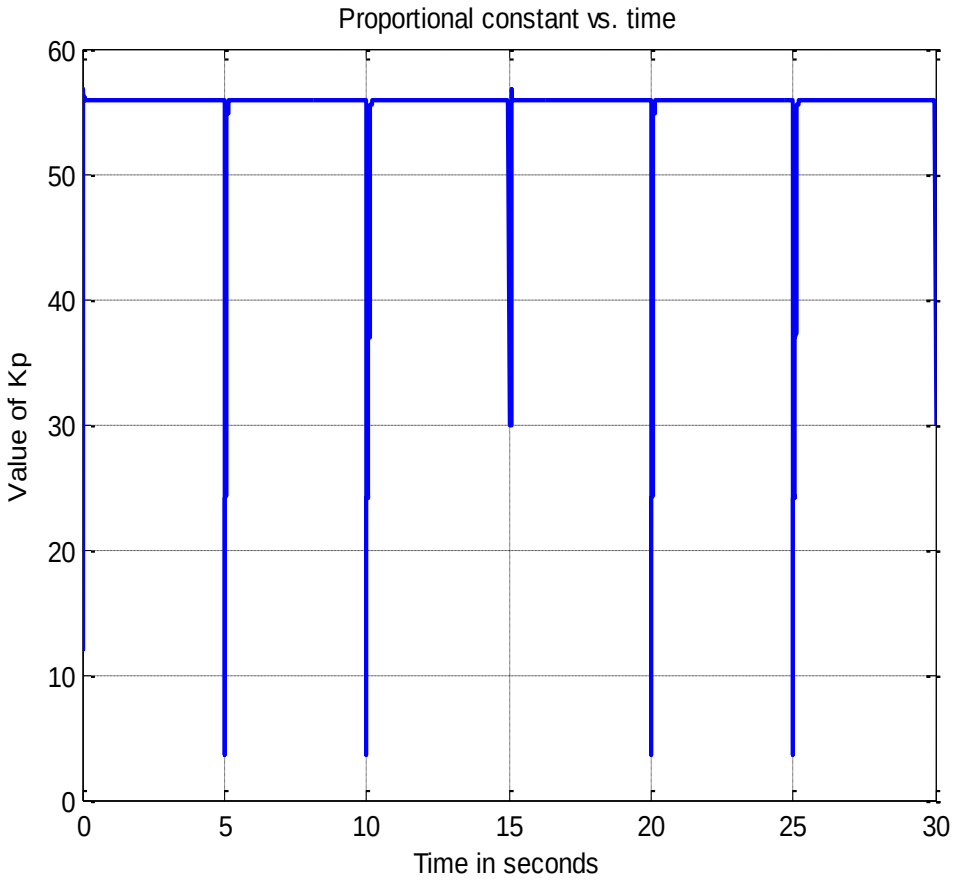

Figure 8.6: Variation of K_p with time

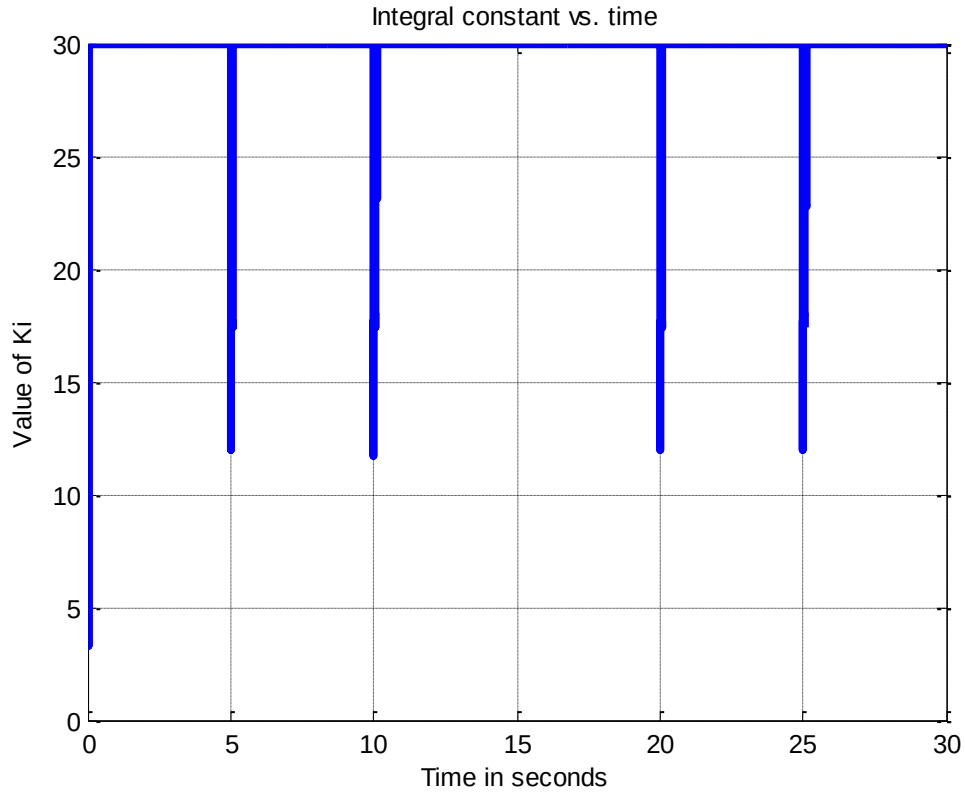

Figure 8.7: Variation of K_i with time

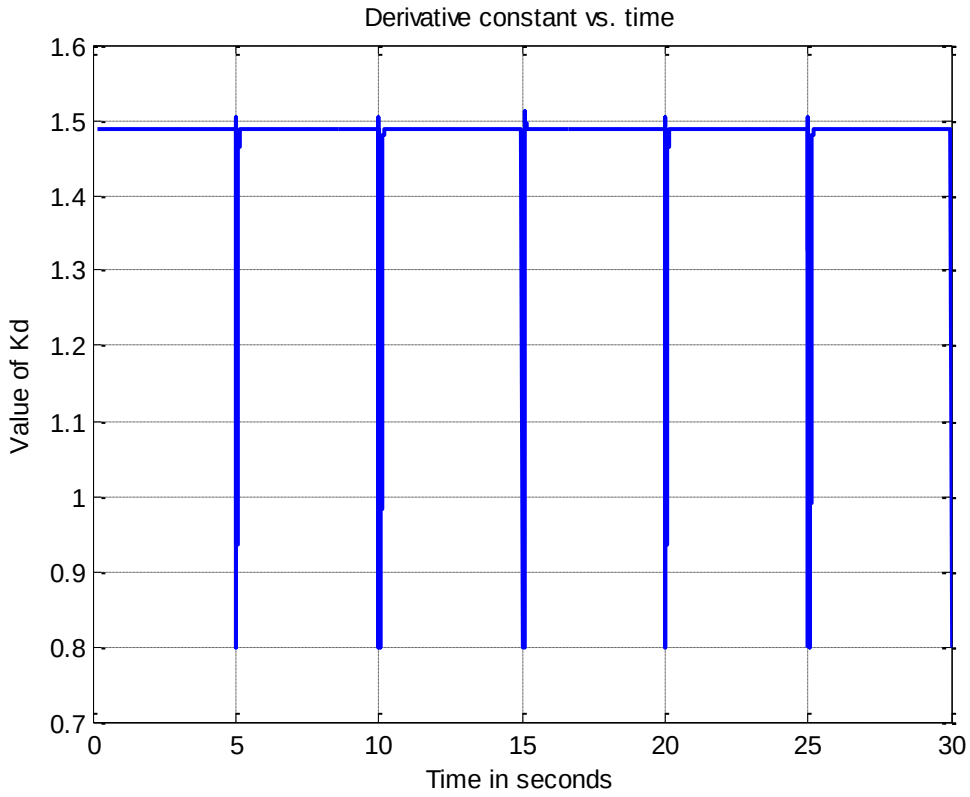

Figure 8.8: Variation of K_d with time

The values of K_p, K_i and K_d only changes when the output speed needs to change its value, i.e., when the input speed changes. This can be noticed from the spikes in Fig. 8.6, 8.7 and 8.8. At all other times, they remain constant to ensure constant output speed.

