Sudan University of Science and Technology

College of Graduate Studies

PID Controller tuning optimization for Computer Numerical Control using particle Swarm Optimization algorithm

تحسين توليف المتحكم التناسبي التكاملي التفاضلي بغرض التحكم الرقمي في الحاسب بأِستخدام خوارزمية سرب الجسيمات

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

Prepared by:

Tagwa Mukhtar Mohammed Doka

Supervised by:

Dr. Alaaeldin Awoda

December 2018

الاستهلال

بسم الله الرحمن الرحيم

قال تعالى:

(وما اوتيتم من العلم الا قليلا")

صدق الله العظيم

DEDICATION

I dedicate this work to those who were very caring, helpful and encouraging. To those who carried me for advancement and success.

To my Mother, my family who supported me and believed in my capabilities, my friends and my students who were there for me.

ACKNOWLEDGEMENT

All praise due to Allah, the all mighty god, who spread his and wisdom across the globe, all our thanks to our lord.

To the supervisor Dr. AlaEldin Awouda, the guider of my work, who gave me his trust in the knowledge and work, being determined and dedicated, helped and motivated me, to reach this form.

To my Mom, friends and families, those who pushed my goals till clarity, supported me to the limit for it to be, sometimes over pushed it just so i succeed.

ABSTRACT

In this thesis, an artificial intelligence method Particle Swarm Optimization (PSO) algorithm is presented for determining the optimal proportional-integral-derivative (PID) controller parameters of a servo motion system used in Computer Numerical Control (CNC). This thesis demonstrates in detail on how to employ the PSO method to search efficiently the optimal PID controller parameters of the servo motor. In order to assist estimating the performance of the proposed PSO-PID controller is modeled using MATLAB environment. The proposed approach yields better solution in term of rise time, settling time, maximum overshoot and steady state error condition of the system. Compared to conventional Ziegler – Nichols method, the proposed method was indeed more efficient and robust in improving the step response of the servo motor.

المستخلص

في هذه الأطروحة ، بطريقة الذكاء الاصطناعي يتم تقديم خوارزمية تجسيم سرب الجرعة (PSO) لتحديد المتغيرات المثلى المتسقة (PID) للتحكم لنظام الحركة المؤازر المستخدم في التحكم العددي للكمبيوتر (CNC). توضح هذه الرسالة بالتفصيل كيفية استخدام طريقة PSO للبحث بكفاءة عن معايير تحكم PID المثلى للمحرك المؤازر. للمساعدة في تقدير أداء وحدة التحكم PSO-PID المقترحة ، يتم تصميم النموذج باستخدام بيئة MATLAB.وينتج عن النهج المقترح حلًا أفضل من حيث وقت الارتفاع ووقت التسوية والحد الأقصى للخطأ وحالة الخطأ الثابتة في النظام. مقارنة بالطريقة التقليدية Ziegler - Nichols ، كانت الطريقة المقترحة أكثر فاعلية وقوة في تحسين استجابة خطوة المحرك المؤازر.

TABEL OFCONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACO Ant Colony Optimization

FL Fuzzy Logic

GA Genetic Algorithm

ISE Integral Square of Error

IAE Integral Absolute of Error

PID Proportional Integral Derivative

PSO Particle Swarm Optimization

PWM Pulse Width Modulation

TF Transfer function

ZN Ziegler Nichols

CHAPTER ONE

INTRODUCTION

CHAPTER ONE

INTRODUCTION

1.1 Background:

Computer Numerical Control (CNC) is a computer assisted process to control general-purpose machines according to the instructions generated by a processor from numeric instructions. There are many different types of CNC machine tools, which can be divided into two main groups: cutting machines and non-cutting machines. Cutting machines perform removal process to make a finished part. Examples of such machines are a milling machine and a turning machine. Noncutting machines apply force to a blank material to change the shape of the blank material. A good example of this type of a machine is a press machine. Also welding, painting, and cutting robot systems can be classified as CNC machine tools[1].

Servo driving mechanism is a system that transforms the commands form NC to a linear machine motion. It can consist of a motor and a power transmission device. Commands from the numerical control cause the motor to rotate a ball screw. The rotation of the ball screw is transformed into a linear movement of a nut, which is fixed to the table with the work piece. A servo driving mechanism controls the velocity and torque of the table via the servo driving device of each axis based on the velocity commands from the NC.

Encoder is a device that detects the angular position and sends the information to the control system. It is usually fixed into the shaft of the power-transmission or integrated in the motor. In order to control the velocity, it must be measured. Velocity can be measured by a sensor or calculated with the position control data from an encoder Encoders can be classified in two main categories: absolute- and incremental encoders. Absolute encoders give the actual angular position and incremental encoders detect changes in rotation[2].

Servo motor is an automatic device that uses error sensing feedback to correct the performance of a mechanism. The term correctly applies only to the systems where the feedback or error correction signals help to control mechanical position or other parameters. A common type of servo provides position control. Servos are commonly electrical or partially electronic in nature, using an electric motor as the primary means of creating mechanical force. Other types of servos use hydraulics, pneumatics, or magnetic principles. Usually, servos operate on the principle of negative feedback, where the control input is compared to the actual position of the mechanical system as measured by some sort of transducer at the output. Any difference between the actual and wanted values (error signal) I amplified and used to drive the system in the direction necessary to reduce or eliminate the error[3].

1.2 Problem statement:

PID controllers used in most applications to stabilize the system and get the required closed loop responses. This is due to its robust nature and wide operating range. In spite of this, When PID controller is used to

control servo motor used in CNC system some obstacles appears such as behaviors in terms of nonlinearity , time response, adjusting parameters based on online changes and lastly engineering goals such as cost and reliability. Therefore, an optimization algorithm is needed to find the optimal tuning parameters.

1.3 Proposed solution:

Using particle Swarm Optimization algorithm which is can automatically tune PID controller parameters during system run. Therefore, it can improve the speed behavior of the servo motor used in CNC machines. Moreover, it can enhance the characteristics of the engines, and makes the system more robustness.

1.4 Objectives:

- Optimize PID controller behavior using intelligent tuning method PSO.
- Improve time response parameters for the servo motor -used in CNC machines- speed response (overshoot, settling time rise time, and steady state error).
- Improve frequency response for the servo motor speed response. Performance evaluation for the system by comparing proposed tuning method with traditional methods.

1.5 Methodology:

PID controller will be used to control the servo motor speed. Firstly; PID will be tunes using traditional method. The second step tuning using PSO algorithm. The system will be tested under different condition and the result will be carried out with different scenarios. To simulate the proposed system MATLAB/SIMULINK will be used.

1.6 Scope of the work:

This thesis meanly focuses 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 servo motor for CNC machines.

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

LITERATURE REVIEW

CHAPTER TWO

LITERATURE REVIEW

2.1 Previous studies:

In the latest high-speed and high-acceleration NC machine tools, the structural vibration is one of the most critical factors to deteriorate the machine's contouring performance. Particularly on such a machine, the parameters in a CNC servo control system must be carefully tuned, since too high response of the latest CNC units often causes severe structural vibration. This paper presents a practical servo tuning method for high-speed machine tools to optimize its contouring accuracies. In order to reduce the structural vibration with the minimum sacrifice of control bandwidth, the tuning is based on iterative measurement and simulation of the machine's contouring performance. A case study shows that a proper tuning of servo parameters significantly reduces the structural vibration and improves the machine's overall contouring accuracy[4].

Based on the load torque ratio, load / rated speed ratio, the temperature rise ratio and load / motor inertia ratio of servo motor's optimization matching principle, servo motor selection mathematical model of transmission system in CNC machine tool is built, compared to traditional selection algorithm, which is using the actual duty cycle conditions to optimal matching and analyzing. Then motor models available to meet the maximum utilization of the servo system, achieve the purpose

of improving motor performance and reduce costs. In this paper, by taking servo motor selection of three-axis transmission system in a CNC machine tools for example, horizontal, inclined and vertical transmission system's parameters selection and calculation are analyzed in detail. With the engineering test and motor control software, measure data is achieved, the reasonableness of the method in the paper is proved[5].

Servo systems affect the performances of machining in accuracy and surface quality for high speed and precision machine tools. This study introduces an efficient servo tuning technique for Computer Numerical Control (CNC) feed drive systems using particle swarm optimization (PSO) algorithm by virtual machine tool approach. The proposed approach contained a system identification phase and a servo tuning phase based on the same bandwidth for all axes feed drive systems. The PSO algorithm was adopted to obtain the system parameters and maximize the corresponding bandwidth. An efficient two-step servo tuning method based on gain and phase margins was proposed for high speed and precision requirements. All feed drive systems controller gains were optimized simultaneously for synchronization. A remote system called Machine Dr. was established for servo tuning and monitoring. Simulation and experimental results were introduced to illustrate the effectiveness of the proposed approach[6].

