A COMPARISON BETWEEN SOFT COMPUTING METHODS AND CONVENTIONAL (PID) CONTROLLER APPLIED TO A DC MOTOR SYSTEM

A MASTER'S THESIS

in

Mechatronics Engineering

Atilim University

by

FATHI MOHAMED A. DARBI

MAY 2014

A COMPARISON BETWEEN SOFT COMPUTING METHODS AND CONVENTIONAL (PID) CONTROLLER APPLIED TO A DC MOTOR SYSTEM

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

OF
ATILIM UNIVERSITY

BY

FATHI MOHAMED A. DARBI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF MECHATRONICS ENGINEERING

MAY 2014

Approval of the Graduate School of Natural and Applied Sciences, Atilim University.

_____

Prof. Dr. K. Ibrahim Akman

Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

_____

Prof. Dr. Abdulkadir Erden

Head of Department

This is to certify that we have read the thesis "A COMPARISON BETWEEN SOFT COMPUTING METHODS AND CONVENTIONAL (PID) CONTROLLER APPLIED TO A DC MOTOR SYSTEM" submitted by "Fathi Mohamed A. Darbi" and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

_____

Assoc. Prof. Dr. Fuad Aliew

Supervisor

Examining Committee Members

Assoc. Prof. Dr. Sedat Nazlıbilek	(ATU. MECE. Dept.)	_____
Assoc. Prof. Dr. Fuad Aliew	(ATU. MECE. Dept.)	_____
Asst. Prof. Dr. Hakan Tora	(ATU. E.E. Dept.)	_____
Asst. Prof. Dr. Bülent İrfanoğlu	(ATU. MECE. Dept.)	_____
Asst. Prof. Dr. Kutluk Bilge Arıkan	(ATU. MECE. Dept.)	_____

Date: 07.05.2014

I declare and guarantee that all data, knowledge and information in this document has been obtained, processed and presented in accordance with academic rules and ethical conduct. Based on these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Fathi Mohamed A. Darbi

ABSTRACT

A COMPARISON BETWEEN SOFT COMPUTING METHODS AND CONVENTIONAL (PID) CONTROLLER APPLIED TO A DC MOTOR SYSTEM

Fathi Mohamed A. Darbi

M.S. Mechatronics Engineering Department

Supervisor : Assoc. Prof. Dr. Fuad Aliew

May 2014 99 pages

This research will compare the Soft computing techniques its advantages with the optimal proportional-integral derivative (PID) controller parameters, for control of a different parameters of a DC motor. The research approach is to establish superior features, including easy implementation, stable convergence characteristic and good computational efficiency.

The methods for designing of PID Controllers will been compared and analyzed with the intelligent tuning techniques like Fuzzy Sets, Neural Networks and Genetic algorithms, Soft computing method was more efficient in improving the step response characteristics such as, reducing the steady-states error; rise time, settling time and maximum overshoot in position control of a DC motor.

Keywords: Fuzzy Logic, Artificial Neural Network (ANN), Non-Linear Systems Modeling, Controller, Matlab/Simulink, Real Time Control.

ÖZ

DC MOTOR SİSTEMİNE UYGULANAN PID KONTROLÜN VE YAZILIM HESAPLAMA METHODLARININ KARŞILAŞTIRILMASI

Fathi Mohamed A. Darbi

Mekatronik Mühendisliği Departmanı Yüksek Lisans Programı

Danışman : Doç. Dr. Fuad Aliew

Mayıs 2014 99 sayfa

Bu araştırma, DC motorun farklı kontrol parametreleri için; yazılım hesaplama tekniklerini ve avantajlarının optimal PID kontrolcü parametrelerini karşılaştıracaktır. Araştırma yaklaşımı; kolay uygulama, kararlı yakınsama karakteristiği ve iyi hesaplama verimliliğinin dahil olduğu üst düzey özellikleri içerir.

Bulanık kümeler, yapay sinir ağları ve genetik algoritmalar gibi PID kontrolcü tasarım methodları, akıllı ayarlama teknikleri ile analiz edilecek ve karşılaştırılacaktır. Basamak tepkisinin karakteristiğinin geliştirilmesi, yazılım hesaplama yöntemleri için daha verimlidir. Örneğin, DC motorun pozisyon kontrolünde; kararlı hal hataları, yükselme zamanı, ayarlama zamanı ve maksimum sapmanın düşürülmesidir.

Anahtar Kelimeler**:** Bulanık Mantık, Yapay Sinir Ağı (ANN), Doğrusal Olmayan Sistemlerin Modellemesi, Kontrolcü, Matlab/Simulink, Gerçek Zamanlı Kontrol.

To My Parents

ACKNOWLEDGMENTS

I would like to thank my supervisor Dr. Fuad Aliew as he was beside me in every step throughout my research, and for his great effort to show me the best techniques and sources to solve different problems. Many thanks goes to my mother who every time helps me with her prays. Also I would like to thank Prof.Dr.Abdulkadir Erden, for his great support to the graduate students and for his effort to keep us on track and to be on time. Many thanks also goes to M.Hassan Gol Mohammad Zadeh for his support. And to my family, I offer sincere thanks for their continuous support and patience during this period.

TABLE OF CONTENTS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

FLC – Fuzzy Logic Control

ANN – Artificial Neural Network

PID – Proportional-Integral-Derivative

SISO – Single-Input-Single-Output

MIMO – Multi- Input-Multi-Output

DC – Direct Current

AC – Alternating Current

MV - Manipulated Variable

PS – Setpoint

PV - Process Variable

FL – Fuzzy Logic

MIN – Minimum

MAX – Maximum

TS – Takagi-Sugeno

COG – Center Of Gravity

BOA – Bisector Of Area

MOM – Mean Of Maximum

COGS – Center Of Gravity For Singletons

PCU – Process Control Unit

EC – Rate of Change of error

IMC – Internal Model Control

MM-IMC – Multiple Model Internal Model Control

NB – Negative Big

NS – Negative Small

ZE – Zero

PS – Positive Small

PB – Positive Big

FIS – Fuzzy Inference System

NN – Neural Network

LM – Levenberg-Marquardt algorithm

AI – Artificial Intelligence

BP – Back-Propagation

LIST OF SYMBOLS

T_d – delay time

T_r – rise time

T_p – peak time

M_p – maximum percent overshoot

T_s – settling time

P_{out} – proportional term of output

I_{out} – integral term of output

D_{out} – derivative term of output

K_p – proportional gain

K_i – integral gain

K_d – derivative gain

e – error

V_{nom} – motor normal input voltage

R_m – motor armature resistance

L_m – motor armature inductance

K_t – motor torque constant

η_m – motor efficiency

K_m – back-emf constant

K_g – high-gear total gearbox ratio

η_g – gearbox efficiency

J_{eq} – equivalent high-gear moment of inertia without external load

B_{eq} – high-gear viscous damping coefficient found experimentally

I_{max} – maximum input current

ω_{max} – maximum motor speed

θ_l – load shaft position

J_l – moment of inertia at the load

T_l – torque applied at the load

θ_m – motor shaft position

T_m – torque generated by the motor

J_m – motor moment of inertia

I_m – armature circuit current

V_m – armature circuit input voltage

C – controller

P, G – plant

F – sensor

r – reference signal

y – output signsl

E(s) – error signal

G(s) – forward gain

H(s) – feedback gain

C(s) – output signal

$U(s)$ – manipulated input

δ_{oi} – error signal

$d^{(k)}$ – output pattern

$x^{(k)}$ – input pattern of NN

$y^{(k)}$ – actual output of NN

y_i – output signal of the neuron

x_j – input signals

w_{iq} – synaptic weights

L – delay time

T – time constant

T_i – integral time

T_d – derivative time

ω_{cr} – frequency of the sustained oscillations

K_{cr} – critical gain

P_{cr} – critical period

ω_n – undamped natural frequency

ζ – damping r

μ – membership grades

CHAPTER 1

INTRODUCTION

Mechatronics systems often use DC motors to drive their work loads, where DC motors are used to provide rotary or linear motion to a variety of electromechanical devices and servo systems. Depending on application (e.g. robots, electric vehicles, low-to-medium power machine-tools etc.) and desired dynamic and steady state performances [6]. DC motor is a nonlinear system that has played an important role in the improvement of the industrial revolution, making it the heart of different applications in robotics and mechatronics beside AC motor systems [25].

DC motors are one of the main components of automatic systems; any automatic system should have an actuator module that makes the system to actually perform its function. The most common actuator used to perform this task is the DC motor. Historically, DC motors also played a vital role in the development of the many devices and drive systems; which make them one of the most important components in our life that we cannot live without it. Due to their importance, the design of controllers for these systems has been an interesting area for researchers from all over the world.

However, even with all of their useful applications and usage, DC motor systems still suffer from several non-linear behaviors and parameters affecting their performance which may lead for the motor to require more complex controlling schemes, or having higher energy consumption and faulty functions in some cases. For these purposes the controller design of DC motor system is an interesting area that still offers multiple topics for research, especially after the discovery of Fuzzy Logic, Artificial Neural Networks (ANN) and their possible usage for intelligent control purposes. From this point of view and the importance of having high efficient DC motor systems, the use of Fuzzy Logic and Artificial Neural Networks (ANN) played a vital role in designing smart controllers that can eliminate or cope with the

non-linear effects found in DC motor systems and improve the functions they are used for [27].

1.1 Aim and Scope

PID Controller is important part in industrial plants and control engineering because it is simple and robust. Improvement and development the performance of PID Controller is the aim of a lot of researchers. In this research we will using soft computing techniques and classical techniques for tuning parameters of PID Controller and compare between them applied to DC Motor.

The Soft computing techniques like Fuzzy Sets and Neural Networks its advantages with the optimal proportional-integral derivative (PID) controller parameters, for control of a different parameters of a DC motor. However, the Fuzzy Sets techniques for tuning parameters of PID Controller has some problems of computation complexity and the performance. To solve these problems, we will reconstruct the Fuzzy PID Controller Based on feed-forward Neural Network using Back-propagation algorithm.

1.2 PID Controller

A proportional–integral–derivative controller (PID controller) is a generic control loop feedback mechanism (controller) widely used in industrial control systems – a PID is the most commonly used feedback controller. A PID controller calculates an "error" value as the difference between a measured process variable and a desired setpoint. The controller attempts to minimize the error by adjusting the process control inputs. In the absence of knowledge of the underlying process, PID controllers are the best controllers. However, for best performance, the PID parameters used in the calculation must be tuned according to the nature of the system – while the design is generic, the parameters depend on the specific system.

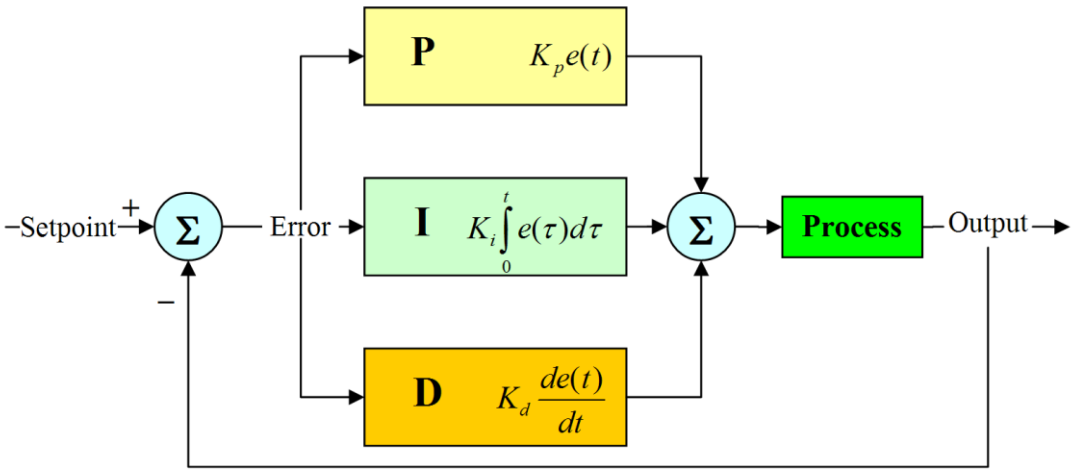


Figure 1 Block diagram of PID controller

The PID controller calculation (algorithm) involves three separate parameters, and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted P, I, and D. The proportional value determines the reaction to the current error, the integral value determines the reaction based on the sum of recent errors, and the derivative value determines the reaction based on the rate at which the error has been changing. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element. Heuristically, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change.

By tuning the three constants 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 setpoint 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.

Some applications may require using only one or two modes to provide the appropriate system control. This is achieved by setting the gain of undesired control outputs 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 value may prevent the system from reaching its target value due to the control action [28].

1.2.1 PID Control Theory

The PID controller is probably the most-used feedback control design. PID is an acronym for Proportional-Integral-Derivative, referring to the three terms operating on the error signal to produce a control signal. If $u(t)$ is the control signal sent to the system, $y(t)$ is the measured output and $r(t)$ is the desired output, and tracking error $e(t) = r(t) - y(t)$, a PID controller has the general form

$$u(t) = K_p\, e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{d}{dt} e(t). \quad (1)$$

The desired closed loop dynamics is obtained by adjusting the three parameters K_p, K_i and K_D, often iteratively by "tuning" and without specific knowledge of a plant model. Stability can often be ensured using only the proportional term. The integral term permits the rejection of a step disturbance (often a striking specification in process control). The derivative term is used to provide damping or shaping of the response. PID controllers are the most well established class of control systems: however, they cannot be used in several more complicated cases, especially if MIMO systems are considered.

Applying Laplace transformation results in the transformed PID controller equation

$$u(s) = K_p e(s) + K_I \frac{1}{s} e(s) + K_D s e(s) \quad (2)$$

$$u(s) = (K_p + K_I \frac{1}{s} + K_D s) e(s) \quad (3)$$

With the PID controller transfer function

$$C(s) = \left(K_p + K_I \frac{1}{s} + K_D\, s\right). \quad (4)$$

In other words, The PID control scheme is named after its three correcting terms, whose sum constitutes the manipulated variable (MV). Hence:

$$MV(t) = P_{out} + I_{out} + D_{out} \quad (5)$$

Where

(MV) The input to the process is called the manipulated variable. P_{out}, I_{out}, and D_{out} are the contributions to the output from the PID controller from each of the three terms, as defined below [28].

1.2.2 Proportional term

The proportional term (sometimes called gain) makes a change to the output that is proportional to the current error value. The proportional response can be adjusted by multiplying the error by a constant K_p, called the proportional gain.

The proportional term is given by:

$$P_{out} = K_p e(t) \tag{6}$$

Where

P_{out}: Proportional term of output

K_p :Proportional gain, a tuning parameter

$e: Error = PS - PV$ (7)

PS: The desired value is called the setpoint

PV: The measurement of process value or process variable

t: Time or instantaneous time (the present)

A high proportional gain results in a large change in the output for a given change in the error. If the proportional gain is too high, the system can become unstable. In contrast, a small gain results in a small output response to a large input error, and a less responsive (or sensitive) controller. If the proportional gain is too low, the control action may be too small when responding to system disturbances

In the absence of disturbances, pure proportional control will not settle at its target value, but will retain a steady state error (known as droop) that is a function of the proportional gain and the process gain. Specifically, if the process gain – the long-term drift in the absence of control, such as cooling of a furnace towards room temperature – is denoted by G and assumed to be approximately constant in the error, then the droop is when this constant gain equals the proportional term of the output, P_{out}, which is linear in the error, G=$K_p e$, so e=G/K_p. This is when the proportional term, which is pushing the parameter towards the set point, is exactly offset by the process gain, which is pulling the parameter away from the set point. If the process

gain is down, as in cooling, then the steady state will be below the set point, hence the term "droop".

Only the drift component (long-term average, zero-frequency component) of process gain matters for the droop – regular or random fluctuations above or below the drift cancel out. The process gain may change over time or in the presence of external changes, for example if room temperature changes, cooling may be faster or slower.

Droop is proportional to process gain and inversely proportional to proportional gain, and is an inevitable defect of purely proportional control. Droop can be mitigated by adding a bias term (setting the setpoint above the true desired value), or corrected by adding an integration term (in a PI or PID controller), which effectively computes a bias adaptively.

Despite the droop, both tuning theory and industrial practice indicate that it is the proportional term that should contribute the bulk of the output change.

1.2.3 Integral term

The contribution from the integral term (sometimes called reset) is proportional to both the magnitude of the error and the duration of the error. Summing the instantaneous error over time (integrating the error) gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain and added to the controller output. The magnitude of the contribution of the integral term to the overall control action is determined by the integral gain, K_i.

The integral term is given by:

$$I_{out} = K_i \int_0^t e(\tau) d\tau \tag{8}$$

Where

I_{out}: Integral term of output

K_i: Integral gain, a tuning parameter

e: $Error = PS - PV$ (9)

t: Time or instantaneous time (the present)

τ: Variable of integration; takes on values from time 0 to the present t

1.2.4 Derivative term

The rate of change of the process error is calculated by determining the slope of the error over time (i.e., its first derivative with respect to time) and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term (sometime called rate) to the overall control action is termed the derivative gain, K_d.

The derivative term is given by:

$$D_{out} = K_d \frac{d}{dt} e(t) \tag{10}$$

Where

D_{out}: Derivative term of output

K_d: Derivative gain, a tuning parameter

e: $Error = PS - PV$

t: Time or instantaneous time (the present)

The derivative term slows the rate of change of the controller output and this effect is most noticeable close to the controller setpoint. Hence, derivative control is used to reduce the magnitude of the overshoot produced by the integral component and improve the combined controller-process stability. However, differentiation of a signal amplifies noise and thus this term in the controller is highly sensitive to noise in the error term, and can cause a process to become unstable if the noise and the derivative gain are sufficiently large. Hence an approximation to a differentiator with a limited bandwidth is more commonly used. Such a circuit is known as a Phase-Lead compensator.

The proportional, integral, and derivative terms are summed to calculate the output of the PID controller. Defining $u(t)$ as the controller output, the final form of the PID algorithm is:

$$u(t) = MV(t) = K_p\, e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{d}{dt} e(t) \tag{11}$$

where the tuning parameters are:

Proportional gain, K_p

Larger values typically mean faster response since the larger the error, the larger the proportional term compensation. An excessively large proportional gain will lead to process instability and oscillation.

Integral gain, K_i

Larger values imply steady state errors are eliminated more quickly. The trade-off is larger overshoot: any negative error integrated during transient response must be integrated away by positive error before reaching steady state.

Derivative gain, K_d

Larger values decrease overshoot, but slow down transient response and may lead to instability due to signal noise amplification in the differentiation of the error.

1.3 Overview of tuning methods

There are several methods for tuning a PID loop. The most effective methods generally involve the development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively inefficient, particularly if the loops have response times on the order of minutes or longer.

The choice of method will depend largely on whether or not the loop can be taken "offline" for tuning, and the response time of the system. If the system can be taken offline, the best tuning method often involves subjecting the system to a step change in input, measuring the output as a function of time, and using this response to determine the control parameters.

Table 1 Selection of Tuning Method

Method	Advantages	Disadvantages
Manual Tuning	No math required. Online method.	Requires experienced personnel.
Ziegler–Nichols	Proven Method. Online method.	Process upset, some trial and-error, very aggressive tuning.
Software Tools	Consistent tuning. Online or offline method. May include valve and sensor analysis. Allow simulation before downloading	Some cost and training involved.
Cohen-Coon	Good process models.	Some math. Offline method. Only good for first-order processes.

1.3.1 Manual tuning

If the system must remain online, one tuning method is to first set K_i and K_d values to zero. Increase the K_p until the output of the loop oscillates, then the K_p should be set to approximately half of that value for a "quarter amplitude decay" type response. Then increase K_i until any offset is correct in sufficient time for the process. However, too much K_i will cause instability. Finally, increase K_d, if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much K_d will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the setpoint more quickly; however, some systems cannot accept overshoot, in which case an over-damped closed-loop system is required, which will require a K_p setting significantly less than half that of the K_p setting causing oscillation [28], [29].

Table 2 Effects of increasing a parameter independently

Parameter	**Rise time**	**Overshoot**	**Settling time**	**Steady-state error**	**Stability**
$\boldsymbol{K_p}$	Decrease	Increase	Small change	Decrease	Degrade
$\boldsymbol{K_i}$	Decrease	Increase	Increase	Decrease significantly	Degrade
$\boldsymbol{K_d}$	Minor change	Decrease	Decrease	No effect in theory	Improve if K_d small

1.3.2 Ziegler–Nichols method

Another heuristic tuning method is formally known as the Ziegler–Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols. Ziegler and Nichols proposed rules for determining values of the proportional gain K_p integral time T_i, and derivative time T_d, based on the transient response characteristics of a given plant. Such determination of the parameters of PID controllers or tuning of PID controllers can be made by engineers on-site by experiments on the plant. (Numerous tuning rules for PID controllers have been proposed since the Ziegler-Nichols proposal. They are available in the literature and from the manufacturers of such controllers.)

There are two methods called Ziegler-Nichols tuning rules: the first method and the second method [3].
You can show more details about Ziegler-Nichols method in chapter (3).

CHAPTER 2

DC MOTOR

2.1 Quanser Motion Servo Plant (SRV02) Model (006 S)

The Quanser SRV02 rotary servo plant, pictured in Figure 2, consist of a DC motor that is encased in a solid aluminum frame and equipped with a planetary gearbox. That is, the motor has its own internal gearbox that drives external gears. The basic SRV02 units comes with an potentiometer sensor that can be used to measure angular position of the load gear. The SRV02 device can also be fitted with an encoder to obtain a digital position measurement [32].

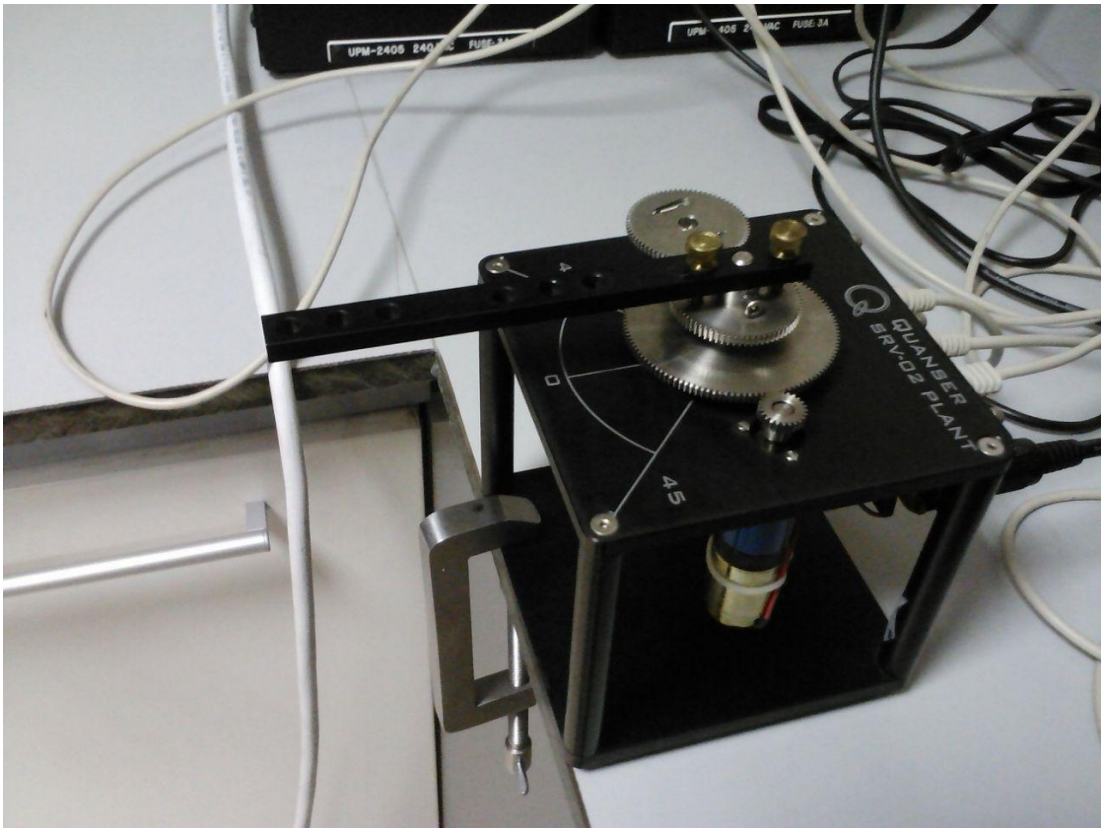


Figure 2 Top view of Servo Plant with higher gear configuration

2.2 Mathematical Modelling of Position Control for DC motor (SRV02)

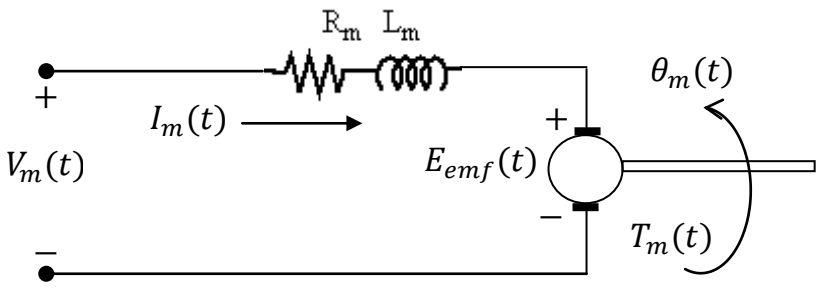


Figure 3 Armature circuit in the time-domain

Using Kirchhoff's voltage law, we obtain the following equation:

$$V_m - R_m I_m - L_m \frac{dI_m}{dt} - E_{emf} \tag{12}$$

Since $L_m \ll R_m$, we can disregard the motor inductance leaving us with:

$$I_m = \frac{V_m - E_{emf}}{R_m} \tag{13}$$

We know that the back emf created by the motor is proportional to the motor shaft velocity ω_m such that:

$$I_m = \frac{V_m - K_m \dot{\theta}_m}{R_m} \qquad \left(\dot{\theta}_m - \omega_m\right) \tag{14}$$

We now shift over to the mechanical aspect of the motor and begin by applying Newton's 2nd law of motion to the motor shaft:

$$J_m \ddot{\theta}_m = T_m - \frac{T_l}{\eta_g K_g} \tag{15}$$

Where

$$\frac{T_l}{\eta_g K_g}$$

Is the load torque seen thru the gears. And η_g is the efficiency of the gearbox

We now apply the 2^{nd} law of motion at the load of the motor:

$$J_l\ddot{\theta}_l = T_l - B_{eq}\dot{\theta}_l \tag{16}$$

Where B_{eq} is the viscous damping coefficient as seen at the output.

