بسم الله الرحمن الرحيم

Sudan University of Science and Technology

College of Graduate Studies

A thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Computer Engineering and Networking

Auto Tuning of PID Controller using Particle Swarm Optimization Method for DC Motor Speed Control

توليف تلقائي للمتحكم التناسبي التكاملي التفاضلي بإستخدام خوارزمية سرب الجزيئـــــــات الأمثل للتحـــكم في سرعة محـــــرك تيار مباشر

By:

Ahmed Humaida Ahmed Berair

Supervisor:

Dr. Alaa-eldein Awoda

Sep-2017

الإستـــــــــــــهلال

بسم الله الرحمن الرحيم

(إِنَّ فِي خَلْقِ السَّمَاوَاتِ وَالْأَرْضِ وَاخْتِلَافِ اللَّيْلِ وَالنَّهَارِ وَالْفُلْكِ الَّتِي تَجْرِي فِي الْبَحْرِ بِمَا يَنْفَعُ النَّاسَ وَمَا أَنْزَلَ اللَّهُ مِنَ السَّمَاءِ مِنْ مَاءٍ فَأَحْيَا بِهِ الْأَرْضَ بَعْدَ مَوْتِهَا وَبَثَّ فِيهَا مِنْ كُلِّ دَابَّةٍ وَتَصْرِيفِ الرِّيَاحِ وَالسَّحَابِ الْمُسَخَّرِ بَيْنَ السَّمَاءِ وَالْأَرْضِ لَآيَاتٍ لِقَوْمٍ يَعْقِلُونَ)

البقرة: 164]

Dedication

Throughout my life one person has always been there during those difficult and trying times. I would like to dedicate this thesis and everything I do to my mother. In addition to his I have always been surrounded with strong supportive women. Although our time together was brief, her contributions to my life will be felt forever.

Acknowledgments

Throughout my graduate education **Dr.Alaa Eldien Awoda** has been an excellent mentor and teacher. He truly understands what hard work and dedication can bring to one's life. I would like to thank **Dr. Abuaagla Babiker** for introducing me to the history of engineer. This thesis would not have happened without his support. I would also like to express gratitude toward **Dr. Salah Edam** who pushed toward improving my writing in class and offered many comments during various theses sessions. Without the support and motivation provided by **Dr.Ibtihal Haider** in graduate school would have been mundane. Her presence, patience, and emotional support, I will never forget.

I will always owe a great deal of gratitude toward these professors.

Abstract

The proportional-integral-derivative (PID) controllers are the most popular controllers used in industry because of their remarkable effectiveness, simplicity of implementation and broad applicability. However, tuning of these controllers is time consuming, not easy and generally lead to poor performance especially with non-linear systems. This research presents an artificial intelligence (AI) method of particle swarm optimization (PSO) algorithm for tuning the optimal (PID) controller parameters for DC motor speed control (DCMSC) system to achieve the mean objective which is the tracking between the reference speed and the output speed. This approach has superior features, including easy implementation, stable convergence characteristic and good computational efficiency over the conventional methods. The PID conventional controller had been applied and results were compared with the automatic tuning PSO-PID for (DCMSC) using Simulink of MATLAB. The DC Motor Scheduling PID-PSO controller is modeled in MATLAB environment. Simulation results for the proposed method give optimum input/output tracking and the error approximately equal zero without using the conventional solutions for DC motor speed control.

المستخلص

المتحكم التناسبي التكاملي التفاضلي الأكثر شيوعاً في الإستخدام الصناعي بسبب فعاليته الملحوظة. بساطة التنفيذ والتطبيق الواسع. ومع ذلك ، توليف هذا النوع من المتحكمات يعد غاية في الصعوبة ويستغرق وقت طويل ، كما تؤدي إلى ضعف في الآداء خصوصا مع الأنظمة اللاخطية. يقدم هذا البحث أسلوب من الذكاء الإصطناعي وهو خوارزمية سرب الجزيئات الأمثل للتوليف الأمثل لمعاملات المتحكم التناسبي التكاملي التفاضلي للسيطرة على سرعة محرك تيار مباشر لتحقيق الهدف الرئيسي وهو التتابع الأمثل بين سرعة الدخل وسرعة الخرج ، والطريقة المقترحة لها ميزات متفوقة تضمن سهولة التطبيق ، خصائص تقارب مستقرة والكفاءة الحسابية جيدة بخلاف الطرق التقليدية .تم تطبيق المتحكمات التناسبية التكاملية التفاضلية التقليدية وتمت مقارنة النتائج مع التوليف التلقائي لمعاملات المتحكم التناسبي التكاملي التفاضلي بإستخدام خوارزمية سرب الجزيئات الأمثل وبإستخدام برنامج المحاكاة في الماتلاب .نتائج المحاكاة للطريقة المقترحة أعطت التتابع الأمثل بين الدخل والخرج وسجلت قيمة خطأ تقترب إلى الصفر دون إستخدام الحلول التقليدية للتحكم في سرعة المحرك.

TABLE OF CONTENTS:

الإستـــــهلال 1

Dedication III

Acknowledgments IV

ABSTRACT V

المستخلص VI

CHAPTER ONE 1

1. INTRODUCTION 1

1.1 Background 1

1.2. Problem Statement: 3

1.3. Proposed Solution: 3

1.4. Objectives: 3

1.5. Methodology 3

1.6. Scope of the work: 4

1.7. Theses Organization: 4

CHAPTER TWO 5

2. LITRATURE REVIEW 5

2.1. Overview 5

2.2. Previous Studies 5

2.3 DC Motor Overview 6

2.4 CLASSICAL CONTROLLER DESIGN PROCEDURES 7

2.5 Loop Tuning: 14

2.6 Optimization 20

2.8 Particle Swarm Optimization Algorithm 22

CHAPTER THREE 26

3. SYSTEM DESIGN 26

3.1 The Parameters of the DC Motor: 26

3.3 PID Parameters: 33

3.4 Particle Swarm Optimization (PSO) Algorithm: **36**

CHAPTER FOUR 40

4. SIMULATION RESULTS 40

4.1 DC Motor with PID manual tuning 40

4.2. Discussion Results 42

4.3 Implementation of PSO-PID Controller 43

4.4 Robustness Investigation 45

CHAPTER FIVE 46

5. CONCLUTION AND RECOMMENDATIONS 46

5.1 Conclusions 46

5.2 Recommendations 47

6. References 48

7. APPENDIXES **50**

LIST OF FIGURES:

Figure 1: PID Controller for General System 10
Figure 2: Ziegler-Nichols P-I-D Controller Tuning Method 17
Figure 3: Cohen-Coon P-I-D Tuning Method 19
Figure 4: Block Diagram for DC motor TF from simulink 26
Figure 5: Schematic Representation of DC motor 26
Figure 6: DC motor model 28
Figure 7: P-D control on first and second order plants 30
Figure 8: of P-I control on first and second order plants 31
Figure 9: P-I-D control on first and second order plants 32
Figure 10: Classical Controller 34
Figure 11: Model of DC motor with PID Controller 35
Figure 12: System Block Diagrams from MATLAB/Simulink 35
Figure 13: Structure of the PID controller with PSO algorithm 37
Figure 14: System Flowcharts with PSO-PID Control System 38
Figure 15: PID Controllers Effects 40
Figure 16: Open-Loop Step Responses 42
Figure 17: Closed-Loop Step Responses 43
Figure 18: PID step response using Ziegler-Nichols Method 43
Figure 19: Step Responses with PSO-PID 44

LIST OF TABLES:

Table 1: Ziegler-Nichols P-I-D controller tuning method, adjusting Kp, Ki and K. 18
Table 2: Cohen-Coon P-I-D Tuning Method, adjusting Kp, Ki and Kd 19
Table 3: the Basic Variant of PSO 25
Table 4: Used symbols 27
Table 5: Affection Various O/P Parameters of P, Pi and Pid Controller 35
Table 6: Comparison of Gain Response Of P, Pi And Pid Controllers..... 36
Table 7: PID-Controller 40
Table 8: All Parameters-Controllers 41
Table 9: Comparison System between Zigler-Nicols and PSO algorith 44
Table 10: proposed model the comparison of performance index 45

LIST OF ABBREVIATIONS

PID	Proportional Integral Derivative
PSO	Particle Swarm Optimization
ZN	Ziegler Nichols
GA	Genetic Algorithm
FL	Fuzzy Logic
IAE	Integral Absolute of Error
ISE	Integral Square of Error
PWM	Pulse Width Modulation
PMDC	Permanent Magnet Direct Current
TF	Transfer function

Chapter One

1. INTRODUCTION

1.1 Background

PID controllers are widely used in industrial plants because it is simple and robust. Industrial processes are subjected to variation in parameters and parameter perturbations, which when significant makes the system unstable. So the control engineers are on look for automatic tuning procedures. From the control point of view, DC motor exhibit excellent control characteristics because of the decoupled nature of the field. Recently, many modern control methodologies such as nonlinear control, optimal control, variable structure control and adaptive control have been extensively proposed for DC motor. However, these approaches are either complex in theoretical bases or difficult to implement. PID control with its three-term functionality covering treatment to both transient and steady-states response, offers the simplest and yet most efficient solution too many real-world control problems. In spite of the simple structure and robustness of this method, optimally tuning gains of PID controllers have been quite difficult {Prasad, 2012}.
The particle Swarm Optimization (PSO) methods have been employed successfully to solve complex optimization problems. PSO first introduced by Kennedy and Eberhart is one of the modern heuristic algorithms; it has been motivated by the behavior of organisms, such as fish schooling and bird flocking. Generally, PSO is characterized as a simple concept, easy to implement, and computationally efficient. Unlike the other heuristic techniques, PSO has a flexible and well-balanced

mechanism to enhance the global and local exploration abilities {Li, 2017}.

Majority of the industries uses **DC** motor for their industrial used compare to **AC** motor based on few characteristics**.** The main reason using **DC** motor because **DC** motor can provide the speed control and stability. This is the main reason why majority of the industry machines will use **DC** motor to adjust their machine speed for their application like the conveyor belt so that it can improve the performance of their industry applications. Although **DC** motor is much stable than **AC** motor, they found that there have some unstable performance by DC motor in early state**.** The overshoot and undershoot will occur after started run the **DC** motor. This situation will decrease the accuracy and performance for the industry applications**.** Beside than overshoot problem, high rise time, settling time, and state-state error will also decrease the performance of the system. Therefore, the **PID** controller will be implemented to **DC** motor to solve those problems for improve the performance of the system. The PSO Algorithm will apply in **PID** controller for tuning the **PID** parameters to get a better values for proportional gain, Kp , integral gain,**Ki** and derivative gain, **Kd**.DC machines are characterized by their versatility. By means of various combinations of shunt-, series-, and separately-excited field windings they can be designed to display a wide variety of volt-ampere or speed-torque characteristics for both dynamic and steady- state operation. Because of the ease with which they can be controlled systems of DC machines have been frequently used in many applications requiring a wide range of motor speeds and a precise output motor control. In this research, the separated excitation DC motor model is chosen according to his good electrical and mechanical performances more than other DC motor models. The DC motor is driven by applied voltage.

1.2. Problem Statement:

The rotational speed of the DC motor changed at random according to different parameters, many real-time systems cannot tolerate overcome such a change. The accidential change of different parameters makes it difficult to tune or control the system.

1.3. Proposed Solution:

The proposed solution is by using PSO algorithm which is can automatically tune PID controller parameters during system run. Therefore, it can improved the speed behavior of the DC motor. Moreover, it can enhanced the characteristics of the engines, and makes the system robustness.

1.4. Objectives:

The main aim is to improve the DC motor speed behavior, the Objectives are:

1. To optimize PID controller behavior using intelligent tuning method PSO
2. To improve time response parameters for the DC motor speed response (overshot, settling time rise time, and steady state error).
3. To improve frequency response for the DC motor speed response.
4. Performance evaluation for the system by comparing proposed tuning method with traditional methods.

1.5. Methodology

PID controller was used to control the DC motor speed. Firstly; PID was tuned using traditional method. The second step was tuned using PSO algorithm. The system was tested under different condition

and the result was carried out with different scenarios. To simulate the proposed system using MATLAB/SIMULINK.