In this study the basic idea is use the fuzzy logic controller with refer to Proportional Integral Derivative (PID) method. The choice of the fuzzy logic is based on its main feature; that its logic flow approaches real time situations more than most of the other known algorithms. The idea of perfection is to provide an even more smooth control to the AC servo motor

and to minimize deficiencies of the traditional Proportional Integral Derivative (PID) method[7].

2.2 Servo Motor:

Servo motor is an automatic device that uses error sensing feedback to correct the performance of a mechanism. The term correctly applies only to the systems where the feedback or error correction signals help to control mechanical position or other parameters. A common type of servo provides position control. Servos are commonly electrical or partially electronic in nature, using an electric motor as the primary means of creating mechanical force. Other types of servos use hydraulics, pneumatics, or magnetic principles. Usually, servos operate on the principle of negative feedback, where the control input is compared to the actual position of the mechanical system as measured by some sort of transducer at the output. Any difference between the actual and wanted values (error signal) is amplified and used to drive the system in the direction necessary to reduce or eliminate the error. Servomotors are available as AC or DC motors. Today, servo motor are used in automatic machine tools, satellite tracking antennas, remote control airplanes, automatic navigation systems on boats and planes, and antiaircraft gun control systems[8].

As the name suggests, a servo motor is a servomechanism. More specifically, it is a closed –loop servomechanism that uses position feedback to control its motion and final position. The input to its control is some signal, either analogue or digital, representing the position commanded for output shaft. The motor is paired with some type of encoder to provide position and speed feedback. In the simplest case, only the

position is measured. The measured position of output is compared to the command position, the external input to the controller. If the output position that required, an error signal is generated which then causes the motor to rotate in either direction, as needed to bring the output shaft to the appropriate position. As the position approach, the error signal reduces to zero and the motor stops. The very simplest servo motor use position-only sensing via a potentiometer and bang-bang control of their motor, the motor always rotates at full speed (or stopped).this motor not widely used in industrial motion control, but they form the basis of simple and cheap servos for radio-controlled models. More sophisticated servo motors measure both the position and also the speed of the output shaft. They may also control the speed of their motor, rather than always running at full speed. Both of these enhancements, usually in combination with a PID control algorithm, allow the servo motor to be brought to its commanded position more quickly and more precisely, with less overshooting[9].

2.3 PID controller:

A proportional–integral–derivative controller (PID controller) is a generic control of 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. 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. Simply, 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, a damper, or the power supplied to a heating element. In the absence of knowledge of the underlying process, a PID controller has historically been considered to be 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 set point, 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[10]. A PID controller is one of the most commonly used controllers because it is simple and robust. This controller is extremely popular because it can usually provide good closed loop response characteristics, can be tuned using relatively simple rules and easy to construct using either analogue or digital components. Figure (2.3) below illustrates the block diagram of PID controller.

2.3.1 P Controller:

In general it can be said that P controller cannot stabilize higher order processes. For the 1st order processes, meaning the processes with one - energy storage, a large increase in gain can be tolerated. Proportional controller can stabilize only 1st order unstable process. Changing controller gain K can change closed loop dynamics. A large controller gain will result in control system with:

a) Smaller steady state error, i.e. better reference following

b) Faster dynamics, i.e. broader signal frequency band of the closed loop system and larger sensitivity with respect to measuring noise.

c) Smaller amplitude and phase margin.

When P controller is used, large gain is needed to improve steady state error. Stable systems do not have problems when large gain is used. Such systems are systems with one-energy storage (1st order capacitive systems). If constant steady state error can be accepted with such processes, than P controller can be used. Small steady state errors can be accepted if sensor will give measured value with error or if importance of measured value is not too great anyway[11].

2.3.2 PD Controller:

D mode is used when prediction of the error can improve control or when it necessary to stabilize the system. From the frequency characteristic of D element it can be seen that it has phase lead of 90°.

Often derivative is not taken from the error signal but from the system output variable. This is done to avoid effects of the sudden change of the reference input that will cause sudden change in the value of error signal. Sudden change in error signal will cause sudden change in control output. To avoid that it is suitable to design D mode to be proportional to the change of the output variable.

PD controller is often used in control of moving objects such are flying and underwater vehicles, ships, rockets etc. One of the reason is in

stabilizing effect of PD controller on sudden changes in heading variable y(t). Often a "rate gyro" for velocity measurement is used as sensor of heading change of moving object.

2.3.3 PI Controller:

PI controller will eliminate forced oscillations and steady state error resulting in operation of on-off controller and P controller respectively.

However, introducing integral mode has a negative effect on speed of the response and overall stability of the system.

Thus, PI controller will not increase the speed of response. It can be expected since PI controller does not have means to predict what will happen with the error in near future. This problem can be solved by introducing derivative mode which has ability to predict what will happen with the error in near future and thus to decrease a reaction time of the controller.

PI controllers are very often used in industry, especially when speed of the response is not an issue. A control without D mode is used when:

a) Fast response of the system is not required.

b) Large disturbances and noise are present during operation of the process.

c) There is only one-energy storage in process (capacitive or inductive).

d) There are large transport delays in the system.

2.4 PID Controller Design

A PID controller is one of the most commonly used controllers because it is simple and robust. This controller is extremely popular because it can usually provide good closed loop response characteristics that can be tuned using relatively simple rules and easy to construct using either analogue or digital components.

PID controller has all the necessary dynamics: fast reaction on change of the controller input (D mode), increase in control signal to lead error towards zero (I mode) and suitable action inside control error area to eliminate oscillations (P mode).

Derivative mode improves stability of the system and enables increase in gain K and decrease in integral time constant Ti, which increases speed of the controller response.

PID controller is used when dealing with higher order capacitive processes (processes with more than one energy storage) when their dynamic is not similar to the dynamics of an integrator (like in many thermal processes). PID controller is often used in industry, but also in the control of mobile objects (course and trajectory following included) when stability and precise reference following are required. Conventional autopilot is for the most part PID type controllers[12]. The PID controller can be defined as equation (3.13) by the following relationship between controller input e(t) and the controller output V(t) that is applied to the motor armature.

$$V(t) = Kpe(t) + Ki\int_0^t e(t)\,dt + \frac{Kd}{dt}de(t) \qquad (3.13)$$

Where, Kp = proportional gain Ti = integral time Td= 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 (Kp) times the magnitude of the error plus the integral gain (Ki) times the integral of the error plus the derivative gain (Kd) times the derivative of the error.

2.5 Effects of Coefficients:

Table (2.1) 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.

Table 2. 1: Comparison of gain response of P,Pi,PID controllers

Parameter	Speed of response	Stability	Accuracy
Increasing Kp	Increases	deteriorate	improves
Increasing Ki	decreases	deteriorate	improves
Increasing Kd	increases	improves	No impact

2.6 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).

2.6.1 Ziegler-Nichols Tuning

The earliest known and most popular tuning methodology was proposed by Ziegler and Nichols (ZN) in 1942. They proposed the closed-loop (or ultimate sensitivity) method and the open-loop (or process reaction curve) method. The ZN tuning rules has a serious shortcoming in that it uses insufficient process information to determine the tuning parameters. This disadvantage leads to system performances that have poor robustness. The Ziegler-Nichols tuning method is based on the determination of processes inherent characteristics such as the process gain (Kp), process

time constant (PT) and process dead time (PL). These characteristics are used to determine the controller tuning parameters. Although the Ziegler-Nichols methods attempt to yield optimum settings, the only criterion stated is that the response has a decay ratio of quarter (see Figure 4.1). This is viewed as a shortcoming because a controller tuned with this criterion may not be at its optimal setting[13].

The closed-loop tuning method proposed by ZN requires the determination of the ultimate gain and ultimate period. The method can be interpreted as a technique of positioning one point on the Nyquist curve. This can be achieved by adjusting the controller gain (Kc) till the system undergoes sustained oscillations (at the ultimate gain or critical gain), whilst maintaining the integral time constant (iT) at infinity and the derivative time constant (dT) at zero. Consider Figure 4.2: the closed loop response is considered stable if there is no encirclement of the point (-1 + j0) by the Nyquist plot (Figure 4.2a) of the system (Ogata, 1970). For a proportional gain (Kc) = 2 the closed-loop response is stable and the Nyquist stability criterion is met (Figure 4.2b). For = 8 $c\ K$, sustained oscillations are produced since there is an encirclement of the point (-1 + j0) by the Nyquist locus. In both simulations, $i\ T$ and $d\ T$ =0 is used with a change only in the proportional gain $c\ K$ to move the process closer to the ultimate point[14].