Substituting (15) into (16), we are left with:

$$J_l\ddot{\theta}_l = \eta_g K_g T_m - \eta_g K_g J_m \ddot{\theta}_m - B_{eq}\dot{\theta}_l \tag{17}$$

We know that $\theta_m = K_g\theta_l$ and $T_m = \eta_m K_t I_m$ (where η_m is the motor efficiency),We can re-write (17) as:

$$J_l\ddot{\theta}_l + \eta_g {K_g}^2 J_m\ddot{\theta}_l + B_{eq}\dot{\theta}_l = \eta_g\eta_m K_g K_t I_m \tag{18}$$

Finally, we can combine the electrical and mechanical equations by substituting (14) into (18), yielding our desired transfer function:

$$\frac{\theta_l(s)}{V_m(s)} = \frac{\eta_g\eta_m K_g K_t}{J_{eq}R_m s^2 + (B_{eq}R_m + \eta_g\eta_m K_m K_t {K_g}^2)s} \tag{19}$$

Where:

$J_{eq} = J_l + \eta_g J_m {K_g}^2$

This can be interpreted as the being the equivalent moment of inertia of the motor system as seen at the output [32].

2.3 Specifications of DC motor (SRV02)

$V_{nom} = 6.0\ V$ Motor normal input voltage

$R_m = 2.6\ \Omega$ Motor armature resistance

$L_m = 0.18\ mH$ Motor armature inductance

$K_t = 0.0077\ N.m$ Motor torque constant

$\eta_m = 0.69$ Motor efficiency

$K_m = 0.0077\ V/(rad/sec)$ Back-emf constant

$K_g = 70$ High-gear total gearbox ratio

$\eta_g = 0.9$ Gearbox efficiency

$J_{eq} = 0.0021\ K_g.m^2$ Equivalent high-gear moment of inertia with external load

$B_{eq} = 0.015\ N.m/(rad/s)$ High-gear viscous damping coefficient found experimentally

$I_{max} = 1.0\ A$ Maximum input current

$\omega_{max} = 628.3\ rad/s$ Maximum motor speed

The specifications of the DC motor (SRV02) model (006 S) is found in [Appendix B and C].

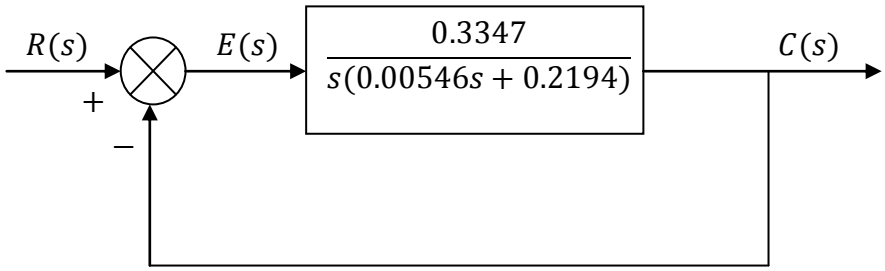


Figure 4 Block diagram of Transfer Function of DC motor position Control

We have two poles in our system:

$s_1 = 0$ & $s_2 = -40.183$

Both of them locate in left hand side of S-Plane diagram as shown in Figure 5, that's mean the system is stabile.

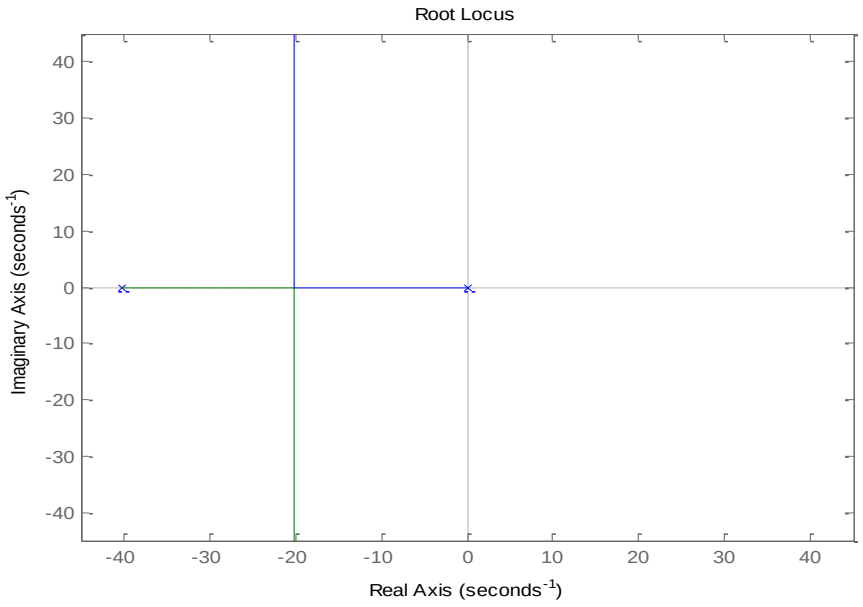


Figure 5 Root Locus Diagram of DC Motor System

CHAPTER 3

DESIGN OF PID CONTROLLER BY USING ZIEGLER-NICHOLS METHOD

3.1 Ziegler–Nichols method

Another heuristic tuning method is formally known as the Ziegler–Nichols method, introduced by John G. Ziegler and Nathaniel B. Nichols. Ziegler and Nichols proposed rules for determining values of the proportional gain K_p integral time T_i, and derivative time T_d, based on the transient response characteristics of a given plant. Such determination of the parameters of PID controllers or tuning of PID controllers can be made by engineers on-site by experiments on the plant. (Numerous tuning rules for PID controllers have been proposed since the Ziegler-Nichols proposal. They are available in the literature and from the manufacturers of such controllers.)

There are two methods called Ziegler-Nichols tuning rules: the first method and the second method. We shall give a brief presentation of these two methods.

First Method: In the first method, we obtain experimentally the response of the plant to a unit-step input, as shown in Figure 6. If the plant involves neither integrator(s) nor dominant complex-conjugate poles, then such a unit-step response curve may look S-shaped, as shown in Figure 7.This method applies if the response to a step input exhibits an S-shaped curve. Such step-response curves may be generated experimentally or from a dynamic simulation of the plant.

The S-shaped curve may be characterized by two constants, delay time L and time constant T. The delay time and time constant are determined by drawing a tangent line at the inflection point of the S-shaped curve and determining the intersections of the tangent line with the time axis and line $c(t) = K$, as shown in Figure 7. The transfer

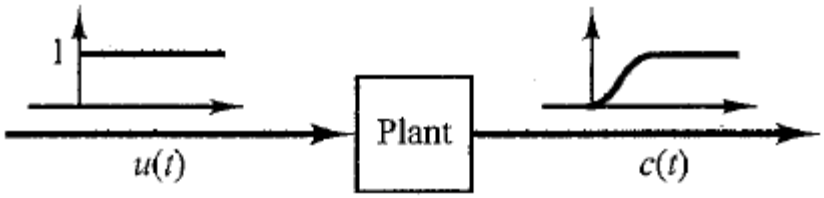


Figure 6 Unit-step response of a plant

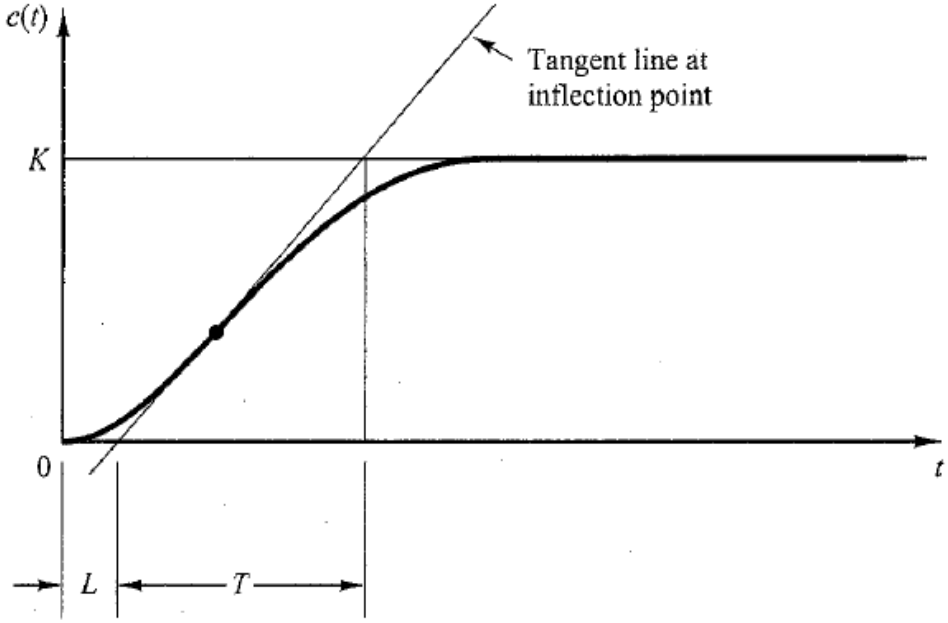


Figure 7 S-shaped response curve

Table 3 Ziegler- Nichols Tuning Rule Based on Step Response of plant First Method

Type of Controller	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

Function $C(s)/U(s)$ may then be approximated by a first-order system with a transport

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \tag{20}$$

Ziegler and Nichols suggested to set the values of K_p ,T_i and T_d according to the formula shown in Table 3.

Notice that the PID controller tuned by the first method of Ziegler-Nichols rules gives

$$\begin{aligned} G_c(s) &= K_p\left(1+\frac{1}{T_i s}+T_d s\right) \\ &= 1.2\frac{T}{L}\left(1+\frac{1}{2Ls}+0.5Ls\right) \\ &= 0.6\frac{\left(s+\frac{1}{L}\right)^2}{s} \end{aligned} \tag{21}$$

Thus, the PID controller has a pole at the origin and double zeros at $= -1/L$.

Second Method: In the second method, we first set $T_i = \infty$ and. $T_d = 0$ Using the proportional control action only (see Figure 8), increase K_p from 0 to a critical value K_{cr} at which the output first exhibits sustained oscillations. (If the output does not exhibit sustained oscillations for whatever value K_p may take, then this method does not apply.) Thus, the critical gain K_{cr} and the corresponding period P_{cr} are experimentally

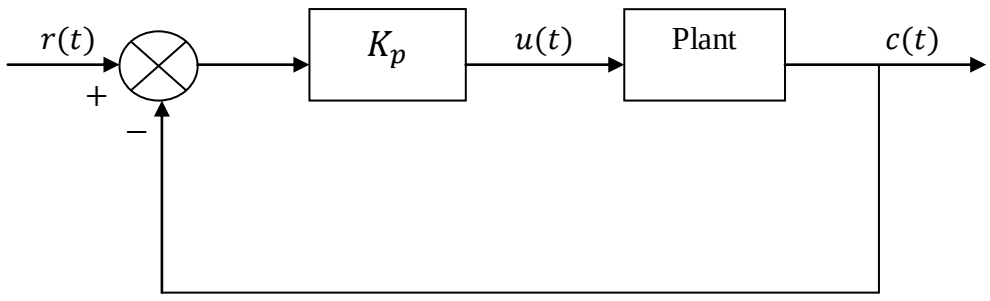


Figure 8 Closed-loop system with a proportional controller

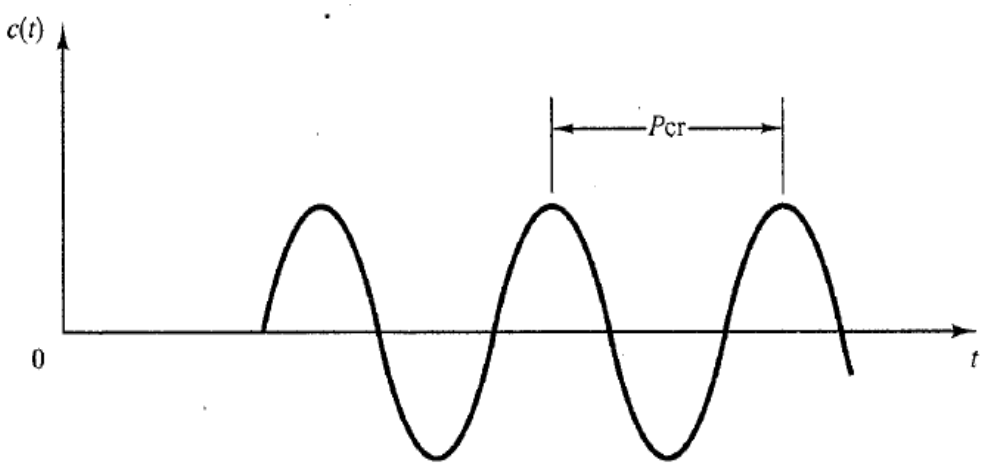


Figure 9 Sustained Oscillation with period $\boldsymbol{P_{cr}}$

Determined (see Figure 9). Ziegler and Nichols suggested that we set the values of the parameters K_p ,T_i and T_d according to the formula shown in Table 4.

Table 4 Ziegler-Nichols Tuning Rule Based on Critical $\boldsymbol{K_{cr}}$ and Critical Period $\boldsymbol{P_{cr}}$ (Second Method)

Type of Controller	K_p	T_i	T_d
P	$0.5K_{cr}$	∞	0
PI	$0.45K_{cr}$	$\frac{1}{1.2}P_{cr}$	0
PID	$0.6K_{cr}$	$0.5P_{cr}$	$0.125P_{cr}$

Notice that the PID controller tuned by the second method of Ziegler-Nichols rules gives

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

$$= 0.6K_{cr}\left(1 + \frac{1}{0.5P_{cr}s} + 0.125P_{cr}s\right) \qquad (22)$$

$$= 0.075K_{cr}P_{cr}\frac{\left(s + \frac{4}{P_{cr}}\right)^2}{s}$$

Thus, the PID controller has a pole at the origin and double zeros at $= -4/P_{cr}$.

Note that if the system has a known mathematical model (such as the transfer function), then we can use the root-locus method to find the critical gain K_{cr} and the frequency of the sustained oscillations ω_{cr}, where $2\pi/\omega_{cr} = P_{cr}$. These values can be found from the crossing points of the root-locus branches with the $j\omega$ axis. (Obviously, if the root-locus branches do not cross the $j\omega$ axis, this method does not apply.)

Comments: Ziegler-Nichols tuning rules (and other tuning rules presented in the literature) have been widely used to tune PID controllers in process control systems where the plant dynamics are not precisely known. Over many years, such tuning rules proved to be very useful. Ziegler-Nichols tuning rules can, of course, be applied to plants whose dynamics are known. (If the plant dynamics are known, many analytical and graphical approaches to the design of PID controllers are available, in addition to Ziegler-Nichols tuning rules.)[3].

3.2 Position Control Design by Ziegler-Nichols method

The control system shown in figure 10 in which a PID is used to control the system. The PID controller has the transfer function

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right) \tag{23}$$

Although many analytical methods are available for the design a PID controller for the present system, let us apply a Ziegler-Nichols tuning rule for the determination of the values of parameters K_p, T_i and T_d

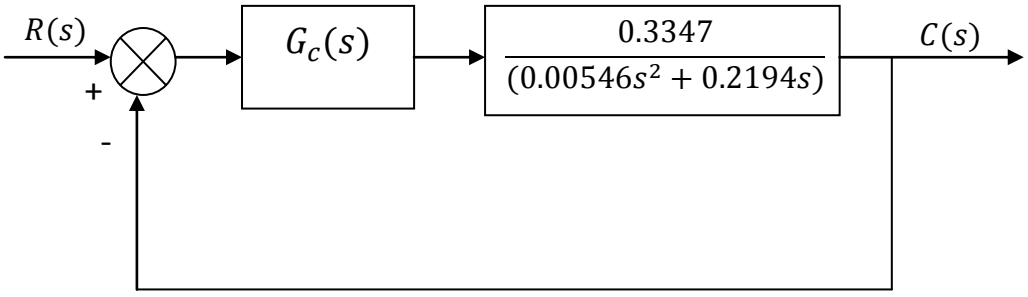


Figure 10 The Controller System

3.2.1 Ziegler-Nichols rules for tuning PID controllers

In this research we will use second method of Ziegler-Nichols tuning rules. By setting $T_i = \infty$ and $T_d = 0$, we obtain the closed-loop transfer function as following:

$$\frac{C(s)}{R(s)} = \frac{0.3347K_p}{0.00546s^2 + 0.2194s + 0.3347K_p} \tag{24}$$

The plant characteristic equation is:

$$0.00546s^2 + 0.2194s + 0.3347K_p = 0$$

The value of K_p that makes the system Critical stable so that sustained oscillation occurs can be obtained by use the following:

Let $\zeta = 1$

The standard second-order transfer function

$$\frac{C(s)}{R(s)} = \frac{{\omega_n}^2}{s^2 + 2\zeta\omega_n s + {\omega_n}^2} \tag{25}$$

The characteristic equation is

$$s^2 + 2\zeta\omega_n s + {\omega_n}^2 = 0$$

Comparing the plant characteristic equation given in (24) with the standard second-order characteristic equation (25), we find

$$2\zeta\omega_n = 0.2194$$

$$\omega_n = \frac{0.2194}{2} = 0.11$$

$$\omega_n = 0.11$$

$$0.11 = 0.3347K_{cr}$$

$$K_{cr} = \frac{0.11}{0.3347} = 0.33$$

With gain K_p set to equal $K_{cr} = 0.33$ The characteristic equation becomes

$$0.00546s^2 + 0.2194s + 0.33 = 0$$

To find the frequency of the sustained oscillation. We substitute $s = j\omega$ into this characteristic equation as follow:

$$-0.00546\omega^2 + 0.2194j\omega + 0.33 = 0$$

$$-0.00546\omega^2 + 0.33 = 0$$

$$0.00546\omega^2 = 0.33$$

$$\omega^2 = 60.44$$

$$\omega = 7.77$$

From which we find the frequency of the sustained oscillation to be $\omega = 7.77$.

Hence, the period of sustained oscillation is

$$P_{cr} = \frac{2\pi}{\omega} = \frac{2\pi}{7.77} = 0.808$$

Referring to Table (4), we determine K_p, T_i and T_d as follows:

$$K_p = 0.6K_{cr} = 0.2$$

$$T_i = 0.5P_{cr} = 0.404$$

$$T_d = 0.125P_{cr} = 0.101$$

The transfer function of the PID controller is:

$$G_c(s) = K_p(1 + \frac{1}{T_i s} + T_d s)$$

$$G_c(s) = 0.2(1 + \frac{1}{0.404s} + 0.101s)$$

$$G_c(s) = \frac{(0.0808s + 0.2 + 0.01s^2)}{0.404s}$$

$$G_c(s) = \frac{(0.025s^2 + 0.2s + 0.55)}{s}$$

The PID controller has a pole at the origin and double zero at s = -4

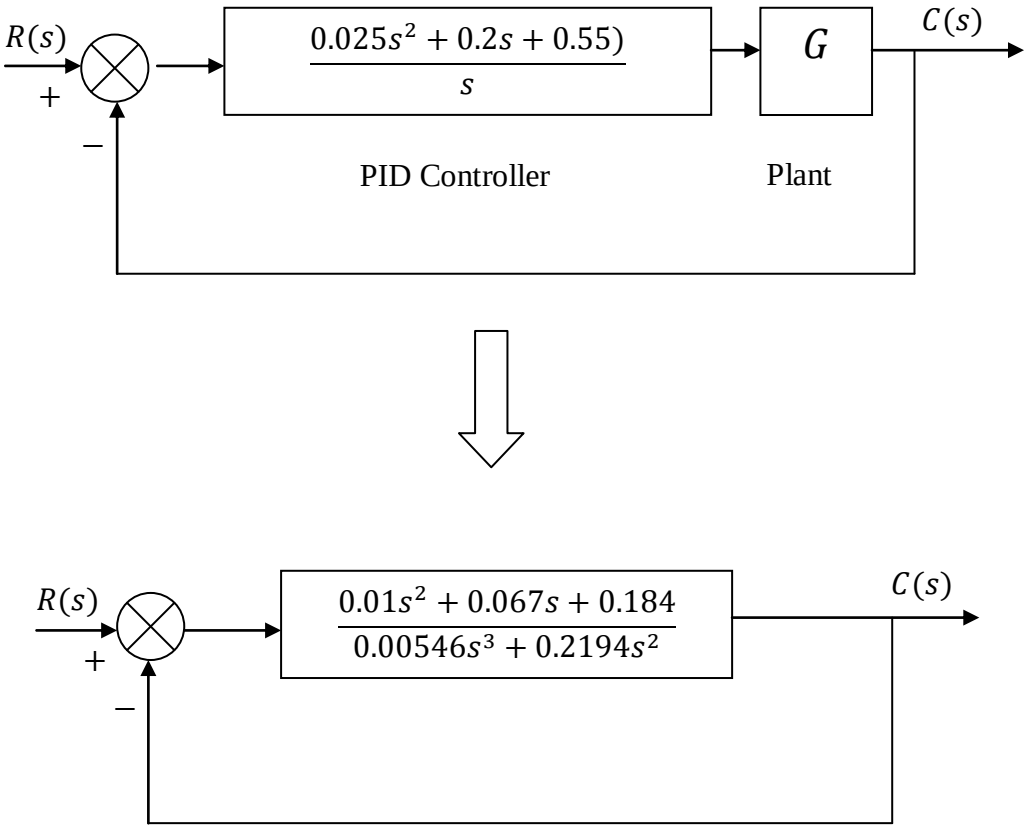


Figure 11 Block diagram of the plant with PID Controller designed by use of Ziegler-Nichols tuning rule (second method)

The closed-loop transfer function $C(s)/R(s)$ is given by

$$\frac{C(s)}{R(s)} = \frac{0.01s^2 + 0.067s + 0.184}{0.00546s^3 + 0.2294s^2 + 0.067s + 0.184}$$

The unit-step response of this system can be obtained easily with MATLAB. See MATLAB program (1), the resulting unit-step response curve is shown in figure (12)

MATLAB program (1)

```
num = [0     0.01     0.067     0.184];

den = [0.00546      0.2294       0.067     0.184];

step (num,den)
```

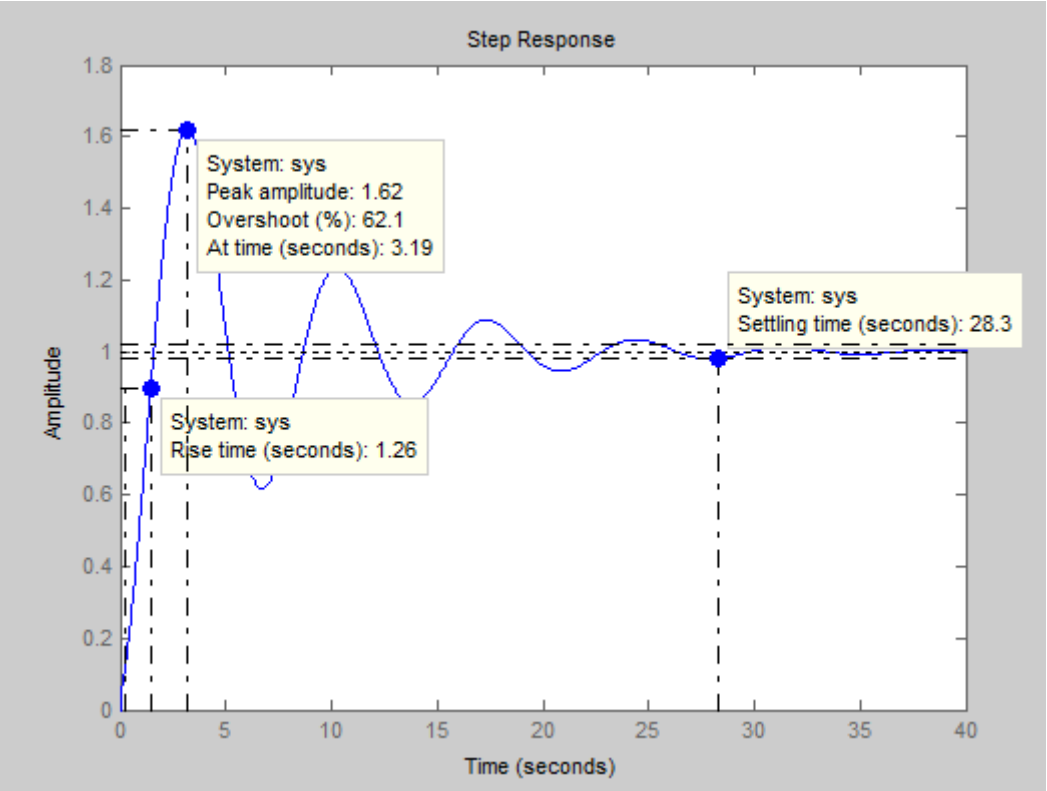


Figure 12 Unit-Step Response for the plant and PID Controller

From the figure (12) for unit step response for the plant and PID controller we note the overshoot, rise time and settling time are increase. We need to reduce it.

