بسم الله الرحمن الرحيم

Sudan University of Science and Technology

كلية الدراسات العليا

قسم الهندسة الإلكترونية

College of post graduate

Using ant colony optimization algorithm for direct current motor speed controller

استخدام خوارزمية مستعمرة النمل المثلى للتحكم في سرعة موتور التيار المباشر

A thesis submitted in partial fulfillment of the requirements for the Degree of Master of Computer Engineering and Networking

By:

Suha Salah Eldin Awad

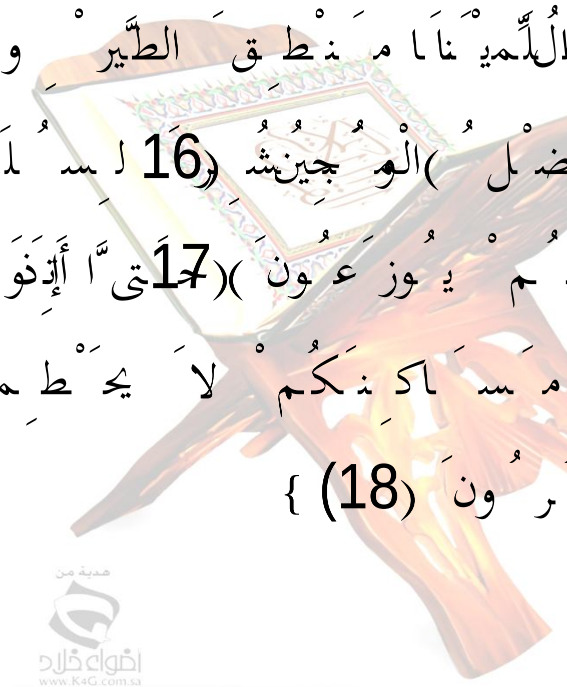
Supervisor:

Dr. Alaa Eldeen Awooda

November 2018

الآيـــــــــة

قال تعالى:



{وَوَرِثَ سُلَيْمَانُ دَاوُودَ وَقَالَ يَا أَيُّهَا النَّاسُ عُلِّمْنَا مَنطِقَ الطَّيْرِ وَأُوتِينَا
شَيْءٍ إِنَّ هَذَا لَهُوَ الْفَضْلُ الْمُبِينُ (16) وَحُشِرَ لِسُلَيْمَانَ
نِّ وَالْإِنسِ وَالطَّيْرِ فَهُمْ يُوزَعُونَ (17) حَتَّى إِذَا أَتَوْا عَلَى
ةٌ يَا أَيُّهَا النَّمْلُ ادْخُلُوا مَسَاكِنَكُمْ لَا يَحْطِمَنَّكُمْ
جُنُودُهُ وَهُمْ لَا يَشْعُرُونَ (18) }

صدق الله العظيم

سورة النمل الآية {16-18}

Dedication

To

My Mother,

Words cannot express how grateful I am for all of the sacrifices that you’ve made to me.

To my husband who has always supported me.

Acknowledgments

It is a great pleasure, a privilege, an honor, a source of much personal satisfaction, to be here with all of you this moment. First of all, I must insist: I have not done this work alone. Many people have helped to keep my research alive and, if you will bear with me, for just a few moments, I believe that the present context is an appropriate one in which to express my heartfelt gratitude.

Before I say anything more, I want to thank my Supervisor

Dr. ***Alaa Eldeen Awooda***

Who has been Always There to listen and give advice. I am deeply grateful to him for the long discussions that helped me sort out the technical details of my work. I am also thankful to him for encouraging the me of correct grammar and consistent notation my writings and for carefully reading and commenting on countless revisions of this manuscript.

And Special Thanks for my special friend alaa altahir .

Also I very thankful for Osman abdalall and Ahmed humeida for giving me their time and help me .

Abstract

The development of high performance motor drives is very important in all industrial fields and shows the importance of development in high-precision industries that respond to the rapid response to the organization of load to perform the task high. In this project, the most common PID control is introduced in industrial use because of its remarkable efficiency, simplicity of implementation and application, and how to synthesize this type of controller using artificial intelligence . the ants colony algorithm which synthesizes the integral differential control coefficients to control the speed and response of the motor The main objective. The results were compared between the DC motor without controller , using automatic tuning of PID controller, using the ants colony algorithm and fuzzy logic the last one have result from previous study . using MATLAB software to give the result.

المستخلص

إن تطوير المحركات ذات الأداء العالي مهم جدا في جميع المجالات الصناعية ويظهر أهمية التطور في الصناعات عالية الدقة التي تستجيب للاستجابة السريعة لتنظيم الحمل لأداء المهمة العالية. في هذا المشروع ، يتم إدخال المتحكم التناسبي التفاضلي التكاملي أكثر أنواع التحكم شيوعًا في الاستخدام الصناعي بسبب كفاءته البارزة وبساطته في التنفيذ والتطبيق ، وكيفية تجميع هذا النوع من أجهزة التحكم باستخدام الذكاء الاصطناعي. خوارزمية مستعمرات النمل التي تقوم بتوليف معاملات التحكم التفاضلية المتكاملة للتحكم في سرعة واستجابة المحرك الهدف الرئيسي. وتمت مقارنة النتائج بين محرك التيار المستمر بدون وحدة تحكم ، و باستخدام الضبط التلقائي لوحدة تحكم التناسبي التفاضلي التكاملي ، وباستخدام خوارزمية مستعمرات النمل والمنطق الضبابي ، وكانت النتيجة الأخيرة نتيجة للدراسة السابقة. وتم استخدام برنامج الماتلاب لإعطاء النتيجة

Table of Content

List of tables

List of Figures

List of Abbreviations

ACO	Ant colony optimization
DC	Direct Current
EMF	electromotive force
EP	Evolutionary Programming
GA	Genetic Algorithm
PID	Proportional – Integral – Derivative
PSO	Particle Swarm Optimization
ZN	Ziegler Nichols

CHAPTER ONE

INTRODUCTION

Chapter one

1. Introduction

1.1 Background

Today industries are increasingly demanding process automation in all sectors. Automation results into better quality, increased production and reduced costs. The variable speed drives, which can control the speed of A.C/D.C motors, are indispensable controlling elements in automation systems. Depending on the applications, some of them are fixed speed and some of the variable speed drives.

Direct currents (DC) motors have been used in variable speed drives for a long time. The versatile characteristics of dc motors can provide high starting torques which is required for traction drives. Control over a wide speed range, both below and above the rated speed can be very easily achieved. The methods of speed control are simpler and less expensive than those of alternating current motors.

In many industrial applications, dc-motors serve as simple devices for implementing velocity tracking tasks. Feedback control schemes, typically, are of the proportional–integral (PI) and proportional–integral–differential (PID)-type. Usually, the voltages for the speed control is supplied by thyristor-based phase controlled rectifiers operated at higher switching frequencies and are designed to provide a continuous armature current under various load situations

Swarm intelligence (SI), which is an artificial intelligence (AI) discipline, is concerned with the design of intelligent multi-agent systems by taking inspiration from the collective behavior of social insects such as ants, termites, bees, and wasps, as well as from other animal societies such as flocks of birds or schools of fish. Colonies of social insects have fascinated researchers for many years, and the mechanisms that govern their behavior remained unknown for a long time. Even though the single member of these colonies are non-sophisticated individuals, they are able to achieve complex tasks in cooperation. Coordinated colony behavior emerges from relatively simple actions or interactions between the colonies and the individual members. Many aspects of the collective activities of social insects are self organized and work without a central control. Clustering means the act of partitioning an unlabeled data set into groups of similar objects. Each group, called a cluster, consists of objects that are similar among themselves and dissimilar to objects of groups. In the past few decades, cluster analysis has played a central role in a variety of fields ranging from engineering (machine learning, artificial intelligence, pattern recognition, mechanical engineering, electrical engineering), computer sciences (web mining, spatial database analysis, textual document collection, image segmentation), life and medical. [1]

1.2 problem statement

DC motor speed needed to be control in precise speed for several application such as (biomedical application ex: ARTIFICAL HEART) using PID controller should optimize Dc motor speed .the only problem is the KP, KI, KD gains for PID controller should be tuned patricianly.

1.3 problem solution

Using Ant colony optimization (ACO) method to design auto tuning PID controller for DC motor speed.

1.4 Aim and objectives

The aim of this research is to optimize dc motor speed using ACO method for tuning PID parameter

Objectives:

- To apply (ACO) on PID controller for tuning.

- To simulate the system for performance evaluation.
- Build up the system using programming tools.

1.5 methodology

First the system will simulate dc motor for characterizing the transfer function. A PID controller with convert tuning method Z-N will be used to control system. ACO will then introduce to the system to tune PID controller parameter. Simulation will be run in MATLAB environment for performance evaluation. The system will be built in programming method for particular implementation.

1.6 thesis organization

Chapter one: proposed introduced the general overview, the problem also objective that will be achieved

Chapter Two: show the Related Works, overview about ant colony method ,PID and DC motor.

Chapter Three: discuss system design, show system block and all transfer function for DC motor

Chapter four: shown the result of simulation and discuss all result.

Chapter five: proposed conclusion and recommendation**.**

CHAPTER TWO

LITERATURE REVIEW

Chapter Two

2. Literature Review

2.1 Literature Review and Related Works Covered

Linares-Flores[2] al has described in the paper " DC motor velocity control through a DC/DC power converter", a design for smooth angular velocity controllers for a dc/dc Buck converter–dc motor system, wherein the effectiveness of the proposed controllers was verified only by numerical simulations. A smooth starter, based on the differential flatness approach, to regulate the velocity of a dc motor powered by a dc/dc Buck converter was presented. This starter was designed through a simplified second-order model, obtained by considering the motor armature inductance and the converter capacitor current to be negligible. Similarly they introduced an average GPI control law, implemented through a $\Sigma - \Delta$ modulator, for the angular velocity trajectory tracking task, exploiting the flatness of the combined system. To accomplish this, they employed the mathematical model obtained. Likewise, for the same system and task, the design of a dynamic output feedback controller was presented based on a fourth-order model deduced carried out by using the energy shaping and damping injection method.

WANG [3] has proposed an ACO based fuzzy controller to the brushless DC motor servo system. The fuzzy rules are optimized offline, while the parameters of the fuzzy controller are tuned online. The membership functions of the controller have been optimized by the ACS algorithm. Combined with the linear load observer, the comprehensive performance is enhanced. Its adaptability has been verified by the experimental results.

YASEEN,[4],ACO is a class of algorithms, whose first member, called Ant System, was initially proposed by Colorni, Dorigo and Maniezzo. The main underlying idea, loosely inspired by the behavior of real ants, is that of a parallel search over several constructive computational threads based on local problem data and on a dynamic memory structure containing information on the quality of previously obtained result. Use ACO concepts for goods propagation process, help to find a systematic, effective procedure to find good (but may be not the optimum) paths for goods propagation with respect to some predefined cost and constrains functions.

AST, ˇSKA ,[5],has take about a new method to the design of an optimal controller for continuous-time, continuous-state dynamic systems based on ACO. It has been described how the dynamics of the system can be transformed into a nondeterministic discrete automaton, where the original state space is quantized into a finite set of states and the transition function is a probability distribution over these states under an action from a discrete action set. The ACO algorithm does not require the assignment of heuristic values to the state-action pairs and is therefore straightforward to apply.

MANASA , [6],has described in his paper, “position control of a DC motor using PID controller”, optimal PID parameters were obtained by trial and error method. The simulation results indicate that the presented approach works effectively and provides a good relation between the objective function that optimizes the PID Controller and dynamic response of the system to be controlled. It is demonstrated that the position of the DC Motor is accurately controlled due to the finely tuned PID parameters, and the motor could reach the required position as quickly as possible without any error. Despite the enormous interest that modern control techniques have sparked among academics during the last three or four decades, PID controllers are still preferred in industrial process control. The reason is that controllers designed with the aid of modern control techniques are usually of high order, difficult to implement, and virtually impossible to re-tune on line. PID controllers, on the other hand, are simple, easy to implement, and comparatively easy to re-tune on line.

In “DC Motor Speed Control using PID Controllers” Nikunj A. Bhagat said that The PID controller for the DC motor speed control was successfully implemented. The effect of variation in load was studied and the response is found satisfactory. It is found that for the control of a first order plant like the DC motor, the PI controller is sufficient to obtain the desired performance.[7]

2.2 **DC motor (direct current)**

Electrical motors are everywhere around us. Almost all the electro-mechanical movements we see around us are caused either by a AC or a DC motor. Here we will be exploring DC motors. This is a device that converts DC electrical energy to a mechanical energy.

2.2.1 Principle of DC Motor

This DC or direct current motor works on the principal, when a current carrying conductor is placed in a magnetic field, it experiences a torque and has a tendency to move. This is known as motoring action. If the direction of current in the wire is reversed, the direction of rotation also reverses. When magnetic field and electric field interact they produce a mechanical force, and based on that the working principle of DC motor is established.