By three parameters, namely the process static gain $p\ K$, the process time constant $p\ T$ and $p\ L$. These parameters are used to determine the controller's tuning parameters (see Table 2.1).

Table 2. 2:Ziegler-Nichols open-loop tuning parameter

Controller	K_c	T_i	T_d
P	$\frac{T_p}{L_p K_p}$	∞	0
PI	$0.9\frac{T_p}{L_p K_p}$	$3.33 L_p$	0
PID	$1.2\frac{T_p}{L_p K_p}$	$2L_p$	$0.5L_p$

2.6.2 Cohen-Coon tuning (Open-loop tuning)

The ZN method was designed for a process that cannot regulate itself. To account for self-regulation, Cohen-Coon (CC) introduced the self-regulation index or controllability ratio given by (4.3) (Cohen and Coon, 1953)

$$\varepsilon = \frac{L_p}{T_p}$$

With regards to (4.3), *p L* refers to the process dead time and *p T* denotes the process time constant. This method is based on a first-order-plus-dead-time (FOPDT) process models.

$$G_p(s) = \frac{K_p \exp(-L_p s)}{(T_p s + 1)}$$

A summary of the CC method is given in Table 2.2.

Table 2. 3: Cohen Coon tuning formula

Controller	K_c	T_i	T_d
P	$\frac{1}{K_p}\left[0.35+\frac{1}{\varepsilon}\right]$	∞	0
PI	$\frac{1}{K_p}\left[0.083+\frac{0.9}{\varepsilon}\right]$	$\left[\frac{3.3+0.31\varepsilon}{1+2.2\varepsilon}\right]T_p$	0
PID	$\frac{1}{K_p}\left[0.25+\frac{1.35}{\varepsilon}\right]$	$\left[\frac{2.5+0.46\varepsilon}{1+0.61\varepsilon}\right]T_p$	$\left[\frac{3.7}{1+0.19\varepsilon}\right]T_p$

2.7 Comparison between ZN and CC Tuning

A fundamental difference between the ZN and CC methods is as follows: The ZN method associates the integral and derivative constants completely with the process dead-time, whereas the CC method adjusts the integral and derivative time constants according to the particular relationship between the process dead time and the process time constant. For both methods, the controller gain is a function of this relationship. Since processes having different controllability ratios experience different dynamic behaviors, the Cohen- Coon method may perform better than the Ziegler-Nichols method. For example, for dead-time dominant processes i.e. processes having a large controllability ratio, the derivative time constant tends towards zero according to the Cohen-Coon tuning formulae. This is reasonable since the derivative action should not be used when the process contains large process time lag. The method does suffer from the decay ratio being too small. This results in closed-loop systems that are

characterized by low damping and high sensitivity. Furthermore, the tuning formula tends to produce a very oscillatory set-point change closed-loop response because it was derived to give a quarter wave decay ratio following a load disturbance response[15].

2.8 Software Methods:

In this section, the software methods means the optimization algorithms that can be used to tune the PID controller.

2.8.1 Ant Colony Optimization

In ACO artificial ants build solutions by traversing a problem space. Similar to real ants, they deposit artificial pheromone on the workspace in a manner that makes it possible for future ants to build better solutions. In real ant colonies the pheromone is used to find the shortest path to food. Using ACO, finite size colonies of artificial ants communicate with each other via artificial pheromones to find quality solutions to optimization problems. ACO has been applied to a wide range of optimization problems such as the traveling salesman problem, and routing and load balancing in packet switched networks[16].

2.8.2 Particle Swarm Optimization

The PSO approach utilizes a population based stochastic optimization algorithm proposed by Eberhart and Kennedy (1995). It was inspired from the computer simulation of the social behavior of bird flocking by Reynolds (1987). Reynolds used computer graphics to model complicated flocking behaviour of birds. He was mainly interested in simulating the flight

patterns of birds for visual computer simulation purposes, observing that the flock *appears* to be under central control. Reynolds proceeded to model his flocks using three simple rules, namely *collision avoidance*, *velocity matching* and *flock centering*.

Using these rules Reynolds showed how the behavior of each agent inside the flock can be modeled with simple vectors. This characteristic is one of the basic concepts of PSO. Boyd and Recharson (1985) examined the decision making process of human beings and developed the concept of individual learning and culture transmission. According to their examination, people utilize two important kinds of information in decision-making processes, namely:

Their own experience: They have tried the choices and know which state has been better so far, and they know how good it was and *Other* people's experiences: They have knowledge of how the other agents around them have performed. In other words, they know which choices their neighbours have found positive so far and how positive the best pattern of choice was. Each agent's decisions is based upon his own experience and other people's experience. This characteristic is another basic concept of PSO. Eberhart and Kennedy (1995) incorporated these ideas into the development of their PSO method and invented simple velocity and position algorithms that mimic natural swarm behaviour. In PSO, a set of randomly generated agents propagate in the design space towards the optimal solution over a number of iterations. Each agent has a memory of its best position and the swarm's best solution. PSO is similar to EC techniques in a sense that both approaches are population-based and each individual is evaluated according to a specified fitness function. The major

difference is that PSO is influenced by the simulation of social behaviour rather than the survival of the fittest[17]. Added to this, each individual benefits from its history and its interactions with its peers. PSO is also easy to implement and the fact that no gradient information is required makes it a good candidate for a wide variety of optimization problems. PSO has been successfully applied to solve a broad range of optimization problems ranging from Artificial Neural Network (ANN) training to reactive power and voltage control. The PSO method is also computationally less burdening in comparison to other EC techniques such as GA's[18].

2.8.3 Comparison between ACO and PSO:

PSO is based on the intelligence. It can be applied into both scientific research and engineering use, have no overlapping and mutation calculation. The search can be carried out by the speed of the particle. During the development of several generations, only the most optimist particle can transmit information onto the other particles, and the speed of the researching is very fast. The calculation in PSO is very simple, Compared to the other developing calculations, it occupies the bigger optimization ability and it can be completed easily. PSO adopts the real number code, and it is decided directly by the solution. The number of the dimension is equal to the constant of the solution.

In ACO the theoretical analysis is difficult and Sequences of random decisions (not independent). Probability distribution changes by iteration, Research is experimental rather than theoretical and Time to convergence uncertain.

CHAPTER THREE

SYSTEM DESIGN

CHAPTER THREE

SYSTEM DESIGN

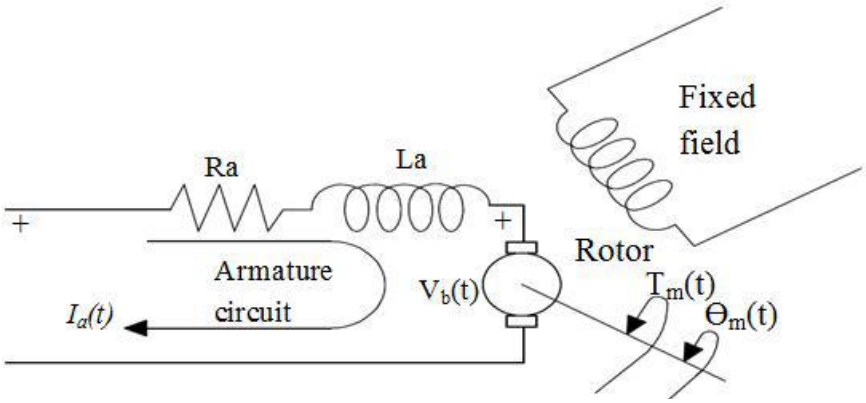
3.1 overview:

A servomotor is a packaged combination of several components: a motor, a gear train to reduce the many rotations of the motor to a higher torque rotation, a position encoder that identifies the position of the output shaft and an inbuilt control system. The input control signal to the servo indicates the desired output position. Any difference between the position commanded and the position of the encoder gives rise to an error signal that causes the motor and gear train to rotate until the encoder reflects a position matching that commanded. A simple low-cost servo of this type is widely used for radio-controlled model.

3.2 Servo Motor Modeling:

Servomotor is used for position or speed control in closed loop control systems.

The equivalent circuit diagram of servomotor is presented in Figure 3.1. The armature is modeled as a circuit with resistance Ra connected in series with an inductance ,La and a voltages source Vb(t) representing the back emf in the armature when the rotor rotates.



