Sudan University of Science & Technology

College of Graduate studies

Auto Tuning Of Proportional Integral Derivative Controller Using Artificial Bee Colony Algorithm for Dc Motor Speed Control

التوليف التلقائي للمتحكم التناسبي التكاملي التفاضلي بإستخدام خوارزمية مجتمع النحل الإصطناعي للتحكم بمحرك التيار المستمر

A Research Submitted in Partial fulfillment for the Requirements of the Degree of M.Sc. in Computer and network Engineering.

Prepared by:

Shaymaa Mohammed Abd alRaheem

Supervised By:

Dr: Mohammed Alnour

May 2021

الإستهلال

بسم الله الرحمن الرحيم

قال تعالى:

"وَأَوْحَىٰ رَبُّكَ إِلَى النَّحْلِ أَنِ اتَّخِذِي مِنَ الْجِبَالِ بُيُوتًا وَمِنَ الشَّجَرِ وَمِمَّا يَعْرِشُونَ (68) ثُمَّ كُلِي مِن كُلِّ الثَّمَرَاتِ فَاسْلُكِي سُبُلَ رَبِّكِ ذُلُلًا يَخْرُجُ مِن بُطُونِهَا شَرَابٌ مُّخْتَلِفٌ أَلْوَانُهُ فِيهِ شِفَاءٌ لِّلنَّاسِ إِنَّ فِي ذَٰلِكَ لَآيَةً لِّقَوْمٍ يَتَفَكَّرُونَ(69) "

سورة النحل آية[68-69]

Dedication

I dedicate this work

To the candle which burns to light my life

My mother

To the source of inspiration

My father

To those who have made this work it possible

My teachers

To those who encouraged me

Sisters, brothers and friends

Shaymaa...

Acknowledgments

Firstly I would like to give thanks to Allah

I want to thanks my Supervisor

Dr. Mohammed Alnour

Who has given me his time and encouraged to do this work

Thank you Dr. Alaa Aldeen and Eng. Ayat younis

Finally all of thanks go to my colleagues who supported me

to made this work possible

Abstract

Since DC motor plays a significant role in modern industry the aim of this work is to design a speed controller of a DC motor by selection of PID parameters using Artificial Bee Colony algorithm. This algorithm is come under a category of bio-inspired optimization techniques. The model of a DC motor is considered as a second order system for speed control. Here there is a comparison between conventional tuning methods and optimization techniques of parameters for PID controller.

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. The purpose of a motor speed controller is to take a signal representing the demanded speed, and drive a motor at that speed.

The PID conventional controller had been applied and results were compared with the automatic tuning ABC-PID for DC motor speed control using Simulink of MATLAB. The DC Motor Scheduling PID-ABC controller is modeled in MATLAB environment. Simulation results for the proposed method gave optimum solution compare with the results that we got from Ziegler-Nichols method, whereas the overshoot in Z-N equal 47% but in ABC algorithm method the overshoot = 0%.

المستخلص

يلعب محرك التيار المستمر دورًا مهمًا في الصناعة الحديثة لذا الهدف من هذا العمل هو تصميم جهاز تحكم في السرعة لمحرك التيار المستمر عن طريق اختيار معلمات المتحكم التناسبي التكاملي التفاضلي باستخدام خوارزمية مستعمرة النحل الاصطناعي .تأتي هذه الخوارزمية ضمن فئة من تقنيات التحسين المستوحاة من الحيوية. يعتبر نموذج محرك التيار المستمر بمثابة نظام من الدرجة الثانية للتحكم في السرعة، وهنا مقارنة بين أساليب الضبط التقليدية وأساليب تحسين المعلمات لوحدة التحكم التناسبي التكاملي التفاضلي .

تطبيق خوارزمية التحسين على وحدة المتحكم التناسبي التكاملي التفاضلي يجعله يعطي إخراج مثالي من خلال البحث عن أفضل مجموعة من الحلول لمعلمات المتحكم التناسبي التكاملي التفاضلي. والغرض من وحدة التحكم في سرعة المحرك هو أخذ إشارة تمثل السرعة المطلوبة ، وقيادة المحرك بهذه السرعة.

تم تطبيق وحدة المتحكمات التناسبية التكاملية التفاضلية التقليدية وتمت مقارنة النتائج مع التوليف التلقائي لمعاملات المتحكم التناسبي التكاملي التفاضلي بإستخدام خوارزمية مستعمرة النحل الإصطناعي وبإستخدام برنامج المحاكاة (الماتلاب) أعطت نتائج المحاكاة للطريقة المقترحة الحل الأمثل.

Table of Contents

Chapter Three: System Model

Chapter Four:Result and Discussion

Chapter Five: Conclusion and Recommendations

List of Figures

Figures **Pages**

List of Tables

Tables **pages**

List of Abbreviations

ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AIS	Artificial Immune Systems
BFO	Bacterial Foraging Optimization
DC	direct current
DE	Differential Evolution
GA	Genetic Algorithm
ITAE	Integral Time Absolute Error
ISE	Integral Square error
ITSE	Integral Time square Error
IAE	Integral Absolute Error
PID	Proportional Integral Derivative
PSO	Particle Swarm Optimization
P_{cr}	Critical period
Z-N	Ziegler Nichols

List of Symbols

R_a: Armature resistance

La: Armature inductance

Ia : Armature current

R_f : Field winding resistance

Lf : Field winding inductance

I_f: Field current

e_a: Input voltage

eb : Back electromotive force (EMF)

T_m : Motor torque

ω: An angular velocity of rotor

J: Rotating inertial measurement of motor bearing

Kb : EMF constant

KT: Torque constant

B: Friction constant

Va : Armature voltage

Vf : Field winding voltage

Kp: proportional tuning constant

Ki: integral tuning constant

Kd: derivative tuning constant

Chapter One

Introduction

1.1Background:

The direct current (DC) motor is a device that used in many industries in order to convert electrical energy into mechanical energy. This is all result from the availability of speed controllers is wide range, easily and many ways. In most applications, speed control is very important. For example, if we have DC motor in radio controller car, if we just apply a constant power to the motor, it is impossible to maintain the desired speed. It will go slower over rocky road, slower uphill, faster downhill and so on. So, it is important to make a controller to control the speed of DC motor in desired speed. [1]

DC motor plays a significant role in modern industry. The purpose of a motor speed controller is to take a signal representing the demanded speed, and to drive a motor at that speed. There are numerous applications where control of speed is required, as in rolling mills, cranes, hoists, elevators, machine tools, transit system and locomotive drives. These applications may demand high-speed control accuracy and good dynamic responses. The control of DC motor uses the digital signal processing system. Proportional Integral Derivative (PID) controller has been widely used for processes and motion control system in industry. Now more than 90% of control systems are still with PID controllers. [1]

To implement these controllers, three parameters which are the proportional, integral and derivative gains should be tuned to provide stable response as well as minimum error while tracking the input.

However, computing the gains does not always give best results because the tuning criteria presume a one-fourth reduction in the first two-peaks that's why system stability here is matter of unreliable stability that's why we are adopting ABC (Artificial Bee Colony algorithm) here.

In the ABC algorithm, the colony of artificial bees contains three groups of bees: employed bees, onlookers and scouts. A bee waiting on the dance area for making decision to choose a food source is called an onlooker and a bee going to the food source visited by itself previously is named an employed bee. A bee carrying out random search is called a scout. In the ABC algorithm, first half of the colony consists of employed artificial bees and the second half constitutes the onlookers. For every food source, there is only one employed bee. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source is exhausted by the employed and onlooker bees becomes a scout.[2]

1.2 Problem statement:

Proportional Integral Derivative (PID) controller has been widely used for processes and motion control system in industry. The most critical step in the application of PID controller is the accurate and efficient tuning of parameters. The accidental change of different parameters makes it difficult to tune or control the system.

1.3 Proposed solution:

Use Artificial Bees Colony (ABC) optimization algorithm for Auto tuning of PID controller parameters for DC motor. Improve performance of DC motor.

The efficiency of Control Algorithm is presented through a simulation, and compared with the quality of PID controller.

1.4 Objectives:

- The main of this project is to control the speed of DC motor using Artificial Bee Colony Algorithm to tune PID controller.
- To improve the performance of DC motor behavior based on PID controller.
- To improve PID controller behavior using ABC auto–tuning algorithm.
- To simulate the system for performance evaluation.
- Performance evaluation for the system by comparing proposed tuning method with traditional methods.

1.5 Methodology:

DC Motor transfer function was carried out and PID controller was used to control the DC motor speed. Firstly; PID was tuned using traditional method. The second step was tuned using ABC algorithm. The system was tested under different condition and the result was carried out with different scenarios. To simulate the proposed system MATLAB, SIMULINK will be used.

This research meanly focuses on how tuning PID controller using ABC algorithm, different tuning methods will be covered. The system under study is DC motor.

1.6 Thesis outlines:

Chapter1: introduction, which gives a brief background and stated the problem along with the proposed solution.

Chapter2: literature review, it gives a comprehensive study for the Components used in the design.

Chapter3: System Design mainly discuses on the system design of the project.

Chapter4: Simulation and Discussion result, it was presented the results of the project.

Chapter5: conclusion and recommendation, Concludes overall about the project and future recommendation was also discussed in this chapter.

Chapter Two

Literature Review

2.1 Overview:

There are many reasons for the practical deficiencies of control systems. Unconsidered conditions or any changes in environment may result in undesired outputs in a control system. For example, if the model of the process is inaccurate, model-based control can provide unsatisfactory results. Even with an accurate model, approximations are applied if parameter values are partially known or vague.

Algorithmic control based on such incomplete information will not usually give satisfactory results. Often, the environment with which the process interacts may not be completely predictable and it is normally not possible for a hard control to respond accurately to a condition that it did not anticipate.[3]

The Proportional-Integral-Derivative (PID) controller has been proved the most popular controller of this century for its remarkable effectiveness, easiness of implementation and vast applicability. But it is also hard to tune the PID controller. A number of tuning methods are done manually which are difficult and time consuming. For using PID controller efficiently, the optimal tuning of its parameters has become a significant research area. Optimization problems have been resolved with the aid of numerous techniques. Today's, an alternative approach to the traditional methods for operations research, meta-heuristic methods are implanted to simplify optimization difficulties.[4]

2.2 Related work:

According to Ali Eydgahi and Mohammad Fotouhi, a fuzzy knowledge-based to tune control parameters of a proportional-integral-derivative (PID) controller of a robot joint, The fuzzy knowledge-based is implemented as a set of fuzzy rules with an inference mechanism to tune the PID controller in the system. Software is developed in which users can define the rule base. The program generates the fuzzy decision table based on all inputted information and in a descriptive fashion. Then, the decision table is used to modify the parameters required by the fuzzy tuner in on-line operations. A simulation environment based on MATLAB® and SIMULINK® is used for demonstration purpose. Two control systems with the same structures are constructed. One system is using a fuzzy knowledge-based tuner and the other one is using a conventional PID control loop. Simulation results show improvement on system response of fuzzy knowledge-based control structure[3].

S. Pareek M. Kishnani and R.Gupta(2014) research about comparison between designing a PID controller by selection of PID parameters using Bacterial Foraging Optimization, Particle Swarm Optimization (PSO) and Genetic Algorithm. From the results, the designed PID controllers using BFO based optimization have less overshoot compared to other applied tuning algorithms. GA based optimization offers least rise time and settling time among all the introduced techniques. The obtained results have higher fitness and faster convergence. Results show that meta-heuristic algorithms are even efficient enough to optimize the non-linear and unstable systems which cannot be optimized at all by conventional methods like Ziegler-Nichols. [4]

According to Mohammed E. El telbany, was applied Artificial Bees Colony (ABC) optimization algorithm for regulation parameters of PID

controller of DC motors and compare the time domain performance of the PID controller for DC-motor system tuned by ABC algorithm. The ABC algorithm showed better performance than the other population based optimization.[5]

K. V. Lakshmi Narayana, Vaegae Naveen Kumar2, M. Dhivya1 and R. Prejila Raj(2015), their research is designed the tuning of PID controller with Ant Colony Optimization (ACO) results in optimum values of $k_{p,}k_{i,}\,k_d$ when compared with Ziegler Nichols technique for non-linear processes like conical tank. The desired response was achieved by optimizing the value of the PID tuning parameters. ACO provides good disturbance rejection as well as set point tracking than the conventional PID tuning.[6]

Dong Hwa Kim, Jae Hoon Cho, was designed an intelligent tuning method of PID controller by bacterial foraging based optimal algorithm is suggested for robust control with disturbance rejection function on control system of motor control loop. Simulation results are showed satisfactory responses. The object function can be minimized by gain selection for control. Artificial Immune Systems (AIS) are computational systems inspired by the principles and processes of the vertebrate immune system, which learns about the foreign substances to defend the body against them. The immune system has two types of responses, primary and secondary. The former is the response when it first encounters the antigen. In this period, the system learns about the antigen, creating a memory of it. The later occurs when the antigen is encountered for the second time, which is a more rapid and larger response. The cells primarily involved in this system are B cells. Against the antigen, the level to which a B cell is stimulated relates partly to how well its antibody binds the antigen.[7]

2.3 DC Motor Overview:

A motor is an electromechanical component that yields a displacement output for a voltage input, that is, a mechanical output generated by an electrical input. Its operation is based on principle that when a current carrying conductor is placed in a magnetic field, the conductor is experiences a mechanical force. The direction of the force is given by Fleming's left hand rule.