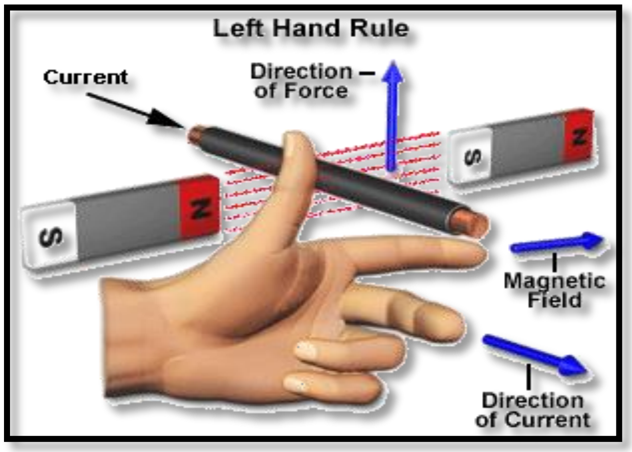


Figure (2.1): Fleming's left hand rule

The direction of rotation of a this motor is given by Fleming's left hand rule, which states that if the index finger, middle finger and thumb of your left hand are extended mutually perpendicular to each other and if the index finger represents the direction of magnetic field, middle finger indicates the direction of current, then the thumb represents the direction in which force is experienced by the shaft of the DC motor.

Structurally and construction wise a direct current motor is exactly similar to a DC generator, but electrically it is just the opposite. Here we unlike a generator we supply electrical energy to the input port and derive mechanical energy from the output port. We can represent it by the block diagram shown below.

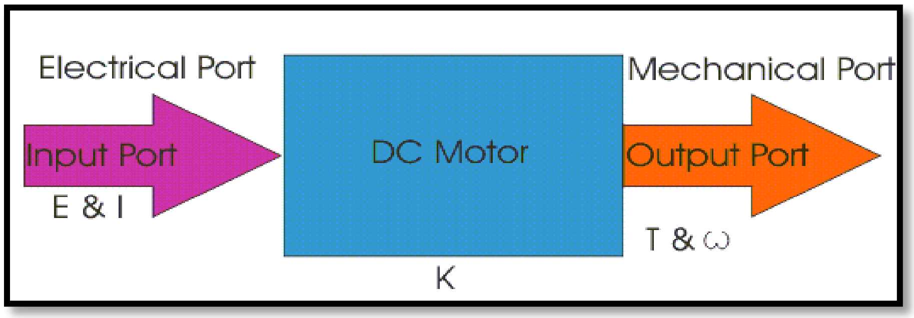


Figure (2.2): DC motor opposite energy

Here in a DC motor, the supply voltage E and current I is given to the electrical port or the input port and we derive the mechanical output i.e. torque T and speed ω from the mechanical port or output port. The input and output port variables of the direct current motor are related by the parameter K. So from the picture above we can well understand that motor is just the opposite phenomena of a DC generator, and we can derive both motoring and generating operation from the same machine by simply reversing the ports. [8]

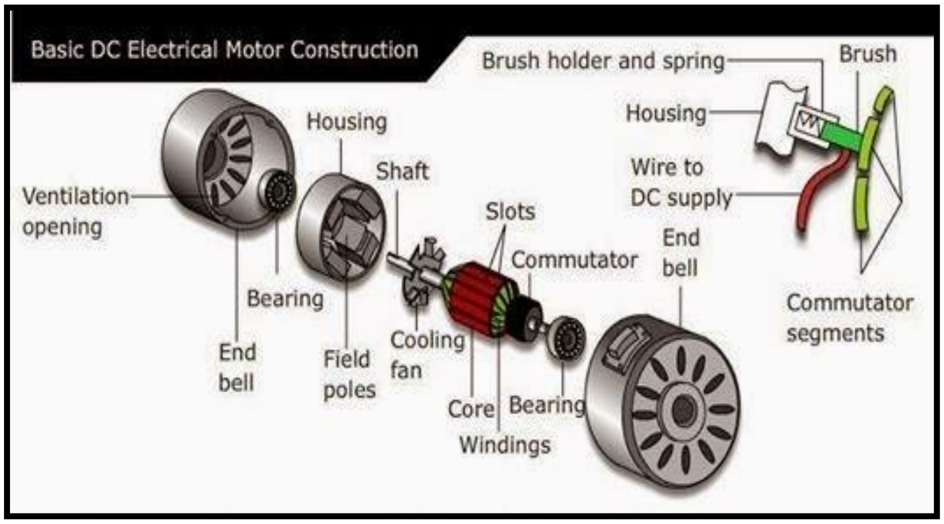


Figure (2.3): basic DC electrical motor construction

2.3 PID controller

During the 1930s three mode controllers with proportional, integral, and derivative (PID) actions became commercially available and gained widespread industrial acceptance. These types of controllers are still the most widely used controllers in process industries. This succeed is a result of many good features of this algorithm such as simplicity, robustness and wide applicability. Many various tuning methods have been proposed from 1942 up to now for gaining better and more acceptable control system response based on our desirable control objectives such as percent of overshoot, integral of absolute value of the error (IAE), settling time, manipulated variable behavior and etc. Some of these tuning methods have considered only one of these objectives as a criterion for their tuning algorithm and some of them have developed their algorithm by considering more than one of the mentioned criterion. [9]

PID controller has several important function: it provide feedback, it has the ability to eliminate steady state offset integral action, it can anticipate the future derivative action.

PID controllers have survived many changes in technology, from mechanics and pneumatics to microprocessors via electronic tubes, transistors, integrated circuits. The microprocessor has had a dramatic influence on the PID controller. Practically all PID controllers made today are based on microprocessors. This has given opportunities to provide additional features like automatic tuning, gain scheduling, and continuous adaptation.[10]

2.3.1 PID Structure

The structure of PID includes proportional term, integral term and derivation term. The PID controller is mainly to adjust appropriate proportional gain (k_p), integral gain (k_i), and differential gain (k_D) to achieve the optimal control performance. PID structure as shown in figure (2.4)

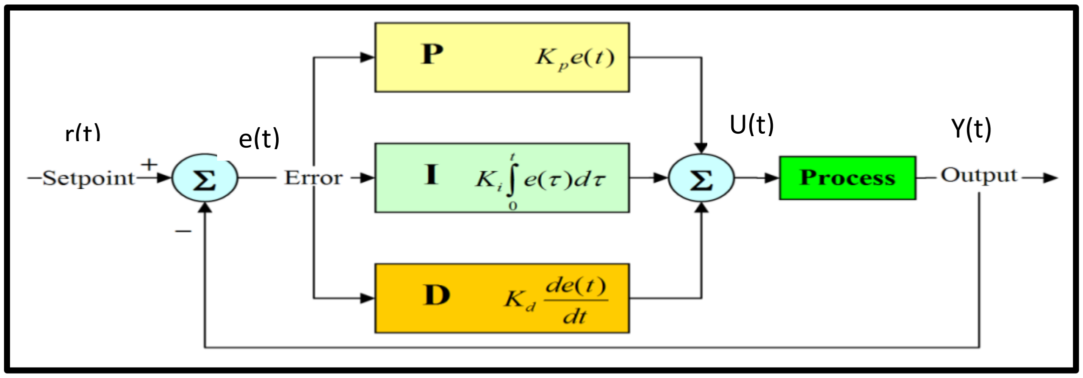


Figure (2.4) PID structure [11]

Where r(t)is the set point the value that system need to work perfect, e(t) error from system , u(t) controller output and y(t) is system output Relation between the input e(t) and output u(t) can be formulated the following:

$$E(t)=r(t)-y(t) \quad \textbf{(2.1)}$$

$$U(t)=k_p.e(t)+k_i\int_0^t e(t)dt+k_d.\frac{de(t)}{dt} \quad \textbf{(2.2)}$$

The transfer function is expressed :

$$C(s)=\frac{U(s)}{E(s)}=k_p+\frac{ki}{s}+k_d S \quad \textbf{(2.3)}$$

2.4 Efftect of PID controller parameter

each parameter in PID have effecte in the output response

2.4.1 P-Controller

Proportional or P- controller gives output which is proportional to current error e (t). It compares desired or set point with actual value or feedback process value. The resulting error is multiplied with proportional constant to get the output. If the error value is zero, then this controller output is zero. This controller requires biasing or manual reset when used alone. This is because it never reaches the steady state condition. It provides stable operation but always maintains the steady state error. Speed of the response is increased when the proportional constant Kc increases

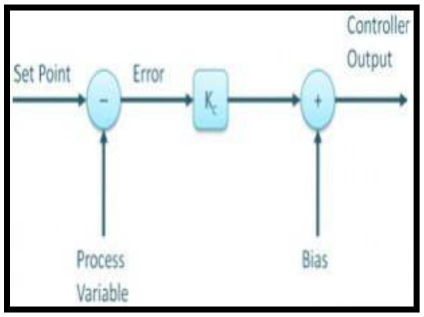


Figure (2.5): P-controller

2.4.2 I-controller

Due to limitation of p-controller where there always exists an offset between the process variable and set point, I-controller is needed, which provides necessary action to eliminate the steady state error. It integrates the error over a period of time until error value reaches to zero. It holds the value to final control device at which error becomes zero. Integral control decreases its output when negative error takes place. It limits the speed of response and affects stability of the system. Speed of the response is increased by decreasing integral gain Ki.

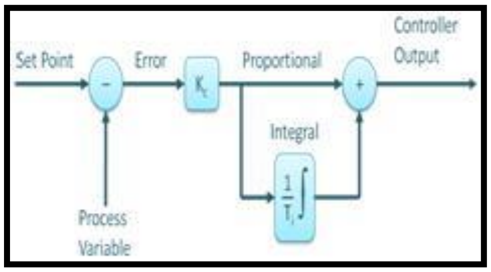


Figure (2.6): PI controller

In above figure, as the gain of the I-controller decreases, steady state error also goes on decreasing. For most of the cases, PI controller is used particularly where high speed response is not required.

While using the PI controller, I-controller output is limited to somewhat range to overcome the integral wind up conditions where integral output goes on increasing even at zero error state, due to nonlinearities in the plant.

2.4.3 D-Controller

I-controller doesn't have the capability to predict the future behavior of error. So it reacts normally once the set point is changed. D-controller overcomes this problem by anticipating future behavior of the error. Its output depends on rate of change of error with respect to time, multiplied by derivative constant. It gives the kick start for the output thereby increasing system response.

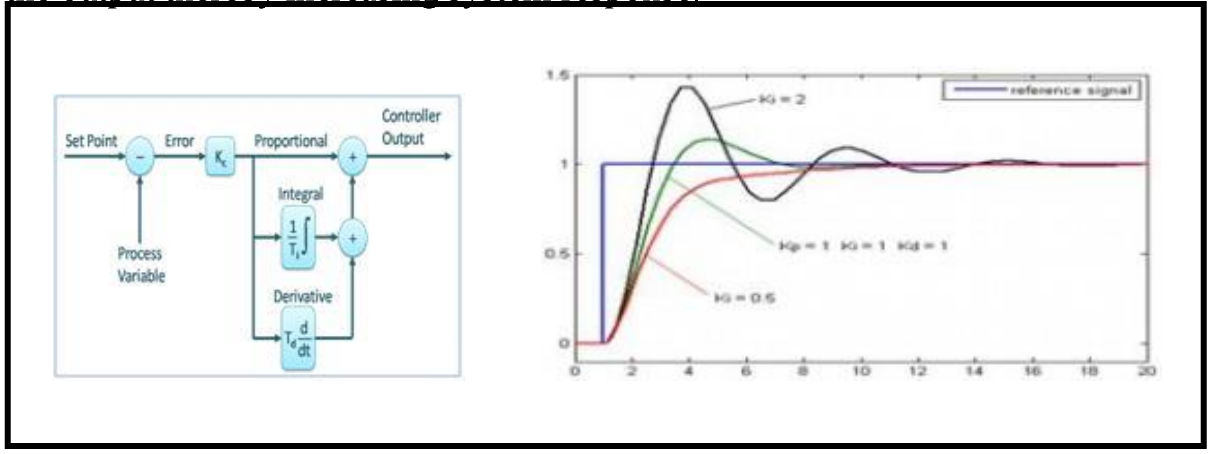


Figure (2.7): PID controller response

In the above figure response of D controller is more, compared to PI controller and also settling time of output is decreased. It improves the stability of system by compensating phase lag caused by I-controller. Increasing the derivative gain increases speed of response.

So finally we observed that by combining these three controllers, we can get the desired response for the system. Different manufactures designs different PID algorithms

2.5 Tuning PID controller

Several methods for tuning of controller parameters in PID controllers, that is, methods for finding proper values of k_p , k_i and k_d.