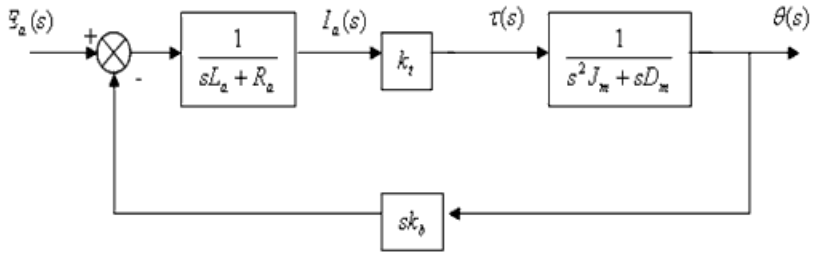


Figure 3.1: Servomotor system

Kirchhoff's voltage law is used to map the armature circuitry dynamic of the motor. Thus, assume the inductance La can be ignored, which in the case for servomotor. the supply voltage Ea(t) will be:

$$Ea(t)=Ia(t)Ra+ Vb(t) \qquad (3,1)$$

Since the current carrying armature is rotating in a magnetic field, its back electromotive force is proportional to speed. Vb(t) is the velocity of the conductor normal to the magnetic field.

$$Vb(t)=KB\ \omega\ (t) \tag{3.2}$$

The typical equivalent mechanical loading on a motor, that connected to the motor shaft including total moment of inertia jm and total viscous friction.

Assume that T(t) is the torque developed by the motor.

$$T(t)=Jm\alpha(t)+B\alpha(t) \tag{3.3}$$

The developed motor output torque for the servo motor can be given by:

$$T(t)=KTIa(t) \tag{3.4}$$

By using Laplace transforms on the equation (3.1), (3.2), (3.3) and (3.4) and

neglecting initial condition we have:

$$Ea(s)=RaIa(s)+Vb(s) \tag{3.5}$$

$$V_b(s)=K_BS\ \theta_m(s) \tag{3.6}$$

$$T(s)=J_mS2\ \theta_m\ (s)+BS\ \theta_m\ (s) \tag{3.7}$$

$$T(s)=KTIa(s) \tag{3.8}$$

Substitute Equation (3.8) into Equation (3.7), we have:

$KTIa(s)=Jms2\ \theta_m\ (s)+Bs\ \theta_m\ (s)$ (3.9)

Equation (3.5) is rearranged to obtain:

$Ia(s)=Ea(s)-vb(s)/Ra$

Substitute equation (3.10) into Equation (3.9), we get:

$K_T\ [Ea(s)-vb(s)/Ra]= J_mS2\ \theta_m\ (s)+Bs\ \theta_m\ (s)$

From equation (3.11), the transfer function between the input voltage Ea(s) and the output $\theta_m\ (s)$ can be obtained as:

$$\frac{\theta m(s)}{Ea(s)}=\frac{KT}{JmRaS^2+(BRa+KTKB)s}$$

The parameters for used servo motor are:

KT (N.m/A) = 0.121

KB [V/(rad/s)] = 0.121

Ra (Ω) = 2.23

B [N.m/(rad/s)] = 0.0000708

Jm (kg.m2) = 0.00006286

B [N.m/(rad/s)] = 0.0000708

Substitute there parameters in Equation (3.12), the transfer function becomes as follow:

$$\frac{\theta m(s)}{Ea(s)} = \frac{3.839}{0.004S^2 + 0.34s + 1}$$

3.3 PID controller:

A proportional-integral-derivative controller (PID controller) is basically a generic control loop feedback mechanism widely used in industrial control systems. A PID controller calculates an "error" value as the difference between a measured plant variable and a desired set-point. The controller attempts to minimize the error by adjusting the process control inputs. Fig.3.2 shows a basic structure of a closed loop controller.

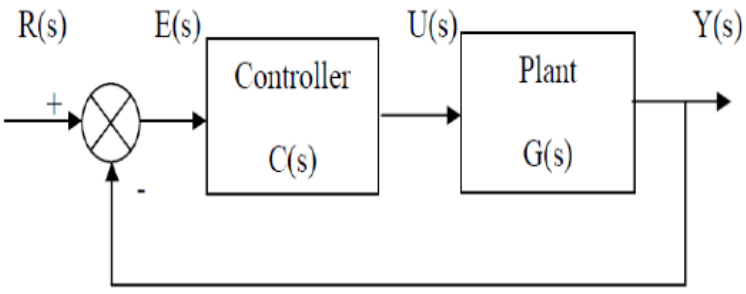


Figure 3. 2 Classical Controller

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.

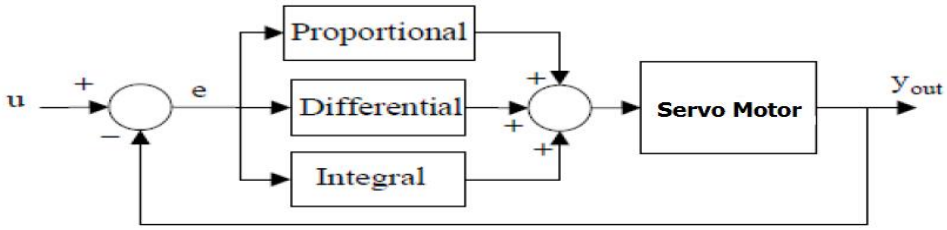


Figure 3. 3:block diagram of PID controller with servo motor

In this thesis the PID controller is tuned using MATLAB software. Firstly, the PID tuned using Z_N to find the step response of P, PI, PD and PID. The tuning is done using a comparator found in the program which can generate the step response of each controller.

The output of each controller tuned using Z_N is compared the result of the PSO output in chapter four.

3.4 Particle Swarm Optimizations

The algorithm proposed by Eberhart and Kennedy (1995) uses a 1-D approach for searching within the solution space. For this study the PSO algorithm will be applied to a 2-D or 3-D solution space in search of optimal tuning parameters for PI, PD and PID control[19]. Consider position *i s* ,n. of the *i*-the particle as it traverses a *n*-dimensional search space:

The previous best position for this *i*-th particle is recorded and represented as *pbesti,n* .

The best performing particle among the swarm population is denoted as g*besti,n* and the velocity of each particle within the *n*-th dimension is represented as *i n v* , . The new velocity and position for each particle can be calculated from its current velocity and distance with (6.1) and (6.2), respectively:

$$V_{i,n}^{(k+1)} = X\left[V_{i,n}^{k} + C1rand1 * \left(Pbest_{i,n} - S_{i,n}^{k}\right) + C2rand2 * \left(gbest_{i,n} - S_{i,n}^{k}\right)\right]$$

$$S_{i,n}^{(k+1)} = S_{i,n}^{k} + V_{i,n}^{(k+1)}$$

With regards to (6.2) and (6.3):

i = number of agents 1,2,….,p;

n = dimension 1,2,3;

(1),$k+i$ n v = velocity of agent ‘ i ’ at iteration (k + 1) for n-dimension;

c = constriction factor;

$V^{k}_{i,n}$= velocity of agent $i$at current iteration k for n dimension;

$C1$ = cognitive acceleration constants (self confidence);

C2 = social acceleration constant (swarm confidence);

$Rand_{1,2}$ = random number between 0 and 1;

$pbest_{I,n}$= personal best of agent i for n dimension;

$gbest_{I,n}$ = global best of the population for n dimension;

$S_{i,n}^{k}$ = current position of agent $i$at iteration k for n dimension;

$S_{i,n}^{(k+1)}$ = position of agent i at iteration (k + 1) for n dimension and;

p = number of particles in the population.

For PI, PD and PID control $n = 2, 3$ respectively. All other variables have the same meanings[20].

3.5 Scheduling PSO for PID Controller Parameters:

In this work, the PSO is used to find the optimal PID parameter’s values that is used to control the servo motor. The structure of the PID controller with PSO algorithms is shown in Fig. .