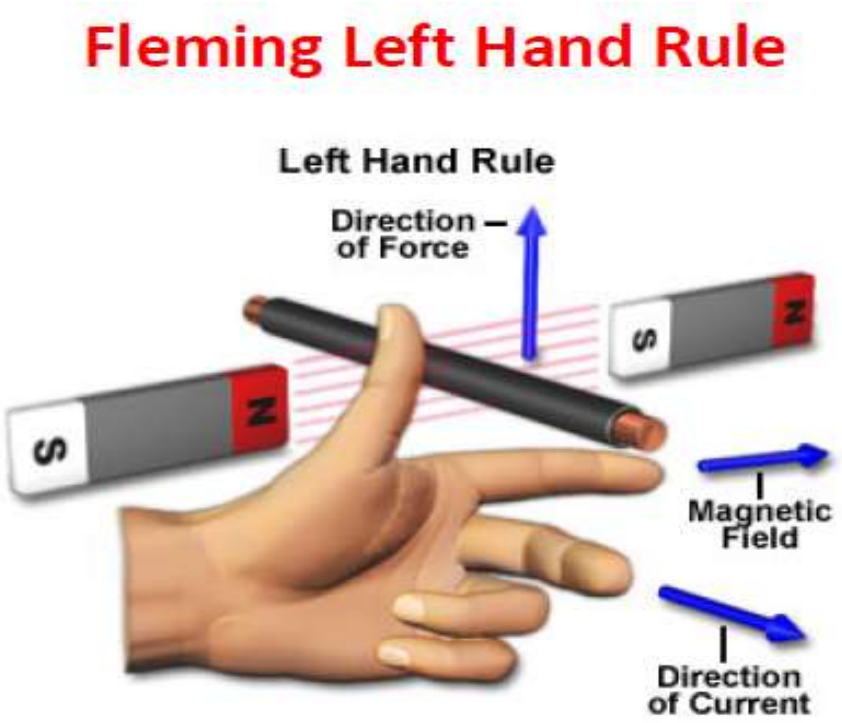

Figure 2.1: Fleming's left hand rule.

2.4 DC Machine Classification:

DC Machines can be classified according to the electrical connections of the armature winding and the field windings. The different ways in which these windings are connected lead to machines operating with different characteristics. The field winding can be either self-excited or separately-excited, that is, the terminals of the winding can be connected across the input voltage terminals or fed from a separate voltage source. Further, in self-excited motors, the field

winding can be connected either in series or in parallel with the armature winding.[8]

2.4.1Separately excited machines

- The armature and field winding are electrically separate from each other.

- The field winding is excited by a separate DC source.

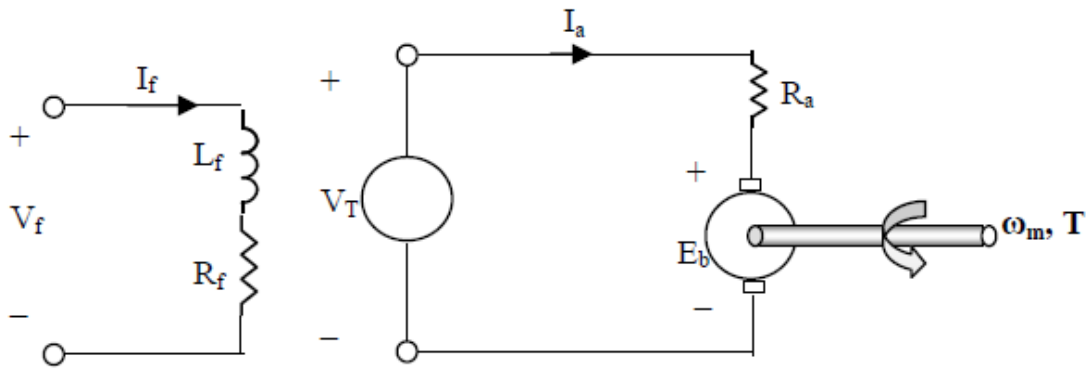

Figure 2.2: Separately excited DC Motor [8]

Figure 2.1 showed a DC motor has two distinct circuits: Field circuit and armature circuit. The input is electrical power and the output is mechanical power. In this equivalent circuit, the field winding is supplied from a separate DC voltage source of voltage V_f. R_f and L_f represent the resistance and inductance of the field winding. The current I_f produced in the winding establishes the magnetic field necessary for motor operation. In the armature (rotor) circuit, V_T is the voltage applied across the motor terminals, I_a is the current flowing in the armature circuit, R_a is the resistance of the armature winding, and E_b is the total voltage induced in the armature.[8]

2.4.2 Self excited machines

In these machines, instead of a separate voltage source, the field winding is connected across the main voltage terminals.

- **Shunt machine**

  - The armature and field winding are connected in parallel.
  - The armature voltage and field voltage are the same

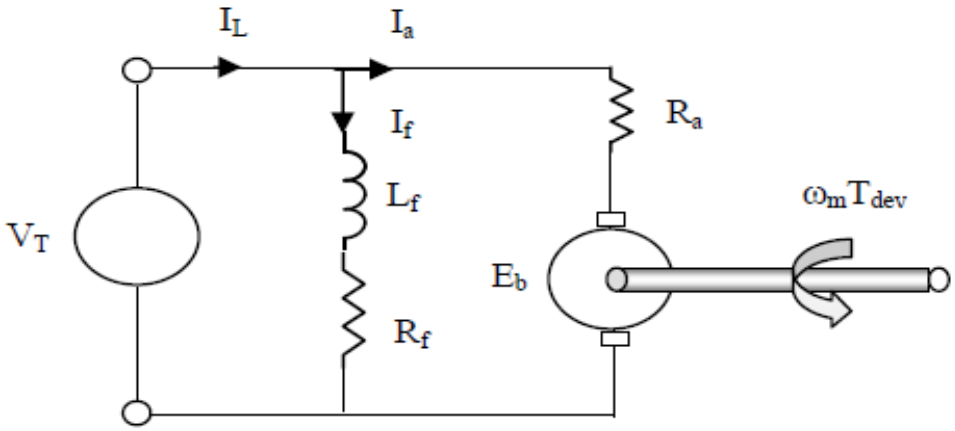

Figure 2.3: Shunt DC Motor [8]

Total current drawn from the supply, $I_L = I_f + I_a$

Total input power = $V_T I_L$

- **Series DC machine:**

  - The field winding and armature winding are connected in series.

  - The field winding carries the same current as the armature winding.

A series wound motor is also called a universal motor. It is universal in the sense that it will run equally well using either an ac or a dc voltage source. Reversing the polarity of both the stator and the rotor cancel out. Thus the motor will always rotate the same direction regardless of the voltage polarity

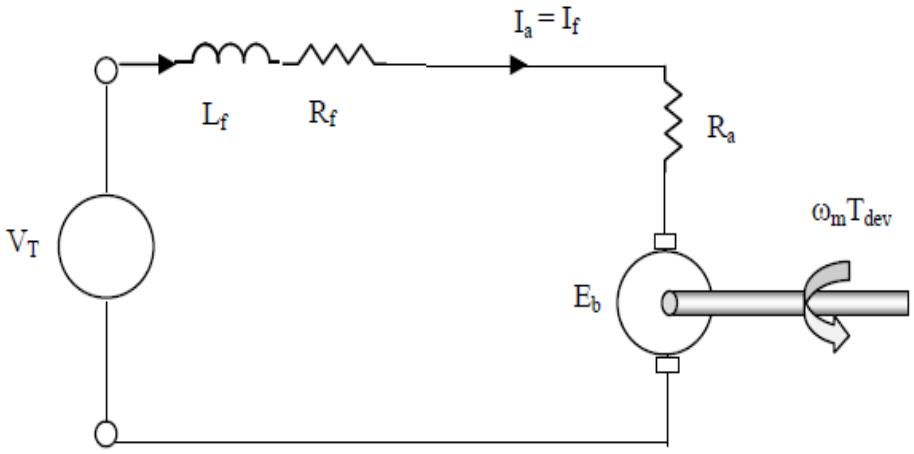

Figure 2.4: Series DC Motor [8]

- **Compound DC machine:**

If both series and shunt field windings are used, the motor is said to be compounded. In a compound machine, the series field winding is connected in series with the armature, and the shunt field winding is connected in parallel. Two types of arrangements are possible in compound motors:

Cumulative compounding - If the magnetic fluxes produced by both series and shunt field windings are in the same direction (i.e., additive), the machine is called cumulative compound.

Differential compounding - If the two fluxes are in opposition, the machine is differential compound. In both these types, the connection can be either short shunt or long shunt.[8]

2.5 PID Controller:

The basic principle in PID tuning is to adjust the controller parameters fast and accurately as possible for aims of the control design. These parameters are entitled by proportional tuning constant, Kp, integral tuning constant, Ki, and

derivative tuning constant, Kd. For a typical PID controller, the controller output signal u (t) can be written as the following in time and s domain:

$$u(t) = kp * e(t) + ki * \int_0^t e(t)dt + kd \frac{de(t)}{dt} \quad \textbf{(2.1)}$$

$$U(s) = \left(kp + \frac{ki}{s} + kd * s \right) * E(s) \quad \textbf{(2.2)}$$

Where e (t) is the error signal, and calculated by difference between the reference input r (t), and system output y (t); (s) is a complex variable which has a real part and an imaginary part. The variables of s and (1/s) in (2.2) can be considered as differential and integral operators in time domain, respectively. [9]

It is a well-known fact that parameter tuning approach directly affects behavior of the PID controller and satisfactoriness of the control system designed for the control aims. Therefore, in order to meet the desired closed loop system specifications in time and/or frequency domain, PID controller parameters must be carefully determined. For this purpose, one of the efficient approaches is to use the specifications of the system's step response. Definition of an appropriate objective function based on the use of important performance indicators such as the overshoot, settling time, and the integral time absolute error (ITAE) value can provide the remarkable solutions for the controller tuning. Properties of P, I and D are discussed briefly here:

2.5.1(P-controller) Proportional term:

This term speeds up the response as the closed loop time constant decreases with the proportional term but does not change the order of the system as the output is just proportional to the input. The proportional term minimizes but does not eliminate the steady state error, or offset.

Proportional Control Equations are as follows[10]:

- Set Position – Current Position $= (t)$

Motor Speed $= K_p * e\ (t)$ **(2.3)**

K_p= *proportional* gain and is the problem dependent.

$e\ (t)$ = Error.

2.5.2 Integral term:

This term eliminates the offset as it increases the type and order of the system by 1. This term also increases the system response speed but at the cost of sustained oscillations. The integral Control Equations are as follows:

- Set Position – Current Position $= (t)$
- Integral of e (t) = $\int e(t)dt$

$Ki * \int e(t)dt =$ Motor Speed **(2.4)**

Ki is integral gain.

2.5.3 Derivative term:

This term primarily reduces the oscillatory response of the system. It neither changes the type and order of the system nor affects the offset [11].The Derivative Control Equations are as follows:

- Set Position – Current Position $= e\ (t)$
- $Derivative\ of\ e(t) = \frac{d\ e(t)}{dt}$

Kd* $\frac{d\ e(t)}{dt}$ = Motor Speed **(2.5)**

- K_p determines how fast the object reaches the set point.

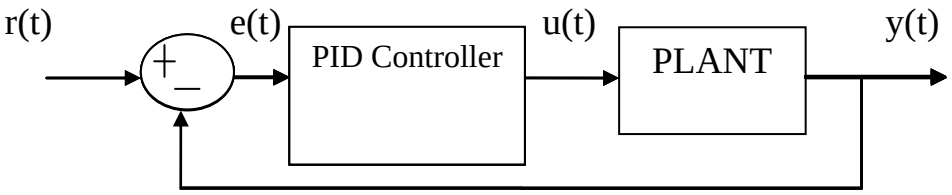

Figure 2.5: Feedback system architecture

Jitender Kaushal (2012) was explained in the table below the effect of parameters of PID controller on the response of the systems: [18]

Table 2.1: Effect of increasing parameter independently

Controller response	Rise Time	Overshoot	Settling Time	Steady State Error
Kp	Decrease	Increase	Small Change	Decrease
Ki	Decrease	Increase	Increase	Eliminate
Kd	Small Change	Decrease	Decrease	Small change

2.6 Transient Response of P, PI, PD, PID Controller:

2.6.1 P Controller:

P-I controller is mainly used to eliminate the steady state error resulting from P controller. However, in terms of the speed of the response and overall stability of the system, it has a negative impact. This controller is mostly used in areas where speed of the system is not an issue. Since P-I controller has no ability to predict the future errors of the system it cannot decrease the rise time and eliminate the oscillations. If applied, any amount of I guarantees set point overshoot.[12]

The main usage of the P controller is to decrease the steady state error of the system. When the proportional gain factor K increases the steady state error of the system decreases. However, despite the reduction, P control can never manage to eliminate the steady state error of the system.

- Increasing the proportional gain, it provides smaller amplitude and phase margin.
- Increasing the proportional gain leads to larger sensitivity to the noise.

We can use this controller only when our system is tolerable to a constant steady state error. In addition, it can be easily concluded that applying P controller P control also causes oscillation if sufficiently aggressive in the presence of lags and/or dead time. The more lags (higher order), the more problem it leads. Plus, it directly amplifies process noise.

2.6.2 P-I Controller:

P-I controller is mainly used to eliminate the steady state error resulting from P controller. However, in terms of the speed of the response and overall stability of the system, it has a negative impact. This controller is mostly used in areas where speed of the system is not an issue. Since P-I controller has no ability to predict the future errors of the system it cannot decrease the rise time and eliminate the oscillations. If applied, any amount of I guarantees set point overshoot.

2.6.3 P-D Controller:

The aim of using P-D controller is to increase the stability of the system by improving control since it has an ability to predict the future error of the system response. In order to avoid effects of the sudden change in the value of the error signal, the derivative is taken from the output response of the system

variable instead of the error signal. Therefore, D mode is designed to be proportional to the change of the output variable to prevent the sudden changes occurring in the control output resulting from sudden changes in the error signal. In addition D directly amplifies process noise therefore D-only control is not used.