The main aim of the controller tuning if possible to obtain both of following

- Fast responses, and
- Good stability

Unfortunately, for practical systems these two wishes cannot be achieved simultaneously. In other words:

- The faster response, the worse stability, and
- The better stability, the slower response.

So, for the control system, we look for the following compromise: Acceptable stability and medium fastness of response Figure (2.5) illustrates the two mutual excluding wishes presented above, and the compromise. The figure shows the response in the process output variable due to a step change of the set point. (The responses are with three different controller gains in a simulated control system.) It is useful to have this compromise in mind when you perform controller tuning.

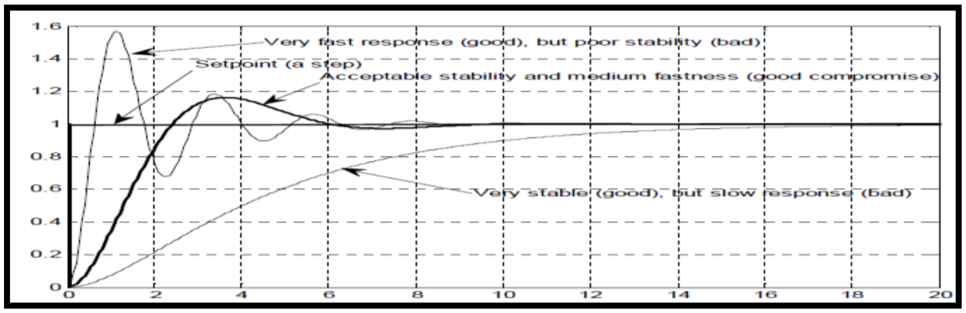


Figure (2.8) the response in the process output variable due to a step change of the set point [11]

There are several methods for tuning a PID loop. The most effective methods generally involve The development of some form of process model, then choosing P, I, and D based on the dynamic model parameters. Manual tuning methods can be relatively time consuming, particularly for systems with long loop times.

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

2.5.1 Manual tuning

If the system must remain online, one tuning method is to first set Ki and Kd values to zero. Increase the Kp until the output of the loop oscillates, then the Kp should be set to approximately

half of that value for a "quarter amplitude decay" type response. Then increase Ki until any offset is corrected in sufficient time for the process. However, too much Ki will cause instability.

Finally, increase Kd , if required, until the loop is acceptably quick to reach its reference after a load disturbance. However, too much Kd will cause excessive response and overshoot. A fast PID loop tuning usually overshoots slightly to reach the set point more quickly; however, some systems cannot accept overshoot, in which case an over-damped closed-loop system is required, which will require a Kp setting significantly less than half that of the Kp setting that was causing oscillation Effects of varying PID parameters (Kp,Ki,Kd) on the step response of a system. [12]

Table (2.1): Effects of increasing a parameter independently

Parameter	**Rise time**	**Overshoot**	**Settling time**	**Steady-state error**	**stability**
Kp	Decrease	Increase	Small change	Decrease	Degrade
Ki	Decrease	Increase	Increase	Eliminate	Degrade
Kd	Minor change	Decrease	Decrease	No effect in theory	Improve if small

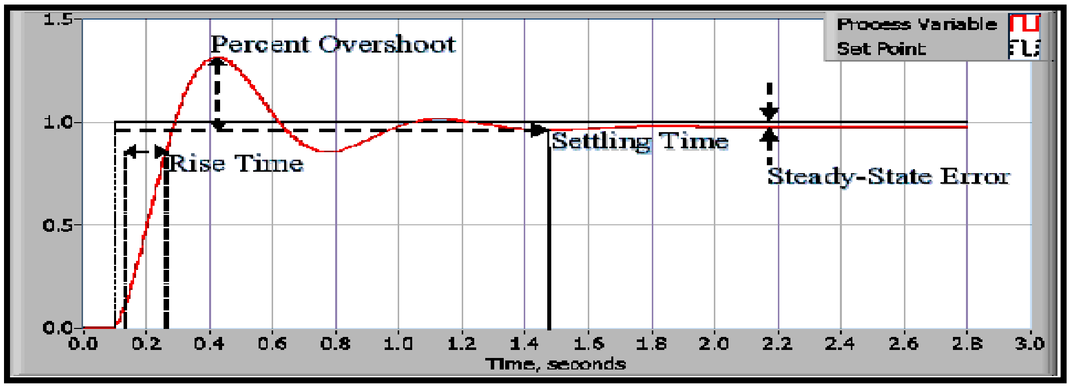


Figure (2.9): parameter of response signal

With the use of low cost simple ON-OFF controller only two control states are possible, like fully ON or fully OFF. It is used for limited control application where these two control states are enough for control objective. However oscillating nature of this control limits its usage and hence it is being replaced by PID controllers.

PID controller maintains the output such that there is zero error between process variable and set point/ desired output by closed loop operations.

2.5.2 Ziegler- Nichols method

J.G. Ziegler and N.B. Nichols published two tuning methods for PID controllers in 1942. This article describes in detail how to apply one of the two methods, sometimes called the Ultimate Cycling method. (The other one is called the process reaction-curve method.) I have seen many cryptic versions of this procedure, but they leave a lot open for interpretation, and a practitioner may run into difficulties using one of these abbreviated procedures.

Before we get started, here are a few very important notes:

- Read the entire procedure before beginning.
- This tuning method does not work for inherently unstable processes like temperature control of exothermic reactions.
- This procedure cannot be used if the Process Variable oscillates when the controller is in Manual control mode. If the loop is already oscillating in Auto, make sure the cycling stops in Manual.
- If the controller drives a control valve or damper, and this device has dead band or stiction problems, this tuning method cannot be used and will lead to inaccurate results and poor tuning at best.
- Care should be taken to always keep the process in a safe operating region.
- An experienced operator should oversee the entire test and must have the authority to terminate the test at any time.
- Keep note of the original controller settings and leave them with the operator in case he/she needs to revert back to them later. Process conditions can change significantly, and your new tuning settings might only work for the conditions at which the process tests were done.

The steps below apply to a controller with a Controller Gain setting. If your controller uses Proportional Band instead, do the reciprocal of any Controller Gain changes. E.g. if the procedure calls for increasing the Controller Gain by 50%, the Proportional Band should be decreased by 50%, etc.

To apply the Ziegler-Nichols Closed-Loop method for tuning controllers, follow these steps:

1. Stabilize the process. Make sure no process changes (e.g. product changes, grade changes, load changes) are scheduled.
2. If the loop is currently oscillating, make sure that the Process Variable stops oscillating when the controller is placed in Manual mode.
3. Remove Integral action from controller.
    - If your controller uses Integral Time (Minutes or Seconds per Repeat), set the Integral parameter to a very large number (e.g. 9999) to effectively turn it off.
    - If your controller uses Integral Gain (Repeats per Minute or Repeats per Second), set the Integral parameter to Zero.
4. Remove Derivative action by setting the Derivative parameter to Zero.
5. Place the controller in Automatic control mode if it is in Manual mode.
6. Make a Set Point change and monitor the result.
7. If the Process Variable does not oscillate at all, double the Controller Gain.
8. If the Process Variable oscillates and the amplitude of the peaks decreases, increase the Controller Gain by 50% (or less if you are getting close to a constant amplitude).

9. If the Process Variable oscillates and the amplitude of the peaks increases, decrease the controller gain by 50% (or less if you are getting close to a constant amplitude).
10. If the Process Variable or Controller Output hits its upper or lower limits, decrease the controller gain by 50%. The Process Variable and Controller Output must oscillate freely for this method to work.
11. If the oscillations have died out, go to Step 6.
12. If the loop is oscillating, but not with a constant amplitude, repeat Steps 8, 9, and 10 until oscillations with a constant amplitude are obtained.
13. If the Process Variable is oscillating with constant amplitude, and neither the Process Variable nor the Controller Output hits its limits, do the following:
    - Take note of the “Ultimate” Controller Gain (K_u). If your controller has Proportional Band, note down the “Ultimate Band” (PB_u).
    - Measure the period of the oscillation (t_u). If your controller’s Integral and Derivative units are in minutes, measure tu in minutes. It the controller uses seconds, measure t_u in seconds.
14. Cut the Controller Gain in half to let the control loop stabilize while you do the calculations.
15. Calculate new controller settings using the equations below, enter them into the controller, and make a Set Point change to test them.

The Ziegler-Nichols Closed-Loop Tuning Method

The Ziegler-Nichols tuning rules were designed for a ¼ amplitude decay response. This results in a loop that overshoots its set point after a disturbance or set point change. The response in general is somewhat oscillatory, the loop is only marginally robust and it can withstand only small changes process conditions. I recommend using slightly different settings (also shown below) to obtain a robust loop with increased stability.

Rules for a PI Controller

The PI tuning rule can be used on controllers with interactive or non-interactive algorithms.

Controller Gain (Kc)

- Ziegler-Nichols Rule: Kc = 0.45 K_u
- For robust control use: Kc = 0.22 K_u

Proportional Band (PB)

- Ziegler-Nichols Rule: PB = 2.2 PB_u
- For robust control use: PB = 4.4 PB_u

Integral Time in Minutes per Repeat or Seconds per Repeat

- Ziegler-Nichols Rule: Ti = 0.83 t_u
- For level control (integrating processes) use: Ti = 1.6 t_u

Integral Gain in Repeats per Minutes or Repeats per Seconds

- Ziegler-Nichols Rule: Ki = 1.2 / t_u
- For level control (integrating processes) use: Ki = 0.6 / t_u

Rules for a PID Controller

The PID tuning rule was designed for a controller with the Interactive algorithm. The tuning settings should be converted for use on controllers with Non-interactive and Parallel algorithms.

Controller Gain (Kc)

- Ziegler-Nichols Rule: Kc = 0.6 K_u

- For robust control use: Kc = 0.3 K_u

Proportional Band (PB)

- Ziegler-Nichols Rule: PB = 1.7 PB_u
- For robust control use: PB = 3.3 PB_u

Integral Time in Minutes per Repeat or Seconds per Repeat

- Ziegler-Nichols Rule: Ti = 0.5 t_u
- For level control (integrating processes) use: Ti = 1.0 t_u

Integral Gain in Repeats per Minutes or Repeats per Seconds

- Ziegler-Nichols Rule: Ki = 2.0 / t_u
- For level control (integrating processes) use: Ki = 1.0 / t_u

Derivative Time or Derivative Gain

- Td or Kd = 0.125 x t_u

2.6 optimization methods

Proportional Integral Derivative (PID) control schemes continue to provide the simplest and effective solutions to most of the control engineering applications today. However PID controller are poorly tuned in practice with most of the tuning done manually which is difficult and time consuming. This article comes up with a hybrid approach involving Genetic Algorithm (GA), Evolutionary Programming (EP), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). The proposed hybrid algorithm is used to tune the PID parameters and its performance has been compared with the conventional methods like Ziegler Nichols and Cohen Coon method. The results obtained reflect that use of heuristic algorithm based controller improves the performance of process in terms of time domain specifications, set point tracking, and regulatory changes and also provides an optimum stability. Speed control of DC motor process is used to assess the efficacy of the heuristic algorithm methodology.

2.6.1 GA based tuning of the controller

The optimal value of the PID controller parameters ,K_P,K_I,K_d is to be found using GA. All possible sets of controller parameters values are particles whose values are adjusted to minimize the objective function, which in this case is the error criterion, and it is discussed in detail. For the PID controller design, it is ensured the controller settings estimated results in a stable closed-loop system.

Genetic Algorithms are a stochastic global search method that mimics the process of natural evolution. It is one of the methods used for optimization. John Holland formally introduced this method in the United States in the 1970 at the University of Michigan. The continuing performance improvement of computational systems has made them attractive for some types of optimization. The genetic algorithm starts with no knowledge of the correct solution and depends entirely on responses from its environment and evolution operators such as reproduction, crossover and mutation to arrive at the best solution By starting at several independent points and searching in parallel, the algorithm avoids local minima and converging to sub optimal solutions.

2.6.2 EP based tuning of the controller

EP is generally used to optimize real-valued continuous functions. EP uses selection and mutation operators and does not use the crossover operator. The focus is on the observed characteristics of the population. The selection operator is used to determine chromosomes for mating in order to generate new chromosomes.

There are two important ways in which EP differs from GAs.

First, there is no constraint on the representation. The typical GA approach involves encoding the problem solutions as a string of representative tokens, the genome. In EP, the representation follows from the problem. A neural network can be represented in the same manner as it is implemented, for example, because the mutation operation does not demand a linear encoding

Second, the mutation operation simply changes aspects of the solution according to a statistical distribution which weights minor variations in the behavior of the offspring as highly probable and substantial variations as increasingly unlikely.