1.6. Scope of the work:

This research is meanly focus on PID controller. Different tuning method will be covered, optimization method will be also covered and PSO method will be highlighted. The system under study is DC motor.

1.7. Theses Organization:

- **Chapter One:** Introduction, which gives a brief background and stated the problem along with the proposed solution?

- **Chapter Two:** Literature Review, it gives a comprehensive study for the components used in the design.

- **Chapter Three:** System Design mainly discuses on the system design of the project. Details on the progress of the project were explained in this chapter.

- **Chapter Four:** Simulation and Discussion result, it was presented the results of the project .The discussion focused on the result obtained from simulation.

- **Chapter Five:** Conclusion and Recommendations' Concludes overall about the project, Obstacle faced and ***future recommendatio***n was also discussed in this chapter.

Chapter Two

2. LITRATURE REVIEW

2.1. Overview

In this chapter, a brief overview of some classical approaches in system identification and stabilizing controller design procedures found in control engineering literature are presented. This chapter also reviews the work carried out in the existing literature on optimization algorithms, optimization algorithm based system identification methods and optimization algorithm based controller design procedures{Bahgaat, 2016 }.

2.2. Previous Studies

According to Jalilvand, A. Kimiyaghalam, A. Ashouri, H. Kord (2011), the design of block diagram for DC motor in the Simulink software will based on the transfer function , The purpose of implement PID controller into the system is to improve the dynamic response and reduce the steady-state error.There is a lot of other journal that research on the topic of PID controller and the comparison with other controller to observe the performance of the DC motor.MeghaJaiswal and MohnaPhadnis(2013) designed the block diagram for the DC motor with and without the genetic algorithm based PID controller. The purpose for this research is to do a comparison of output response of DC motor between the system with genetic algorithm and system without the genetic algorithm.RituSoni, DBV Singh, PramodPandey and Priyanka Sharma (2013) research about the comparison between the fuzzy logic controller and PID controller for speed control of DC motor. DC motor

is the best choice for speed and position control in industrial applications.

According to the AdityaPratap Singh, Udit Narayan and AkashVerma (2013) , their research is designed the PID controller to control the speed response ofthe DC motor. Different value of PID parameters which is proportional gain, Kpintegral gain, Ki and derivative gain, Kd will provide different effect to the performance of the DC motor.Research of Philip A. Adewuyi(2013) is to make the comparison between theperformance of Fuzzy logic controller and PID controller by using the block diagramin Simulink.

G. SUDHA and DR.R. ANITA (2012) developed the design to make thecomparison between the output response of the Fuzzy logic controller and the PIDcontroller. The purpose of this research is to check which controllers can providebetter performance of the DC motor.

2.3 DC Motor Overview

DC motors are used extensively in adjustable-speed drives and position control applications. Their speeds below the base speed can be controlled by armature-voltage control. Speeds above the base speed are obtained by field-flux control. As speed control methods for DC motors are simpler and less expensive than those for the AC motors, DC motors are preferred where wide speed range control is required. Phase controlled converters provide an adjustable DC voltage from a fixed AC input voltage. DC choppers also provide DC output voltage from a fixed DC input voltage. The use of phase controlled rectifiers and DC choppers for the speed control of DC motors have revolutionized the modern industrial controlled applications. DC drives are classified as follows:

a) Single phase DC drives

b) Three phase DC drives

c) Chopper drives

2.3.1 Application of DC shunt motor

For a given field current in a shunt motor, the speed drop from No-load to full load is invariably less than 6% to 8%. In view of this, the shunt motor is termed as a constant speed motor. Therefore, for constant speed drives in industry DC shunt motors are employed{Roy, 2015}.

2. When constant speed service at low speeds is required, DC shunt motors are preferred over synchronous motors.

3. When the driven load requires a wide range of speed control, both below and above the base speed, a DC shunt motor is employed. Eg: Lathes.

4. DC shunt motor can be used as a separately excited motor, if the field winding is disconnected from armature and connected to a external voltage source.

2.4 CLASSICAL CONTROLLER DESIGN PROCEDURES

Although many modern controlling methods have been proposed for industrial process control applications, classical and modified structured PID controllers are still widely implemented in the industries. There are lot of works available in literature on the stabilizing controller design for a class of unstable systems. Adequate number of tuning procedures is existing to design a classical PI / PID controllers for unstable systems (Åström and Hägglund 1995; Huang and Chen 1999; Sree et al 2004; Jung et al 1999, Lee et al 2000; Padmasree and

Chidambaram 2006). During the reference tracking problem, classical PI / PID controller shows excessive overshoot due to the effects such as proportional and derivative kick (Johnson and Moradi 2005). This can be minimized by considering the modified structured PID controller {Prasad, 2012}.

Padhy and Majhi (2006) have proposed a relay based controller design for unstable FOPTD model. Here, the process model parameters are considered to design a two degree of freedom controller structure (PI-PD) based on the desired loop phase margin and gain margin criteria. The proposed method effectively works for the unstable FOPTD model with θ/τratio up to 0.8. Liu et al (2005) developed a novel two degree of freedom control scheme for unstable systems. A detailed analytical expression is discussed for setpoint tracking controller part and disturbance estimator part separately. The 2DOF controller is implemented and validated with unstable FOPTD and SOPTD models and offers better result for reference tracking and disturbance rejection.

Yang et al (2002) have discussed an Internal Model Controller (IMC) based single loop controller design procedure. The procedure helps to accomplish a basic PID controller or a higher order controller structure for unstable processes with time delay. Tan et al (2003) have proposed a modified IMC for FOPTD and SOPTD unstable systems. In this method, the necessary analytical expressions are developed to tune modified IMC parameters (K_0, K_1, and K_2) using process parameters such as gain (K), time constant (τ) and delay (θ). The overall performance of the modified IMC is better compared to other IMC based methods existing in the literature {, 6}.

Panda (2009) has elaborately discussed about the synthesis method based development of IMC equivalent PID tuning rules for low order time delayed unstable systems. This method provides simple analytical expressions to find the controller parameters such as proportional gain (K_p), integral time constant (T_i), and derivative time constant (T_d). The proposed method is validated on a class of stable and unstable process models, Kharitonov polynomial analysis also carried to ensure the robustness of the IMC-PID controller. Shamsuzzoha and Lee (2007) designed an IMC-PID to improve the disturbance rejection response. The efficacy of the controller is validated using a class of stable and unstable system with time delay. The method also shows a robust performance for the system with perturbed model parameters. Araki and Taguchi (2003) presented a detailed study on various two degrees of freedom controllers such as feed forward structure, feedback structure, setpoint filter type structure, filter and preceded-derivative structure and setpoint weighted PID structure{Bahgaat, 2016}.

Setpoint weighted PID controller is widely addressed by many of the researchers, since it provides better reference tracking and disturbance rejection responses for stable and unstable systems (Chidambaram 2000; Prashanti and Chidambaram 2000; Padmasree and Chidambaram 2005; Dey et al 2006). Chen et al (2008) have discussed setpoint weighted PID controller design for a class of unstable systems; and in this method, the existing PID controller is reformed into a setpoint weighted PID control system. The design procedure does not require process information and the information on the design methods of its original error feedback PID control system. Their work also reports that, a PID-PD controller structure can be used to achieve basic and a variety of modified PID structures by assigning appropriate values for set-point weighting

parameters such as and . The proposed method is validated on unstable FOPTD, SOPTD, and TOPTD systems and better results are obtained. Vijayan and Panda (2012) have presented a comprehensive procedure to implement a double feedback loop and a set-point filter for stable and unstable processes. Anusha and Rao (2012) have examined the design of PID controller for an unstable SOPTD system based on IMC and H_2 minimization to support the desired closed loop performance. Maclaurin series is considered to approximate the controller expression as a PID controller and the bounds for the tuning parameter are analyzed using the maximum sensitivity criterion.

In this research work, the controller structures such as classical PID, setpoint weighted PID and PID with prefilter are considered to stabilize the unstable processes.

2.4.1 Theory of PID controllers

A proportional–integral–derivative controller (PID controller) is a generic control loop feedback mechanism (controller) widely used in industrial control systems – a PID is the most commonly used feedback controller. A PID controller calculates an "error" value as the difference between a measured process variable and a desired set point. The controller attempts to minimize the error by adjusting the process control inputs.

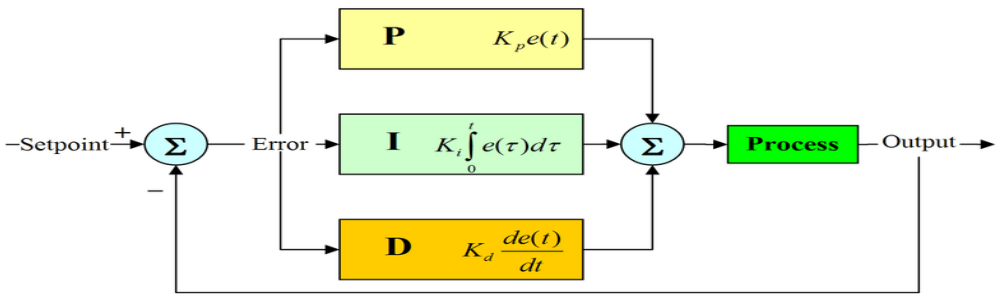

Figure 1 PID Controller for General System

The PID controller calculation (algorithm) involves three separate constant parameters, and is accordingly sometimes called three-term control: the proportional, the integral and derivative values, denoted P, I, and D. Heuristically, these values can be interpreted in terms of time: P depends on the present error, I on the accumulation of past errors, and D is a prediction of future errors, based on current rate of change. The weighted sum of these three actions is used to adjust the process via a control element such as the position of a control valve or the power supply of a heating element {Li, 2017}.

In the absence of knowledge of the underlying process, a PID controller is the best controller. By tuning the three parameters in the PID controller algorithm, the controller can provide control action designed for specific process requirements. The response of the controller can be described in terms of the responsiveness of the controller to an error, the degree to which the controller overshoots the setpoint and the degree of system oscillation. Note that the use of the PID algorithm for control does not guarantee optimal control of the system or system stability.

2.4.2 PID tuning

Tuning a control loop is the adjustment of its control parameters (gain/proportional band, integral gain/reset, derivative gain/rate) to the optimum values for the desired control response. Stability (bounded oscillation) is a basic requirement, but beyond that, different systems have different behavior, different applications have different requirements, and requirements may conflict with one another.

PID tuning is a difficult problem, even though there are only three parameters and in principle is simple to describe, because it must satisfy complex criteria within the limitations of PID control. There are

accordingly various methods for loop tuning, and more sophisticated techniques are the subject of patents {Bahgaat, 2016}.

2.4.3 Aim of the Recitation:

Aim of the recitation was to introduce the concept of Discrete Time P-I-D controllers and how they can be implemented on real life projects.

It was first intended to explain the usage of continuous time P-I-D controllers. In the first part of the recitation, it was aimed to show the how P, P-I, P-I-D controllers change the steady state response of the closed loop systems. Moreover, the methods to tune P-I-D controllers were introduced. It was meant to show that how hard it could get to properly tune a P-I-D controller. Secondly, it was intended to show how P, P-D, P-I, and P-I-D controllers affect the transient response of the closed loop system. It was meant to show how one can gain a feature but lose the other. Thirdly, it was intended to show how one should estimate the dynamics of the continuous time plant and use proper sampling time for discrete time P-I-D controller. It was also meant to show how changing transformation method may cause different pole locations on the z-plane. Lastly, it was intended to show how one could control the velocity and the position of the vehicle of the 'Gate' project by implementing a discrete time P-I-D controller in that project.

2.4.4 P Controller:

P controller is mostly used in first order processes with single energy storage to stabilize the unstable process. The main usage of the P controller is to decrease the steady state error of the system. As 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. As we increase the proportional gain, it provides smaller amplitude and phase margin, faster dynamics satisfying wider frequency band and 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 decreases the rise time and after a certain value of reduction on the steady state error, increasing K only leads to overshoot of the system response. 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 {Roy, 2015}.

2.4.5 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.

2.4.6 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.

2.4.7 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.