By increase K_p =10 , $T_i = 10$ and $T_d = 0.101$

$$G_c(s) = 10(1 + \frac{1}{10s} + 0.101s)$$

$$G_c(s) = \frac{(1.01s^2 + 10s + 1)}{s}$$

The PID controller has a pole at the origin and double zero at $s_1 = -9.8$

$s_2 = -0.101$

$$\frac{C(s)}{R(s)} = \frac{0.338s^2 + 3.347s + 0.3347}{0.00546s^3 + 0.5574s^2 + 3.347s + 0.3347}$$

the resulting unit-step response curve is shown in figure (13)

MATLAB program (2)

num =[0 0.338 3.347 0.3347];

den = [0.00546 0.5574 3.347 0.3347];

step(num,den)

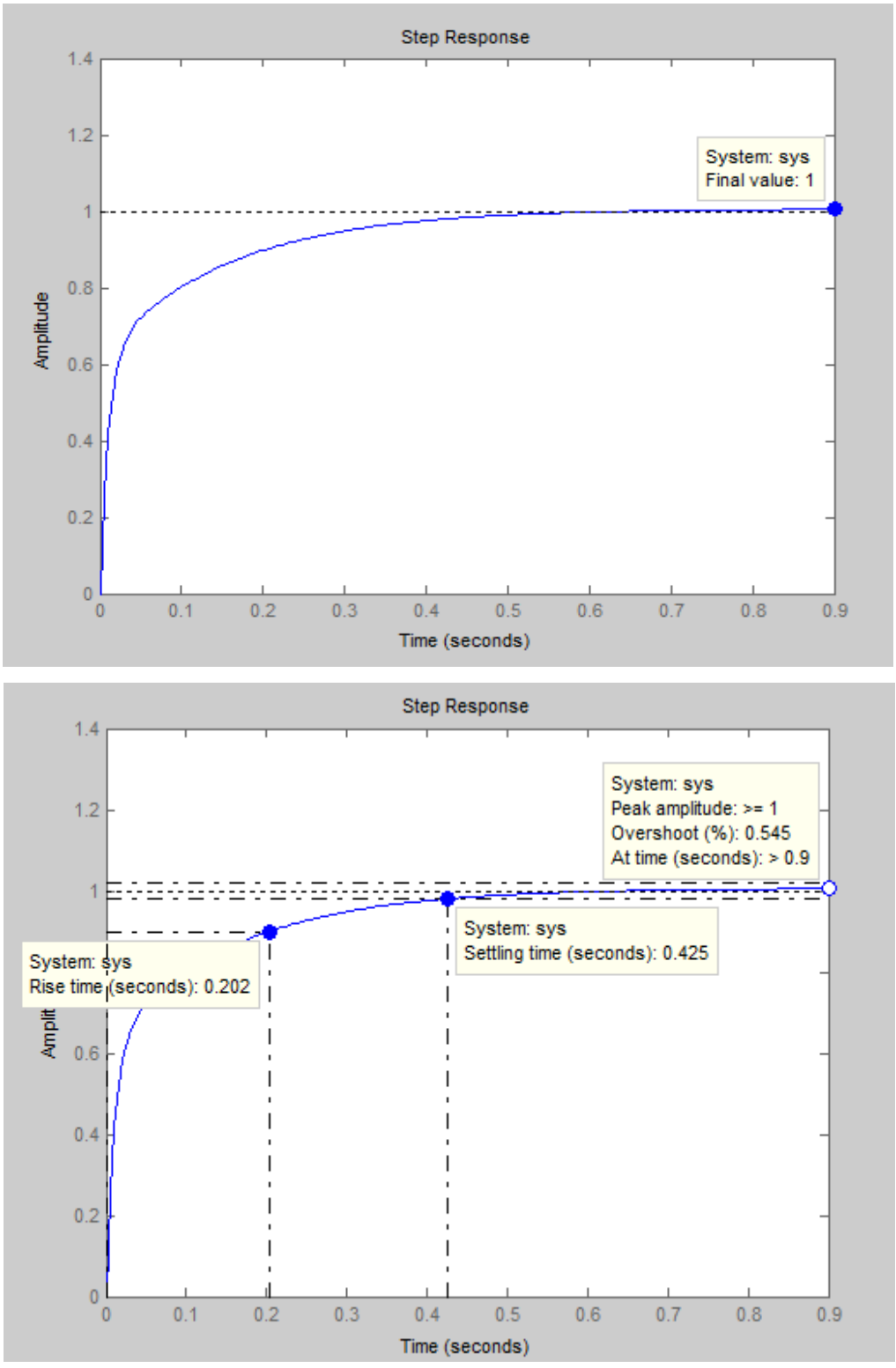


Figure 13 Unit-Step Response for the plant and PID Controller

I found good response as show in the figure (15). when the parameters of PID controller are: K_p = 10 & K_i = 0.1 and K_d = 0.101

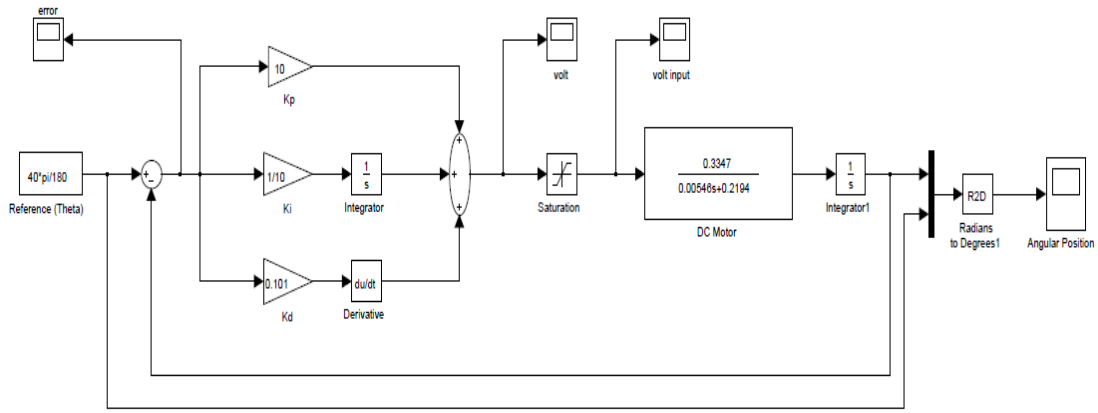


Figure 14 Simulink diagram of DC motor

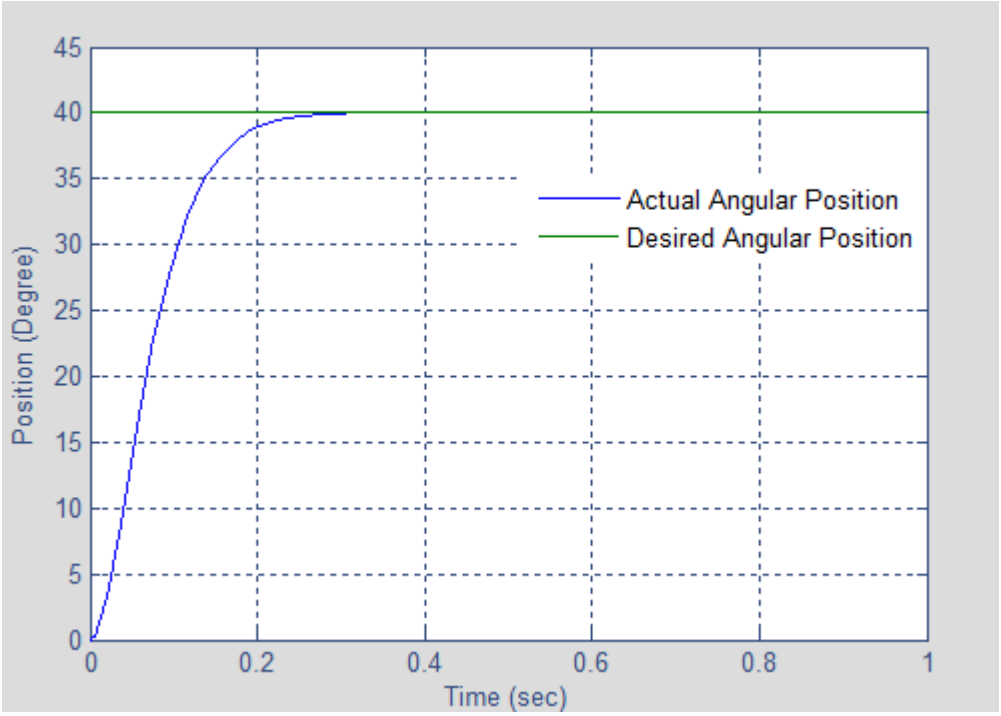


Figure 15 Position Control Response using Ziebler-Nichols tuning rule

Rise time (seconds): 0.147 & Overshoot (%): 0.08

Settling time (seconds): 0.211 & Accuracy Error (%): 0.077

CHAPTER 4

FUZZY LOGIC

4.1 Introduction

Fuzzy logic has rapidly become one of the most successful of today's technologies for developing sophisticated control systems. The reason for which is very simple. Fuzzy logic addresses such applications perfectly as it resembles human decision making with an ability to generate precise solutions from certain or approximate information. It fills an important gap in engineering design methods left vacant by purely mathematical approaches (e.g. linear control design), and purely logic-based approaches (e.g. expert systems) in system design. While other approaches require accurate equations to model real-world behaviours, fuzzy design can accommodate the ambiguities of real-world human language and logic. It provides both an intuitive method for describing systems in human terms and automates the conversion of those system specifications into effective models [30].

Why Fuzzy Logic ?

- Fuzzy self-tuning PID parameters controller can automatically adjust PID parameters in accordance with the error and the rate of error-change.
- Ability to translate imprecise vague knowledge of human experts.
- Simple, easy to implement technology.
- Rules contribute to inferences even when facts do not exactly match antecedent and Rule-based systems may be analyzed and improved.
- Technology is easy to transfer from product to product.
- Robust and Smooth controller behavior.
- Ability to control unstable systems [39].

4.2 Fuzzy sets

According to Cantor a set X is a collection of definite, distinguishable objects of our intuition which can be treated as a whole. The objects are the members of X. The concept 'objects of our intuition' gives us great freedom of choice, even sets with

infinitely many members. Objects must be 'definite': given an object and a set, it must be possible to determine whether the object is, or is not, a member of the set. Objects must also be 'distinguishable': given a set and its members, it must be possible to determine whether any two members are different, or the same.

The members completely define a set. To determine membership, it is necessary that the sentence x is a member of X, where x is replaced by an object and X by the name of a set, is either true or false. We use the symbol ∈ and write $x \in X$ if object x is a member of the set X. The assumption that the members determine a set is equivalent to saying: Two sets X and Y are equal, X = Y, iff (if and only if) they have the same members. The set whose members are the objects x_1, x_2, . . . , x_n is written $\{x_1, x_2, \dots, x_n\}$ [35].

Fuzzy set: in a universe of discourse U is characterized by a membership function $\mu_A(X)$ that takes values in the interval [0, 1].
In classical sets the membership function of a classical set can only take zero and one
In fuzzy set the membership function is a continuous function with range [0, 1].[26]

4.3 Operations on fuzzy sets

We extend the classical set theoretic operations from ordinary set theory to fuzzy sets. We note that all those operations which are extensions of crisp concepts reduce to their usual meaning when the fuzzy subsets have membership degrees that are drawn from {0, 1}. For this reason, when extending operations to fuzzy sets we use the same symbol as in set theory.

Let A and B are fuzzy subsets of a nonempty (crisp) set X.

4.3.1 Union of Two Fuzzy Sets

The union of two fuzzy sets A and B defined over the same universe of discourse X is a new fuzzy set A∪B also on X with membership function which is the maximum of the grades of membership function of every x to A and B:

$$\mu_{A\cup B}(X) = \mu_A(X) \vee \mu_B(X) = \max\big(\mu_A(X), \mu_B(X)\big) \qquad (26)$$

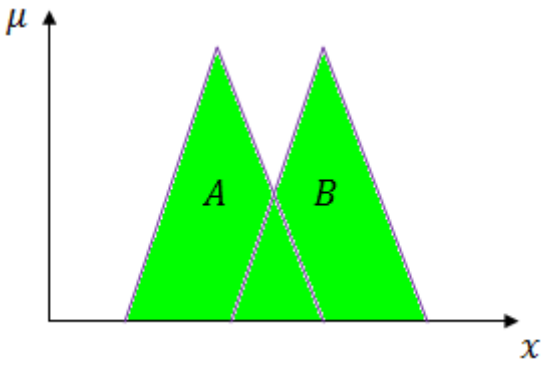


Figure 16 union of two fuzzy sets

4.3.2 Intersection of Two Fuzzy Sets

The intersection of two fuzzy sets A and B is a new fuzzy set A∩B also on X with membership function which is the minimum of the grades of membership function of every x in X to the sets A and B:

$$\mu_{A\cap B}(X) = \mu_A(X) \wedge \mu_B(X) = \min\left(\mu_A(X), \mu_B(X)\right) \tag{27}$$

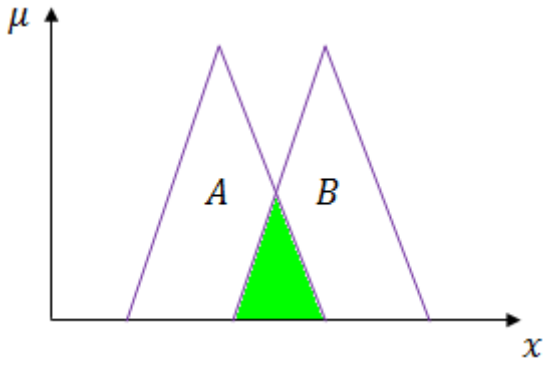


Figure 17 intersection of two fuzzy sets

4.3.3 Complement of a Fuzzy Sets

The complement of a fuzzy set A is a new fuzzy set A also on X with membership function:

$$\mu_{\bar{A}}(X) = 1 - \mu_A(X) \tag{28}$$

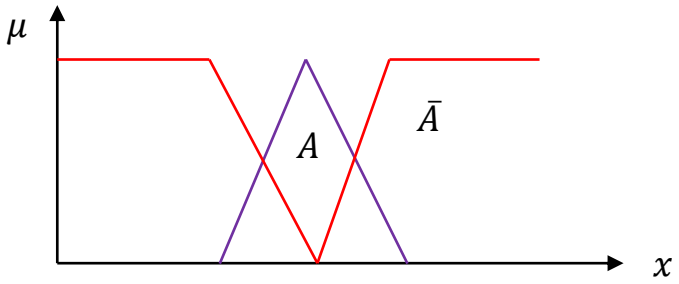


Figure 18 complement of a fuzzy sets

4.4 Membership Functions

We will usually consider one of the following membership functions:

- Triangular: $\text{tri}(x; a, b, c) = max\left\{min\left\{\frac{x-a}{b-a}, \frac{c-x}{c-b}\right\}, 0\right\}$ (29)
- Trapezoidal: $\text{trap}(x; a, b, c, d) = max\left\{min\left\{\frac{x-a}{b-a}, \frac{d-x}{d-b}, 1\right\}, 0\right\}$ (30)
- Gaussian: $\text{gauss}(x; c, \sigma) = exp\left[-\frac{1}{2}\left(\frac{x-c}{\sigma}\right)^2\right]$ (31)
- Generalised Bell: $\text{gbell}(x; a, b) = \frac{1}{1+\left|\frac{x-b}{a}\right|^{2a}}$ (32)

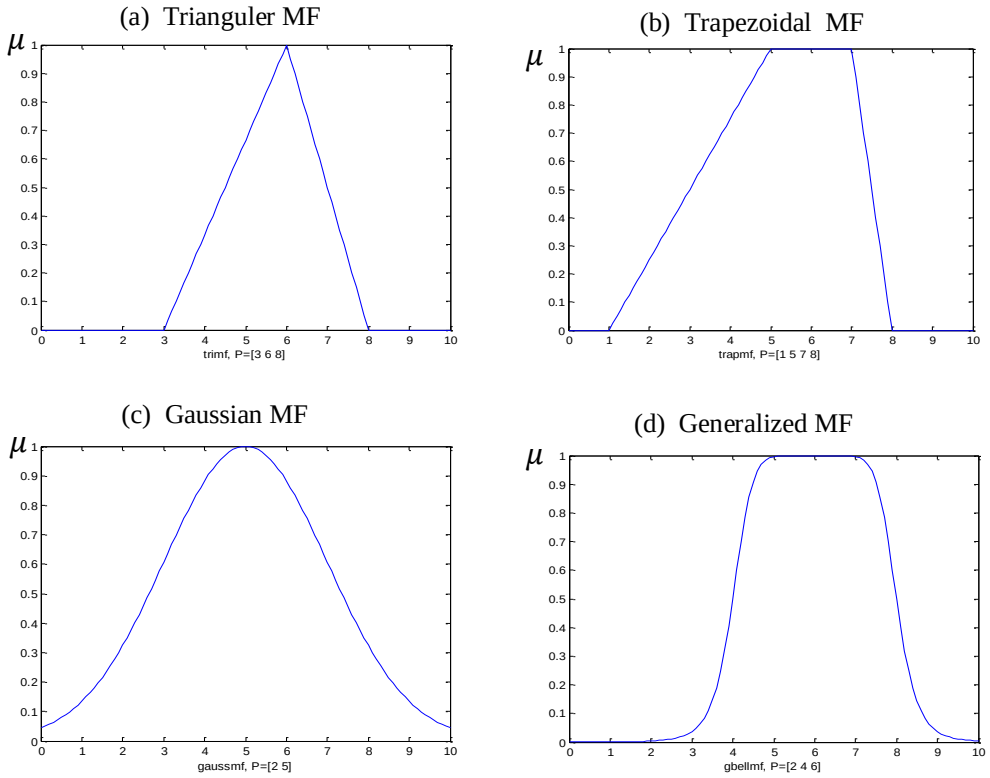


Figure 19 Shape of different Membership Function

4.5 Fuzzy Systems

A Fuzzy System can be contrasted with a conventional (crisp) system in three main ways:

- A Linguistic Variable is defined as a variable whose values are sentences in a natural or artificial language. Thus, if tall, not tall, very tall, very very tall, etc. are values of height, then height is a linguistic variable.
- Fuzzy Conditional Statements are expressions of the form If A THEN B, where A and B have fuzzy meaning, e.g. If x is small THEN y is large, where small and large are viewed as labels of fuzzy sets.
- A Fuzzy Algorithm is an ordered sequence of instructions which may contain fuzzy assignment and conditional statements, e.g., x = very small, IF x is small THEN y is large. The execution of such instructions is governed by the compositional rule of inference and the rule of the preponderant alternative. [33]

4.6 Fuzzy Logic Control System

In contrast to conventional control techniques, fuzzy logic control (FLC) is best utilized in complex ill-defined processes that can be controlled by a skilled human operator without much knowledge of their underlying dynamics.

The basic idea behind FLC is to incorporate the "expert experience" of a human operator in the design of the controller in controlling a process whose input – output relationship is described by collection of fuzzy control rules (e.g., IF-THEN rules) involving linguistic variables rather than a complicated dynamic model.

The utilization of linguistic variables, fuzzy control rules, and approximate reasoning provides a means to incorporate human expert experience in designing the controller.

FLC is strongly based on the concepts of fuzzy sets, linguistic variables and approximate reasoning.

We will introduce the basic architecture and functions of fuzzy logic controller.

A typical architecture of FLC is shown below, which comprises of four principal comprises: a fuzzifier, a fuzzy rule base, inference engine, and a defuzzifier.

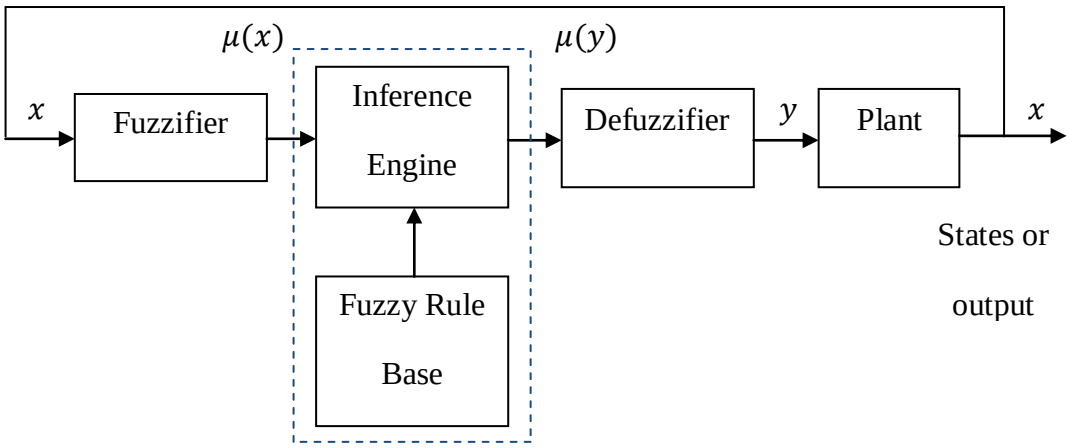


Figure 20 Structure of fuzzy logic controller

If the output from the defuzzifier is not a control action for a plant, then the system is fuzzy logic decision system.

The **fuzzifier** has the effect of transforming crisp measured data (e.g. speed is 10 mph) into suitable linguistic values (i.e. fuzzy sets, for example, speed is too slow).

The **fuzzy rule base** stores the empirical knowledge of the operation of the process of the domain experts.

The **inference engine** is the kernel of a FLC, and it has the capability of simulating human decision making by performing approximate reasoning to achieve a desired control strategy.

The **defuzzifier** is utilized to yield a nonfuzzy decision or control action from an inferred fuzzy control action by the inference engine.

The control action must be in the form of a crisp value. Defuzzification is the process of transforming the fuzzy set assigned to a control output variable into such a crisp value. There are various methods for defuzzification. The following Three are the most prominent in fuzzy control.

Center of gravity (COG)

Bisector of area (BOA)

Mean of maximum (MOM)

a) Center of gravity (COG)

For 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 is calculated as follows:

$$u_{COGS} = \frac{\sum_i \mu_c(x_i) x_i}{\sum_i \mu_c(x_i)} \tag{33}$$

Where x_i is a point in the universe of the conclusion ($i = 1, 2, 3 \ldots$) and $\mu_c(x_i)$ is the membership value of the resulting conclusion set. For continuous sets summations are replaced by integrals.

b) Bisector of area (BOA)

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

$$\left| \sum_{i=1}^{j} \mu_c(x_i) - \sum_{i=j+1}^{i_{max}} \mu_c(x_i) \right|, i < j < i_{max} \tag{34}$$

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

c) Mean of maximum (MOM)

In this method the crisp value is to choose the point with the highest membership. There may be several points in the overall implied fuzzy set which have maximum membership value. Therefore it's a common practice to calculate the mean value of these points. This method is called mean of maximum (MOM) and the crisp value is calculated as follows:

$$u_{MOM} = \frac{\sum_{i \in I} x_i}{|I|}, I = \{i | \mu_c(x_i) = \mu_{max}\} \tag{35}$$

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) [29].

CHAPTER 5

FUZZY SELF-TUNING PID CONTROLLER

5.1 Literature Review

Fuzzy self-tuning PID Controller is widely used in many applications in industry. Therefore, a lot of literature is available related with this topic:

Tao C. W. in 2000 had proposed a flexible complexity reduced design approach for PID-like fuzzy controllers. With the linear combination of input variables as a new input variable, the complexity of the fuzzy mechanism of PID-like fuzzy controllers is significantly reduced. However, the performance of the complexity reduced fuzzy PID controller may be degraded since the degree of freedom is decreased by the combination of input variables. To alleviate the drawback and improve the performance of the complexity reduced PID-like fuzzy controller, a flexible complexity reduced design approach is introduced in which the functional scaling factors are heuristically generated. Since the functional scaling factors are heuristically created, they can be easily adjusted for the flexible complexity reduced PID-like fuzzy controller without a priori knowledge of the exact mathematical model of the plant. Moreover, heuristic scaling factors are implemented as functionals. Therefore, the complexity of the flexible PID-like fuzzy controller will not be increased. Further, the stability of the fuzzy control system with a flexible complexity reduced PID-like fuzzy controller is discussed [12].

According to Gawthrop, P.J.'s paper in 1996 'Self-tuning PID control structures', Multiple-model self-tuning PID controllers give a neat way of handling nonlinear or time varying systems. The basic concepts of PID control can be generalized within the same structure but allowing for the control of complicated dynamic systems using advanced control design algorithms. The structure arises naturally from the system description and does not need to be imposed artificially. Recent advances in Local Model networks give a neat extension of the basic (generalized) PID structure to handle nonlinear or time varying systems [9].

According to Xie, W.F. (1998) , in 'Fuzzy Adaptive Internal Model Control', Fuzzy

adaptive internal model control scheme consists of two main parts - 1. Fuzzy dynamic model, 2. Fuzzy model-based controller. Fuzzy dynamic model is identified on-line by using the input and output measurement of the plant. It serves as the internal model and tries to track the output of plant adaptively. The fuzzy model-based controller is designed to point wise minimize an H, -performance objective based on the identified fuzzy model. It aims at improving the robust performance of the close-loop control system. The application of fuzzy adaptive internal model controller in the laboratory scale Process Control Unit(PCU) from Bytronic shows that this kind of control scheme is appropriate for controlling the time-varying stable plant with time-delay. The whole control system possesses very satisfactory robust performance [11].

Datta, A. in 1998, developed a systematic theory for the design and analysis of adaptive internal model control schemes which is presented in paper 'The Theory and Design of Adaptive Internal Model Control Schemes'. The ubiquitous Certainty Equivalence principle of adaptive control is used to combine a robust adaptive law with robust internal model controllers to obtain adaptive internal model control schemes with provable guarantees of stability and robustness. Specific controller structures considered include those of the model reference, "partial" pole placement, and Hz, optimal control types. The results here not only provide a theoretical basis for analytically justifying some of the reported industrial successes of existing adaptive internal model control schemes but also open up the possibility of synthesizing new ones by simply combining a robust adaptive law with a robust internal model controller structure [10].

According to Watanabe, K. in 2003, the conventional IMC consists of the forward model of the plant, its inverse-model and low-pass filter. In this paper, they presented a scheme of adaptive IMC for uncertain plants. The forward model was adaptively identified by using the error between the output of the real plant and the model. The inverse model was constructed by the transcription of the parameters of the forward model. The transcription requires the inverse of one parameter and it was performed in adaptive mechanism [14].

According to Zhiqiang Gao in 2002 a closed-loop control system incorporating fuzzy logic has been developed for a class of industrial temperature control problems. A unique fuzzy logic controller (FLC) structure with an efficient realization and a small rule base that can be easily implemented in existing industrial controllers was proposed. The potential of FLC in both software simulation and hardware test in an industrial setting was demonstrated. This includes compensating for thermo mass changes in the system, dealing with unknown and variable delays, operating at very different temperature set points without retuning, etc. It was achieved by implementing, in the FLC, a classical control strategy and an adaptation mechanism to compensate for the dynamic changes in the system. The proposed FLC was applied to two different temperature processes and performance and robustness improvements were observed in both cases. Furthermore, the stability of the FLC was investigated and a safeguard was established [13].

According to Guihua Han in 2005, in order to satisfy the higher control performance requirement of the industrial steam turbine governing system, the electro-hydraulic servo system and turbine governor based on fuzzy PID control are researched. This study presented a nonlinear self-adaptive fuzzy PID controller adopting fuzzy rule and inference to adjust PID parameters on-line, which greatly improves the robustness, the dynamic and static properties of the system. Additionally, the idea of variable universe was employed to improve the disadvantage that the membership functions and the control rules could never be modified once they were defined. The proposed fuzzy controller showed excellent robustness against variations of system parameters and external disturbances by comparison with the commonly used classical PID controller or the fixed fuzzy controller via simulations and experiments, which was useful for the future theory research and practice in hydraulic turbine regulation [17].