2.6.3 PSO based tuning of the controller

PSO is one of the optimization techniques and kind of evolutionary computation technique. The technique is derived from research on swarm such as bird flocking and fish schooling. In the PSO algorithm, instead of using evolutionary operators such as mutation and crossover to manipulate

algorithms, for a d-variable optimization Problem, a flock of particles are put into the d-dimensional

Search space with randomly chosen velocities and positions knowing their best values

2.7 Ant colony optimization method

Ant algorithms were inspired by the observation of real ant colonies. Ants are social insects, that is, insects that live in colonies and whose behavior is directed more to the survival of the colony as a whole than to that of a single individual component of the colony. Social insects have captured the attention of many scientists because of the high structuration level their colonies can achieve, especially when compared to the relative simplicity of the colony’s individuals. An important and interesting behavior of ant colonies is their foraging behavior, and, in particular, how ants can find shortest paths between food sources and their nest. While walking from food sources to the nest and vice versa, ants deposit on the ground a substance called *pheromone*, forming in this way a pheromone trail. Ants can smell pheromone and, when choosing their way, they tend to choose, in probability, paths marked by strong pheromone concentrations. The pheromone trail allows the ants to find their way back to the food source (or to the nest). Also, it can be used by other ants to find the location of the food sources found by their nestmates. It has been shown experimentally that this pheromone trail following behavior can give rise, once employed by a colony of ants, to the emergence of shortest paths. That is, when more paths are available from the nest to a food source, a colony of ants may be able to exploit the pheromone trails left by the individual ants to discover the shortest path from the nest to the food source and back.

In the following we discuss first the nature inspired characteristics of artificial ants, and then how they differ from real ants

Similarities and differences with real ants

Most of the ideas of ACO stem from real ants. This algorithm consists of four main components:

▪ Ant:

Ants are the agents of the algorithm that explore and exploit the searching area.

▪ Pheromone:

The agents (ants) drop pheromone in the search space and the quantities of these pheromones indicate the probability of that path to be chosen by other agents. The intensity of the pheromone on the paths may be considered as a memory of the system.

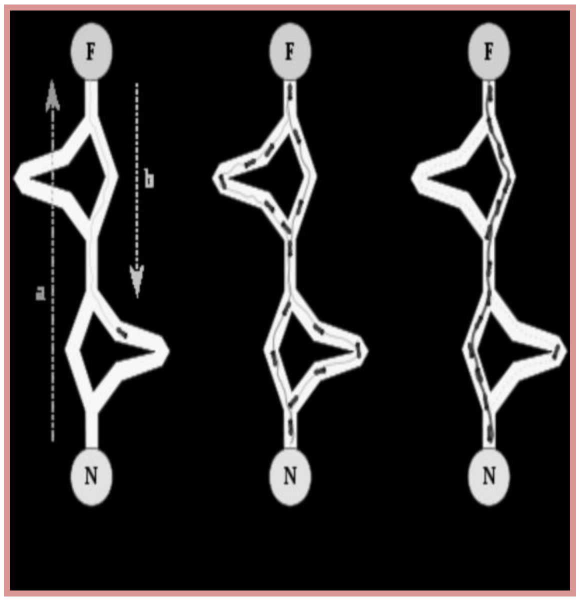
▪ Daemon action:

It is used to gather global information of the system that cannot be done by a single ant. If the convergence is slowly the daemon action adds an extra pheromone to accelerate the process.

▪ Decentralized control:

The decentralized control operated by the algorithm makes it robust and flexible.

Is a probabilistic technique for solving computational problems which can be reduced to find good paths through graphs. In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail; returning and reinforcing it if they eventually find food. Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also has the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. Thus, when one ant finds a good (i.e., short) path from the colony to a food source, other ants are more likely to follow that path, and positive feedback eventually leads all the ants following a single path. The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the graph representing the problem to solve.



Figure(2.10): ant try to find the shortest path between a food and the nest

The original idea comes from observing the exploitation of food resources among ants, in which ant's individually limited cognitive abilities have collectively been able to find the shortest path between a food source and the nest.

1. The first ant finds the food source (F), via any way (a), then returns to the nest (N), leaving behind a trail pheromone (b)
2. Ants indiscriminately follow four possible ways, but the strengthening of the runway makes it more attractive as the shortest route.
3. Ants take the shortest route, long portions of other ways lose their trail pheromones

In a series of experiments on a colony of ants with a choice between two unequal length paths leading to a source of food, biologists have observed that ants tended to use the shortest route. A model explaining this behavior is as follows:

1. An ant (called "blitz") runs more or less at random around the colony;
2. If it discovers a food source, it returns more or less directly to the nest, leaving in its path a trail of pheromone;

3. These pheromones are attractive, nearby ants will be inclined to follow, more or less directly, the track;
4. Returning to the colony, these ants will strengthen the route;
5. If there are two routes to reach the same food source then, in a given amount of time, the shorter one will be traveled by more ants than the long route;
6. The short route will be increasingly enhanced, and therefore become more attractive;
7. The long route will eventually disappear because pheromones are volatile;
8. Eventually, all the ants have determined and therefore "chosen" the shortest route.

The basic philosophy of the algorithm involves the movement of a colony of ants through the different states of the problem influenced by two local decision policies, viz., trails and attractiveness. Thereby, each such ant incrementally constructs a solution to the problem. When an ant completes a solution, or during the construction phase, the ant evaluates the solution and modifies the trail value on the components used in its solution. This pheromone information will direct the search of the future ants. As a very good example, ant colony optimization algorithms have been used to produce near-optimal solutions to the travelling salesman problem. The first ACO algorithm was called the Ant system and it was aimed to solve the travelling salesman problem, in which the goal is to find the shortest round-trip to link a series of cities.

To achieve the most direct path, each ant moves randomly in different paths. Ants will initially wander randomly, and upon finding food return to their colony. Meanwhile, there be laying down pheromone trails. If other ants find those paths they are more likely not to keep travelling at random, but follow the trail, returning and reinforcing it if they eventually find food. The more time it takes an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path is marched over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. This methodology allows ants to convergence, hence founding the optimum path. In ACO algorithm, there are three relevant aspects to be considered:

***Initialization*:** the ants must be initialized, in order to follow paths.

Building paths: the ants will determine the next destiny according to certain probability. The probability is obtained according to equation (2.4)

$$P_{ij}^{k}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \qquad (2.4)$$

In the equation, τij represents the pheromone trail and ηij the visibility between the two cities, while α and β are adjustable parameters that control the relative weight of trail intensity and visibility.

Where N_i^k is the number of destinies visited by the k^{th} ant from i to j, where j ∈ N_i^k . τ_{ij} is the amount of pheromones laid in the path between i and j. η_{ij} is the visibility, inverse of distance between i and j.

Pheromones are a very important component in the algorithm, since it contributes to the memory of the system. Agents leave a trail of pheromones that make paths to be chosen with higher probability by the next agents.

Pheromone operation: in ACO model, each ant leaves a pheromone trail in the road travelled according to (2.5)

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \quad \forall (i,j) \in L \qquad (2.5)$$

$\Delta\tau_{ij}^{k}(t)$ is the amount of laid pheromone in the segment visited by the k^{th} ant in the actual iteration. The amount of pheromones laid down in each segment is related to the quality of the solution(2.6)

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{1}{L_k} & \forall(i,j) \in L \\ 0 & otherwise \end{cases}$$

(2.6)

Where L is the total length of the road travelled for the iteration.

The pheromone trail associated to each segment is reduced by an evaporation constant ρ

According to (2.7)

$$\tau_{ij} = (1-\rho)\tau_{ij} \quad \forall(i,j) \in L$$

(2.7)

Where $0 < \rho < 1$. The evaporation process is a forgetting factor, used to avoid the unnecessary pheromone accumulation, allowing forgetting bad decisions.

CHAPTER THREE

SYSTEM MODEL

Chapter Three

3. System Model

3.1 Model of Dc Motor:

In order to draw equivalent circuit of DC motor, divide it into two main parts.

- Armature
- Field poles.

Voltage is induced in armature. Therefore voltage source must be included. According to lenses law the direction of the induced voltage is opposite to the source, hence the polarity of voltage source should opposite to the external current flowing through it. The armature winding have some resistance which results in I^2R losses. To account for those losses, a resistor is added in series with the voltage source. Due to commutation ,sliding with armature conductor. This voltage drop always occurs in one direction. To account for this drop, another small dc voltage source is added in armature equivalent circuit.

Next is to draw equivalent circuit for field poles. Field poles consist of iron rounded with wire (conductor). The purpose is to produce magnetic flux. The effect produced by pole winding is similar to inductor in electrical circuit. Since the winding conductor must have some resistance so it wall cause I^2R losses. To compensate for I^2R losses, a resistor is added. A variable resistor is added in series which accounts for the real variable resistor in field winding. This variable resistor is used in speed control via field flux. For simplicity both fixed and variable resistors are combined in one variable resistor. Equivalent circuit of stator (field) of dc motor is shown in figure (3.1)

L_a = Inductor representing inductance of field poles

R_a = Resistor of field winding including external added resistor.

There is no electrical connection between stator and rotor of DC motor therefore in final circuit they should be electrically as shown in following figure.

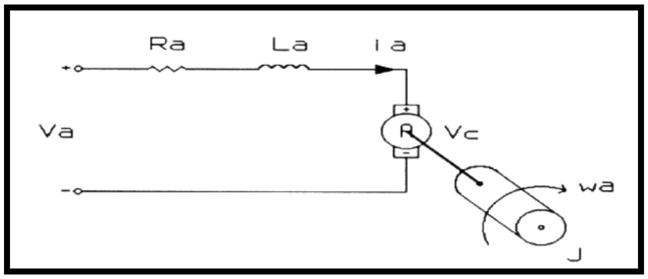


Figure (3.1) : Electrical representation of a dc motor.

3.1.1 Electrical Characteristics

The equivalent electrical circuit of a dc motor is illustrated in Fig. 3.1 It can be represented by a voltage source (Va) across the coil of the armature. The electrical equivalent of the armature coil can be described by an inductance (La) in series with a resistance (Ra) in series with an induced voltage (Vc) which opposes the voltage source. The induced voltage is generated by the rotation of the electrical coil through the fixed flux lines of the permanent magnets. This voltage is often referred to as the back emf (electromotive force).

A differential equation for the equivalent circuit can be derived by using Kirchoff's voltage law around the electrical loop. Kirchoff's voltage law states that the sum of all voltages around a loop must equal zero, or

$$V_a - V_{Ra} - V_{La} - V_c = 0 \qquad (3.1)$$

According to Ohm's law, the voltage across the resistor can be represented as

$$V_{Ra} = i_a R_a \qquad (3.2)$$

where Ia is the armature current. The voltage across the inductor is proportional to the change of current through the coil with respect to time and can be written as

$$V_{la} = l_a \frac{d}{dt} i_a \qquad (3.3)$$

where L_a is the inductance of the armature coil. Finally, the back emf can be written as

$$V_c = k_v \omega_a \tag{3.4}$$

where k_v is the velocity constant determined by the flux density of the permanent magnets, the reluctance of the iron core of the armature, and the number of turns of the armature winding. ω_a is the rotational velocity of the armature. Substituting eqns. (3.2), (3.3), and (3.4) into eqn. (3.1) gives the following differential equation:

$$V_a - i_a R_a - L_a \frac{d}{dt} i_a - k_v \omega_a = 0 \tag{3.5}$$

3.1.2 Mechanical Characteristics

Performing an energy balance on the system, the sum of the torques of the motor must equal zero. Therefore,

$$T_e - T_{\omega'} - T_\omega - T_L = 0 \tag{3.6}$$

Where T_e is the electromagnetic torque, $T_{w'}$ is the torque due to rotational acceleration of the rotor, Tw is the torque produced from the velocity of the rotor, and T_L is the torque of the mechanical load. The electromagnetic torque is proportional to the current through the armature winding and can be written as

$$T_e = k_t i_a \tag{3.7}$$

Where k_t is the torque constant and like the velocity constant is dependent on the flux density of the fixed magnets, the reluctance of the iron core, and the number of turns in the armature winding. $T_{w'}$ can be written as

$$T_{\omega'} = J \frac{d}{dt} \omega_a \tag{3.8}$$

Where J is the inertia of the rotor and the equivalent mechanical load. The torque associated with the velocity is written as

$$T_\omega = B \omega_a \tag{3.9}$$

Where B is the damping coefficient associated with the mechanical rotational system of the machine.