2.6.4 P-I-D Controller:

P-I-D controller has the optimum control dynamics including: Zero steady state error, Fast response (short rise time), No oscillations and higher stability.

The necessity of using a derivative gain component in addition to the PI controller is to eliminate the overshoot and the oscillations occurring in the output response of the system. One of the main advantages of the P-I-D controller is that it can be used with higher order processes including more than single energy storage.

2.7 Importance of PID controller:

The aim in using the P-I-D controller is to make the actual motor speed match the desired motor speed. P-I-D algorithm will calculate necessary power changes to get the actual speed. This creates a cycle where the motor speed is constantly being checked against the desired speed. The power level is always set based on what is needed to achieve the correct results.

By using P-I-D controller, we can make the steady state error zero with integral control. We can also obtain fast response time by changing the P-I-D parameters. P-I-D is also very feasible when it is compared with other controllers.

2.8 Tuning of PID Controllers:

There have been various types of techniques applied for PID tuning, one of the earliest being the Ziegler Nichols technique. These techniques can be broadly classified as classical and computational or optimization techniques.

2.8.1Classical Techniques:

Classical techniques make certain assumptions about the plant and the desired output and try to obtain analytically, or graphically some feature of the process that is then used to decide the controller settings. These techniques are computationally very fast and simple to implement, and are good as a first iteration. But due to the assumptions made, the controller settings usually do not give the desired results directly and further tuning is required.

2.8.1.1Manual tuning:

This method consider for setting the parameters of a PID-mode controller manually. First a PID Controlling loop for the targeted Process is constructed. Then the KI, and Kd values are set to zero. Then increasing the KP value until the output of the loop oscillates, then the KP value is set to approximately half of this value. Then increasing KI until the steady state error is eliminated in sufficient time (the time must be as small as possible). Finally, increasing Kd until a small settling time (time from the start point until the error e is 0 in the loop) is observed.

2.8.1.2 Ziegler–Nichols method

This method was developed by John G. Ziegler and Nathaniel B. Nichols in the 1940‟ s. will discuss the Z-N method for those systems that can become unstable by using proportional control only. The basic steps in Z-M method are:

1. Set $k_i$and k_d equal to zero.

2. Slowly increase $k_p$to a value k_u at which we see sustained oscillations (constant amplitude and periodic).

3. Note the period of oscillation was denoted it by T_u.

4. Use the following values as the initial tuning constants.

Table 2.2: Z-N closed loop tuning parameters

Controller	K	Ti	Td
P	0.5 Ku	-	-
PI	0.4 Ku	0.8 Tu	-
PID	0.6 Ku	0.5 Tu	0.125 Tu

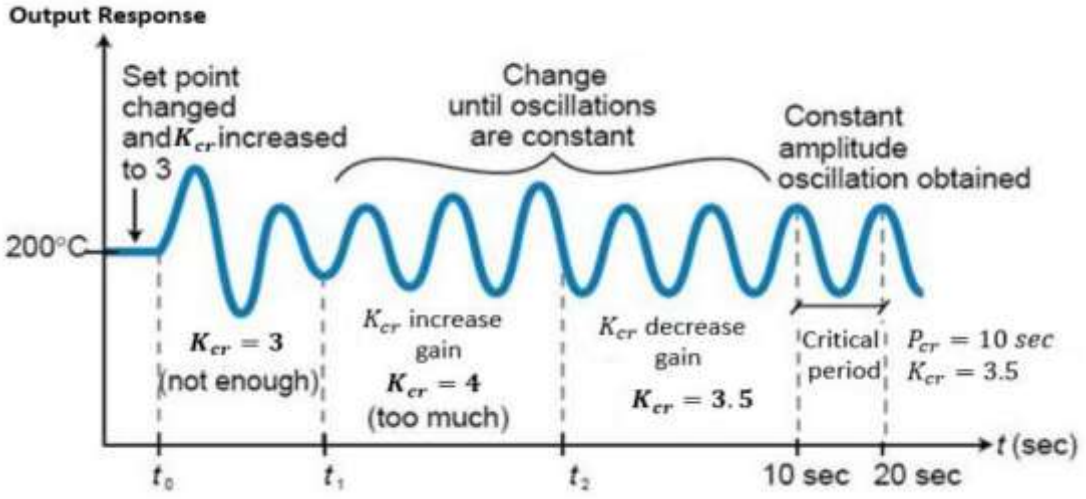

Figure 2.6: System tuned using the Ziegler-Nichols method

➢ Advantages Ziegler-Nichols Closed-Loop Tuning Methods :

1. Easy experiment; only need to change the P controller

2. Includes dynamics of whole process, which gives a more accurate picture of how the system is behaving

- Disadvantages Ziegler-Nichols Closed-Loop Tuning Methods:

1. Experiment can be time consuming

2. Can venture into unstable regions while testing the P controller, which could cause the system to become out of control.

2.8.1.3 Software Method (PID Tuning Toolbox in MATLAB)

PID tuning is the process of finding the values of proportional, integral, and derivative gains of a PID controller to achieve desired performance and meet design requirements.

PID controller tuning appears easy, but finding the set of gains that ensures the best performance of your control system is a complex task. Traditionally, PID controllers are tuned either manually or using rule-based methods. Manual tuning methods are iterative and time-consuming, and if used on hardware, they can cause damage. Rule-based methods also have serious limitations: they do not support certain types of plant models, such as unstable plants, high-order plants, or plants with little or no time delay.

The PID controllers can automatically tune to achieve the optimal system design and to meet design requirements, even for plant models that traditional rule-based methods cannot handle well.

An automated PID tuning workflow involves:

1. Identifying plant model from input-output test data.
2. Modeling PID controllers in MATLAB using PID objects or in Simulink using PID Controller blocks.
3. Automatically tuning PID controller gains and fine-tunes your design interactively.

4. Tuning multiple controllers in batch mode.
5. Tuning single-input single-output PID controllers as well as multi loop PID controller architectures.

Table 2.3: comparison between advantages and disadvantages

Method	Advantages	Disadvantages
Manual Tuning	No math required	Requires experienced personnel
Ziegler-Nichols	Proven Method	Process upset, some trial-and-error, very aggressive tuning
Software Tools	Consistent tuning; online or offline - can employ computer-automated control system design (CAutoD) techniques;	Some cost or training involved

2.8.2 Computational or Optimization Techniques:

These are techniques which are usually used for data modeling and optimization of a cost function, and have been used in PID tuning. Few examples are neural networks (computational models to simulate complex systems), genetic algorithm and differential evolution. The optimization techniques require a cost function they try to minimize. There are four types of cost functions used commonly:

- Integral Absolute Error

IAE= $\int_0^{\tau} |e(t)|$ **(2.6)**

- Integral Square error

  ISE= $\int_0^{\tau} |e(t)|^2$ **(2.7)**

- Integral Time Absolute Error

  ITAE= $\int_0^{\tau} t\,|e(t)|$ **(2.8)**

- Integral Time square Error

  ITSE= $\int_0^{\tau} t\,|e(t)|^2$ **(2.9)**

The computational models are used for self tuning or auto tuning of PID controllers. Self tuning of PID controllers essentially sets the PID parameters and also models the process by using some computational model and compares the outputs to see if there are any process variations, in which case the PID parameters are reset to give the desired response.

2.9 Artificial Bee Colony Optimization Algorithm (ABC):

Based on many benchmark functions, researches showed the ABC algorithm was competitive to other population-based algorithms, such as GA, Particle swarm optimization (PSO), Differential Evolution (DE), evolution strategies and Particle Swarm inspired Evolutionary Algorithm (PS-EA), etc., with an advantage of employing fewer control parameters. [13]

One of the quite successful optimization algorithms is the artificial bee colony (ABC) algorithm presented by Karaboga. This population based approach presents efficient solutions for numerical optimization problems especially. In main procedure of the algorithm, foraging behaviors of bee colonies are simulated by three groups of the bees.

These are employed, onlooker, and scout bees. Locations of food sources are evaluated as possible solutions for the problem. The possible and new solutions are determined by the employed bees. Nectar amounts (fitness values) in the food sources define the qualities of the solutions. The new possible food sources are selected by the onlooker bees according to information of nectar amounts.

Selection procedure of the scout bees is controlled by an important control parameter called "limit" which is the value of predetermined number of cycles.[9]

- Main steps of the ABC algorithm simulating this behavior are given below:

Step 1: Randomly generate the initial population of N food sources within the range restricted by boundaries of the variables according to Eq. (2.10)

$$x_i^j = x_{min}^j + \text{rand}\,[0, 1]\,(x_{max}^j - x_{min}^j) \quad \textbf{(2.10)}$$

Where i=1…N, j=1…D. N is the number of food source and D is the number of variables to be optimized.

Step 2: Evaluate each food source by calculating its fitness (i.e. calculate the nectar amount) according to Eq. (2.11):

$$fit_i = \begin{cases} \frac{1}{1+f_i} & if\ \ f_i \geq 0 \\ 1 + abs(f)_i\ if\ f_i < 0 \end{cases} \quad \textbf{(2.11)}$$

Where $\boldsymbol{f_i}$ is the cost value of solution $\boldsymbol{x_i}$. For maximization problems, the cost function can be directly used as a fitness function.

Step 3: Each employed bee searches a candidate food source $\boldsymbol{v_i}$ according to Eq. (2.12) Evaluate the candidate food source and apply greedy selection to select a better one as the new food source.

$$v_{ij=}\, x_{ij+}\, \emptyset_{ij}(\, x_{ij-}\ x_{kj}\,) \qquad \textbf{(2.12)}$$

Where j is a random integer within [1,D] and k ϵ {1,2,…..N} is a randomly chosen index that is different from i. $\emptyset_{ij}$ is a uniformly distributed real random number within [-1,1] .

Step 4: Calculate probability based on fitness of the solutions in the population. Each onlooker selects a food source according to Eq. (2.13) by roulette wheel selection and generates a candidate solution according to Eq. (2.12)

$$\mathrm{p} = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \qquad \textbf{(2.13)}$$

Step 5: Evaluate the candidate food source and select a better one as the new food source according to greedy selection.

Step 6: Memorize the best food source position (solution) found so far.

Step 7: If the position of a particular food source cannot be improved through the predetermined number of trials “limit”, then select it as an abandoned one. Replace the solutions by a new position that is randomly produced by a scout according to Eq. (2.10).

Step 8: Repeat the procedure from step 3 until the termination criterion is met. When the algorithm is terminated, the position of optimal food source and its nectar amount are the optimum values of the decision variables and objective function for the considered problem.

2.9.1Algorithm Description:

Similar to other nature-based algorithms, ABC models honey bees but not necessarily precisely. In this model, the honey bees are categorized as employed, onlooker and scout.

An employed bee is a forager associated with a certain food source which she is currently exploiting. She memorizes the quality of the food source and then after returning to the hive, shares it with other bees waiting there via a peculiar communication called waggle dance.

An onlooker bee is an unemployed bee at the hive which tries to find a new food source using the information provided by employed bees. A scout, ignoring the other's information, searches around the hive randomly.

In nature, the recruitment of unemployed bees happens in a nearly similar way. In addition, when the quality of a food source is below a certain level, it will be abandoned to make the bees explore for new food sources.

In ABC, the solution candidates are modeled as food sources and their corresponding objective functions as the quality (nectar amount) of the food source. For the first step, the artificial employed bees are randomly scattered in the search domain producing SN initial solutions. Here, SN represents the number of employed or onlooker bees which are considered equal until the end of algorithm. It is notable that any of these solutions x_i (i=1, 2…, SN) is a D dimensional vector representing D design variables constructing the objective function.

After this initialization, the main loop of the algorithm described hereafter is repeated for a predetermined number of cycles or until a termination criterion is satisfied.

First, all employed bees attempt to find new solutions in the neighbor of the solution (food source) they memorized at the previous cycle. If the quality (the amount of objective function) is higher at this new solution, then she forgets the former and memorizes the new one.[14]

Chapter Three

System Model

3.1 DC Motor System:

In this work the DC motor system was selected is a separately excited DC motor, which is often used to the velocity tuning and the position adjustment.