Xiu-Zhang Jin in 2004 proposed a scheme of adaptive IMC for unstable process with dead time. A negative feedback element was used to make this object stable, and adaptive internal model control strategy was adopted to design a control system. Because of the shortage of the classical internal model control system the performance of the control system will slip back if there exist large errors between the real plant and the model, an adaptive mechanism based single neuron which can

tune the parameters of the internal model and that of controller in the control system on line is designed. The simulation results showed that the adaptive internal model control system designed in their paper had good performance of overcoming disturbance and deviations of model parameters [16].

Xianwen Gao in 2006 presented an application of Fuzzy Adaptive PID Control in Coke Oven Temperature Control System aiming at the coke oven's temperature characteristics of great inertia, pure time-delay, non-linear and time changeable based on the immune feedback regulating law and the adaptive ability of fuzzy logic ratiocination. The academic analysis and simulation results of coke oven's simple model indicate the feasibility and effectiveness the control method [18].

P.M. Mary. in 2009 had given an improvement over the existing conventional fuzzy logic approach, based on a self-tuning fuzzy logic controller (FLC), for the design of a temperature control process, capable of providing optimal performance over the entire operating range of the process. Since an optimum response of the FLC could be expected only for a limited range of inputs, tuning the input and output gains were done for various range of inputs. The proposed control system had the advantages of self-tuning FLC schemes. To evaluate the performance of the proposed control system methods, the results from the simulation of the process were presented [21].

Juan Chen in 2008 proposed a modified internal model control method is based on internal model control (IMC) aiming at unstable processes with large time delay in chemical industry. The structure of new control method consists of inner-loop control and outer-loop control. The separately design step was used to design forward-feedback controller for disturbance rejection and the IMC controller for set-point tracking. Meanwhile, a method of choosing the inner-loop controller is presented. Simulation results showed that the method presented was not only effective for the dynamics and the stability of control system but also effective for the process robustness. Since the robustness can be improved by tuning the filter time constant when the model mismatch exits between the process model and plant [19], [29].

5.2 Principle of Fuzzy self-tuning PID controller

The principle of fuzzy self-tuning PID is firstly to find out the fuzzy relationship between three parameters of PID and error(e) and error changes(ec). Fuzzy inference engines modify three parameters to be content with the demands of the control system online through constantly checking $e(e = r - y)$ and $\left(ec = \frac{de}{dt}\right)$

$$de(t) = e(t) - e(t-1). \quad (36)$$

Thus, the real plant will have better dynamic and steady performance. The structure of fuzzy self tuning PID is just as figure 21.

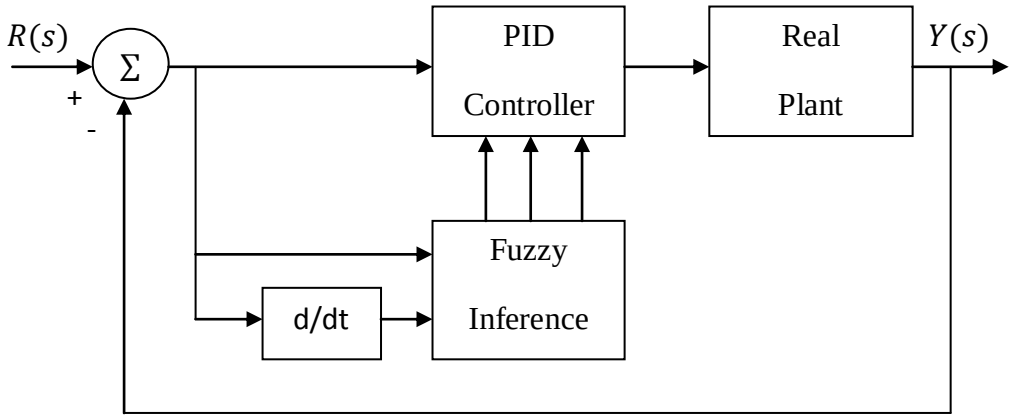


Figure 21 Fuzzy self- tuning PID structure

The unity feedback controller can be realized by a PID controller with filter, then the internal model control can be found approximately through parameter-tuning of PID controller [29].

The mean advantage of Fuzzy self-tuning PID parameters controller is:

The three parameters K_p, K_i, K_d of conventional PID control need to be constantly adjust adjusted online in order to achieve better control performance. Fuzzy self-tuning PID parameters controller can automatically adjust PID parameters in accordance with the error and the rate of error-change [38].

5.3 Control Design of Fuzzy self-tuning PID controller

We have two input variables error (e) and rate of change in error (ec) and three output variables K_p, K_i and K_d.

The Membership Functions for Fuzzy input variables

Table 5 Membership functions for fuzzy input and output variables

NB	NEGATIVE BIG
NS	NEGATIVE SMALL
ZE	ZERO
PS	POSITIVE SMALL
PB	POSITIVE BIG

The membership function for all inputs are shown in figures 22 and 23. By using MATLAB Simulation:

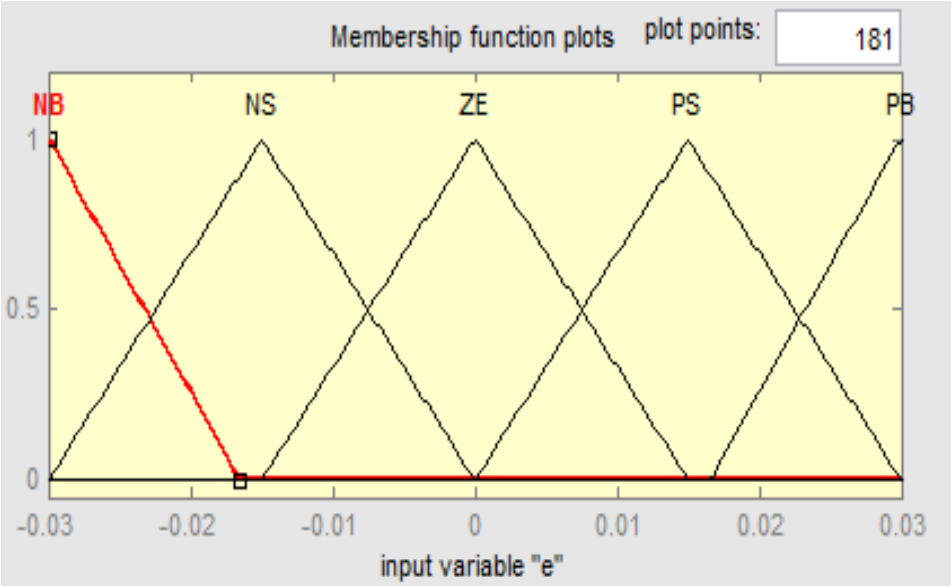


Figure 22 Membership functions for error 'e'

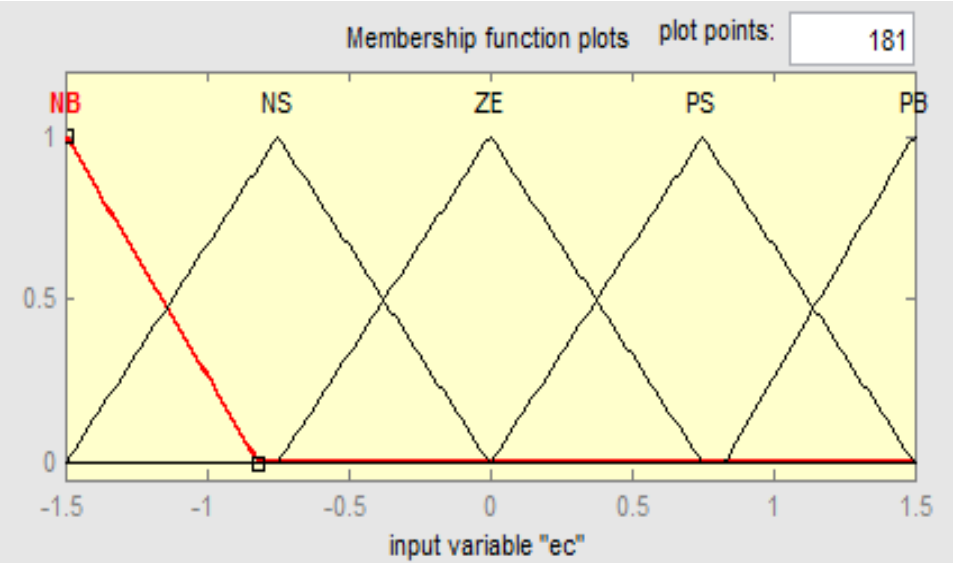


Figure 23 Membership functions for error derivative 'ec'

5.3.1 Fuzzy rules

We have two inputs variables error (e) and rate of change in error (ec) and three output variables K_p, K_i and K_d. Result in 25 rules . These rules are defined using the linguistic variables, shows figure 24.

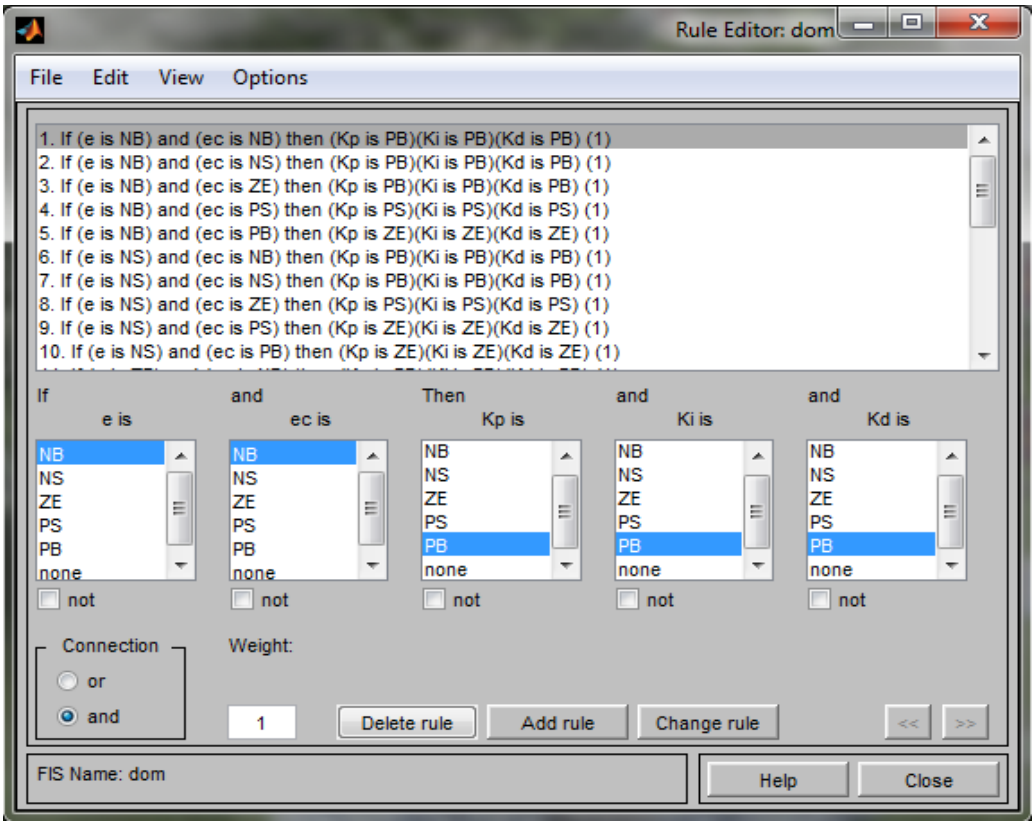


Figure 24 Rule Editor

Table 6 Rule base for K_p

e \ ec	NB	NS	ZE	PS	PB
Kp	NB	NS	ZE	PS	PB
NB	PB	PB	PB	PS	ZE
NS	PB	PB	PS	ZE	ZE
ZE	PB	PS	ZE	NS	NB
PS	ZE	ZE	NS	NB	NB
PB	ZE	NS	NB	NB	NB

The membership functions for fuzzy output variables

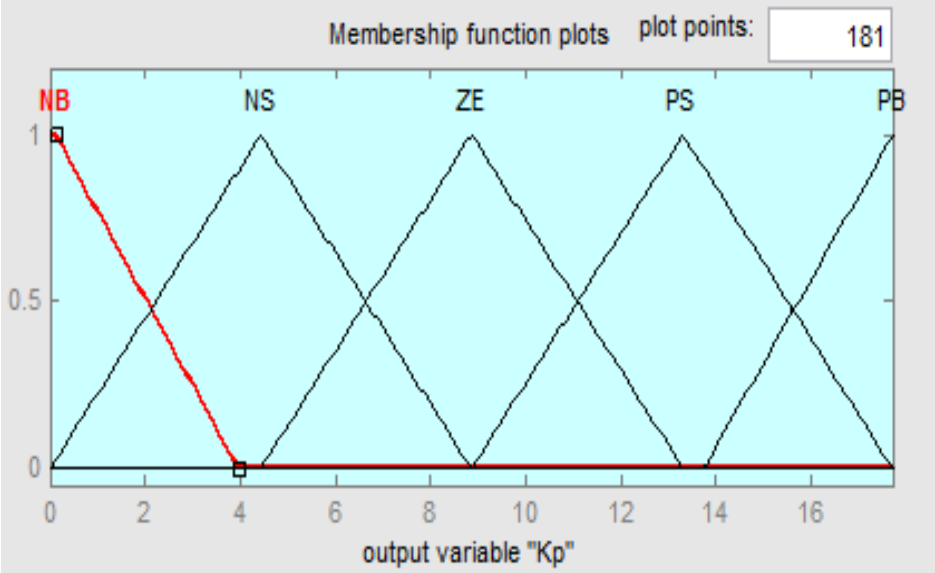


Figure 25 Membership functions for K_p

Table 7 Rule base for K_i

e \ ec	NB	NS	ZE	PS	PB
NB	PB	PB	PB	PS	ZE
NS	PB	PB	PS	ZE	ZE
ZE	PB	PS	ZE	NS	NB
PS	ZE	ZE	NS	NB	NB
PB	ZE	NS	NB	NB	NB

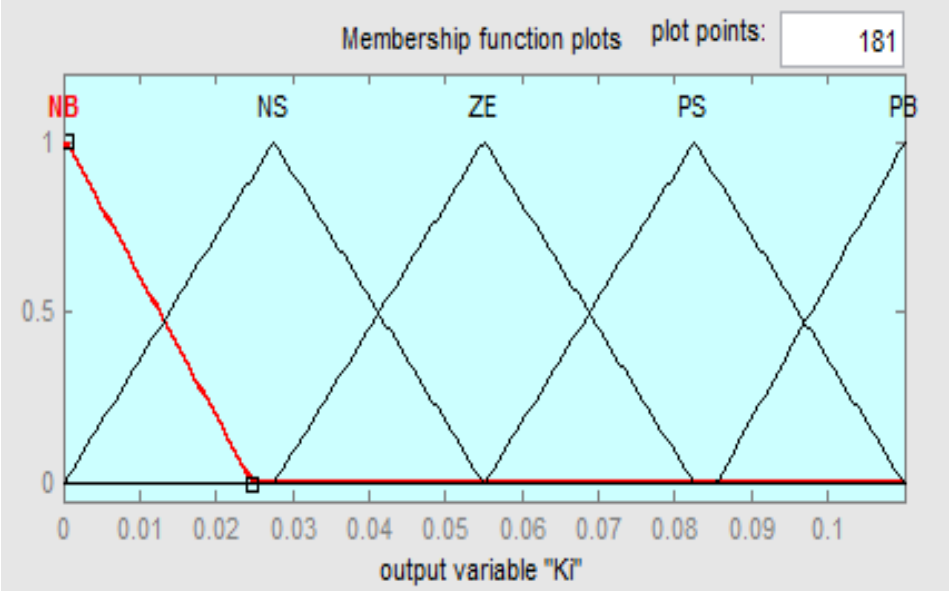


Figure 26 Membership functions for K_i

Table 8 Rule base for K_d

e \ ec	NB	NS	ZE	PS	PB
NB	PB	PB	PB	PS	ZE
NS	PB	PB	PS	ZE	ZE
ZE	PB	PS	ZE	NS	NB
PS	ZE	ZE	NS	NB	NB
PB	ZE	NS	NB	NB	NB

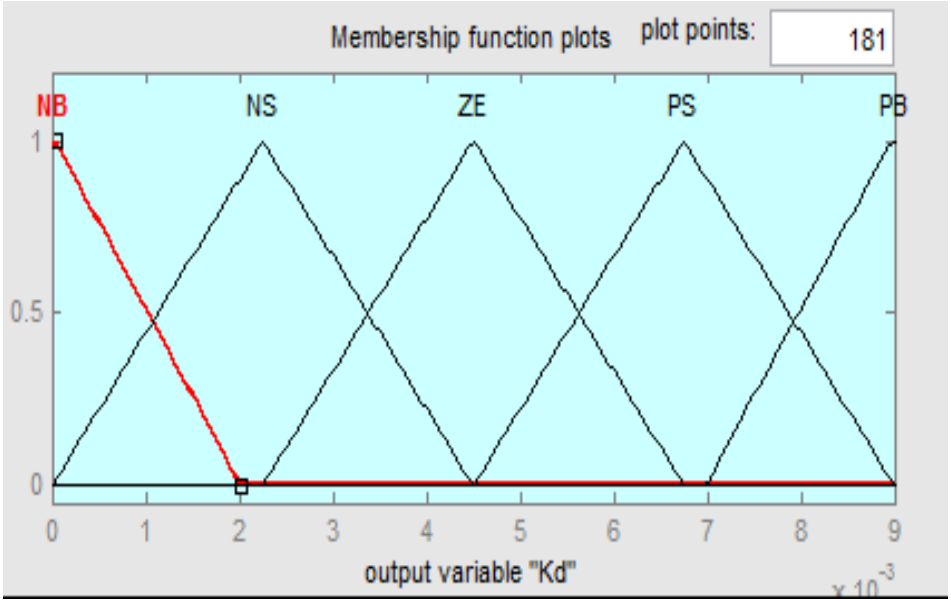


Figure 27 Membership functions for K_d

5.3.2 Simulink Model for Position Control of DC motor using self tuned fuzzy PID Controller

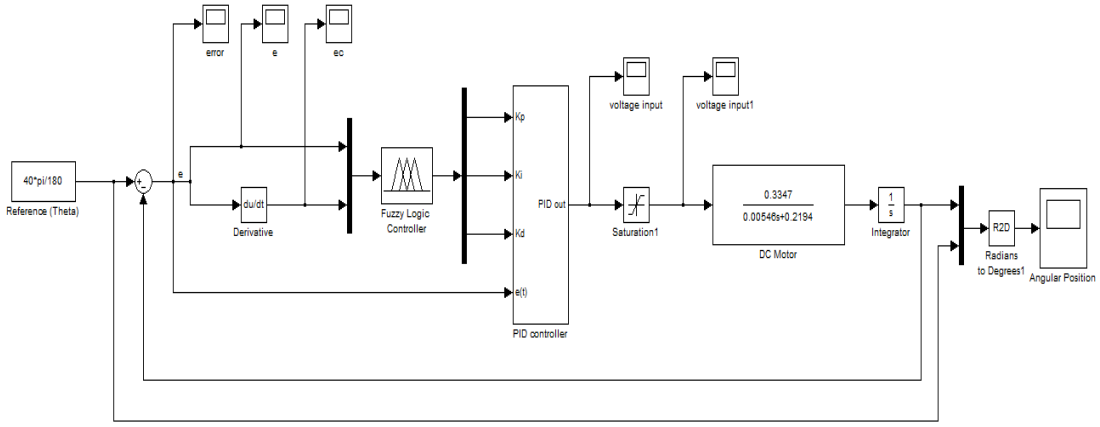


Figure 28 Simulink Model for Position Control of DC motor using self tuned fuzzy PID controller

