Comparative performance study of PSO and BFO algorithm based PID Controller Tuning for Speed Control of DC Motor Drives

A Thesis report

submitted towards the partial fulfillment of the

requirements of the degree of

Master of Engineering

in

Power System and Electric Drives

submitted by

Brijesh Silswal

(801041032)

Under the supervision of

Mr. Souvik Ganguli

Assistant Professor, EIED

DEPARTMENT OF ELECTRICAL AND INSTRUMENTATION ENGINEERING

THAPAR UNIVERSITY
PATIALA – 147 004

JUNE, 2012

CERTIFICATE

I hereby declare that the report entitled **"Comparative performance study of PSO and BFO algorithm based PID Controller Tuning for Speed Control of DC Motor Drives"** is an authentic record of my own work carried out as requirements for the award of degree of M.E. (Power System & Electric Drives) at Thapar University, Patiala under the guidance of Mr. Souvik Ganguli (Assistant Professor, EIED) during January to June, 2012.

Date: 15/06/2012

Place: Thapar University, Patiala

Brijesh

Brijesh Silswal

Roll No. 801041032

It is certified that the above statement made by the candidate is correct to best my knowledge and belief.

S. Ganguli

Mr. Souvik Ganguli

Assistant Professor, EIED

(Supervisor)

Thapar University, Patiala

S. Ghosh 6/7/12

Dr. Smarajit Ghosh

Professor & Head of Department, EIED

Thapar University, Patiala

Dr. S. K. Mohapatra

Dean of Academic Affairs

Thapar University, Patiala

ACKNOWLEDGEMENT

I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of this thesis. I would like to express my gratitude and sincere thanks to Mr. Souvik Ganguli, Assistant Professor whose guidance, support and continuous encouragement helped me being motivated towards excellence throughout the course of this work.

I wish to extend my gratitude towards Dr. Smarajit Ghosh, Professor and Head, EIED, who has given me an inspiration throughout this work.

I would thank all of them who have been helpful and were associated with me directly and indirectly throughout the work. I would also like to thank my friends who helped me in completion of my thesis.

Finally, no words to thanks my dearest parents whose support and take care me during this workout.

Date: Brijesh Silswal

Place: Thapar University, Patiala (801041032)

ABSTRACT

Particle Swarm Optimization (PSO) algorithm and Bacterial Foraging Optimization (BFO) algorithm are stochastic optimization techniques that come under the category of bio-inspired optimization techniques.

The objective of thesis is to analyze the implementation of PSO algorithm and BFO algorithm to PID Controller optimal tuning for DC machine model servo speed control. The model of a separately excited DC motor is considered as a second order system for speed control. The comparative study is based on step response, time domain analysis and frequency domain analysis. First various classical PID tuning methods are discussed and implemented on DC machine model and their results are analyzed. Next optimal tuning methods of PID tuning using various Error Performance indices like IAE, ISE, ITAE and ISTE is applied for DC machine model. Than PSO algorithm is utilized to optimize the parameters of PID controller to obtain optimal performance for DC machine model. Next BFO algorithm is implemented for DC machine model control using PID controller by obtaining optimal values of Kp, Ki, Kd parameters. The results obtained for PSO algorithm and BFO algorithm are compared mainly on basis of time domain analysis and frequency domain analysis. MATLAB is used for all the calculation and analysis necessary for the thesis.

The conventional approach is not very efficient due to the presence of non-linearity in the system. The output of the conventional PID system has a quite high overshoot and settling time. The main aim of thesis is to apply two techniques namely PSO and BFO to design and tuning of PID controller to get an output with better dynamic and static performance. The application of PSO and BFO to the PID controller imparts it the ability of tuning itself automatically in an on-line process while the application of optimization algorithm to the PID controller makes it to give an optimum output by searching for the best set of solutions for the PID parameters.

CONTENTS **PAGE NO.**

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AI	-	Artificial Intelligence
ZN	-	Ziegler-Nichols
AMIGO	-	Approximate M-constrained Integral Gain Optimization
IMC	-	Internal Model Control
CHR	-	Chien-Hrones-Reswick
IAE	-	Integral of Absolute magnitude of Error
ISE	-	Integral of Square of Error
ITAE	-	Integral of Time multiplied by Absolute Error
ISTE	-	Integral of Square of Time multiplied by Error
PSO	-	Particle Swarm Optimization
BFO	-	Bacterial Foraging Optimization
K	-	Gain
Θ	-	Dead time or time delay
T	-	Open loop process Time constant
K_u	-	Ultimate gain
P_u	-	Ultimate period

CHAPTER 1:-

INTRODUCTION

1.1- OBJECTIVE OF THESIS

Control systems appear practically everywhere in our homes, in industry, in communications, information technologies, etc. Increasing problems with process non-linearity's, operating constraints, time delay, uncertainty etc. have lead to development of more sophisticated control strategies. So there is always a continuous search for a better technique which can consider all such problems and provide stability to the system. Artificial Intelligence (AI) based techniques are the new entry in field of PID tuning. AI based techniques have proved their capabilities in various fields and so these are considered to give a new pace to PID tuning search technique. The objective of thesis is to give a comparative analysis of performance of PSO and BFO based on results of DC Motor Servo response for speed control and find out which of the two methodologies gives better results for the selected DC motor model.

Matlab is used to analyze the performance of PSO and BFO for DC motor model for servo response (speed control). The comparative study is based on the tuning parameters obtained for PID controller by the two techniques namely PSO and BFO; and the time domain analysis (rise time, settling time, maximum overshoot) of DC motor with this optimal PID controller.

The basic model of a separately excited DC motor model has been discussed. Transfer functions are derived for Speed Control and Position control as well. The step response of DC motor for open loop and closed loop are observed without controller as well as with controller for Speed control only. The values of rise time, settling time, maximum overshoot etc. are observed for uncontrolled system and then controller is designed so as to be keep them within range for stable operation.

The aspect of ideal PID controller is briefly discussed along with its basic block diagram for parallel form. There are various classical PID tuning methods for open loop and closed loop which have been popular for many years before emergence of Artificial

Intelligence based tuning methods. These classical methods namely Ziegler-Nichols (ZN) Open Loop method, Cohen-Coon Tuning method, AMIGO Tuning method, IMC Tuning method, CHR Step response Tuning method, Ziegler-Nichols (ZN) Closed loop Method, Modified Ziegler-Nichols Tuning Method, Tyreus-Luyben Tuning Method, Chien-Hrones-Reswick PI Tuning Method, Astrom-Hagglund PI Tuning Method are described in brief. These methods are then implemented to our DC motor model for PID parameter tuning. These methods are compared on basis of their results to get the method, among all classical methods listed above, which best suits our objective.

Due to limitations of classical PID tuning techniques, there was need for a method which gave optimal performance for PID controller. Two such error indices based optimal PID tuning methods, one proposed by P.W. Murril *et al.* and other proposed by M. Zhuang *et al.* , are briefly discussed. These methods are applied for PID controller optimal tuning in case of DC motor model and the results are observed. The results obtained with these methods are compared to Classical PID tuning methods for rise time, settling time and overshoot.

The introduction of Artificial Intelligence based PID tuning methods are becoming popular these days over Classical tuning methods due to their higher performance and faster speed. Particle Swarm Optimization (PSO) and Bacterial Foraging Optimization (BFO) are two bio-inspired optimization techniques which have gained popularity recently in field of PID tuning. These two methods are discussed along with their detailed algorithm structure. BFO and PSO algorithms are implemented for PID tuning. Their performance is analyzed on basis of their tuning results for DC motor model and better one is concluded.

1.2- CHAPTER LAYOUT

The chapter layout of the thesis is as follows:

Chapter 1: This chapter includes a brief introduction of the thesis.

Chapter 2: Literature Review related to the thesis is given in chapter 2.

Chapter 3: Dc motor modeling and its transfer functions for speed and position

control are discussed in chapter 3.

Chapter 4: Ideal PID controller structure and its various tuning methods (open loop and closed loop tuning methods) are included in this chapter for Speed Control.

Chapter 5: Optimal tuning of PID controller is also discussed here.

Chapter 6: PSO and its application in optimal PID tuning is included in this chapter.

Chapter 7: Optimal tuning of PID controller using BFO is discussed in this chapter.

CHAPTER 2:-

LITERATURE REVIEW

There are a lot of literature which includes work related to PSO and BFO. The literature review of those literatures which are relevant with this thesis work is given below.

Zafer Bingul *et al.* conducted their research on '*Tuning of Fractional PID Controllers Using PSO Algorithm for Robot Trajectory Control*'. In their research of robot trajectory control, Fractional order PID (FOPID) controller is tuned with PSO and the results are compared with Fuzzy logic controller & PID tuned by PSO. Proposed method of FOPID controller using PSO is better than other techniques [1].

V. K. Kadiyala *et al.* in their research paper titled *'Design and implementation of fractional order PID controller for Aero fin control system*' proposed a Fractional order PID (FOPID) controller of electromagnetic actuator (EMA) system for Aero fin control (AFC) using PSO. The EMA is realized with permanent magnet brush DC motor which is driven by a constant current driver. Using the non-linear model of EMA-AFC system which includes the non-linearities of DC motor, an FOPID position controller is designed using different soft computing techniques like PSO in SIMULINK so that the system satisfies all the design requirements. The proposed method gives results better than conventional methods for Electromagnetic actuator (EMA) system for AFC [2].

C. H. Liu *et al.* in '*Design of a Self-Tuning PI Controller for a STATCOM Using Particle Swarm Optimization*' suggested a self tuning PI controller for STATCOM in which controller gains are adapted using PSO technique. Based on the estimated system load, a PSO algorithm, which takes the best particle gains, the best global gains, and previous change of gains into account, is employed to reach the desired controller gains. Results of self tuned PI controller are compared with fixed gain PI controller [3].

D. Maiti *et al.* in their paper '*Tuning PID and* $PI^{\lambda}D^{\delta}$ *Controllers using the Integral Time Absolute Error Criterion*' proposed application of PSO to the problem of designing a fractional order PID ($PI^{\lambda}D^{\delta}$) controller comprising integral order λ , derivative order δ along with ITAE minimization criterion. The digital realization of the deigned system utilizes the

Tustin operator-based continued fraction expansion scheme. This paper also attempts to study the behavior of fractional PID controller vis-à-vis that of its integer-order counterpart and demonstrates the superiority of the former to the latter. [4]

C. M. Lin *et al.* in their research paper '*Robust PID control system design for chaotic systems using particle swarm optimization algorithm*' proposed Robust PID (RPID) control system for chaotic systems comprising a self tuning PID (SPID) controller. The gradient descent method and H_∞ control technique are utilized to derive the on-line tuning laws of SPID controller and the robust controller, so that the robust tracking performance of the system can be yielded. In order to achieve the best solution for SPID controller, the particle swarm optimization (PSO) algorithm is adopted to select the optimal learning rates of SPID controller. A robust controller is discussed in this paper and PSO is used achieve best solution resulting in favorable tracking performance. [5]

W.W. Cai *et al.* worked on '*Design and simulation of intelligent PID controller based on particle swarm optimization*' and proposed an Intelligent PID control method in this paper in which the different parameters are controlled using size of system error and PSO algorithm is used to tune this PID further. According to the size of the system error, this algorithm controls the system with different subsections of different parameters, by using the particle swarm optimization (PSO) to optimize the parameters of the PID controller to solve the problems, such as lag, time-variety and non-linearity. A group of parameters of the PID controller that minimize the evaluation function is calculated rapidly by searching in the given controller parameters area. This method is better than PID optimized by PSO for 1^{st} and 2^{nd} order system with time delay [6].

Y. Bo *et al.* research paper '*A New PSO-PID Tuning Method for Time-delay Processes*' suggested a method in which a relation is developed between coefficients of corresponding powers of s in the numerator and those in denominator of the closed loop transfer function for FOPTD process by using PSO algorithm. The proposed method originates from processes with small time delay; however, it is still effective even if the time delay is quite large. The results show that the proposed method gives significantly better dynamic performances than IMC-PID method [7].

In paper of A. A. El-Gammal *et al.* titled '*A Modified Design of PID Controller For DC Motor Drives Using Particle Swarm Optimization PSO*', PID controller is made adaptive by using PSO algorithm by adjustment of gains to obtain minimum IAE between speed demand and output response. The new technique converts all objective functions to a single objective function by deriving a single aggregate objective function using specified or selected weighting factors. The weighting factors are also treated as dynamic optimization parameters within PSO. Experimental results show that the performance of the optimal PID controller is better than that of the traditional PID controller [8].

In research paper of V. Rajinikanth *et al.* titled *'I-PD Controller Tuning for Unstable System using Bacterial Foraging Algorithm: A study based on various Error Criterion* '; a novel method to tune the I-PD controller structure for the time-delayed unstable process (TDUP) using Bacterial Foraging Optimization (BFO) algorithm is proposed. The tuning process is focused to search the optimal controller parameters $\left(K_p, K_i, K_d\right)$ by minimizing the multiple objective performance criterion. The results shows that the tuning approach is a model independent approach and provides enhanced performance for the set point tracking with improves time domain specifications [9].

E. Salim Ali *et al.* worked on '*Optimal PID Tuning for Load Frequency Control Using Bacteria Foraging Optimization Algorithm*' and proposed a BFOA based Load Frequency Control (LFC) for the suppression of oscillations in power system. A two area non reheat thermal system is considered to be equipped with PID controllers. BFOA is employed to search for optimal controllers parameters to minimize certain performance index. The performance of the proposed controllers has been evaluated with the performance of the conventional integral (I) controller in order to demonstrate the superior efficiency of the proposed BFOA in tuning PID controllers [10].

T. Jain *et al.* worked on *'Optimization of PD-PI Controller Using Swarm Intelligence'*. In this paper, the idea of model generation and optimization is explored for PD-PI controller. Most commonly known, the highly nonlinear Inverted Pendulum system is used as a test system for this approach. A comparison between Evolutionary Algorithms namely GAs (Genetic Algorithms), and Swarm Intelligence, i.e., PSO (Particle Swarm Optimization) and BG (Bacterial Foraging) has been carried out on the basis of performance indices: ITAE

(Integral Time Absolute Error), ISE (Integral Square Error), IAE (Integral Absolute Error) and MSE (Mean Square Error) and settling time. The simulations are tabulated in section IV to analyze which technique gives promising results for the system. [11].

In the research work of D. H. Kim *et al.* titled *'A Biologically Inspired Intelligent PID Controller Tuning for AVR Systems'*; a hybrid approach involving Genetic Algorithm (GA) and Bacterial Foraging (BF) for tuning the PID controller of an AVR is proposed. The proposed method is first illustrated using four test functions and the performance of the algorithm is studied with an emphasis on mutation, crossover, variation of step sizes, chemotactic steps, and the life time of the bacteria [12].

W. Korani worked on the research paper titled *'Bacterial Foraging Oriented by Particle Swarm Optimization Strategy for PID Tuning'*. This paper presents a new algorithm for PID controller tuning based on a combination of the foraging behavior of E coli bacteria foraging and Particle Swarm Optimization (PSO). The new algorithm is proposed to combines both algorithms advantages in order to get better optimization values. The proposed algorithm is applied to the problem of PID controller tuning and is compared with conveniently Bacterial Foraging algorithm and Particle swarm optimization [13].

T. Jain *et al.* created a research paper on *'Implementation of PID Controlled SIMO Process on FPGA Using Bacterial Foraging for Optimal Performance*'. In this paper, the design and implementation of a controlled Single Input Multi Output (SIMO) process is carried out on Field Programmable Gate Array (FPGA). The process is controlled by Proportional-Integral-Differentiator (PID) controller. Bacterial Foraging (BF) algorithm is used for tuning the parameters of PID controller for optimal performance [14].

CHAPTER 3:-

DC MOTOR MODELING

3.1- INTRODUCTION

In this chapter the basic model of a separately excited DC motor model has been discussed. Transfer functions are derived for Speed Control and Position control as well. Its performance curves without any controller and with controller (PID) are also included with which we can see that PID controller stabilizes the system faster.

3.2- DC MOTOR MODEL (SEPARATELY EXCITED)

The design method uses the concepts of the system theory, such as signals and systems, transfer functions, direct and inverse Laplace transforms. This requires building the appropriate Laplace model for each component of the whole system.

In order to build the DC motor's transfer function, its simplified mathematical model has been used. This model consists of differential equations for the electrical part, mechanical part and the interconnection between them. The electric circuit of the armature and the free body diagram of the rotor are shown in figure 3.1.

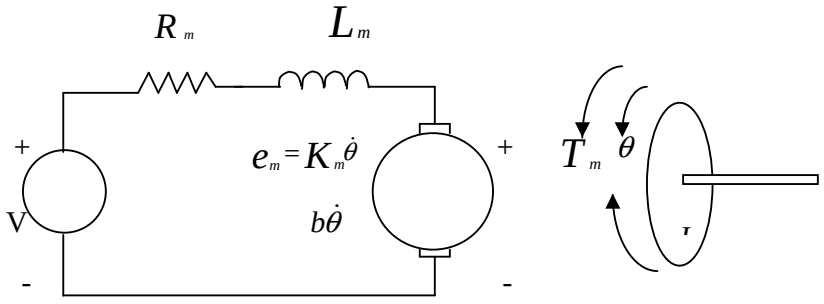

Figure 3.1-The electric circuit of the armature and the free body diagram of the rotor for a DC motor

The motor torque T_m is related to the armature current, i, by a constant factor K_t. The back emf, e_m, is related to the rotational speed, $\dot{\theta}$ by the following equations:-

$$T_m = K_t i \quad \text{... (3.1)}$$

$$e_m = K_e \dot{\theta} \quad \text{... (3.2)}$$

Assuming that K_t (torque constant) = K_e (electromotive force constant) = K_m (motor constant).

From figure 1 and above known values, the following equations can be written based on Newton's law combined with Kirchhoff's law:

$$T_m - T_L = J \frac{d^2\theta(t)}{dt^2} + B \frac{d\theta(t)}{dt} \quad \text{... (3.3)}$$

$$J\ddot{\theta} + B\dot{\theta} = K_m i - T_L \quad \text{... (3.4)}$$

$$L_m \frac{di}{dt} + R_m i = V - K_m \dot{\theta} \quad \text{... (3.5)}$$

3.3- TRANSFER FUNCTION:-

Using Laplace transform, the above equations can be expressed in terms of s-domain

$$(Js + B)\dot{\theta}(s) = K_m I(s) - T_L(s) \quad \text{... (3.6)}$$

$$(L_m s + R_m) I(s) = V(s) - K_m s\dot{\theta}(s) \quad \text{... (3.7)}$$

By eliminating $I(s)$, the following open-loop transfer function can be obtained, where the rotational speed $\dot{\theta}$ is the output and the voltage V is the input.

a) Speed Control:-

Assuming T_L (load torque) = 0,

T_f (friction torque) = 0 and,

feedback constant $K_v = K_m$;

the transfer function is:-

$$\left.\frac{\dot{\theta}(s)}{V(s)}\right|_{T_L(s)=0} = \frac{K_m}{(Js+B)(L_m s+R_m)+K_m{}^2} \quad \text{... (3.8)}$$

The block diagram obtained from this equation is shown in figure 3.2:

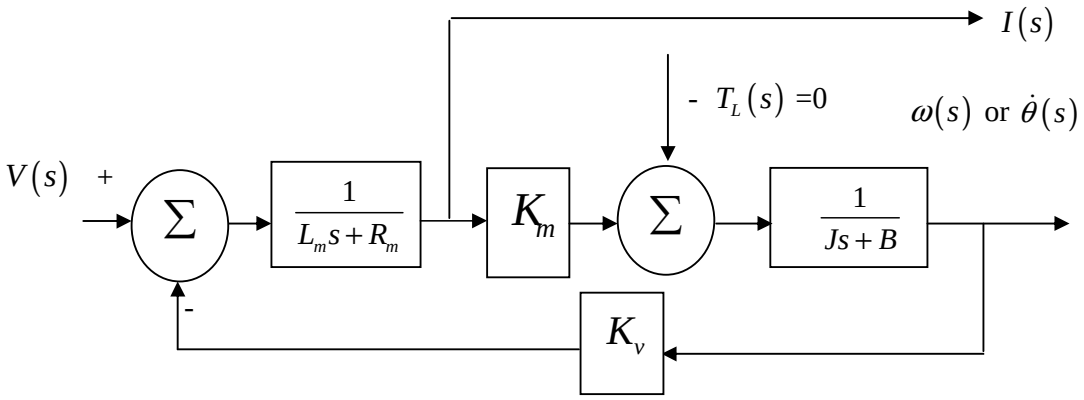

Figure 3.2- Block diagram representation for Speed Response with T_L =0

Assume V(s) = 0;

For Speed Control, the transfer function is given (from equations 3.6 and 3.7):-

$$\left.\frac{\dot{\theta}(s)}{T_L(s)}\right|_{V(s)=0} = \frac{-(L_m s+R_m)}{\left[(Js+B)(L_m s+R_m)+K_m{}^2\right]} \quad \text{... (3.9)}$$

The block diagram obtain for this equation is shown in figure 3.3 drawn below:-

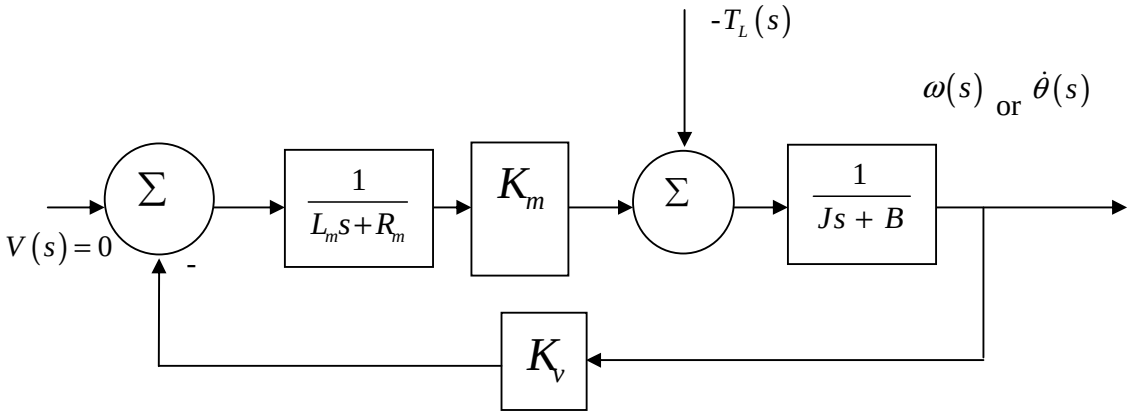

Figure 3.3- Block diagram representation for Speed Response with V(s) = 0

b) Position Control:-

Assuming T_L =0, for position control the transfer function is given as:-

$$\left.\frac{\dot{\theta}(s)}{V(s)}\right|_{T_L(s)=0} = \frac{K_m}{s\left[(Js+B)(L_m s+R_m)+K_m{}^2\right]} \quad \text{... (3.10)}$$

Assuming V(s)=0, for Position Control, the transfer function is given as:-

$$\left.\frac{\dot{\theta}(s)}{T_L(s)}\right|_{V(s)=0} = \frac{-(L_m s+R_m)}{s\left[(Js+B)(L_m s+R_m)+K_m{}^2\right]} \quad \text{... (3.11)}$$

3.4- DESIGN REQUIREMENTS:-

Since the most basic requirements of a motor are that it should rotate at the desired speed, the steady-state error e_{ss} of the motor speed should be less than 1%. The other performance requirement is that the motor must accelerate to its steady-state speed $\dot{\theta}_{ss}$ as soon as it turns on. In this case, it is desirable to have a settling time T_s in less than 2 sec. since speed faster than the reference may damage the equipment, a peak overshoot M_p of less than 5% is wanted. For a system to be stable, gain margin must be greater than 6 dB (min 4dB), phase margin must fall in the range between 30^0 to 60^0**;** robustness parameters include Modulus Margin (must be greater than 0.5 or -6dB, minimum -8 dB) and Delay Margin (must be greater than 1.5 ms)

3.5- PID CONTROL DESIGN METHOD:-

Different characteristics of the motor response (steady-state error, peak overshoot, rise time, etc.) are controlled by selection of the three gains that modify the PID controller dynamics.

Figure 3 shows how the PID controller works in the closed-loop system. The variable e represents the tracking error; the difference between the desired input value R and the actual output Y. This error signal will be sent to the PID controller computes both the derivative and the integral of this signal.

Therefore, the PID controller is defined by the relationship between the controller input e and computes output u that is applied to the motor armature:

$$u = K_p e + K_i \int e dt + K_d \frac{de}{dt} \qquad \text{... (3.12)}$$

Where K_p - Proportional gain

K_i - Integral gain

K_d - Derivative gain.

The signal u will be sent to the plant and the new output Y will be obtained and sent back to the sensor again to find the new error signal e. The controller takes e and computes its derivative and it's integral again.

This process goes on and on. By adjusting the weighting constants K_p, K_i and K_d, the PID controller can be set to give the desired performance.

(PID controller is discussed in detail in Chapter 4)

Taking Laplace Transform of equation (3.12) gives the following transfer function:

$$K(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s$$

$$= \frac{K_d s^2 + K_p s + K_i}{s} \qquad \text{... (3.13)}$$

This transfer function clearly illustrates the proportional, integral and derivative gains that make up the PID compensation.

CHAPTER 4:-

PID CONTROLLER AND ITS TUNING METHODS

4.1- INTODUCTION:-

PID is the most common and most popular feedback controller used in Industrial Process today. A PID controller calculates an "error" value as the difference between a measured process variable and a desired ‘set point’. PID controller is also known as three-term control:- the proportional (P), integral (I) and derivative (D). By tuning these three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements.

4.2- P+I+D CONTROLLER:-

As we have stated that PID controller is a sum of P, I and D controller, so now we must also discuss the function of each.

- Proportional (P) controller calculates the term proportional to the ‘error’.
- Integral (I) controller calculates a term proportional to integral of ‘error’. An integral controller gives zero SSE (Steady State Error) for a step input but reduces the speed of a system.
- Derivative (D) controller calculates a term proportional to derivative of ‘error’. A derivative control terms often produces faster response.

4.2.1- Proportional (P) term:

The proportional term 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:

$$P_{out} = K_p e(t) \quad \ldots (4.1)$$

4.2.2- Integral (I) term:

The contribution from the integral term is proportional to both the magnitude of the error and the duration of the error. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously. The accumulated error is then multiplied by the integral gain (K_i) and added to the controller output.

The integral term is given by:

$$I_{out} = K_i \int_0^t e(\tau) d\tau \qquad \ldots (4.2)$$

4.2.3- Derivative (D) term:

The derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_d. The magnitude of the contribution of the derivative term to the overall control action is termed the derivative gain, K_d.

The derivative term is given by:

$$D_{out} = K_d \frac{d}{dt} e(t) \qquad \ldots (4.3)$$

4.2.4- PID Term:-

The PID controller output can be obtained by adding the three terms.

$$G_c(s) = P + I + D = K_p + \frac{K_i}{s} + K_d s \qquad \ldots (4.4)$$

Or

$$G_c(s) = K_p \left(1 + \frac{1}{T_i s} + T_d s \right) \qquad \ldots (4.5)$$

Where Kp= Proportional gain,

Ki= integration coefficient,

Kd= derivative coefficient,

Ti= integral action time,

Td= derivative action time

The block diagram of PID controller is shown below:-

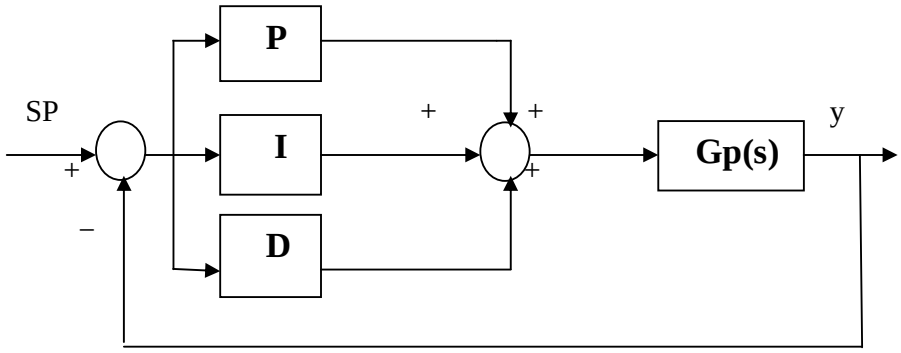

Figure 4.1- Block diagram representation of a PID controller

4.3- TUNING OG PID CONTROLLER

It have been noticed that mostly the PID's are poorly tuned and so efforts are being made systematically in this matter. Here we will discuss some popular classical tuning techniques and then some recently developed special tuning methods.

4.3.1- OPEN LOOP PID TUNING METHODS

4.3.1.1-INTRODUCTION

Open loop tuning methods are those where the feedback controller is disconnected and the experimenter excites the plant and measures the response. The key point here is that since the controller is disconnected, clearly the plant is no longer strictly under control. If the loop is critical, then this test could be hazardous. Indeed if the process is open-loop unstable, then we will be in trouble before we begin. Notwithstanding for many process control applications, open loop type experiments are usually quick to perform, and deliver

informative results. If the system is steady at set point, and remains so, then we have no information about how the process behaves.

There are various tuning strategies based on an open-loop step response. While they all follow the same basic idea, they differ slightly in:- how they extract the model parameters from the recorded response and relate appropriate tuning constants to the model parameters. There are four different methods, the classic Ziegler-Nichols open loop method, the Cohen-Coon method, Internal Model Control (IMC) method and Approximate M-constrained Integral Gain Optimization (AMIGO) method. Naturally if the response is not sigmoidal or 'S' shaped and exhibits overshoot, or an integrator, then this tuning method is not applicable.

This method implicitly assumes the plant can be adequately approximated by a first order transfer function with time delay,

$$G_p = \frac{Ke^{-\theta s}}{Ts+1} \qquad \text{... (4.6)}$$

Where: K- is gain,

θ- is the dead time or time delay, and

T- is the open loop process time constant.

Once we have recorded the open loop input/output data, and subsequently measured the times T and θ, the PID tuning parameters can be obtained directly from the given tables for different classical methods.