3.2 DC Motor Model

The control equivalent circuit of the DC motor by the armature voltage control method is shown in Fig. 3.1: [15]

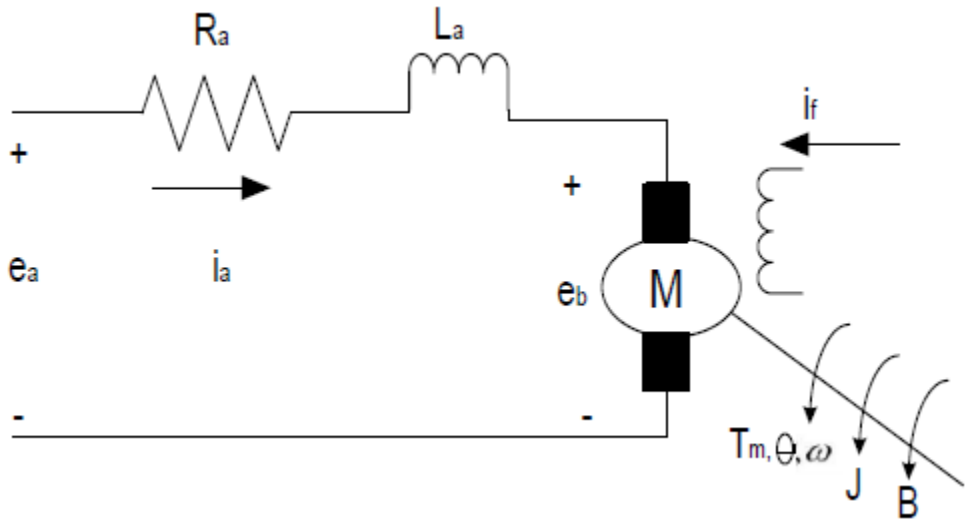

Figure 3.1: The Equivalent Circuit of DC Motor [19]

Where: Ra: Armature resistance, La: Armature inductance, Ia : Armature current, i_f: Field current, ea : Input voltage, eb : Back electromotive force (EMF), Tm : Motor torque, **ω**: An angular velocity of rotor, J: Rotating inertial measurement of motor bearing, Kb : EMF constant, KT: Torque constant, B:

Friction constant, Because the back EMF e_b is proportional to speed ω directly then,

$$e_b(t) = K_b \, d\,\frac{\theta}{dt} = K_b\omega(t) \quad \textbf{(3.1)}$$

$$e_a(t) = R_b i_a(t) + L_a \,\frac{dia(t)}{dt} + e_b(t) \quad \textbf{(3.2)}$$

From Newton law, the motor torque can be obtained as

$$T_m(t) = J\,\frac{d\,\theta(t)}{dt} + \mathrm{B}\,\frac{d\theta}{dt} = K_T\, I_a(\mathrm{t}) \quad \textbf{(3.3)}$$

Take (3.1), (3.2) and (3.3) into Laplace transform, respectively, the equation can be formulated as

$$E_a(s) = (R_a + L_a s)\, I_a(s) + E_b(s) \quad \textbf{(3.4)}$$

$$E_b(s) = K_b\, \omega(s) \quad \textbf{(3.5)}$$

$$T_m(s) = \mathrm{B}\,\omega\,(\mathrm{s}) + \mathrm{JS}\,\omega\,(\mathrm{s}) = K_T\, I_a(\mathrm{s}) \quad \textbf{(3.6)}$$

The transfer function of DC motor speed with respect to the input voltage can be written as follows:

$$\mathrm{G(s)} = \frac{\omega\,(s)}{E_a(s)} = \frac{K_T}{(L_a S + R_a)\,(JS + B) + K_b\, K_T} \quad \textbf{(3.7)}$$

The figure below is describes the DC motor armature control system function block diagram from equations (3.1) to (3.6).

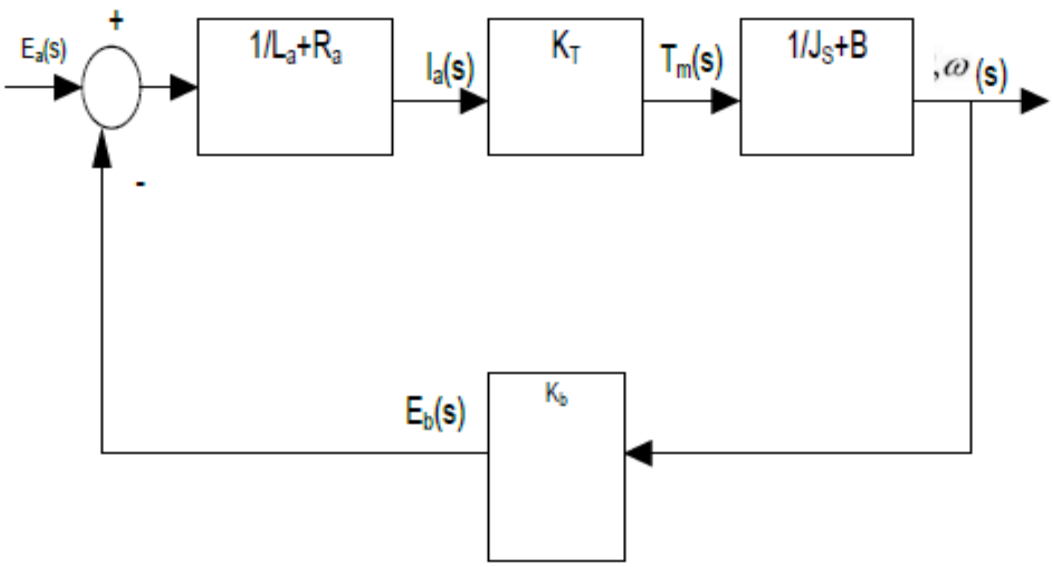

Figure 3.2: Block diagram of DC Motor[19]

The parameters of the DC motors may change according to different torque and rpm values of the DC motors. For 1000 rpm DC motor that was used in this work:

- Rotor moment of inertia (Jm) =0.01kg*m_2/s_2
- Resistance=1Ω
- Inductor=0.5H
- Electromotive Force Constant Kt=0.01Nm/Amp
- Motor Viscous Friction Constant (Beq) =0.1Nms

The transfer function of DC Motor is:

$$= \frac{0.01}{(0.5s + 1)(0.01s + 0.1) + 0.1}$$

3.3 The Design Requirements of the System:

The design requirements of the systems may vary from one system to another. The overshoot of the system should not be higher than %5 and the settling time should be smaller than 2 seconds. Steady state error should be less than 1%.

3.4 PID Tuning:

PID controller can be investigated under 3 main categories. Each controller has different properties in terms of controlling the whole system.

In proportional control, adjustments are based on the current difference between the actual and desired speed. In integral control, adjustments are based on recent errors. In derivative control, adjustments are based on the rate of change of errors.

3.4.1 Ziegler-Nichols Closed - Loop Tuning Method

The Ziegler-Nichols closed-loop tuning method allows you to use the critical gain value, K_{cr}, and the critical period of oscillation, P_{cr}, to calculate. It is a simple method of tuning PID controllers and can be refined to give better approximations of the controller. You can obtain the controller constants, T_i, and T_d in a system with feedback. The Ziegler-Nichols closed-loop tuning method is limited to tuning processes that cannot run in an open-loop environment. Determining the ultimate gain value, K_{cr} is accomplished by finding the value of the proportional-only gain that causes the control loop to oscillate indefinitely at steady state. This means that the gains I and D controller are set to zero so that the influence of P can be determined. It tests the robustness of the K_p value so that it is optimized for the controller.

Another important value associated with this proportional-only control tuning method is the critical period (P_{cr}). The ultimate period is the time required to complete one full oscillation while the system is at steady state. These two parameters, K_{cr} and P_{cr}, are used to find the loop-tuning constants of the controller (P, PI, or PID).

- The Tuning Procedure:

1. Remove integral and derivative action. Set integral time (Ti) to ∞ or its largest value and set the derivative controller (Td) to zero.

2. Create a small disturbance in the loop by changing the set point. Adjust the proportional, increasing and/or decreasing, the gain until the oscillations have constant amplitude.

3. Record the gain value (Kcr) and period of oscillation (Pcr).

4. Plug these values into the Ziegler-Nichols closed loop equations and determine the necessary settings for the controller.

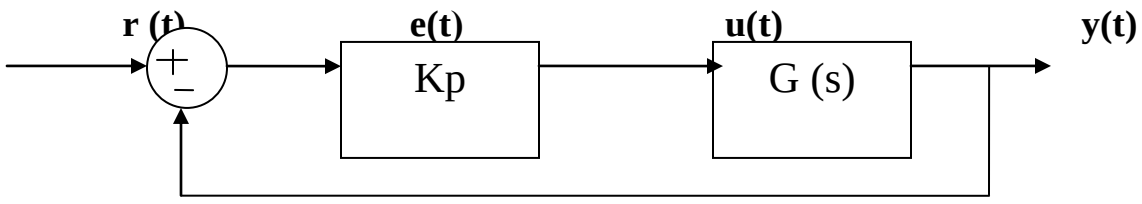

Figure 3.3: the control system with Gain Kp

In the table 3.1 represent how to calculate parameters of controller with Ziegler-Nechols method:

Table 3.1: Calculation of ($K_p.T_i.T_d$)

Controller	K	Ti	Td
P	$\frac{Kcr}{2}$	∞	0
PI	$\frac{Kcr}{2.2}$	$\frac{pcr}{1.2}$	0
PID	$\frac{Kcr}{1.7}$	$\frac{pcr}{2}$	$\frac{pcr}{8}$

3.4.2 Artificial Bee Colony Optimization (ABC) Algorithm:

The artificial bee colony (ABC) is one of the swarm intelligence algorithms used to solve optimization problems which is inspired by the foraging behavior of the honey bees. Since its advent (Karaboga, 2005) ABC and its variants have attracted increasing interest and has been applied to solve many real-world optimization problems (Karaboga and Basturk, 2008; Lin and Su, 2012; El-Telbany, 2013). The ABC has the advantages of strong robustness, fast convergence and fewer setting parameters.

3.5 Block Diagram:

In this work a PID controller was used to control the DC motor speed, PID was tuned using ABC algorithm, the structure of PID controller with ABC algorithm is shown in figure 3.3:

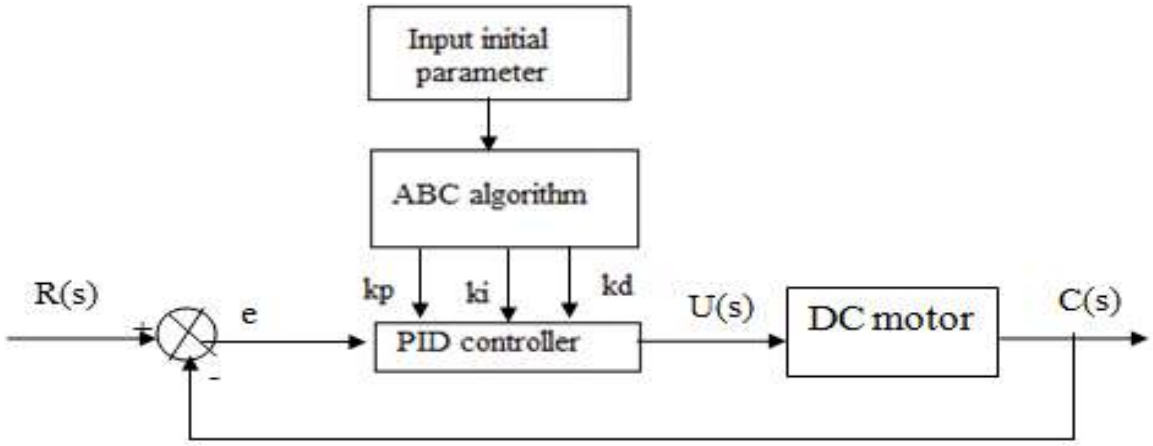

Figure 3.4: the structure of PID controller with ABC algorithm

❖ Detailed pseudo-code of the ABC algorithm is given below:

1. Initialize the population of solutions xi,j, $i = 1. . .SN$, $j = 1. ..D$ by eq. (2.10).
2. Evaluate the fitness of population by eq.(2.11).
3. Cycle=1.
4. **Repeat**.
5. Produce new solutions $v_{i,j}$ for the employed bees by using eq.(2.12) and evaluate them.
6. Apply the greedy selection process.
7. Calculate the probability values $P_{i,j}$ for the solutions $x_{i,j}$ by eq.(2.13).
8. Produce the new solutions $v_{i,j}$ for the onlookers from the solutions $x_{i,j}$ selected depending on $P_{i,j}$ and evaluate them.

9. Apply the greedy selection process.

10. Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution $x_{i,j}$ by (2.10).

11. Memorize the best solution achieved so far

12. Cycle =cycle+1.

Until cycle=MCN(maximum cycle number).[16]

3.6 System Flow Chart:

In the figure 3.5 it's showed the steps of how the ABC algorithm working and how the parameters of controller was tuned by algorithm.

Firstly initialization population was generated randomly, run DC motor with initial value for controller parameters, and then evaluates the fitness for each employed bee to calculate probability based on fitness of the solutions in the population after that can apply greedy selection process to select best solutions and increase cycle by one, testing if it reach the maximum cycle? If yes then stop the algorithm, if NO return again the search process.