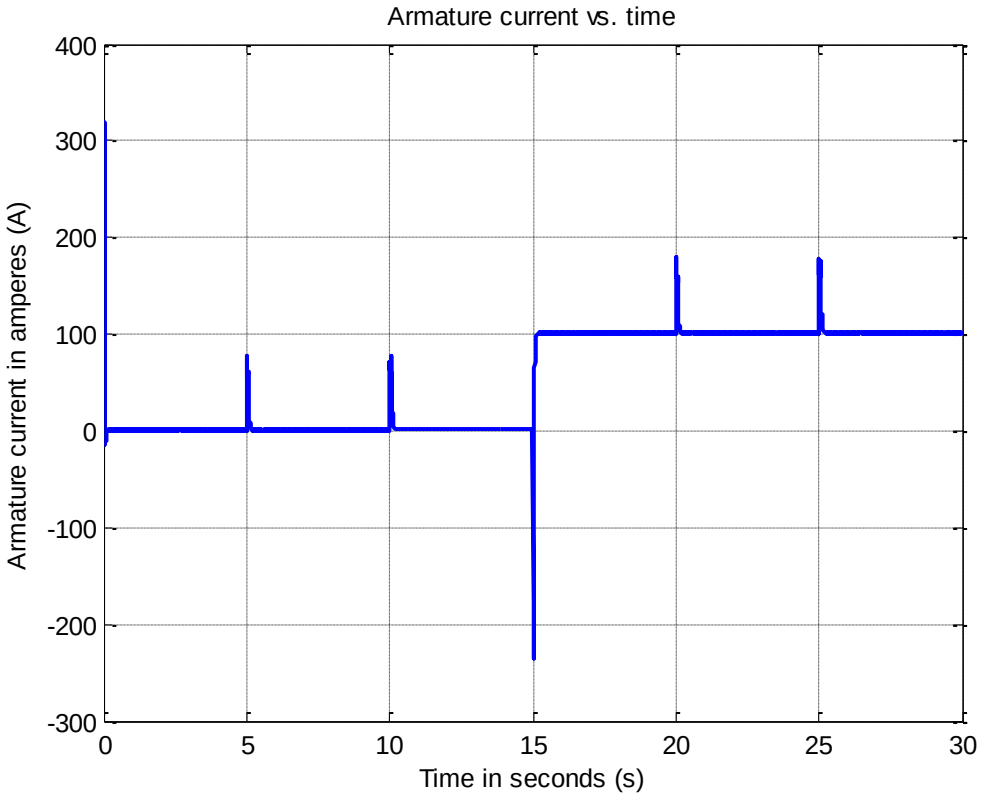

Figure 8.9: Variation of armature current with time

Right when simulation starts, there is high armature current flow as the motor just starts from rest. But as the motor reaches steady speed, the armature current falls down to a very small value. This happens very sharply because the rise time of the motor from rest to steady speed is extremely small. The armature current only increases at the instances when the output speed changes. This is shown by the positive spikes in Fig. 8.9. At 15 seconds, the speed of the motor decreases sharply. In order to cause this, the armature current flows in the opposite direction by a large amount. This is shown by the negative spike in Fig. 8.9. At the same time, a load torque is added to the motor. The current immediately increases in order for the motor to take the load, and retains a value much higher than that when there was no load.