5.3.3 Fuzzy inference system

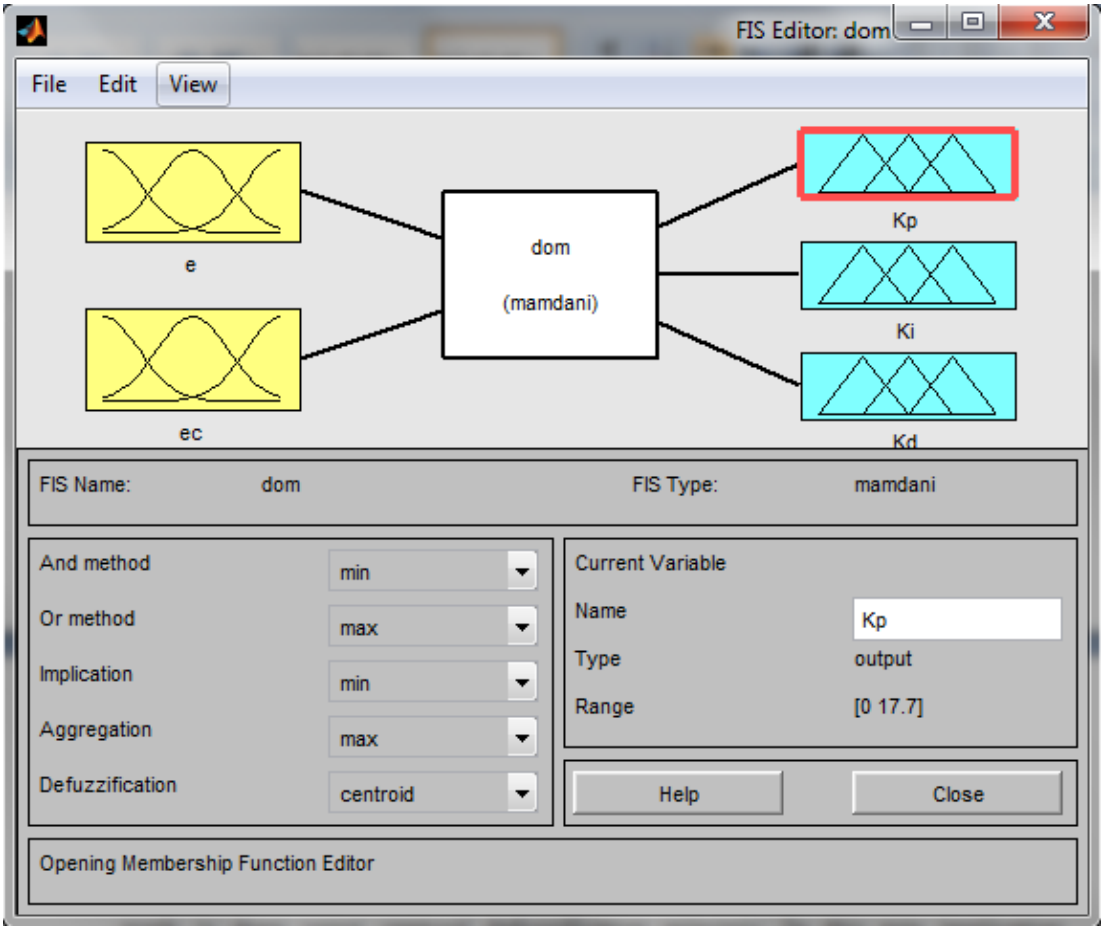


Figure 29 Fuzzy inference system

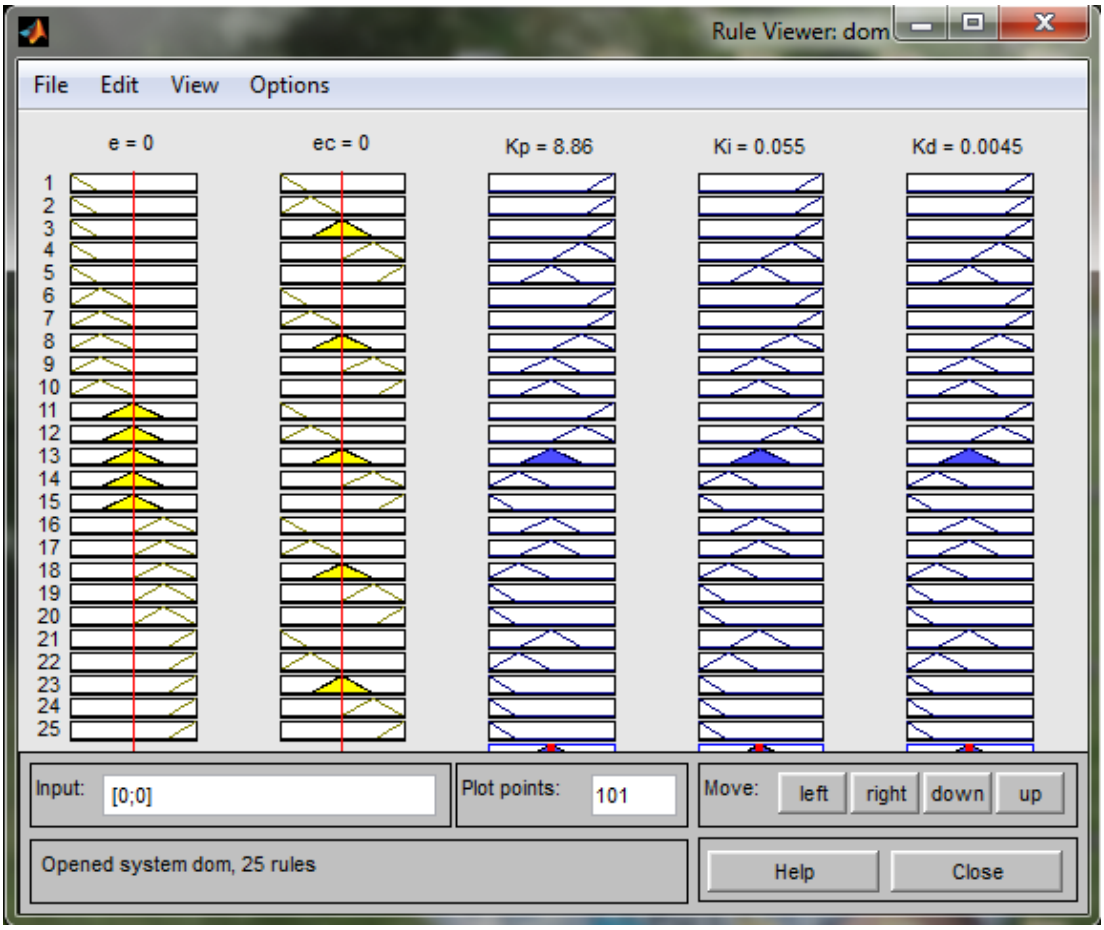


Figure 30 Rule viewer

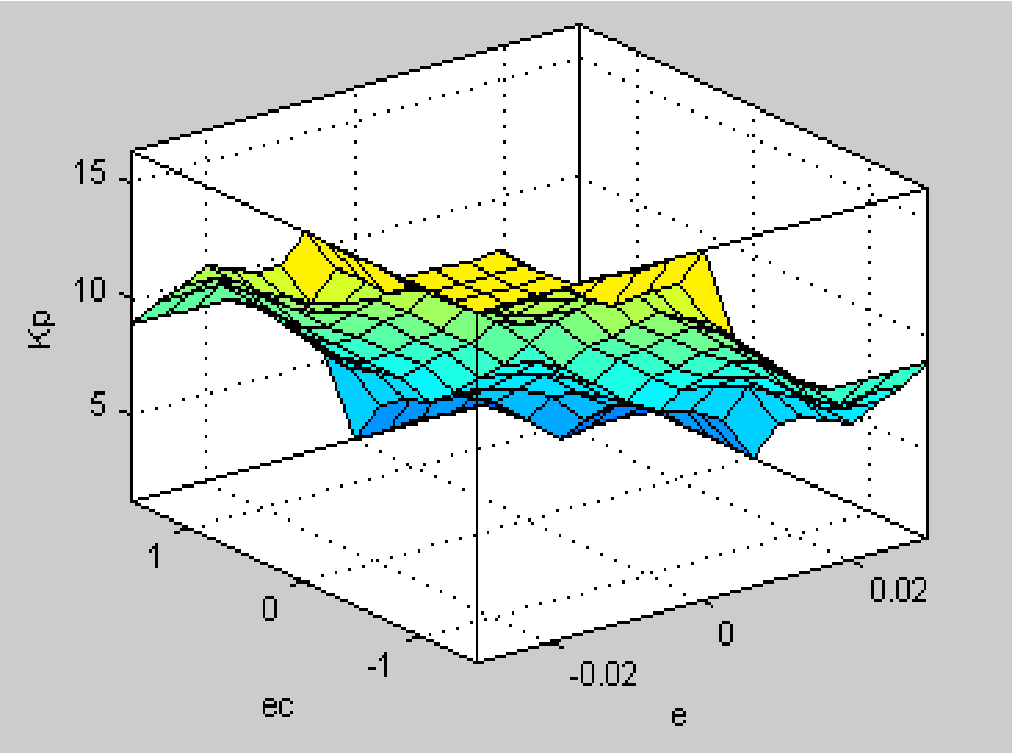


Figure 31 Rule surface viewer of $\boldsymbol{K_p}$

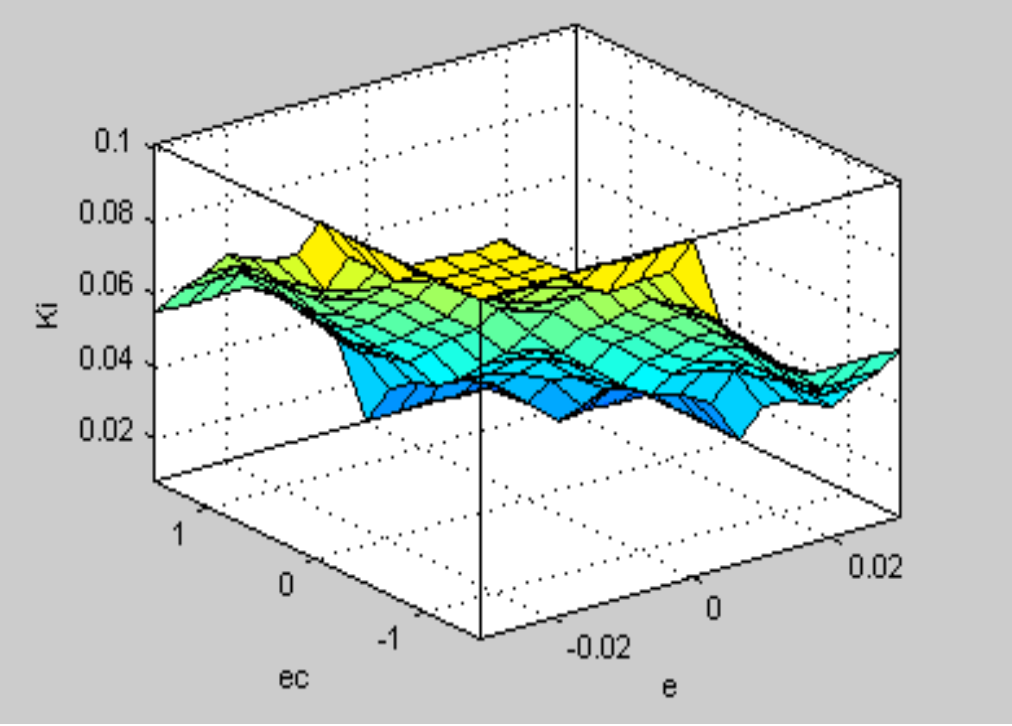


Figure 32 Rule surface viewer of $\boldsymbol{K_i}$

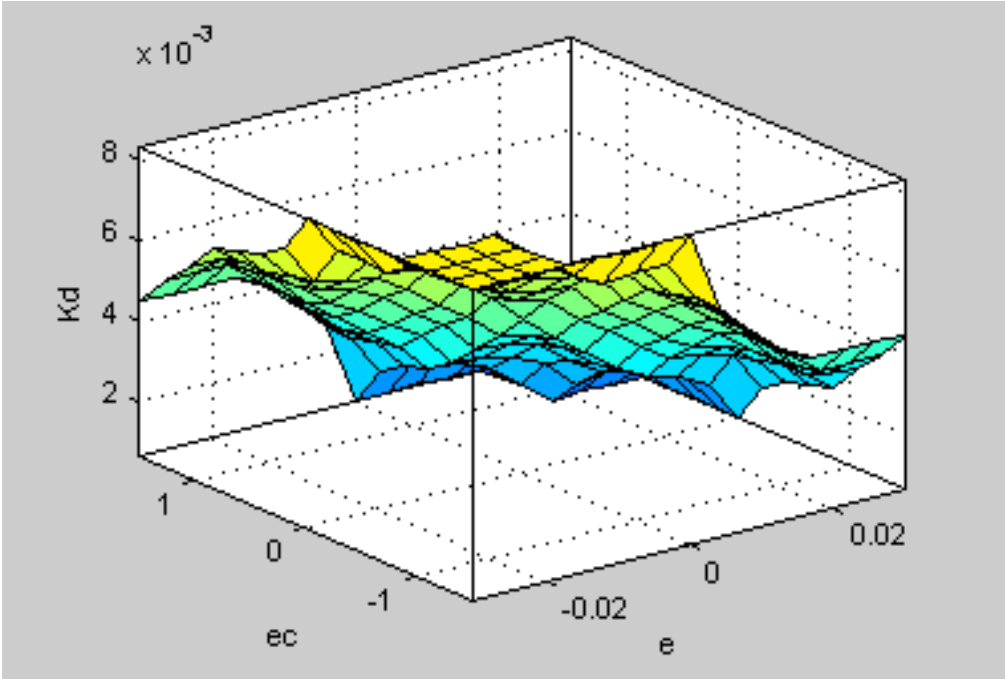


Figure 33 Rule surface viewer of $\boldsymbol{K_d}$

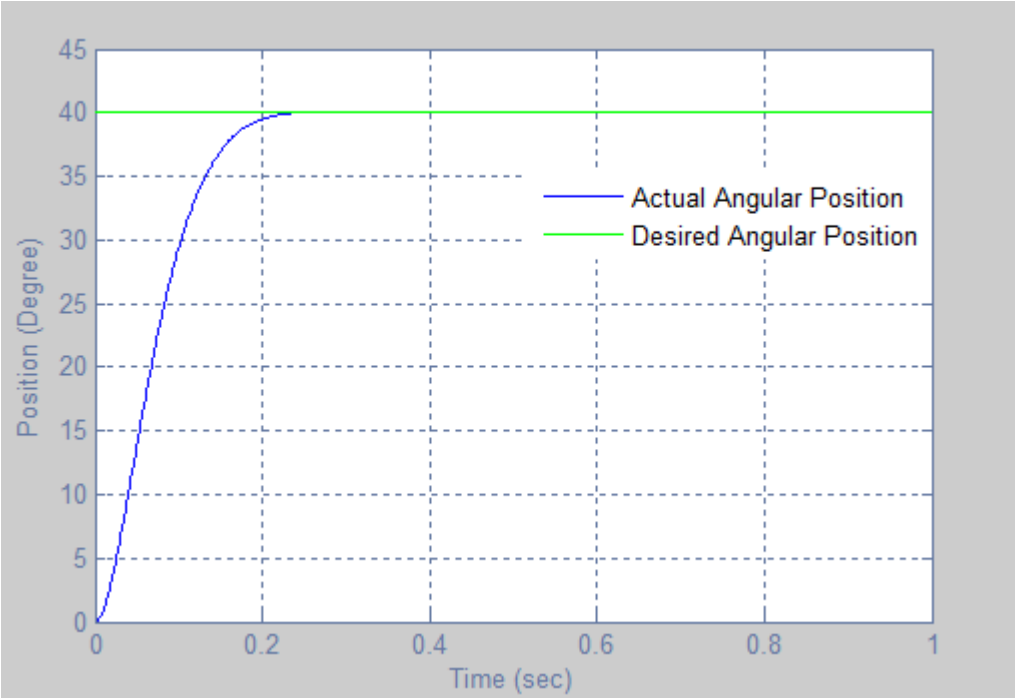


Figure 34 Position Control Response of fuzzy tuned PID controlled DC motor

Rise time (seconds): 0.14 & Overshoot (%): 0.296

Settling time (seconds): 0.189 & Accuracy Error (%): 0.03

K_p = 8.996 & K_i = 0.055 and K_d = 0.004

CHAPTER 6

ARTIFICIAL NEURAL NETWORKES

6.1 Introduction

An Artificial Neural Network (ANN) is a mathematical model that tries to simulate the structure and functionalities of biological neural networks [24]. It is powerful tools for modeling and solving complex dynamic systems, especially when the underlying data relationship is unknown. ANN can identify and learn correlated patterns between input data sets and corresponding target values. After training, ANN can be used to predict the outcome of new independent input data. ANN imitate the learning process of the human brain and can process problems involving non-linear and complex data even if the data are imprecise and noisy. Thus it is ideally suited for the modeling of agricultural data which are known to be complex and often non-linear. ANN has great capacity in predictive modeling i.e., all the characters describing the unknown situation can be presented to the trained ANN, and then prediction of agricultural systems is guaranteed.

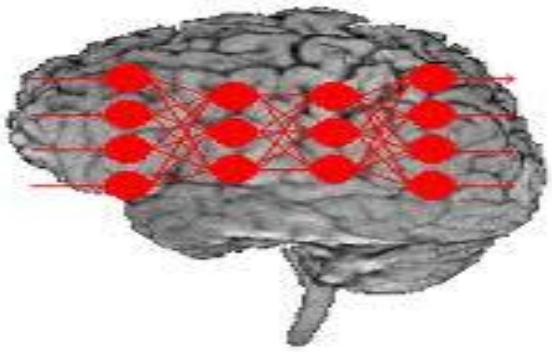


Figure 35 Human Brain

An ANN is a computational structure that is inspired by observed process in natural networks of biological neurons in the brain. It consists of simple computational units called neurons, which are highly interconnected. ANNs have become the focus of

much attention, largely because of their wide range of applicability and the ease with which they can treat complicated problems. ANNs are parallel computational models comprised of densely interconnected adaptive processing units. These networks are fine-grained parallel implementations of nonlinear static or dynamic systems. A very important feature of these networks is their adaptive nature, where "learning by example" replaces "programming" in solving problems. This feature makes such computational models very appealing in application domains where one has little or incomplete understanding of the problem to be solved but where training data is readily available. ANNs are now being increasingly recognized in the area of classification and prediction, where regression model and other related statistical techniques have traditionally been employed. The most widely used learning algorithm in an ANN is the Backpropagation algorithm. There are various types of ANNs like Multilayered Perceptron, Radial Basis Function and Kohonen networks. These networks are "neural" in the sense that they may have been inspired by neuroscience but not necessarily because they are faithful models of biological neural or cognitive phenomena. In fact majority of the network are more closely related to traditional mathematical and/or statistical models such as non-parametric pattern classifiers, clustering algorithms, nonlinear filters, and statistical regression models than they are to neurobiology models [20].

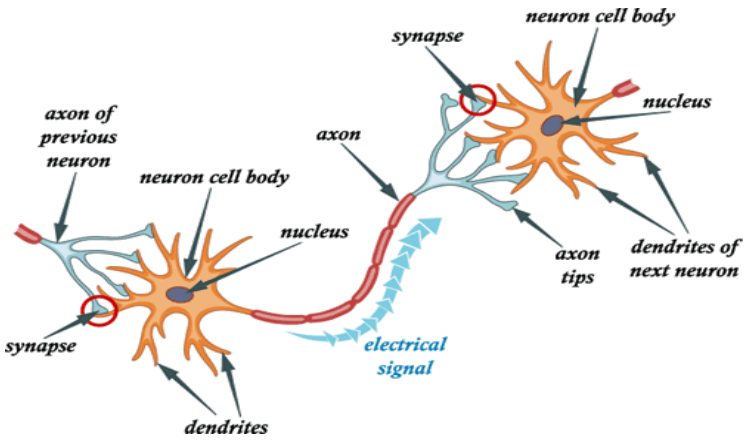


Figure 36 Biological neuron model

6.2 Neuron Model

6.2.1 Simple Neuron

The fundamental building block for neural networks is the single input neuron as shown in figure 37

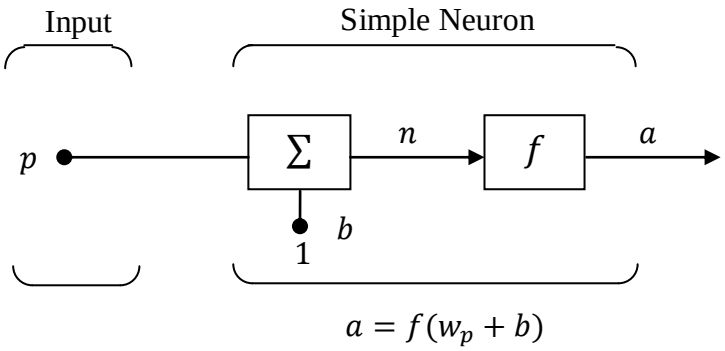


Figure 37 Artificial neuron structure

There are three distinct functional operations that take place in this example neuron. First, the scalar input p is multiplied by the scalar weight w to form the product wp, again a scalar. Second, the weighted input wp is added to the scalar bias b to form the net input n. (In this case, you can view the bias as shifting the function f to the left by an amount b. The bias is much like a weight, except that it has a constant input of 1.) Finally, the net input is passed through the transfer function f, which produces the scalar output a. The names given to these three processes are: the weight function, the net input function and the transfer function [41]

$$a = f(w_p + b) \tag{37}$$

6.2.2 Neuron with Vector Input

The simple neuron can be extended to handle inputs that are vectors. A neuron with a single R-element input vector is shown below. Here the individual input elements

$$P_1, P_2, \ldots, P_R$$

are multiplied by weights

$$w_{1,1}, w_{1,2} \ldots, w_{1,R}$$

and the weighted values are fed to the summing junction. Their sum is simply w_p , the dot product of the (single row) matrix w and the vector P. (There are other

weight functions, in addition to the dot product, such as the distance between the row of the weight matrix and the input vector, as in Radial Basis Networks.)

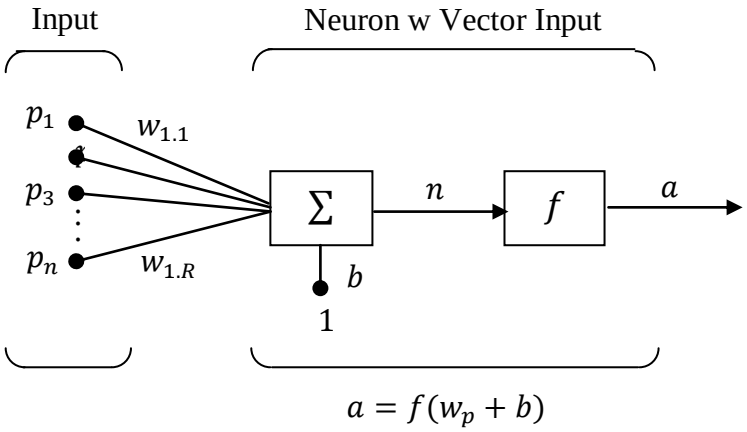


Figure 38 Multi input neuron

Where

R = number of elements in input vector

The neuron has a bias b, which is summed with the weighted inputs to form the net input n. (In addition to the summation, other net input functions can be used, such as the multiplication that is used in Radial Basis Networks.) The net input n is the argument of the transfer function f [41].

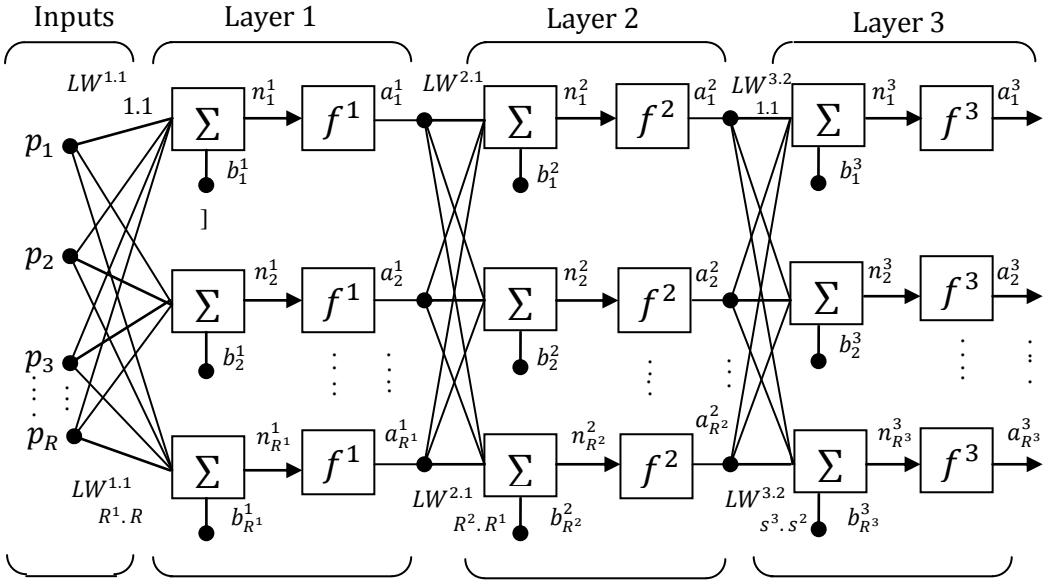
$$n = w_{1,1}p_1 + w_{1,2}p_2 + w_{1,R}p_R + b \tag{38}$$

This expression can be written as

$$n = w * p + b \tag{39}$$

6.2.3 Multiple Layers of Neurons

A network can have several layers. Each layer has a weight matrix w, a bias vector b, and an output vector a. To distinguish between the weight matrices, output vectors, etc., for each of these layers in the figures, the number of the layer is appended as a superscript to the variable of interest. You can see the use of this layer notation in the three-layer network shown in figure 39



$$a^1 = f^1(LW^{1.1}p + b^1),\ a^2 = f^2(LW^{2.1}a^1 + b^2),\ a^3 = f^3(LW^{3.2}a^2 + b^3)$$

Figure 39 Three layer neural network

$$a^3 = f^3(w^3 f^2(w^2 f^1(w^1 p + b^1) + b^2) + b^3) \tag{40}$$

The network shown above has R^1 inputs, S^1 neurons in the first layer, S^2 neurons in the second layer, etc. It is common for different layers to have different numbers of neurons. A constant input 1 is fed to the bias for each neuron.

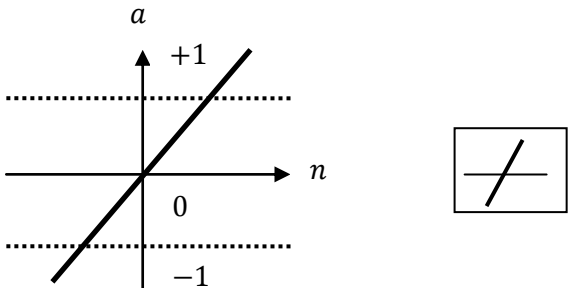
Note that the outputs of each intermediate layer are the inputs to the following layer. Thus layer 2 can be analyzed as a one-layer network with S^1 inputs, S^2 neurons, and an $S^2 \times S^1$ weight matrix w^2. The input to layer 2 is a^1; the output is a^2. Now that all the vectors and matrices of layer 2 have been identified, it can be treated as a single-layer network on its own. This approach can be taken with any layer of the network.

The layers of a multilayer network play different roles. A layer that produces the network output is called an output layer. All other layers are called hidden layers. The three-layer network shown earlier has one output layer (layer 3) and two hidden layers (layer 1 and layer 2). Some authors refer to the inputs as a fourth layer.

The architecture of a multilayer network with a single input vector can be specified with the notation $R - S^1 - S^2 -...- S^m$, where the number of elements of the input vector and the number of neurons in each layer are specified [41].

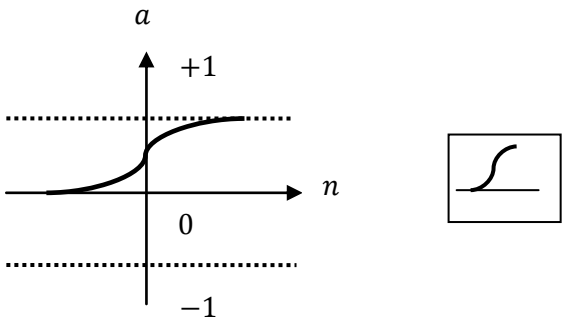
6.3 Transfer Functions

Many transfer functions can be used in the Neural Network, the most common transfer functions used with ANNs are shown in figures 40, 41, 42 and 43



$a = purelin(n)$

Figure 40 The Linear transfer function

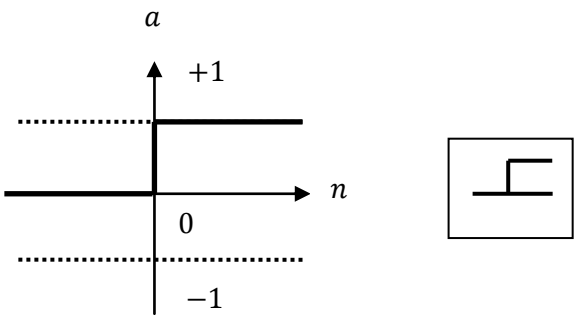


$a = logsig(n)$

Figure 41 The Log-Sigmoid transfer function

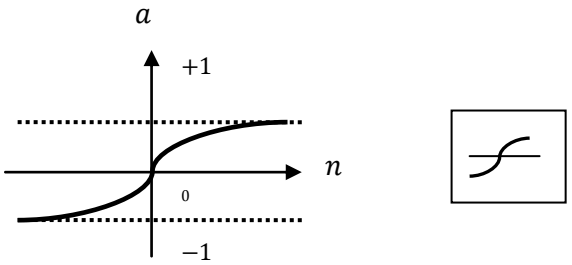
The output (a) of a log-sigmoid transfer function is calculated according to the expression:

$$a = \frac{1}{1 + e^{-n}} \tag{41}$$



$a = hardlim(n)$

Figure 42 Hard-Limit transfer function



$a = tansig(n)$

Figure 43 Tan-Sigmoid Transfer Function

The output (a) of a hyperbolic tangent sigmoid transfer function is calculated according to the expression [41]:

$$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}} \tag{42}$$

6.4 Backpropagation With Feedforward Network

Back-propagation has reawakened the scientific and engineering community to the modeling and processing of many quantitative phenomena using neural-networks. The learning algorithm is applied to the multilayer feed forward networks consisting of processing elements with continuous differentiable activation functions. Such networks associated with the back-propagation learning algorithm are also called back-propagation networks. Given a training set of input-output pairs $\{(x^{(k)}, d^{(k)})\}, k = 1,2, \ldots\ldots, p$, the algorithm provides a procedure for changing the weights in the back-propagation network to classify the given input patterns correctly. The basis for this weight update algorithm is simply the gradient descent method as used for simple perceptrons with differentiable units.

For a given input-output pair $(x^{(k)}, d^{(k)})$, the back-propagation algorithm performs two phases of data flow. First, the input pattern $x^{(k)}$ is propagated from the input layer to the output layer and, as a result of this forward flow of data, it produces an actual output $y^{(k)}$. Then the error signals resulting from the difference between $d^{(k)}$ and $y^{(k)}$ are backpropagated from the output layer to the previous layers for them to update their weights. Consider a three-layer network as shown in Figure 44 to illustrate the details of the backpropagation learning algorithm. Here we have m processing elements in the input layer, j processing elements in the hidden layer, and

n processing elements in the output layer. First, let us consider an input-output training pair $(x\ , d)$, where the superscript k is omitted for notation simplification.