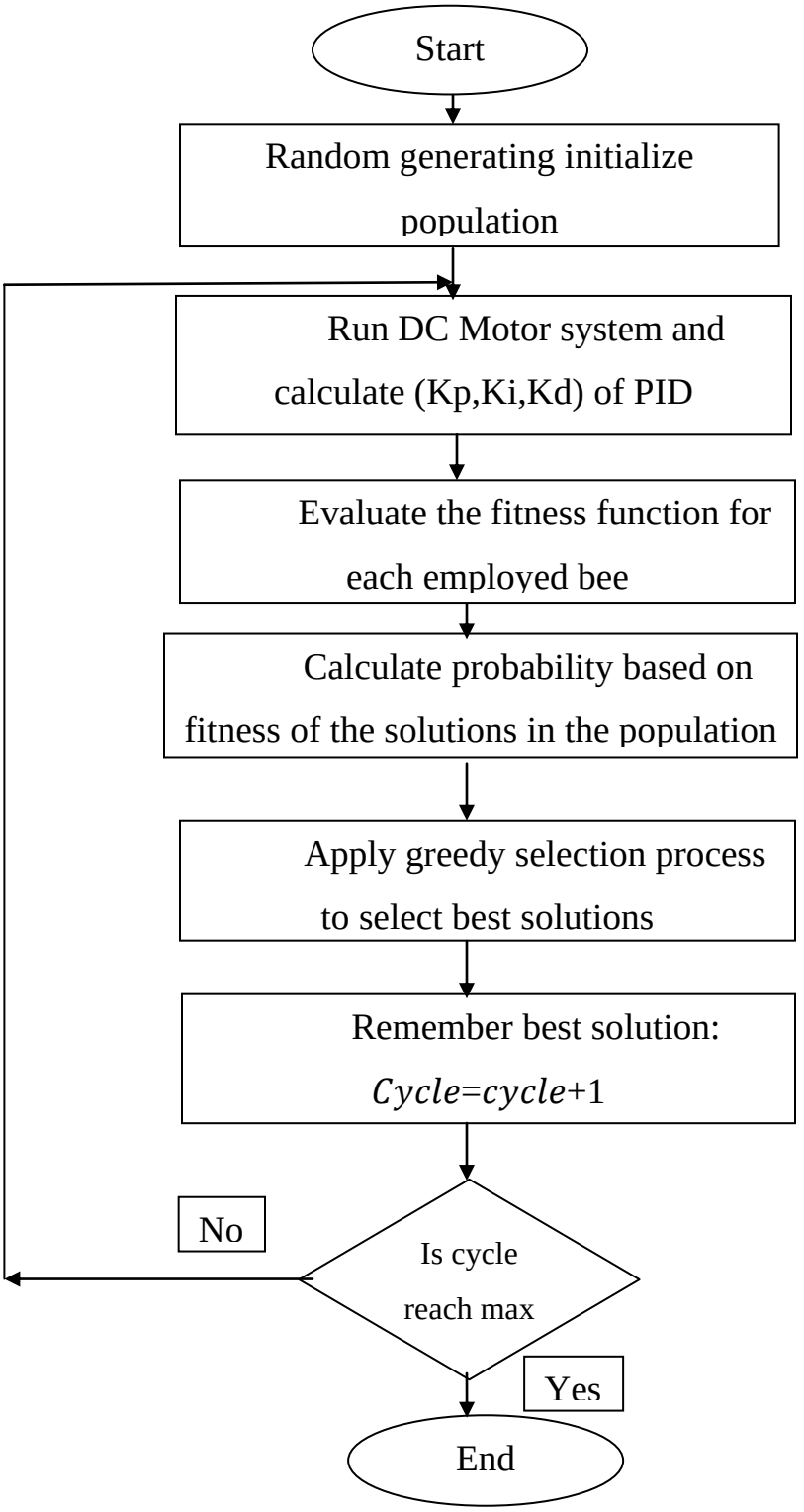

Figure 3.5: System Flowcharts of ABC-PID Control System

Chapter Four

Result and Discussion

4.1 Overview of Matlab:

MATLAB is one of a number of commercially available, sophisticated mathematical computation tools, which also include Maple, Mathematica, and MathCad. Because MATLAB ® is so easy to use, you can perform many programming tasks with it It excels at numerical calculations—especially matrix calculations—and graphics, MATLAB ® is available in both a professional and a student version. The professional version is probably installed in your college or university computer laboratory, Student editions are available for Microsoft Windows, Mac OSX, and Linux operating systems and can be purchased from college bookstores or online from The MathWorks at www.mathworks.com. [17]

The parameters of the DC motors may change according to different torque and rpm values of the DC motors. For 1000 rpm DC motor that we have used in this work:

- Rotor moment of inertia (Jm) =0.01kg*m2/s2
- Resistance=1Ω
- Inductor=0.5H
- Electromotive Force Constant Kt=0.01Nm/Amp
- Motor Viscous Friction Constant (Beq) =0.1Nms

The transfer function of the DC motor is equal:

$$G(s) = \frac{\omega\,(s)}{E_a(s)} = \frac{K_T}{(L_aS+R_a)\,(JS+B)+ K_b\,K_T} \qquad \textbf{(4.1)}$$

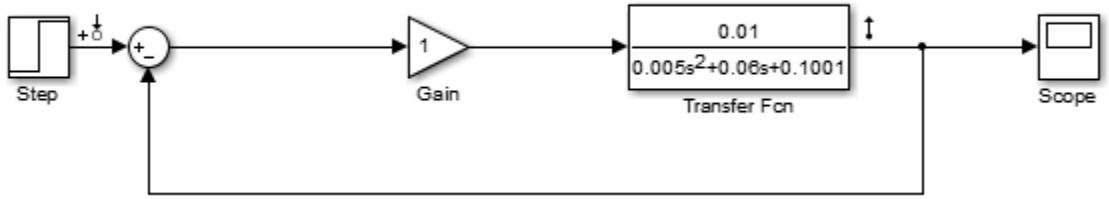

Figure 4.1: the simulink block diagram of DC Motor

The step response of DC motor without controller is shown in figure 4.2 where the step response less than one then the system needed controller.

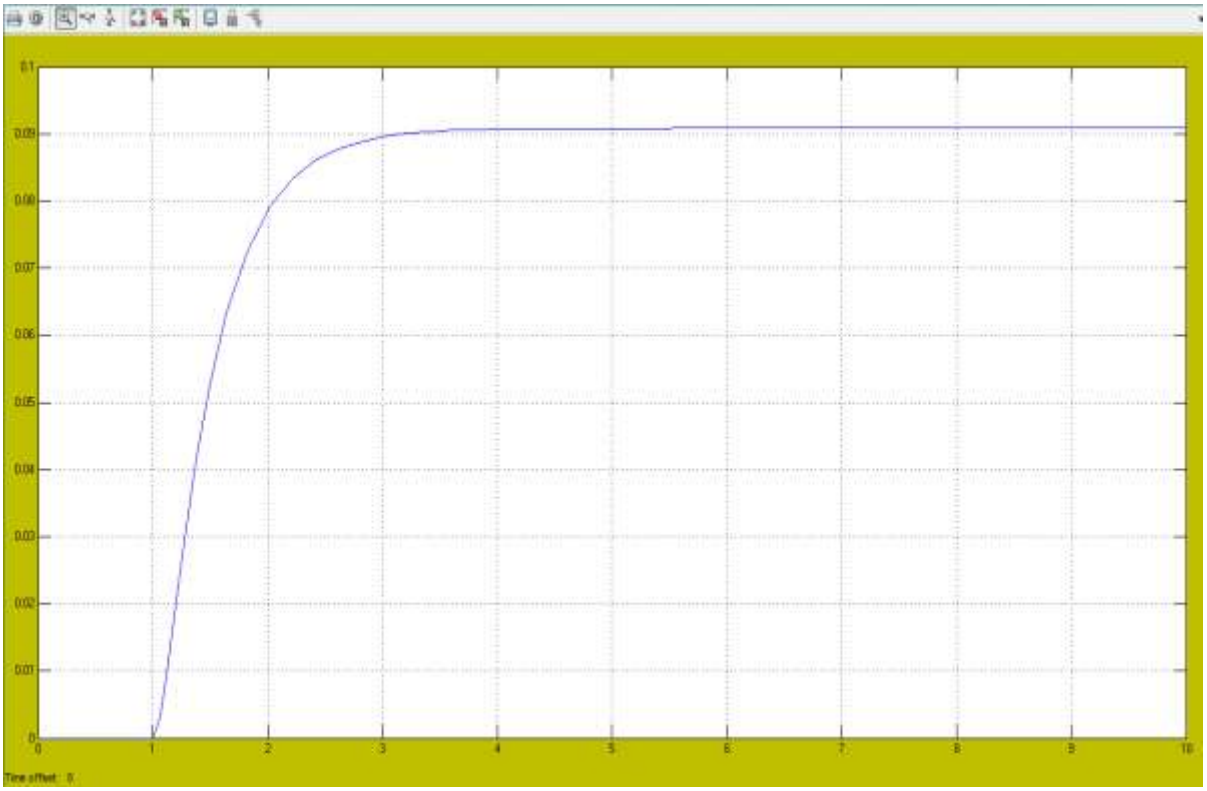

Figure 4.2: step response without controller

To make the block of the system by simulink the many steps was followed: firstly open the matlab and create new model, open simulink library browser to add the blocks, configure the blocks and write the parameters of dc motor, do the system is closed loop. Finally performing tuning of PID controller manually, tuning by Ziegler Nichols method and PID controller by using ABC algorithm.

4.2 DC Motor with manual tuning of PID:

The figures below explaining how the different values of parameters of PID effect on the DC motor performance by using try and error method and block tune:

4.2.1 Proportional (p) effect on Dc motor:

Firstly make block diagram by simulink on matlab with proportional controller only connected to DC motor as in the figure 4.3 to showing the result

- When the value of the p= 63

From figure 4.4 the result observed the overshoot on tuned response is decreased by 9% comparing with block response, but the rise time by manual tuning it is very small.

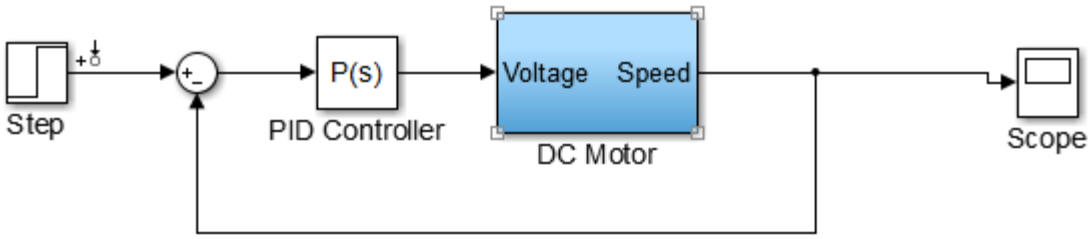

Figure 4.3: the block diagram of (p) controller

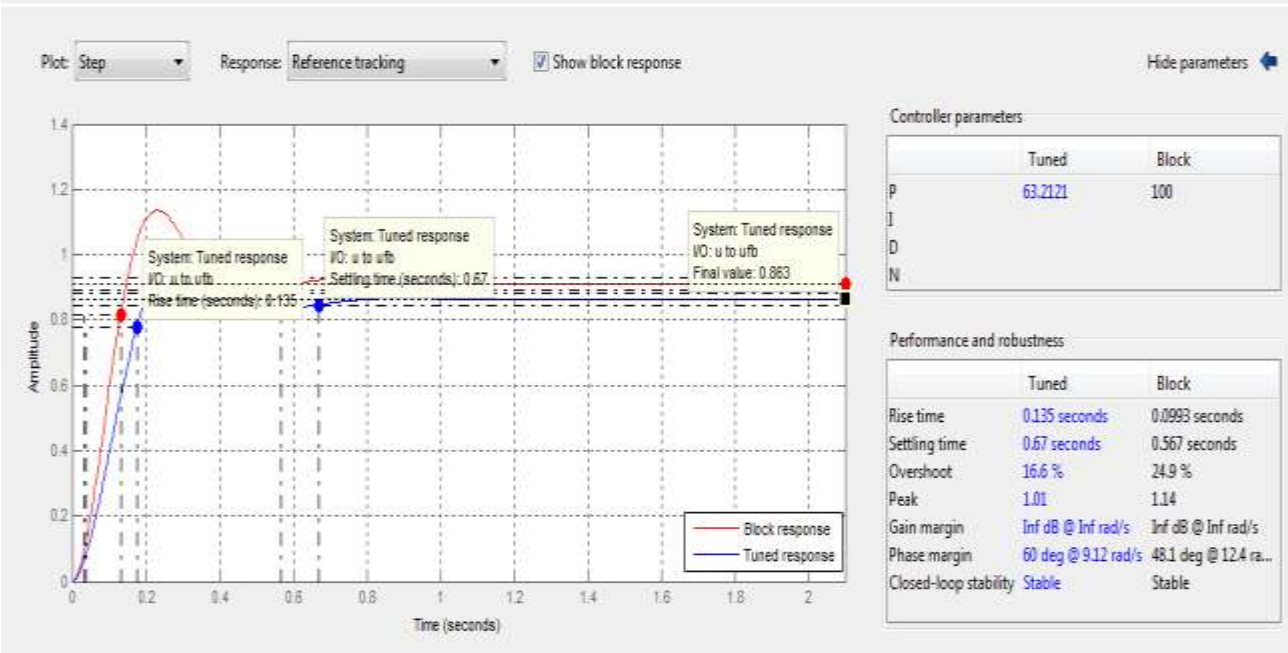

Figure 4.4: Proportional (p) controller

4.2.2 Integral (I) Effect on DC motor:

The second step make block diagram by simulink on matlab with integral controller only, connected to DC motor as in the figure 4.5 to showing the result

- When the value of the I= 10

From figure 4.6 the result observed no overshoot on block response, but the rise time by manual tuning it is increased.