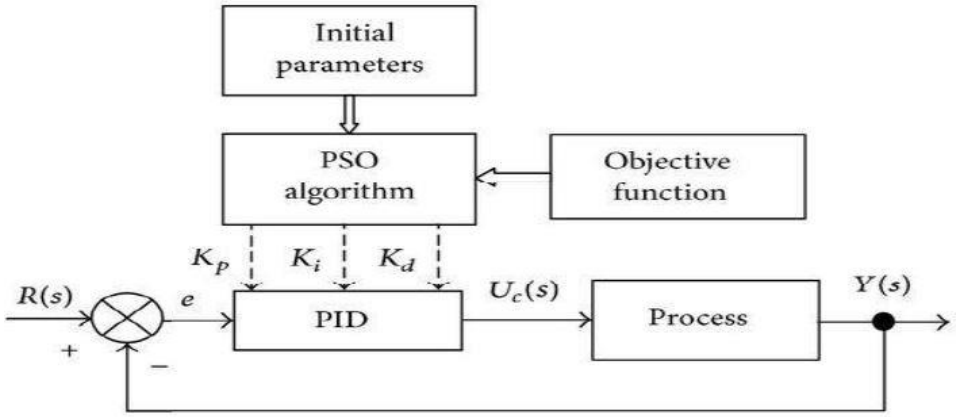


Figure 3. 4:PID controller with PSO algorithm

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.

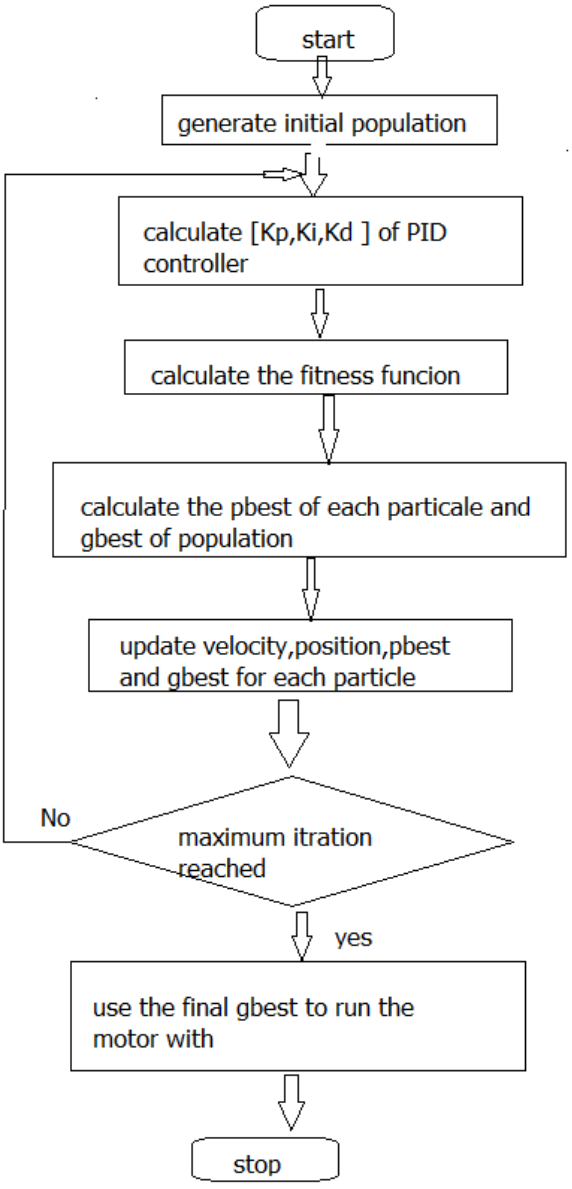
(c) For every iteration, compare each particle's fitness with its previous best fitness ($pbest$) obtained. If the current value is better than $pbest$, then set $pbest$ equal to the current value and the $pbest$ location equal to the current location in the d-dimensional space.

(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.

3.6 System flow chart:



CHAPTER FOUR

SIMULATION RESULT

CHAPTER FOUR

SIMULATION RESULT

4.1 Overview:

MATLAB is a widely used in all areas of applied mathematics , in education and research at universities and in the industry. MATLAB stand for MATrixLABoratory and the software particularly useful for linear algebra but MATLAB is also a great tool for solving algebraic and differential equations and for numerical integration. MATLAB has power full graphic tools and can produce nice pictures in both 2D and 3D. It is also a programming languages for writing mathematical programs. MATLAB also has some tools boxes useful for signal processing, image processing, optimization, Simulink , etc . In Simulink it is very straightforward to represent and then simulate a mathematical model representing a physical system. Models are represented graphically in Simulink as block diagrams. A wide array of blocks are available to the user in provided libraries for representing various phenomena and models in a range of formats. One of the primary advantages of employing Simulink (and simulation in general) for the analysis of dynamic systems that it allows us to quickly analyze the response of complicated systems that may be prohibitively difficult to analyze analytically. figure (4.1) shows the SIMULINK block diagram of servo motor without controller.

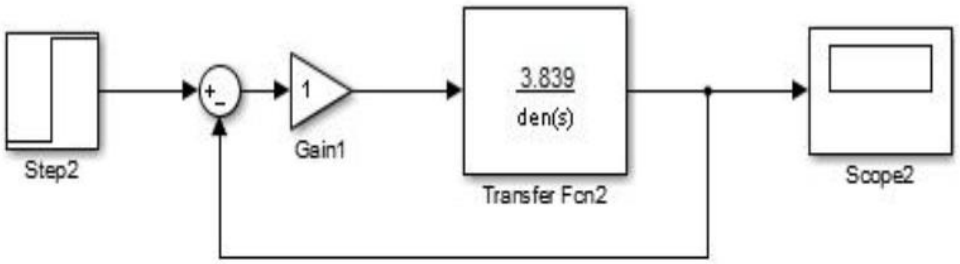


Figure 4. 1The Simulink block diagram for servo motor

The step response of servo motor without controller is shown in figure(4.2). Where the step response is less than one, there for a controller is needed.

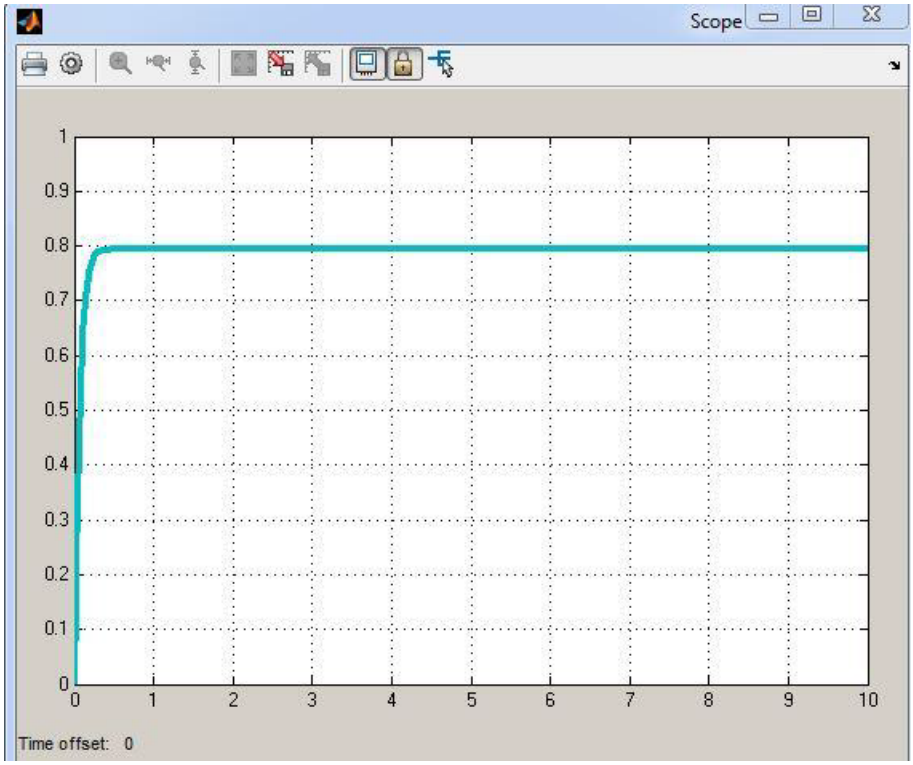


Figure 4. 2: the step response of DC servo motor without controller

4.2 Servo motor and PID with Manual Tuning:

The following figures shows the effect of PI, PD and PID using zeglar-Necolas manual tuning.