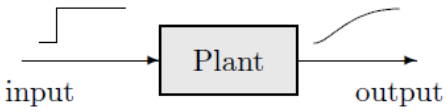

Figure 4.2- Open Loop response for a step input

The method is based on computing the times t1 and t2 at which the 35.3% and 85.3% of the system response is obtained as shown in the figure:

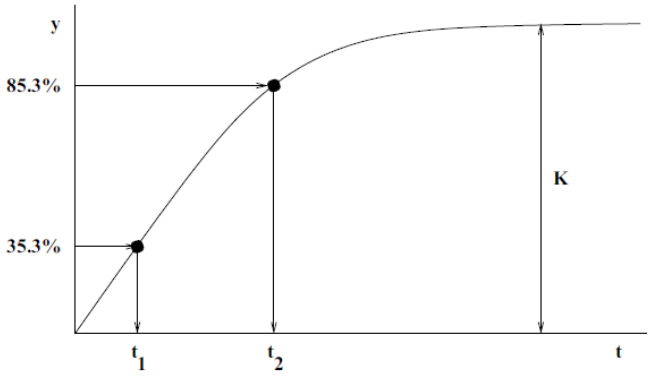

Figure 4.3- Calculation of t1, t2 and K

After computing the t1 and t2 times, the time delay (θ) and process time constant (T) can be obtained from the following equations:

$$\theta = 1.3t_1 - 0.29t_2$$
$$T = 0.67(t_2 - t_1) \quad \ldots (4.7)$$

The Dc motor model parameters are given below:

Armature Resistance, R_a = 1.15 Ω,

Armature Inductance, L_a = 0.0015 mH,

Viscous Friction Constant, B = 0.0088 Nm-sec/rad,

Moment of Inertia, J = 0.0021 kg m^2

Feedback $K_v = K_m$ =1.

The uncontrolled DC Motor Open Loop transfer function is shown below along with its time domain analysis and step response.

Transfer function:

$$\frac{1}{3.15e\text{-}006\ s^2 + 0.002428\ s + 0.01012}$$

Time Domain Analysis:

RiseTime	:	0.5245
SettlingTime	:	0.9349
Overshoot	:	0

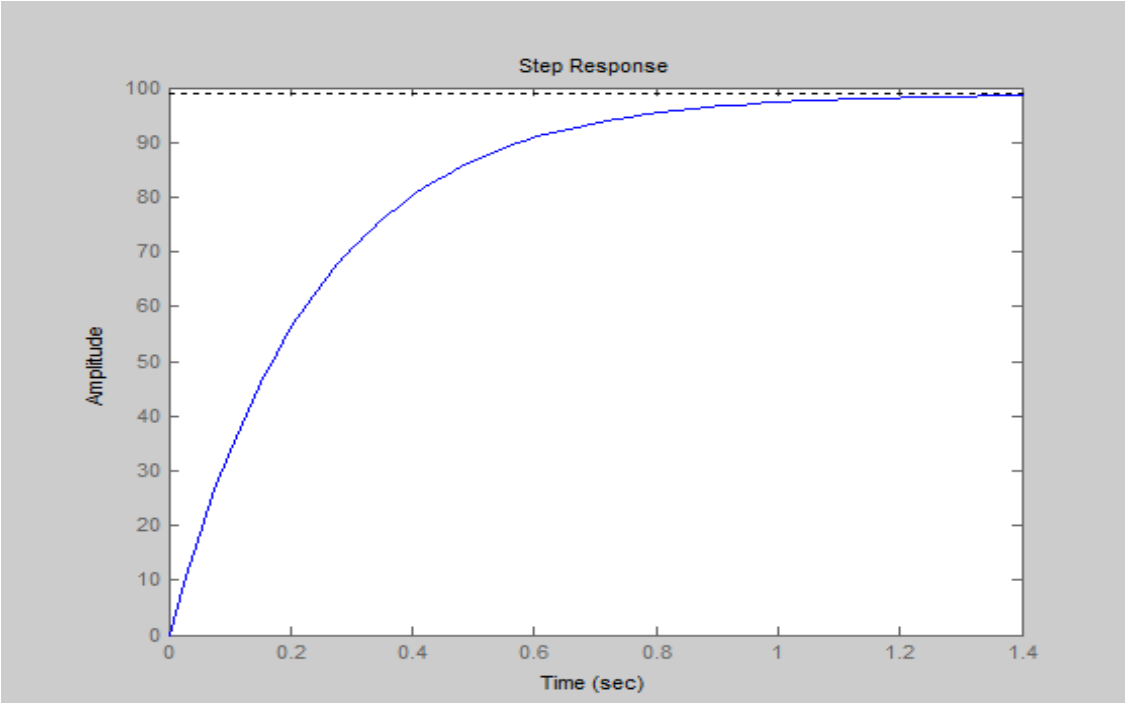

Figure 4.4- Open Loop Step Response of uncontrolled DC motor model

The transfer function can be represented by an equivalent first order time delay form. The time delay form of our DC motor open loop transfer function is given below.

First Order Time Delay Form:

$$\exp(-0.456 s) * \frac{98.52}{0.2345 s + 1}$$

From above first order time delay form we get:

Gain, K = 98.5218

Dead time or time delay, theta (θ) = 0.4560

Open loop process time constant, tau (T) = 0.2345

These parameters are used in various open loop PID tuning methods which are discussed below.

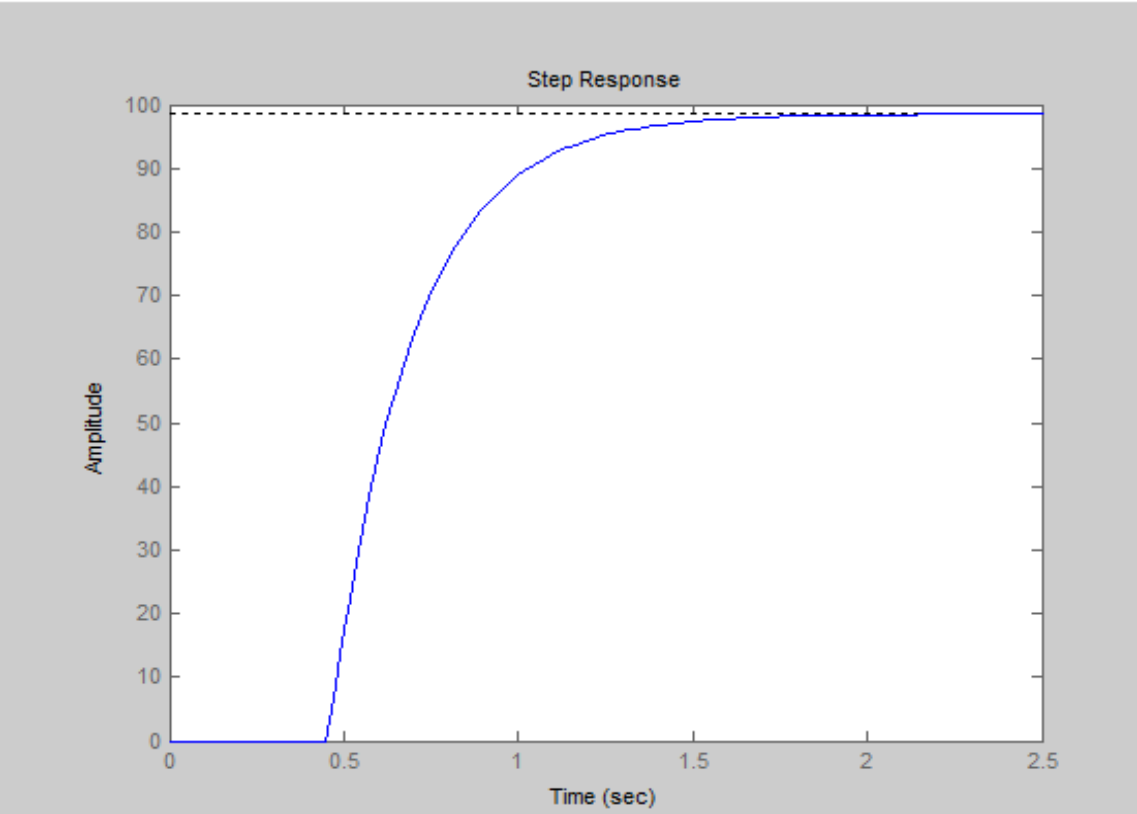

Figure 4.5- Time delay step response for uncontrolled DC motor model open loop transfer function

4.3.1.2- ZIEGLER-NICHOLS (ZN) OPEN LOOP TUNING METHOD

Ziegler-Nichols open loop tuning method is one of the most popular and most widely used classical tuning methods. The Ziegler-Nichols open-loop method is also referred to as a process reaction method, because it tests the open-loop reaction of the process to a change in the control variable output. The PID tuning parameters as a function of the open loop model parameters K, T and θ from equation (4.6) as derived by Ziegler-Nichols are given in table below:

Table 4.1- Ziegler-Nichols open loop PID tuning method

Controller		K_p	T_i	T_d
Ziegler-Nichols Method (Open loop)	P	$\frac{T}{K\theta}$	-	-
	PI	$\frac{0.9T}{K\theta}$	$\frac{\theta}{0.3}$	-
	PID	$\frac{1.2T}{K\theta}$	2θ	0.5θ

The step response of DC motor model (speed controller) controlled by Ziegler-Nichols open loop PID controller is shown below:

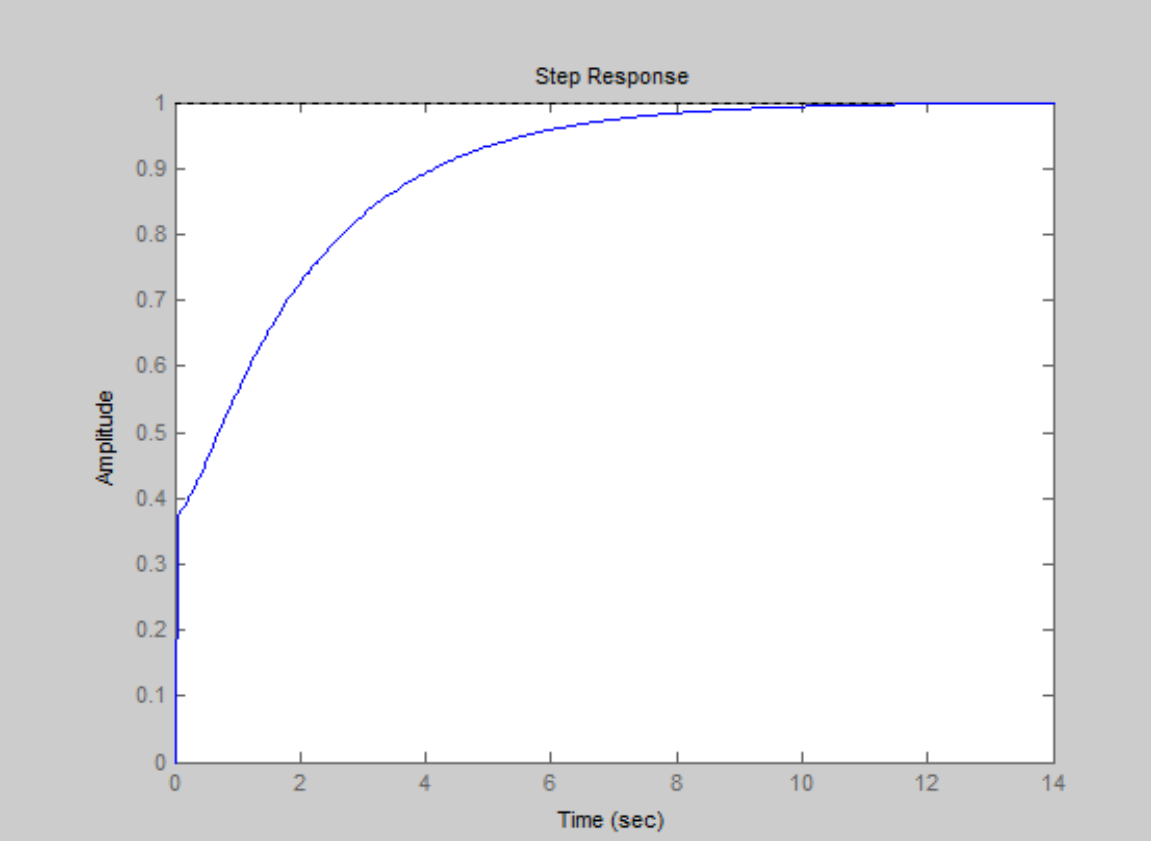

Figure 4.6(a)- Step Response of ZN open loop PID tuning method.

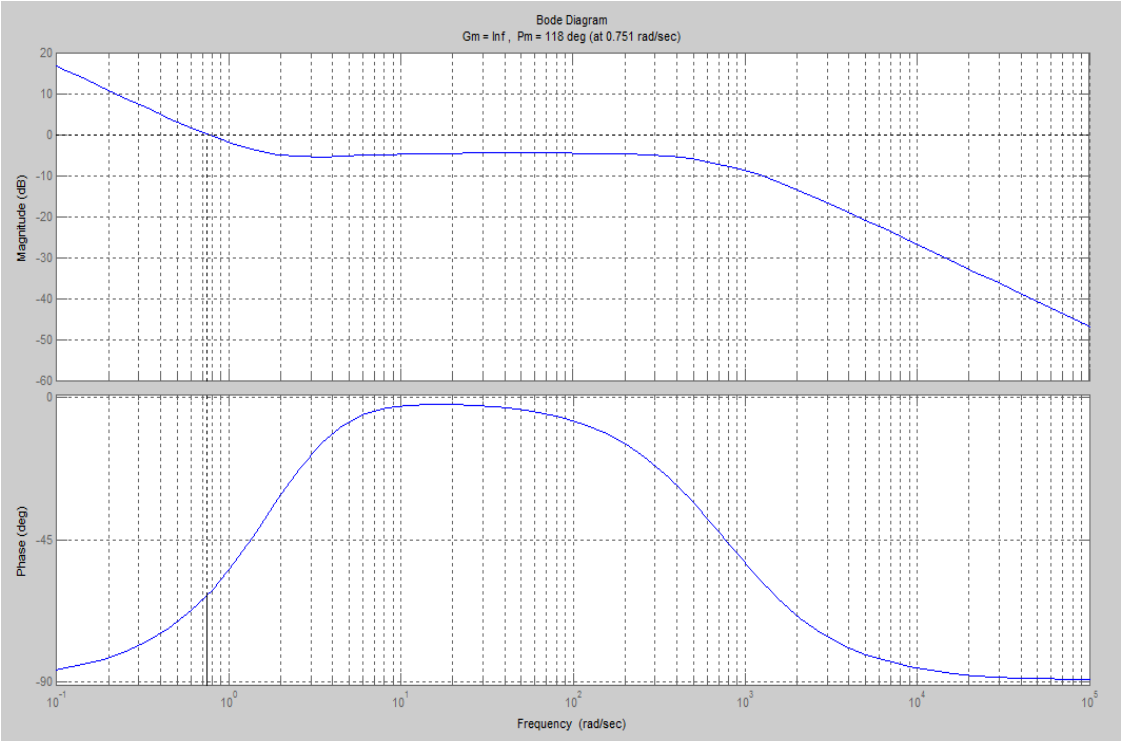

Figure 4.6(b) - Bode plot for ZN open loop PID tuning method.

4.3.1.3- COHEN-COON TUNING METHOD

Cohen-Coon tuning procedure uses the parameters obtained from open loop transfer function as in ZN method. This method is only used for first-order models with time delay, due to the fact that the controller does not instantaneously respond to the disturbance (the step disturbance is progressive instead of instantaneous). The Cohen-Coon method is classified as an 'offline' method for tuning, meaning that a step change can be introduced to the input once it is at steady-state. Then the output can be measured based on the time constant and the time delay and this response can be used to evaluate the initial control parameters.

The PID tuning parameters as a function of the open loop model parameters K, T and θ from equation (4.6) as derived by Cohen-Coon are:

Table 4.2- Cohen-Coon PID tuning method

Controller		K_p	T_i	T_d
Cohen-Coon Method (Open loop)	P	$\frac{1}{K}\frac{T}{\theta}\left(1+\frac{\theta}{3T}\right)$	-	-
	PI	$\frac{1}{K}\frac{T}{\theta}\left(0.9+\frac{\theta}{12T}\right)$	$\theta\left(\frac{30+3\theta/T}{9+20\theta/T}\right)$	-
	PID	$\frac{1}{K}\frac{T}{\theta}\left(\frac{4}{3}+\frac{\theta}{4T}\right)$	$\theta\left(\frac{32+6\theta/T}{13+8\theta/T}\right)$	$\theta\left(\frac{4}{11+2\theta/T}\right)$

The figure (4.7) shown below shows step response of DC motor model for speed control by Cohen- Coon method tuned PID controller. It is observed that Cohen-Coon method shows a smoother step response as compared to that shown by Ziegler-Nichols method for PID controller.

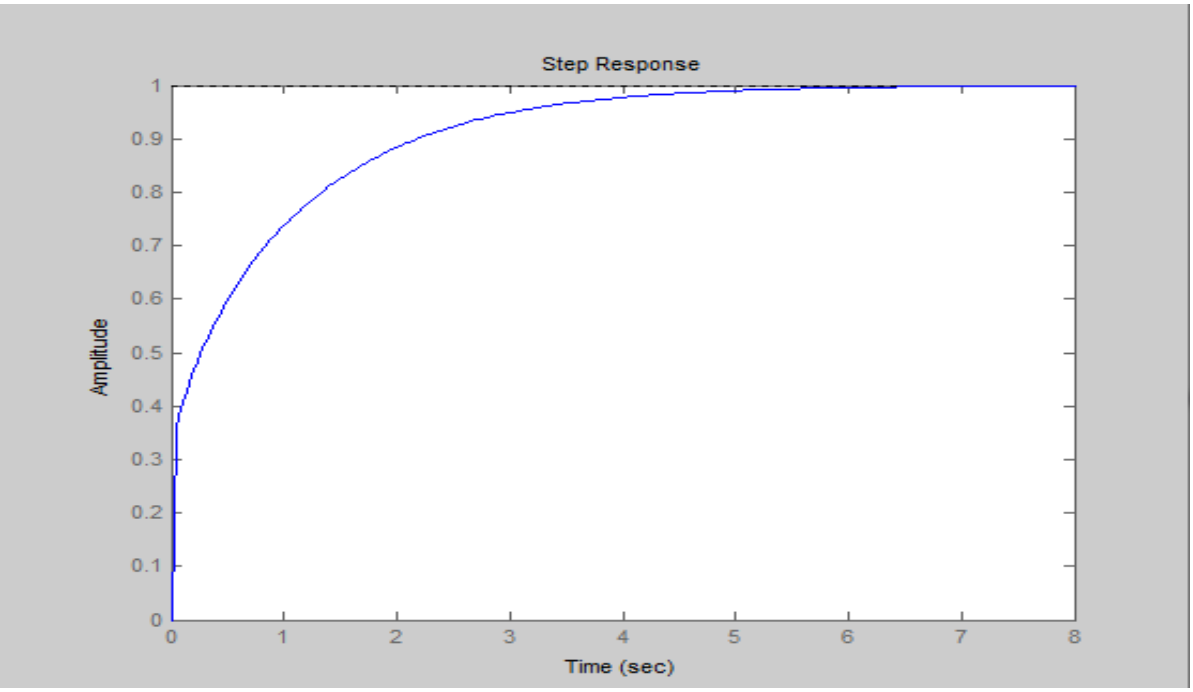

Figure 4.7(a) - Step response of Cohen- Coon PID tuning method

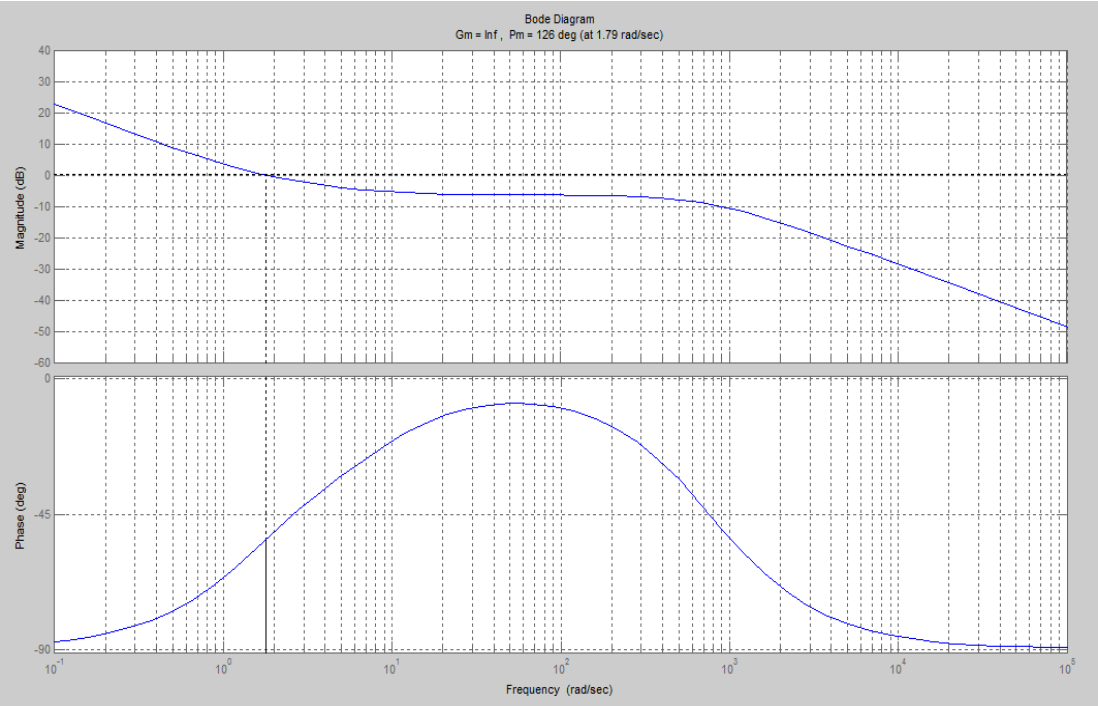

Figure 4.7(b) - Bode plot for Cohen- Coon PID tuning method

4.3.1.4- APPROXIMATE M-CONSTRAINED INTEGRAL GAIN OPTIMIZATION (AMIGO) TUNING METHOD

The PID tuning parameters as a function of the open loop model parameters K, T and θ from equation (4.6) as derived by Approximate M-constrained Integral Gain Optimization (AMIGO):

Table 4.3- AMIGO PID tuning method

Controller		K_p	T_i	T_d
AMIGO Method (Open loop)	PID	$\frac{1}{K}\left(0.2+0.45\frac{T}{\theta}\right)$	$\theta\left(\frac{0.4\theta+0.8T}{\theta+0.1T}\right)$	$\frac{0.5\theta T}{0.3\theta+T}$

$\tau = \frac{\theta}{\theta + T}$ represents the relative dead time.

The tuning rule above describes the controller gain K_p well for process with τ > 0.3. For small τ, the controller gain is well fitted to processes G_p, but the AMIGO rule underestimates the gain for other processes.

The integral time T_i is well described by the tuning rule above for τ > 0.2. For small τ, the integral time is well fitted to processes G_p, but the AMIGO rule overestimates it for other processes.

The tuning rule above describes the derivative time T_d well for process with τ > 0.5. In the range 0.3 < τ < 0.5, the derivative time can be up to a factor of 2 larger than the value given by the AMIGO rule. Figure (4.8) shows step response for speed control of DC motor model with PID controller using AMIGO tuning method.

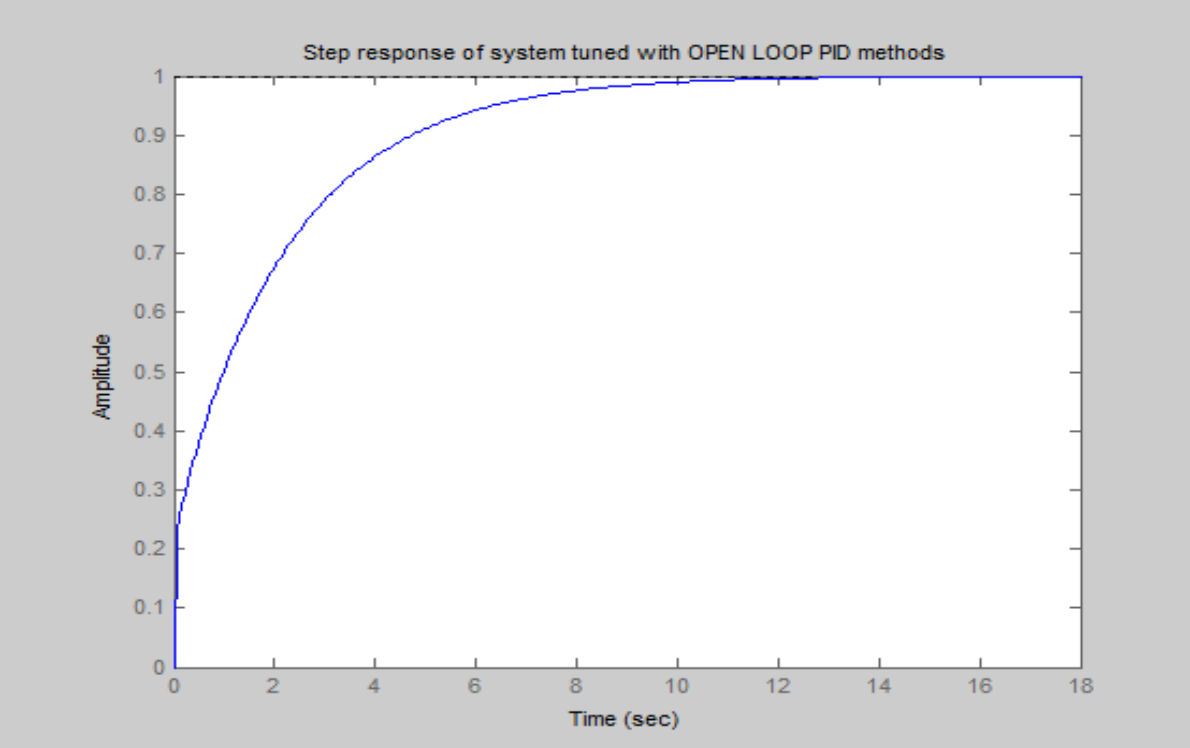

Figure 4.8(a) - Step response of AMIGO method based PID tuning.

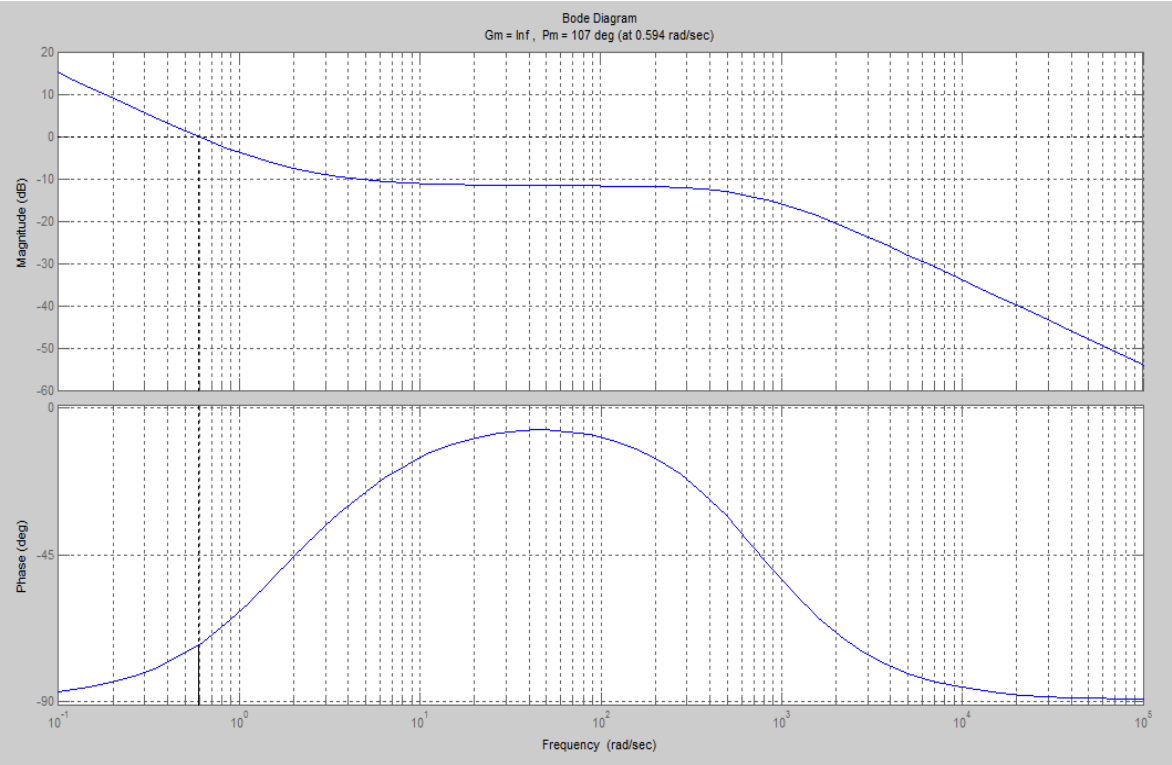

Figure 4.8(b)- Bode plot for AMIGO method based PID tuning.

4.3.1.5-INTERNAL MODEL CONTROL (IMC) TUNING METHOD

IMC strategy proposed by Rivera *et al.*, needs selection of the tuning parameter λ for PID design that is almost equivalent to the closed-loop time constant.

The PID tuning parameters as a function of the open loop model parameters K, T and θ from equation (4.6) as derived by Internal Model Control (IMC):

Table 4.4- IMC PID tuning method

Controller		K_p	T_i	T_d
IMC Method (Open loop)	PI	$\frac{2T+\theta}{2\lambda}$	$T+\frac{\theta}{2}$	-
	PID	$\frac{2T+\theta}{2(\lambda+\theta)}$	$T+\frac{\theta}{2}$	$\frac{T\theta}{2T+\theta}$

Where

$\lambda \geq 1.7\theta$ for PI controller and

$\lambda \geq 0.25\theta$ for PID controller

Figure (4.9) shows the step response for speed control of DC motor model with PID controller tuned by IMC method.