Give an input x, a processing element q in the hidden layer receives a net input of

$$net_q = \sum_{j=1}^{m} v_{qj}\, x_j \tag{43}$$

And produces an output of

$$z_q = a(net_q) = a\left(\sum_{j=1}^{m} v_{qj}\, x_j\right) \tag{44}$$

The net input for a processing element i in the output layer is then

$$net_i = \sum_{q=1}^{l} w_{iq}\, z_q = \sum_{q=1}^{l} w_{iq}\, a\left(\sum_{j=1}^{m} v_{qj}\, x_j\right) \tag{45}$$

And it produces an output of

$$y_i = a(net_i) = a\left(\sum_{q=1}^{l} w_{iq}\, z_q\right) = a\left(\sum_{q=1}^{l} w_{iq}\, a\left(\sum_{j=1}^{m} v_{qj}\, x_j\right)\right) \tag{46}$$

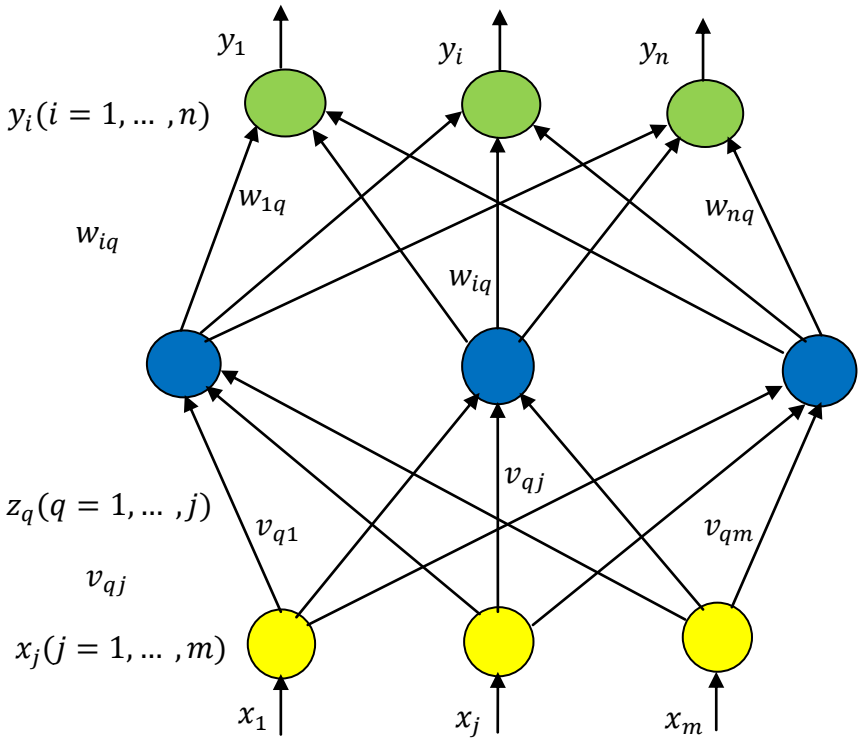


Figure 44 Three layer back-propagation network

The above equations indicate the forward propagation of input signals through the layers of neurons till output layer.

The cost function is defined as:

$$E(w) = \frac{1}{2}\sum_{i=1}^{n}(d_i - y_i)^2 = \frac{1}{2}\sum_{i=1}^{n}[d_i - a(net_i)]^2 \tag{47}$$

$$= \frac{1}{2}\sum_{i=1}^{n}\left[d_i - a\left(\sum_{q=1}^{l} w_{iq}\, z_q\right)\right]^2 \tag{48}$$

Error signal is propagated back up to input layer and according to the gradient-descent method, the weights in the hidden-to-output connections are updated by

$$\Delta w_{iq} = -\eta \frac{\partial E}{\partial w_{iq}} \tag{49}$$

Using Eqs. (45)-(48) and the chain rule for $\partial E/\partial w_{iq}$, we have

$$\Delta w_{iq} = -\eta \left[\frac{\partial E}{\partial y_i}\right]\left[\frac{\partial y_i}{\partial net_i}\right]\left[\frac{\partial net_i}{\partial w_{iq}}\right] = \eta[d_i - y_i][a'(net_i)]\left[Z_q\right] \triangleq \eta \delta_{oi} z_q \quad (50)$$

Where δ_{oi} is the error signal and its double subscript indicates the *i*th node in the output layer. The error signal is defined

$$\delta_{oi} \triangleq -\frac{\partial E}{\partial net_i} = -\left[\frac{\partial E}{\partial y_i}\right]\left[\frac{\partial y_i}{\partial net_i}\right] = [d_i - y_i][a'(net_i)] \quad (51)$$

Where net_i is the net input to processing element i of the output layer and $a'(net_i) = \partial a(net_i)/\partial net_i$.

For the weight update on the input-to-hidden connections, the weight update on the link weight connecting processing element j in the input layer to processing element q in the hidden layer

$$\Delta v_{qj} = -\eta \left[\frac{\partial E}{\partial v_{qj}}\right] = -\eta \left[\frac{\partial E}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qi}}\right] = -\eta \left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right]\left[\frac{\partial net_q}{\partial v_{qi}}\right] \quad (52)$$

From Eq. (47) & (48), it is clear that each error term $[d_i - y_i], i = 1,2, \ldots \ldots, n,$ is a function of z_p. Evaluating the chain rule, we have

$$\Delta v_{qj} = \eta \sum_{i=1}^{n} \left[(d_i - y_i) a'(net_i) w_{iq}\right] a'\left(net_q\right) x_j \quad (53)$$

Using Eq. (51), we can rewrite Eq. (53) as

$$\Delta v_{qj} = \eta \sum_{i=1}^{n} \left[d_{oi} w_{iq}\right] a'\left(net_q\right) x_j = \eta \delta_{hq}\, x_j \quad (54)$$

Where δ_{hq} is the error signal of processing element q in the hidden layer and is defined as

$$\delta_{hq} \triangleq -\frac{\partial E}{\partial net_q} = -\left[\frac{\partial E}{\partial z_q}\right]\left[\frac{\partial z_q}{\partial net_q}\right] a'\left(net_q\right) \sum_{i=1}^{n} \delta_{oi} w_{iq} \quad (55)$$

Where net_q is the net input to the hidden processing element q [Eq. (43)]. The error signal of a processing element in a hidden layer is different from the error signal of a

processing element in the output layer, as in Eqs. (51) & (55). Because of this difference, the above weight update procedure is called the generalized delta learning rule [36].

6.4.1 Back-propagation Learning Algorithm Steps

Consider a network with *Q* feed forward layers, *q= 1,2,.....,Q,* and let q_{net_i} and q_{y_i} Denote the net input and output of the *i*th in the *q*th layer, respectively. The network has *m* input nodes and *n* output nodes. Let $q_{w_{ij}}$ denote the connection weight from $q-1_{y_j}$to q_{y_i}.

Input**:** A set of training pairs $\{(x^{(k)}, d^{(k)})|k = 1,2,\ldots.,p\}$, where the input vectors are augmented with the last elements as -1, that is, $x_{m+1}^{(k)} = -1$.

Step 0: Choose $\eta > 0$ and E_{max} (maximum tolerable error). Initialize the weights to small random values. Set $E = 0$ and $k = 1$.

Step 1 (Training loop): Apply the k^{th} input pattern to the input layer ($q = 1$):

$$q_{y_i} = 1_{y_i} = x_i^{(k)} \qquad \text{for all } i. \tag{56}$$

Step 2: Propagate the signal forward through the network using

$$q_{y_i} = a(q_{net_i}) = a\left(\sum_j q_{w_{ij}}\, q_{y_j}^{-1}\right) \tag{57}$$

Step 3: Compute the error value and error signals Q_{δ_i} for the output layer:

$$E = \frac{1}{2}\sum_{i=1}^{n}\left(d_i^{(k)} - Q_{y_i}\right)^2 + E \tag{58}$$

$$Q_{\delta_i} = \left(d_i^{(k)} - Q_{y_i}\right) a'(Q_{net_i}) \tag{59}$$

Step 4: Propagate the errors backward to update the weights and compute the error signals $q^{-1}\delta_i$ for the preceding layers:

$$\Delta q_{w_{ij}} = \eta q_{\delta_i}^{q-1} y_j \qquad and\ \ q_{w_{ij}}^{new} = q_{w_{ij}}^{old} + \Delta q_{w_{ij}} \tag{60}$$

$$q^{-1}\delta_i = a'(q^{-1}net_i)\sum_j q_{w\,ij}\, q\delta_j \qquad for\ q = Q, Q-1, \ldots\ldots, 2 \qquad (61)$$

Step 5 : Check whether the whole set of training data has been cycled once. If $k < p$, then $k = k + 1$ and go to step 1; otherwise, go to step 6.

Step 6: Check whether the current total error is acceptable: If $E < E_{max}$, then terminate the training process and output the final weights; otherwise, $E = 0,\ k = 1$, and initiate the new training epoch by going to step 1.

End BP [36].

6.5 Reconstructing of Fuzzy PID Controller Based on feed-forward Neural Network using Back-propagation algorithm

The combination of widely-used PID controller with Fuzzy system generates fuzzy PID controller which possesses excellent control quality. However, the fuzzy PID controller has some problems of computation complexity and the performance. To solve these problems, the research expounds the process of training Back-propagation neural network through its universal function approximating ability and PID controller's input and output couple. This research reveal that the fuzzy PID controller can be effectively replaced by the trained Back-propagation neural network. Therefore, the method can simplify the computation complexity and improving the performance of fuzzy PID controller [37].

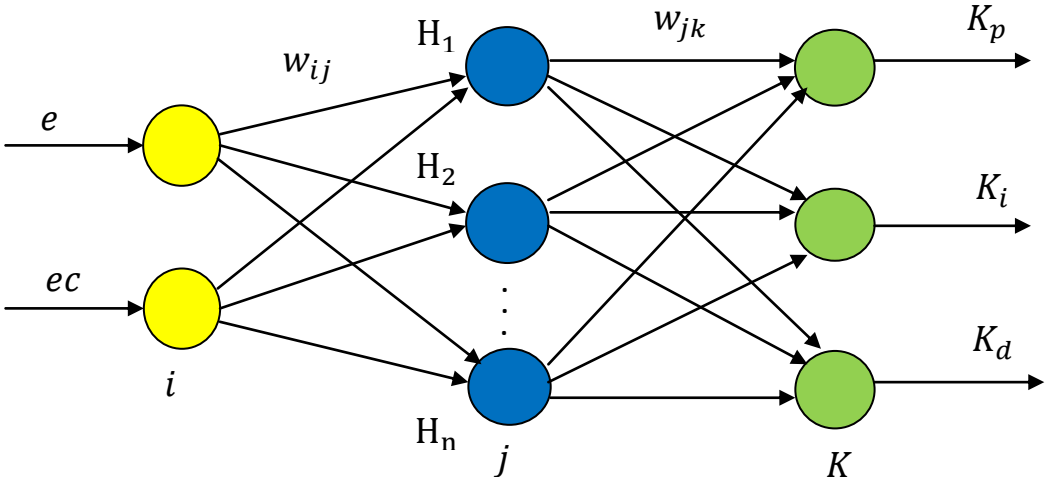


Figure 45 Reconstructing of Fuzzy PID Controller based on BP Neural Network

6.6 Simulink Model for Position Control of DC motor using Neural Network

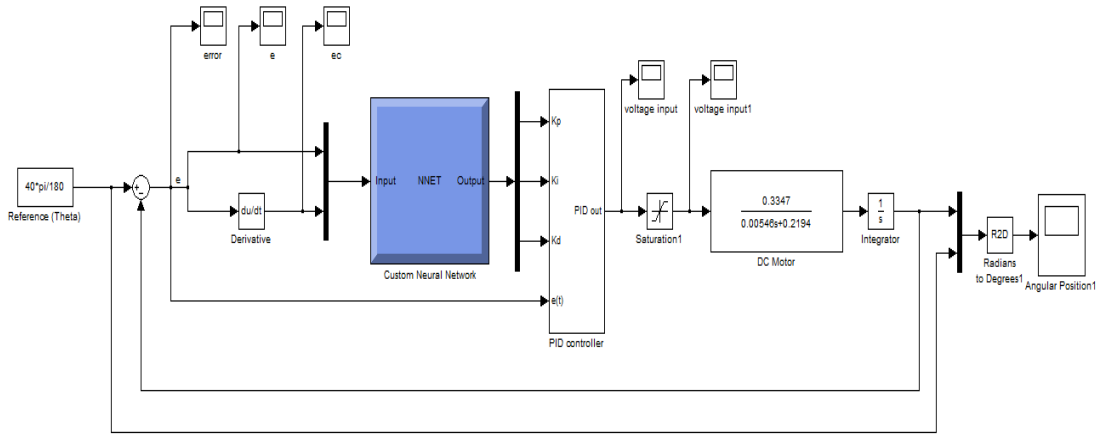


Figure 46 Simulink Model for Position Control of DC motor using Neural Network

6.7 Artificial Neural Network Model

The feed forward neural network configuration as shown in figure 47 consists of three layers namely input layer, one hidden layer and one output layer. The input layer has two inputs as theta error(e) and the rate of change in theta error (ec) of DC motor and three output neurons as $K_p, K_i, \& K_d$, which are the input gains of PID controller for PID automatic tuning. The hidden layer is considered to have twenty-five neurons.

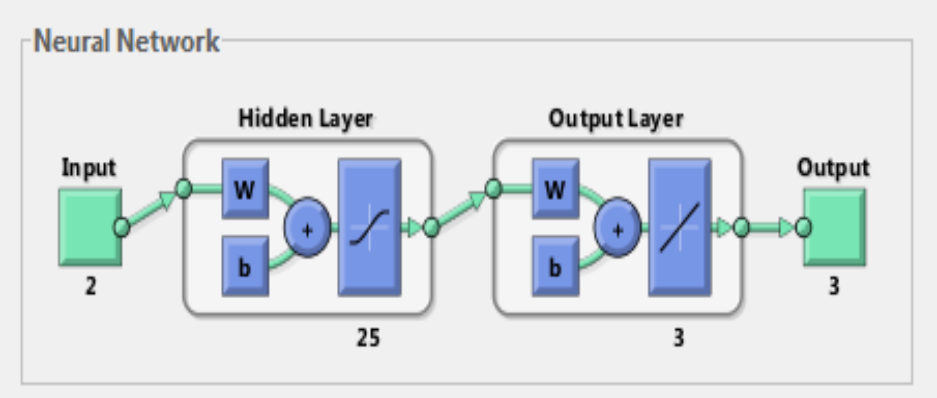


Figure 47 ANN feed-forward model

We can use MATLAB command to build and train our neural networks, we use this command "gensim(net,st)" to generate neural network SIMULINK diagram. The function used the gensim function; this function can be used after successful training of the neural network, the block generated then can be used to build the total system, shows figure 48 and shows the full code of the neural network in Appendix A.

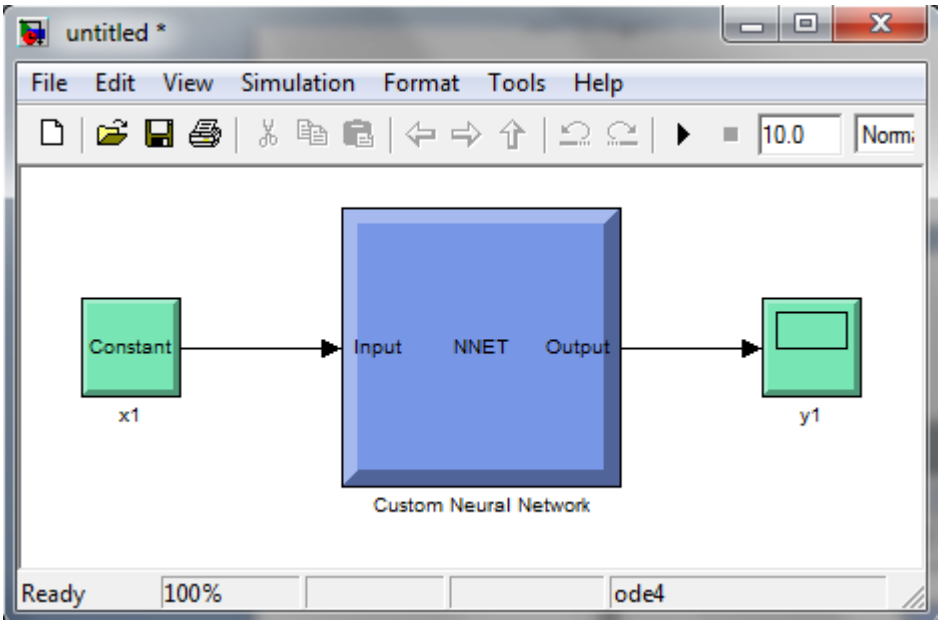


Figure 48 NN Model Block of Fuzzy self- tuning PID controller

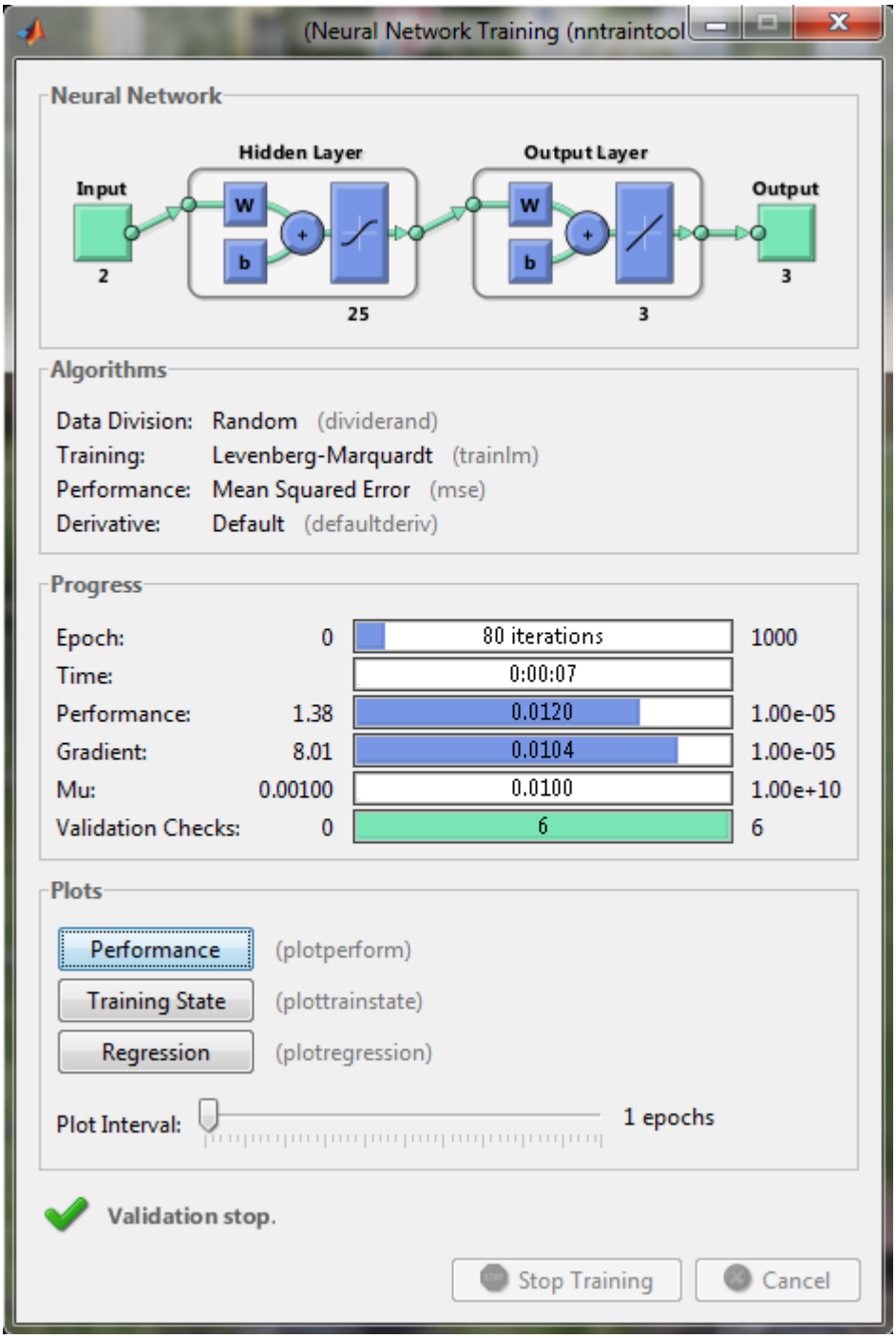


Figure 49 Training tool window of the NN controller model

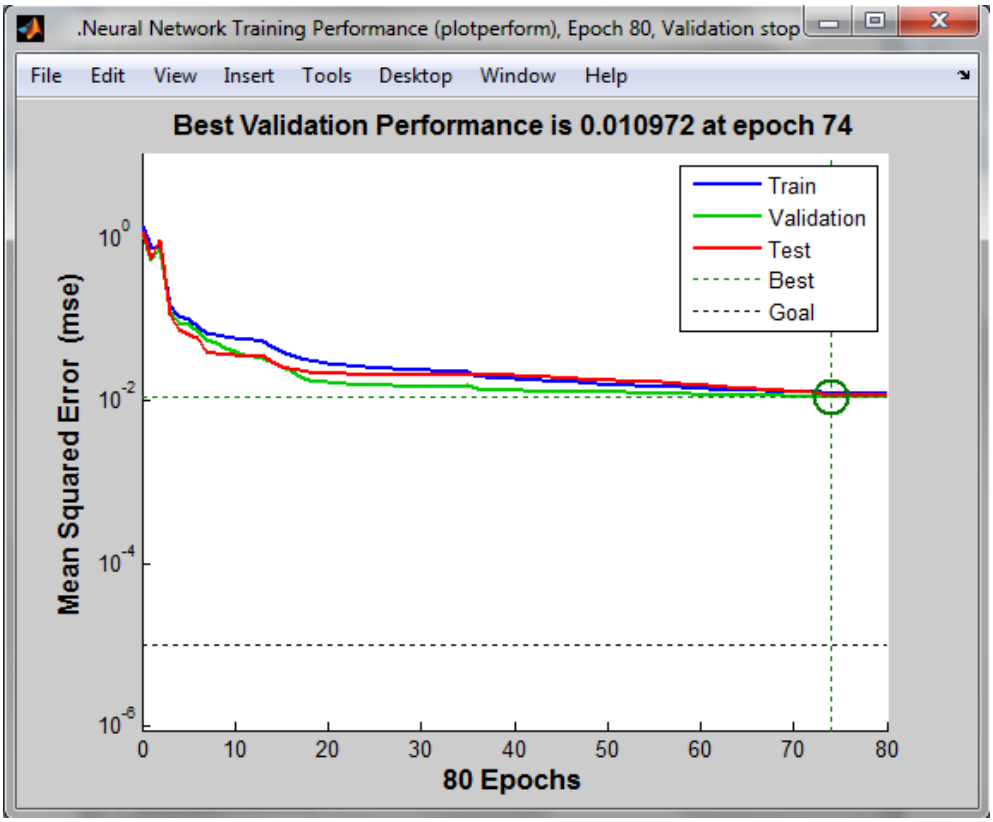


Figure 50 Performance plot of the NN controller

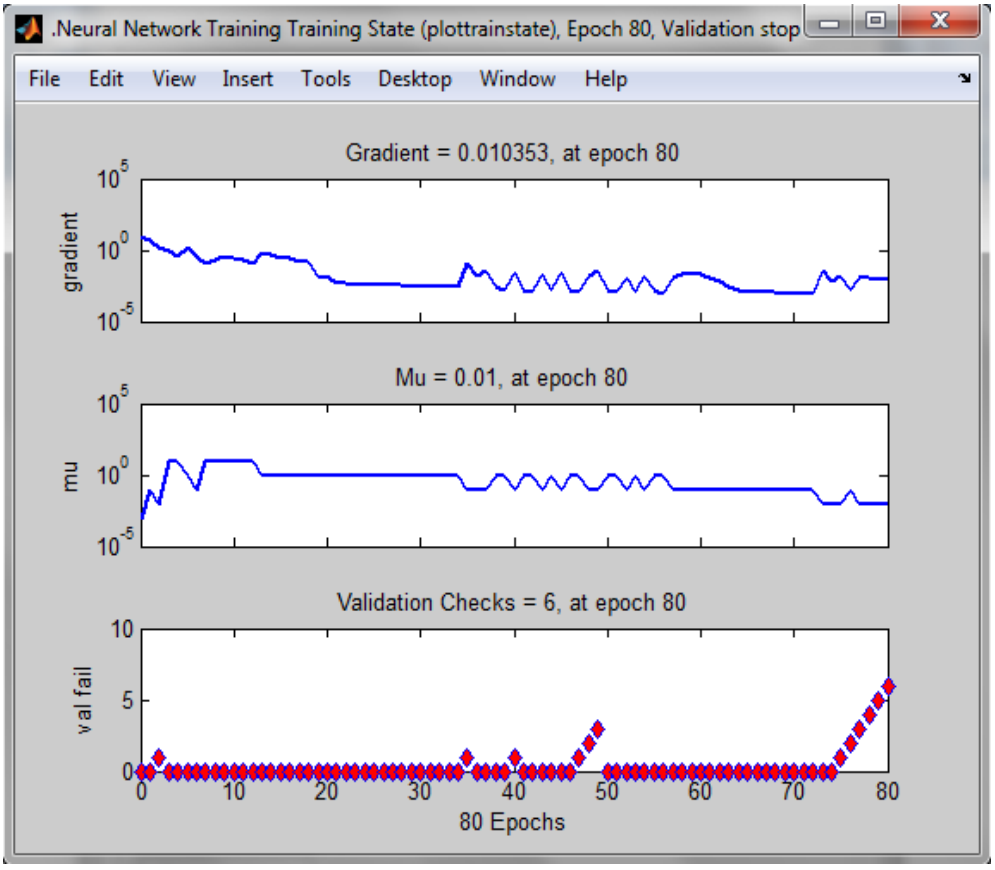


Figure 51 Training State

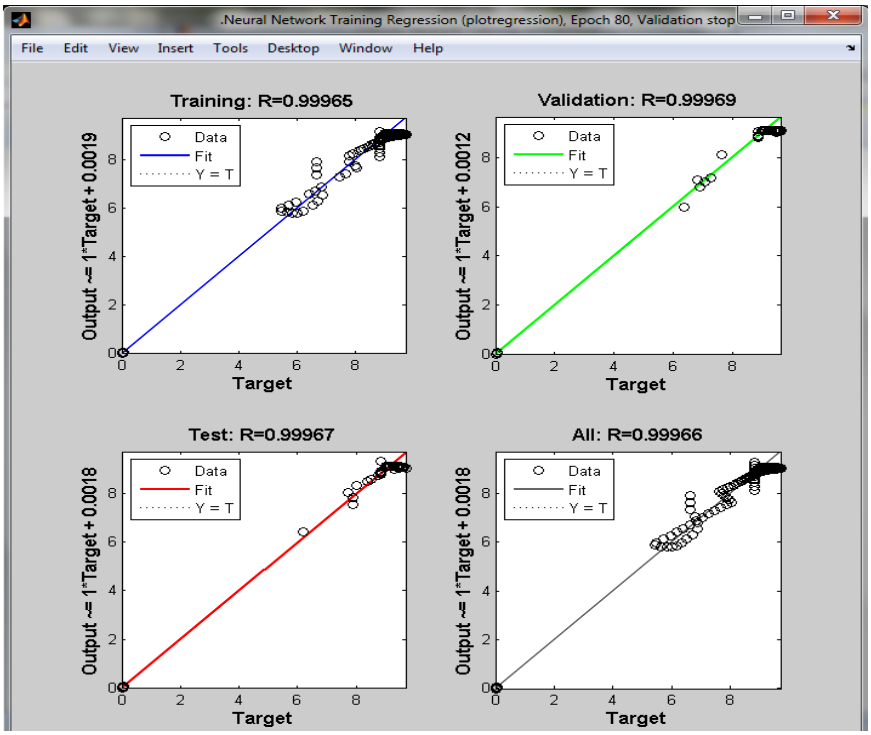


Figure 52 Regression plot

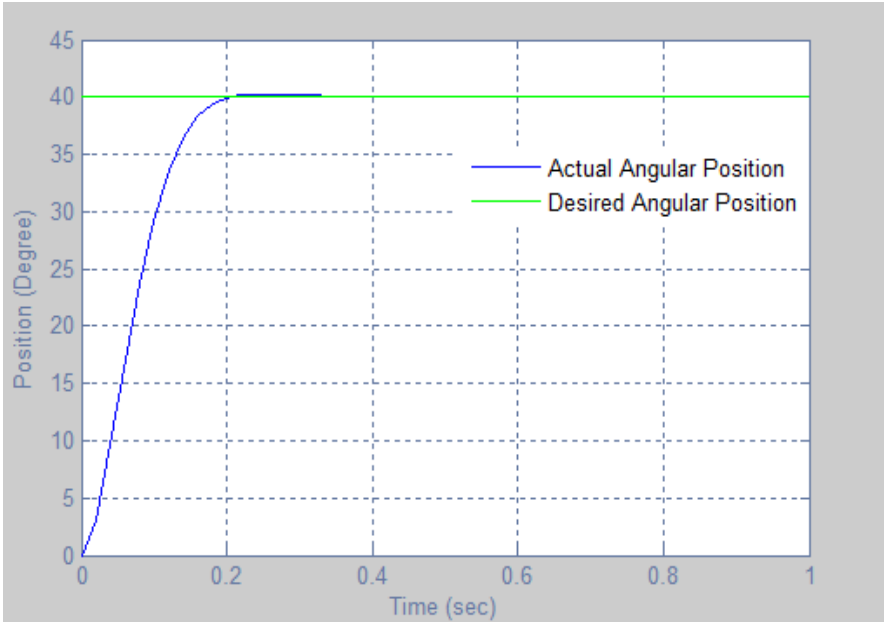


Figure 53 Position Control Response of Neural Network tuned PID controlled DC motor

Rise time (sec): 0.136 & Overshoot (%): 0.68

Settling time (sec): 0.176 & Accuracy Error (%): 0.03

K_p = 9.084 & K_i = 0.056 and K_d = 0.004

CHAPTER 7

REAL TIME CONTROL IMPLEMENTATION

The real time controller is implemented in Matlab/Simulink by using Real-Time Windows Target. The Quanser SRV02 rotary servo plant Model (006 S), pictured in Figure 54, consist of a DC motor that is encased in a solid aluminum frame and equipped with a planetary gearbox. That is, the motor has its own internal gearbox that drives external gears. The basic SRV02 units comes with an potentiometer sensor that can be used to measure angular position of the load gear. The SRV02 device can also be fitted with an encoder to obtain a digital position measurement.