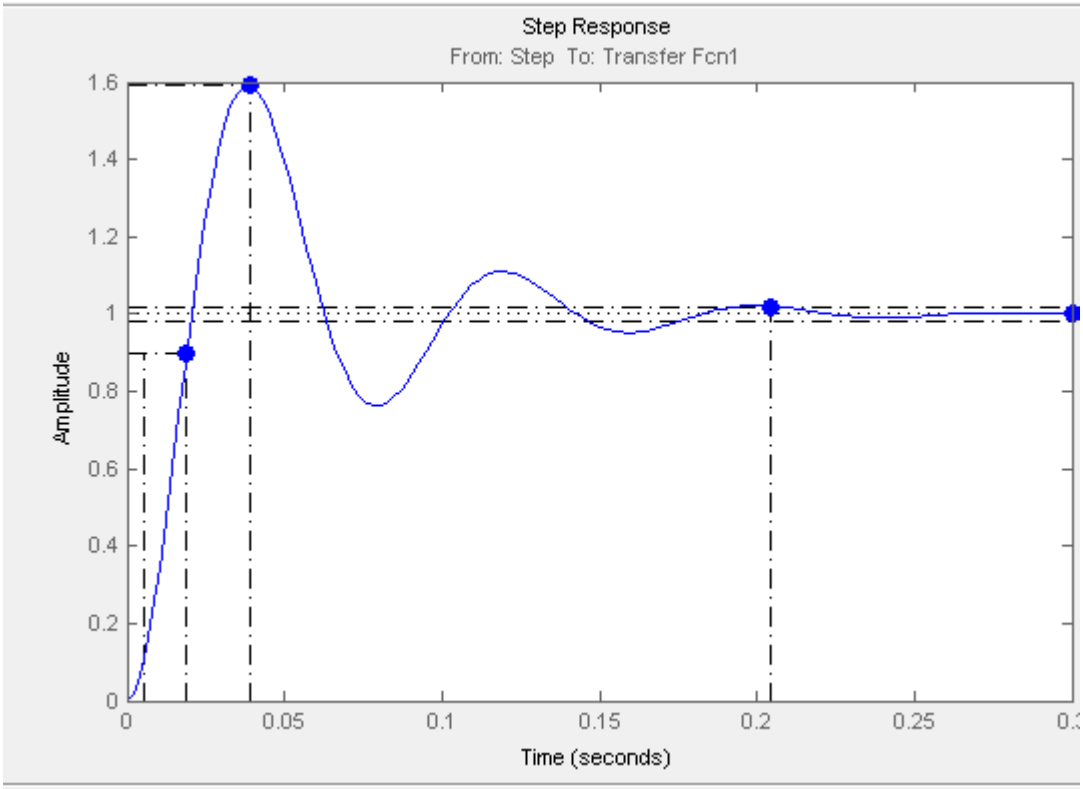


Figure 4. 3: step response of PI controller

In the above figure the value of Kp=11.622 and Ki=409.03439 calculated using Z-N, as noticed the step response has a good rise time, but the value of settling time and overshoot are not good compared to the PID response.

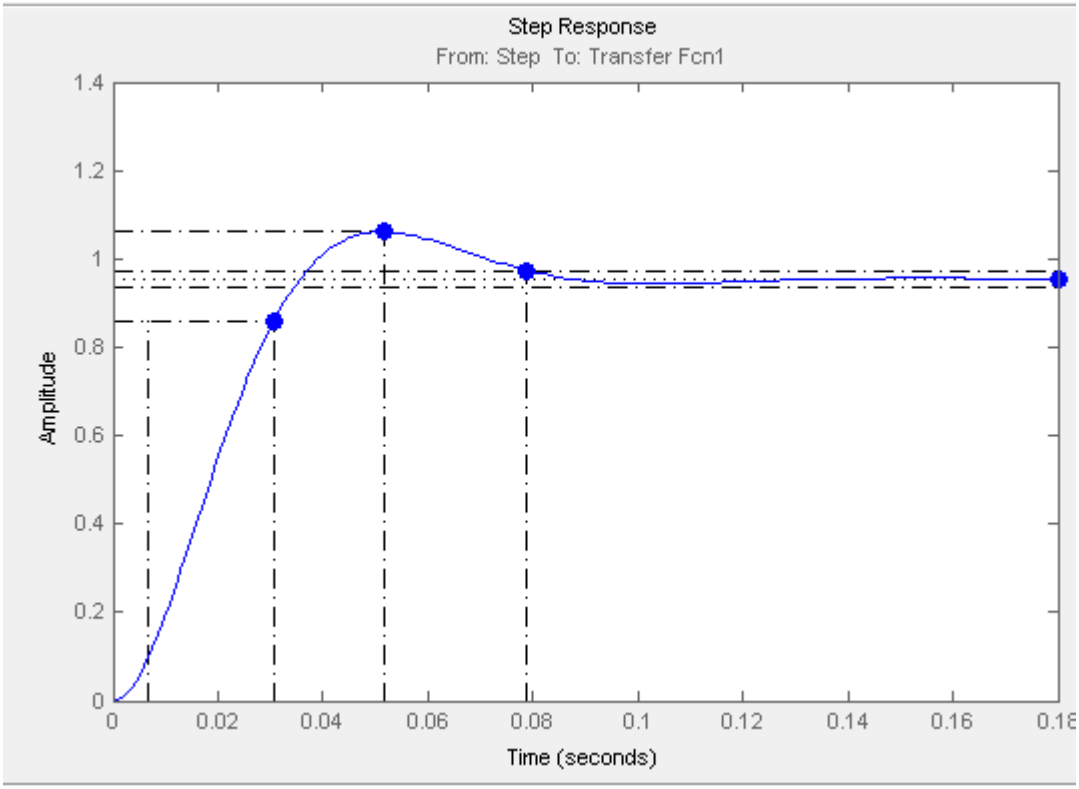


Figure 4. 4:step response of PD controller

In the above figure, the value of Kp=27.9380 and Kd=0.16194 calculated using Z-N in MATLAB, as noticed the step response of PD controller has better settling time and overshoot compared to PI and PID controllers.

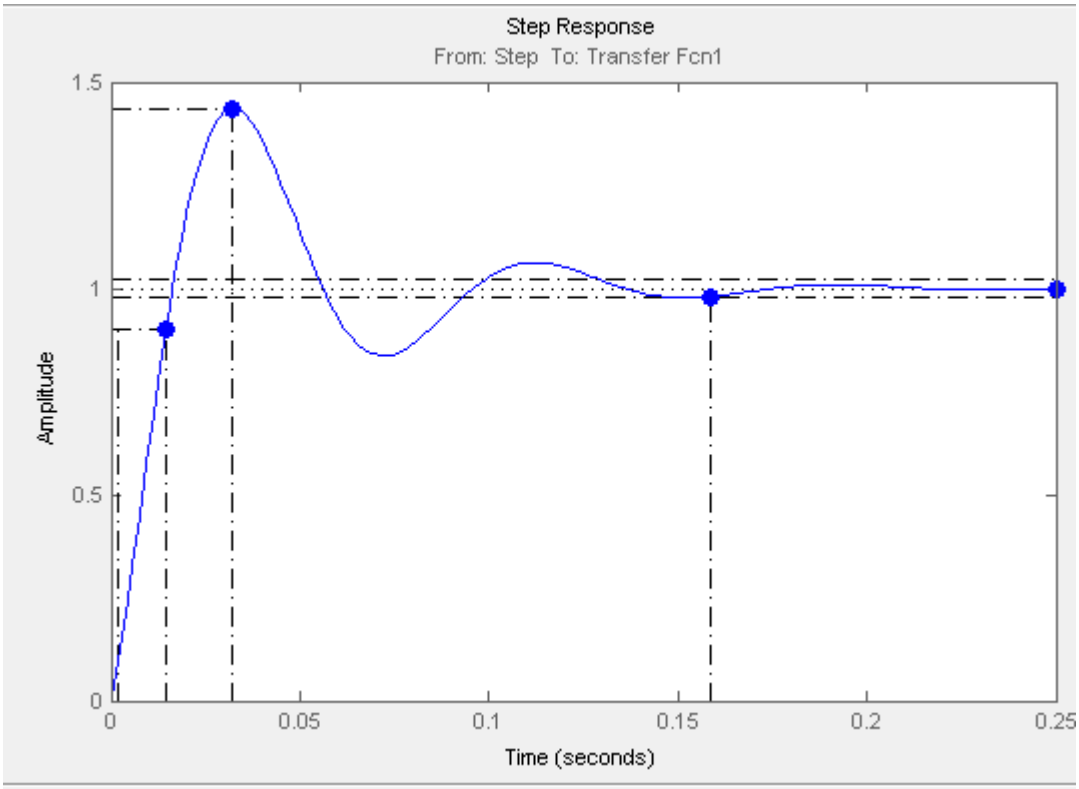


Figure 4. 5: step response of PID controller

In the above figure, the value of Kp=15.4966 Ki=818.0687 and Kd=0.07338 calculated using Z-N in MATLAB, as noticed the step response of PID has the best rise time value compared to other two controllers.

Table (4.1) 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.