Substituting eqns. (3.7), (3.8), and (3.9) into eqn. (3.6) gives the following differential equation:

$$k_t i_a - J\frac{d}{dt}\omega_a - B\omega_a - T_L = 0 \tag{3.10}$$

3.1.3 State Space Representation

The differential equations given in eqns. (3.5) and (3.10) for the armature current and the angular velocity can be written as

$$\frac{d}{dt}i_a = -\frac{R_a}{L_a}i_a - \frac{k_v}{L_a}\omega_a + \frac{V_a}{L_a} \tag{3.11}$$

$$\frac{d}{dt}\omega_a = \frac{k_t}{J}i_a - \frac{B}{J}\omega_a - \frac{T_L}{J} \tag{3.12}$$

Which describe the dc motor system. Putting the differential equations into state space form gives

$$\frac{d}{dt}\begin{bmatrix} i_a \\ \omega_a \end{bmatrix} = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_v}{L_a} \\ \frac{k_t}{J} & -\frac{B}{J} \end{bmatrix}\begin{bmatrix} i_a \\ \omega_a \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix}\begin{bmatrix} V_a \\ T_L \end{bmatrix} \tag{3.13}$$

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} i_a \\ \omega_a \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} V_a \\ T_L \end{bmatrix} \tag{3.14}$$

which is expressed symbolically as

$$\frac{d}{dt}\underline{x} = \underline{\underline{A}}\,\underline{x} + \underline{\underline{B}}\underline{u} \tag{3.15}$$

$$\underline{y} = \underline{C}\underline{x} + \underline{D}\underline{u} \tag{3.16}$$

Where $\underline{x}$ is the state vector, $\underline{u}$ is the input vector, and $\underline{y}$ is the output vector.

3.1.4 Transfer Function Block Diagram

A block diagram for the system can be developed from the differential equations given in eqns. (3.11) and (3.12). Taking the Laplace transform of each equation gives

$$sI_a(s) - i_a(0) = -\frac{R_a}{L_a} I_a(s) - \frac{k_v}{L_a}\Omega_a(s) + \frac{1}{L_a} V_a(s) \tag{3.17}$$

$$s\Omega_a(s) - \omega_a(0) = \frac{k_t}{J} I_a(s) - \frac{B}{J}\Omega_a(s) - \frac{1}{J} T_L(s) \tag{3.18}$$

If perturbations around some steady state value are considered, the initial conditions go to zero and all the variables become some change around a reference state, and the equations can be expressed as follows:

$$I_a(s) = \frac{-k_v\Omega_a(s) + V_a(s)}{L_a s + R_a} \tag{3.19}$$

$$\Omega_a(s) = \frac{-k_t I_a(s) - T_L(s)}{Js + B} \tag{3.20}$$

The above equations can then easily be put into block diagram form. The block diagram obtained from these equations for a permanent magnet dc motor is shown in Fig. 3.2

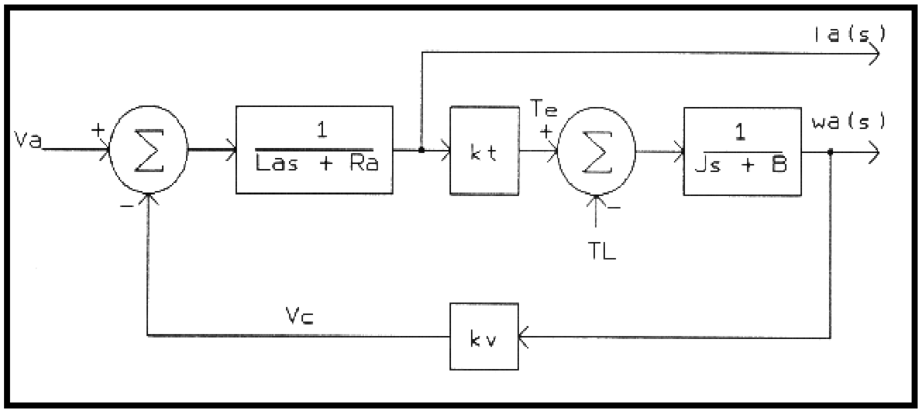


Figure (3.2):. Block diagram representation

The block diagram in Fig. 3.2 can be simplified by making the assumption that the load torque is constant. In the case of a sun tracking servo system, the only load torque to be concerned with is the friction in the system, which is relatively constant while the motor is moving. Since the change in T_L is zero, it does not need to appear in the block diagram. Also, if one only focuses on the angular velocity as the response of interest, the block diagram becomes as shown in Fig. 3.3.

This block diagram is then easily reduced by block diagram algebra to an overall transfer function. Several steps in this process are shown in Fig.3.4, with the overall transfer function between the output angular velocity and input applied voltage given within the last block in Fig. 3.4.

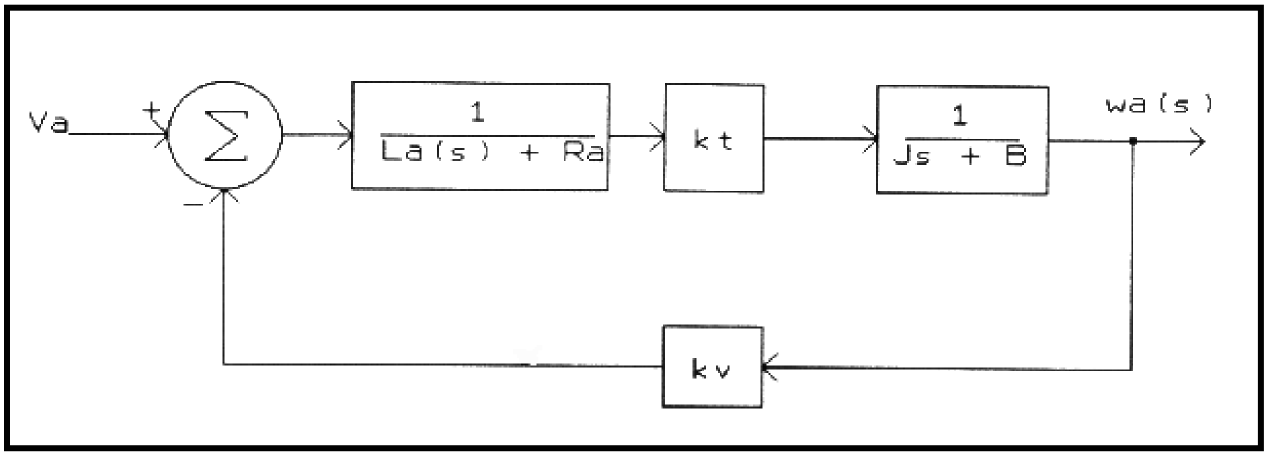


Figure (3.3): Block diagram of the dc motor as modeled in this study.

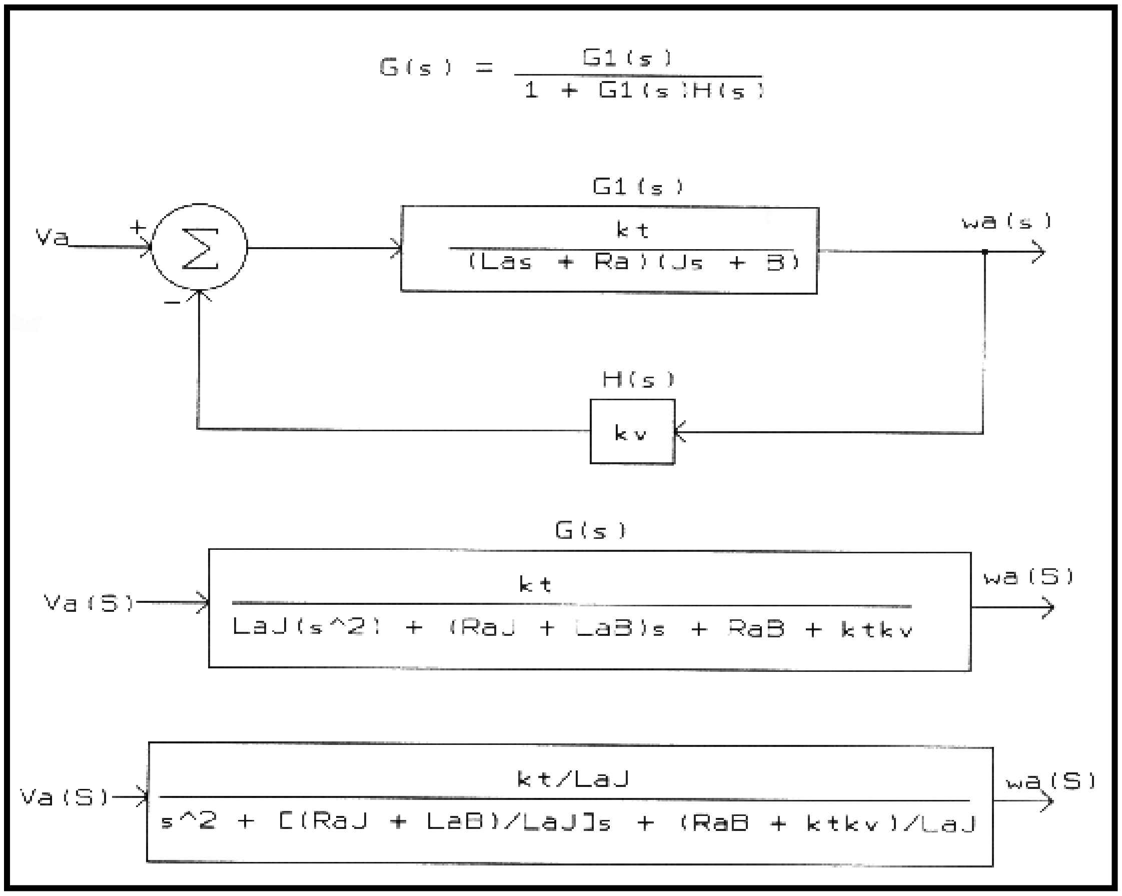

G(s) = G1(s) / 1 + G1(s)H(s)
G1(s)
Va
+
−
kt / (Las + Ra)(Js + B)
wa(s)
H(s)
kv
G(s)
Va(S)
kt / LaJ(s^2) + (RaJ + LaB)s + RaB + ktkv
wa(S)
Va(S)
kt/LaJ / s^2 + [(RaJ + LaB)/LaJ]s + (RaB + ktkv)/LaJ
wa(S)

Figure (3.4) Overall transfer function for the dc motor

3.2 Simulation Model Of Dc Motor:

According to the figure(3.3) design the Dc motor whereas:

Armature resistance (Ra) = 0.5 Ω

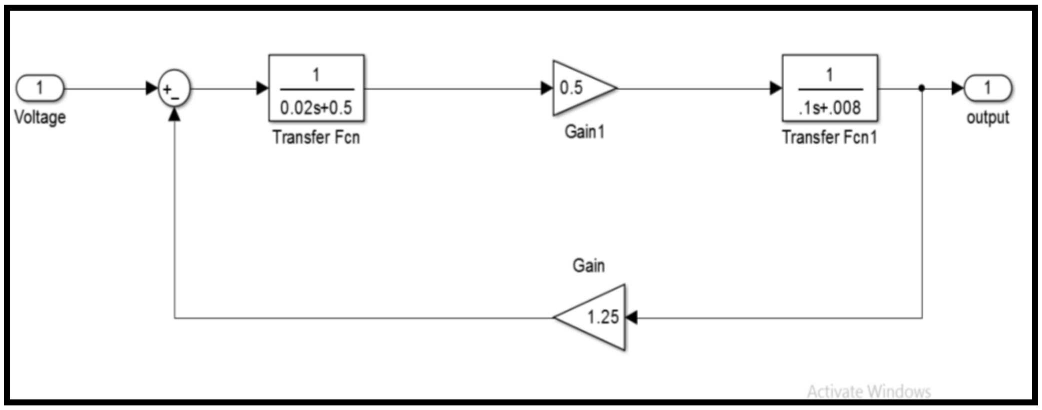
Torque constant (Kt) = 0.5 Nm/A

Armature inductance (La) =0.02 H

Friction constant (B) =0.008 Nms/rad

Rotor inertia (J) = 0.1 Kgm^2

Back emf constant (Kb) = 1.25 Vs/rad



Figure(3.5) simulation model of Dc motor

3.3 PID Algorithm:

PID controllers are found in a wide range of applications for industrial process control. Approximately 95% of the closed loop operations of industrial automation sector use PID controllers. PID stands for Proportional-Integral-Derivative. These three controllers are combined in such a way that it produces a control signal.

As a feedback controller, it delivers the control output at desired levels. Before microprocessors were invented, PID control was implemented by the analog electronic components. But today all PID controllers are processed by the microprocessors. Programmable logic controllers also have the inbuilt PID controller instructions. Due to the flexibility and reliability of the PID controllers, these are traditionally used in process control applications.

3.4 PID with DC motor:

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. In our project, first of all we have obtained the P-I-D parameters for our system. Then we have constituted our own P-I-D algorithm with coding.