In order to observe the basic impacts, described above, of the proportional, integrative and derivative gain to the system response, see the simulations below prepared on MATLAB in continuous time with a transfer function $\frac{1}{s^2+10s+20}$ and unit step input. The results will lead to tuning methods

2.5 Loop Tuning:

Tuning a control loop is arranging the control parameters to their optimum values in order to obtain desired control response. At this point, stability is the main necessity, but beyond that, different systems lead to different behaviors and requirements and these might not be compatible with each other. In principle, P-I-D tuning seems completely easy, consisting of only 3 parameters, however, in practice; it is a difficult problem because the complex criteria at the P-I-D limit should be satisfied. P-I-D tuning is mostly a heuristic concept but existence of many objectives to be met such as short transient, high stability makes

this process harder. For example, sometimes, systems might have nonlinearity problem which means that while the parameters work properly for full load conditions, they might not work as effective for no load conditions. Also, if the P-I-D parameters are chosen wrong, control process input might be unstable, with or without oscillation; output diverges until it reaches to saturation or mechanical breakage{th International Power, 2011}.

For a system to operate properly, the output should be stable, and the process should not oscillate in any condition of set point or disturbance. However, for some cases bounded oscillation condition as a marginal stability can be accepted.

As an optimum behavior, a process should satisfy the regulation and command breaking requirements. These two properties define how accurately a controlled variable reaches the desired values. The most important characteristics for command breaking are rise time and settling time. For some systems where overshoot is not acceptable, to achieve the optimum behavior requires eliminating the overshoot completely and minimizing the dissipated power in order to reach a new set point.

In today's control engineering world, P-I-D is used over %95 of the control loops. Actually, if there is control, there is P-I-D, in analog or digital forms. In order to achieve optimum solutions K_p, K_i and K_d gains are arranged according to the system characteristics. There are many tuning methods, but most common methods are as follows:

- **Manual Tuning Method**
- **Ziegler-Nichols Tuning Method**
- **Cohen-Coon Tuning Method**
- **PID Tuning Software Methods (ex. MATLAB**)

2.5.1 Manual Tuning Method:

Manual tuning is achieved by arranging the parameters according to the system response. Until the desired system response is obtained K_i, K_p and K_d are changed by observing system behavior.

Example (for no system oscillation): First lower the derivative and integral value to 0 and raise the proportional value 100. Then increase the integral value to 100 and slowly lower the integral value and observe the system's response. Since the system will be maintained around set point, change set point and verify if system corrects in an acceptable amount of time. If not acceptable or for a quick response, continue lowering the integral value. If the system begins to oscillate again, record the integral value and raise value to 100. After raising the integral value to 100, return to the proportional value and raise this value until oscillation ceases. Finally, lower the proportional value back to 100.0 and then lower the integral value slowly to a value that is 10% to 20% higher than the recorded value when oscillation started (recorded value times 1.1 or 1.2).

Although manual tuning method seems simple it requires a lot of time and experience

2.5.2 Ziegler–Nichols method

This method was developed by John G. Ziegler and Nathaniel B. Nichols in the 1940's. Here we will discuss the Z-M 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. We will denote it by T_u.
4. Use the following values as the initial tuning constants

2.5.2.1 PID controller with Ziegler-Nichols Method:

More than six decades ago, P-I controllers were more widely used than P-I-D controllers. Despite the fact that P-I-D controller is faster and has no oscillation, it tends to be unstable in the condition of even small changes in the input set point or any disturbances to the process than P-I controllers. Ziegler-Nichols Method is one of the most effective methods that increase the usage of P-I-D controllers.

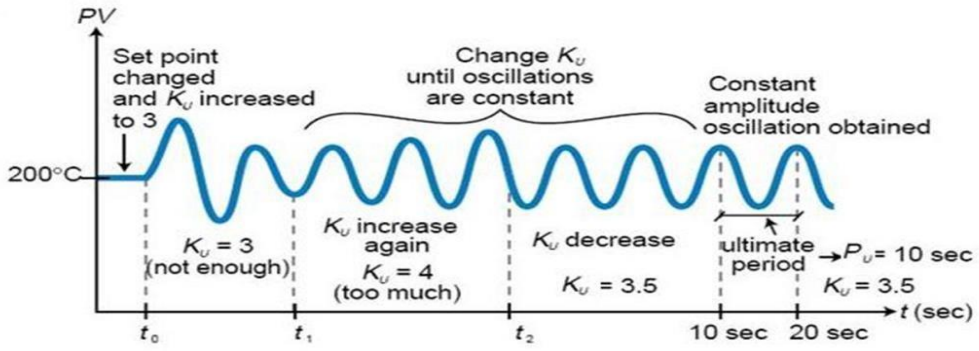

Figure 2 Ziegler-Nichols P-I-D Controller Tuning Method

The logic comes from the neutral heuristic principle. Firstly, it is checked that whether the desired proportional control gain is positive or negative. For this, step input is manually increased a little, if the steady state output increases as well it is positive, otherwise; it is negative. Then, K_i and K_d are set to zero and only K_p value is increased until it creates a periodic oscillation at the output response. This critical K_p value is attained to be "ultimate gain", K_c and the period where the oscillation occurs is named as P_c "ultimate period". As a result, the

whole process depends on two variables and the other control parameters are calculated according to the table in the Figure 9.

Table 1.1: Ziegler-Nichols P-I-D controller tuning method, adjusting Kp, Ki and Kd

Ziegler–Nichols method giving *K'* values (loop times considered to be constant and equal to dT)			
Control Type	Kp	Ki'	Kd'
P	$0.50K_c$	0	0
PI	$0.45K_c$	$1.2K_p dT / P_c$	0
PID	$0.60K_c$	$2K_p dT / P_c$	$K_p P_c / (8dT)$

Advantages:

- ✓ **It is an easy experiment; only need to change the P controller**
- ✓ **Includes dynamics of whole process, which gives a more accurate picture of how the system is behaving**

Disadvantages:

- ➢ **Experiment can be time consuming**
- ➢ **It can venture into unstable regions while testing the P controller, which could cause the system to become out of control**
- ➢ **For some cases, it might result in aggressive gain and overshoot**

2.5.3 Cohen-Coon Tuning Method:

This tuning method has been discovered almost after a decade than the Ziegler-Nichols method. Cohen-Coon tuning requires three parameters which are obtained from the reaction curve as in the Figure 10.3.

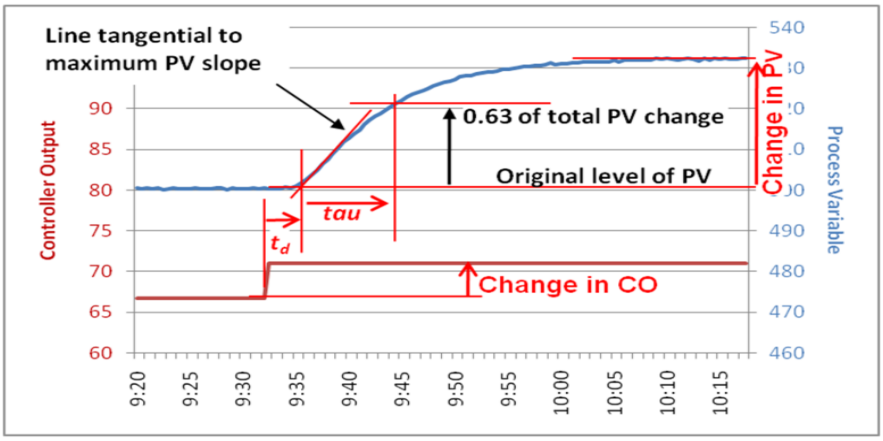

Figure 3 Cohen-Coon P-I-D Tuning Method

The controller is manually placed and after the process settled out a few percent of the change is made in the controller output (CO) and waited for the process variable (PV) to settle out at a new value.

After converting the time variables into the same units and applying couple of tests until to find similar result, these three variables are used to define new control parameters using the table in the Figure 11 below.

Table 2 Cohen-Coon P-I-D Tuning Method, adjusting Kp, Ki and Kd

	Controller Gain	Integral Time	Derivative Time
P Controller:	$K_C = \frac{1.03}{g_p}\left(\frac{\tau}{t_d} + 0.34\right)$		
PI Controller:	$K_C = \frac{0.9}{g_p}\left(\frac{\tau}{t_d} + 0.092\right)$	$T_I = 3.33 t_d \frac{\tau + 0.092 t_d}{\tau + 2.22 t_d}$	
PD Controller:	$K_C = \frac{1.24}{g_p}\left(\frac{\tau}{t_d} + 0.129\right)$		$T_D = 0.27 t_d \frac{\tau - 0.324 t_d}{\tau + 0.129 t_d}$
PID Controller: (Noninteracting)	$K_C = \frac{1.35}{g_p}\left(\frac{\tau}{t_d} + 0.185\right)$	$T_I = 2.5 t_d \frac{\tau + 0.185 t_d}{\tau + 0.611 t_d}$	$T_D = 0.37 t_d \frac{\tau}{\tau + 0.185 t_d}$

2.5.4 Comparison of the two methods:

If we want to compare these two methods, Ziegler-Nichols can be used for any order of the systems, especially for the higher ones, while Cohen-Coon can only be used for first order systems. Therefore, Ziegler-Nichols tuning method is more widely used. However, for the first order

systems Cohen-Coon is more flexible since as Ziegler-Nichols is only applicable when the dead time is less than $\frac{1}{2}$ of the time constant, Cohen-Coon is tolerable until $\frac{3}{4}$ of this value and it can be even extended. Therefore, for systems having time delay this tuning method is more convenient. All in all, despite the fact that tuning a system seems easy to apply, in practice, it is really hard to analyze and pick a tuning method satisfying all system requirements. Using the logic of arranging the control parameters described above, some PID tuning software methods are developed which are easier to apply and saves time to get an optimum solution.

2.6 Optimization

There is a continuous need to increase production efficiency and maximize economic benefits for any given process plant. Typically, hundreds of PID loops are employed at the basic regulatory level for controlling the process. In the case where advanced control strategies are employed to derive additional economic benefits, these applications are dependent on an optimum base level of regulatory controls. Good regulatory control system performance is essential in order to maintain safe operation, maintain product quality and minimize operating cost. An important objective of evaluating the PID loop control performance is to define the methods and implementation strategy that will reduce a large amount of process information to a few easily interpreted metrics numbers that provide a clear picture of overall control loop performance. To maintain desired PID loop operation, a sustained control performance software offering should include features for historical data collection, controller performance metrics, process modeling and loop tuning/simulation capability, and utilize an efficient user-interface for

display and reporting functions. Integration of these functions provides an opportunity to validate PID control performance, quickly identify control loop deficiencies, prioritize efforts in resolution of control issues, and distribute some of the efforts associated with this activity on a continuous basis for sustaining optimum regulatory control. Although economic benefits derived from implementing a performance monitoring strategy may be difficult to quantify on a per-loop basis, often a single control issue can manifest into contributing to poor overall unit or process performance. The ability to provide sustained regulatory control performance is easily justified through improvements in online time, resource reductions, improved safety, and control effectiveness or quality of process financial performance {Bahgaat, 2016}.

2.7 PID Tuning Software Methods for DCM Using GA

MeghaJaiswal and MohnaPhadnis (2013) designed the block diagram for the DC motor with and without the genetic algorithm based PID controller. The purpose for this research is to do a comparison of output response of DC motor between the system with genetic algorithm and system without the genetic algorithm. The transfer function for the DC motor is:

$$\frac{0.5}{0.0077s^2+0.09007s+0.25018}$$

That obtained from the model of the DC motor. The block diagram of the DC motor system will be constructed based on this transfer function. The output response for the system without using the genetic algorithm has these problems such as high overshoot, rise time, settling time and steady-state error. The performance of DC motor will become not stable and accurate due to these problems.

After that, PID controller block diagram is inserted into system and then simulate the circuit to observe the output response. The output response with and without the genetic algorithm will be compared. Hence, the output response for the system with the genetic algorithm shows that it will eliminate the maximum overshoot problem. Besides that, it also reduces the rise time and settling time for the whole system. The steady-state error for the system with and without the genetic algorithm is almost zero percentage. Therefore, system with the genetic algorithm is much better performance for the system without the genetic algorithm due to overall characteristics {Li, 2017}.

2.8 Particle Swarm Optimization Algorithm

The prior works of Particle Swarm Optimization (PSO) mostly applied to a wide range of engineering optimization problems, including path finding, scheduling, object recognition, face detection, and other application areas. PSO also provides a new way for industrial process identification and controller design.