Data Acquisition card that is used for this purpose was Sensoray 626.

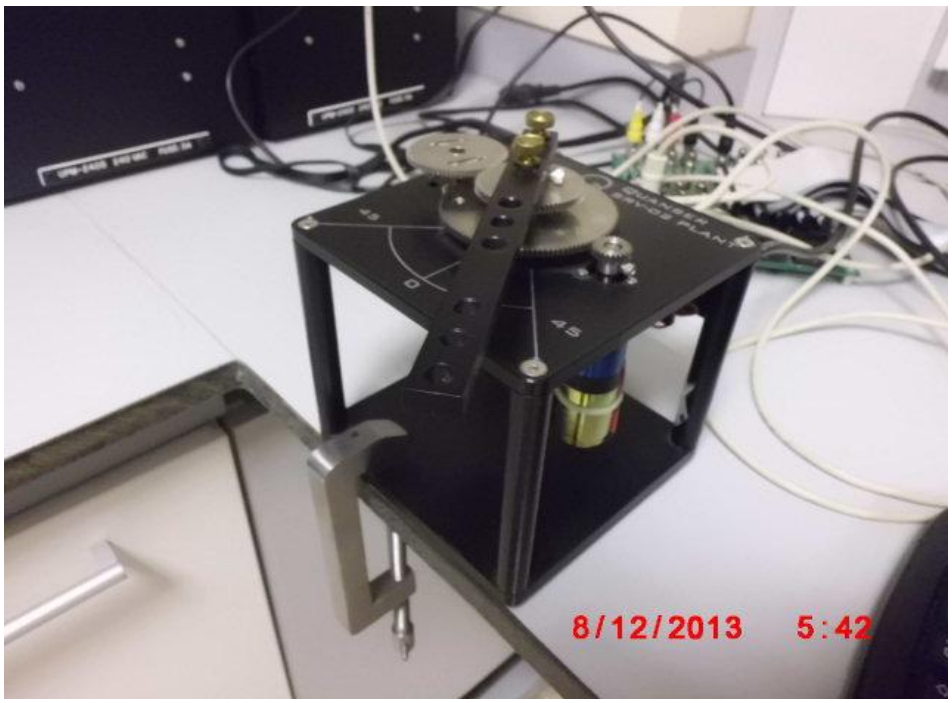


Figure 54 Top view of Servo Plant with higher gear configration

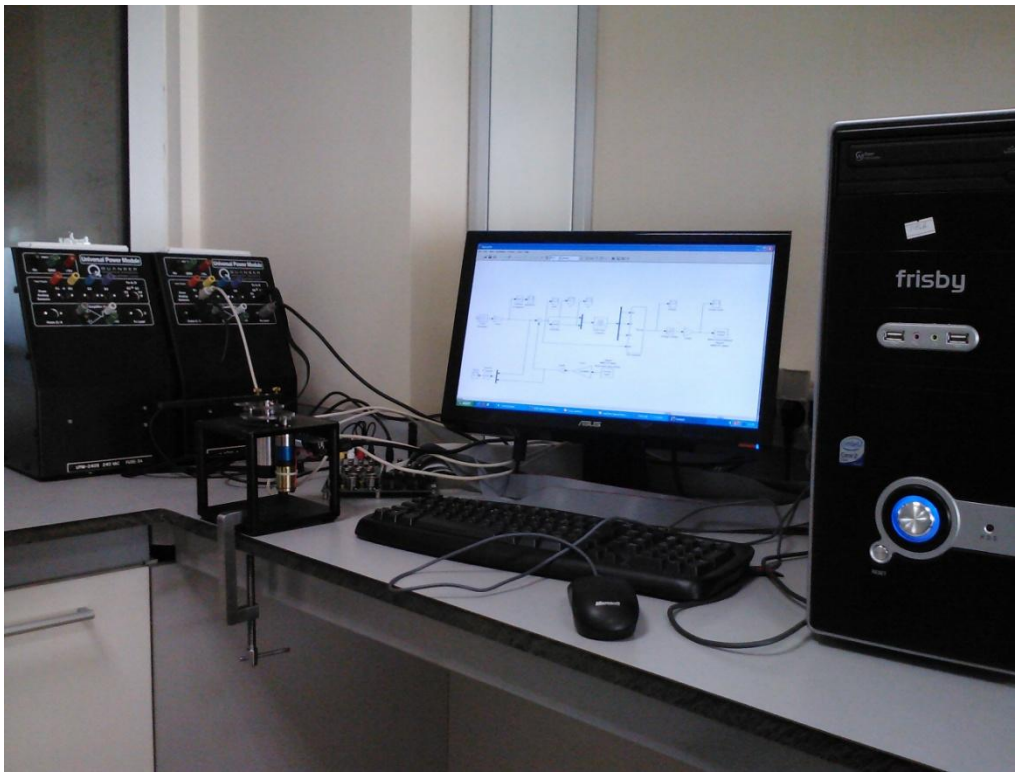


Figure 55 Experimental setup

Figure 56 Top view of Servo Plant with higher gear configuration

7.1 Implementation of PID Controller in Real Time

There is small gab between simulink and real time, for this reason. We need to redesign PID controller in real time, using Ziegler-Nichols tuning rule.

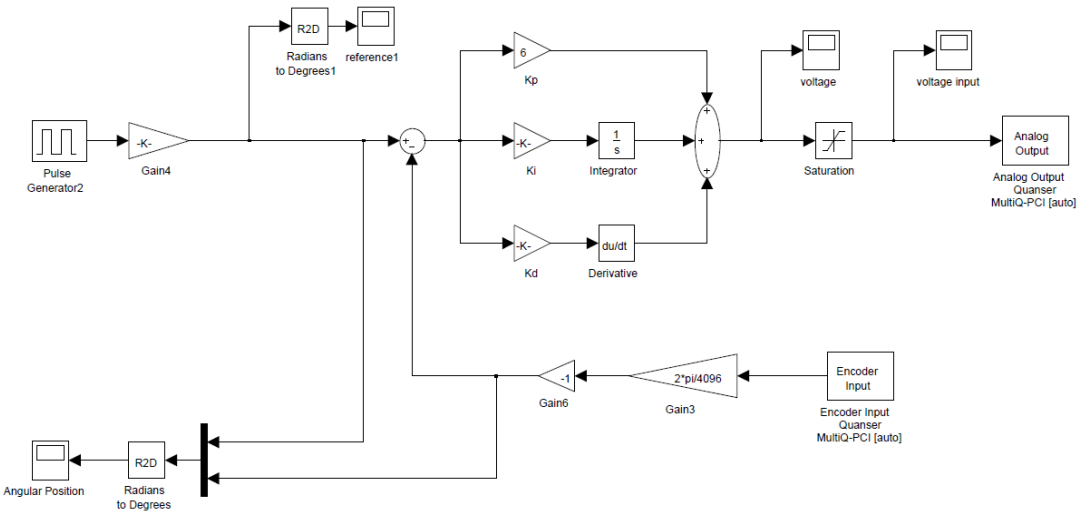


Figure 57 Simulink diagram of DC motor in real time

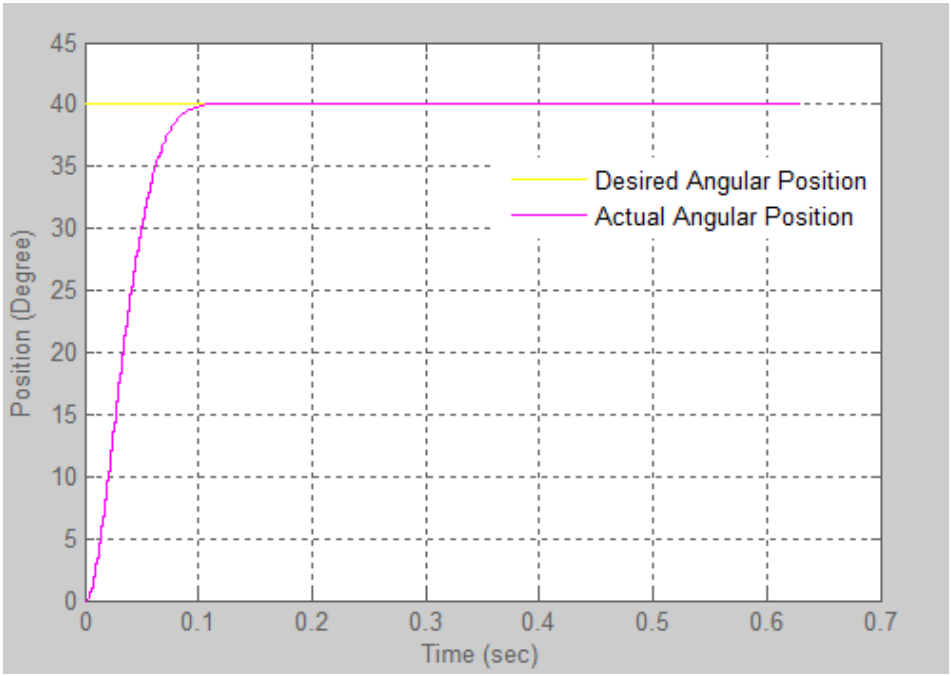


Figure 58 Position Control Response using Ziegler-Nichols tuning rule in real time

Rise time (sec): 0.066 & Accuracy Error (%): 0.025

Settling time (sec): 0.086 & Overshoot (%): 0

K_p = 6 & K_i = 0.1 and K_d = 0.101

7.2 Implementation of Fuzzy self-tuning PID Controller in Real Time

There is small gab between simulink and real time, for this reason. We need to redesign PID controller in real time based on Fuzzy self-tuning.

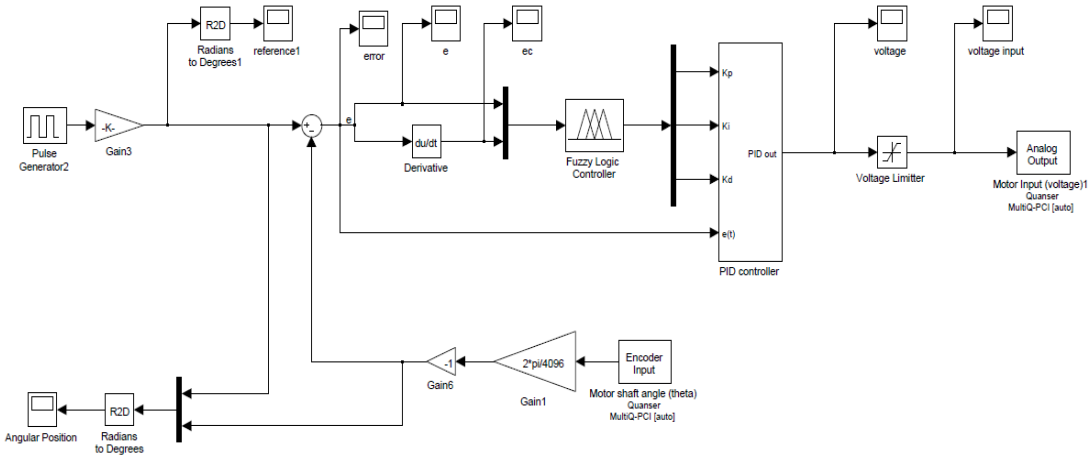


Figure 59 Simulink Model for Position Control of DC motor using self tuned fuzzy PID controller in real time

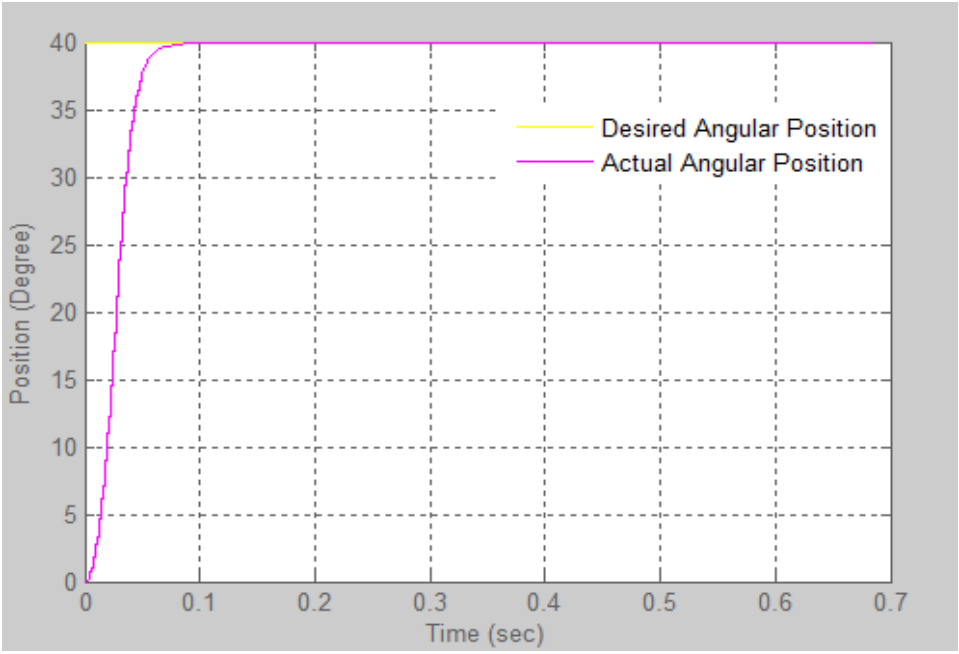


Figure 60 Position Control Response using Fuzzy self-tuning PID Controller in real time

Rise time (sec): 0.045 & Settling time (sec): 0.06 & Accuracy Error (%): 0.025

Overshoot (%): 0 & K_p = 34.74, K_i = 0.25 and K_d = 0.446

7.3 Implementation of Neural Network tuning PID Controller in Real Time

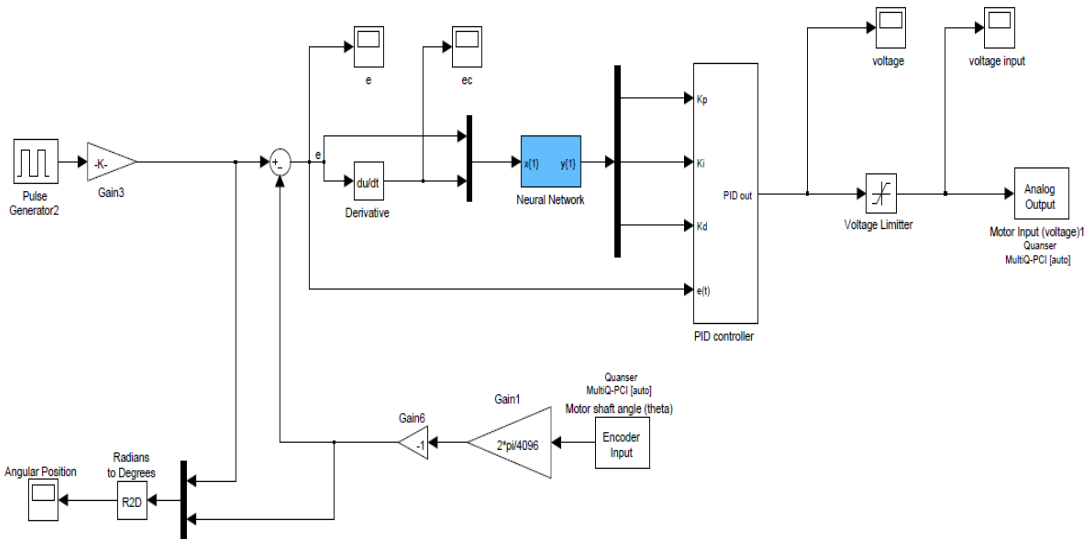


Figure 61 Simulink Model for Position Control of DC motor using Neural Network in real time

The real time implementation of Neural Network is not available in our laboratory, because Sensoray 626 Data Acquisition Card is not compatible to latest MATLAB versions such as (R2011a-R2012a), I need to work with these latest Matlab versions because it's support the block of neural network Matlab Function in real time.

The old Matlab versions such as (R2008b – R2009a) is compatible to sensoray 626 data acquisition card, but unfortunately doesn't support the block of neural network Matlab Function in real time. For this reason I need to work with latest Matlab version such as (R2011a-R2012a) to implement the Neural Network in real time.

In this case I compelled to apply neural network in Simulink/Matlab environment, show Artificial Neural Network simulation in chapter 6.

Figure 62 Sensoray 626 Data Acquisition Card

CHAPTER 8

RESULTS & CONCLUSIONS AND FUTURE WORK

8.1 A Comparison between Soft Computing Methods and Classical Methods for tuning PID Controller in Simulink

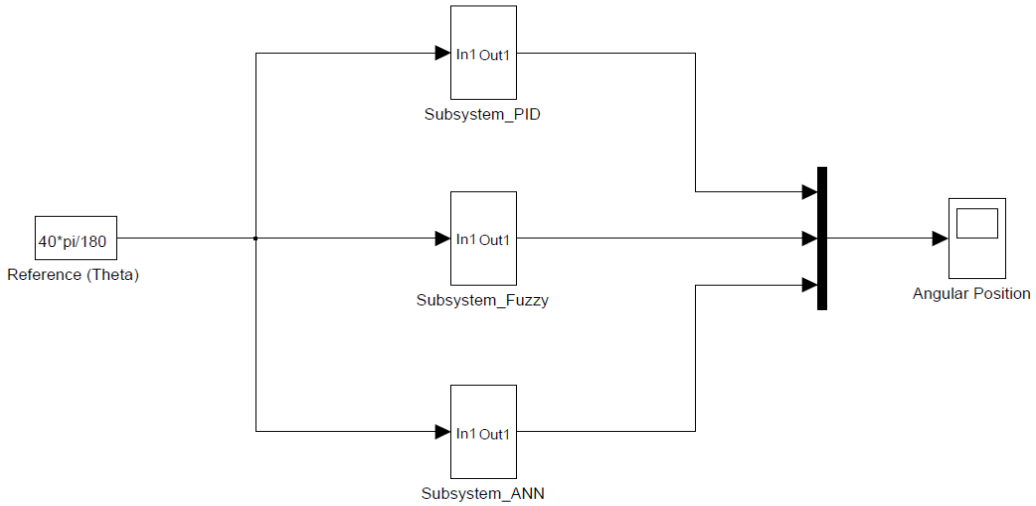


Figure 63 Subsystem Simulink Model for Position Control of DC motor using Fuzzy Logic, Neural Network and Ziegler-Nichols for tuning PID Controller

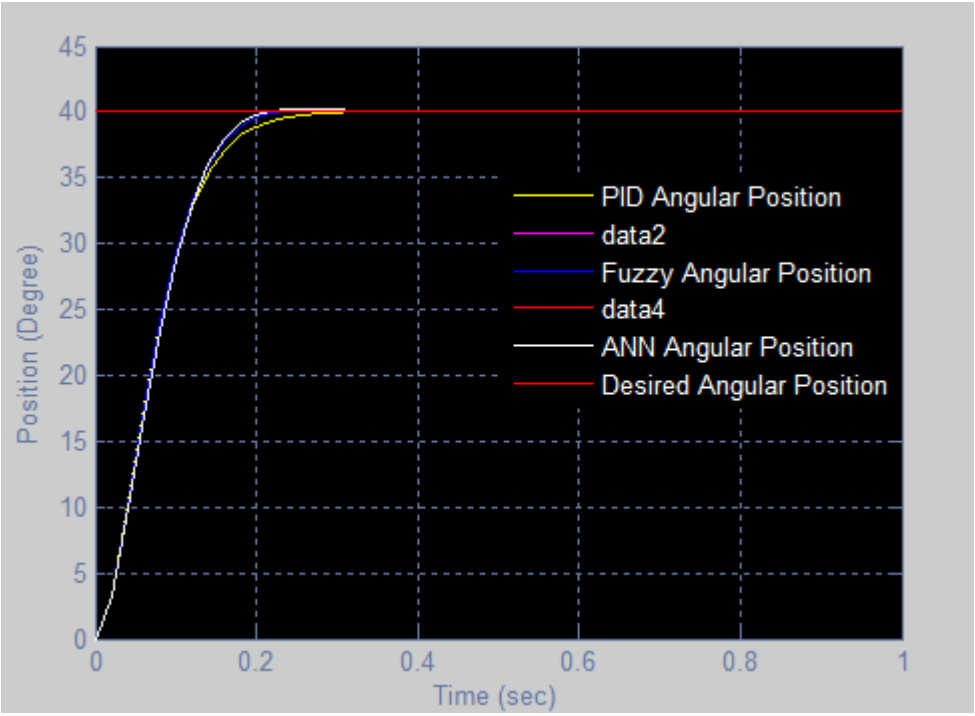


Figure 64 Position Control Response of DC motor using Fuzzy Logic, Neural Network and Ziegler-Nichols for tuning PID controller

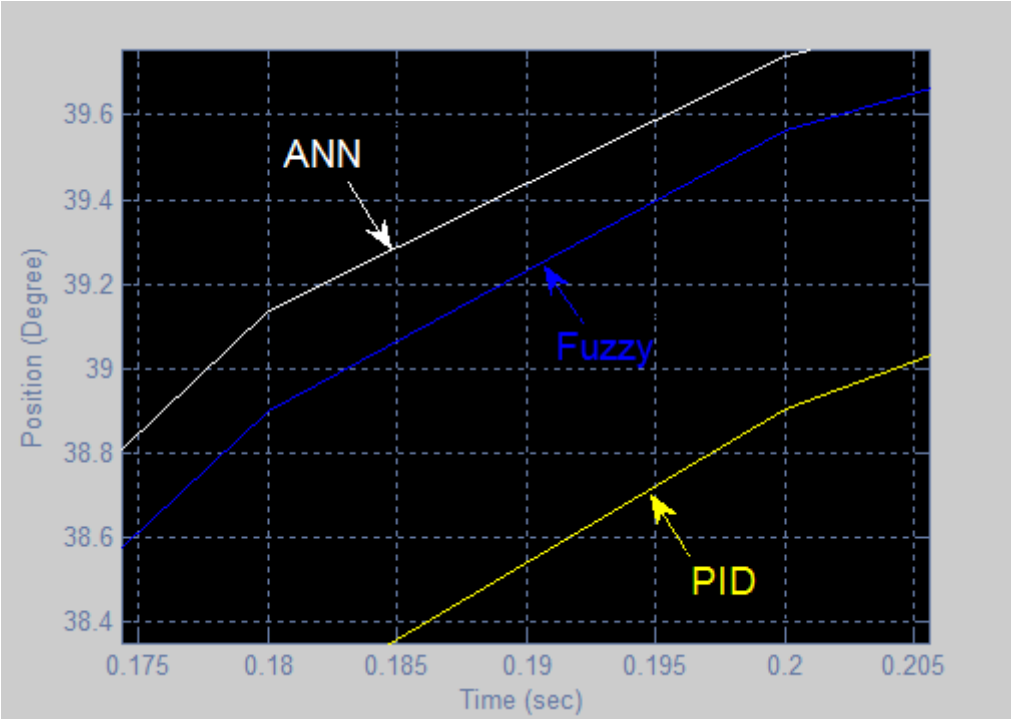


Figure 65 Clarification of Position Control Respnse of DC motor using Fuzzu Logic, Neural Network and Ziegler-Nichols for tuning PID controller

Table 9 Comparison between Soft Computing Methods and Classical Methods for tuning PID Controller in Simulink

Controller	**Angular Position (deg)**	**Rising Time (sec)**	**Settling Time (sec)**	**Overshoot (%)**	**Accuracy Error (%)**
Ziegler-Nichols based PID Controller	40	0.147	0.211	0.08	0.077
Fuzzy Logic based PID Controller	40	0.14	0.189	0.29	0.03
ANN based PID Controller	40	0.136	0.176	0.68	0.03

8.2 A Comparison between Soft Computing Methods and Classical Methods for tuning PID Controller in Real Time

Table 10 Comparison between Soft Computing Methods and Classical Methods for tuning PID Controller in Real Time

Controller	Angular Position (deg)	Rising Time (sec)	Settling Time (sec)	Overshoot (%)	Accuracy Error (%)
Ziegler-Nichols based PID Controller	40	0.066	0.086	0	0.025
Fuzzy Logic based PID Controller	40	0.045	0.06	0	0.025

8.3 Conclusion

From the results obtained for both the Classical Methods for tuning PID controller such as (Ziegler-Nichols) and Soft Computing Methods for tuning PID controller such as (Fuzzy Logic & Neural Network), it is clear that the overall performance of Soft Computing Methods for tuning PID controller proposed was better than the Classical Methods for tuning PID controller. Soft Computing Methods for tuning PID controller has shown good results and improvements of the system behavior. Soft Computing Methods for tuning PID controller was able to deal with all the non-linear parameters found on the system. The efficiency of Soft Computing methods (Fuzzy Logic and Neural Network for tuning parameters of PID controller) was also better in terms of the the step response characteristics such as, reducing the steady-states error; rise time, settling time and maximum overshoot, as shown in Figures (15), (34), (53), (58), (60), (64), (65) and tables (9), (10)

8.4 Future Work

(1) Using Genetic Algorithm technique for tuning PID Controller applied to A DC Motor.