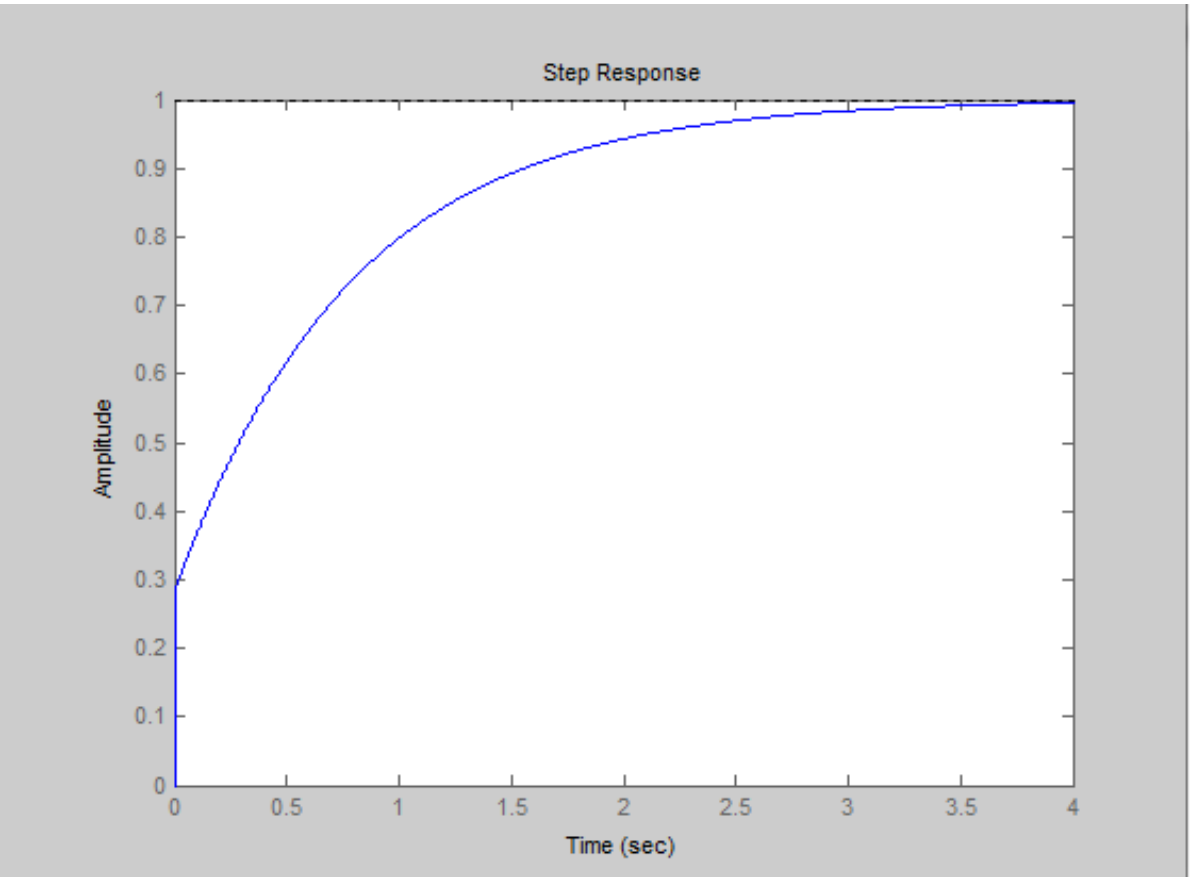

Figure 4.9(a)- Step response of IMC PID tuning method

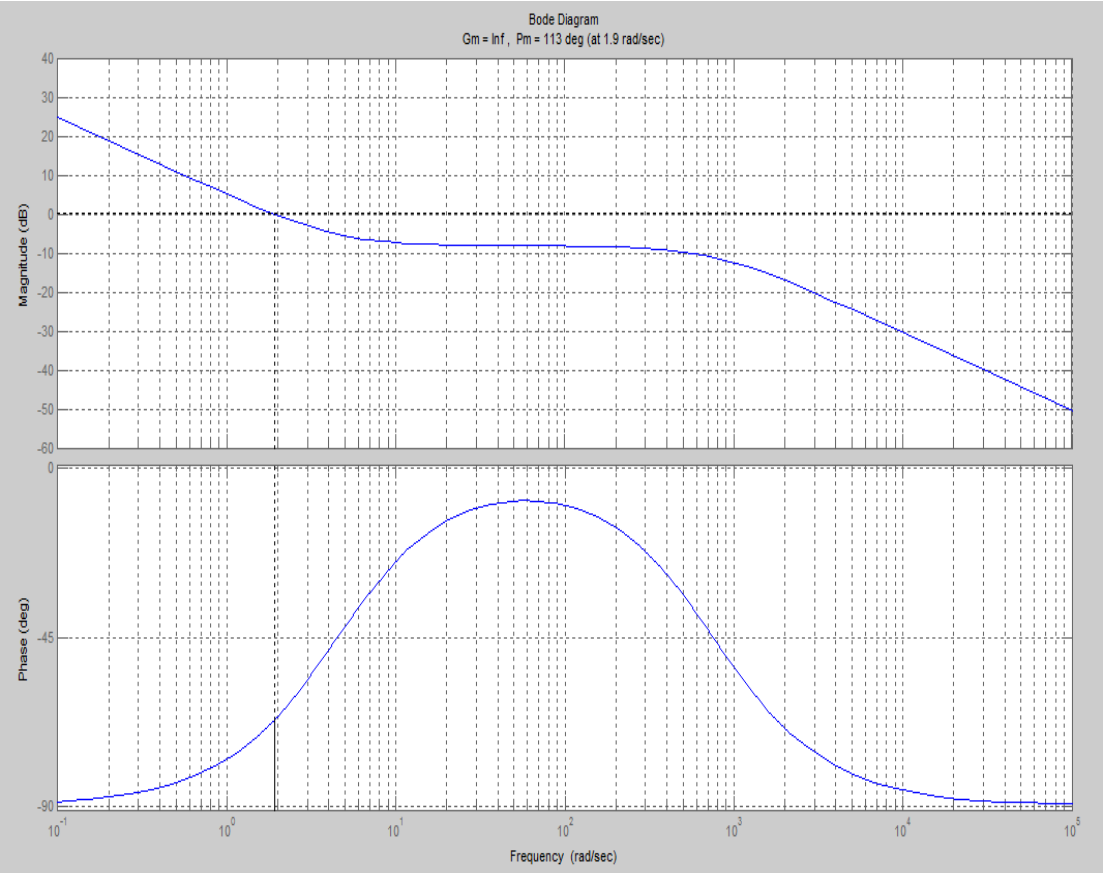

Figure 4.9(b)- Bode plot for IMC PID tuning method

4.3.1.6-CHIEN-HRONES-RESWICK (CHR) STEP RESPONSE TUNING METHOD

In process industry controller parameters are often tuned according to CHR recommendations. They are based on time parameters of open loop step reference change response. Chien, Hrones and Reswick also gave recommendation for the choice of the type of the controller.

Table given below shows Chien-Hrones-Reswick method for set point (0% overshoot)

Table 4.5- CHR PID tuning method for set point (0% overshoot)

Controller		K_p	T_i	T_d
Cohen-Coon Method (Open loop)	P	$\frac{0.3T}{K\theta}$	-	-
	PI	$\frac{0.35T}{K\theta}$	$1.2T$	-
	PID	$\frac{0.6T}{K\theta}$	T	0.5θ

Table below shows Chien-Hrones-Reswick method for disturbance (0% overshoot)

Table 4.6- CHR PID tuning method for disturbance (0% overshoot)

Controller		K_p	T_i	T_d
Cohen-Coon Method (Open loop)	P	$\frac{0.3T}{K\theta}$	-	-
	PI	$\frac{0.6T}{K\theta}$	4θ	-
	PID	$\frac{0.95T}{K\theta}$	2.4θ	0.42θ

For our DC motor model, only the set point (0% overshoot) method is considered and its step response is shown in figure (4.10) below.

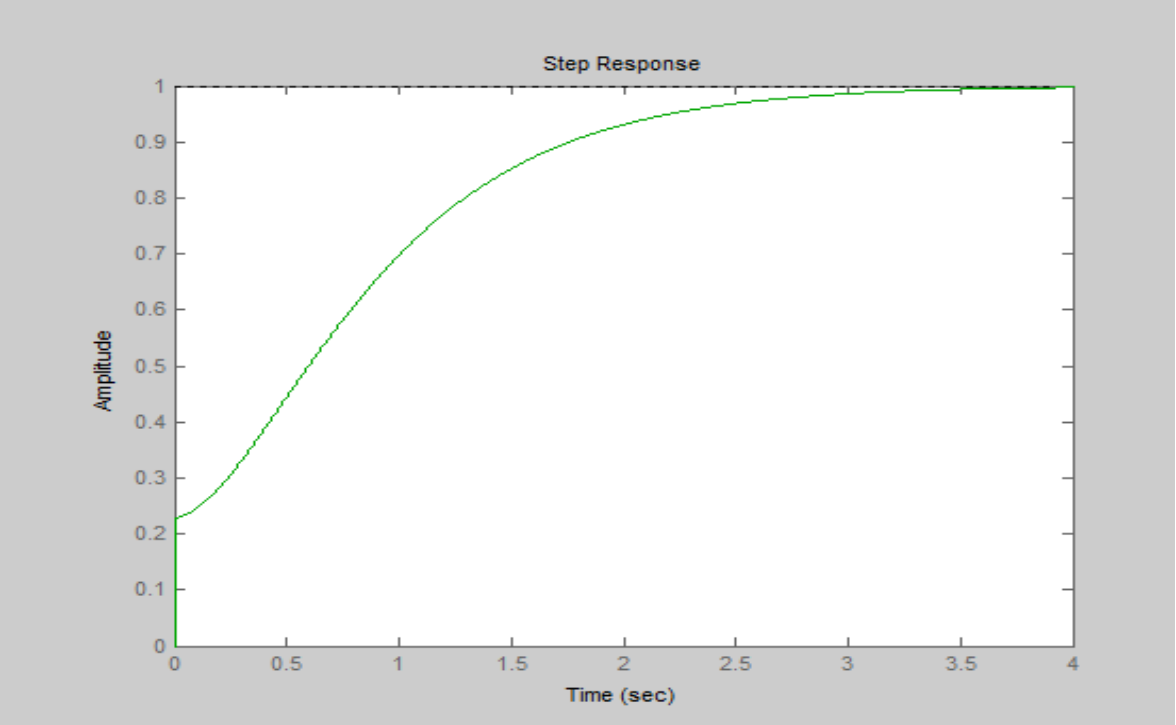

Figure 4.10(a) - Step response of CHR PID tuning method (For set point 0% overshoot)

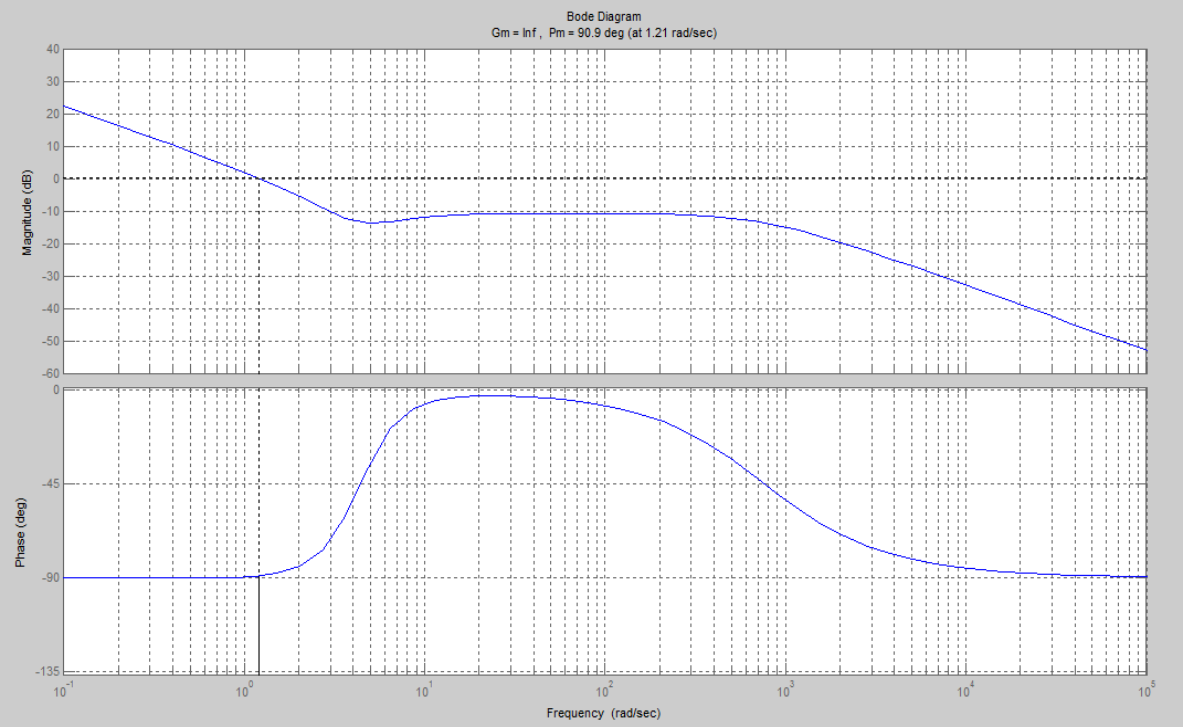

Figure 4.10(b) – Bode plot for CHR PID tuning method (For set point 0% overshoot)

4.3.1.7- RESULTS

Table 4.7- Open loop PID tuning method time domain analysis

Method	PID Parameters			Time Domain Analysis		
	Kp	Ti	Td	Rise Time	Settling Time	Overshoot
Ziegler-Nichols Open Loop Method	0.0063	0.9120	0.2280	4.1280	7.5531	0
Cohen-Coon	0.0095	0.6973	0.1225	2.1688	4.1556	0
AMIGO	0.0044	0.7717	0.1440	4.7027	8.4658	0
IMC(Lambda=0.1140)	0.0082	0.4625	0.1156	1.5591	2.8330	0
Chien- Hrones- Reswick	0.0031	0.2345	0.2280	1.7625	2.7839	0

Table 4.8- Open loop PID tuning method frequency domain & robustness analysis

Method	Frequency Domain Analysis		Robustness Analysis	
	Gain margin	Phase margin	Modulus margin (in dB)	Delay margin (In sec)
Ziegler-Nichols Open Loop Method	inf	117.5800	-25.6538	2.7339
Cohen-Coon	inf	126.4979	-33.2029	1.2358
AMIGO	inf	107.3894	-66.7191	3.1563
IMC(Lambda=0.1140)	inf	112.9561	-42.2833	1.0351
Chien- Hrones- Reswick	inf	90.9162	-58.8578	1.3111

COMPARISON OF OPEN LOOP TUNING METHODS

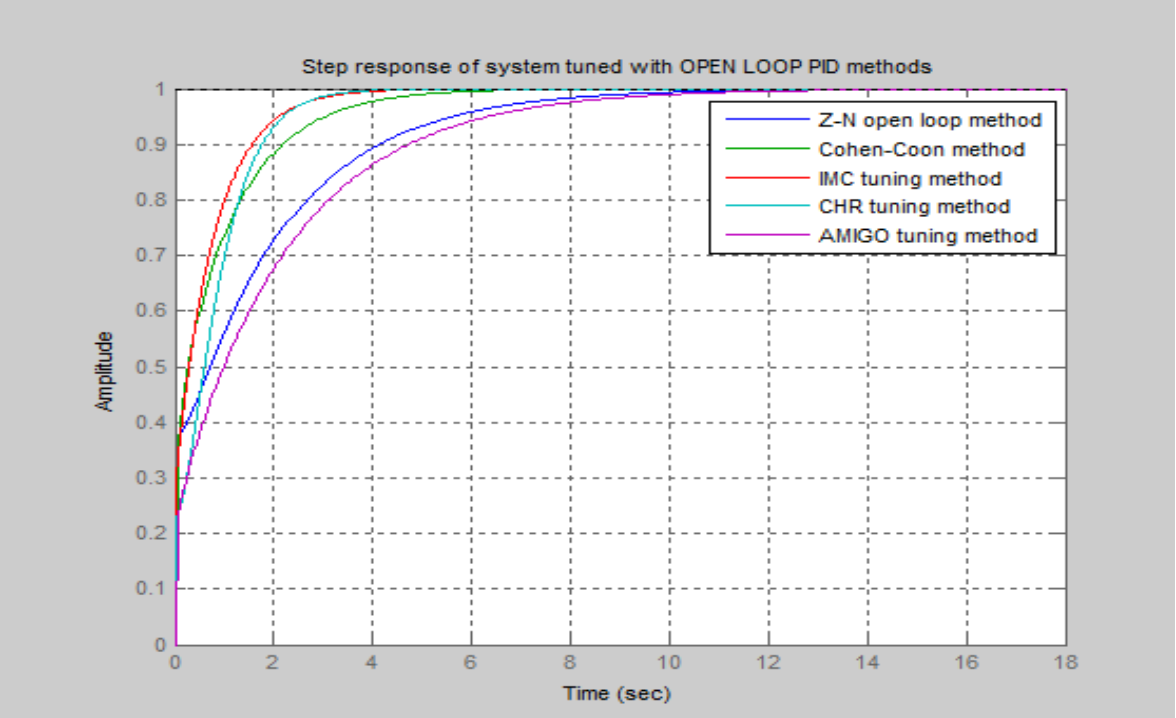

Figure 4.11- Comparison of OPEN LOOP PID tuning methods

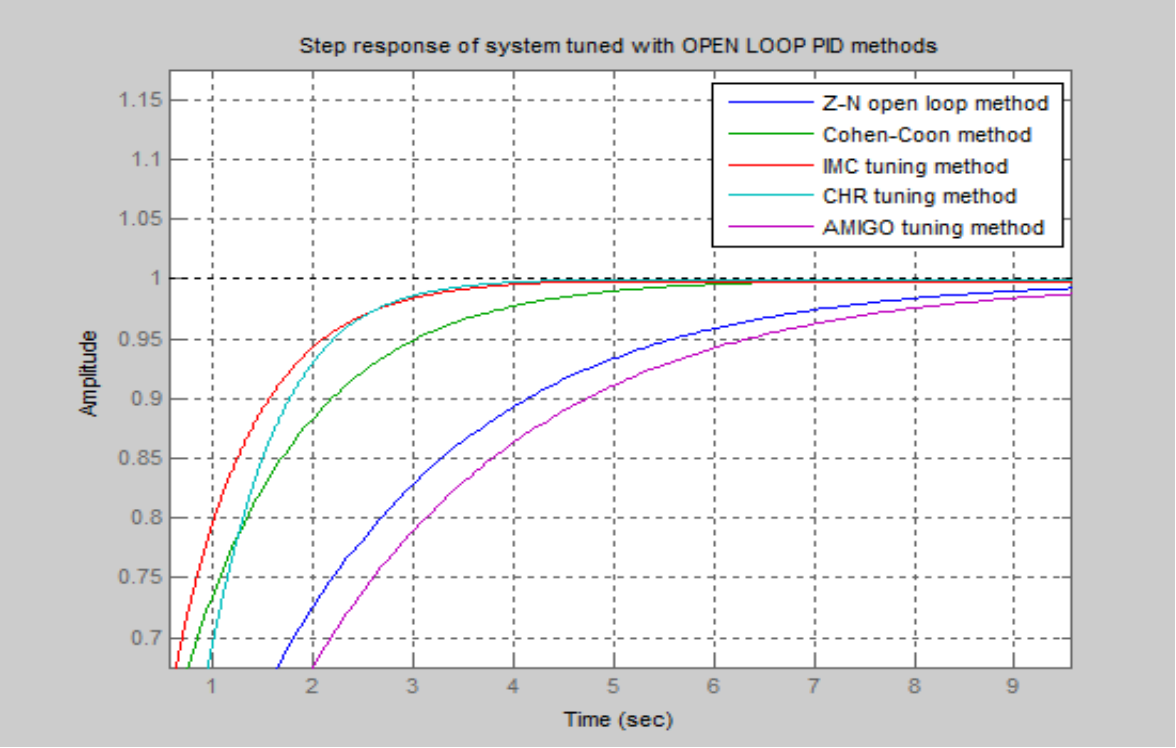

Figure 4.12- Comparison of OPEN LOOP PID tuning methods (1Xzoom)

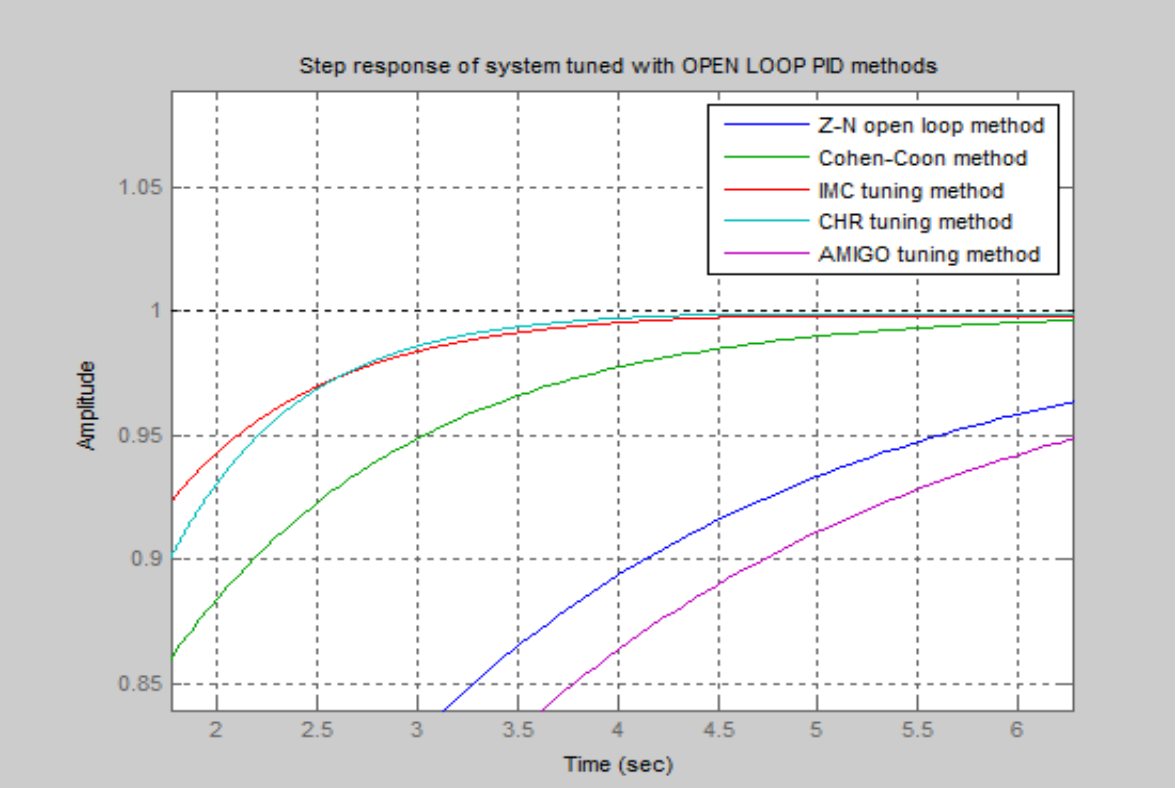

Figure 4.13- Comparison of OPEN LOOP PID tuning methods (2Xzoom)

4.3.1.8- CONCLUSION

For DC motor model, the Open loop tuning methods give 0 overshoot as shown in table (4.7). The IMC tuning method and CHR step response method give the best value of rise time and settling time for DC motor model as given in table (4.7). Figure (4.13) shows clearly that out of all the Open loop tuning methods mentioned above, IMC and CHR tuning method gives best performance for DC motor model. AMIGO tuning method shows the least satisfactory performance for DC motor model.

The frequency domain analysis and robustness analysis are not achievable (only delay margin is within range) within required range for Open loop tuning methods for selected DC motor model as visible in table (4.8).

4.3.2- CLOSED LOOP PID TUNING METHODS

Closed loop step response for both transfer function and first order time delay form for uncontrolled DC motor model (speed control) is given below along with time domain specifications:

Transfer function:

$$\frac{1}{3.15e\text{-}006\ s^2 + 0.002428\ s + 1.01}$$

Time domain analysisi is given below:

RiseTime: 0.0037

SettlingTime: 0.0106

Overshoot: 5.3875

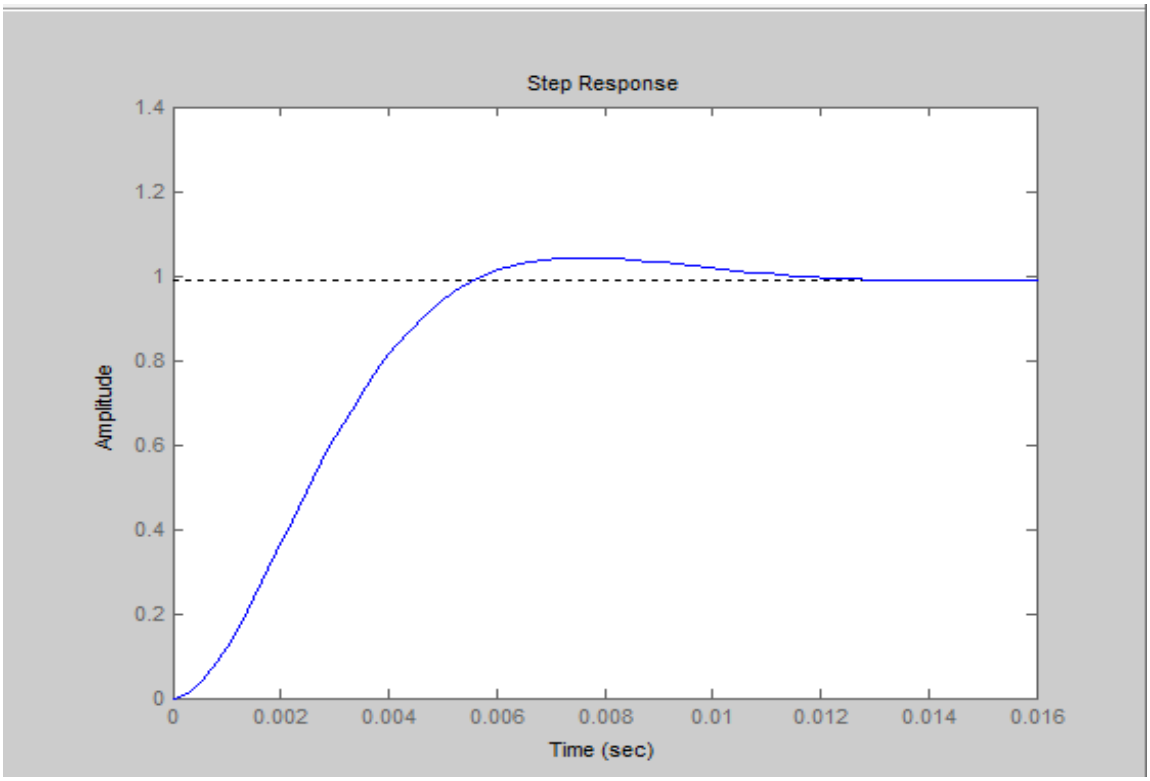

Figure 4.14- Step response of uncontrolled Closed loop transfer function

First Order Time Delay Form:

$$\exp(-0.00275*s) * \frac{0.9871}{0.001408\,s + 1}$$

Time domain analysis for time delay form is given below:

RiseTime: 0.0031

SettlingTime: 0.0083

Overshoot: 0

The parameters for first order time delay form for DC motor model are:

Gain, K = 0.9871

Theta, θ = 0.0028

Tau, T = 0.0014

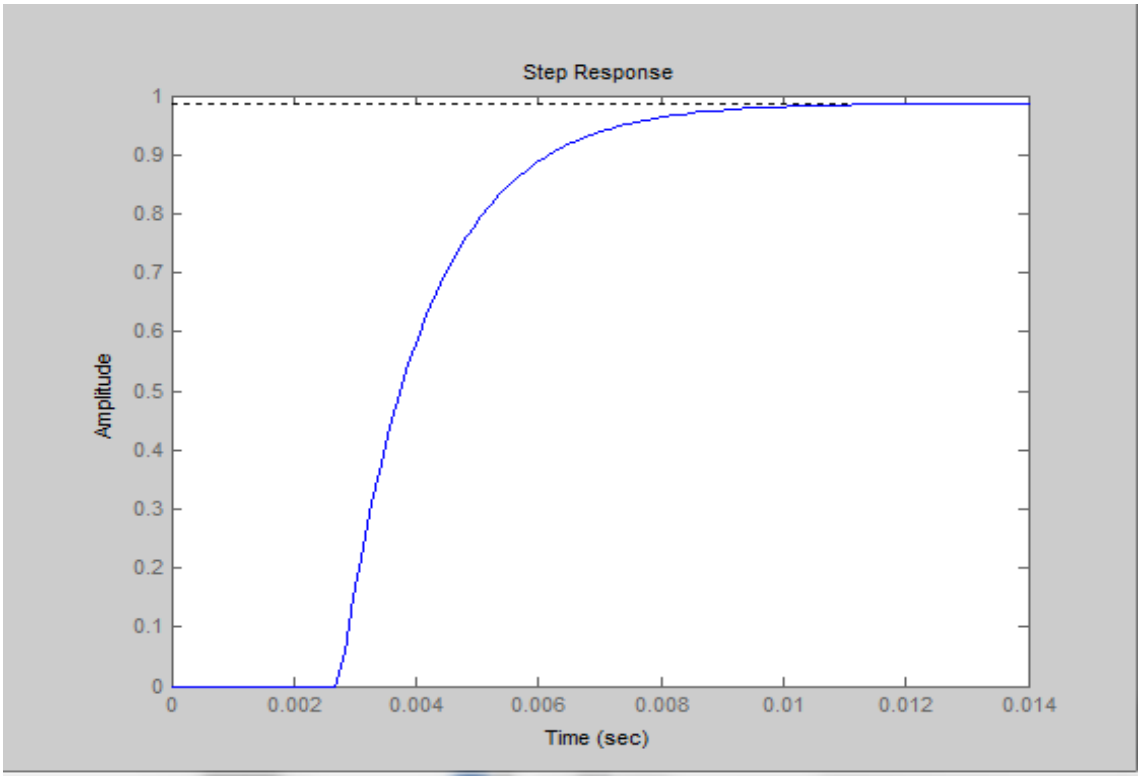

Figure 4.15- Step response for uncontrolled closed loop first order time delay form

The Frequency domain analysis for first order time delay form gives:

Gain margin, Gm =1.5566

Phase margin, Pm = Infinite

The gain margin Gm = ∞ and phase margin Pm = -180 for selected DC motor model as the transfer function is of second order.