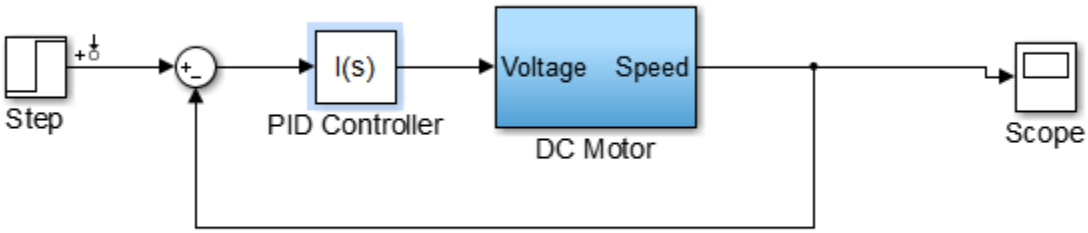

Figure 4.5: the block diagram of (I) controller

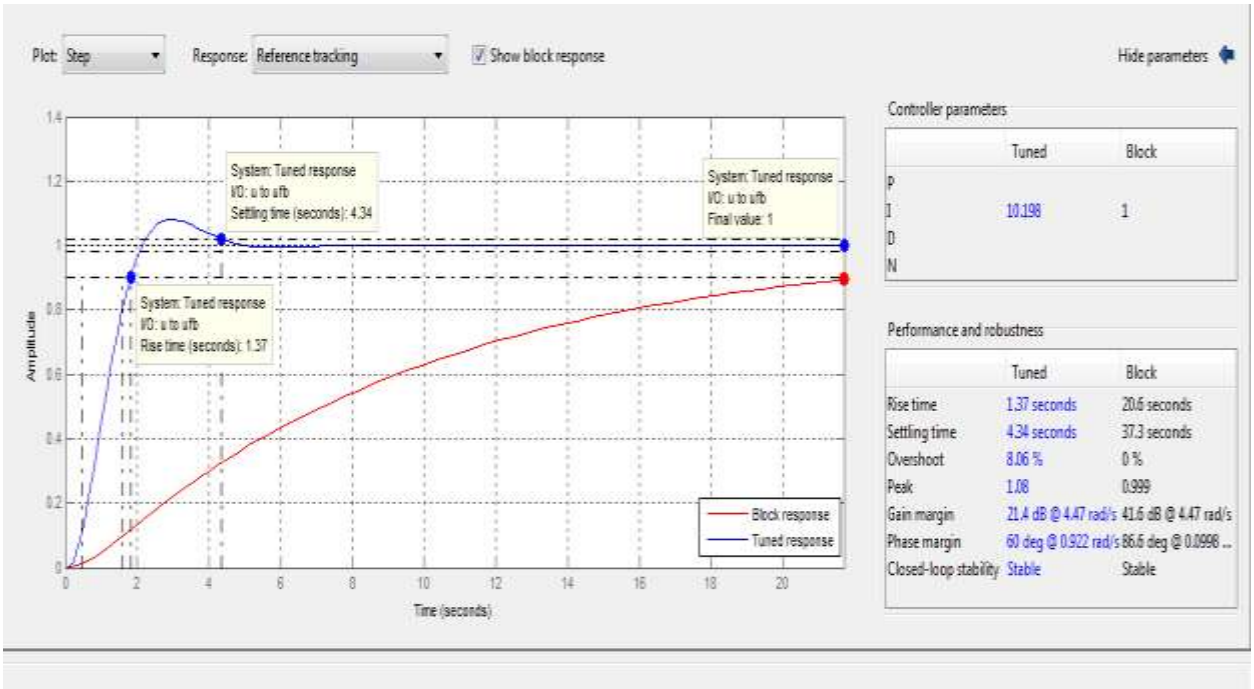

Figure 4.6: Integral (I) controller

4.2.3 Proportional-Integral (PI) Effect on DC motor:

In this part make block diagram by simulink on matlab with 3 Proportional-Integral controller, connected to DC motor as in the figure 4.7

- When the value of the P= 100, I=1

From figure 4.8 the result observed the overshoot was increased on block response, but the rise time is smallest value.

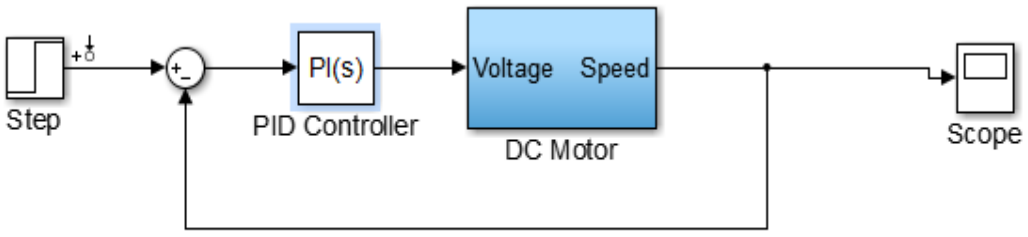

Figure 4.7: the block diagram of (PI) controller

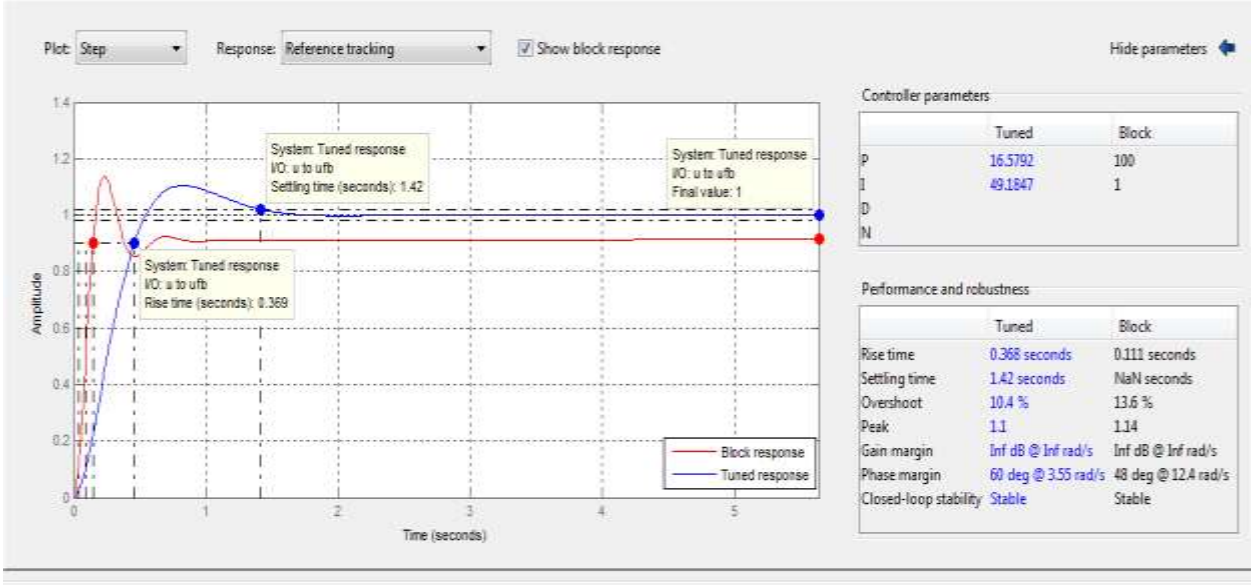

Figure 4.8: Proportional-Integral (PI) controller

4.2.4 Proportional-Derivative (PD) Effect on DC motor:

Either this part makes block diagram by simulink on matlab with Proportional-Integral controllers, connected to DC motor as in the figure 4.7

- When the value of the P= 100,D=1

From figure 4.10 the result observed the overshoot was increased on block response, but the rise time is smallest value.

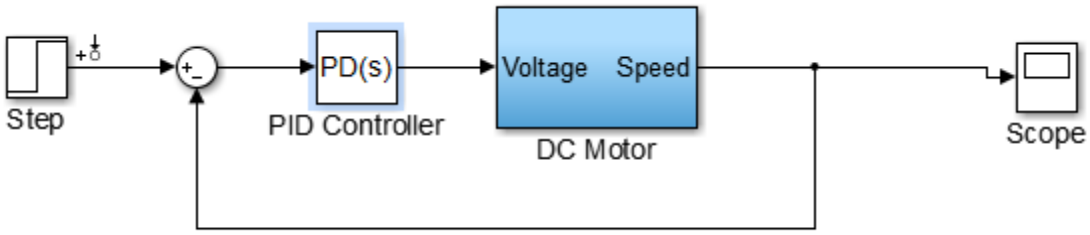

Figure 4.9: the block diagram of (PD) controller

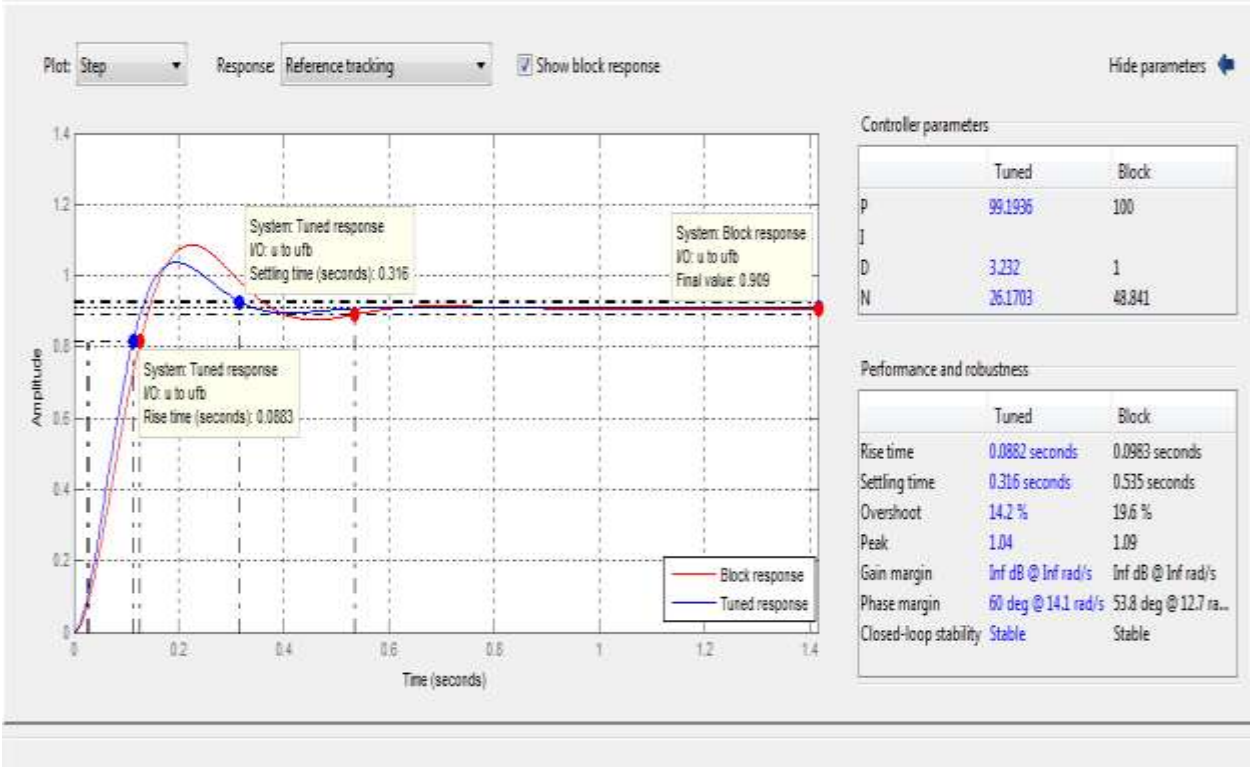

Figure 4.10: Proportional-Derivative (PD) controller

4.2.5 PID Effect on DC motor:

Finally make block diagram by simulink on matlab with Proportional-Integral- Derivative controller, connected to DC motor as in the figure 4.11

When the value of the P= 100, D=1, I=1 in the figure 4.12 was showed the overshoot value was decreased, and the rise time is smallest value, then the PID controller is the best choice.

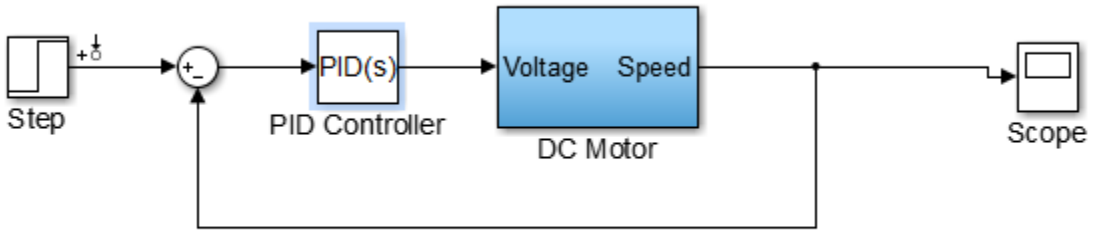

Figure 4.11: the Simulink block diagram of

PID controller with DC Motor

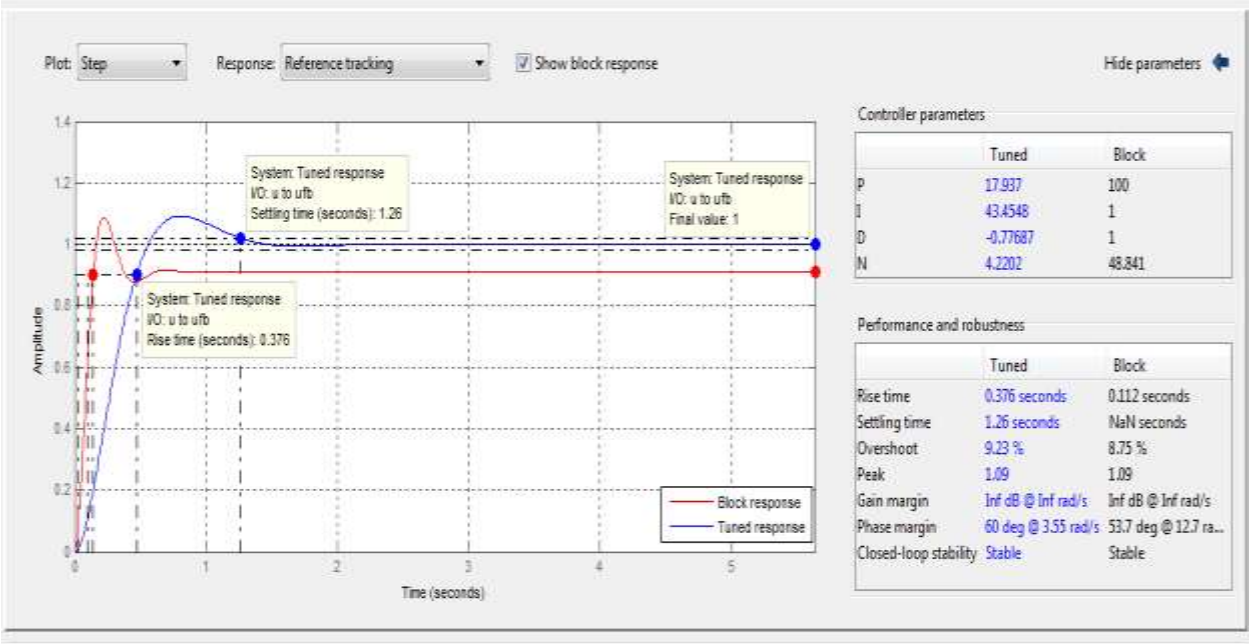

Figure 4.12: PID controller

Other method was used to tuning the PID controller called Ziegler-Nichols, the next section showed different responses:

4.3 Tuning (P) using Ziegler-Nichols:

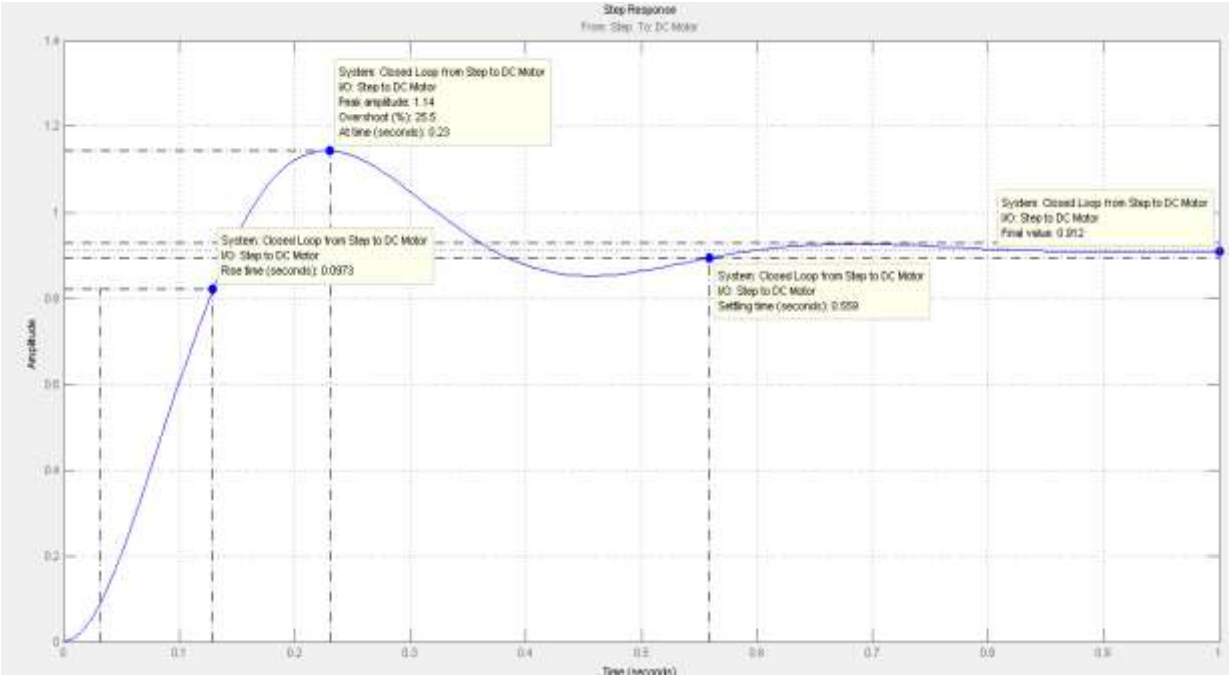

Figure 4.13: dc motor response in tuning P using Z-N

The value of **p** parameter = 103.179.

4.4 Tuning (PI) using Ziegler-Nichols:

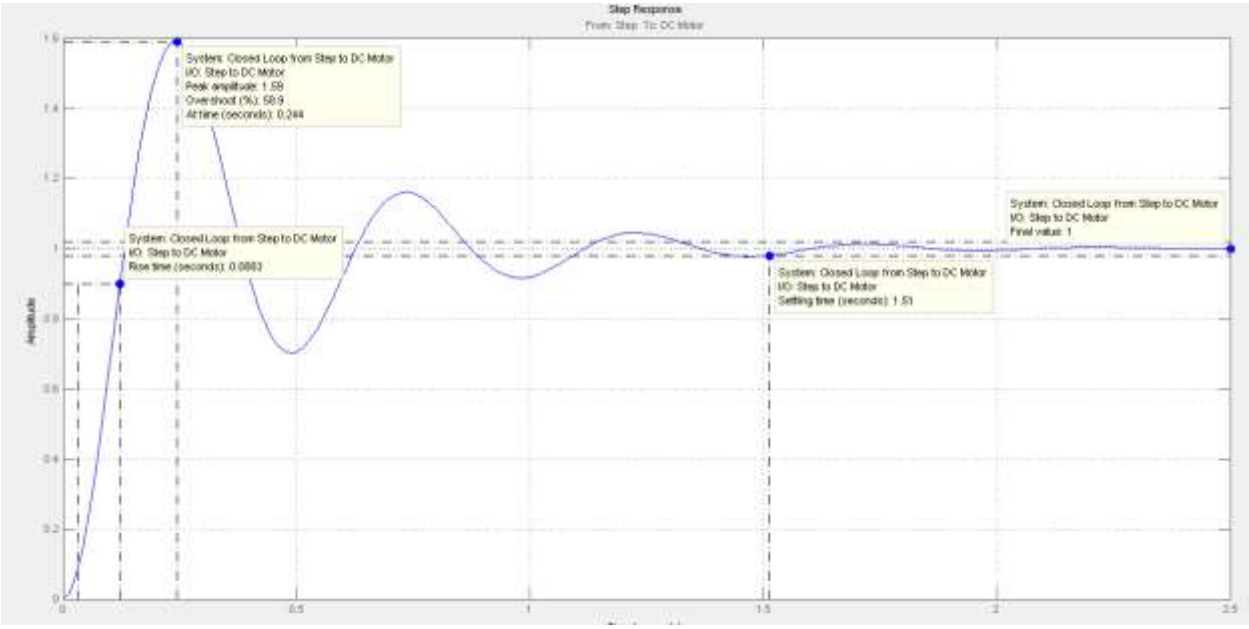

Figure 4.14: dc motor response in tuning PI using Z-N

The values of **P** and **I** parameters is equal **92.86**and I =**578.9** respectively.

4.5 Tuning (PID) using Ziegler-Nichols:

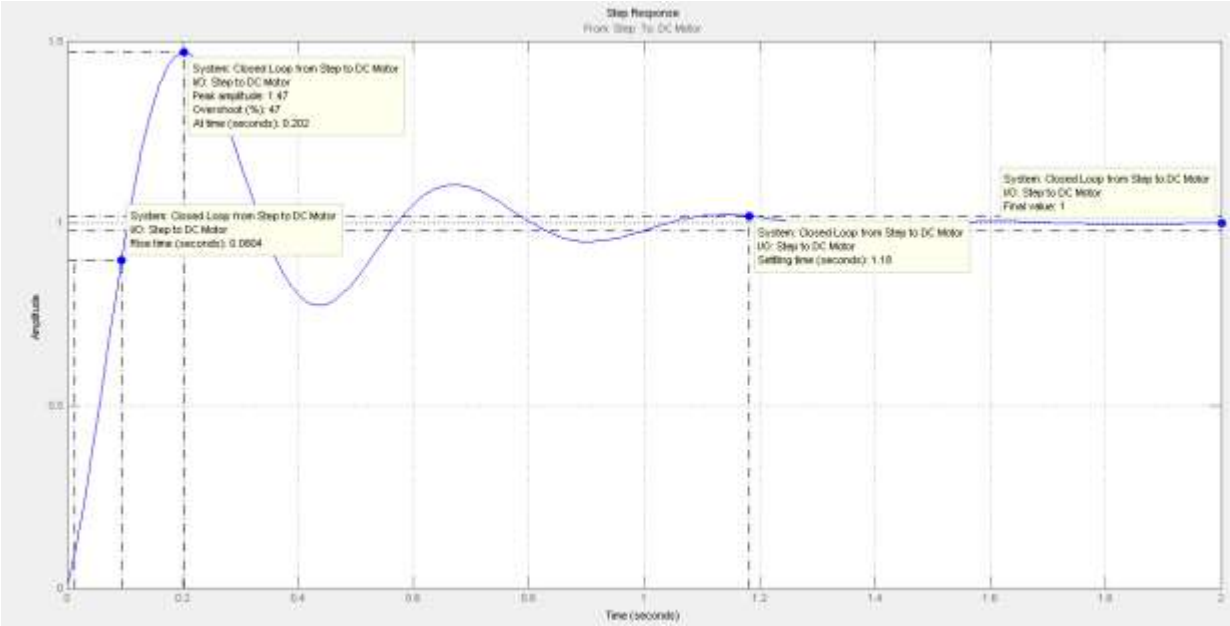

Figure 4.15: PID with Z.N method

Finally shown in the figure 4.11 the response of PID using Z-N method by value:

p=123.8**, I=1157.8 and D=3.31**

4.6 Tuning PID using ABC:

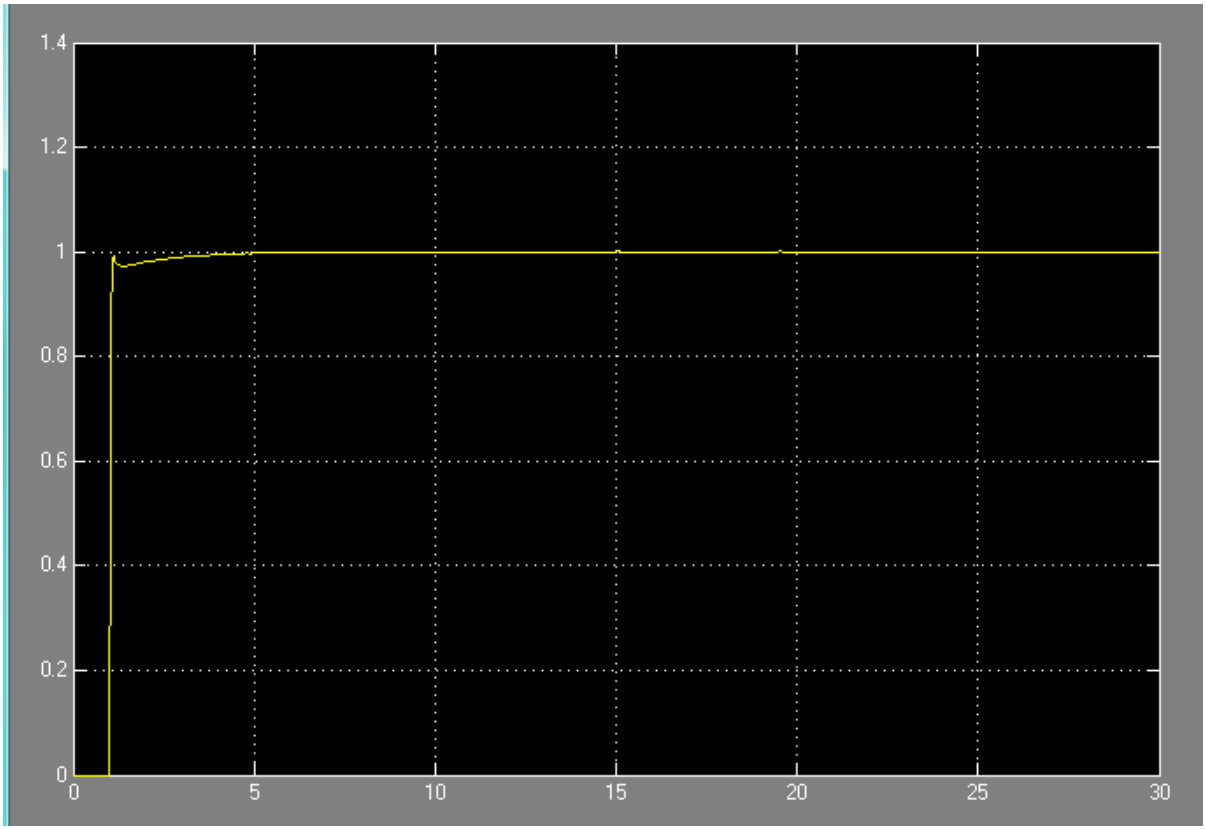

Figure 4.16: step response for ABC-PID

To find out the values of controller parameters, performance and robustness the figures below represent PID values by using ABC algorithm to improve performance of dc motor