Jain and Nigam (2008) proposed a PD-PI controller design for a highly nonlinear inverted pendulum system using PSO algorithm. The effectiveness of the method is validated through a comparative study with GA. Zamani et al (2009) discussed about PSO based H PID controller design for Single Input Single Output (SISO) and Multi Input Multi Output (MIMO) process models. A novel weighted sum of multiple objective function is developed using the frequency domain specifications, time domain specifications and the error. The superiority of the proposed method is validated with GA and simulated annealing algorithms. Zamani et al (2009a) designed a fractional order PID controller for an Automatic Voltage Regulator (AVR) system using

PSO, and better robustness is achieved for the system with model uncertainties. Chang and Shih (2010) developed an improved PSO algorithm to design an optimal PID controller for reference tracking problem of a nonlinear inverted pendulum system. In this algorithm, a third learning parameter C_3 is introduced into the original velocity updating formula inorder to enhance the optimization search ability of basic PSO which results in improved convergence compared to existing PSO{Prasad, 2012}.

Kanthaswamy and Jovitha (2011) proposed a novel procedure, simplex derivative pattern search and implicit filtering based hybrid PSO algorithm. With simulation study, it is conformed that, proposed method provides improved convergence compared to original PSO. The method is tested and validated on a class of stable and unstable systems. Pillay and Govender (2011) proposed PSO based setpoint weighted PID controller tuning for a class of unstable FOPTD systems. Minimization of Integral Time Absolute Error (ITAE) is prioritised as the performance index, and the method provides better result compared to existing classical tuning procedure in the case of set-point tracking and disturbance rejection operations.

Modares et al (2010; 2010a) proposed an adaptive PSO algorithm to estimate the model parameters for a class of nonlinear systems in both offline and online methods. The accuracy and search speed of the proposed adaptive PSO is confirmed with linearly decreasing inertia weight PSO, dynamic inertia weight PSO, nonlinear inertia weight PSO, and GA. Alfi and Modares (2011) discussed an adaptive PSO based system identification and control procedure for stable and discrete nonlinear systems. In system identification procedure, the structure of a system is assumed to be known previously, and the algorithm is allowed

to search the system parameters in D dimensional search space. The identified model is then considered to design an optimal PID controller. The method achieves faster convergence speed and better solution accuracy with minimum incremental computational burden compared to PSO algorithm with linearly decreasing inertia weight and GA. Alfi (2011) discussed an adaptive PSO algorithm to estimate the parameters of a class of nonlinear systems. Initially, search ability of the proposed algorithm is tested with benchmark functions such as Griewank, Rosenbrock and Rastrigrin function. Later, the identification performance of adaptive PSO is compared with a nonlinearly decreasing weight PSO and a real-coded GA, in terms of parameter accuracy and convergence speed. The weighted sum of error function is chosen as the objective function to identify the global optimal values. Alfi (2012) implemented PSO algorithm in identification of parameters of Lorenz chaotic system. A dynamic inertia weight is assigned for the PSO algorithm, to cope with the online system parameter identification problem. Inorder to increase the search efficiency and convergence rate, the inertia weight for every particle is dynamically updated based on the feedback taken from the fitness of the best previous position found by the particle. The performance of the discussed method is validated with real coded genetic algorithm {, 6}.

2.8.1 Objective function

The objective function for the optimization algorithm is chosen to maximize the domain constrains or to minimize the preference constrains. For system identification and controller design problem, there exists a variety of objective functions which should be minimized during the optimization. The functions such as single objective function, multiobjective function (Pareto optimality), and weighted sum of

multiple objective functions are very popular in heuristic algorithm based optimization procedures particularly in controller design (Chiha et al 2012) In this research work, an attempt has been made with single objective function (minimization of Sum of Squared Error) for system identification practice. The PID and modified structured PID controller design procedure, single and the weighted sum of multiple objective functions are adopted.

Table 3 the Basic Variant of PSO

Basic Variant	Function	Advantages	Disadvantages
Velocity Clamping	Control the global exploration of the particle Reduces the size of the step velocity, so that the particles remain in the search area, but it cannot change the search direction of the particle	VC reduces the size of the step velocity so it will control the movement of the particle	If all the velocity becomes equal to v_{max} the particle will continue to conduct searches within a hypercube and will probably remain in the optima but will not converge in the local area.
Inertia Weight	Controls the momentum of the particle by weighing the contribution of the previous velocity,	A larger inertia weight in the end of search will foster the convergence ability.	Achieve optimality convergence strongly influenced by the inertia weight
Constriction Coefficient	To ensure the stable convergence of the PSO algorithm [21]	Similar with inertia weight	when the algorithm converges, the fixed values of the parameters might cause the unnecessary fluctuation of particles
Synchronous and Asynchronous Updates	Optimization in parallel processing	Improved convergence rate	Higher throughput: More sophisticated finite element formulations Higher accuracy (mesh densities)

Chapter Three

3. SYSTEM DESIGN

3.1 The Parameters of the DC Motor:

The parameters of the DC motors may change according to different torque and rpm values of the DC motors

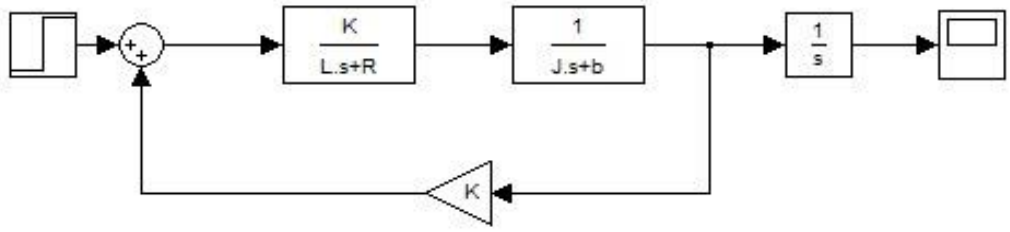

Figure 4 Block Diagram for DC motor TF from simulink

3.1.1 DC Motor System

According to Jalilvand, A. Kimiyaghalam, A. Ashouri, H. Kord (2011), the design of block diagram for DC motor in the Simulink software will based on the transfer function that obtain from the Figure 2.1.

The Schematic of the DC Motor:

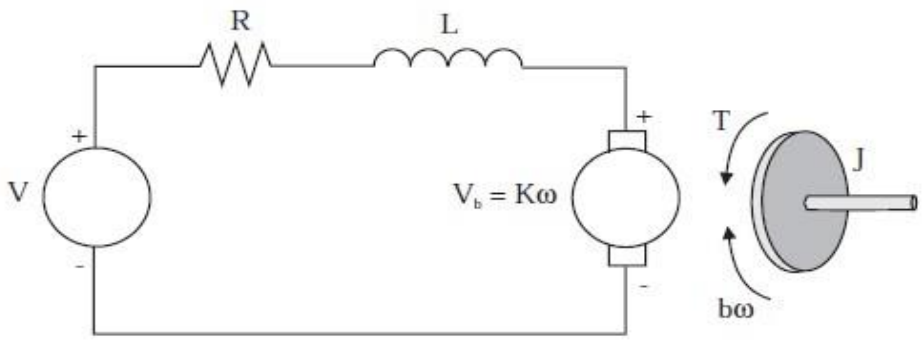

Figure 5 Schematic Representation of DC motor

3.1.2 Model of DC motor

In this work, the separated excitation DC motor model is chosen according to his good electrical and mechanical performances more than other DC motor models. The DC motor is driven by applied voltage. Fig.1 show the equivalent circuit of DC motor with separate excitation. The characteristic equations of the DC motor are represented as:

$$\frac{d}{dt}i_{ex} = \left(-\frac{R_{ex}}{L_{ex}}\right).i_{ex} + \left(\frac{1}{L_{ex}}\right).V_{ex} \quad (1)$$

$$\frac{d}{dt}i_{ind} = \left(-\frac{R_{ind}}{L_{ind}}\right).i_{ind} + \left(\frac{-L_{index}}{L_{ind}}\right).w_r.i_{ex} + \left(\frac{1}{L_{ind}}\right).V_{ind} \quad (2)$$

$$\frac{d}{dt}w_r = \left(\frac{L_{index}}{J}\right).i_{ex}.i_{ind} + \left(\frac{-Cr}{J}\right) + \left(\frac{-fc}{J}\right).w_r \quad (3)$$

Table 4 Used symbols

Symbols	Designations	Units
iex and iind	Excitation current and Induced current.	**[A]**
Wr	Rotational speed of the DC Motor.	**[Rad/Sec]**
Vex and Vind	Excitation voltage and Induced voltage	**[Volt]**
Rex and Rind	Excitation Resistance and Induced Resistance.	**[Ω]**
Lex , Lind and Lindex	Excitation Inductance, Induced Inductance and Mutual Inductance.	**[mH]**
J	Moment of Inertia.	**[Kg.m2]**
Cr	Couple resisting.	**[N.m]**
fc	Coefficient of Friction.	**[N.m.Sec/Rad]**

The block diagram for DC motor will construct as Figure 2.2 by using the transfer Function that obtained from the schematic representation of DC motor.

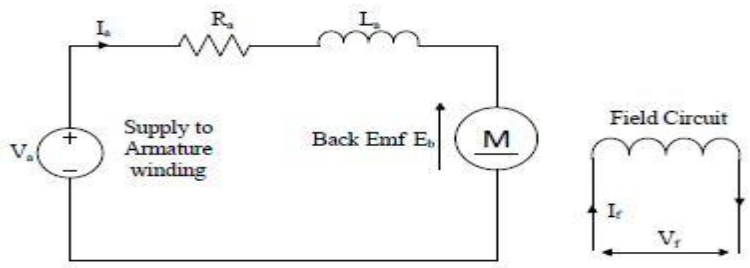

Figure 6 3 DC motor model

Some useful relations are:

Va(t) = Raia(t) + La +Eb (t) (1)
Eb (t) = Kbω(t)…….. (2)
Tm(t) = Ktia(t) ……...(3)

Tm(t) - TL(t) = jm + Bmω(t)……..…… (4)

Speed Control of DC Motor

Substitute (3) in (2) and (4) in (5), we get
Va(t) = Raia(t) + La +Kbω (t) ……..…..(5)
Ktia(t)= jm +Bmω(t) …(6)
Taking Laplace transform of equation (6) and (5),
Va(s) = Raia(s) + sLa Ia (s) +Kbωs ……..….(7)

KtIa(s)= s jm ω(s) + Bmω(s) ……(8)

The transfer function of DC motor is:

$$\frac{\theta(s)}{V(s)} = \frac{K}{L_a J s^3 + (R_a J + B L_a) s^2 + (K^2 + R_a B) s}$$

L_a = armature Inductance
R_a = armature resistance
K = motor constant
J = moment of inertia
B = mechanical friction

The parameters of the electric DC motor have the following value respectively, J=0.042, B=0.01625, K=0.9, L=0.025, R=5 as a nominal value.

The transfer function of the electric DC motor is[1]

$$P(s) = \frac{0.9}{0.00105 s^3 + 0.2104 s^2 + 0.8913 s}$$

$K_p = 10 K_p = 20$

3.2 Transient Responses of P, P-D, P-I and P-I-D controllers:

In this part, transient performances of P, P-D, P-I and P-I-D controllers are explained. Their steady state error performances are also discussed.

3.2.1 Transient Response of P Controller:

As a general rule, increasing proportional gain decreases the steady state error. However, the actual performance of P controller depends on the order of the plant. If P controller is used to control a second order plant, it has following properties:

- **Increasing gain decreases rise time (Advantage)**
- **Increasing gain increases percent overshoot and number of oscillations (Disadvantage)**
- **Increasing gain decreases steady state error (Advantage)**
- **Steady state is never zero if only-P type controller is used (Disadvantage)**
- **In order to have zero steady state error gain should be infinity(Physically impossible)**

The discussion above shows that only-P control is not enough to control second order plants. In fact, only-P control is usually used to control first order plants, because there are no natural oscillations in first order plants and P control is easy to implement.

3.2.2 Transient Response of P-D Controller:

Derivative action is usually used to improve transient response of the closed loop system. Only D control is not used because it amplifies high frequency noise which is never desired. Derivative action decreases rise time and oscillations. However, it does not have any effect on steady state performance of the closed loop.