4.3.2.1- ZIEGLER-NICHOLS (ZN) CLOSED LOOP TUNING METHOD

The Ziegler-Nichols continuous cycling method is one of the best known closed loop tuning strategies. The controller gain is gradually increased (or decreased) until the process output continuously cycles after a small step change or disturbance. At this point, the controller gain is selected as the ultimate gain, K_u, and the observed period of oscillation is the ultimate period, P_u. Ziegler and Nichols originally suggested PID tuning constants as a function of the ultimate gain and ultimate period.

Table 4.9- Ziegler-Nichols closed loop PID tuning method

Controller		K_p	T_i	T_d
Ziegler-Nichols Method (Closed loop)	P	$0.5K_u$	-	-
	PI	$0.45K_u$	$P_u/1.2$	-
	PID	$0.6K_u$	$P_u/2$	$P_u/8$

For servo speed control of DC motor model, the step response obtained by ZN closed loop tuning for PID controller is shown in figure (4.16 a).

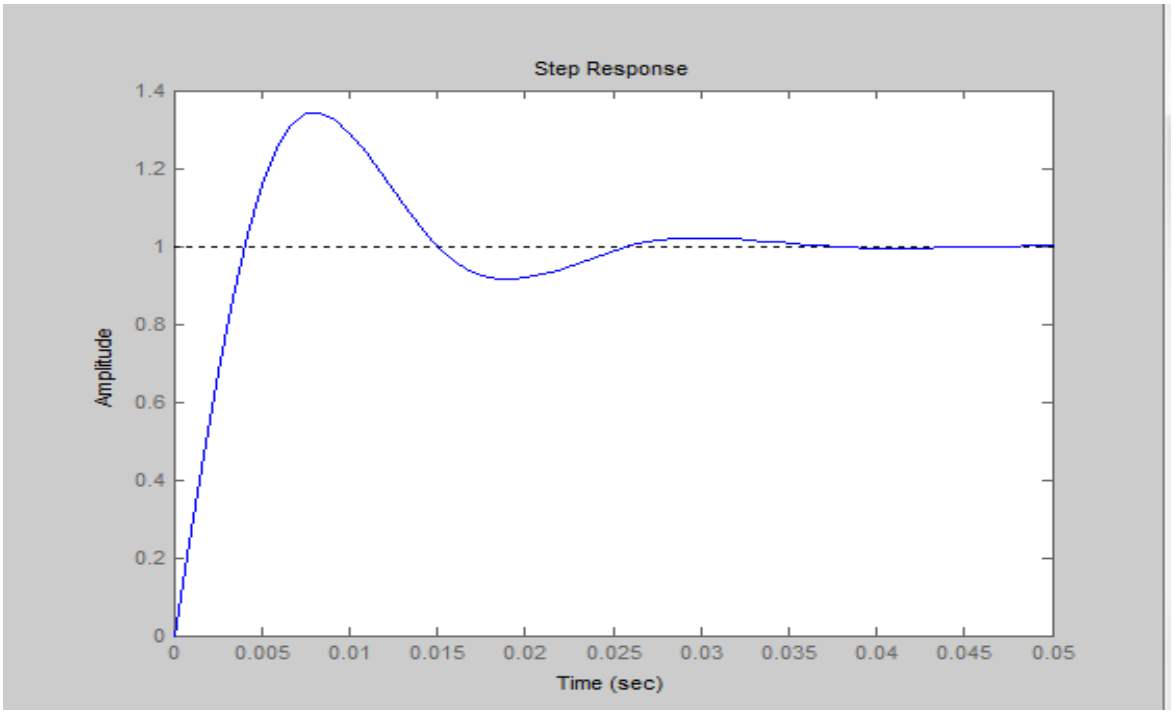

Figure 4.16(a) - ZN closed loop step response

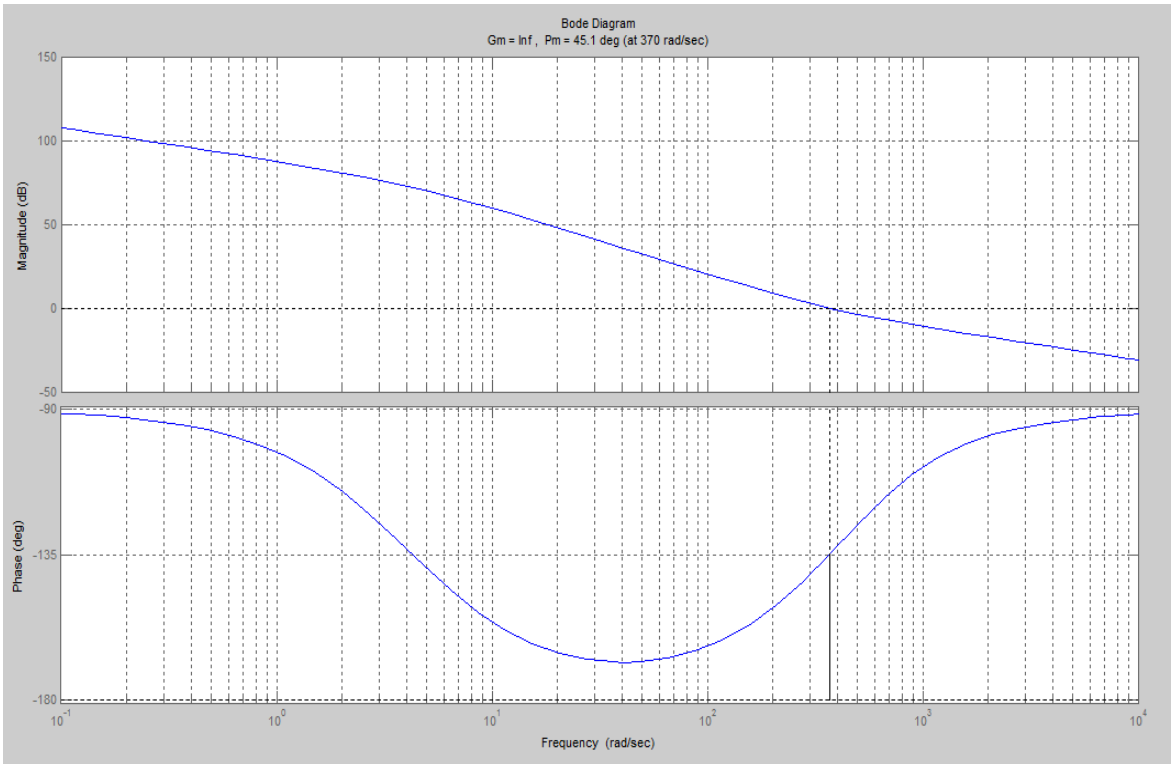

Figure 4.16(b) - ZN closed loop bode plot

4.3.2.2- MODIFIED ZIEGLER-NICHOLS TUNING METHOD

This is a modified approach of Ziegler-Nichols closed loop PID tuning method. The table for modified Ziegler- Nichols tuning method is given below:

Table 4.10- Modified Ziegler-Nichols PID Tuning method

Controller		K_p	T_i	T_d
Ziegler-Nichols Method (Closed loop) PID Controller	No Overshoot	$0.2K_u$	$P_u / 2$	$P_u / 2$
	Some Overshoot	$0.33K_u$	$P_u / 2$	$P_u / 3$

Figure (4.17 a) & (4.18 a) shows step response for above two cases of modified ZN method for PID controller in DC motor model. Figure (4.19) shows their comparison.

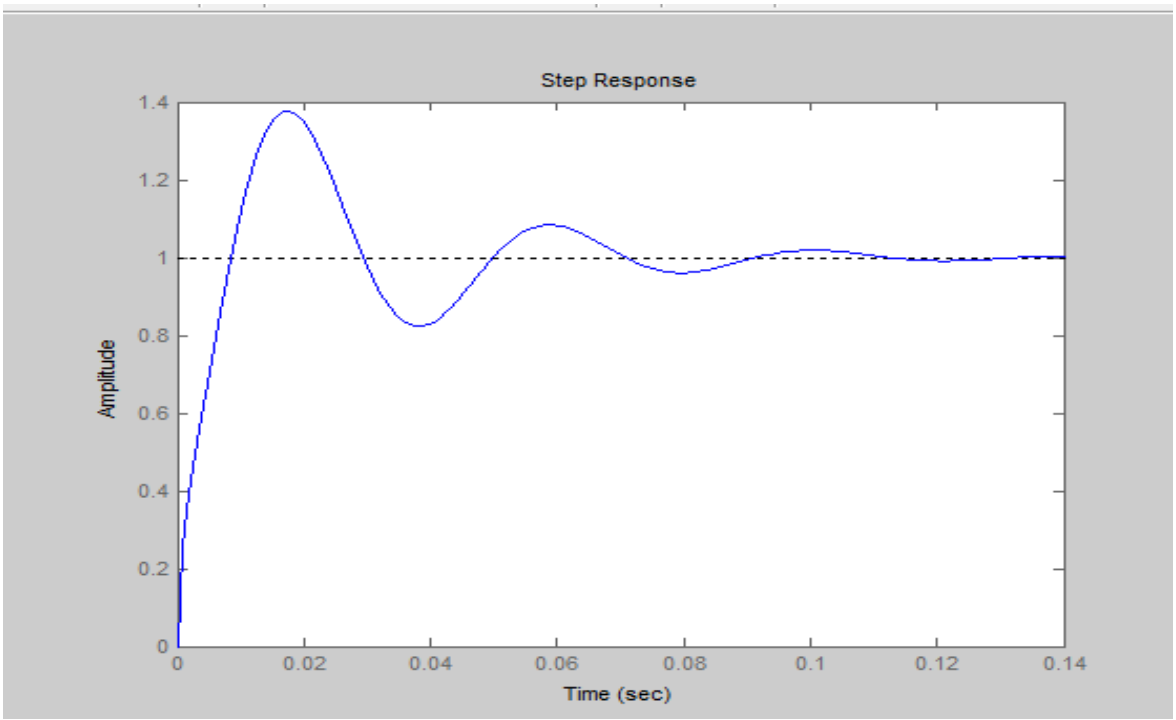

Figure 4.17(a) - Modified ZN method step response for no overshoot

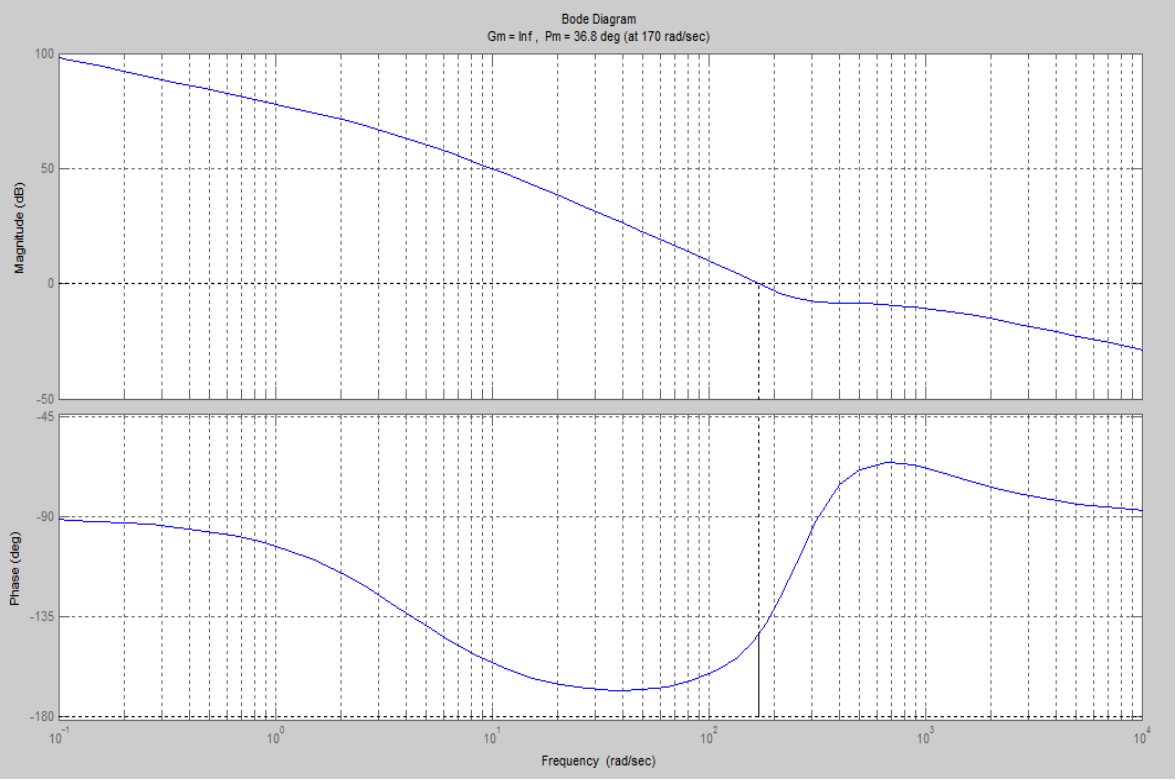

Figure 4.17(b) - Modified ZN method bode plot for no overshoot

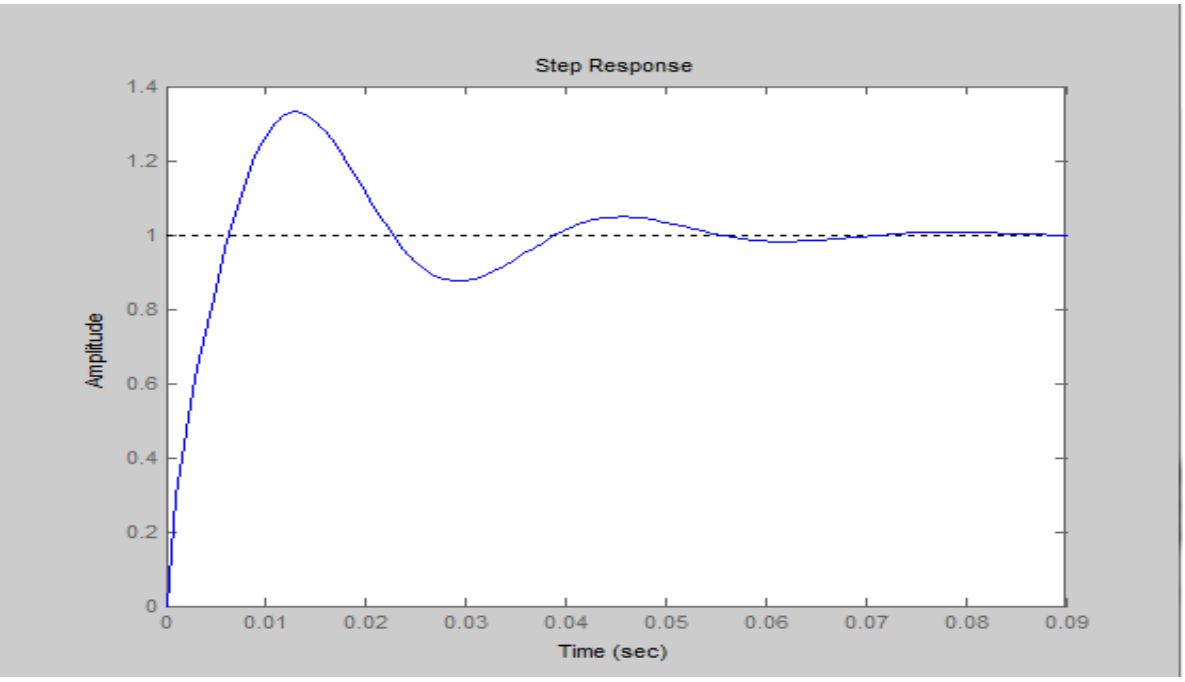

Figure 4.18(a) - Modified ZN method step response for some overshoot

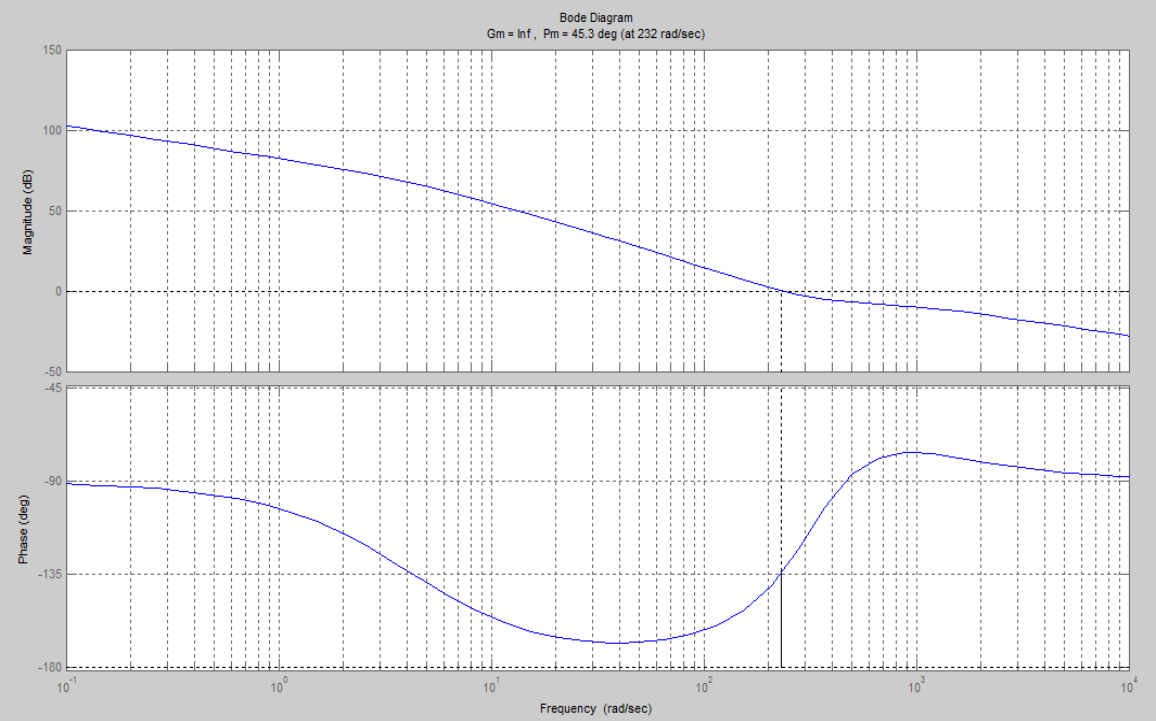

Figure 4.18(b) - Modified ZN method bode plot for some overshoot

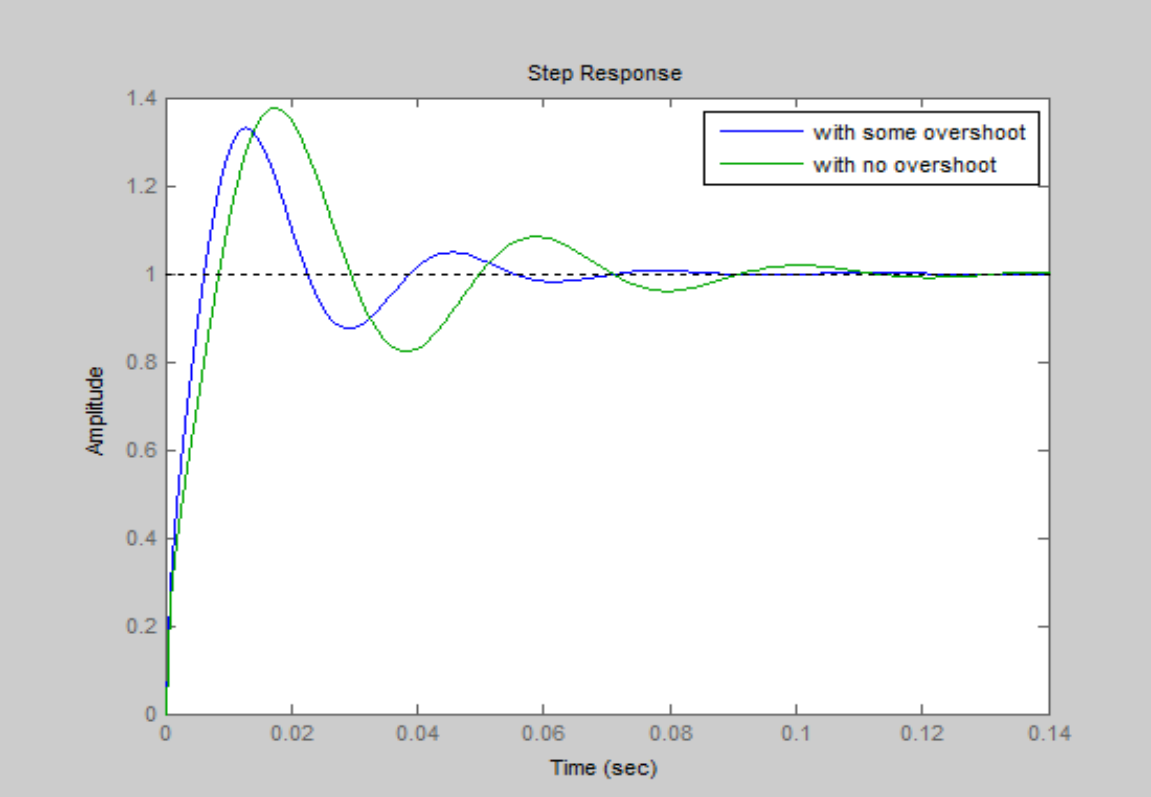

Figure 4.19- Comparison of no overshoot and some overshoot methods response.

4.3.2.3- TYREUS-LUYBEN TUNING METHOD

The Ziegler-Nichols settings give much more underdamped behavior (high peak in the closed-loop log modulus curve) for a slightly larger closed-loop time constant (smaller resonant or breakpoint frequency). Tyreus-Luyben PID tuning method overcomes this drawback of Ziegler Nichols method.

The table of Tyreus-Luyben tuning method is given below:

Table 4.11- Tyreus-Luyben PID tuning method

Controller		K_p	T_i	T_d
Tyreus-Luyben Method (Closed loop)	PI	$0.31K_u$	$2.2P_u$	-
	PID	$0.45K_u$	$2.2P_u$	$P_u / 6.3$

Figure (4.20 a) shows step response of Tyreus- Luyben method for PID controller in DC motor model.

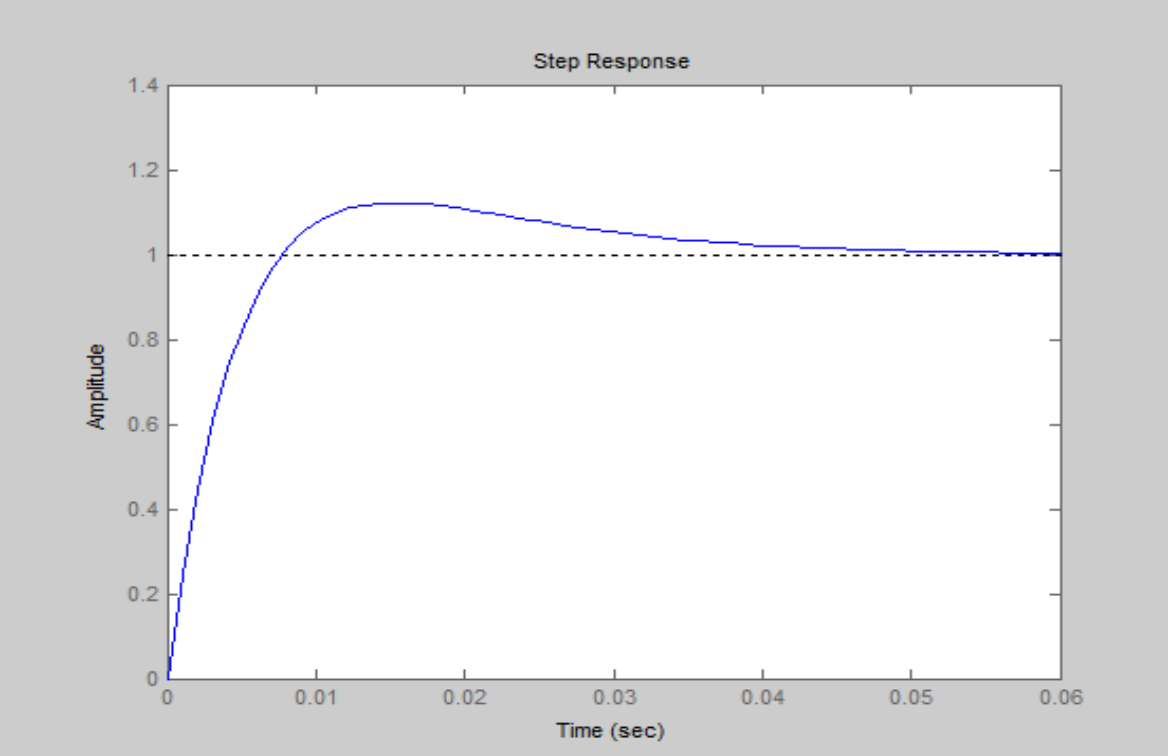

Figure 4.20(a) - Tyreus- Luyben method for PID controller step response

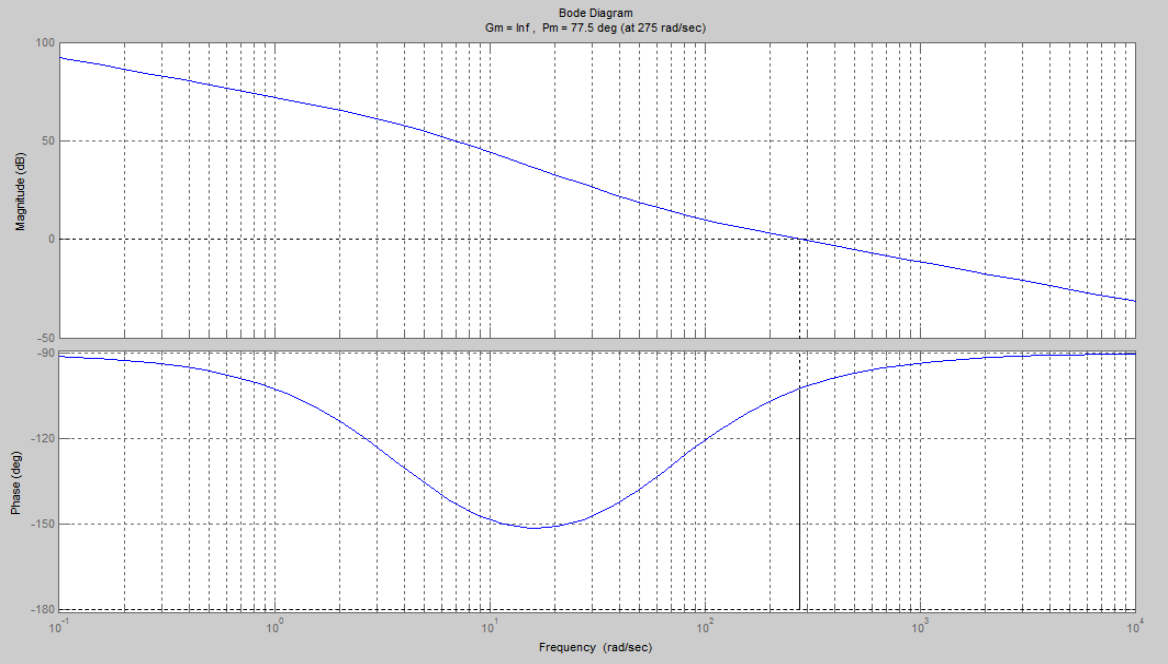

Figure 4.20(b) - Tyreus- Luyben method for PID controller Bode plot

4.3.2.4- CHIEN-HRONES-RESWICK PI TUNING METHOD

The table for Chien-Hrones-Reswick PI tuning method is given in the table below:-

Table 4.12- CHR frequency PI tuning method

Controller		K_p	T_i	T_d
Chien-Hrones-Reswick Method	PI	$0.47K_u$	1	-

The step response of CHR method for PI controller in DC motor model is shown below in figure (4.21 a).