Table 4. 1comparison between, PI, PD and PID maximum rise time, settling time and overshoot

	Rising time(Sec)	Settling time(Sec)	Overshoot (%)	Overshoot time (sec)
PI	0.0138	0.204	59%	0.0392
PD	0.024	0.0788	11.2%	0.0518
PID	0.0122	0.155	43.4%	0.115

4.3 Simulation result discussion:

The model of servo motor and the optimal control of speed were numerically simulated using a state space model and Matlab/Simulink software for a separated exited servo motor with the following parameters

KT (N.m/A) = 0.121

KB [V/(rad/s)] = 0.121

Ra (Ω) = 2.23

B [N.m/(rad/s)] = 0.0000708

Jm (kg.m2) = 0.00006286

B [N.m/(rad/s)] = 0.0000708

Substitute there parameters in Equation (3.12), the transfer function becomes as follow:

$$\frac{\theta m(s)}{Ea(s)} = \frac{3.839}{0.004S^2 + 0.34s + 1}$$

The simulation procedure may be summarized as follows: • First input the servo motor data, • Write the differential equations for the model then get the state space representation • Get the closed loop step response • Finally performing the performance of PID controller by Ziegler Nichols method and PID controller by using PSO

Figure (4.3) shows the SIMULINK block diagram of position control of servo motor using PID controller .the PID controller gain which was the output of the PSO algorithm as Kp= 79.1195 ,Ki= 43.1362 ,Kd= 43.6804. Figure 4.4 shows the step response of PID controller.

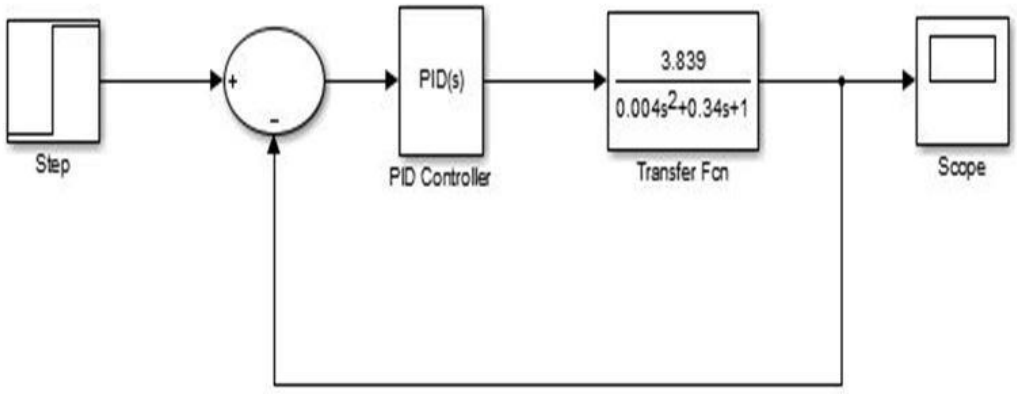


Figure 4. 6: the Simulink block diagram of PID controller

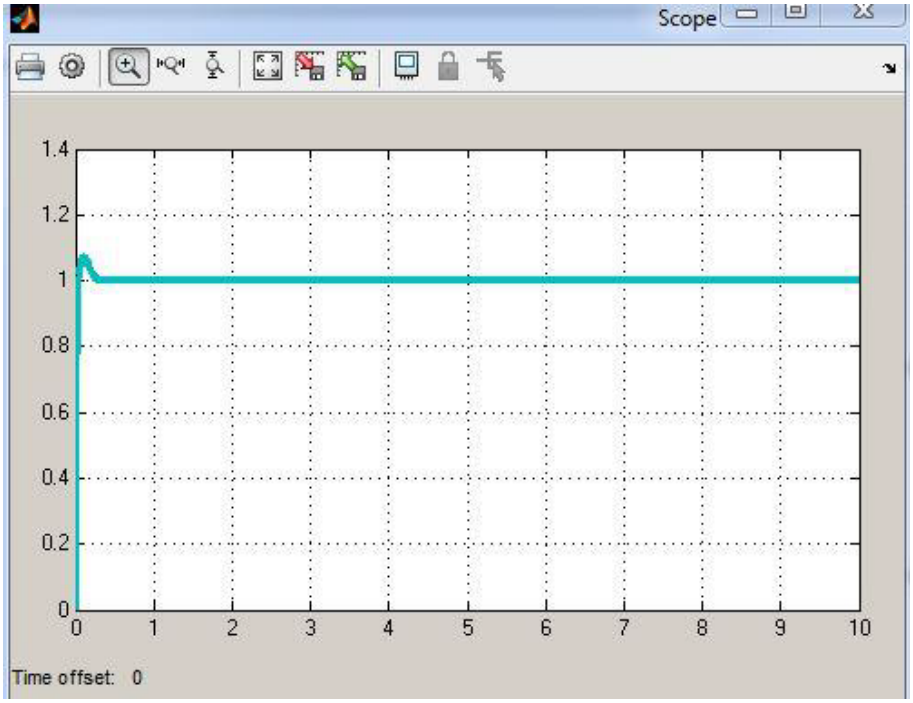


Figure 4. 7: the step response of servo motor with controller

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

4.4 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 servo 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