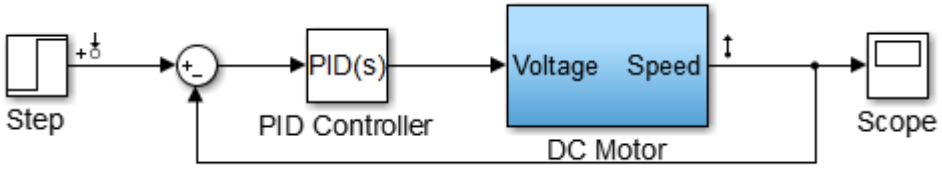

Figure 4.17: the Simulink block diagram of ABC-PID

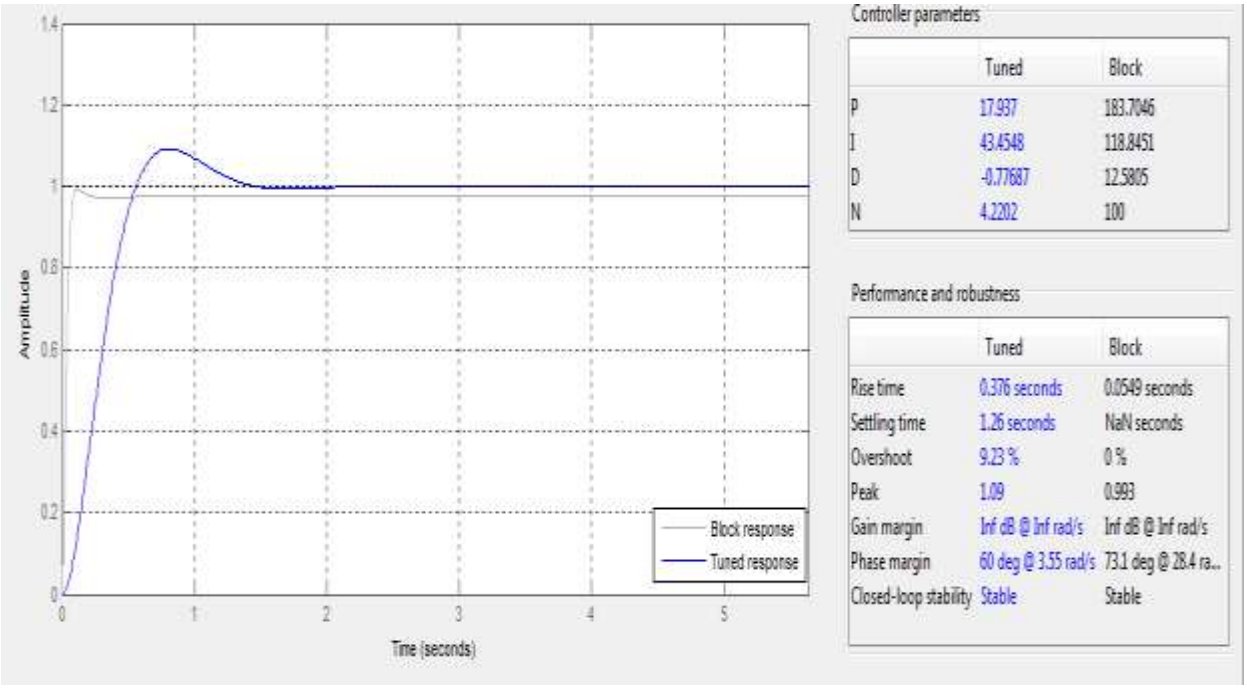

Figure 4.18: PID with ABC algorithm

By using PID controller the step response of the DC motor has improved, the system works with less rise time and the step response is equal to one with overshoot equal zero.

Table 4.1: show parameters between different methods:

parameters / method	Z-N	ACO	ABC
Rise Time	0.0004 sec	0.184 sec	0.0549 sec
Settling Time	1.18 sec	0.291 sec	NaN sec
Overshoot	47 %	0 %	0 %

From the results was shown in table 4.1comparison between different responses for DC motor with different techniques of controller firstly with Z-N method, ACO algorithm and finally ABC algorithm, where it turns out the ABC algorithm is the best method because it was gave the best result, no overshoot, small rise time and the settling time it is very small.

Chapter Five

5. Conclusion and Recommendations

5.1 Conclusion:

The artificial bee colony (ABC) algorithm was applied to tune PID controller for DC motor drive system. The ABC method optimized PID parameter as well as rise time, settling time and overshoot. Result showed this method has been successful comparison to classical approach and optimal PID tuning using performance indices can considered as powerful tuning scheme for controllers.

At the end of analysis the rise time, settling time and the maximum overshoot of the control system are optimized with ABC algorithm, it turned out the proposed method can be search for optimum solution and improve dynamic performance for the system with better way. By comparison with Z.N-PID and ACO algorithm method the PID-ABC controller is the best which presented satisfactory performances and possesses good robustness (No overshoot, minimal rise time, Steady state error approximately = 0).

5.2 Recommendations:

- Can using the ABC to controlling of more complex and more wobbling systems.
- Can applying this algorithm on others types of Motors.
- Use the ABC algorithm with another optimization algorithm together on one experiment to testing the effectiveness of the proposed controllers.
- The students and researchers can apply and develop optimization algorithms in their research to discover new algorithms to improve the performance of controllers and thus get a better performance for dc motor.

References:

[1] P. Varma, “Control OF DC Motor Using Artificial Bee Colony based PID Controller,” vol. 2, no. 3, 2013.

[2] D. Karaboga and B. Basturk, “algorithm,” pp. 459–471, 2007.

[3] A. Eydgahi and M. Fotouhi, “A Fuzzy Knowledge-Based Controller to Tune PID Parameters.”

[4] S. P. P. G. Student, M. Kishnani, P. G. Student, and R. Gupta, “Optimal Tuning of PID Controller Using Genetic Algorithm and Swarm Techniques,” vol. 7, no. 2, pp. 189–194, 2014.

[5] M. E. El-telbany, “Tuning PID Controller for DC Motor : An Artificial Bees Optimization Approach,” vol. 77, no. 15, pp. 18–21, 2013.

[6] K. V. L. Narayana, V. N. Kumar, M. Dhivya, and R. P. Raj, “Application of Ant Colony Optimization in Tuning a PID Controller to a Conical Tank,” vol. 8, no. January, pp. 217–223, 2015.

[7] D. H. Kim and J. H. Cho, “Robust Tuning of PID Controller With Disturbance Rejection Using Bacterial Foraging Based Optimization,” 2004.

[8] D. C. Motors, “4. dc motors,” no. i, pp. 1–19.

[9] A. Bagis and H. Senberber, “ABC Algorithm Based PID Controller Design for Higher Order Oscillatory Systems,” pp. 3–9, 2017.

[10] S. Kavianpour, “Student Projects and the Application of PID Control,” 2017.

[11] H. O. Bansal, R. Sharma, and P. R. Shreeraman, “PID Controller Tuning Techniques : A Review,” no. November 2012, 2017.

[12] S. Temel, S. Yağli, and S. Gören, “P , Pd , Pi , Pid Controllers,” https://www.google.com.pr/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0CB0QFjAAahUKEwig6ZPnyInJAhUG5CYKHX6oBDI&url=http%3A%2F%2Fwww.researchgate.net%2Ffile.PostFileLoader.html%3Fid%3D54685991d11b8bc9668b461a%26assetKey%3DAS%253A27363520017, p. 63, 2012.

[13] Y. Tian, L. Huang, and Y. Xiong, “A General Technical Route for Parameter Optimization of Ship Motion Controller Based on Artificial Bee Colony Algorithm,” vol. 9, no. 2, pp. 133–137, 2017.

[14] M. Abachizadeh, M. Reza, H. Yazdi, and A. Yousefi-koma, “Optimal Tuning of PID Controllers Using Artificial Bee Colony

Algorithm," no. August, 2010.

[15] A. Y. Al-Maliki and K. Iqbal, "FLC-based PID controller tuning for sensorless speed control of DC motor," Proc. IEEE Int. Conf. Ind. Technol., vol. 2018–February, no. September, pp. 169–174, 2018.

[16] D. Karaboga and B. Basturk, "Artificial Bee Colony (ABC) Optimization Algorithm for Solving Constrained Optimization," pp. 789–798.

[17] Holly Moore, " MATLAB Of Engineers", Third Edition.

[18] J. Kaushal, " Comparative Performance Study of ACO& ABC Optimizaion based PID Controller Tuning for Speed Control of DC Motor Drives ", June.2012

[19] M. KUSHWAH and A. PATRA"PID Controller Tuning using Ziegler-Nichols Method for Speed Control of DC Motor", June.2014

Appendix

```
clear all
close all
clc

%/* Control Parameters of ABC algorithm*/
NP=20; %/* The number of colony size (employed bees+onlooker bees)*/
FoodNumber=NP/2; %/*The number of food sources equals the half of the colony
size*/
limit=100; %/*A food source which could not be improved through "limit" trials is
abandoned by its employed bee*/
maxCycle=20; %/*The number of cycles for foraging {a stopping criteria}*/


%/* Problem specific variables*/
objfun='Sphere'; %cost function to be optimized
D=3; %/*The number of parameters of the problem to be optimized*/
ub=ones(1,D)*1000; %/*lower bounds of the parameters. */
lb=ones(1,D)*(-1000);%/*upper bound of the parameters.*/

runtime=3;%/*Algorithm can be run many times in order to see its robustness*/

for m=1:maxCycle
   GlobalMins=zeros(runtime,1);

for r=1:runtime

% /*All food sources are initialized */
%/*Variables are initialized in the range [lb,ub]. If each parameter has different
range, use arrays lb[j], ub[j] instead of lb and ub */

Range = repmat((ub-lb),[FoodNumber 1]);
Lower = repmat(lb, [FoodNumber 1]);
Foods = rand(FoodNumber,D) .* Range + Lower;

ObjVal=feval(objfun,Foods);
Fitness=calculateFitness(ObjVal);
```

```
%reset trial counters
trial=zeros(1,FoodNumber);

%/*The best food source is memorized*/
BestInd=find(ObjVal==min(ObjVal));
BestInd=BestInd(end);
GlobalMin=ObjVal(BestInd);
GlobalParams=Foods(BestInd,:);

iter=1;
while ((iter <= maxCycle)),

%%%%%% EMPLOYED BEE PHASE %%%%%%
   for i=1:(FoodNumber)

       %/*The parameter to be changed is determined randomly*/
       Param2Change=fix(rand*D)+1;

       %/*A randomly chosen solution is used in producing a mutant solution of the
solution i*/
       neighbour=fix(rand*(FoodNumber))+1;

       %/*Randomly selected solution must be different from the solution i*/
          while(neighbour==i)
             neighbour=fix(rand*(FoodNumber))+1;
          end;

      sol=Foods(i,:);
      %  /*v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij}) */
      sol(Param2Change)=Foods(i,Param2Change)+(Foods(i,Param2Change)-
Foods(neighbour,Param2Change))*(rand-0.5)*2;

      %  /*if generated parameter value is out of boundaries, it is shifted onto the
boundaries*/
       ind=find(sol<lb);
       sol(ind)=lb(ind);
       ind=find(sol>ub);
       sol(ind)=ub(ind);
       %evaluate new solution
       ObjValSol=feval(objfun,sol);
```

```
       FitnessSol=calculateFitness(ObjValSol);
      % /*a greedy selection is applied between the current solution i and its
mutant*/
      if (FitnessSol>Fitness(i)) %/*If the mutant solution is better than the current
solution i, replace the solution with the mutant and reset the trial counter of
solution i*/
          Foods(i,:)=sol;
          Fitness(i)=FitnessSol;
          ObjVal(i)=ObjValSol;
          trial(i)=0;
       else
          trial(i)=trial(i)+1; %/*if the solution i can not be improved, increase its trial
counter*/
      end;
   end;
prob=(0.9.*Fitness./max(Fitness))+0.1;
i=1;
t=0;
while(t<FoodNumber)
   if(rand<prob(i))
      t=t+1;
      %/*The parameter to be changed is determined randomly*/
      Param2Change=fix(rand*D)+1;

      %/*A randomly chosen solution is used in producing a mutant solution of the
solution i*/
      neighbour=fix(rand*(FoodNumber))+1;

      %/*Randomly selected solution must be different from the solution i*/
          while(neighbour==i)
             neighbour=fix(rand*(FoodNumber))+1;
          end;

      sol=Foods(i,:);
      %  /*v_{ij}=x_{ij}+\phi_{ij}*(x_{kj}-x_{ij}) */
      sol(Param2Change)=Foods(i,Param2Change)+(Foods(i,Param2Change)-
Foods(neighbour,Param2Change))*(rand-0.5)*2;
  ind=find(sol<lb);
      sol(ind)=lb(ind);
      ind=find(sol>ub);
```

```
        sol(ind)=ub(ind);

        %evaluate new solution
        ObjValSol=feval(objfun,sol);
        FitnessSol=calculateFitness(ObjValSol);

       % /*a greedy selection is applied between the current solution i and its
mutant*/
       if (FitnessSol>Fitness(i)) %/*If the mutant solution is better than the current
solution i, replace the solution with the mutant and reset the trial counter of
solution i*/
           Foods(i,:)=sol;
           Fitness(i)=FitnessSol;
           ObjVal(i)=ObjValSol;
           trial(i)=0;
        else
           trial(i)=trial(i)+1; %/*if the solution i can not be improved, increase its trial
counter*/
       end;
    end;

    i=i+1;
    if (i==(FoodNumber)+1)
       i=1;
    end;
end;

%/*The best food source is memorized*/
        ind=find(ObjVal==min(ObjVal));
        ind=ind(end);
        if (ObjVal(ind)<GlobalMin)
        GlobalMin=ObjVal(ind);
        GlobalParams=Foods(ind,:);
        end;
    %%%%%%%%% SCOUT BEE PHASE %%%%%%%%%%
%/*determine the food sources whose trial counter exceeds the "limit" value.
%In Basic ABC, only one scout is allowed to occur in each cycle*/

ind=find(trial==max(trial));
ind=ind(end);
```

```
if (trial(ind)>limit)
   trial(ind)=0;
   sol=(ub-lb).*rand(1,D)+lb;
   ObjValSol=feval(objfun,sol);
   FitnessSol=calculateFitness(ObjValSol);
   Foods(ind,:)=sol;
   Fitness(ind)=FitnessSol;
   ObjVal(ind)=ObjValSol;
end;
%/* The best food source is memorized */
ind = find ( ObjVal == min ( ObjVal ));
ind = ind ( end );
if ( ObjVal ( ind )< GlobalMin )
GlobalMin = ObjVal ( ind );
GlobalParams = Foods (ind ,:);
end ;

fprintf('ter=%d ObjVal=%g\n',iter,GlobalMin)
% GlobalParams;
iter=iter+1;
end % End of ABC
GlobalMins(r)=GlobalMin
end; %end of runs
kp=round(GlobalMins(1));
ki=GlobalMins(2);
kd=GlobalMins(3);

 if (185>kp) &&(kp > 183)
    fprintf('kp=%d\n ki =%d kd=%d\n',kp,ki,kd)
   break
else
  continue
end

 end
```