The discussion above indicates that with P-D control, steady state error is still non-zero. Derivative control is usually used to decrease oscillations in closed loop system outputs. The following simulations were done on MATLAB-Simulink to illustrate the performance of P-D control on first and second order plants.

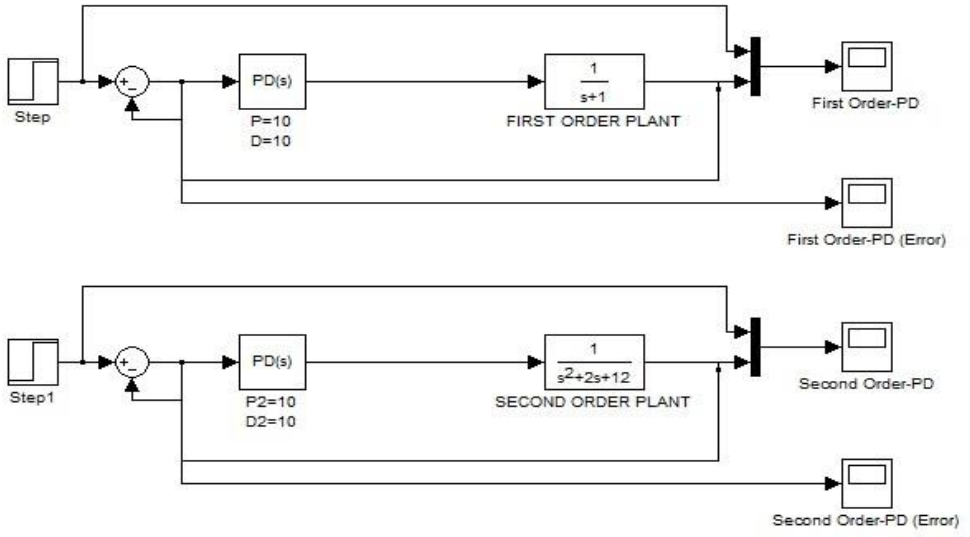

Figure 7 P-D control on first and second order plants

First order continuous plant transfer function:

$$G_p(s) = \frac{1}{s+1}$$

Second order continuous plant transfer function:

$$G_p(s) = \frac{1}{s^2+2s+12}$$

3.2.3 Transient Response of P-I Controller:

Integral action eliminates steady state error. However, it has very poor transient response. Using integral action increases the oscillations in the output of the closed loop systems.

The discussion above indicates that with P-I control, steady state error is non-zero. However, Integral control causes too many oscillations in closed loop system outputs. The following simulations were done on MATLAB-Simulink to illustrate the performance of P-I control on first and second order plants.

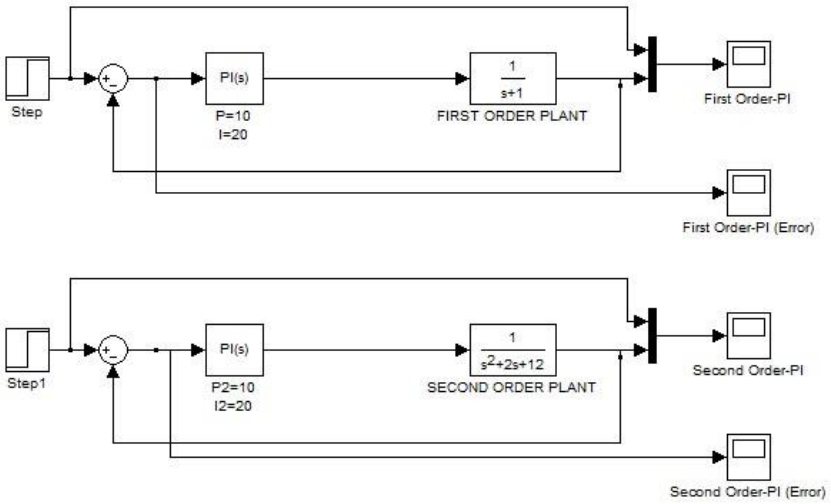

Figure 8 of P-I control on first and second order plants

First order continuous plant transfer function:

$$G_p(s) = \frac{1}{s+1}$$

Second order continuous plant transfer function:

$$G_p(s) = \frac{1}{s^2+2s+12}$$

3.2.4 Transient Response of P-I-D Controller:

P-I-D controller is the optimal controller for high order plants. It has zero steady state error together with acceptable transient response. The only problem with P-I-D control is tuning. Fortunately, MATLAB has automatic tuning option. However, automatic tuning does not usually provide the best results, it only provides optimal results. P-I-D tuning is an engineering art and should be manually done by control engineers.

The following simulations were done on MATLAB-Simulink to illustrate the performance of P-I-D control on first and second order plants {Prasad, 2012}.

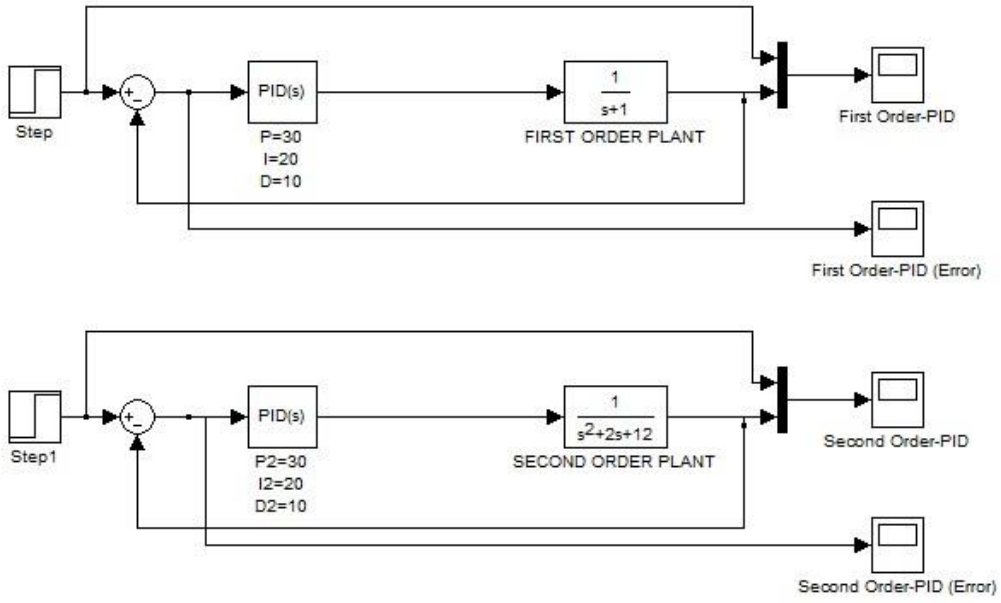

Figure 9 P-I-D control on first and second order plants

First order continuous plant transfer function:

$$G_p(s) = \frac{1}{s+1}$$

Second order continuous plant transfer function:

$$G_p(s) = \frac{1}{s^2+2s+12}$$

Why do we need to control the speed of DC motor?

For many cases, we cannot obtain the same desired results in terms of theoretical and practical cases. For that project, we have to make theoretical power calculations for DC motors to obtain the desired DC motor speed. However, in practice, we could not obtain the same results as it is calculated theoretically. For that purpose, we have to use controllers to minimize the error between actual and theoretical results.

Why to choose P-I-D as 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. 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. The P-I-D algorithm and the whole code segments can be seen in Appendix.

3.3 PID Parameters:

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.

3.3.1 CLASSICAL PID CONTROLLER

A proportional-integral-derivative controller (PID controller) is basically a generic control loop feedback mechanism widely used in industrial control systems [2]. A PID controller calculates an "error" value as the difference between a measured plant variable and a desired setpoint. The controller attempts to minimize the error by adjusting the

process control inputs. Fig.1 shows a basic structure of a closed loop controller.

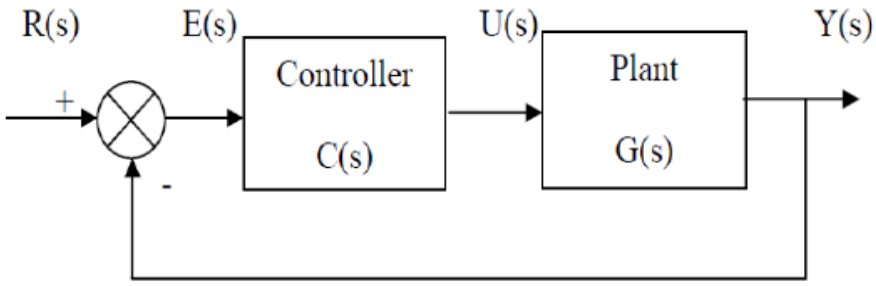

Figure 10 Classical Controller

The differential equation of a PID controller is given by:

$$U(t) = Kp\ e(t) + 1\ Ti\int e(t)dt + Td \times de(t)\ dt + P0$$

and the transfer function is given by:

$$G_{PID}(s) = K_p + \frac{Ki}{s} + s \cdot K_d$$

Where,

K_p = proportional gain T_i = integral time T_d= derivative time The variable e(t) represents the tracking error which is the difference between the desired input value and the actual output. This error signal will be sent to the PID controller and the controller computes both the derivative and the integral of this error signal. The signal U(t) from the controller is now equal to the proportional gain (K_p) times the magnitude of the error plus the integral gain (K_i) times the integral of the error plus the derivative gain (K_d) times the derivative of the error [2].

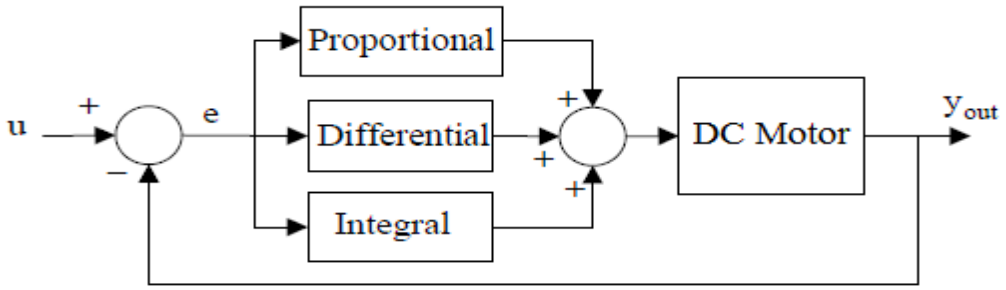

Figure 11 Model of DC motor with PID Controller

The DC motor has a PID controller which is presented in the below:

$$PID = K_p + \frac{K_i}{s} + K_d s$$

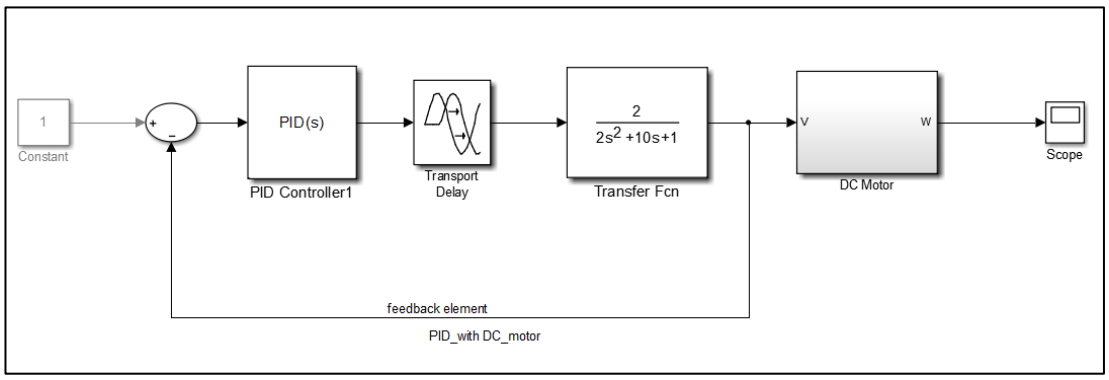

Figure 12 System Block Diagrams from MATLAB/Simulink

The PID controller makes a control loop respond faster with less overshoot and most popular method of control by a great margin. The combined action has the advantages of each of the three individual control actions.