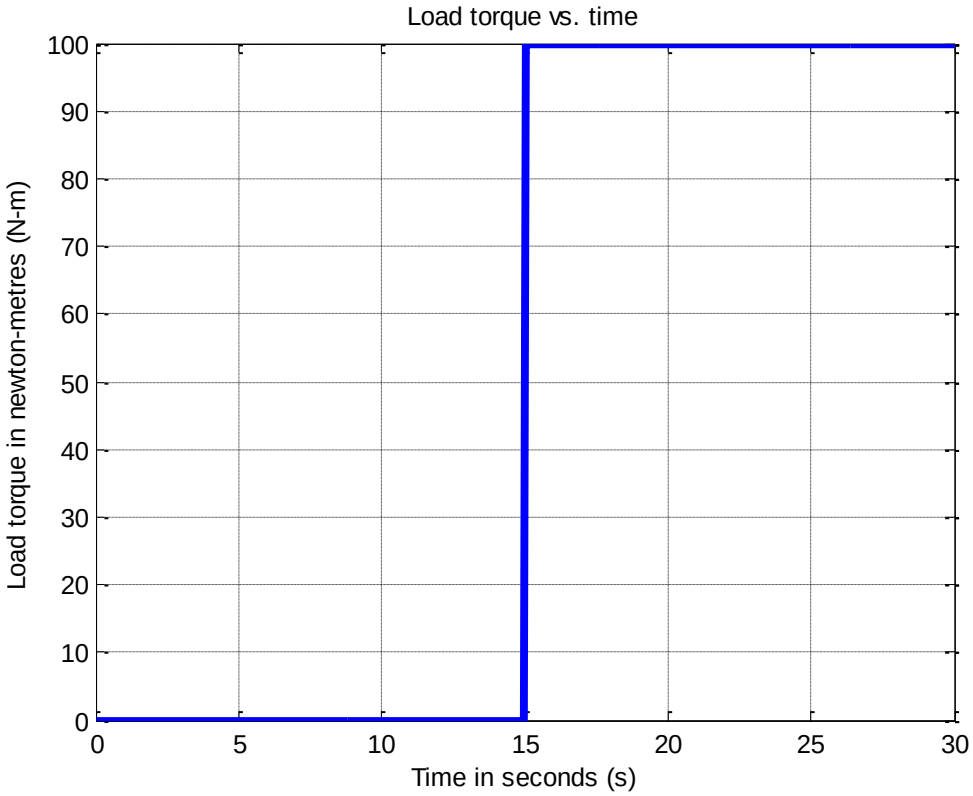

Figure 8.10: Addition of load torque

8.4: FUZZY LOGIC TOOLBOX OF MATLAB

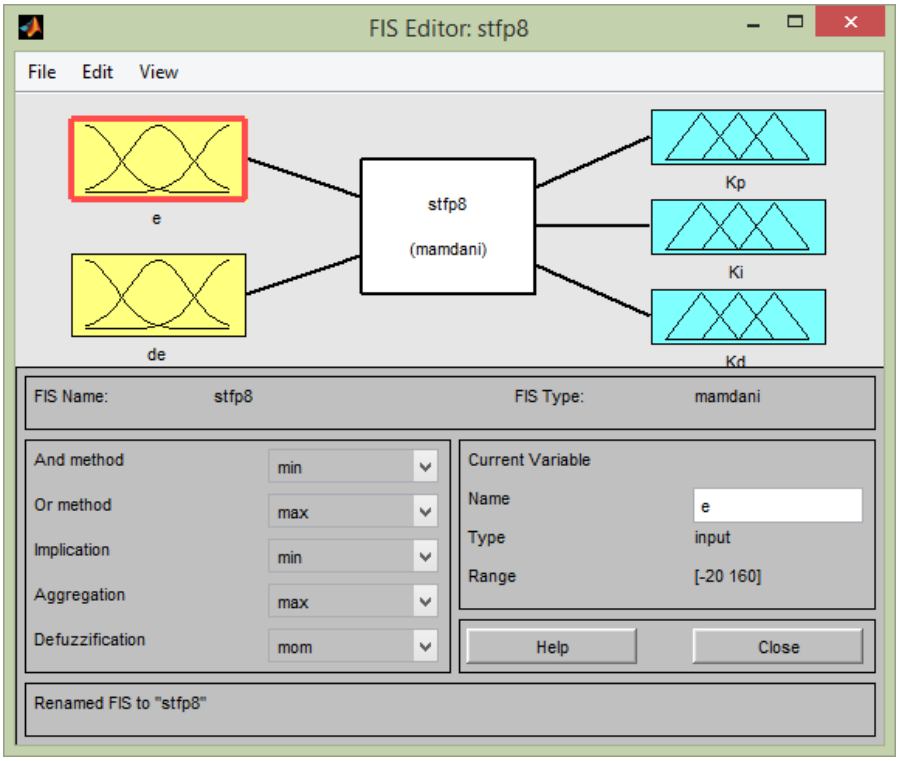

Figure 8.11: FIS Editor of MATLAB

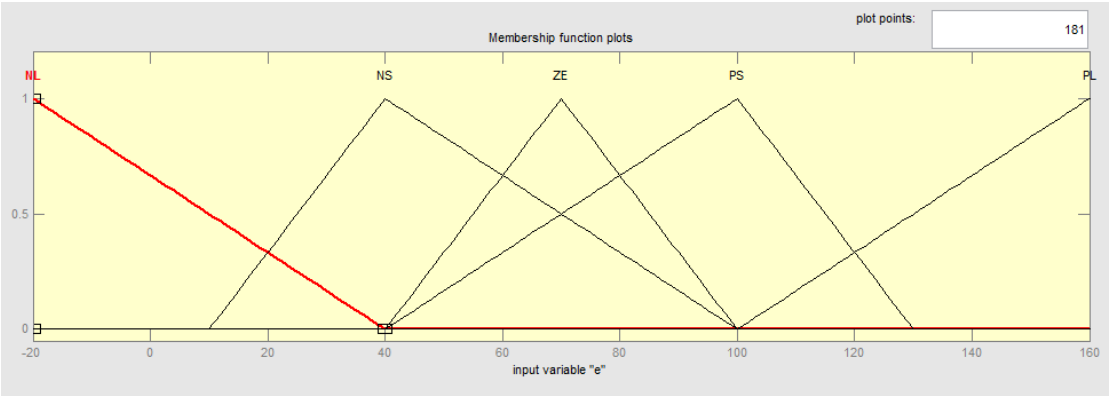

Figure 8.12: Membership function for speed error

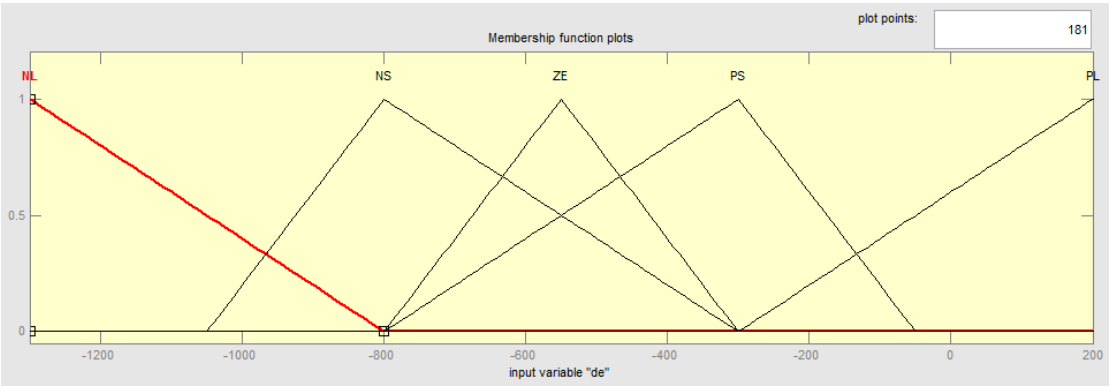

Figure 8.13: Membership function for rate of change of speed error

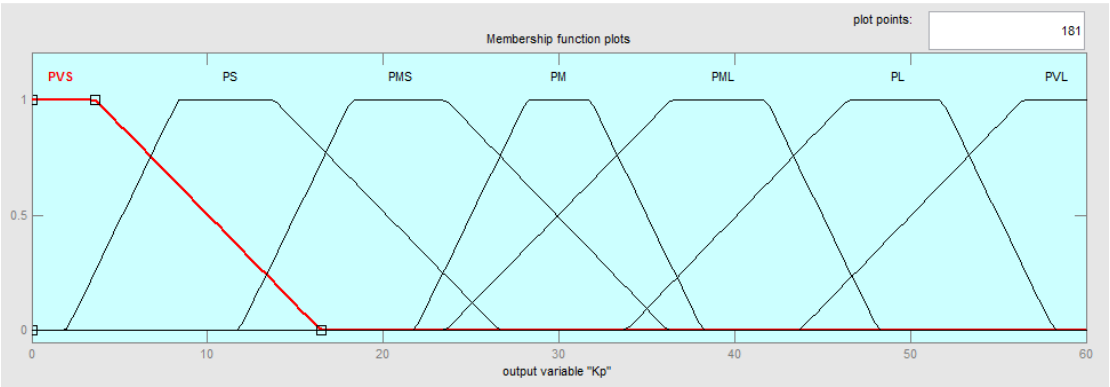

Figure 8.14: Membership function for K_p

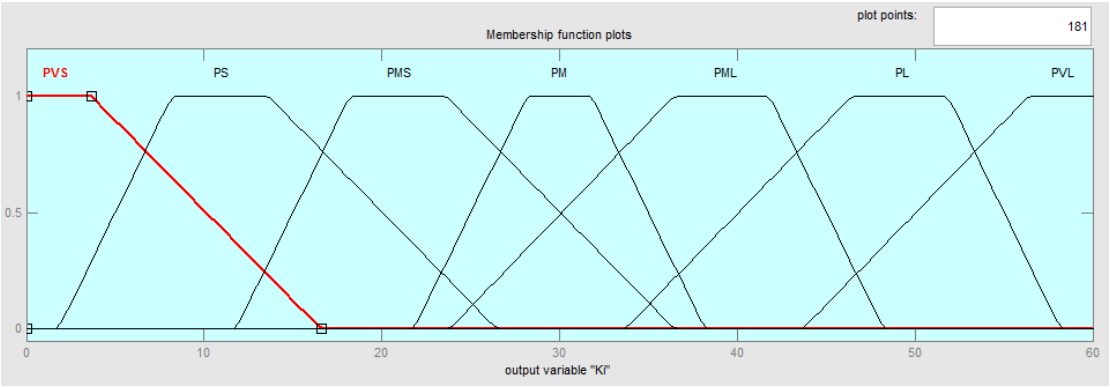

Figure 8.15: Membership function for K_i

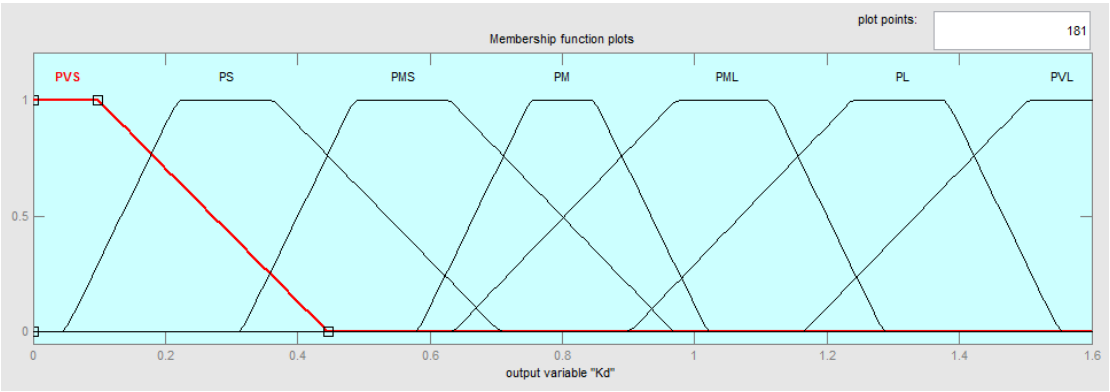

Figure 8.16: Membership function for K_d

CHAPTER 9

OUTPUT RESPONSE AND PERFORMANCE ANALYSIS FROM SPEED-TIME CHARACTERISTICS