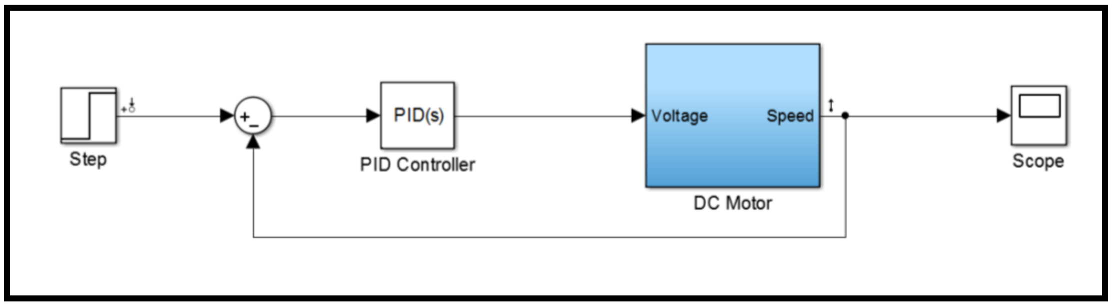


Figure (3.6): simulation of DC with PID

3.5 Ant colony modeling in matlab:

Algorithm code has been written using MATLAB. The code generates the values of the coefficients of the PID based on the values of the kinetic parameters. This parameter adjustable according to Last parameters obtained. This process is repeated with the number of iteration, when reach the maximum the program stopped.

3.6 System block diagram:

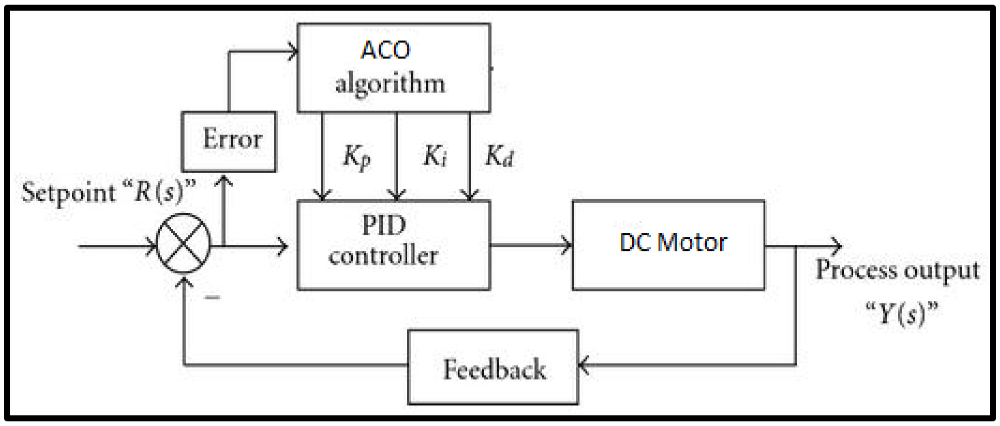
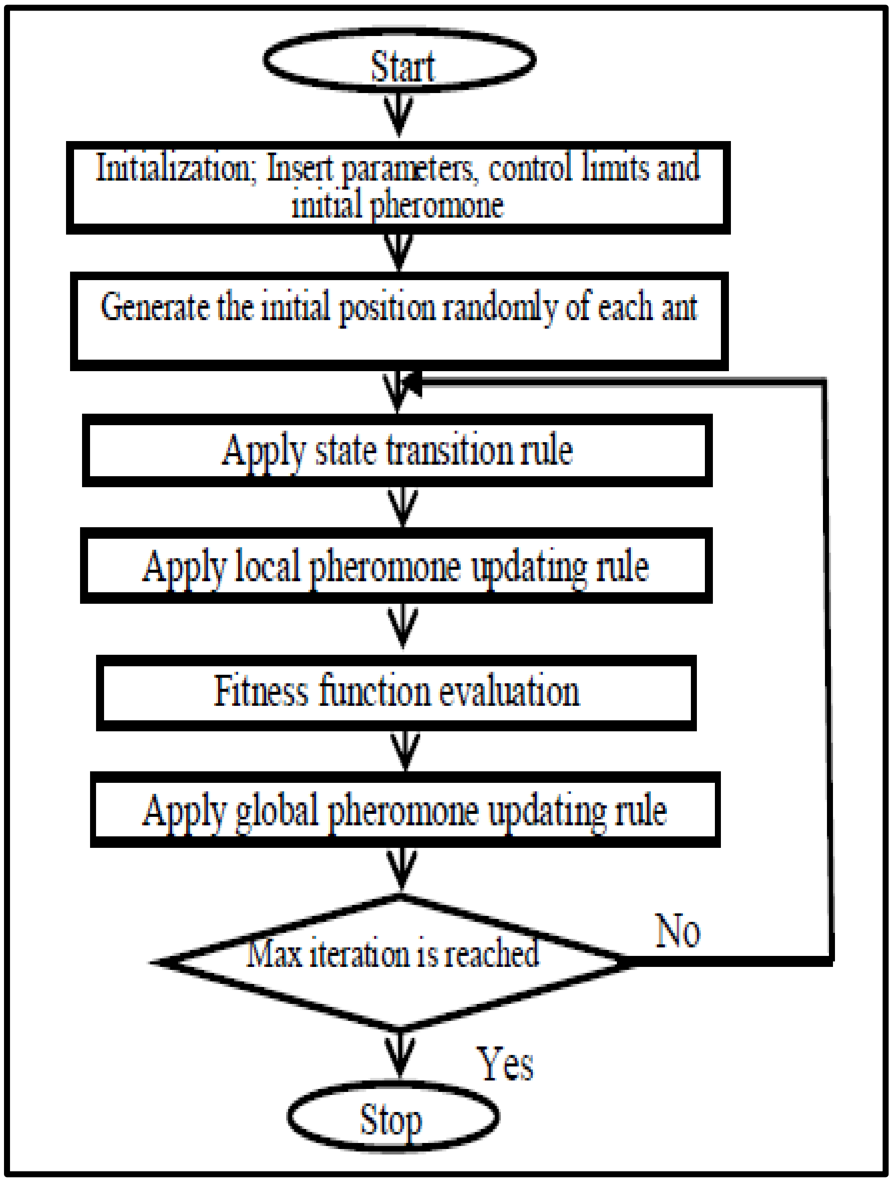


Figure (3.7): system block diagram

Our system, as shown in fig (3.8), consists of the PID controller which used to control the power and match the motor speed with the desired speed. The system inputs represent the set points

R(s) which are specific for each application. The set points will be presented To the PID controller, a sample of the process output will be fed back to the input for the purpose of constant check, the error will be entered to the ACO algorithm to regenerate the necessary parameters (Kp,Ki,Kd) required for better tuning of motor speed by the PID. Our system integrates the PID controller with ACO algorithm.

3.7 System flow chart:


Start
Initialization; Insert parameters, control limits and initial pheromone
Generate the initial position randomly of each ant
Apply state transition rule
Apply local pheromone updating rule
Fitness function evaluation
Apply global pheromone updating rule
Max iteration is reached
No
Yes
Stop

Figure(3.8):flow chart of ACO algorithm

Chapter Four

Result and Discussion

Chapter Four

Result and Discussion

In this chapter, after simulating the process control of speed control system model by using MATLAB/SIMULINK &M.FILE, the performance of the controllers was analyzed by comparing the output signal represented by the graph.

in below figure(4.1) shown the dc motor without any controller after that shown with figures the response using ziegler-nichols method to tune PID controller and ACO . Then have compare between ACO and PSO to show the different between two of the swarm intelligent.

4.1 systems without control system

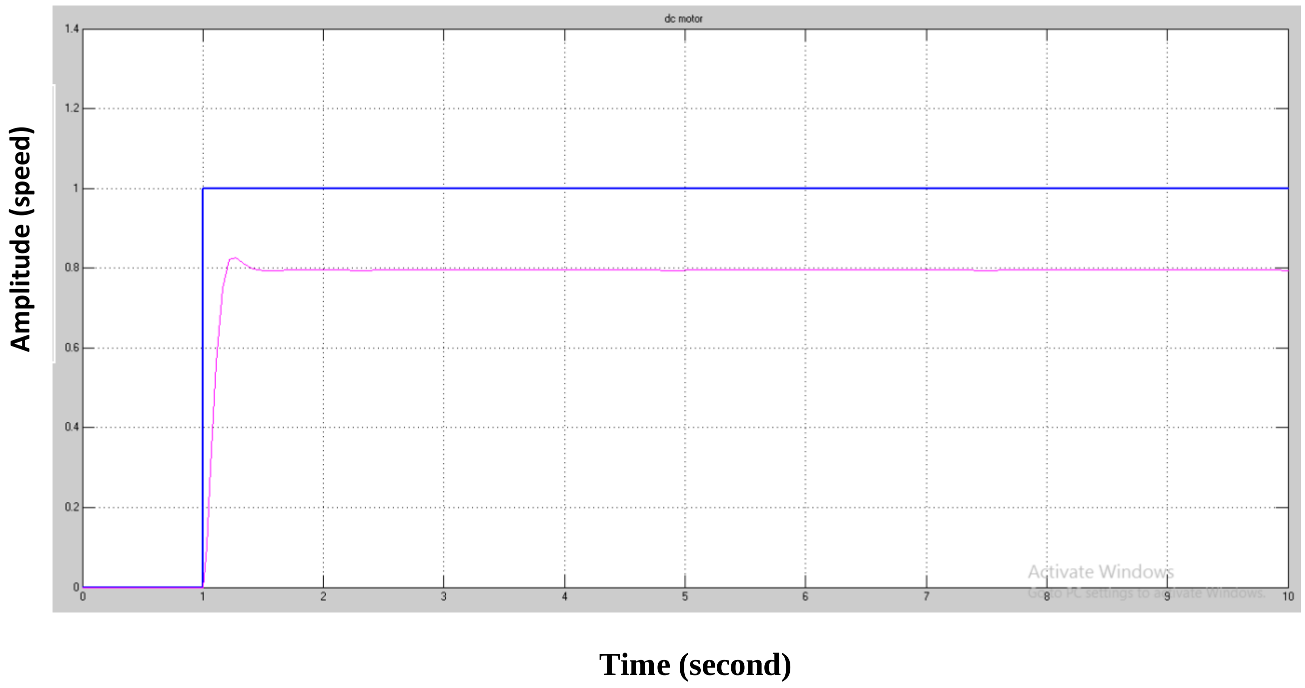


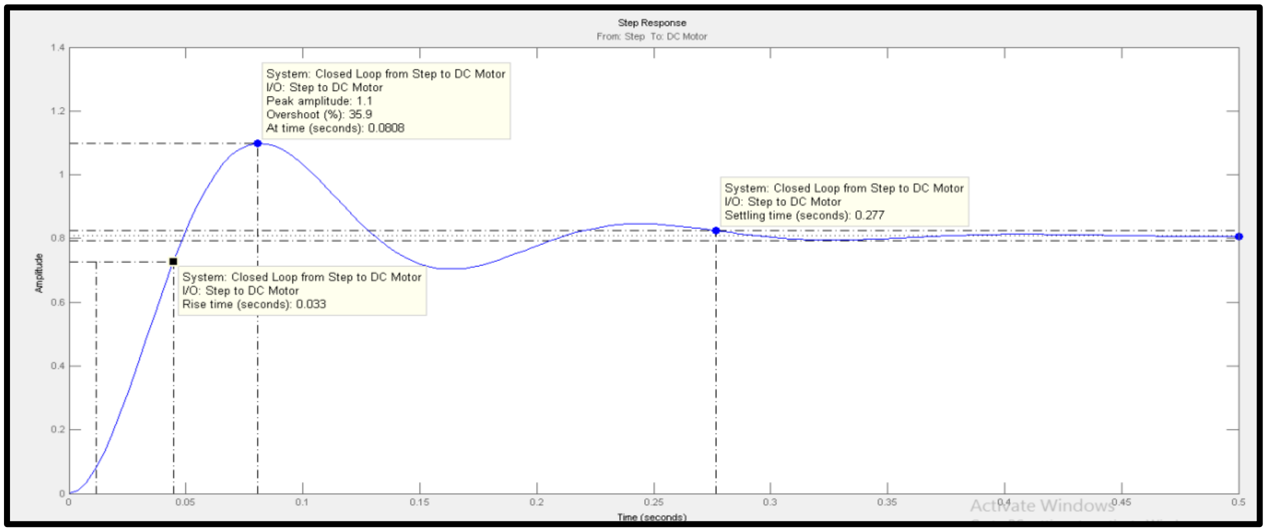
Figure (4.1): dc motor response in open loop and without any controller

The figure above show the speed response of DC motor without any controller system .the speed settled on value 0.8 and not reach the 1.