Table 5 Affection Various O/P Parameters of P, Pi and Pid Controller

Parameter	**P Controller**	**PI Controller**	**PID Controller**
Rise time	Decrease	Decrease	Minor Decrease
Overshoot	Increase	Increase	Minor Decrease
Settling time	Small change	Increase	Minor Decrease
Steady state error	Decrease	Significant change	No change
Stability	Worse	Worse	If Kd Small Better.

Table 6 Comparison of Gain Response Of P, Pi And Pid Controllers.

Parameter	Speed of Response	Stability	Accuracy
increasing k	Increase	Deteriorates	Improves
increasing ki	Decrease	Deteriorates	Improves
increasing kd	Increase	Improves	No impact

Table 1 and 2 shows the effects of coefficients and effects of changing control parameters respectively . As we can there see is a decrease in rise time, overshoot and settling time and there is no change in steady state error PID Controller is better than P and PI controller.

3.4 Particle Swarm Optimization (PSO) Algorithm:

PSO is a method for optimizing hard numerical functions on metaphor of social behavior of flocks of birds and schools of fish. The original PSO algorithm is discovered through simplified social model simulation. It was first designed to emulate birds seeking food which is defined as a cornfield vector. The bird would find food through social cooperation with other birds around it (within its neighborhood). It was then expanded to multidimensional search. In PSO each particle in swarm represents a solution to the problem and it is defined with its position and velocity {, 6}.

3.4.1 Scheduling PSO for PID Controller Parameters

In this work, An PID controller used PSO Algorithms to find the optimal parameters of DC Motor speed control system. The structure of the PID controller with PSO algorithms is shown in Fig. 4.21.

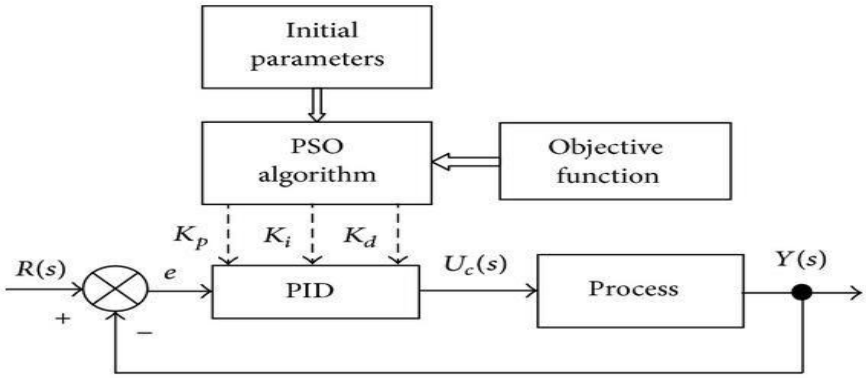

Figure 13 Structure of the PID controller with PSO algorithm

The main steps in the particle swarm optimization and selection process are described as follows:

The main steps in the particle swarm optimization and selection process are described as follows:

(a) Initialize a population of particles with random positions and velocities in d dimensions of the problem space and fly them.

(b) Evaluate the fitness of each particle in the swarm.

(c) For every iteration, compare each particle's fitness with its previous best fitness ($pbest$) obtained. If the current value is better than $pbest$, then set $pbest$ equal to the current value and the $pbest$ location equal to the current location in the d-dimensional space.

(d) Compare $pbest$ of particles with each other and update the swarm global best location with the greatest fitness ($gbest$).

(e) Change the velocity and position of the particle According to equations (11) and (12) respectively.

(f) Repeat steps (a) to (e) until convergence is reached based on some desired single or multiple criteria.

3.4.2 System Flow Chart

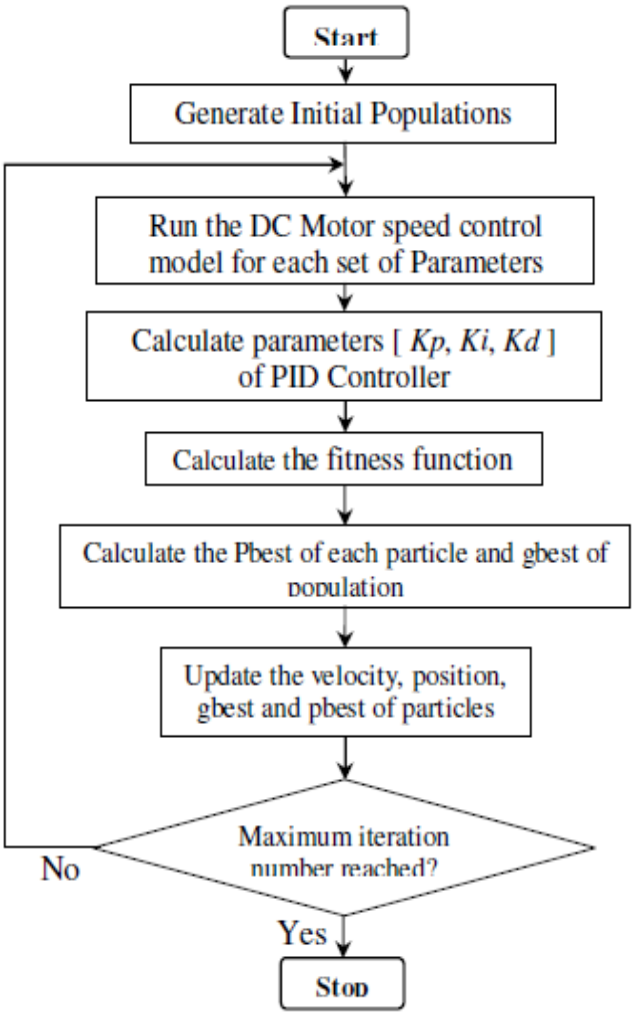

Figure 14 System Flowcharts with PSO-PID Control System

The best previous position of the *i*-th particle is recorded and represented as:

$$\text{Pbest}_i = (\text{Pbest}_{i,1}, \text{Pbest}_{i,2}, \ldots, \text{Pbest}_{i,d}) \quad (4)$$

The index of best particle among all of the particles in the group is gbestd. The velocity for particle i is represented as $v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,d})$. The modified velocity and position of each particle can be calculated using the current velocity and the distance from $\text{Pbest}_{i,d}$ to gbest_d as shown in the following formulas [9, 10, 11]:

$$v_{i,m}^{(t+1)} = w.v_{i,m}^{(t)} + c_1 * rand\,() * (Pbest_{i,m} - x_{i,m}^{(t)}) + c_2 * Rand\,() * (gbest_{m} - x_{i,m}^{(t)}) \quad (5)$$

$$x_{i,m}^{(t+1)} = x_{i,m}^{(t)} + v_{i,m}^{(t+1)} \quad ; \quad i=1,2,\ldots,n \quad ; \quad m=1,2,\ldots,d \quad (6)$$

where:

N Number of particles in the group,

d dimension,

t Pointer of iterations(generations),

$v_{i,m}^{(t)}$ Velocity of particle I at iteration t, $V_d^{min} \le v_{i,d}^{(t)} \le V_d^{max}$

w Inertia weight factor,

c_1, c_2 Acceleration constant, *rand*() Random number between 0 and 1, *Rand*()

$x_{i,d}^{(t)}$ Current position of particle i at iterations,

$Pbest_i$ Best previous position of the i-th particle, gbest Best particle among all the particles in the population.

Realization of a PID-PSO Controller Tuning Optimal Parameters

Fitness Function

The general equation of PID controller is [8]:

$$U(t) = K_p \times e(t) + \frac{1}{T_i}\int e(t)dt + T_d \frac{de(t)}{dt} \quad (7)$$

where: K_p = proportional gain; T_i = integral time; T_d= derivative time.

The IAE, ISE, and ITSE performance criterion formulas are as follows:

$$IAE = \int_0^{\infty} |r(t) - y(t)dt| = \int_0^{\infty} |e(t)|dt \quad (8)$$

$$ISE = \int_0^{\infty} e^2(t)dt \quad (9)$$

$$ITSE = \int_0^{\infty} t \cdot e^2(t)dt \quad (10)$$

In this work a time domain criterion is used for evaluating the PID controller. A set of good control parameters P, I and D can yield a good step response that will result in performance criteria minimization in the time domain. These performance criteria in the time domain include the overshoot, rise time, settling time, and steady-state error.

Chapter Four

4. SIMULATION RESULTS

4.1 DC Motor with PID manual tuning

Graphs after simulation are:

4.1.1 PID-Controller Effect on DC motor:

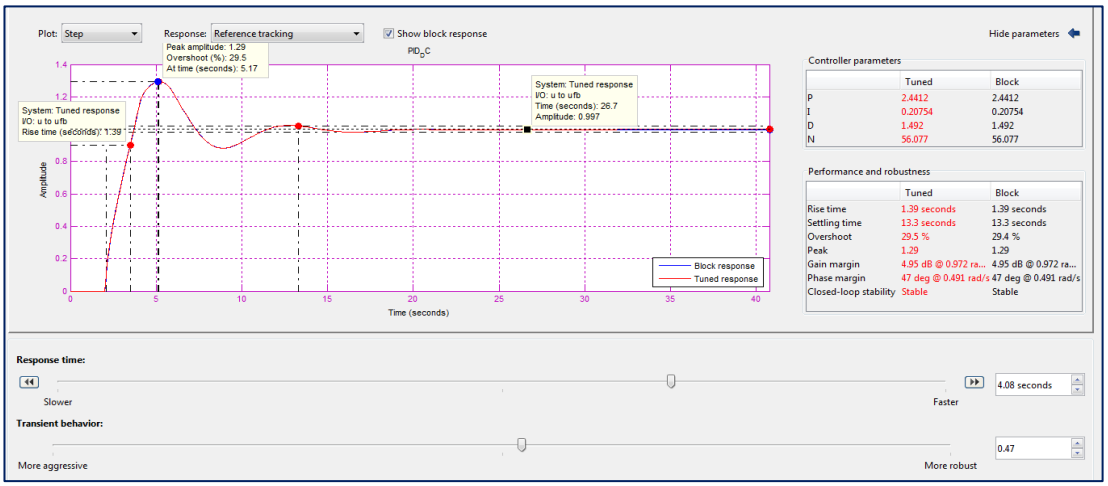

Figure 15 PID Controllers Effects

Table 7 PID-Controller

Performance and Robustness	Proportional Integral Derivative-DC motor		
	Tuned	Block	Note
Rise time	1.39sec	1.39se	No error (system stable)
Settling time	13.3sec	13.3se	No error (system stable)
Overshoot	29.5%	29.4%	Normal effect (0.1%)
Peak	1.29	1.29	No error (system stable)
Gain margin	4.95dB	4.95dB	No error (system stable)
Phase margin	47deg	47deg	No error (system stable)
Stability	stable	stable	No error (system stable)

Table (10-4) showing the effects of coefficients and effects of changing control parameters respectively.As we can there see is a decrease in rise time, overshoot and settling time and there is no change in steady state error PID Controller is better than P and PI controller.