9.1: BASIC PERFORMANCE PARAMETERS

Before going to the performance analysis of the speed time characteristics it is necessary to get familiar with following few terms [27]:

Rise time: The time required for the waveform to go from 0.1 of the final value (steady-state) to 0.9 of the final value (Fig. 9.1).

Fall time: The time required for the waveform to go from 0.9 of the final value to 0.1 of the final value. (In the figure fall time is not present; this is a term which is used due to the rapid decrease in the input or reference speed)

Peak time: The time required to reach the first, or maximum, peak (Fig. 9.1).

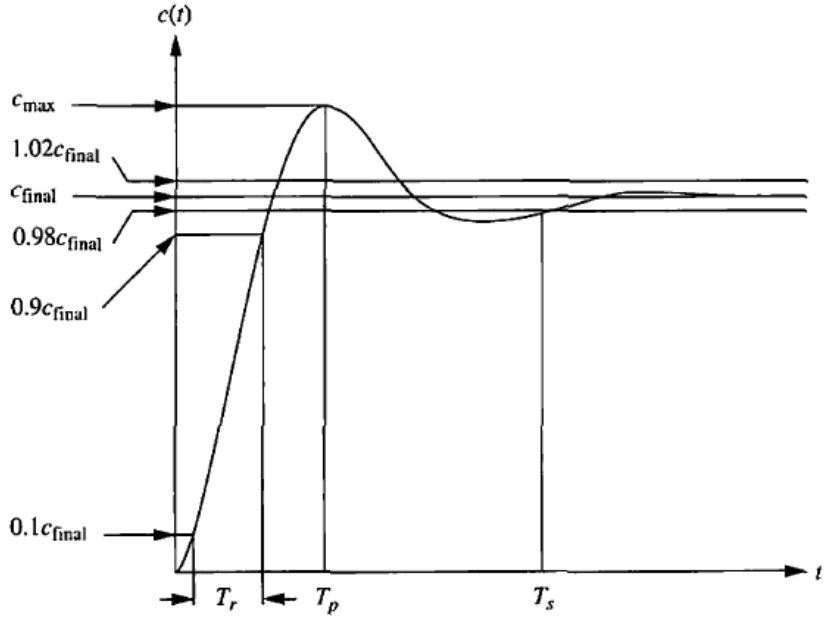

Figure 9.1: General output response specifications

Percentage overshoot (%OS): The amount that the waveform overshoots the steady-state, or final, value at the peak time, expressed as a percentage of the steady-state value (Fig. 9.1).

Settling time: The time required for the transient's damped oscillations to reach and stay within ±2% of the steady-state value (Fig. 9.1).

9.2: RISE TIME AND FALL TIME ANALYSIS

Load condition	*Speed variation (r.p.m.)*	*Rise time/fall time (seconds)*
Unloaded	0 – 1000	0.0404
	1000 – 1500	0.0866
	1500 – 2000	0.0863
Loaded	2000 – 1000	0.0559
	1000 – 1500	0.0852
	1500 – 2000	0.0866

Table 9.1: Rise times and fall times of the output speed of DC motor controlled by fuzzy PID controller.