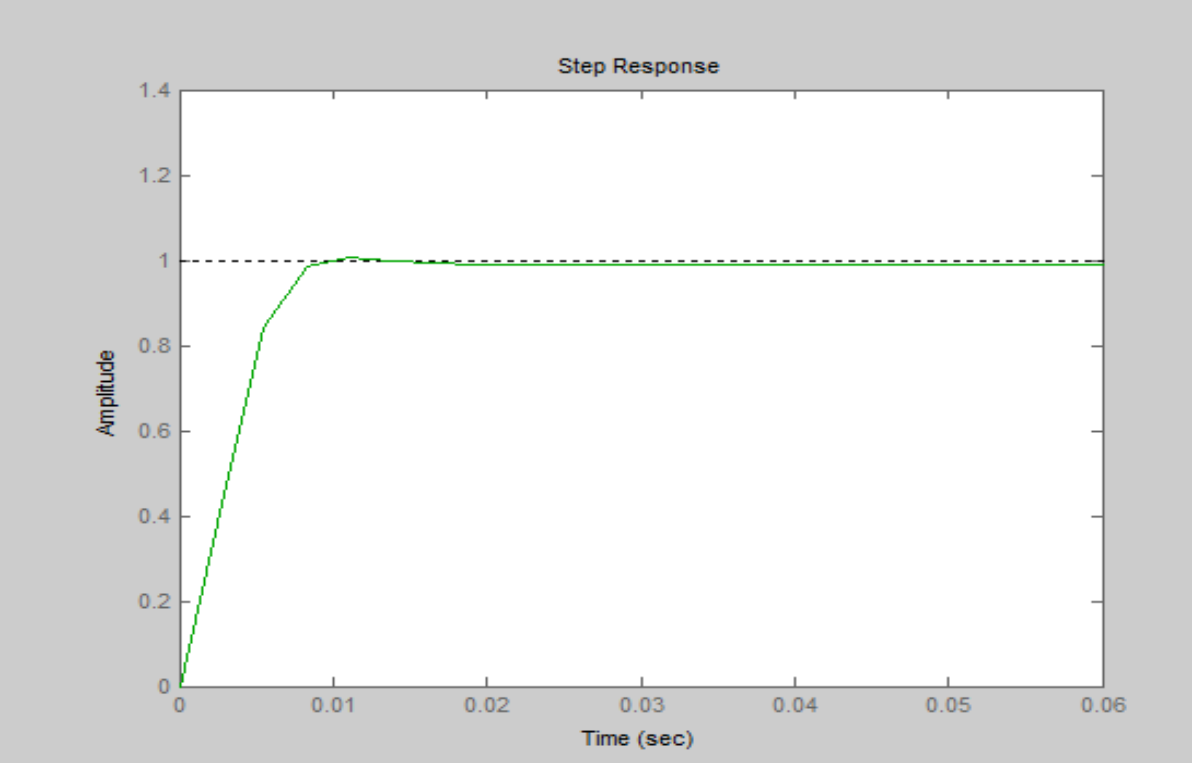

Figure 4.21(a) - CHR method PI controller step response

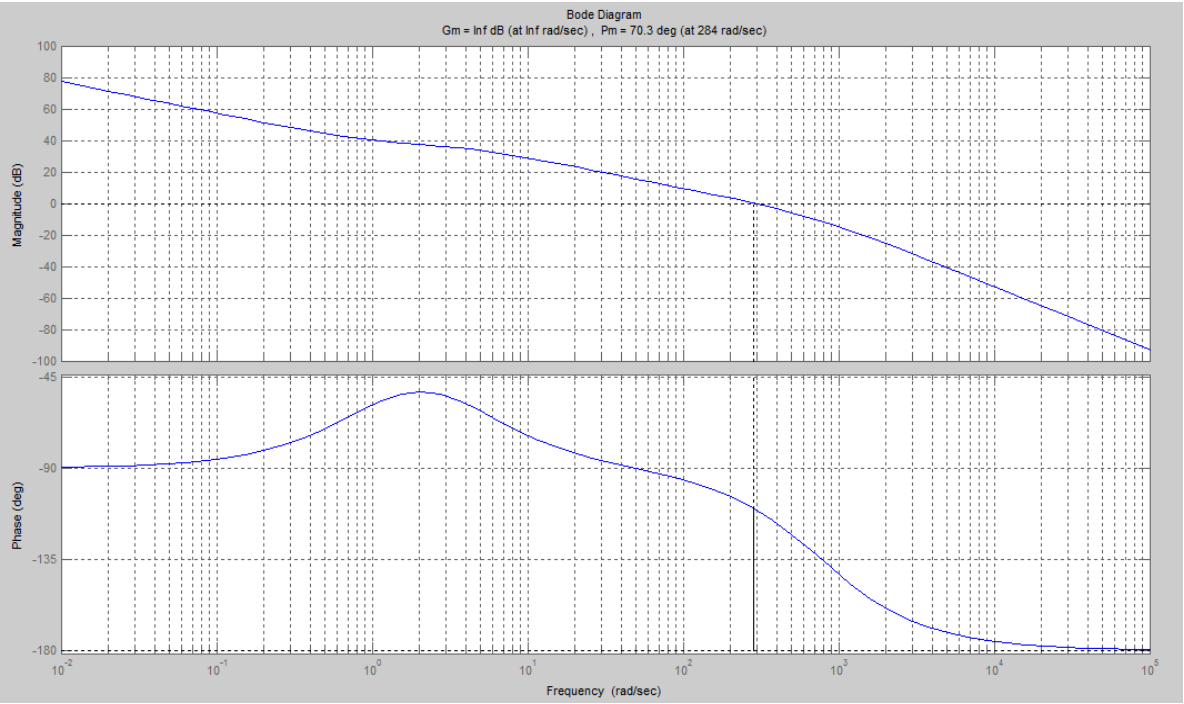

Figure 4.21(b) - CHR method PI controller bode plot

4.3.2.5- ASTROM-HAGGLUND PI TUNING METHOD

The Astrom-Hagglund PI tuning method table is given below:

Table 4.13- The Astrom-Hagglund PI tuning method

Controller		K_p	T_i	T_d
Astrom-Hagglund Method	PI	$0.32K_u$	0.94	-

Astrom- Hagglund Step response for PI controller is shown in figure (4.22 a)

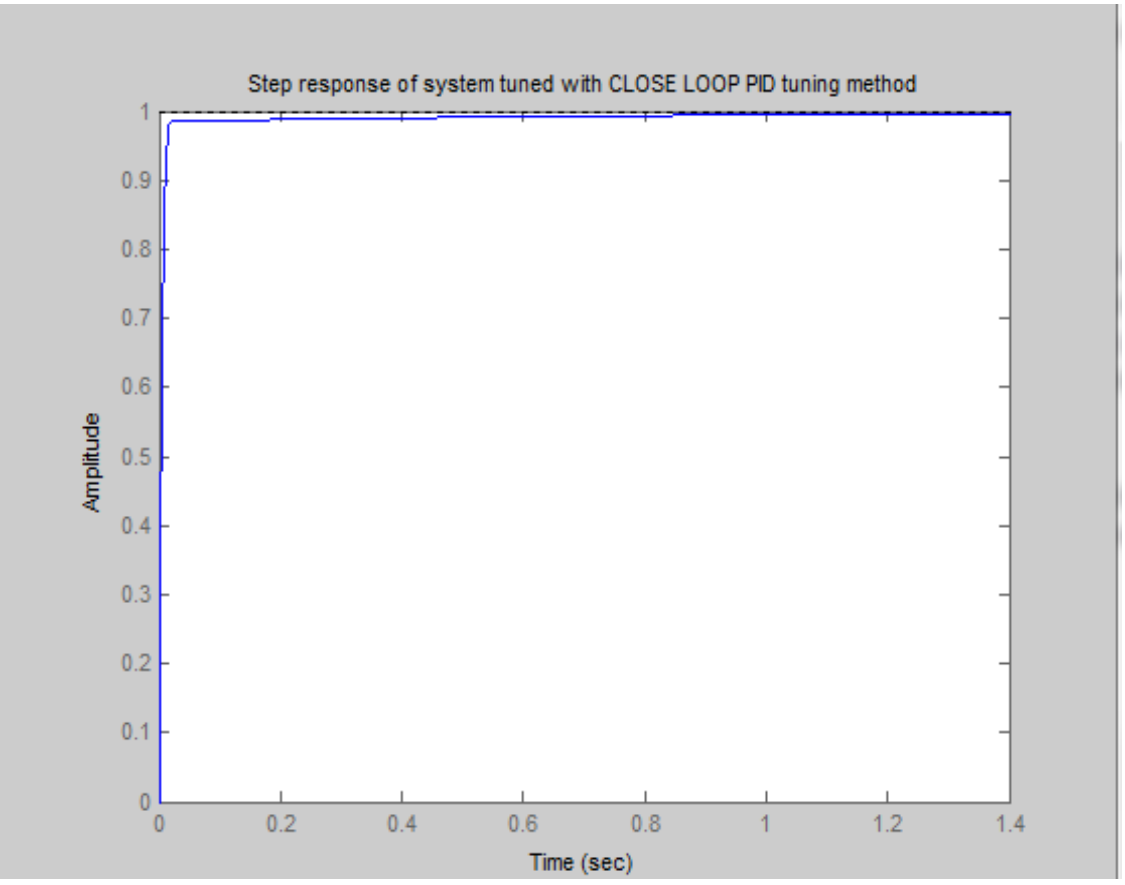

Figure 4.22(a) - Step response of Astrom-Hagglund PI method

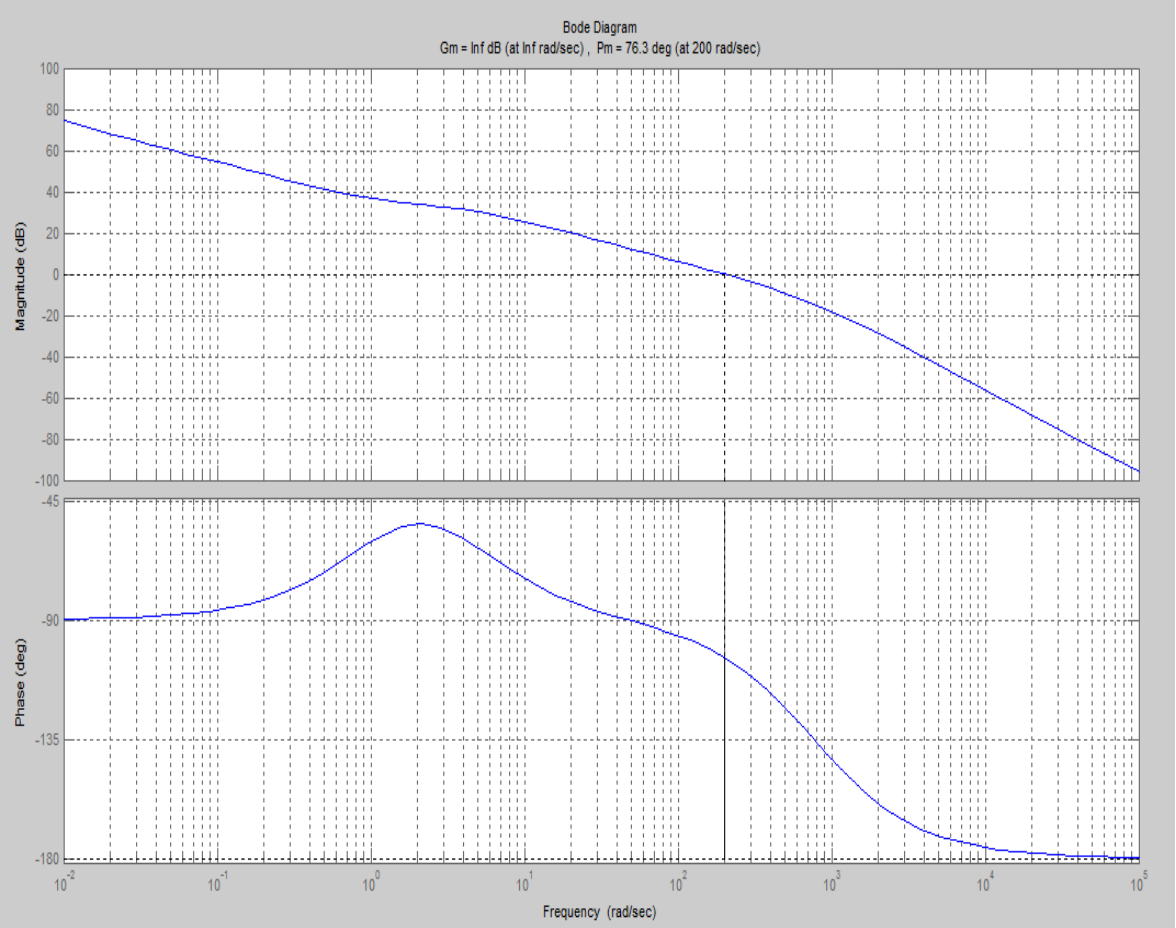

Figure 4.22(b) – Bode plot Astrom-Hagglund PI method

4.3.2.6- RESULTS

Table (4.14) shows the PID and PI parameters and time domain analysis; table (4.15) shows frequency domain analysis, modulus margin and delay margin of closed loop PID and PI tuning methods for servo speed control of DC motor model.

Table 4.14- Closed loop tuning method results.

Tuning Method	PID Parameters			Time Domain Analysis		
	Kp	Ti	Td	Rise Time	Settling Time	Overshoot
Ziegler-Nichols closed loop Method	0.9340	0.0038	9.4781e-004	0.0032	0.0313	34.3114
Modified ZN method (no overshoot)	0.3113	0.0038	0.0038	0.0070	0.0869	37.6038
Modified ZN method (some overshoot)	0.5137	0.0038	0.0025	0.0050	0.0521	33.1004
Tyreus-Luyben Method	0.7005	0.0167	0.0012	0.0056	0.0408	12.1640
Chien- Hrones- Reswick PI method	0.7316	1	0	0.0060	0.0082	0.5665
Astrom-Hagglund PI method	0.4981	0.9400	0	0.0090	0.0166	0

Table 4.15- Closed loop tuning method frequency domain parameters with modulus margin and delay margin.

Method	Frequency Domain Analysis		Modulus Margin (in dB)	Delay Margin (in seconds)
	Gain Margin	Phase margin (in degrees)		
Ziegler-Nichols closed loop Method	Inf	45.0790	0.4441	0.0021
Modified ZN method (no overshoot)	Inf	36.7854	0.2484	0.0038
Modified ZN method (some overshoot)	Inf	45.3030	0.4453	0.0034
Tyreus-Luyben Method	Inf	77.5336	-608.5485	0.0049
Chien- Hrones- Reswick PI method	Inf	70.3118	0.5749	0.0043
Astrom-Hagglund PI method	Inf	76.3053	0.7797	0.0067

Figures (4.23), (4.24) show comparison of PID tuning methods while figures (4.25), (4.26) show comparison of PI tuning methods. Figure (4.26) is a zoomed version of figure (4.25) for better observation.

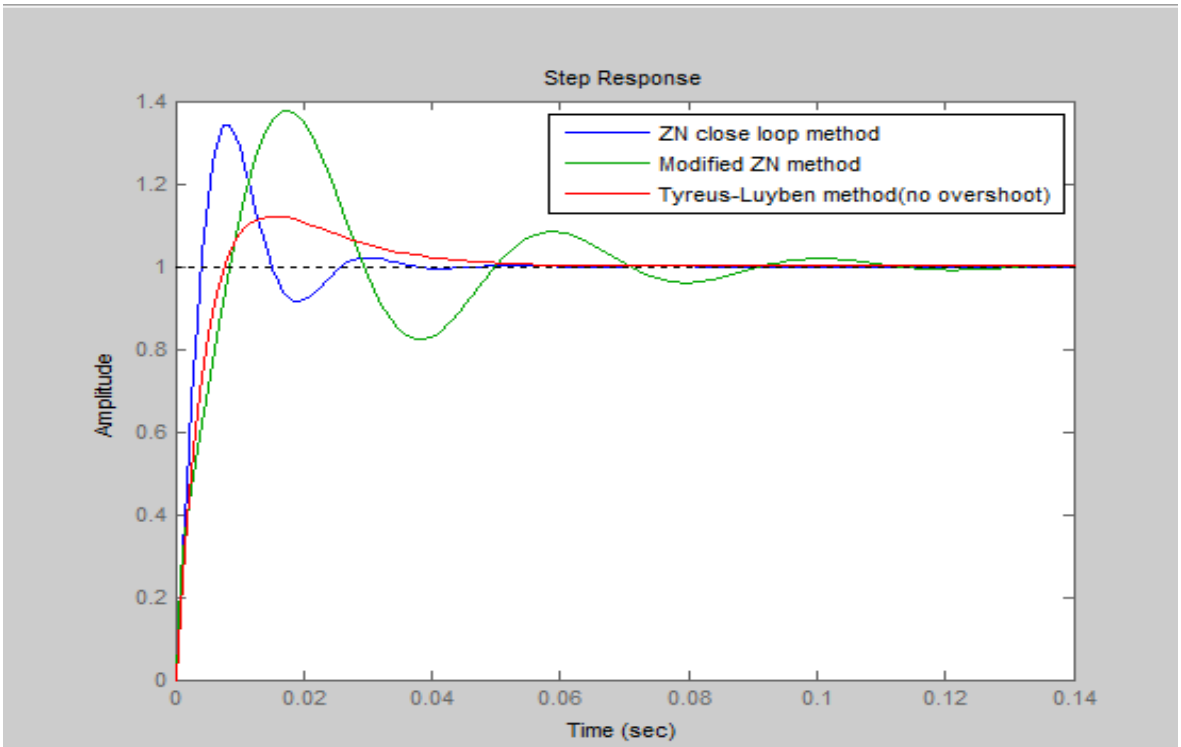

Figure 4.23- Comparison of CLOSED LOOP PID tuning methods (no overshoot)

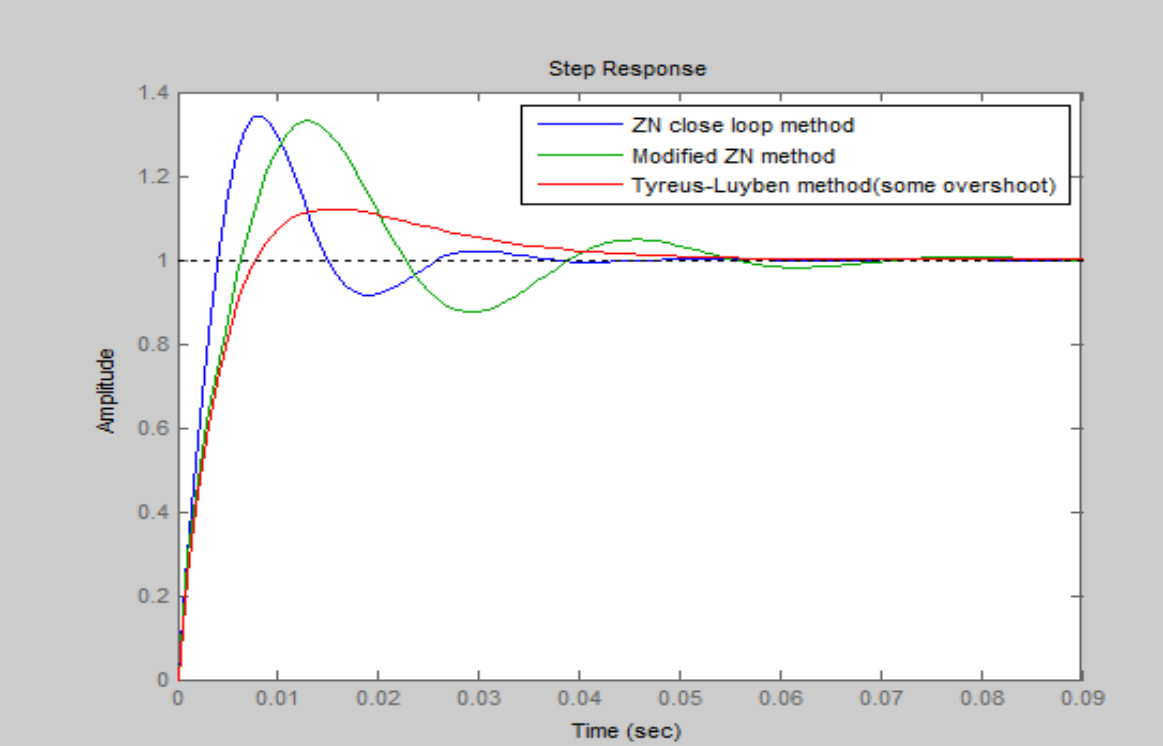

Figure 4.24- Comparison of CLOSED LOOP PID tuning methods (some overshoot)

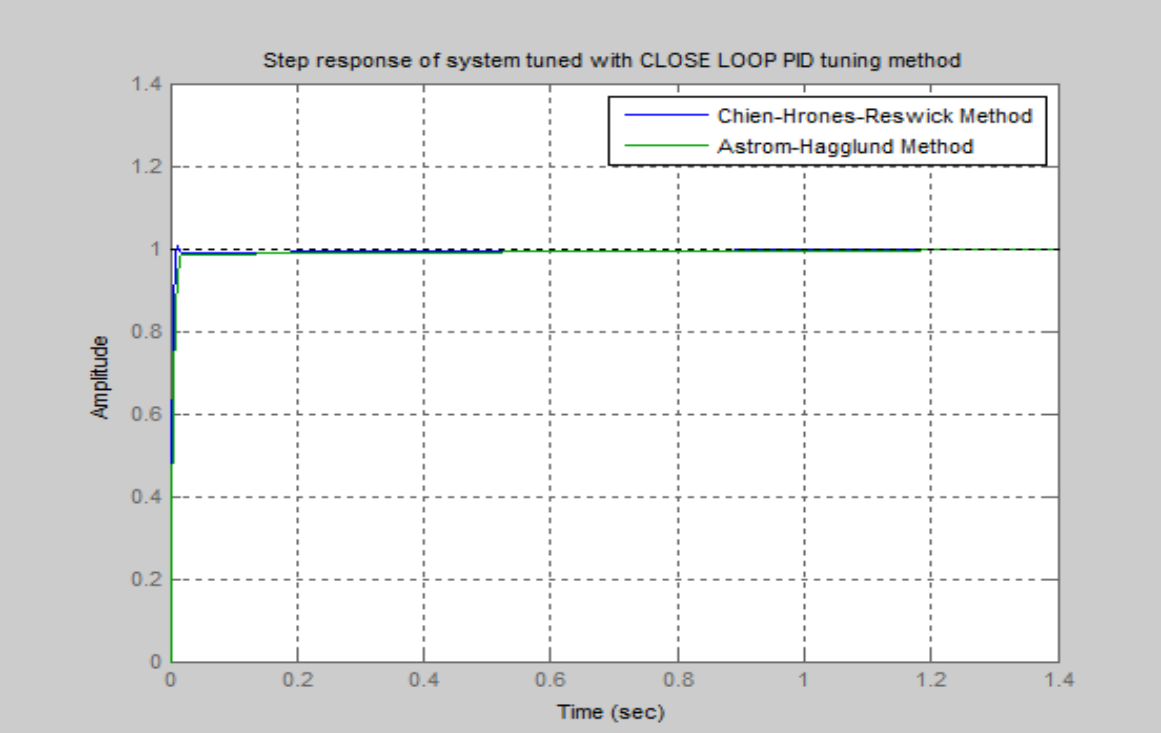

Figure 4.25- Comparison of CLOSED LOOP PI tuning methods

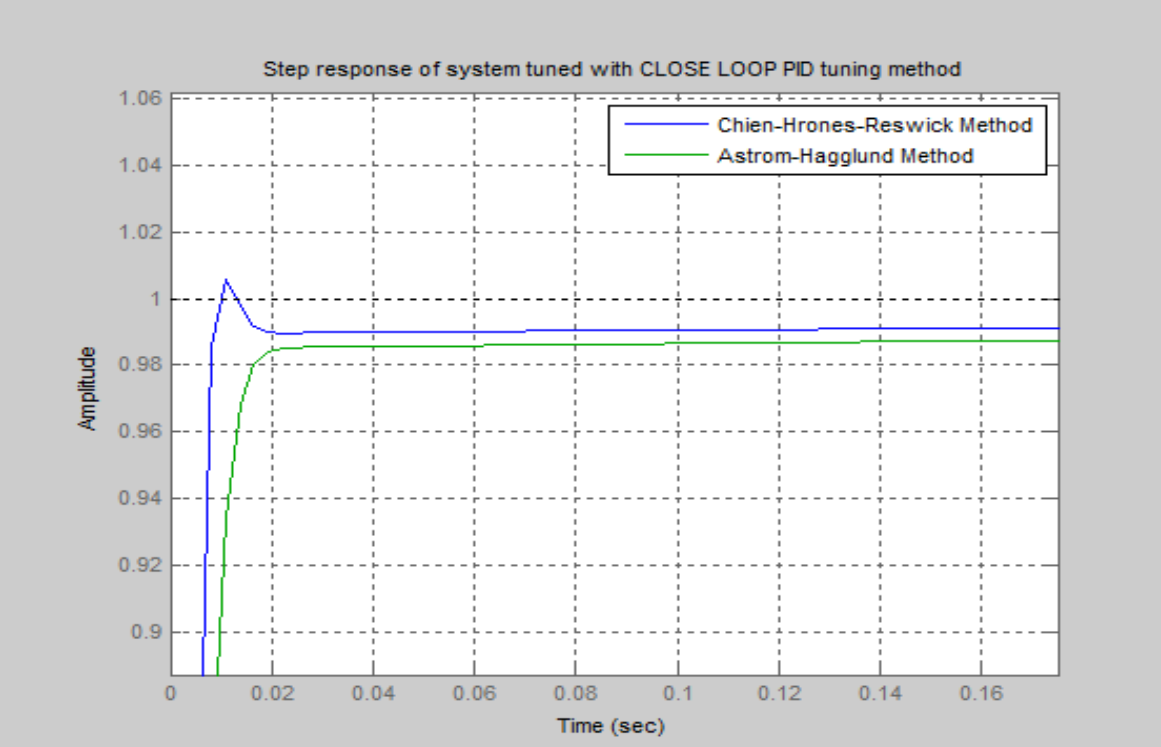

Figure 4.26- CLOSED LOOP PI tuning methods comparison with 3x zoom

4.3.2.7- CONCLUSION

Table (4.14) shows the time domain analysis of Closed loop tuning methods for DC motor model. Here values of rise time and settling time are very low. But the overshoot value is not 0 and some overshoot remains in system. Figure (4.23) and (4.24) shows the step response of various Closed loop tuning methods and it is seen that Tyreus- Luyben tuning method gives best performance for DC machine model.

Table (4.15) shows frequency domain analysis and robust parameters (namely modulus and delay margin) for DC machine model obtained using Closed loop tuning methods. In frequency domain analysis, only ZN closed loop and Modified ZN tuning method gives value of phase margin within range but gain margin is not achieved. In robustness analysis, modulus margin does not fall within specified range for DC machine model while delay margin is within required range for all closed loop tuning methods.

4.4- LIMITATIONS OF CLASSICAL TUNING METHODS

From the results obtained in section (4.3.1.8) and (4.3.2.7) it is observed that

For Open Loop Tuning Methods

- Overshoot is 0 for DC motor model.
- Rise time and Settling time values are not as good as closed loop PID tuning methods.
- All parameters for Frequency and robustness analysis are not falling entirely within range.

For Closed loop Tuning Methods

- Overshoot cannot be minimized to specified range of 2 to 5% for DC motor model.
- Rise time and settling time values are better than those obtained by Open loop Tuning methods.
- All parameters for Frequency and robustness analysis are not falling entirely within range.

Also the results show that the frequency domain analysis and robust parameters cannot be obtained within required range for DC motor model as the model transfer function is of 2^{nd} order. Thus classical PID tuning methods cannot work for any uncertainties arising in load for DC motor model.

Open loop and Closed loop tuning methods do not satisfy all the requirements for stable operation. Open loop tuning methods can be selected for tuning if 0 overshoot is our main objective, but objective to achieve better rise time and settling time will suggest Closed loop tuning methods as first choice. This limits the performance of Open and Closed loop PID tuning methods.

There are certain error indices based optimal PID tuning methods which can provide better performance as compared to classical Open loop and Closed loop tuning methods. These methods utilize IAE, ISE, ITAE, ISTE based rules for optimal PID tuning and these are discussed next for selected DC motor model.

CHAPTER 5:-

OPTIMAL PID TUNING USING PERFORMANCE INDICES

5.1- INTRODUCTION

Classical PID tuning methods are not able to give optimal performance and there is a compromise in performance of system during selection of tuning method. To obtain better parameters for PID controller in order to achieve optimal performance, the optimal PID tuning method utilizing Performance (error) indices is adopted. This method is considered better than classical PID tuning methods and will be implemented for DC motor model.

5.2- PERFORMANCE INDICES

To optimize the performance of a PID controlled system, the PID gains of the system are adjusted to minimize a certain performance index. The performance index is calculated over a time interval; *T*, normally in the region of $0 \le T \le t_s$ where t_s is the settling time of the system. The performance indices used were:

5.2.1- Integral of Absolute magnitude of Error (IAE)

$$IAE = \int_0^T |e(t)|dt \qquad \text{... (5.1)}$$

5.2.2- Integral of Square of Error (ISE)

$$ISE = \int_0^T e^2(t)dt \qquad \text{... (5.2)}$$

5.2.3- Integral of Time multiplied by Absolute Error (ITAE)

$$ITAE = \int_0^T t|e(t)|dt \qquad \text{... (5.3)}$$

5.2.4- Integral of Square of Time multiplied by Error (ISTE)

$$ISTE = \int_0^T \left(te(t)\right)^2 dt = \int_0^T t^2 e^2(t) dt \qquad \text{... (5.4)}$$

5.3.3-OPTIMAL TUNING OF PID USING ERROR INTEGRALS

The performance indices namely integral square error (ISE), integral absolute error (IAE), integral time absolute error (ITAE) performance criteria and integral square time error (ISTE) are used for Optimal PID tuning. The optimal equations for obtaining PID parameters are suggested by P.W. Murril *et al.*and M. Zhuang *et al.* Both methods are discussed briefly in the section below.

1- Method proposed by P.W. Murril *et al.:*

This method proposed by P.W. Murrill *et al.* for optimal tuning of PID controller using integral performance indices for Servo and Regulatory response.

1.1- SERVO RESPONSE for speed control:

If the tuning parameters are calculated for a set-point change (i.e. for Servo Response), the integral time will be longer and the derivative time will be shorter, and they will depend mostly on the time constant of the process.

The relationship between the controller settings based on integral criterion and the ratio t_o / τ is expressed by the tuning relationship given in equation 5.5

$$Y = A\left(\frac{t_0}{\tau}\right)^B \qquad \text{... (5.5)}$$

Where $Y = KK_c$ for proportional mode, τ / T_i for reset mode; A, B=constants for given controller and mode; t_0, τ =pure delay time and first-order lag time constant respectively.