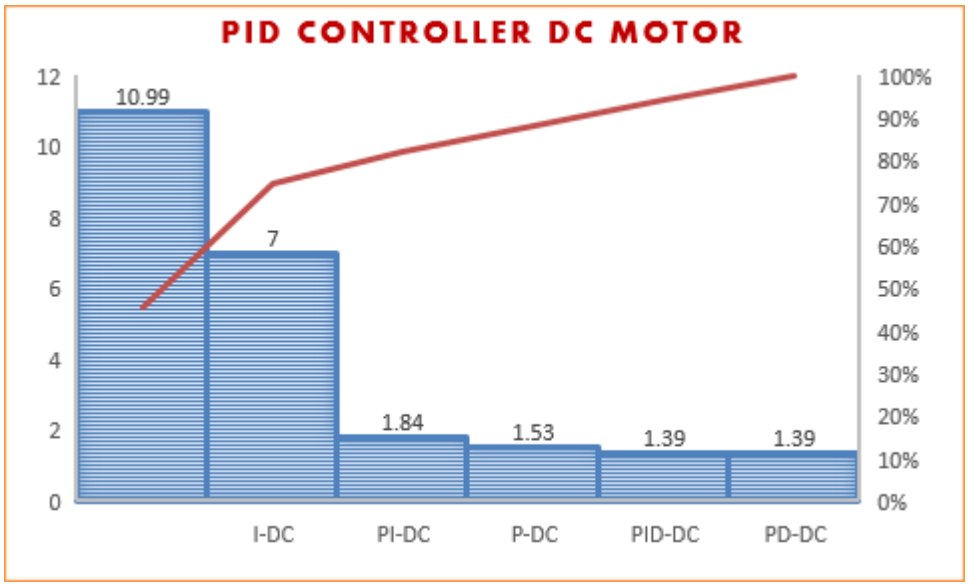

Table 8 All Parameters-Controllers

All Parameters	PID-DC		PD-DC		PI-DC		I-DC		P-DC	
	Tuned	Block	Tuned	Block	Tuned	Block	Tuned	Block	Tuned	Block
Rise time	1.39sec	1.39se	1.39se	1.12sec	1.84sec	1.67sec	7sec	5.32s	1.53s	1.51
Settling time	13.3sec	13.3se	24sec	15.9sec	286sec	32.4sec	200se	740se	25.9s	26sec
Overshoot	29.5%	29.4%	50.7%	38.9%	25.6%	53.3%	68%	91.1%	52.4%	53.8%
Peak	1.29	1.29	1.23	1.15	1.26	1.53	1.68	1.92	1.26	1.28
Gain margin	4.95dB	4.95dB	3.9dB	4.87dB	3.85%	3.36%	4.73dB	1.08dB	4.13dB	3.99dB
Phase margin	47deg	47deg	47deg	54.1deg	38.9deg	29.7deg	14.7deg	3.37deg	41.8deg	40.6deg
Stability	stable	stable	stable	stable	stable	stable	stable	stable	stable	stable

4.1.2 Performance Comparison of P, PI, and PID

It is to be noted that, when gain is increasing speed of response is increasing in case of P and PID controller but in PI controller gain of response is decreasing. In PID controller there is a minor decrease or no changes are shown in various parameter which can see from table (5-4).

Hence there is no change in steady state error so PID controller is better than P and PID controller.

4.2. Discussion Results

The model of DC motor and the optimal control of speed were numerically simulated using a state space model and Matlab/Simulink software for a separated exited DC motor with the following parameters: DC motor 30W, 12 V, 300 rad/s, total inertia 0.01kgm2. The simulation procedure may be summarized as follows: • First input the DC motor data, • Write the differential equations for the model then get the state space representation • Get the open loop transfer function and the closed loop step response • Finally performing the performance of PID controller by Ziegler Nichols method and PID controller by using PSO algorithm and compare the results.

Graphs after simulation are:

4.2.1 Open-Loop Step Responses (Without Control)

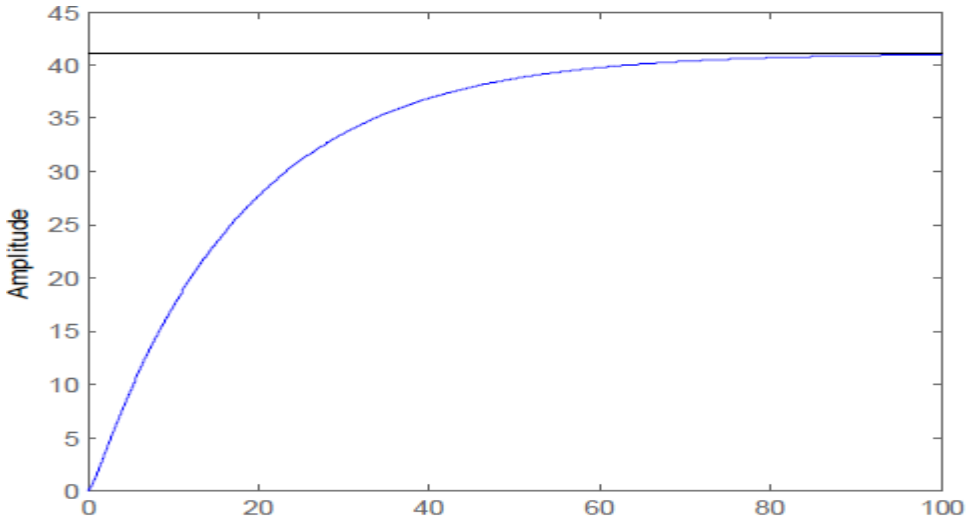

Figure 16 Open-Loop Step Responses

4.2.2 Closed-Loop Step Responses (Without Control)

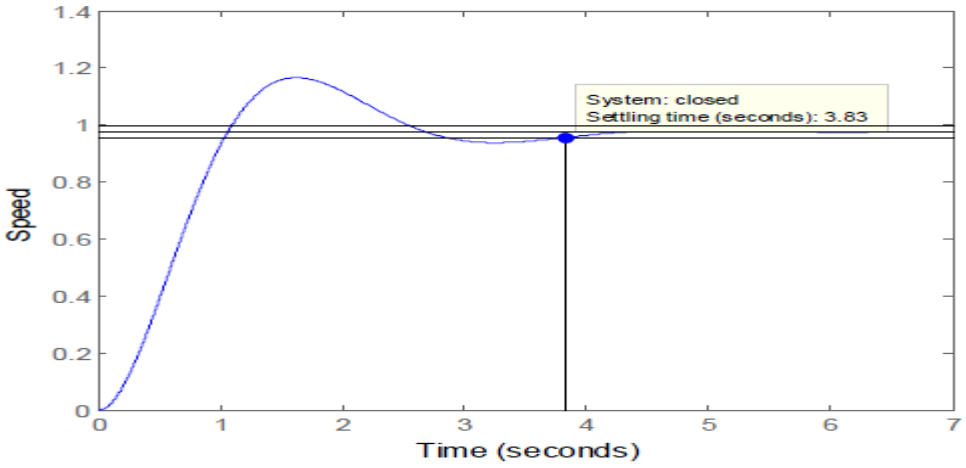

Figure 17 Closed-Loop Step Responses

4.2.3 PID step response using Ziegler-Nichols Method

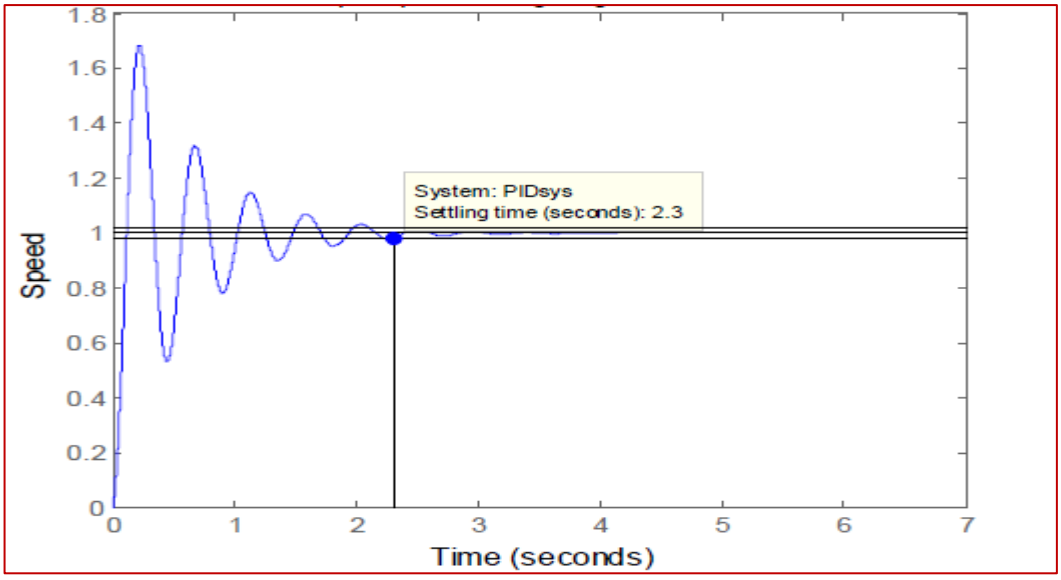

Figure 18 PID step response using Ziegler-Nichols Method

4.3 Implementation of PSO-PID Controller

In this work, a PID controller using the PSO algorithm is developed to improve the results of speed control of DC motor. The PSO algorithm is mainly utilized to determine three optimal controller parameters kp, ki,and kd, such that the controlled system could obtain a desired step response output

4.3.1 Result of PSO-PID Controller

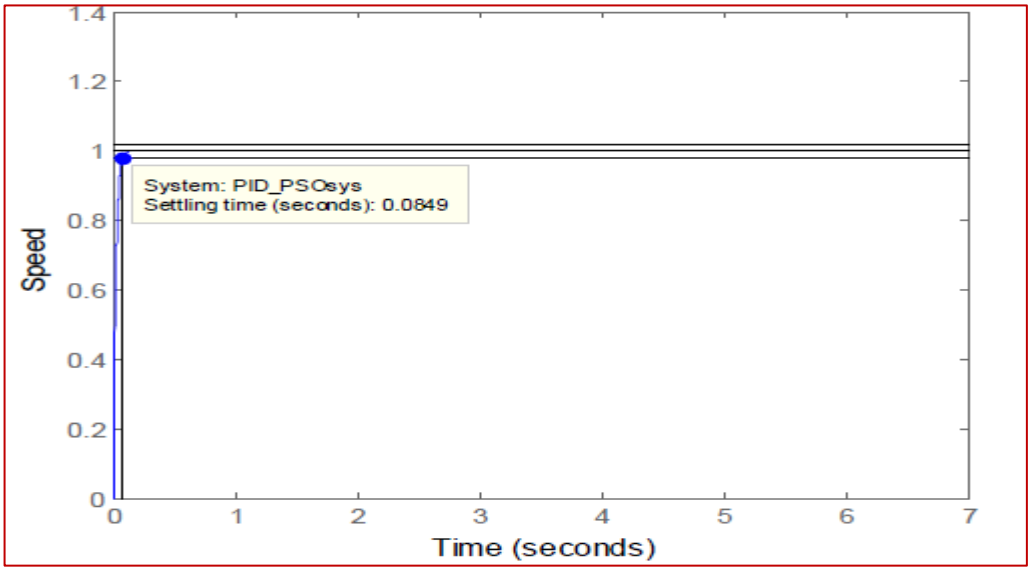

Figure 19 Step Responses with PSO-PID

To show the effectiveness of the proposed approach, is made a comparison with designed of PID controller using Ziegler-Nichols method, Linear Quadratic Regulator and PSO algorithm method in Table 2. Table.2 Comparison of the system

Table 9 Comparison System between Zigler-Nicols and PSO algorithm

Method	*Settling time (sec)*	*Overshoot (%)*	*Rise time (sec)*
Closed System	3.83	19.5	0.721
Z-N PID	2.3	68.6	0.0815
PSO-PID	0.0849	0	0.0451

PID controllers are a widespread control solution due to their simple architecture, generally acceptable control performance and ease of use. In this work PID controller has been tuned using Ziegler-Nichols method and Particle Swarm Optimization (PSO) through simulation of DC motor speed control system. The performance of the PSO algorithm method of tuning a PID controller has been proved to be better than traditional method Ziegler-Nichols method, in terms of the system overshoot, settling time and rise time.

4.4 Robustness Investigation

The PID controllers tuned by the PSO based method should not be compared only with their time domain responses but also with its performance index from the four major error criterion techniques of Integral Time of Absolute Error (ITAE) ,Integral of Absolute Error(IAE) ,Integral Square of Error(ISE)and Mean Square Error (MSE).Robustness of the controller is defined as its ability to tolerate a certain amount of change in the process parameters without causing the feedback system to go unstable.

For the proposed model the comparison of performance index were done and are listed as per the given Table below:

Table 10 proposed model the comparison of performance index

Performance index	ZN	PSO
ITAE	1.1824	0.2157
IAE	3.4496	1.9047
ISE	1.8844	1.2291
MSE	0.0369	0.0241

Table 15.4: Comparison of performance index obtained for Z-N and from these values obtained it is clearly visible that the error magnitude obtained for Z-N is far too high as compared to the proposed tuning method based on PSO algorithm.

Chapter Five

5. CONCLUTION AND RECOMMENDATIONS

5.1 Conclusions

In this work, a PSO method is used to determine PID controller parameters automatically through simulation of DC motor speed control system. The results show that the proposed controller can perform an efficient search for the optimal PID controller by comparing with the conventional controller methods, it shows that this method have exhibited relatively good performance and the output response full tracking with speed reference for all time response and their typical characteristics show a faster and smoother response.