We notice a sharp rise of the output when the input is changed from 0-1000 r.p.m. As the input is changed gradually with the difference of 500 r.p.m. rise time remains almost same (case 2 and 3 in the above chart). Now for the observation of another case if the input is decreased rapidly (2000-1000 r.p.m.) then we notice a sharp fall of the output. Again on the contrary if we consider the input which is increased gradually with step size of 500 r.p.m. similar rise time is noticed as was in the case 2 and 3.

9.3: OVERSHOOT AND UNDERSHOOT ANALYSIS

Load condition	*Speed variation (r.p.m.)*	*Percentage maximum overshoot/undershoot*
Unloaded	0 – 1000	3.55442
	1000 – 1500	0.03333
	1500 – 2000	0.07000
Loaded	2000 – 1000	0.04325
	1000 – 1500	0.07137
	1500 – 2000	0.06978

Table 9.2: Percentage maximum overshoots and undershoot of the output speed of DC motor controlled by fuzzy PID controller

There is a noticeable amount of overshoot when the motor starts from rest and goes to 1000 r.p.m. But in all the other cases (Both loaded and unloaded) amount of overshoot is not that significant.

9.4: SETTLING TIME ANALYSIS

Load condition	*Speed variation (r.p.m.)*	*Settling time (seconds)*
Unloaded	0 – 1000	0.2307
	1000 – 1500	0.1050
	1500 – 2000	0.1008
Loaded	2000 – 1000	0.1520
	1000 – 1500	0.1069
	1500 – 2000	0.1016

Table 9.3: Settling times of the output speed of DC motor controlled by fuzzy PID controller

At first it takes a little bit more time for the motor to reach its desired speed from its rest condition. But for all other cases (except speed fall) settling time is pretty much similar. Due to the rapid fall of reference the output needs a little bit more time than other cases.

9.5: STEADY STATE ERROR ANALYSIS

Steady-State Error: Steady-state error is defined as the difference between the input (reference) and the output of a system in the limit as time goes to infinity (i.e. when the response has reached steady state).

Load condition	*Speed variation (r.p.m.)*	*Steady state error (r.p.m.)*
Unloaded	0 – 1000	0.71870
	1000 – 1500	-0.01823
	1500 – 2000	-0.03354
Loaded	2000 – 1000	0.01240
	1000 – 1500	-0.12005
	1500 - 2000	-0.11092

Table 9.4: Steady state errors of the output speed of DC motor controlled by fuzzy PID controller. Note: Here (-ve) means that output response is greater than the input or reference.

CHAPTER 10

CONCLUSION

10.1: BENEFITS OF SELF-TUNED FUZZY PID CONTROLLER

The three parameters "K_p", "K_i", "K_d" of conventional PID control need to be constantly adjusted online in order to achieve better control performance. Fuzzy self-tuning PID parameters controller can automatically adjust PID parameters in accordance with the speed error and the rate of speed error-change, so it has better self-adaptive capacity. It completely removes the complications that arise when manual tuning of PID controller is done. Fuzzy PID parameter controller has very small overshoot, rising time and settling time and has excellent dynamic response properties and steady-state properties. Steady state error in case of self-tuned fuzzy PID is negligibly small compared to conventional PID controller.

10.2: DISCUSSION

In this project different method for speed control of DC motor are studied with great care. The speed-time characteristics, steady state and dynamic response of the DC motor output speed, and its torque-speeds, torque-current characteristics are studied. We have also studied basic definition and terminology of fuzzy logic and fuzzy set. This project introduces a design method of two inputs and three outputs self-tuning fuzzy PID controller and make use of MATLAB fuzzy toolbox to design fuzzy controller. The fuzzy controller adjusted the proportional, integral and derivate (K_p, K_i, K_d) gains of the PID controller according to both speed error and change in speed error.

The three parameters of the PID controller (K_p, K_i, K_d) needs to be fine-tuned automatically according to the input. Self-tuned fuzzy PID controller can auto-tune the PID parameters using the speed error and the rate of change of speed error to give output that very closely resemble the input.

The conclusion drawn from the simulation analysis shows that self-tuned fuzzy PID controlled DC motor has very low rise time, fall time, overshoot, settling time and steady state error. In other words, its response is extremely good, both in transient and steady-state. Simulations at both unloaded and loaded conditions show the adaptability of the controller and consistency of the output response.

10.3: FUTURE SCOPE

The speed control of DC motor is simulated using MATLAB and Simulink. Hardware implementation of this simulation can be done to compare the simulation results with the actual findings, and to find out whether it is feasible to build such a controller or not. The technique developed in this thesis can be investigated for separately excited, series and shunt DC motor drives, and also different types of AC motor drives. Further researches can be carried out to implement this design of speed controller using other techniques, for example, neuro-fuzzy, or genetic algorithm (GA).