Using these equations:

$$K_c = \frac{A}{K}\left(\frac{t_0}{\tau}\right)^B \quad \text{... (5.6)}$$

$$\frac{1}{T_i} = \frac{A}{\tau}\left(\frac{t_0}{\tau}\right)^B \quad \text{... (5.7)}$$

$$T_d = \tau * A\left(\frac{t_0}{\tau}\right)^B \quad \text{... (5.8)}$$

a) For PI Controller:

The table for coefficients A and B for PI controller optimal tuning for error indices is given in table below:

Table 5.1-Servo PI tuning method by **P.W. Murril *et al***

ERROR INDICES	Controller	A	B
IAE	P	0.758	-0.861
	I	1.020	-0.323
ITAE	P	0.586	-0.916
	I	1.030	-0.165

b) For PID Controller:

The following table shows the value of coefficient A and B for PID controller

Table 5.2- Servo PID tuning method by **P.W. Murril *et al***

ERROR INDICES	Controller	A	B
IAE	P	1.086	-0.869
	I	0.740	-0.130
	D	0.348	0.914
ITAE	P	0.965	-0.855
	I	0.796	-0.147
	D	0.308	0.929

1.2- REGULATORY RESPONSE for speed control:

With tuning parameters calculated for load rejection (i.e. for Regulatory Response), the integral time (T_i) and derivative time (T_d) will depend mostly on the dead time (t_d) of the process

a) For PI Controller:

Table below shows value of A and B for PI controller

Table 5.3-Regulatory PI tuning method by **P.W. Murril *et al***

ERROR INDICES	Controller	A	B
IAE	P	0.984	-0.986
	I	0.608	-0.707
ITAE	P	0.859	-0.977
	I	0.674	-0.680
ISE	P	1.305	-0.959
	I	0.492	-0.738

b) For PID Controller:

Table below shows value of A and B for PID controller

Table 5.4-Regulatory PID tuning method by **P.W. Murril *et al***

ERROR INDICES	Controller	A	B
IAE	P	1.435	-0.921
	I	0.878	-0.749
	D	0.482	1.137
ITAE	P	1.357	-0.947
	I	0.842	-0.738
	D	0.381	0.995
ISE	P	1.495	-0.945
	I	1.101	-0.771
	D	0.560	1.006

2- Method proposed by M. Zhuang *et al.:*

This method proposed by M. Zhuang *et al.* is used to obtain optimum PID controller settings for minimizing time weighted integral performance criteria. FOPDT model is considered here for optimization using method proposed by M. Zhuang *et al.*

2.1- SERVO RESPONSE for speed control:

The FOPDT model transfer function is given by:

$$G(s) = \frac{Ke^{-s\tau}}{Ts+1} \qquad \text{... (5.9)}$$

The following formulae gives the KK_c, T/T_i, T_d/T as the functions of τ/T.

(range of τ/T is between 0.1 to 1.0)

$$K_c = \frac{a_1}{K}\left(\frac{\tau}{T}\right)^{b_1} \qquad \text{... (5.10)}$$

$$T_i = \frac{T}{a_2 + b_2(\tau/T)} \qquad \text{... (5.11)}$$

$$T_d = a_3 T\left(\frac{\tau}{T}\right)^{b_3} \qquad \text{... (5.12)}$$

a) For PI Controller

The values of coefficients for optimal PI controller tuning is given in following table

Table 5.5- Servo PI tuning method by M. Zhuang *et al*

Parameters	ISE	ISTE
a_1	0.980	0.712
b_1	-0.892	-0.921
a_2	0.690	0.968
b_2	-0.155	-0.247

b) For PID Controller

The following table shows PID Controller optimal tuning parameter values:

Table 5.6- Servo PID tuning method by M. Zhuang *et al*

Parameters	ISE	ISTE
a_1	1.048	1.042
b_1	-0.897	-0.897
a_2	1.195	0.987
b_2	-0.368	-0.238
a_3	0.489	0.385
b_3	0.888	0.906

2.2- REGULATORY RESPONSE for speed control:

The tuning formulae for step disturbance input (i.e. regulatory response) are:

$$K_p = \frac{a_1}{K}\left(\frac{\tau}{T}\right)^{b_1} \quad \text{... (5.13)}$$

$$\frac{1}{T_i} = \frac{a_2}{T}\left(\frac{\tau}{T}\right)^{b_2} \quad \text{... (5.14)}$$

$$T_d = a_3 T\left(\frac{\tau}{T}\right)^{b_3} \quad \text{... (5.15)}$$

a) For PI Controller

The values of coefficients for PI controller are:

Table 5.7- Regulatory PI tuning method by M. Zhuang *et al*

Parameters	ISE	ISTE
a_1	1.279	1.015
b_1	-0.945	-0.957
a_2	0.535	0.667
b_2	-0.586	-0.552

b) For PID Controller

The values of coefficients for PID controller are:

Table 5.8- Regulatory PI tuning method by M. Zhuang *et al*

Parameters	ISE	ISTE
a_1	1.473	1.468
b_1	-0.970	-0.970
a_2	1.115	0.942
b_2	-0.753	-0.725
a_3	0.550	0.443
b_3	0.948	0.939

5.4- RESULTS

The time domain analysis of Servo Speed control with its transfer function and step response are given below (for open loop uncontrolled DC motor model).

Servo Speed Control:

Transfer function:

```
              1
-----------------------------------
3.15e-006 s^2 + 0.002428 s + 0.01012
```

Time domain analysis:

RiseTime : 0.5245

SettlingTime : 0.9349

Overshoot : 0

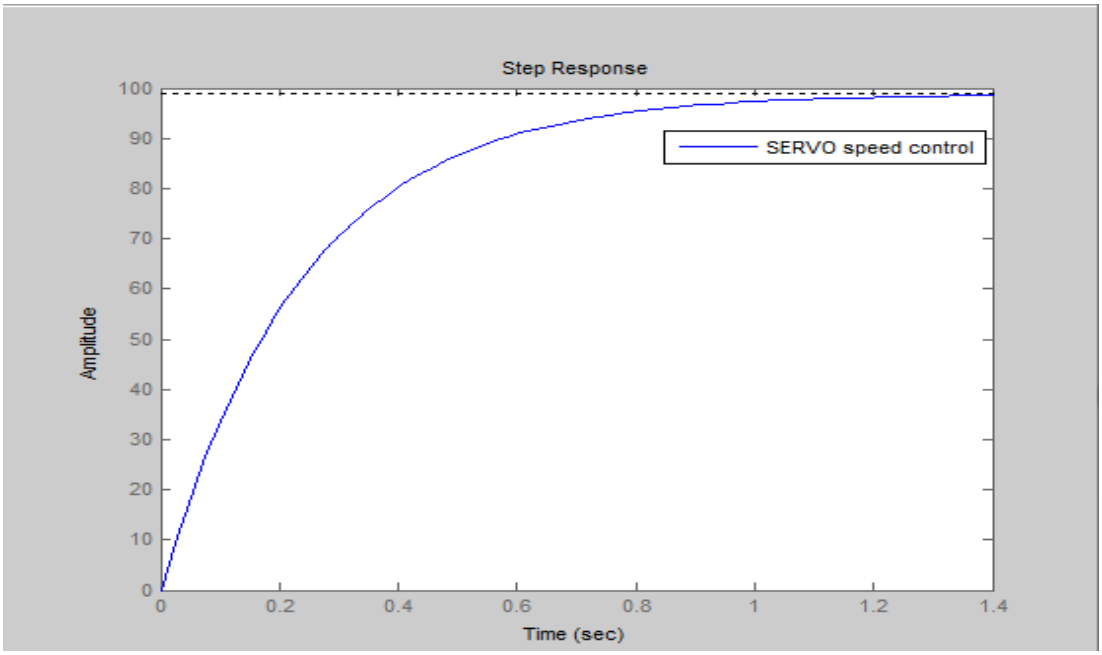

Figure 5.1- Step response of servo speed control

The PID parameters, time domain analysis and frequency domain analysis for Servo Speed Control is shown in table below:

Table 5.9- Servo speed control parameters table.

Error Indices	Optimal PID Parameter			Time domain analysis			Frequency Domain Anlysis		Robust Performance Analysis	
	Kp	Ti	Td	Rise Time	Settling Time	Over shoot	Gain margin	Phase margin	Modulus Margin	Delay Margin
IAE	0.0062	0.4813	0.1499	2.0827	3.7266	0	Inf	108.3044	-43.1700	1.4412
ISE	0.0059	0.4892	0.2070	2.2369	3.9302	0	Inf	108.2180	-31.2342	1.5904
ITAE	0.0056	0.4588	0.1340	2.1626	3.8597	0	Inf	105.5232	-55.2037	1.4894
ISTE	0.0058	0.4473	0.1649	2.0274	3.5755	0	Inf	106.0719	-41.7291	1.4399

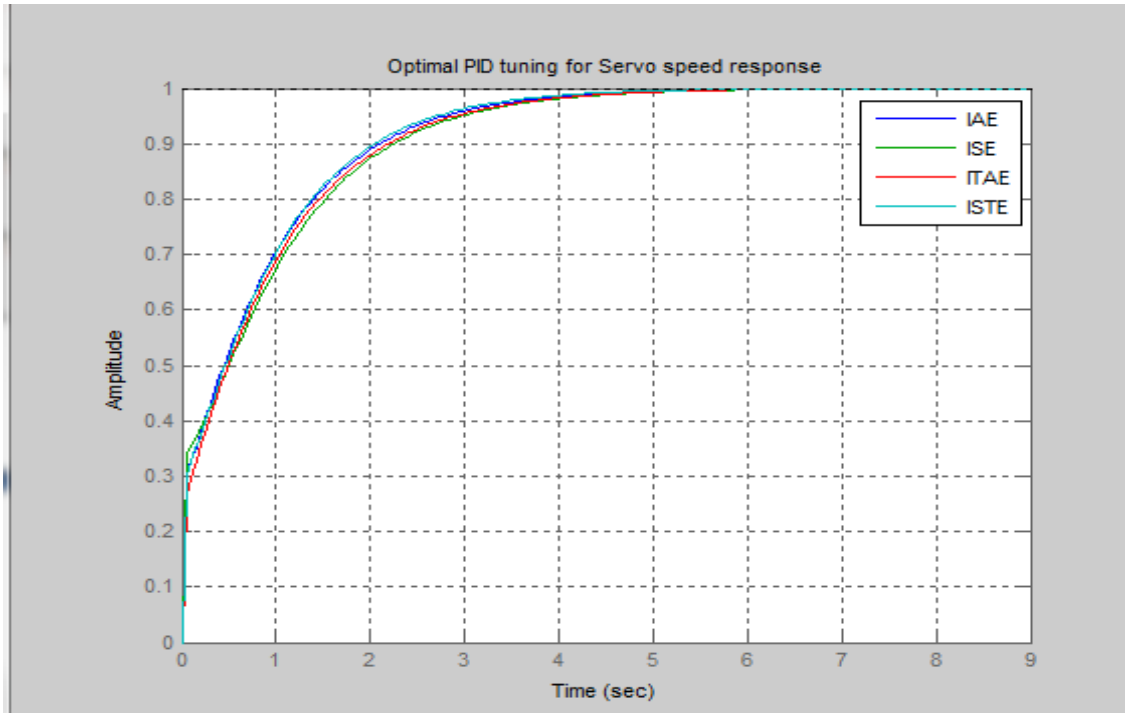

Figure 5.2- Servo speed control error indices step response

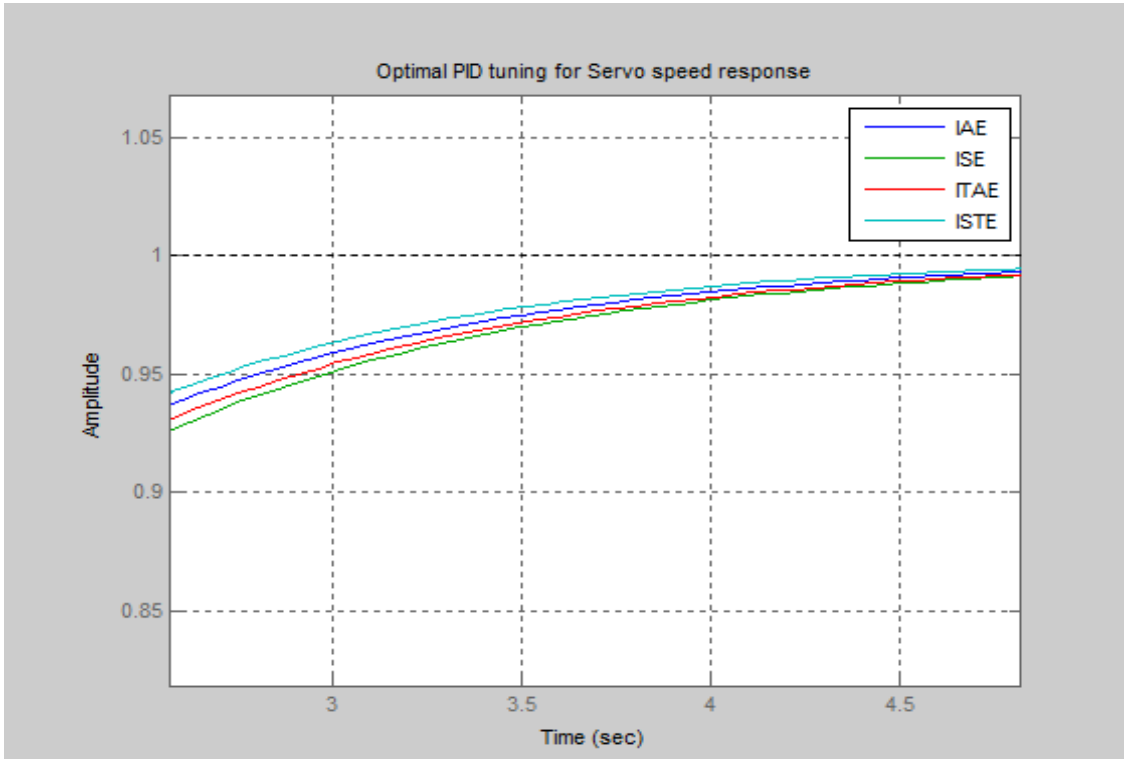

Figure 5.3- Servo speed control error indices step response (2x zoom)

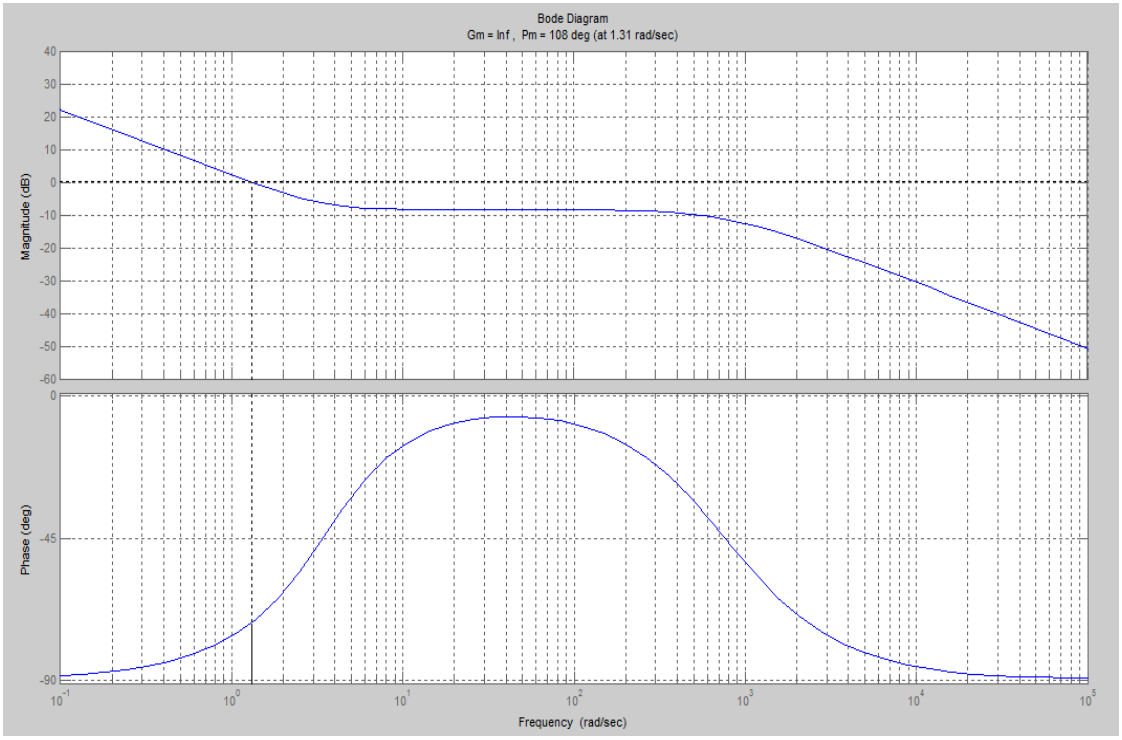

Figure 5.4- Bode plot for IAE based control

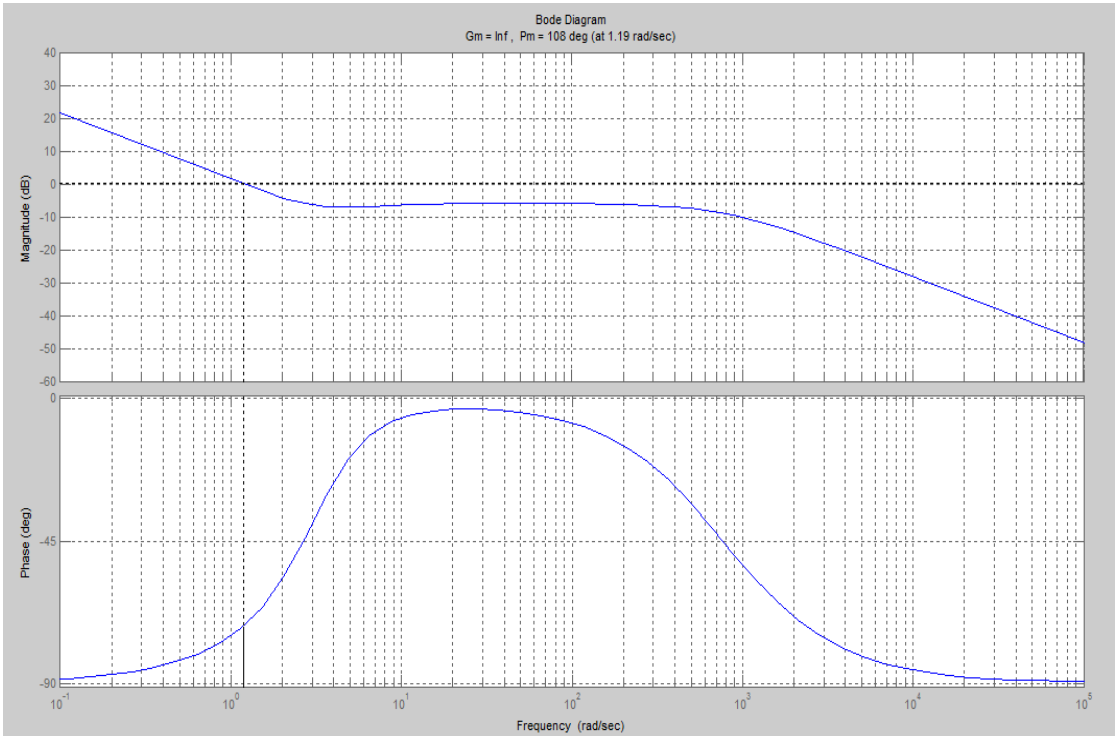

Figure 5.5- Bode plot for ISE based control

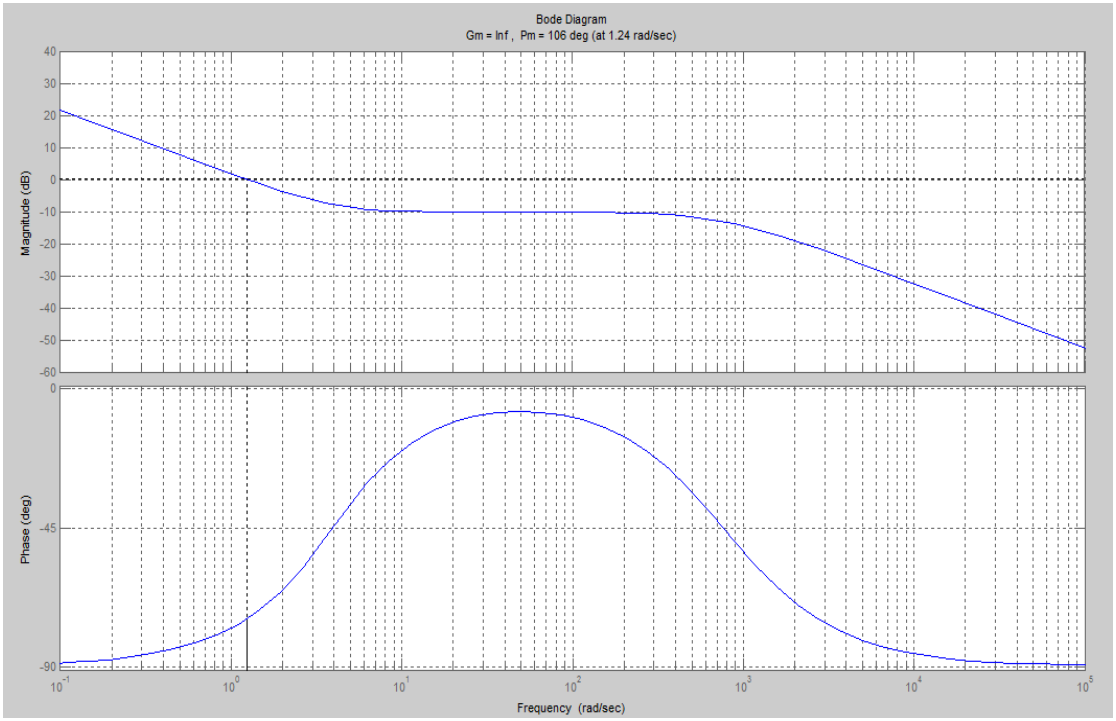

Figure 5.6- Bode plot for ITAE based control

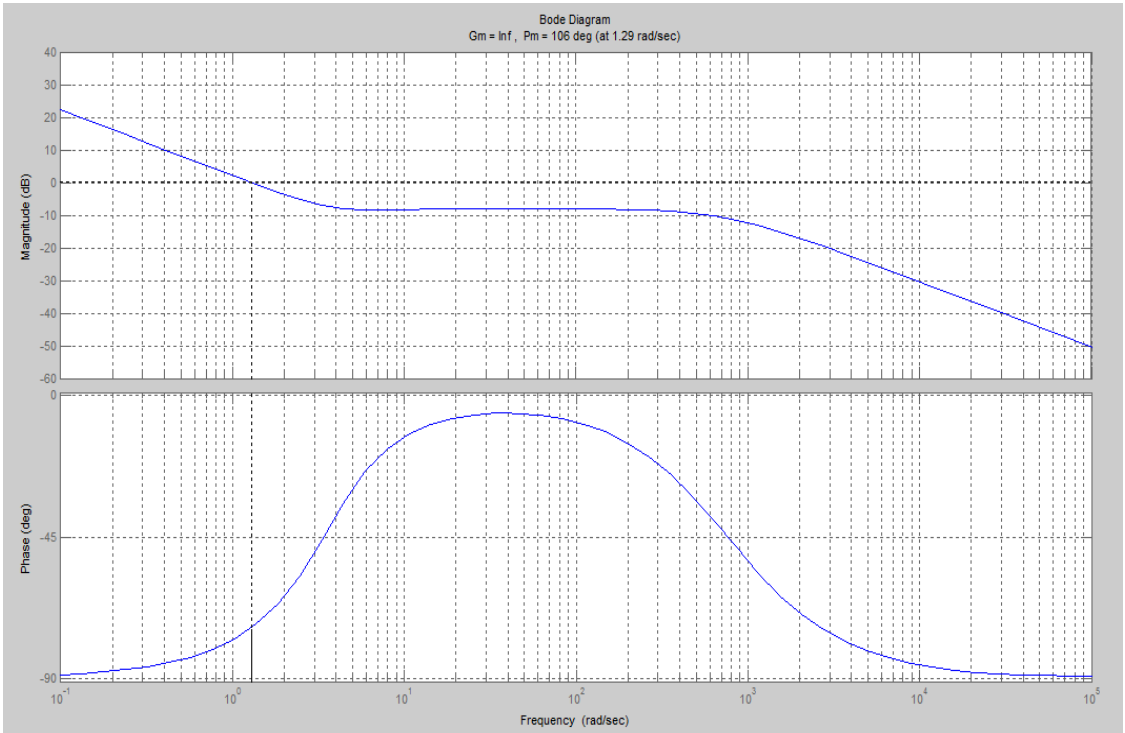

Figure 5.7- Bode plot for ISTE based control

5.5- CONCLUSION

Table (5.9) shows the optimal PID parameters, time domain analysis, frequency domain and robust parameter obtained by Optimal PID tuning method for DC motor model. It is clear from the table that 0 overshoot is achieved like open loop tuning method and also rise time , settling time values are better than open loop tuning method. Thus Optimal tuning methods overcome the drawbacks of both open loop and closed loop tuning methods and gives satisfactory results for time domain analysis. However frequency domain analysis (both gain margin and phase margin) and Robustness analysis (modulus margin) do not fall in required range of stability for DC motor model even after using Optimal PID tuning method. Only delay margin is a robustness parameter that is falling within specified range.

Figure (5.2) and (5.3) shows that for DC motor model, optimal PID tuning with ISTE and IAE gives best performance for step response among the four selected error indices.

AI based techniques are gaining popularity these days due to their high optimization capabilities. PSO and BFO are such two techniques and will be discussed next for DC machine model.

CHAPTER 6

PARTICLE SWARM OPTIMIZATION (PSO) ALGORITHM

6.1- INTRODUCTION

Several optimization techniques using swarming principle have been adopted to solve a variety of engineering problems in past decade. Particle Swarm Optimization (PSO) is such swarming techniques which can be used for PID tuning. Swarming strategies in bird flocking and fish schooling are used in PSO introduced by Eberhart and Kennedy. PSO is applied in PID tuning and its results are compared to the results of some classical tuning methods to find out which is better. Time domain analysis, frequency domain analysis and robustness analysis are tabulated for our DC motor model controlled with optimal value of PID controller obtained using PSO algorithm.

6.2- PSO ALGORITHM

PSO optimizes an objective function by undertaking a population-based search. The population consists of potential solutions, named particles, which are a metaphor of birds in flocks. These particles are randomly initialized and freely fly across the multi-dimensional search space. During flight, each particle updates its own velocity and position based on its own experience and that of the entire population. The updating policy drives the particle swarm to move towards the region with higher objective function value and eventually all particles will gather around the point with the highest objective value.

The detailed operation of particle swarm optimization is given below:-

STEP1: Initialization. The position and velocity of all particles are randomly set within pre-defined ranges.

STEP2: Velocity Updating. Velocities of all the particles are updated at each iteration:-

$$\vec{v}_i = w\vec{v}_i + c_1 R_1(\vec{p}_{i,best} - \vec{p}_i) + c_2 R_2(\vec{g}_{i,best} - \vec{p}_i) \quad \text{... (6.1)}$$

Where

$\vec{p}_i$ and $\vec{v}_i$ - position and velocity of particle i respectively,

$\vec{p}_{i,best}$ and $\vec{g}_{i,best}$ - position with 'best' objective value found so far by particle i and entire population respectively,

w - parameter controlling the flying dynamics,

R_1 and R_2 - random variables in the range of [0,1],

c_1 and c_2 - factors controlling the related weighting of corresponding terms.

After updating $\vec{v}_i$ should be checked and maintained within a pre-specified range to avoid random walking.

STEP3: Position. Assuming a unit time interval between successive iterations, the positions of all particles are updated according to:-

$$\vec{p}_i = \vec{p}_i + \vec{v}_i \quad \text{... (6.2)}$$

After updating $\vec{p}_i$ should be checked and limited to the allowed range.

STEP4: Memory updating. Update $\vec{p}_{i,best}$ and $\vec{g}_{i,best}$ when condition is met.

$$\vec{p}_{i,best} = \vec{p}_i \quad \text{if } f(\vec{p}_i) > f(\vec{p}_{i,best}) \quad \text{... (6.3)}$$

$$\vec{g}_{i,best} = \vec{g}_i \quad \text{if } f(\vec{g}_i) > f(\vec{g}_{i,best}) \quad \text{...(6.4)}$$

Where $f(\vec{x})$ is the objective function subject to maximization.

STEP5: Termination Checking. The algorithm repeats Steps 2 to 4 until certain termination condition are met, such as a pre-defined number of iterations or a failure to make progress for a certain number of iterations.

6.3- PSO FLOWCHART

The flowchart representation of PSO algorithm is shown in figure 5.1 drawn below.

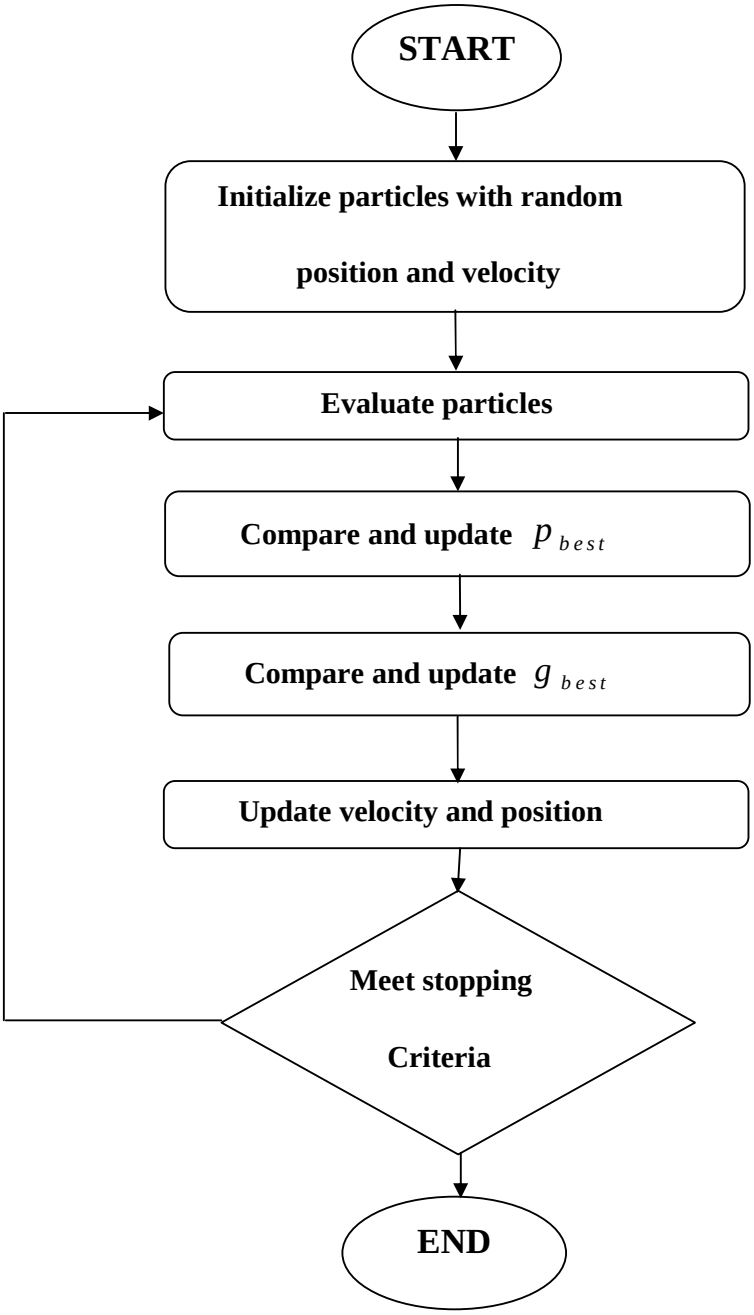

Figure 6.1- PSO Flowchart

6.4- OPTIMAL PID TUNING USING PSO

PSO algorithm is used to obtain the optimal values for PID controller parameters namely K_p, K_i and K_d for DC motor model**.** In order to achieve this, a multi-objective function has to be designed for PSO algorithm.