4.2 system with PID block to control :

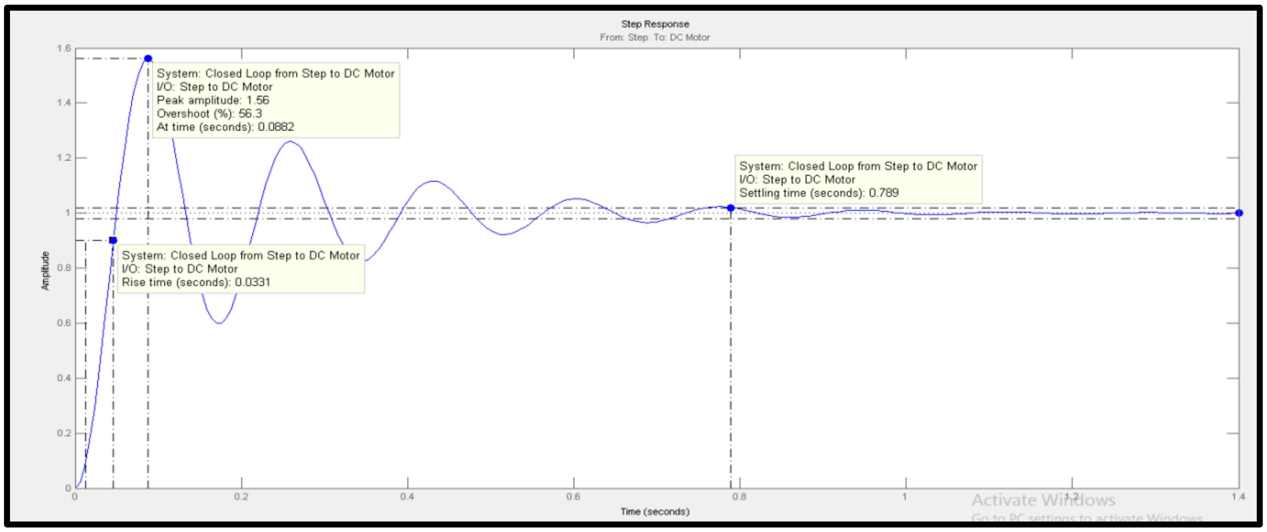
4.2.1 tuning P using Z-N:

In the figure below control PID controller by using Ziegle-Nichols method , selcted the P paramter only with value **5.3072** to show the response.



Figure(4.2):dc motor response in tuning P using Z-N

4.2.2 tunig PI using Z-N:



Figure(4.3): dc motor response in tuning PI using Z-N

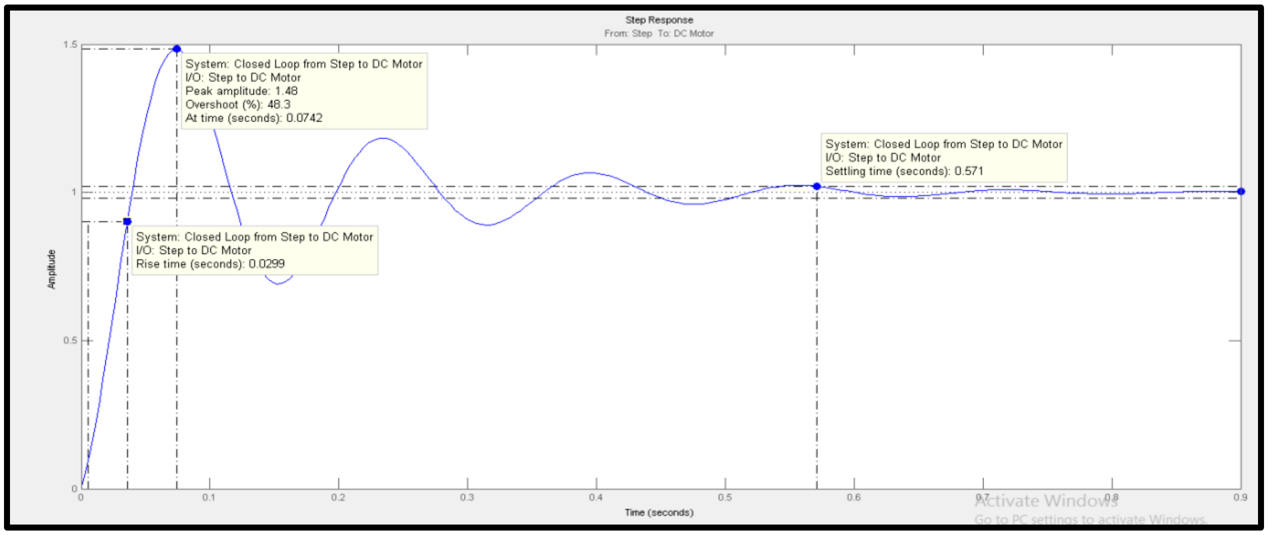
The second response test by using PI in the PID controller block with Z-N method to tune PID the value of PI are

p=4.8167 & I=85.252

4.2.3 tunig PID using Z-N:

finally shown in the figure below the response of PID using Z-N method by value :

p=6.479, I=170.5 and D=0.075



Figure(4.4): dc motor response in tuning PID using Z-N

Using PI rather PID because of :

- Easier to tune
- Easier to implement
- Save time
- D is diffecult to find
- D slow down performance
- D create slight oscillation

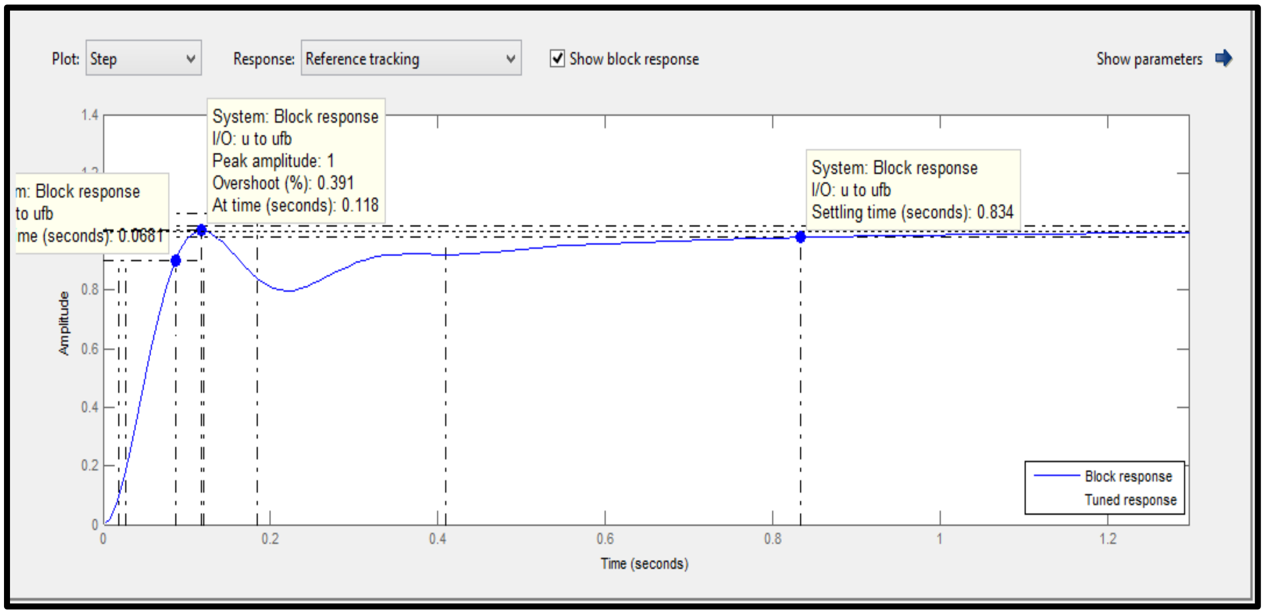
4.3 tuning PI using ACO :

ACO is random method to choose the best soluation .apending to the equation (3.21) have many paramter .change the paramter randomly to have the best soluation

Table 4.1: PID Values Obtained With ACO

Aco parameter / **Pid value**		KP	KI
α = 0.8	**β =0.2**	**2.59297**	**11.4854**
α = 0.1,	**β =0.2**	**0.879652**	**11.5374**
α = 0.1,	**β =0.02**	**1.72931**	**19.7521**
α = 0.2,	**β =0.02**	**0.2955**	**8.7899**

4.3.1 Dc motor rsponse when alpha = 0.8 , beta =0.2



System: Block response
I/O: u to ufb
Peak amplitude: 1
Overshoot (%): 0.391
At time (seconds): 0.118

m: Block response
to ufb
me (seconds): 0.0681

System: Block response
I/O: u to ufb
Settling time (seconds): 0.834

Figurer(4.5) : dc motor response when β=0.2 , α=0.8

4.3.2 Dc motor rsponse when alpha = 0.1 , beta =0.2

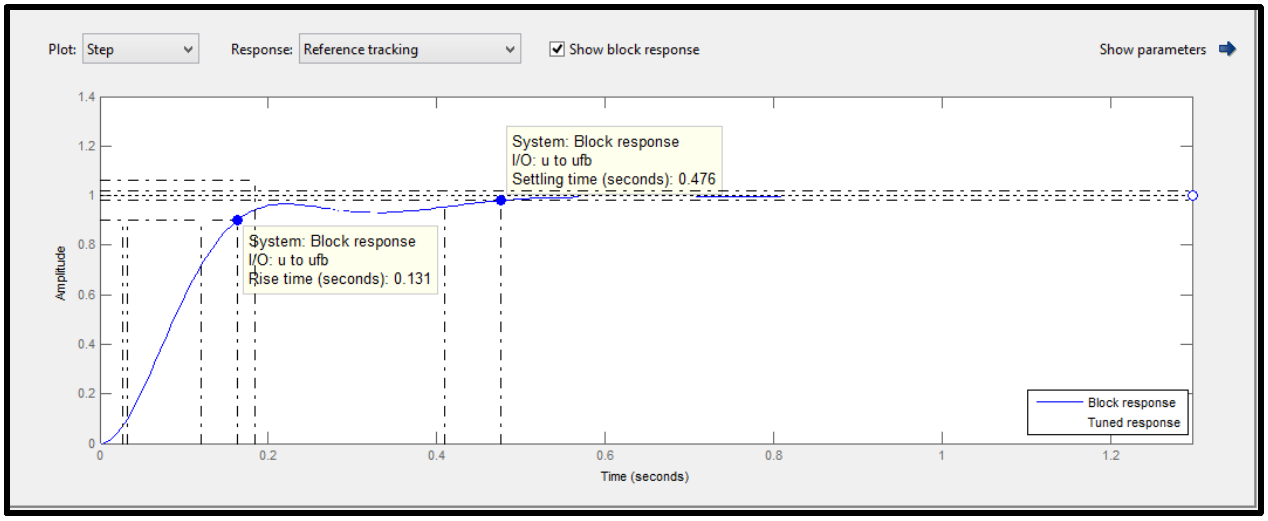
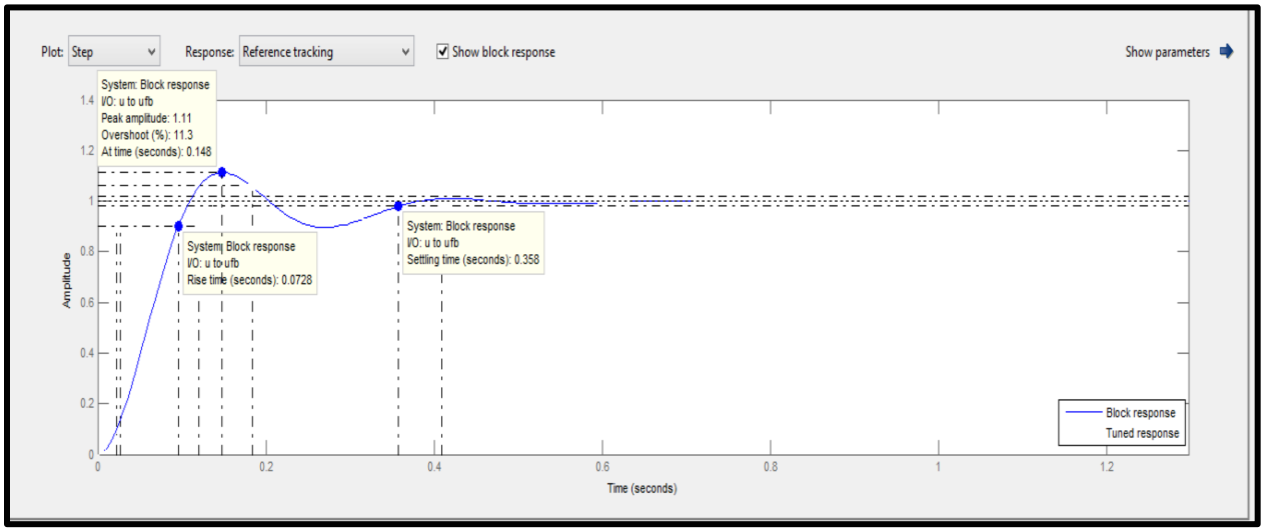