REFERENCES

1. Seda Aydemir, Serkan Sezen, H.Metin Ertunc, "Fuzzy Logic Speed Control of a DC Motor", Kocaeli University Izniit, Kocaeli, Turkey.

2. Umesh Kumar Bansal and Rakesh Narvey, "Speed Control of DC Motor using Fuzzy PID Controller", Advance in Electronic and Electric Engineering. ISSN 2231-1297, Volume 3, Number 9 (2013), pp. 1209-1220.

3. Khoei, A.. Hadidi, Kh., Microprocessor Based Closed-Loop Speed Control Systeni For DC Motor Using Power Mosfet. Electronics Circuits and Systems IEEE International Conference ICECS'96, Vol. 2, 1247-1250, 1996.

4. O. Imoru, J. Tsado. "Modeling of an Electronically Commutated (Brushless DC) Motor Drives with Back-EMF Sensing", Proceedings of the IEEE International Conference on Machine Design, China, pp 828-831, June 2012.

5. Md Mustafa kamal, Dr.(Mrs.)Lini Mathew, Dr. S. Chatterji, "Speed Control of Brushless DC Motor Using Fuzzy Based Controllers", 2014 IEEE Students' Conference on Electrical, Electronics and Computer Science.

6. www.facstaff.bucknell.edu/mastascu/econtrolhtml/pid/pid3.html

7. radio.feld.cvut.cz/matlab/toolbox/fuzzy/fuzzyin2.html

8. Mohan, Undeland, Robbins, "Power Electronics" (John Wiley & Sons) pp. 379-384.

9. Bennett, Stuart (1993), "A History of Control Engineering", 1930-*1955*. IET. pp. 48.

10. Ang, K.H., Chong, G.C.Y., and Li, Y. (2005), "PID Control System Analysis, Design and Technology", *IEEE Trans Control Systems Tech*, 13(4), pp.559-576.

11. www.wikipedia.org/wiki/PID_controller

12. Jinghua Zhong (Spring 2006). "PID Controller Tuning: A Short Tutorial".

13. Prahlad Kumar Sahoo, " Speed Control of Separately Excited DC Motor using Self Tuned Fuzzy PID Controller", 2010-2011 Department of Electrical Engineering, National Institute of Technology Rourkela.

14. L.A. Zadeh, "Making computers think like people," IEEE. Spectrum, 8/1984, pp. 26-32.

15. L.A. Zadeh, Fuzzy Sets, Information and Control, 1965.

16. L.A. Zadeh, Outline of A New Approach to the Analysis of of Complex Systems and Decision, Processes, 1973.

17. L.A. Zadeh, ”Fuzzy algorithms,” Info. & Ctl., Vol. 12, 1968, pp. 94-102.

18. M. Hellmann, “Fuzzy Logic Introduction”, Laboratoire Antennes Radar Telecom, F.R.E CNRS 2272, Equipe Radar Polarimetrie, Universit´e de Rennes, France.

19. L.A. Zadeh, Fuzzy Sets, Information and Control, 1965.

20. www.bindichen.co.uk/post/AI/fuzzy-inference-membership-function.html

21. Seda Aydemir, Serkan Sezen, H.Metin Ertunc, “Fuzzy Logic Speed Control of a DC Motor”, Kocaeli University Izniit, Kocaeli, Turkey.

22. Maher M.F. Algreer and Yhya R. M. Kuraz, “Design Fuzzy Self Tuning of PID Controller for Chopper-Fed DC Motor drive.”

23. Nur Azliza Ali, “Fuzzy Logic Controller for Controlling DC Motor Speed using MATLAB Applications**”.**

24. Manafeddin Namazov and Onur Basturk, “DC Motor Position Control using Fuzzy Proportional-Derivative Controllers with Different Defuzzification Methods”, Turkish Journal of Fuzzy Systems (eISSN: 1309–1190), Vol.1, No.1, pp. 36-54, 2010

25. Wang Xiao-kan, Sun Zhong-liang, Wanglei, Feng Dong-qing, "Design and Research Based on Fuzzy PID-Parameters Self-Tuning Controller with MATLAB," Advanced Computer Theory and Engineering, International Conference on, pp. 996-999, 2008 International Conference on Advanced Computer Theory and Engineering, 2008.

26. M. Chow and A. Menozzi ,”On the Comparison of Emerging and Conventional Techniques for DC Motor Control” proc.IECON, pp.1008-1013, 1992.

27. Norman S Nise, “Control System Engineering”, sixth edition (John Wiley & Sons) pp. 179