The advantage of using PSO 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 PSO-PID technique gives better response than PID controller in terms of trajectory tracking. the results show that the proposed controller can perform an efficient search for the optimal PID controller. By comparison with ZgNc-PSO controller, it shows that this method can improve the dynamic performance of the system in a better way. The PID-PSO controller is the best which presented satisfactory performances and possesses good robustness (no overshoot, minimal rise time, Steady state error approximately = 0).

Finally, the proposed automatic tuning is intelligent method to control a nonlinear input an actuator and to regulate the speed of motor without using the conventional solution (dither signal) that is simplicity but its limited when using disturbance rejection in nonlinear system.

5.2 Recommendations

Although, this thesis has tried to find the suitable topology for PID-PSO designed according some conditions it may be difficult to apply in all practical fields. So we recommend the following.

- Implementation of an adaptive fuzzy logic control technique for brushless motor control.
- Applying the Ant Colony optimization approach in the tuning of PID controller.
- Adapting sophisticated control strategies such as neural network and neurofuzzy control techniques.

6. References

PSO-Based PID Controller Design for a Class of Stable and Unstable Systems. *PSO-Based PID Controller Design for a Class of Stable and Unstable Systems*.

2016. Performance comparison of PID tuning by using Ziegler-Nichols and particle swarm optimization approaches in a water control system.

AGARWAL, S., MATHUR, S., MISHRA, P., KUMAR, V., RANA, K. P. S., INTERNATIONAL CONFERENCE ON COMPUTING, C., AMP & AUTOMATION 2015. Online tuning of fractional order PI controller using particle swarm optimization. 1026-1031.

AGGRAWAL, A., MISHRA, A. K. & ZEESHAN, A. 2014. Speed Control of DC Motor Using Particle Swarm Optimization Technique by PSO Tunned PID and FOPID. *IJETT International Journal of Engineering Trends and Technology,* 16**,** 72-79.

ALLAOUA, B., GASBAOUI, B. & MEBARKI, B. 2009. Setting up PID DC motor speed control alteration parameters using particle swarm optimization strategy. *Leonardo Elect. J. Pract. Technol. Leonardo Electronic Journal of Practices and Technologies,* 7**,** 19-32.

BAGIS, A. 2007. Determination of the PID Controller Parameters by Modified Genetic Algorithm for Improved Performance. *JOURNAL OF INFORMATION SCIENCE AND ENGINEERING,* 23**,** 1469-1480.

BAHGAAT, N. K. & MOUSTAFA HASSAN, M. A. 2016. Swarm Intelligence PID Controller Tuning for AVR System.

BAYOUMI, E. H. E., CONTROL, I. S. O. C. I. I. & AUTOMATION 2013. Minimal overshoot direct torque control for permanent magnet synchronous motors using Hybrid Bacteria Foraging-Particle Swarm Optimization. 112-119.

EL YAKINE KOUBA, N., MENAA, M., HASNI, M., BOUDOUR, M., TH INTERNATIONAL CONFERENCE ON, S. & CONTROL 2015. Optimal control of frequency and voltage variations using PID controller based on Particle Swarm Optimization. 424-429.

FREIRE, H. L., MOURA OLIVEIRA, P. B. & SOLTEIRO PIRES, E. J. 2017. From single to many-objective PID controller design using particle swarm optimization. *INTERNATIONAL JOURNAL OF CONTROL AUTOMATION AND SYSTEMS,* 15**,** 918-932.

GIRIRAJKUMAR, S. M., KUMAR, A. A. & ANANTHARAMAN, N. 2010. Tuning of a PID Controller for a Real Time Industrial Process using Particle Swarm Optimization. *IJCA International Journal of Computer Applications,* ecot**,** 35-40.

INTERNATIONAL CONFERENCE ON SOFT COMPUTING FOR PROBLEM, S., PANT, M., DEEP, K., BANSAL, J. C., NAGAR, A. K. & DAS, K. N. 2016. Proceedings of fifth International Conference on Soft Computing for Problem Solving : SocProS 2015. Volume 2 Volume 2.

KENNEDY, J., SPEARS, W. M. & INTELLIGENCE, I. I. C. O. E. C. P. I. W. C. O. C. 1998. Matching algorithms to problems: an experimental test of the particle swarm and some genetic algorithms on the multimodal problem generator. 78-83.

KIM, D. H. & CHO, J. H. 2011. Retraction of A Biologically Inspired Intelligent PID Controller Tuning for AVR Systems. *Int. J. Control Autom. Syst. International Journal of Control, Automation and Systems,* 9**,** 814.

LATHA, K., RAJINIKANTH, V. & SUREKHA, P. M. 2013. PSO-Based PID Controller Design for a Class of Stable and Unstable Systems. *ISRN Artificial Intelligence ISRN Artificial Intelligence,* 2013**,** 1-11.

LI, X., WANG, Y., LI, N., HAN, M., TANG, Y. & LIU, F. 2017. Optimal fractional order PID controller design for automatic voltage regulator system based on reference model using particle swarm optimization. *Int. J. Mach. Learn. & Cyber. International Journal of Machine Learning and Cybernetics,* 8**,** 1595-1605.

MOHD AMIR FIKRI, A. 2010. *DC motor speed controller.* UMP.

NISHIDA, T., SAKAMOTO, T. & GIANNOCCARO, N. I. 2013. Self-Tuning PI Control Using Adaptive PSO of a Web Transport System with Overlapping Decentralized Control. *EEJ Electrical Engineering in Japan,* 184**,** 56-65.

PRASAD, L. B., TYAGI, B., GUPTA, H. O. & TH ASIA MODELLING, S. 2012. Modelling and Simulation for Optimal Control of Nonlinear Inverted Pendulum Dynamical System Using PID Controller and LQR. 138-143.

ROY, P. K., LAIK, S. K., DEY, S. K., DAS, P. K., SULTANA, S. K. & PAUL, S. K. 2015. Automatic Generation Control of Interconnected Power System using Cuckoo Optimization Algorithm. *International Journal of Energy Optimization and Engineering (IJEOE),* 4**,** 22-35.

SHARAF, A. M. & EL-GAMMAL, A. A. A. 2011. Multi-objective PSO/GA optimization control strategies for energy efficient PMDC motor drives. *ETEP European Transactions on Electrical Power,* 21**,** 2080-2097.

UFNALSKI, B., GRZESIAK, L. M. & KASZEWSKI, A. 2014. Advanced Control and Optimization Techniques in AC Drives and DC/AC Sine Wave Voltage Inverters: Selected Problems.

Myrtellari, A., Marango, P. and Gjonaj, M., 2016. Analysis and Performance of Linear Quadratic Regulator and PSO algorithm in optimal control of DC motor. International Journal of Latest Research in Engineering and Technology, 2(4).

7. Appendixes

Appendix A: PSO- Code

```
%Particle Swarm Optimization Simulation
%find minimum of the objective function
%Ahmed Humaida:(ahmedhumaida@gmail.com)
%MSC student, Electronic Eng. SUST
%%Initialization

Clear
clc
iterations=30;
inertia=1.0;
correction_factor=2.0;
swarms=50;
%---initial swarm position---
swarm=zeros(50,7)
step=1;
for i=1:50
swarm(step,1:7)=i;
step=step+1;
end

swarm(:,7)=1000 %greater than maximum possible value
swarm(:,5)=0    %initial velocity
swarm(:,6)=0    %initial velocity

%%Itrations

foriter=1:iterations

%---position of swarms---
for i=1:swarms
        swarm(i,1)=swarm(i,1)+swarm(i,5)/1.2 %update u position
        swarm(i,2)=swarm(i,2)+swarm(i,6)/1.2 %update v position
       u=swarm(i,1)
       v=swarm(i,2)
value=(u-20)^2+(v-10)^2 %objective function
if value<swarm(i,7) %always true
swarm(i,3)=swarm(i,1) %update best position of u,
swarm(i,4)=swarm(i,2) %update best position of v,
swarm(i,7)=value %best update minimum value
end
end

    [temp,gbest]=min(swarm(:,7)) %gbest position
%---updating velocity of swarms
for i=1:swarms

        swarm(i, 5)=rand*inertia*swarm(i,
5)+correction_factor*rand*(swarm(i, 3)-swarm(i,
1))+correction_factor*rand*(swarm(gbest, 3)-swarm(i, 1)) %u velocity
parameters
        swarm(i, 6)=rand*inertia*swarm(i,
6)+correction_factor*rand*(swarm(i, 4)-swarm(i,
```

```
2))+correction_factor*rand*(swarm(gbest, 4)-swarm(i, 2)) %v velocity
parameters
end

%%plotting
clf
plot(swarm(:,1),swarm(:,2),'x') %drawing swarms
axis([-10 50 -10 50])
pause(.1)
end
```

Appendix B: PSO-PID controller Code

```
Clear
clc
iterations=20;
inertia=1.0;
correction_factor=2.0;
swarm_size=10;
%---initial swarm position---
index = 1;
for i = 1 : 1
for n = 1 : 5
for j = 1 : 4
swarm(index, 1, 1) = n;
swarm(index, 1, 2) = i;
swarm(index, 1, 3) = j;
index = index + 1;
end
end
end
swarm(:, 4, 1) = 10000; % best value so far
swarm(:, 2, :) = 0; % initial velocity
%% Iterations
foriter = 1 : iterations
for i = 1 : swarm_size
%-- evaluating position
swarm(i, 1, 1) = swarm(i, 1, 1) + swarm(i, 2, 1)/1.3; %update x
position
swarm(i, 1, 2) = swarm(i, 1, 2) + swarm(i, 2, 2)/1.3; %update y
position
swarm(i, 1, 3) = swarm(i, 1, 3) + swarm(i, 2, 3)/1.3; %update z
position
if(swarm(i, 1, 1)<0)
swarm(i, 1, 1)=0.1;
end
if(swarm(i, 1, 2)<0)
swarm(i, 1, 2)=0.5;
end
if(swarm(i,1,3)<0)
swarm(i,1,3)=0.4;
end
kp = swarm(i, 1, 1);
ki = swarm(i, 1, 2);
kd = swarm(i, 1, 3);
simBLDC.mdl;
iter
i
```

```
val = min(ISTE); % fitness evaluation (you may replace this
objective function
with any function having a global minima)
ifval< swarm(i, 4, 1) % if new position is better
swarm(i, 3, 1) = swarm(i, 1, 1); % update best x,
swarm(i, 3, 2) = swarm(i, 1, 2); % best y postion
swarm(i, 3, 3) = swarm(i, 1, 3); % best z position
swarm(i, 4, 1) = val; % and best value
end
end
[temp, gbest] = min (swarm(:, 4, 1)); % global best position
%--- updating velocity vectors
for i = 1 : swarm_size
swarm(i, 2, 1) =rand*inertia*swarm(i, 2, 1) +
correction_factor*rand*(swarm(i, 3,
1) - swarm(i, 1, 1)) + correction_factor*rand*(swarm(gbest, 3, 1) -
swarm(i, 1, 1)); %x
velocity component
swarm(i, 2, 2) = rand*inertia*swarm(i, 2, 2) +
correction_factor*rand*(swarm(i, 3,
2) - swarm(i, 1, 2)) + correction_factor*rand*(swarm(gbest, 3, 2) -
swarm(i, 1, 2)); %y
velocity component
swarm(i, 2, 3) = rand*inertia*swarm(i, 2, 3) +
correction_factor*rand*(swarm(i, 3,
3) - swarm(i, 1, 3)) + correction_factor*rand*(swarm(gbest, 3, 3) -
swarm(i, 1, 3)); %z
velocity component
end
% t(iter)=swarm(gbest, 3);
t(iter)=temp;
tx(iter)=iter;
%% Plotting the swarm
clf
%plot(iter,ISTE)
grid on
plot(swarm(:, 1, 1), swarm(:, 1, 2),'x') % drawing swarm movements
axis([-5 5 -5 5]);
pause(.1)
end
x(1,1)=swarm(gbest, 3, 1);
x(1,2)=swarm(gbest, 3, 2);
x(1,3)=swarm(gbest, 3, 3);
kp=swarm(gbest, 3, 1);
ki=swarm(gbest, 3, 2);
kd=swarm(gbest, 3, 3);
kp
ki
kd
```