The multi-objective function in PSO uses only one of many error indices along with overshoot for optimization at a time. The figures (6.2), (6.3), (6.4) and (6.5) show the step response of controlled system using PSO algorithm with IAE, ISE, ITAE and ISTE in the multi-objective function respectively.

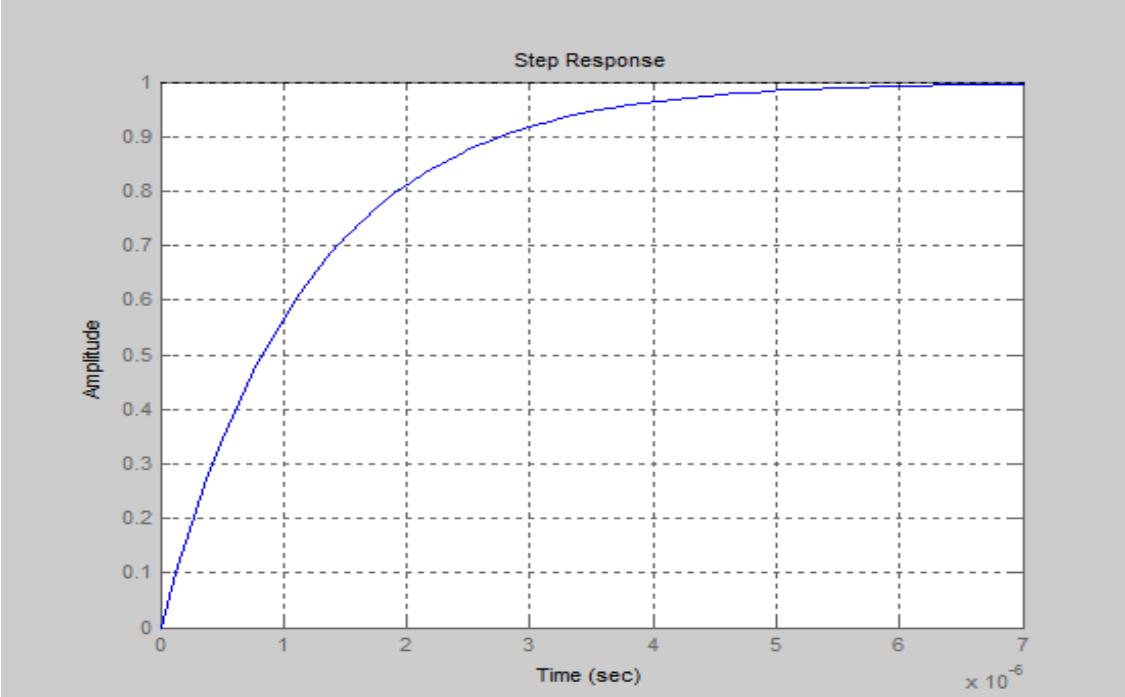

Figure 6.2(a)- Step response of controlled system when Multiobjective function of PSO uses IAE+overshoot

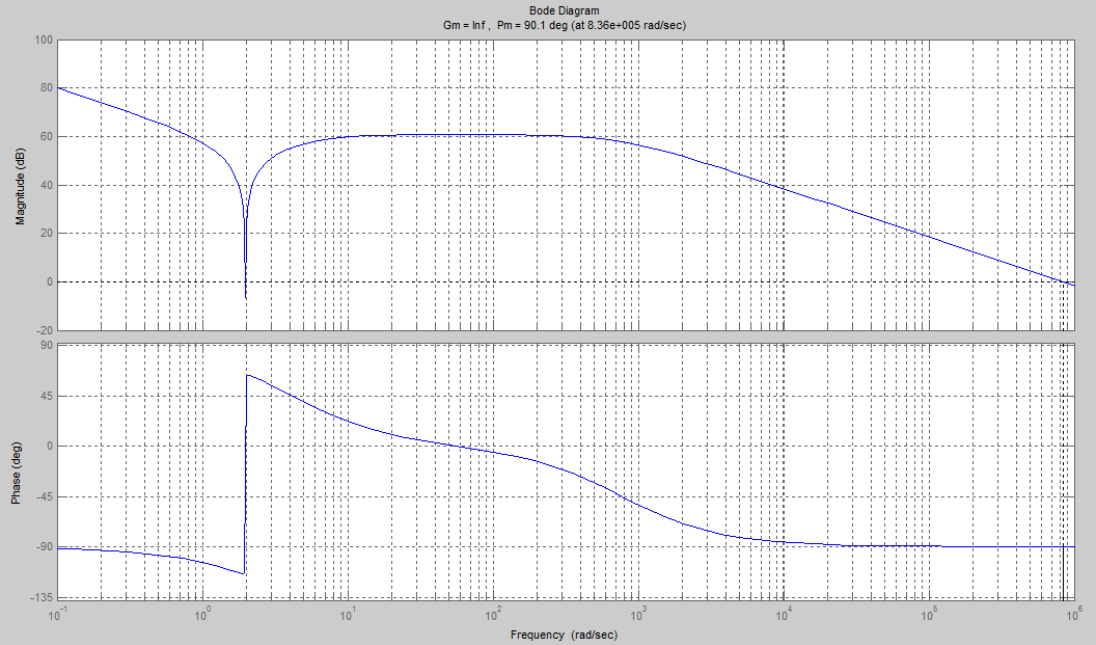

Figure 6.2(b)- Bode plot for controlled system when Multiobjective function of PSO uses IAE+overshoot

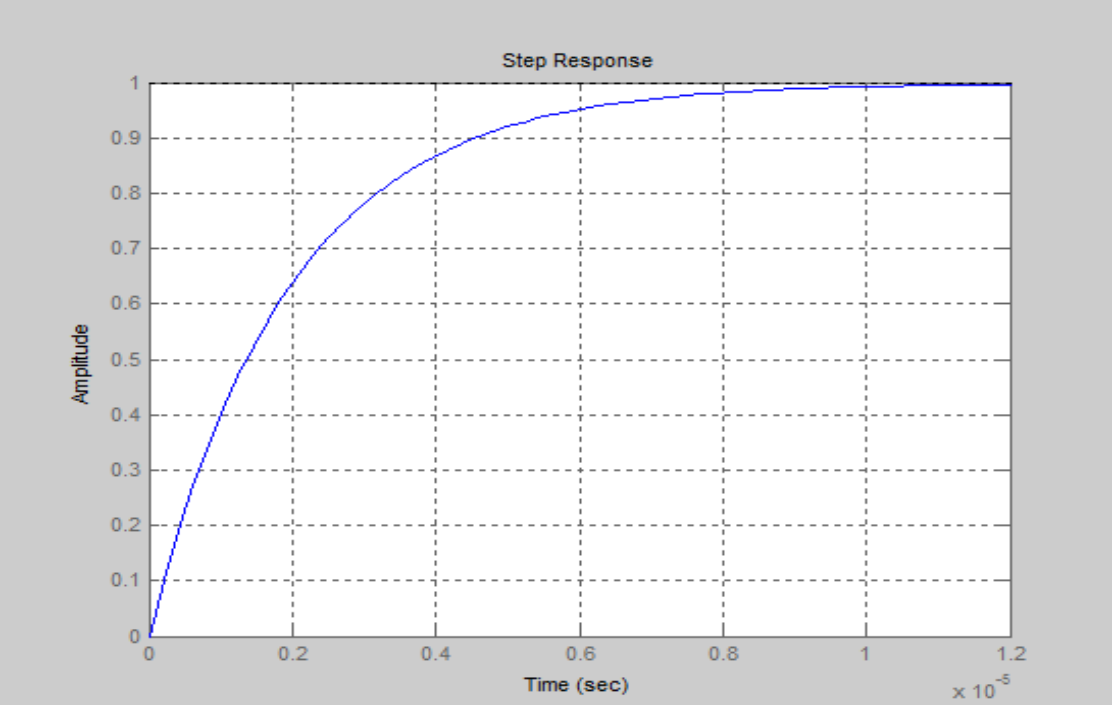

Figure 6.3(a) - Step response of controlled system when Multiobjective function of PSO uses ISE+overshoot

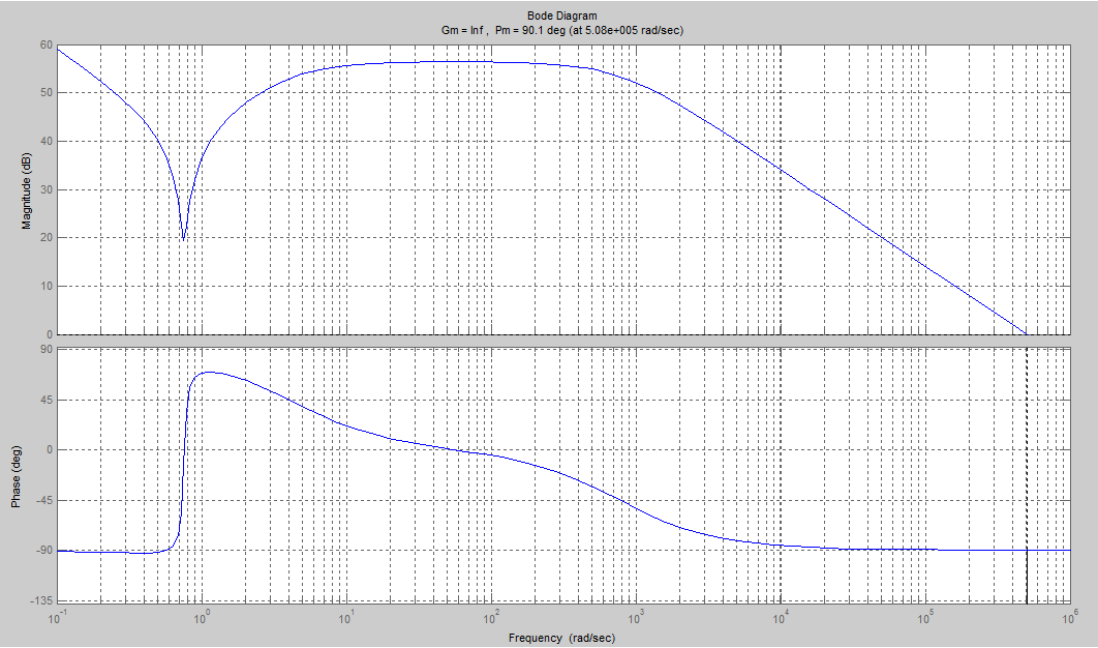

Figure 6.3(b) – Bode plot for controlled system when Multiobjective function of PSO uses ISE+overshoot

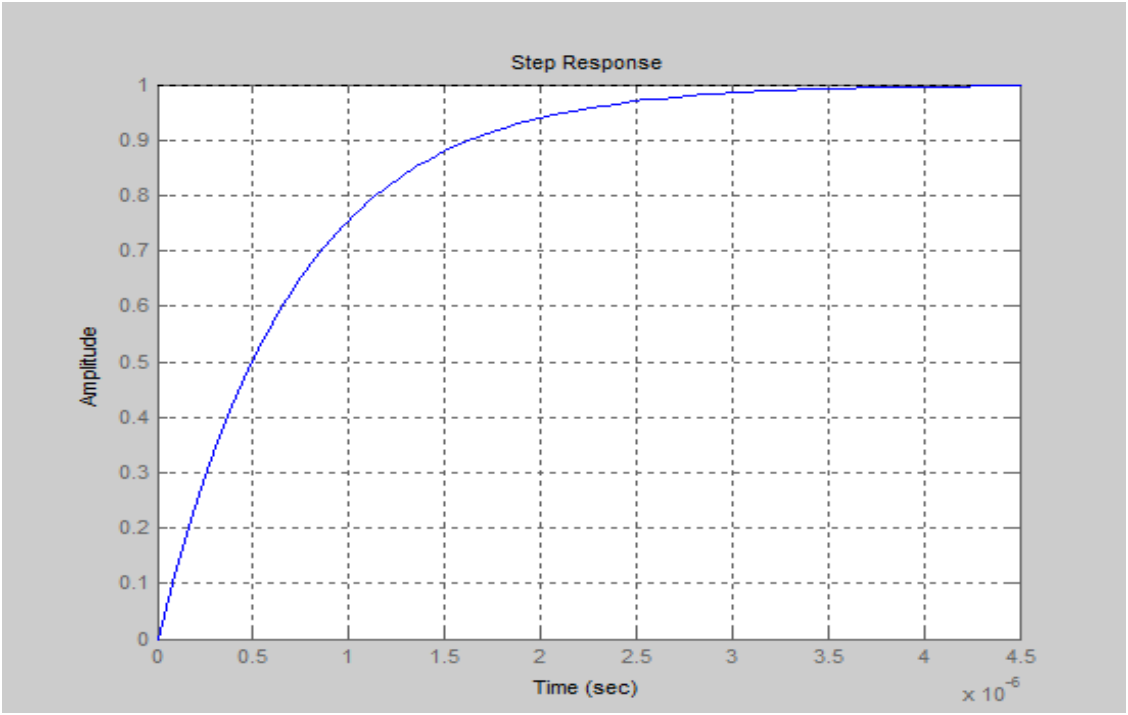

Figure 6.4(a) - Step response of controlled system when Multiobjective function of PSO uses ITAE+overshoot

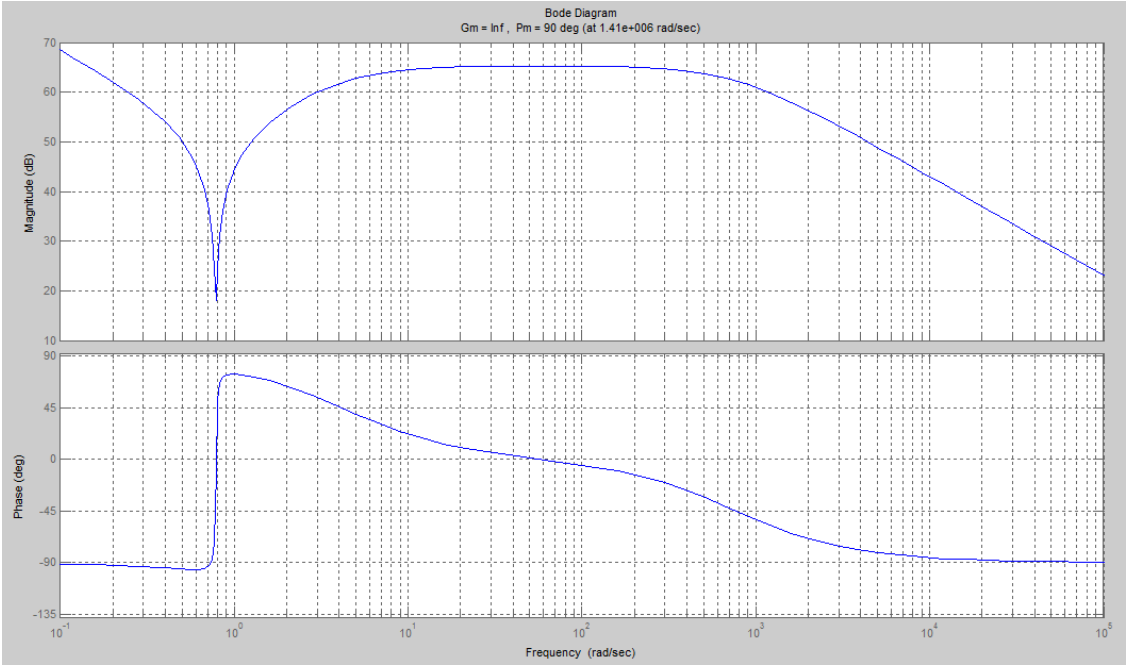

Figure 6.4(b) – Bode plot for controlled system when Multiobjective function of PSO uses ITAE+overshoot

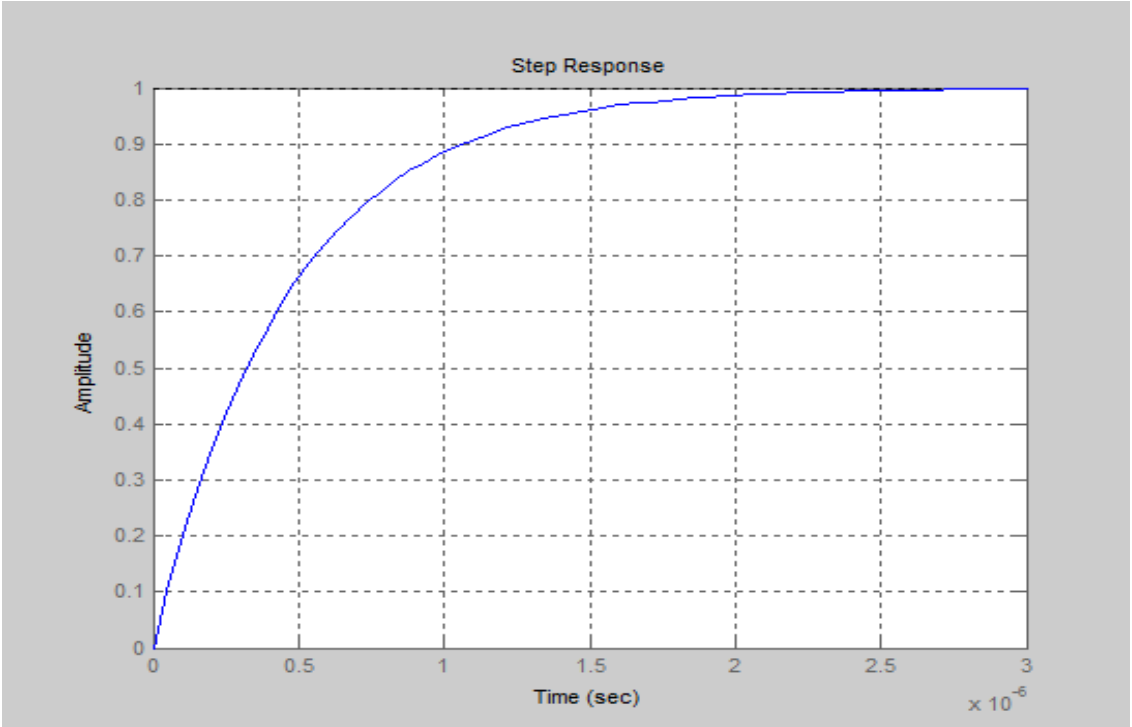

Figure 6.5(a) - Step response of controlled system when Multiobjective function of PSO uses ISTE+overshoot

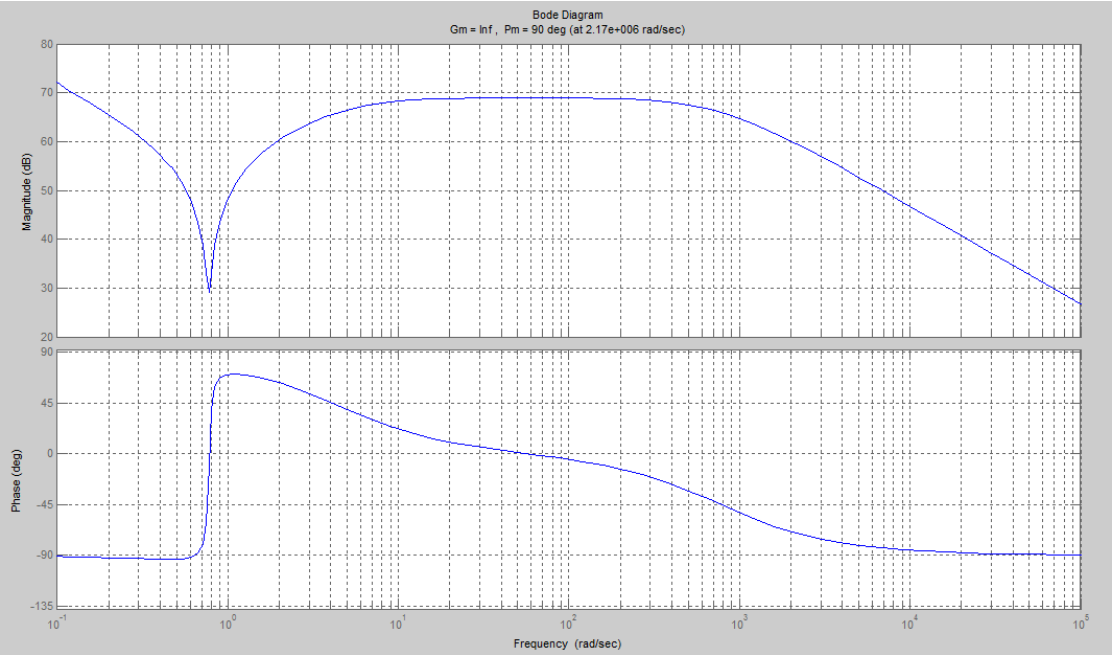

Figure 6.5(b) - Step response of controlled system when Multiobjective function of PSO uses ISTE+overshoot

6.5- RESULTS

The initial parameters for PSO algorithm are given below:

```
n = 50;              % Size of the swarm "no of birds "
bird_step =50;        % Maximum number of "birds steps"
dim = 3;             % Dimension of the problem
c2 =1.2;             % PSO parameter C2
c1 = 0.12;           % PSO parameter C1
w =0.9;              % pso momentum of inertia
```

The weighting factors are alpha=beta=10 for the objective function including error indices along with system overshoot as given below:

Objective function= Alpha*overshoot + beta*Error Indices

Table 6.1 below shows the optimized PID parameters obtained for DC motor model using PSO algorithm along with the time domain analysis of the system controlled with these optimal PID controller values.

Table 6.1- PSO algorithm based optimal PID parameters and time domain analysis of controlled system

Error Indices+ Overshoot	PID Parameters			Time Domain Analysis		
	Kp	Ki	Kd	Rise Time	Settling Time	Overshoot
IAE	0.0052	10.2683	2.6337	2.6364e-006	4.7314e-006	0
ISE	0.0961	0.9152	1.6011	4.3442e-006	7.8390e-006	0
ITAE	0.0815	2.7393	4.4426	1.5612e-006	2.7921e-006	0
ISTE	0.2940	4.0805	6.8244	1.0157e-006	1.8134e-006	0

Table 6.2 shows the frequency domain analysis of the controlled system along with robustness analysis for PSO algorithm. From this table it is concluded that with PSO algorithm the delay margin fall well within range of stable operation while modulus margin values are out of required range for DC motor model.

Table 6.2- Frequency domain and robustness analysis for controlled system obtained by PSO algorithm

Error Indices+ Overshoot	Frequency Domain Analysis		Robust Analysis	
	Gain Margin	Phase margin	Modulus margin	Delay Margin
IAE	Inf	90.0528	-3.2879e+003	1.8798e-006
ISE	Inf	90.0869	-88.9707	3.0934e-006
ITAE	Inf	90.0313	-1.1565e+003	1.1142e-006
ISTE	Inf	90.0204	-490.3441	7.2521e-007

6.6- CONCLUSION

Table (6.1) shows optimized PID parameters and time domain analysis obtained for DC motor model using PSO algorithm. Rise time, settling time and overshoot are noted in the table. For PSO algorithm the results shows that:

- 0 overshoot is obtained with every error indices.
- ISTE gives lowest rise time and settling time, next best is given by ITAE.
- ISE gives poorest result for DC motor model.

Frequency domain analysis (gain margin and phase margin) and robust parameters namely modulus margin and delay margin does not fall within required range for PSO tuned DC motor model. Thus uncertainties cannot be for DC motor model by using PSO optimized PID controller.

CHAPTER 7

BACTERIAL FORAGING OPTIMIZATION (BFO) ALGORITHM

7.1- INTRODUCTION

BFO, a relatively new algorithm proposed by K.M. Passino, is inspired by the social foraging behavior of *Escherichia Coli* (E. Coli) bacterium. Since proposed, BFO algorithm has been applied widely for various optimization problems. PID controller is the most widely used controller in the industry and there are many classical methods available to tune its parameters to achieve stability of a system. BFO is employed to search for optimal controllers parameters to minimize certain performance index for DC motor model. The time domain analysis, frequency domain analysis and robustness analysis are obtained for system controlled by optimal PID values obtained using BFO algorithm.

7.2- BASIC BACTERIAL FORAGING OPTIMIZATION (BFO):

Natural selection tends to eliminate animals with poor foraging strategies and favor the propagation of genes of those animals that have successful foraging strategies, since they are most likely to enjoy reproductive success. After many generations, poor foraging strategies are either eliminated or shaped into good ones. This activity of foraging led the researchers to use it as optimization process. Based on the researches on the foraging behavior of E. Coli bacteria, Prof. K. M. Passino proposed a new evolutionary computation technique known as Bacterial Foraging Optimization Algorithm (BFOA). In BFOA, the foraging (methods for locating, handling and ingesting food) behavior of E. Coli bacteria is mimicked.

The *E. coli* bacteria that are present in our intestines have a foraging strategy governed by four processes, namely chemotaxis, swarming, reproduction, and elimination and dispersal. The E. Coli bacteria moves through swimming and tumbling and this process is known as Chemotaxis. The Chemotactic movement is continued until a bacterium goes in the

direction of positive nutrient gradient (i.e. increasing function). After a certain number of complete swims, the best half of the population undergoes reproduction, eliminating rest of the population. In order to escape local optima, an elimination-dispersion event is carried out, where some bacteria are liquidated at random with a very small probability and new replacements are initialized at random. These four process are discussed below in detail.

Chemotaxis: This process is achieved through swimming and tumbling. Depending upon the rotation of the flagella in each bacterium, it decides whether it should move in a predefined direction (swimming) or an altogether different direction (tumbling), in the entire lifetime of the bacterium. To represent a tumble, a unit length random direction, $\phi(j)$ say, is generated; this will be used to define the direction of movement after a tumble. In particular,

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\phi(j) \qquad \text{... (7.1)}$$

where $\theta^i(j,k,l)$ represents the *i*th bacterium at *j*th chemotactic *k*th reproductive and *l*th elimination and dispersal step. *C*(*i*) is the size of the step taken in the random direction specified by the tumble. "*C*" is termed as the "run length unit".

Swarming: It is always desired that the bacterium that has searched the optimum path of food should try to attract other bacteria so that they reach the desired place more rapidly. Swarming makes the bacteria congregate into groups and hence move as concentric patterns of groups with high bacterial density. Mathematically, swarming can be represented by

$$J_{cc}(\theta^i(j,k,l)) = \sum_{i=1}^{s}[-d_{attractant}\exp(-w_{attractant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)]$$

$$+\sum_{i=1}^{s}[h_{repellant}\exp(-w_{repellant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] \qquad \text{... (7.2)}$$

Where $J_{cc}(\theta^i(j,k,l))$ is the cost function value to be added to the actual cost function to be minimized to present a time varying cost function. "*S*" is the total number of bacteria. "*p*" is

the number of parameters to be optimized that are present in each bacterium. $d_{attractant}$, $w_{attractant}$, $h_{repellant}$, and $w_{repellant}$ are different coefficients that are to be chosen judiciously.

Reproduction: The least healthy bacteria die, and the other healthiest bacteria each split into two bacteria, which are placed in the same location. This makes the population of bacteria constant.

Elimination and Dispersal: It is possible that in the local environment, the life of a population of bacteria changes either gradually by consumption of nutrients or suddenly due to some other influence. Events can kill or disperse all the bacteria in a region. They have the effect of possibly destroying the chemotactic progress, but in contrast, they also assist it, since dispersal may place bacteria near good food sources. Elimination and dispersal helps in reducing the behavior of *stagnation* (i.e., being trapped in a premature solution point or local optima).

7.3- BACTERIAL FORAGING OPTIMIZATION ALGORITHM (BFOA)

The algorithm to search optimal values of parameters is:-

STEP 1:- Initialize parameters $n, N, N_c, N_s, N_{re}, N_{ed}, P_{ed}, C(i)(i = 1, 2.....N), \theta^i$ where

n: Dimension of the search space,

N : The number of bacteria in the population,

N_c: Chemotactic steps,

N_s: Maximum number of swim length,

N_{re}: The number of reproduction steps,

N_{ed} : The number of elimination and dispersal events,

P_{ed} : Elimination and dispersal with probability,

$C(i)$: The size of step taken in the random direction specified by the tumble

$\theta^i(j,k,l)$: Position of the i-th bacterium at j-th chemotactic, k-th reproductive and l-th elimination-dispersal step.

STEP 2:- Eliminate dispersal loop: $l = l+1$;

STEP 3:- Reproductive step: $k = k+1$;

STEP 4:- Chemotaxis loop: $j = j+1$

For i=1,2... N , take a chemotactic step for bacterium as follows:

Compute fitness function, $J(i, j, k, l)$

Let $$J(i, j, k, l) = j(i, j, k, l) + J_{cc}(\theta^i(j,k,l), P(j,k,l)) \quad \text{... (7.3)}$$

Where $$J_{cc}(\theta^i(j,k,l)) = \sum_{i=1}^{s}[-d_{attractant}\exp(-w_{attractant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)]$$

$$+\sum_{i=1}^{s}[h_{repellant}\exp(-w_{repellant}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] \quad \text{... (7.4)}$$

And $\theta = [\theta_1, \theta_2 .....\theta_D]^T$ is a point in D-dimensional space.

Let $J_{last} = J(i, j, k, l)$ to save this value since we may find a better cost via a run.

Tumble: generate a random vector $\Delta(i) \in R^n$ with each element $\Delta_m(i)$, $m = 1, 2, ...p$, a random number on [-1,1].