(2) Design of Control Systems by using Soft Computing techniques for A Robot Arm.

REFERENCES

[1] F. O. Karray and C. de Silva, Soft Computing and Intelligent Systems Design Theory, Tools and Applications, 2004.

[2] A. Zilouchian, M. Jamshidi, Intelligent Control Systems Using Soft Computing Methodologies, 2001.

[3] K.Ogata, Modern Control Engineering, forth edition, Prentice Hall, 2002.

[4] H. Ugur, Artificial neural networks, Ankara 2004.

[5] L.A.Zadeh, Fuzzy Sets, Information and Control,1965.

[6] A. A. Mahfouz, M. M. K, F. A. Salem, Modeling, Simulation and Dynamics Analysis Issues of Electric Motor, for Mechatronics Applications, Using Different Approaches and Verification by MATLAB/Simulink (I), I.J. Intelligent Systems and Applications, 05, 39-57, DOI: 10.5815/ijisa.05.06, April 2013.

[7] A. faramarzi, K. sabahi, Recurrent Fuzzy Neural Network for DC motor control, IEEE Fifth International Conference on Genetic and Evolutionary Computing, 2011.

[8] S.M. Mokhsin, R.A. Hadi and B.N. Sheikh Rahimullah, Design of Artificial Neural Network (ANN) Based Rotor Speed Estimator for DC Drive, IEEE 7803-7565-2002.

[9] P. J. Gawthrop, Self-tuning PID control structures, Getting the Best Our of PID in Machine Control, IEE (Reference No.: 1996/287,1996. Troply, IEEE, 2013.

[10] A. Datta and L. Xing, The theory and design of adaptive internal model control schemes, Proceeding of American Control Conference, 1998.

[11] W. F. Xie and A. B. Rad, "Fuzzy adaptive internal model control", IEEE Trans , 1998.

[12] C. W. Tao and J.-S. Taur, Flexible Complexity Reduced PID-like Fuzzy Controllers , IEEE Transactions on Systems, Man, and Cybernetics, 2000.

[13] Zhiqiang Gao, T.A. Trautzsch and J.G Dawson, A stable self-tuning fuzzy logic control system for industrial temperature regulation, IEEE Transactions on Industry Applications, 2002.

[14] K. Watanabe and E. Muramatsu, Adaptive internal model control of SISO systems (Conference Paper style) , SICE Annual Conference in Fukui, 2003.

[15] Y. Baba, T. Shigemasa , M. Yukitomo, F. Kojima, M. Takahashi and E. Sasamura, Model-driven PID control system in single-loop controller, in Proceedings of the SICE annual conference Fukuri, 2003.

[16] X. Z. Jin, J. P. Sun, J. Z. Liu and L. F. Zhang, Analysis and design of adaptive internal model control for unstable process with dead time, International Conference on Machine Learning and Cybernetics, 2004.

[17] G. Han, L. Chen, J. Shao and Z. Sun, Study of Fuzzy PID Controller for Industrial Steam Turbine Governing System, proceedings of ISCIT, 2005.

[18] X. Gao, Y. Zhao, W. Gao and X. Yu, Simulation and research of fuzzy immune adaptive PID control in coke oven temperature control system, The Sixth World Congress on Intelligent Control and Automation, 2006.

[19] J. Chen, L. Wang and B. Du, Modified internal model control for chemical unstable processes with time-delay, in proceedings of 7th World Congress on Intelligent Control and Automation, 2008.

[20] G. Jha, Artificial Neural Network, Indian Agricultural Research Institute, PUSA, New Delhi-110 012.

[21] P.M. Mary and N.S. Marimuthu, Design of self-tuning fuzzy logic controller for the control of an unknown industrial process, IET control theory & applications, 2009.

[22] F. R. Gutierrez and Y. F Makableh, Efficient Position Control of DC Servomotor Using Backpropagation Neural Network, 2011 Seventh International Conference on Natural Computation, 2011.

[23] R. Kumar and V. Girdhar, High Performance Fuzzy Adaptive Control For D.C. Motor International Journal of Engineering Research, Technology (IJERT) ISSN: 2278-0181, Issue 8, October-2012.

[24] A. Krenker, J. Bester, A. Kos, Introduction to the Artificial Neural Networks, Faculty of Electrical Engineering, University of Ljubljana Slovenia.

[25] E. E. Ezema, I. I. Eneh, Improving the Functional Limitations of DC Motor using Neural Network Controller, International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 3, Issue 5, October 2013.

[26] F. Abdollahi, Computational Intelligence Lecture 9:Fuzzy Sets, Department of Electrical Engineering, Amirkabir University of Technolog, 2011.

[27] Y. Makableh, Efficient Control of DC Servomotor Systems Using Backpropagation Neural Networks, Georgia Southern University, Master Thesis, 2011.

[28] Wikipedia, the free encyclopedia, http://en.wikipedia.org/wiki/PID_controller, 2013.

[29] M. K. Pandey, Analysis of Self Tuning Fuzzy PID Internal Model Control, Thapar University, Master Thesis, July 2010.

[30] V. Syamasudha, N. Siddaiah, Design and Structure of Fuzzy Logic Using Adaptive Online Learning Systems, Annamacharya Institute of Technology & Sciences, Tirupati, A.P, India, International Journal of Emerging Trends & Technolgy in Computer Science, Volume 1, Issue 3, ISSN 2278-6856, September, October 2012.

[31] H. Demuth, M. Beale, M. Hagan, Neural Network Toolbox User's Guide, Mathworks, Inc, 2008.

[32] Quanser Documentation, http://www.quanser.com, 2012.

[33] K. Ahmad, C. Ormond, Fuzzy Logic Notes, 2010.

[34] Instruction Manual, Sensoray Model 626, PCI Multifunction I/O Board http://www.sensoray.com/products/626.htm, January 28, 2004.

[35] J. Jantzen, K. Lyngby, Tutorial On Fuzzy Logic, Technical University of Denmark, Oersted-DTU, Automation, Bldg 326, 2800, DENMARK. Tech. report no 98-E 868 (logic), revised 17 Jun 2008.

[36] A. Gill, Speed Control of Brushles DC Motor Using Artificial Neural Neywork Tuned PID Controller, Thapar University, Master Thesis, July 2012.

[37] W. Debiao, L. Taifu, Z. Bingxiang , Remodeling of Fuzzy PID Controller Based on BP Neural Network, Proceedings of the 7th World Congress on Intelligent Control and Automation, Chongqing, chine, June 25-27, 2008.

[38] K. B. M. Krishna, Fuzzy PID Control for Networked Control System of DC Motor with Random Design, Acharya Nagarjuna University, International Journal of Computer Applications (0975 – 8887) Volume 52 – No. 7, August 2012.

[39] P. P. Bonissone, E. H. Ruspini, Fuzzy Sets & Expert Systems in Computer Eng, 1995-1998.

[40] H. A. F. Almurib, A. A. Mat Isa and H. M.A.A. Al-Assadi, Direct Neural Network Control via Inverse Modelling: Application on Induction Motors, University of Nottingham, ISBN: 978-953-307-220-3, InTech, Available from: http://www.intechopen.com, 2011.

[41] MATLAB Documentation, http://www.mathworks.com, 2012.

[42] A. Atri, M. Ilyas, Speed Control of DC Motor using Neural Network Configuration, International Journal of Advanced Research in Computer Science and Software Engineering , Research Paper. Issue 5, ISSN: 2277 128X, May 2012.

[43] K. N. Sujatha, K.V. Anand, Artificial Intelligence Based Speed Control of Brushless DC Motor, IEEE, 978-1-4244-6551-2010.

[44] N. S. Nise, Control System Engineering, sixth edition, California State Polytechnic University, Copyright# by JohnWiley & Sons, Inc, 2011, 2006, 2003, 1996.

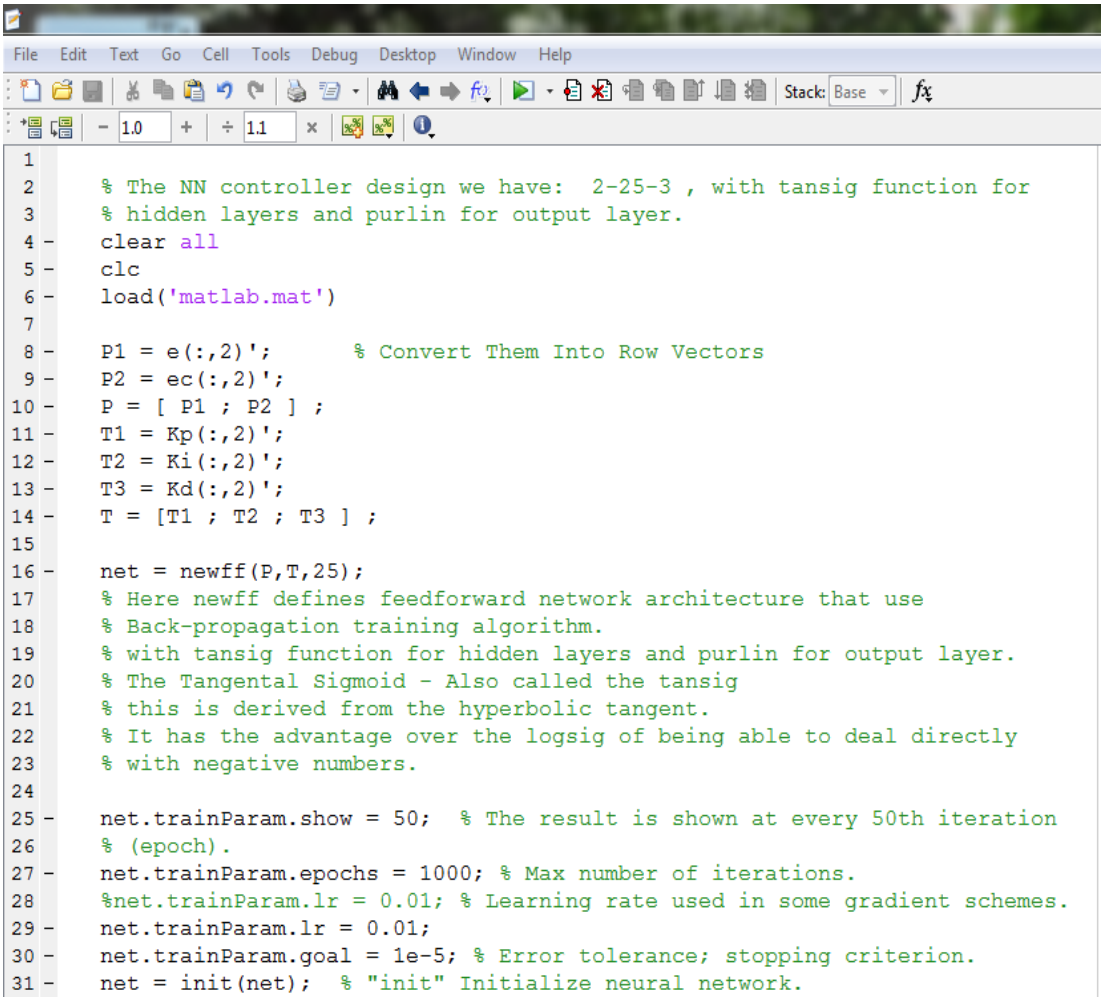
[45] A. P. Engelbrecht, Computational Intelligence, second edition, 2007.

[46] A. Tewari, Modern Control Design With MATLAB and SIMULINK, Indian Institute of Technology, Kanpur, India, 2002.

APPENDICES

APPENDIX A

MATLAB CODE OF PB NEURAL NETWORK



```
% The NN controller design we have:  2-25-3 , with tansig function for
% hidden layers and purlin for output layer.
clear all
clc
load('matlab.mat')

P1 = e(:,2)';      % Convert Them Into Row Vectors
P2 = ec(:,2)';
P = [ P1 ; P2 ] ;
T1 = Kp(:,2)';
T2 = Ki(:,2)';
T3 = Kd(:,2)';
T = [T1 ; T2 ; T3 ] ;

net = newff(P,T,25);
% Here newff defines feedforward network architecture that use
% Back-propagation training algorithm.
% with tansig function for hidden layers and purlin for output layer.
% The Tangental Sigmoid - Also called the tansig
% this is derived from the hyperbolic tangent.
% It has the advantage over the logsig of being able to deal directly
% with negative numbers.

net.trainParam.show = 50;  % The result is shown at every 50th iteration
% (epoch).
net.trainParam.epochs = 1000; % Max number of iterations.
%net.trainParam.lr = 0.01; % Learning rate used in some gradient schemes.
net.trainParam.lr = 0.01;
net.trainParam.goal = 1e-5; % Error tolerance; stopping criterion.
net = init(net);  % "init" Initialize neural network.
```

```
% Train network :
net = train(net,P,T); % Iterates gradient type of loop.
% Resulting network is strored in net.

Y = sim(net,P);
% The cell array Y is the output sequence of the network,
% which is also the output sequence of the second and third layers.
% The values you got for the second row can differ from those shown
% because of different initial weights and biases. However, they will
% almost certainly not be equal to targets T, which is also true of
% the values shown.

gensim(net,-1)  % Command to generate neural network SIMULINK diagram.
% (-1) is simple time.

% Description net = init(net) returns neural network net with weight and
% bias values updated according to the network initialization function
% indicated by net.initFcn, and the parameter values, indicated by
% net.initParam.
```

APPENDIX B

MOTOR SPECIFICATION SHEET

Series 2338 ... S — *See beginning of the Motor Section for Ordering Information*

		2338 S	4.5 S	006 S	009 S	012 S	018 S	024 S	
1	Nominal voltage	U_N	4.5	6	9	12	18	24	Volt
2	Terminal resistance	R ± 12%	1.4	2.6	5.7	10.0	23.5	38.0	Ω
3	Output power	$P_{2\,max.}$	3.39	3.23	3.29	3.31	3.18	3.50	W
4	Efficiency	$\eta_{max.}$	70	69	67	66	67	67	%
5	No-load speed	n_0 ± 12%	7,200	7,200	7,400	7,800	7,400	7,600	rpm
6	No-load current (with shaft ø 0.12 in)	I_0 ± 50%	0.100	0.080	0.060	0.050	0.030	0.025	A
7	Stall torque	M_H	2.55	2.42	2.41	2.29	2.32	2.49	oz-in
8	Friction torque	M_R	0.082	0.086	0.095	0.099	0.095	0.102	oz-in
9	Speed constant	k_n	1,650	1,240	855	678	428	330	rpm/V
10	Back-EMF constant	k_E	0.606	0.804	1.170	1.470	2.340	3.030	mV/rpm
11	Torque constant	k_M	0.818	1.088	1.586	1.997	3.158	4.107	oz-in/A
12	Current constant	k_I	1.222	0.919	0.630	0.501	0.317	0.244	A/oz-in
13	Slope of n-M curve	$\Delta n/\Delta M$	2,824	2,975	3,071	3,406	3,190	3,052	rpm/oz-in
14	Rotor inductance	L	100	180	380	630	1,400	2,600	µH
15	Mechanical time constant	τ_m	20	17	17	17	17	17	ms
16	Rotor inertia	J	$6.797 \cdot 10^{-5}$	$5.523 \cdot 10^{-5}$	$5.240 \cdot 10^{-5}$	$4.815 \cdot 10^{-5}$	$5.098 \cdot 10^{-5}$	$5.381 \cdot 10^{-5}$	oz-in-sec²
17	Angular acceleration	$\alpha_{max.}$	38	44	46	48	46	47	$\cdot 10^3$rad/s²
18	Thermal resistance	$R_{th\,1}$ / $R_{th\,2}$	3 / 24						°C/W
19	Thermal time constant	τ_{w1} / τ_{w2}	5.7 / 645						s
20	Operating temperature range:								
	– motor		– 30 to +85 (– 22 to +185)						°C (°F)
	– rotor, max. permissible		+125 (+257)						°C (°F)
	Note: Special operating temperature models for –55°C to +125°C (– 67°F to +257°F) available on request.								
21	Shaft bearings		sintered bronze sleeves		ball bearings		ball bearings, preloaded		
22	Shaft load max.:		(standard)		(optional)		(optional)		
	– with shaft diameter		0.1181		0.1181		0.1181		in
	– radial at 3,000 rpm (0.12 in from bearing)		9		72		72		oz
	– axial at 3,000 rpm		1.08		7.2		7.2		oz
	– axial at standstill		72		72		72		oz
23	Shaft play:								
	– radial	≤	0.0012		0.0006		0.0006		in
	– axial	≤	0.0079		0.0079		0		in
24	Housing material		steel, zinc galvanized and passivated						
25	Weight		2.47						oz
26	Direction of rotation		clockwise, viewed from the front face						
Recommended values									
27	Speed up to	$n_{e\,max.}$	6,000	6,000	6,000	6,000	6,000	6,000	rpm
28	Torque up to	$M_{e\,max.}$	0.566	0.566	0.566	0.566	0.566	0.566	oz-in
29	Current up to (thermal limits)	$I_{e\,max.}$	1.380	1.000	0.680	0.510	0.330	0.260	A

APPENDIX C

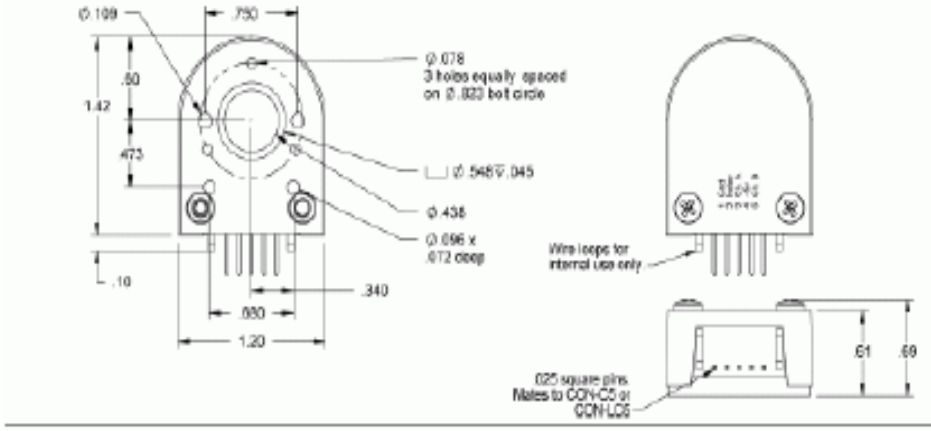
SPECIFICATION SHEET OF DC MOTOR (SRV02)

Symbol	Description	Matlab Variable	Value	Unit	Variation
V_{nom}	Motor nominal input voltage.		6.0	V	
R_m	Motor armature resistance.	Rm	2.6	Ω	±12%
L_m	Motor armature inductance.	Lm	0.18	mH	
k_t	Motor torque constant	k_t	7.68E-03	N·m	±12%
η_m	Motor efficiency.	Eff_M	0.69		±5%
k_m	Back-emf constant	k_m	7.68E-03	V/(rad/s)	±12%
$J_{m,rotor}$	Motor shaft moment of inertia.	Jm_rotor	3.90E-7	kg·m²	±10%
K_{gi}	Internal gearbox ratio.	Kgi	14		
$K_{ge,\ low}$	External low-gear configuration ratio.	Kge	1		
$K_{ge,\ high}$	External high-gear configuration ratio.	Kge	5		
K_g	Low-gear Total gearbox ratio.	Kg	14		
	High-gear total gearbox ratio.		70		
η_g	Gearbox efficiency.	Eff_G	0.90		±10%
J_{tach}	Tachometer moment of inertia.	Jtach	7.06E-08	kg·m²	±10%
m_{24}	Mass of 24-tooth gear.	m24	0.005	kg	
m_{72}	Mass of 72-tooth gear.	m72	0.030	kg	
m_{120}	Mass of 120-tooth gear.	m120	0.083	kg	
r_{24}	Radius of 24-tooth gear.	r24	6.35E-03	m	
r_{72}	Radius of 72-tooth gear.	r72	0.019	m	
r_{120}	Radius of 120-tooth gear.	r120	0.032	m	
J_{eq}	Equivalent high-gear moment of inertia without external load.	Jeq	9.76E-05	kg·m²	±10%
	Equivalent high-gear moment of inertia without external load.	Jeq	2.08E-03	kg·m²	±10%
B_{eq}	Low-gear viscous damping coefficient found experimentally.	Beq	1.50E-04	N·m/(rad/s)	±20%
	High-gear viscous damping coefficient found experimentally.	Beq	0.015	N·m/(rad/s)	±20%
m_b	Mass of bar load.	m_b	0.038	kg	
L_b	Length of bar load (from end-to-end).	L_b	0.1525	m	
m_d	Mass of disc load.	m_d	0.04	kg	
r_d	Radius of disc load.	r_d	0.05	m	
K_{pot}	Potentiometer sensitivity	K_POT	35.2	deg/V	±2%
K_{enc}	SRV02-E encoder resolution.	K_ENC	4096	counts/rev	
K_{enc}	SRV02-EHR encoder resolution.	K_ENC	8192	counts/rev	
K_{tach}	SRV02-T tachometer sensitivity.	K_TACH	1.50	k_{RPM}/V	±2%
m_{max}	Maximum output shaft load.		5.0	kg	
f_{max}	Maximum input voltage frequency.		50.0	Hz	
I_{max}	Maximum input current		1.0	A	
ω_{max}	Maximum motor speed.		628.3	rad/s	

APPENDIX D

ENCODER SPECIFICATION SHEET

Mechanical Drawing:



Mechanical Specifications:

Parameter	Dimension	Units
Moment of Inertia	5.0 x 10^{-4}	oz-in-s^2
Hub Set Screw Size	3-48 or 4-48	in.
Hex Wrench Size	.050	in.
Encoder Base Plate Thickness	.135	in.
3 Mounting Screw Size	0-80	in.
2 Mounting Screw Size	2-56 or 4-40	in.
3 Screw Bolt Circle Diameter	.820 ± .005	in.
2 Screw Bolt Circle Diameter	.750 ± .005	in.
Required Shaft Length	.445 to .570*	in.
With E-option	.485 to .795*	in.
With H-option	>.445*	in.

* Add .125" to the required shaft length when using R-option.

Pin-out:

Pin	Description
1	Ground
2	Index
3	A channel
4	+5VDC power
5	B channel

Electrical Specifications:

For complete details see the *EM1 / HEDS* data sheet.

Phase Relationship:

B leads A for clockwise shaft rotation, and A leads B for counterclockwise rotation viewed from the cover/label side of the encoder (see the *EM1 / HEDS* data sheet).

APPENDIX E

POTENTIOMETER SPECIFICATION SHEET

ELECTRICAL SPECIFICATIONS			
PARAMETER	MIL-PRF-12934/MIL-PRF-39023 TEST PROCEDURES APPLY		
	STANDARD	**SPECIAL**	
Total Resistance: Model 132 Wirewound	5Ω to 20KΩ	to 35KΩ	
Tolerance: 50Ω and above	± 3%	± 1%	
Below 50Ω	± 5%	± 3%	
Model 138 Conductive Plastic	1KΩ to 50KΩ	–	
Tolerance:	± 10%	± 5%	
Model 139 Cermet	500Ω to 2MΩ	–	
Tolerance:	± 20%	± 5%	
Linearity (Independent)	**STANDARD**	**BEST PRACTICAL**	
Total Resistance (132)			
5Ω to 20Ω	± 1.0%	± 0.75%	
20Ω to 200Ω	± 1.0%	± 0.50%	
200Ω and above	± 0.5%	± 0.25%	
138/139	± 0.5%	± 0.25%	
Noise (132)	100Ω ENR		
Output Smoothness (138 & 139)	0.1% maximum		
Power Rating	40°C Ambient		
Model 132	2.75 watts		
Model 138	2 watts		
Model 139	5 watts		
	All Models derated to zero at 125°C		
Electrical Rotation	**MODEL 132**	**MODEL 138**	**MODEL 139**
Continuous	352° ± 2°	345° ± 4°	345° ± 4°
Stops	336° ± 2°	336° ± 4°	336° ± 4°
Insulation Resistance	1000MΩ minimum at 500VDC		
Dielectric Strength	$1000V_{RMS}$, 60Hz		
Absolute Minimum Resistance	1.0% of total resistance or 0.5Ω whichever is greater (132 only)		
Minimum Voltage	0.5% maximum		
Temperature Coefficient of Resistance			
132	Refer to standard resistance element data		
138	± 500ppm/°C maximum		
139	± 100ppm/°C maximum		