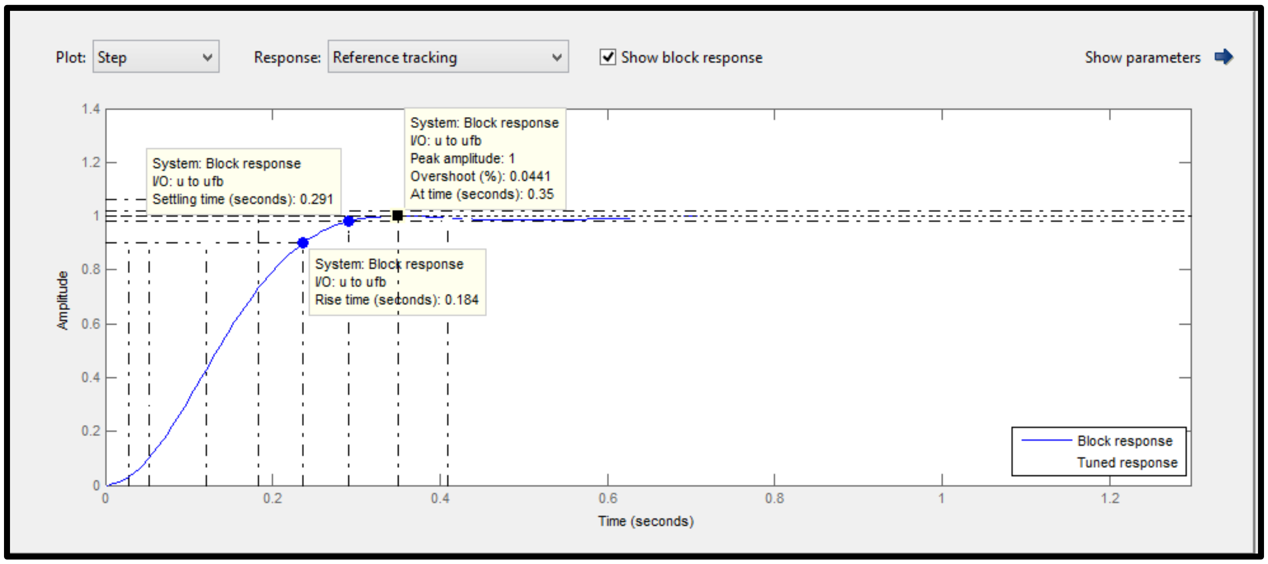
Figure (4.6) : DC motor response with β=0.2 , α=0.1

4.3.3 Dc motor rsponse when alpha = 0.1 , beta =0.02



Figurer(4.7) : Dc motor response with β=0.02 , α=0.1

4.3.5 dc motor rsponse when alpha = 0.2 , beta =0.02

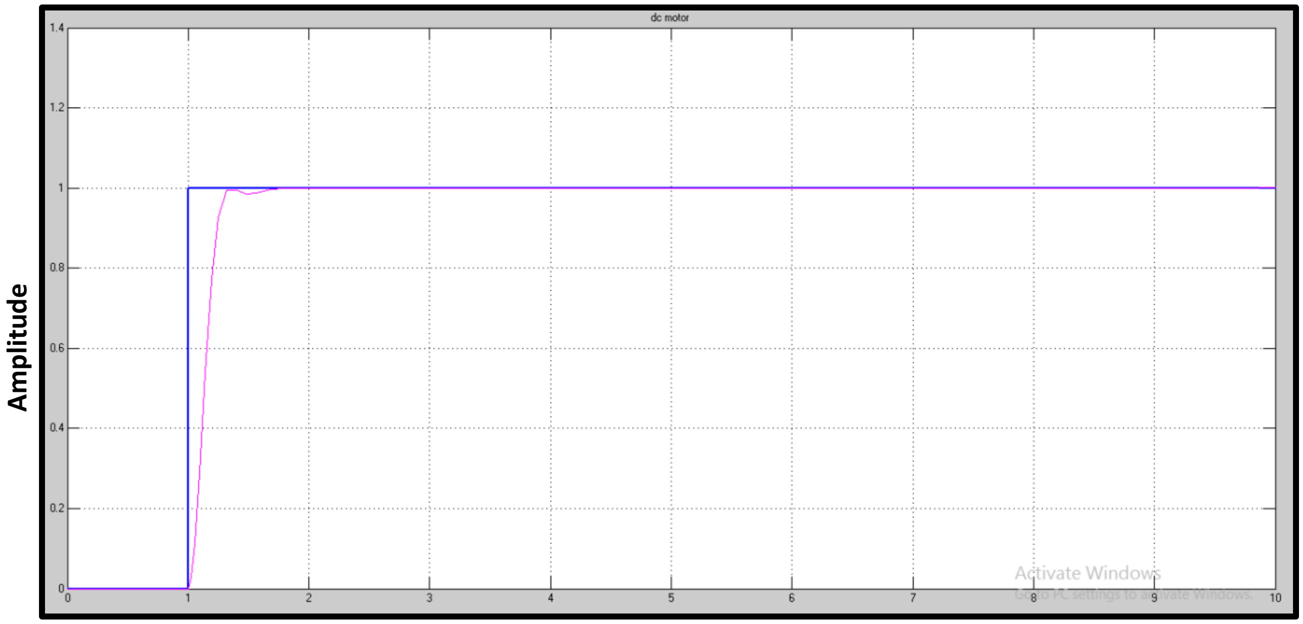


System: Block response
I/O: u to ufb
Settling time (seconds): 0.291

System: Block response
I/O: u to ufb
Peak amplitude: 1
Overshoot (%): 0.0441
At time (seconds): 0.35

System: Block response
I/O: u to ufb
Rise time (seconds): 0.184

(A)



Time (second)

(B)

Figurer(4.8) : (A)&(B) dc motor respose with β=0.02 , α=0.2

Table 4.2 : Comparison between different changed of the parameter of ACO

Aco parameter / characters		Rise time(sec)	Setteling time(sec)	Paek up at time	Overshoot %
α = 0.8	**β =0.2**	**0.0681**	**0.834**	**1 at 0.118 sec**	**0.39%**
α = 0.1,	**β =0.2**	**0.131**	**0.476**	**_**	**Have undershoot**
α = 0.1,	**β =0.02**	**0.072**	**0.358**	**1.11 at 0.148 sec**	**11.3%**
α = 0.2,	**β =0.02**	**0.184**	**0.291**	**1 at 0.35**	**0.004%**

Table 4.3 : Comparison System between Zigler-Nicols,fuzzy logic and ACO algorithm

Characters / method	Without Control	Z-N PI	Fuzzy control	ACO PI
Overshoot (%)	Undershoot 50%	56.3 %	4.5%	0 %
Rise time (sec)	0.2 sec	0.0331sec	0.15 sec	0.184
Settling time (sec)	0.5sec	0.789 sec	0.36 sec	0.291 sec

From the simulation result that is shown in the table above comparison between different responses of the Dc motor with different technical without any controller and self-tuning PID ,tuning PID with Z-N ,Fuzzy logic control and finally with Ant colony optimization method .the Z-N and Fuzzy logic are both have smaller rise time .ACO has better response curve, the smallest overshoot and minimum settling time.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

Chapter Five
5 .Conclusion And Recommendations

5.1 Conclusion

In this thesis, several methods have been tested to control Dc motor speed by using PID, many way to tune PID controller have been test.

Firstly shown the response of dc motor without any controller and compare between it and ZN tuning for PID .the proportional and integral (KP,KI) gain of the PID controller are adjusted according Ant colony optimization method.

The advantage of using ACO tuning PID is the computational efficiency, because it is very easy of the implementation and the computation processes is very fast, comparing with conventional methods. The ACO-PID technique gives better response than PID controller in terms of trajectory tracking.

Using PI controller only in these thesis because Easier to tune , Easier to implement, Save time, KD is diffecult to find ,KD slow down performance and KD create slight oscillation.

Finally have result from previous work that control Dc motor speed using FUZZY LOGIC control and have compare it with the ACO that have 4 try to find the perfect gain (KP,KI) to give the best response curve and parameter between ACO ,fuzzy logic and Z-N.

.

5.2 Recommendations

Recommendation for who wants to continuo in this side of the project because it's very important to execute in real time. These recommended looking after time so that to achieve the wonted results and the objective goals.
Suggested Future Work:

- This technique can be extended to another type of motor.
- Applying the hybrid between Ant colony optimization and particle swarm optimization to tune of PID controller and have optimal result.
- Adapting sophisticated control strategies such as neuro fuzzy control techniques

Reference

[1] V. S. Dr.R.Umarani, "Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques," *International Journal of Computer Applications (0975 – 8887),* vol. 5– No.4, August 2010, 2010.

[2] B. T. Praise, "SPEED CONTROL OF DC MOTOR USING ANT COLONY OPTIMIZATION ALGORITHM," *International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST),* 2016.

[3] F. H.-w. a. X. M.-F. FANG Yong-wang "Control of Brushless DC Motor with Ant Colony Optimization," *International Conference on Manufacturing Science and Engineering (ICMSE 2015),* 2015.

[4] S. G. Yaseen and N. M. A.AL-Slamy, "Ant Colony Optimization," *IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.6, June 2008,* 2008.

[5] R. B. s. J. van Ast, and B. De Schutter, "Ant colony optimization for optimal

control," *Delft Center for Systems and Control,* 2008.

[6] S. R. T. Samadhi Manasa1, M. Veda chary3, "POSITION CONTROL OF A DC MOTOR USING PID CONTROLLER," *International Journal of Scientific Engineering and Applied Science (IJSEAS) - Volume-1, Issue-3, June 2015,* 2015.

[7] N. A. Bhagat and M. Bhaganagare, "DC Motor Speed Control using PID Controllers," *"EE 616 Electronic System Design Course Project, EE Dept, IIT Bombay, November 2009",* 2009.

[8] website, "DC MOTOR," 2011-2017.

[9] M. S. a. A. Zomorrodi, "Comparison of PID Controller Tuning Methods."

[10] k. j. A. strom, "control design " p. 216, 2002.

[11] "tuning of pid controllers."

[12] V. R. ANU PIPPAL, VISHAL YADAV, BALENDRA AND SHEORAJ, "SPEED CONTOL OF DC MOTOR USING PID CONTROLLER," 2015.

Appendix

```
%% Ant colony optimization
clc
close all
clear all
%% ACO paramters
n_iter=20; %number of iteration
NA=20; % Number of Ants
alpha=0.2; % alpha
beta=0.02; % beta
roh=0.7; % Evaporation rate
n_param=3; % Number of paramters
LB=(0.01).*ones(1,27); % lower bound
UB=10.*ones(1,27); % upper bound
n_node=10000; % number of nodes for each param
cost_best_prev=inf;
%% Generating Nodes
T=ones(n_node,n_param).*eps; % Phormone Matrix
dT=zeros(n_node,n_param); % Change of Phormone
for i=1:n_param
    Nodes(:,i) =linspace(LB(i),UB(i),n_node); % Node generation at equal
spaced points
end
%% Iteration loop
for iter=1:n_iter

    for tour_i=1:n_param
        prob(:,tour_i)= (T(:,tour_i).^alpha) .* ((1./Nodes(:,tour_i)).^beta);
        prob(:,tour_i)=prob(:,tour_i)./sum(prob(:,tour_i));
    end

    for A=1:NA
        for tour_i=1:n_param
            node_sel=rand;
            node_ind=1;
            prob_sum=0;
            for j=1:n_node
                prob_sum=prob_sum+prob(j,tour_i);
                if prob_sum>=node_sel
                    node_ind=j;
                    break
                end
            end
            ant(A,tour_i)=node_ind;
        end
        cost(A)=cost_func(Nodes(ant(A,:)),0);
        clc
        disp(['Ant number: ' num2str(A)])
        disp(['Ant Cost: ' num2str(cost(A))])
        disp(['Ant Paramters: ' num2str(Nodes(ant(A,:)))])
        if iter~=1
        disp(['iteration: ' num2str(iter)])
        disp('_________________')
        disp(['Best cost: ' num2str(cost_best)])
        disp(['Best paramters:' num2str(Nodes(ant(cost_best_ind,:)))])
        end


    end
    [cost_best,cost_best_ind]=min(cost);

    % Elitsem
    if (cost_best>cost_best_prev) && (iter~=1)
        [cost_worst,cost_worst_ind]=max(cost);
        ant(cost_worst_ind,:)=best_prev_ant;
        cost_best=cost_best_prev;
        cost_best_ind=cost_worst_ind;
    else
        cost_best_prev=cost_best;
        best_prev_ant=ant(cost_best_ind,:)
    end
    dT=zeros(n_node,n_param); % Change of Phormone
    for tour_i=1:n_param
        for A=1:NA

dT(ant(A,tour_i),tour_i)=dT(ant(A,tour_i),tour_i)+cost_best/cost(A);
        end
    end

    T= roh.*T + dT;



end
```