Move: Let $$\theta(i+1, j, k) = \theta(i, j, k) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad \text{... (7.5)}$$

This results in step size $C(i)$ for bacterium i.

Compute $J(i, j+1, k, l)$

Swim

Let $m = 0$ (counter for swim length)

While $m < N_s$ (if have not climbed down too long).

Let $m = m + 1$;

If $J(i, j, k, l) < J_{last}$ (if doing better),

Let $J_{last} = J(i, j+1, k, l)$ and

$$\theta(i+1, j, k) = \theta(i+1, j, k) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \quad \text{... (7.6)}$$

Use this to calculate the new $J(i, j+1, k, l)$ as in (f);

Else if $m = N_s$. This is end of while statement.

Go to next bacterium $(i+1)$ if $i \neq N$ (i.e., go to (b))

STEP 5:- If $j < N_c$, go to step 3. In this case, continue chemotaxis, since the life of bacteria is not over.

STEP 6:- Reproduction:

For a given k and l, and for each $i = 1, 2....N$, let $J^i_{health} = \sum_{j=1}^{N_c+1} J(i, j, k, l)$ is the health of bacteria. Sort bacteria and chemotactic parameters $C(i)$ in ascending order of cost. J_{health} (higher cost means lower health).

The Sr bacteria with highest J_{health} value die and the remaining Sr bacteria with the best value split.

STEP 7:- If $k < Nre$, got to [STEP 3]. In this case, we have not reached the number of specified reproduction steps, so next generation of the chemotactic loop is started.

STEP 8:- Elimination-Dispersal: For $i = 1, 2, ...N$, with probability P_{ed}, eliminate and disperse each bacterium, and this results in keeping the number of bacteria constant. To do this, if a bacterium is eliminated, simply disperse one at a random location. If $l < N_{ed}$, then go to **[STEP 3]**; otherwise end.

6.4- OPTIMAL PID TUNING USING BFO

BFO algorithm is used to obtain the optimal values for PID controller parameters for our DC motor model so as to keep the operation of entire system within specified range of stability. Just like in PSO algorithm, the multi-objective function in BFO also uses only one of many error indices along with overshoot for optimization as shown below.

Objective function= Alpha*overshoot + beta*Error Indices (alpha=10, beta=10)

The figures (7.1 a), (7.2a), (7.3a) and (7.4a) show the step response of controlled system using BFO algorithm with IAE, ISE, ITAE and ISTE in the multi-objective function respectively.

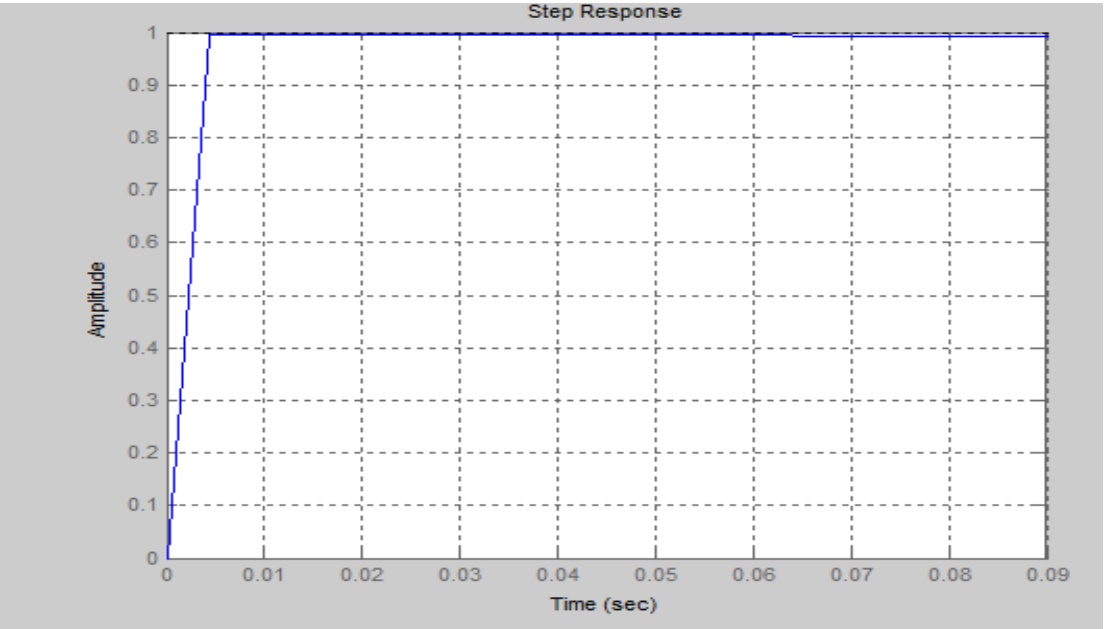

Figure 7.1(a) - Step response of controlled system when Multiobjective function of BFO uses IAE+overshoot

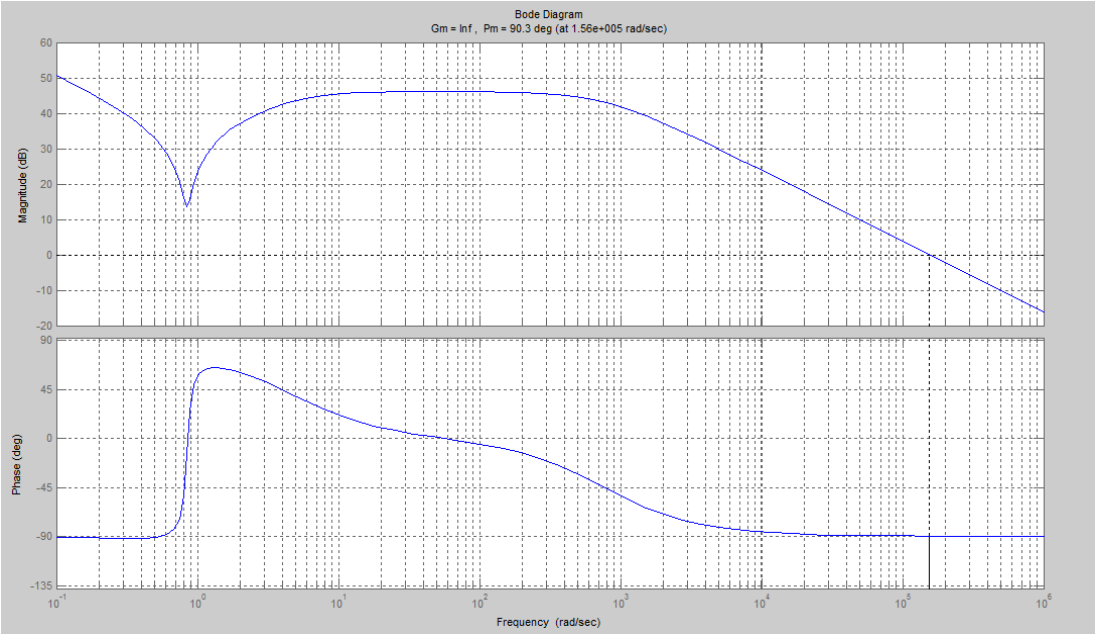

Figure 7.1(b) – Bode plot for controlled system when Multiobjective function of BFO uses IAE+overshoot

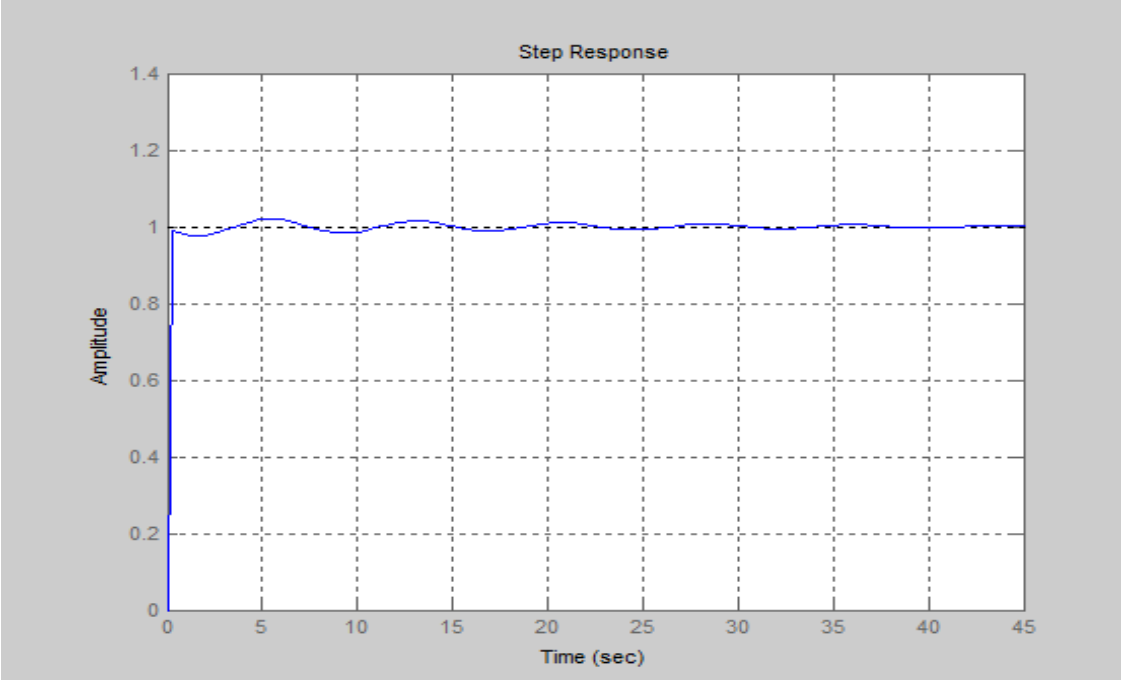

Figure 7.2(a) - Step response of controlled system when Multiobjective function of BFO uses ISE+overshoot

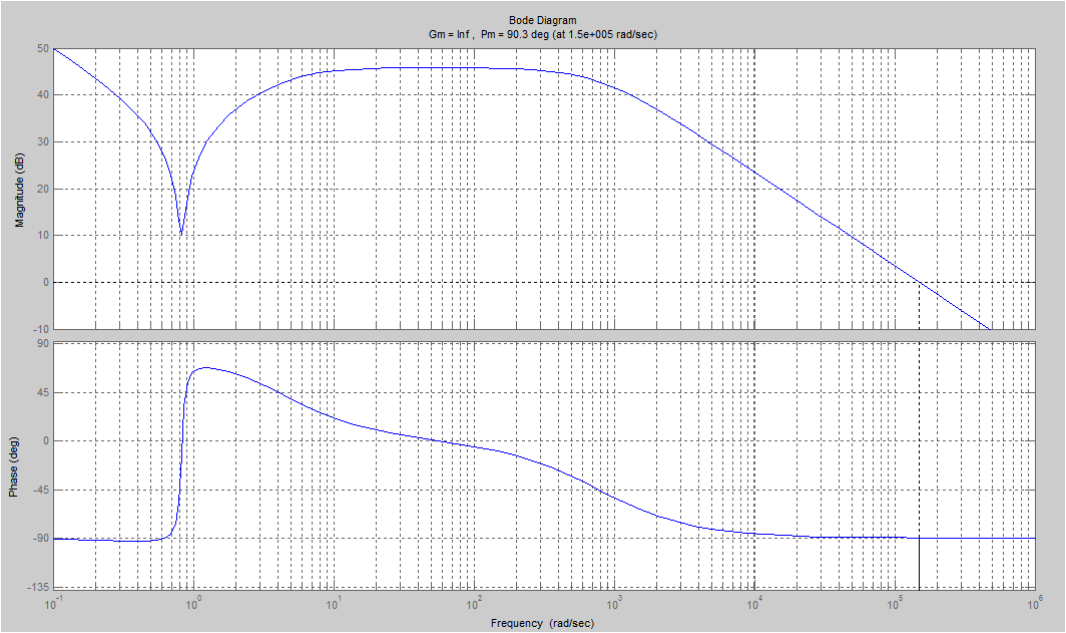

Figure 7.2(b) – Bode plot for controlled system when Multiobjective function of BFO uses ISE+overshoot

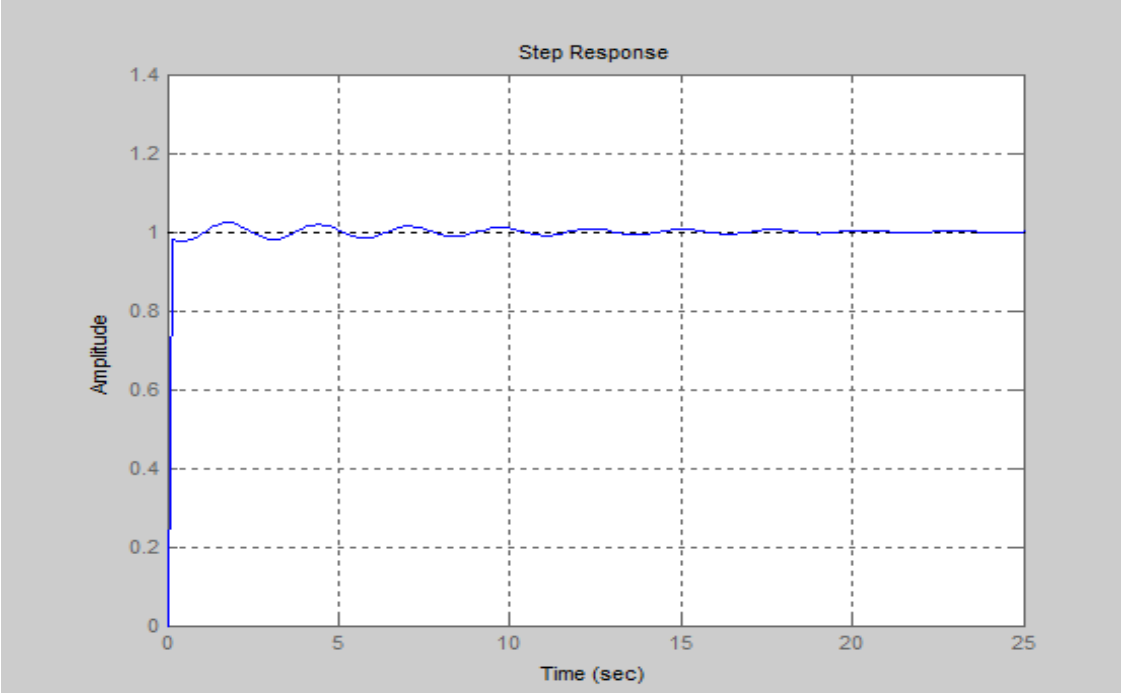

Figure 7.3(a) - Step response of controlled system when Multiobjective function of BFO uses ITAE+overshoot

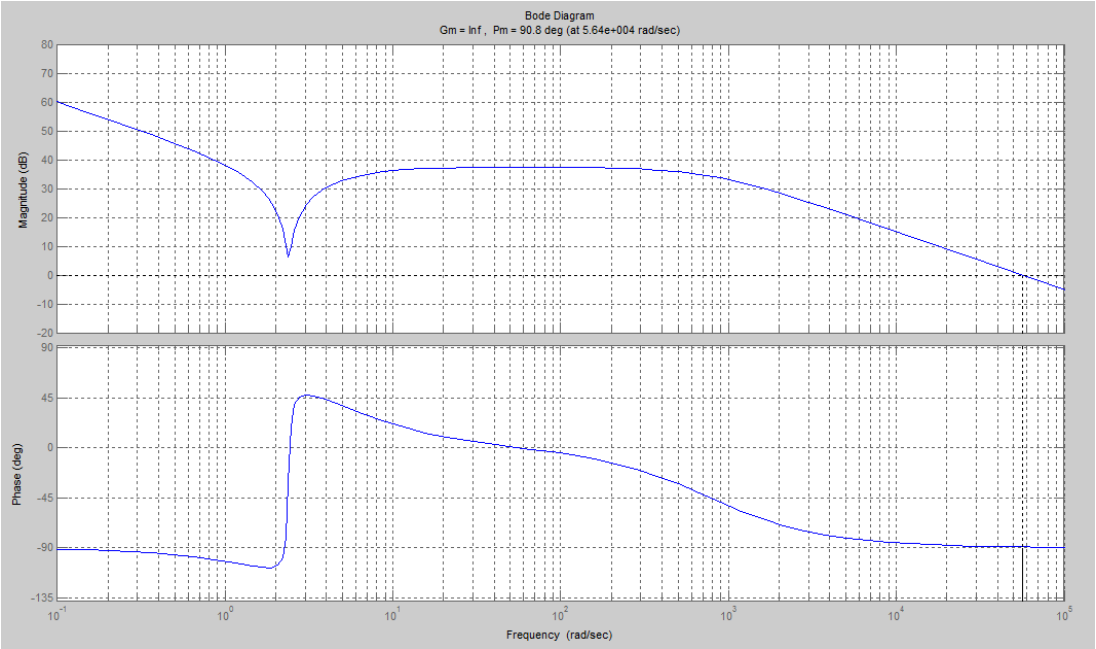

Figure 7.3(b) – Bode plot for controlled system when Multiobjective function of BFO uses ITAE+overshoot

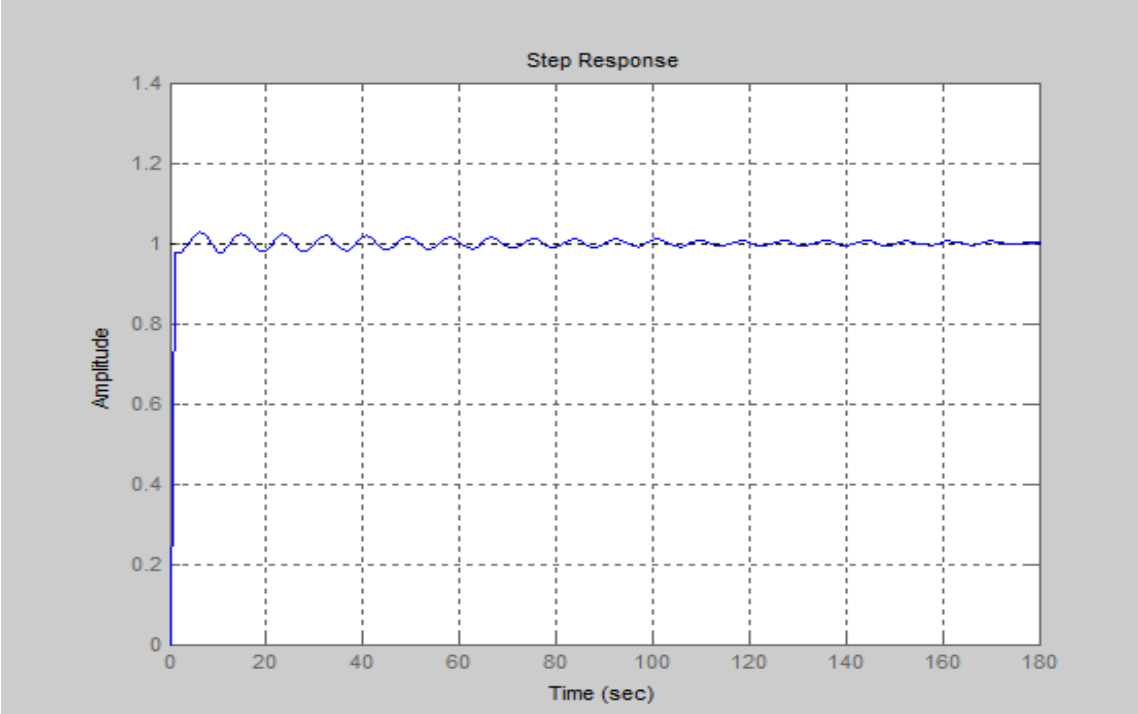

Figure 7.4(a) - Step response of controlled system when Multiobjective function of BFO uses ISTE+overshoot

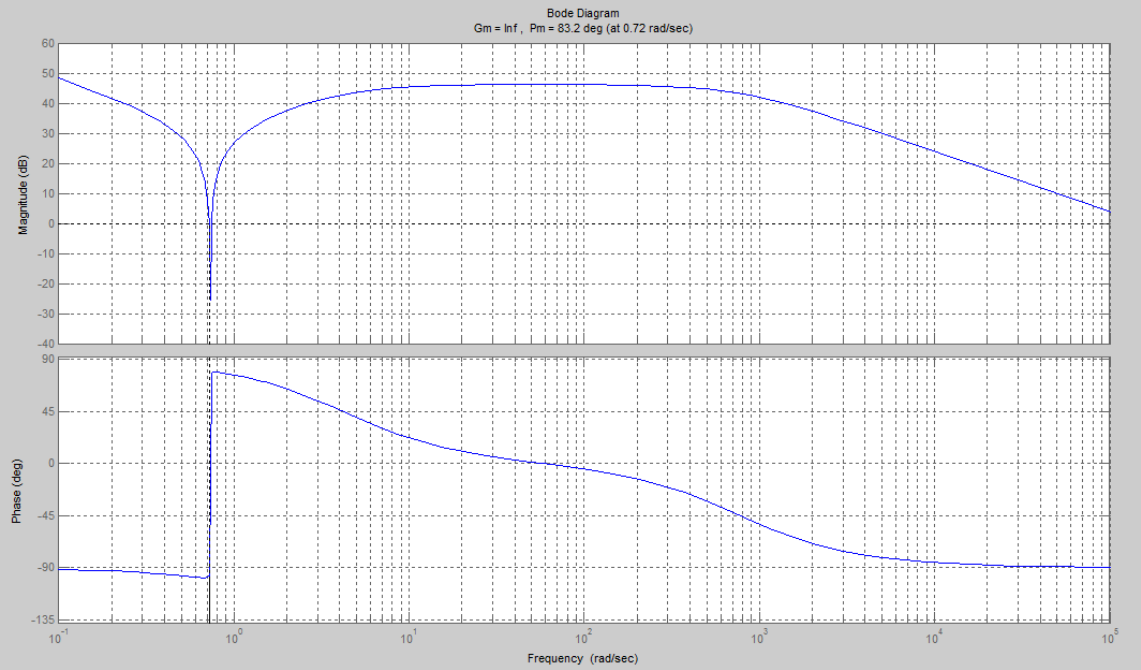

Figure 7.4(b) - Step response of controlled system when Multiobjective function of BFO uses ISTE+overshoot

6.5- RESULTS

The parameters for BFO algorithm used in optimization for DC motor model are given below:

```
p=3;            % dimension of search space
s=10;           % The number of bacteria
Nc=5;           % Number of chemotactic steps
Ns=4;           % Limits the length of a swim
Nre=4;          % The number of reproduction steps
Ned=2;          % The number of elimination-dispersal events
Sr=s/2;         % The number of bacteria reproductions (splits)
                  per generation
Ped=0.25;       % The probability that each bacteria will
```

Table 7.1 below shows the optimal PID parameters obtained for DC motor model using BFO algorithm along with the time domain analysis of the system controlled with these optimal PID controller values.

Table 7.1- BFO algorithm based optimal PID parameters and time domain analysis of controlled system.

Error Indices+ Overshoot	**PID Parameters**			**Time Domain Analysis**		
	Kp	Ki	Kd	Rise Time	Settling Time	Overshoot
IAE	0.0491	0.3482	0.4905	0.0035	0.0043	0
ISE	0.0336	0.3223	0.4740	0.2333	5.5852	2.0274
ITAE	0.0235	1.0106	0.1777	0.1159	1.9545	2.2589
ISTE	5.3963e-004	0.2626	0.4927	1.0094	32.0118	2.6551

Table 7.2 below shows the frequency domain analysis of the controlled system along with robustness analysis. Robust parameters like modulus margin and delay margin are necessary to fall in specified range if system has to tackle uncertainties due to load variations.

Table 7.2- Frequency domain and robustness analysis for controlled system obtained by BFO algorithm.

Error Indices+ Overshoot	**Frequency Domain Analysis**		**Robustness Analysis**	
	Gain Margin	Phase margin	Modulus margin	Delay Margin
IAE	Inf	90.2836	-940.4465	1.0210e-005
ISE	Inf	90.2935	-1.0067e+003	1.0473e-005
ITAE	Inf	90.7828	-7.0431+003	2.8090e-005
ISTE	Inf	90.2824	-932.1073	1.0074e-005

7.6- CONCLUSION

From table (7.1) and (7.2) the BFO algorithm based optimized PID parameters and their performance in DC machine model for time domain, frequency domain and robustness are observed. For BFO algorithm, these results show that:

- IAE gives lowest rise time, settling time and 0 overshoot.
- Next result of ITAE for rise time and settling time are second best after IAE,but for overshoot ISE is second after IAE.
- ISTE gives worst performance in case of the selected DC motor model.
- BFO algorithm gives Kp, Ki, Kd values in lesser time as compared to PSO algorithm.
- Both frequency domain analysis and robustness analysis are not achievable within required range.

CHAPTER 8:-

CONCLUSION AND FUTURE SCOPE

In this thesis, various classical PID tuning methods and Optimal PID tuning methods based on performance indices are discussed and they are implemented for the selected DC motor model for servo speed control. Their results are observed and discussed. Next PID controller is designed using AI based optimization techniques namely Particle Swarm Optimization (PSO) algorithm and Bacterial Foraging Optimization (BFO) algorithm for better servo speed control of DC motor model.

The results of open loop and close PID tuning methods suggest that frequency domain analysis (gain margin) cannot be achieved within suggested range for DC motor model. Also there is a compromise in overshoot or rise time, settling time while making choice between two classical categories of PID tuning. Robust parameter (like modulus margin) is not within required range for DC motor model using Classical PID tuning techniques. So controller cannot deal with uncertainties caused due to load change in case of selected DC motor model.

Optimal PID Tuning technique based on performance indices provide flexibility to range of rise time, settling time along with 0 overshoot for DC motor model and thus proves better to the two categories Classical PID tuning approach. Frequency domain (both gain margin and phase margin) as well as robustness analysis (modulus margin) are still not obtained within specified range by using Optimal PID Tuning method.

After classical approach and Optimal Tuning method, now PSO algorithm is applied for obtaining optimal values for PID parameters for better system performance. Using PSO, we are able to achieve 0 overshoot with good rise time and settling time values.

BFO algorithm is next artificial optimization technique applied for optimization of PID parameters for DC motor model servo speed control. Overshoot , rise time and settling time are achievable within specified range but like PSO overshoot is not 0.

BFO algorithm calculates optimal PID values much faster than PSO. PSO on the other hand is much simpler in structure as compared to BFO. Frequency domain analysis(gain margin) does not fall within specified range of stability for any technique used for DC motor model, while none of the robustness parameters was obtained within specified range when BFO and PSO are used.

PSO and BFO can be further modified for more accurate adaptation of Swarm Intelligence for further studies in this field as well as other fields also. Micro-BFO algorithm, Smart BFO algorithm are some of the modified BFO approaches with fast speed and better performance and can be used for DC motor model speed control. A hybrid approach combining the best qualities of PSO and BFO algorithms can also be applied for better speed control of DC motor model. Here ideal parallel PID controller form is used. This work can be extended further by using other related PID controller configurations like series PID, ideal PID with first order lag etc. for DC motor speed control. PSO and BFO are very good performers when time domain analysis is considered, but their performance for frequency domain and robustness analysis is not very good. So this can be taken as a research work in future so that PSO and BFO gives optimum performance in all three fields of time domain, frequency domain and robustness analysis.

REFERENCE

[1]- Zafer Bingul, Oguzhan Karahan; '*Tuning of Fractional PID Controllers Using PSO Algorithm for Robot Trajectory Control*';International Conference on Mechatronics, 2011, pp.955-960.

[2]- V. K. Kadiyala; R. K. Jatoth; S. Pathalaiah; '*Design and implementation of fractional order PID controller for Aero fin control system*'; IEEE; NaBIC 2009; pp. 696-701.

[3]- C. H. Liu; Y. Y. Hsu; 'Design of a Self-Tuning PI Controller for a STATCOM Using Particle Swarm Optimization'; IEEE 2010; pp. 702-715.

[4]- D. Maiti, A. Acharya, M. Chakraborty, A. Konar, R. Janarthanan; 'Tuning PID and $PI^{\lambda}D^{\delta}$ Controllers using the Integral Time Absolute Error Criterion'; IEEE 2008; pp. 457- 462.

[5]- C. M. Lin, M. H. Lin, C. H. Chen, D. S. Yeung; 'Robust PID control system design for chaotic systems using particle swarm optimization algorithm'; IEEE 2009; pp. 3294-3299.

[6]- W.W. Cai, L. X. Jia, Y. B. Zhang, N. Ni; 'Design and simulation of intelligent PID controller based on particle swarm optimization'; IEEE 2010.

[7]- Y. Bo, L. W. Zhou, Y. Feng; 'A New PSO-PID Tuning Method for Time-delay Processes'; IEEE 2008.

[8]- A. A. A. El-Gammal, A. A. El- Samahy; 'A Modified Design of PID Controller For DC Motor Drives Using Particle Swarm Optimization PSO'; IEEE 2009; pp. 419-424.

[9]- V. Rajinikanth, K. Latha; 'I-PD Controller Tuning for Unstable System using Bacterial Foraging Algorithm: A study based on various Error Criterion ', Applied Computational Intelligence and Soft Computing, Volume 2012.

[10]- E. Salim Ali, S. M. Abd-Elazim; '*Optimal PID Tuning for Load Frequency Control Using Bacteria Foraging Optimization Algorithm*', 14th International Middle East Power System Conference (MEPCON'10), December 2010, pp. 410-415.

[11]- T. Jain, M. J. Nigam; *'Optimization of PD-PI Controller Using Swarm Intelligence'*, International Journal of Computational Cognition, December 2008, Vol. 6, No. 4, pp. 55-59.

[12]- D. H. Kim, J. H. Cho; '*A Biologically Inspired Intelligent PID Controller Tuning for AVR Systems*', International Journal of Control, Automation and Systems, 2006, Vol. 4, No. 5, pp. 624-636.

[13]- W. Korani; 'Bacterial Foraging Oriented by Particle Swarm Optimization Strategy for PID Tuning', GECCO'08, July, 2008, pp. 1823-1826.

[14]- T. Jain, V. Patel, M. J. Nigam, *'Implementation of PID Controlled SIMO Process on FPGA Using Bacterial Foraging for Optimal Performance*', International Journal of Computer and Electrical Engineering, Vol. 1, No. 2, June 2009, pp. 107-110.