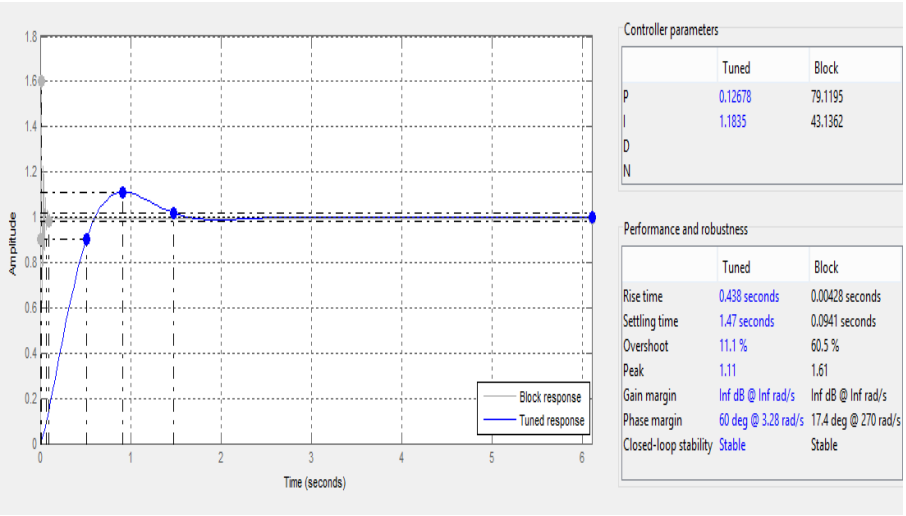


Figure 4. 8:step response of PSO-PI controller

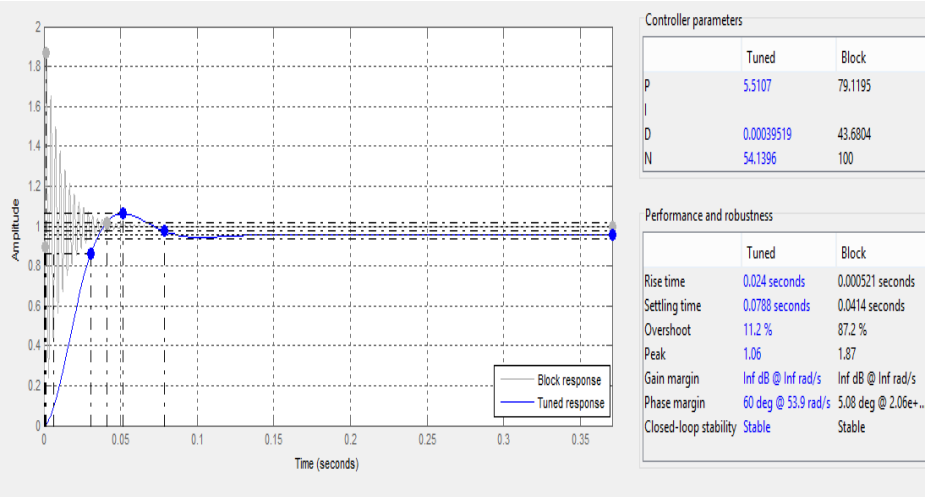


Figure 4. 9: step response of PSO-PD controller

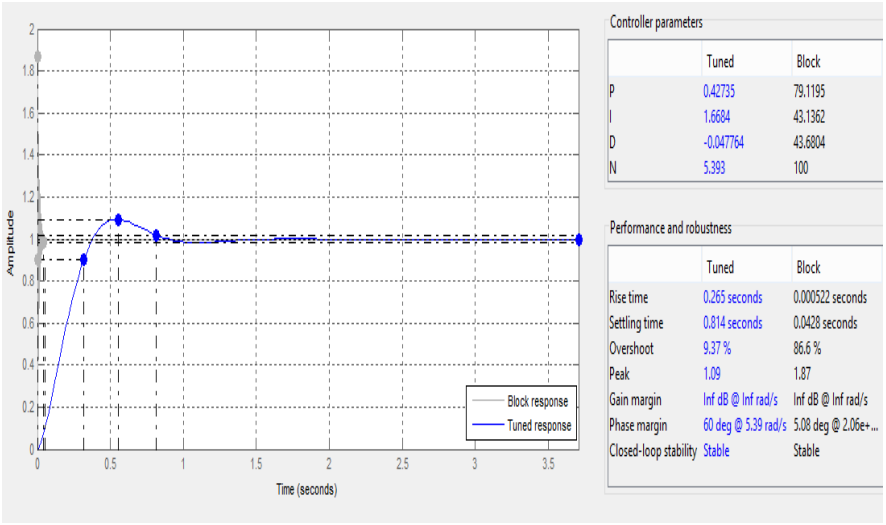


Figure 4. 10:step response of PSO-PID controller

The results of the algorithm on the PI, PD and PID controllers has shown a very good effect on the overall outputs , the comparison between the out but is shown in table (4.2) states that the PID controller has the best result.

Table 4. 2: comparison between, PSO-PI, PD and PID maximum rise time, settling time and overshoot

	Rise time(Sec)	**Settling time(sec)**	**Overshoot (%)**	**Overshoot time (sec)**
PI	0.00428	0.0941	60.5%	0.012
PD	0.000521	0.0414	87.2%	0.00152
PID	0.000522	0.0428	86.6%	0.00152

4.5 Comparison between Z_N and PSO PID:

Table 4. 3:comparison between, PSO- PID and Z-N maximum rise time, settling time and overshoot

PID parameters	Rise time(sec)	Settling time(sec)	Overshoot(%)	Overshoot time (sec)
Z-N	0.0122	0.155	43.4	0.0326
PSO	0.00522	0.0428	86.6	0.00152

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 servo 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 settling time and rise time. Although the overshoot of the PSO seems to have larger percentage than Z-N, but the overshoot time is very small that can be neglected. So PSO has proved a better result in term of overshoot as well.

CHAPTER FIVE

CONCLUTION AND RECOMMENDATIONS

CHAPTER FIVE

CONCLUTION AND RECOMMENDATIONS

5.1 Conclusions

In this work, a PSO method is used to determine PID controller parameters automatically through simulation of servo 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's parameters. 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 the motor.

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.
- Increase the number of iterations of the algorithm for more optimal values.
- Applying another optimization techniques and observe the difference.
- Applying real CNC parameters for better observation.

References:

[1] **C.-Y. L. a. C.-H. Lee, "Remote Servo Tuning System for Multi-Axis CNC Machine Tools Using a Virtual Machine Tool Approach," *Applied Sciences,* vol. 7, p. 776, 2017.**

[2] **R. Ranjan, "Automatic metal sheet cutting machine," *International Journal of Advanced Engineering Applications,* August 2013.**

[3] **M. Sahakangas, "Planning of an electric system for a small CNC machine," *SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES,* Spring 2015.**

[4] **S. I. K. Lee1, A. Matsubara1, Y. Kakino1, Y. Suzuki2, and S. A. J. Braasch4, "A servo parameter tuning method for high-speed NC machine tools based on contouring error measurement," *Journal of the Japan Society for Precision Engineering,* Jun,2010.**

[5] **Y. C. Wang, X. X. Yu, and L. Wu, "Selection and Analysis of Servomotor for Three-Axis Transmission System in CNC Machine Tool," *Advanced Materials Research,* vol. 760-762, pp. 1148-1153, 2013.**

[6] **R.-J. Wai, "Real-Time PID Control Strategy for Maglev Transportation System via Particle Swarm Optimization," *IEEE,* 2, FEBRUARY 2011.**

[7] **K.-Y. Cheng, "Fuzzy Optimization Techniques Applied to the Design of a Digital PMSM Servo Drive," *IEEE TRANSACTIONS ON POWER ELECTRONICS,* NOV 2011.**

[8] **H. J. Fredrik Roos, Jan Wikander, "Optimal selection of motor and gearhead in mechatronic applications[J],"*], Mechatronics, vol.16, pp.63–72,* 2006.**

[9] **I. I. Hubinski P., Jurisica L, " Elimination of residual oscillation in electromechanical systems containing pendulum," *on Electrical Drives and Power Electronics,* october 2001.**

[10] **D. H. C. KIM, J. H, "Retraction of A Biologically Inspired Intelligent PID Controller Tuning for AVR Systems," *Int. J. Control Autom. Syst. International Journal of Control, Automation and Systems,,* 2001.**

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

[12] **J. T. Z. a. T. Y. A. C. W. Wang, "Survey of Advanced PID Parameter Tuning Methods," *Acta Auto-matica, Vol. 26, No. 3,* 2000.**

[13] **B. a. L. Kristiansson, B, "Robust and optimal tuning of PID controllers," *IEE Proc. Control theory and application, V 149,* 2002.**

[14] **A. A. Aly, "odeling and Control of an Elec-tro-Hydraulic Servo Motor Applying Velocity Feedback Control Strategy," *International Mechanical Engineering Conference,* 2004.**

[15] **M. H. Moradi, "New Techniques for PID Controller De-sign," *IEEE Conference on Control Applications, Vol. 2,* 2003.**

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

[17] **X. LI, WANG, Y., LI, N., HAN, M., TANG, Y. & LIU, F, "Optimal fractional order PID controller design for automatic voltage regulator system based on reference model using particle swarm optimization.," *nt. J. Mach. Learn. & Cyber. International Journal of Machine Learning and Cybernetics, 8, 1595- 1605,* 20011.**

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

[19] **A. Myrtellari, Marango, P. and Gjonaj, M., "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,,* 2016.**

[20] **S. M. GIRIRAJKUMAR, KUMAR, A. A. & ANANTHARAMAN, N, " Tuning of a PID Controller for a Real Time Industrial Process using Particle Swarm Optimization," *IJCA International Journal of Computer Application,* 2010.**

Appendix: MATLAB Code

PSO-PID code

```
tic
clc
clear all
close all
rng default
LB=[0 0 0]; %lower bounds of variables
UB=[100 100 100]; %upper bounds of variables
% pso parameters values
m=3; % number of variables
%n=100; % population size
wmax=0.8; % inertia weight
wmin=0.3; % inertia weight
c1=1.5; % acceleration factor
c2=1.5; % acceleration factor
desired = 1;       % desired output, or reference point
desired1 = 1;       % desired output, or reference point

feed1 =0.001 ;         % can be replaced with damping coefficient B or (B/Mass)
feed2 = 0.001;         % can be replaced with spring coefficient K or (K/Mass)
B = feed1;K = feed2;
Kp = 1;            % proportional term Kp
Ki = 0.01;          % Integral term Ki
Kd = 0.01;          % derivative term Kd
dt = 0.01;         % sampling time         % total simulation time in seconds
Time = 10;         % total simulation time in seconds
n = round(Time/dt);
% pso main program----------------------------------------------------start
maxite=1000; % set maximum number of iteration
for run=1:100
run
% pso initialization---------------------------------------------start
for i=1:n
for j=1:m
x0(i,j)=LB(j)+rand()*(UB(j)-LB(j));
end
end
x=x0; % initial population
v=0.1*x0; % initial velocity
%%for i=1:n
```

```
%f0(i,1)=ofun(x0(i,:));
%end

% pre-assign all the arrays to optimize simulation time
Prop1(n+1,3) = 0; Der1(n+1,3) = 0; Int1(n+1,3) = 0; I1(n+1,3) = 0;
PID1(n+1,3) = 0;
FeedBack1(n+1,3) = 0;
Output1(n+1,3) = 0;
Error1(n+1,3) = 0;
state12(n+1,3) = 0; STATE12(n+1,3) = 0;
state22(n+1,3) = 0; STATE22(n+1,3) = 0;
for i = 1:n
   Error1(i+1,3) = desired1 - FeedBack1(i,3); % error entering the PID controller

   Prop1(i+1,3) = Error1(i+1,3);% error of proportional term
   Der1(i+1,3)  = (Error1(i+1,3) - Error1(i,3))/dt; % derivative of the error
   Int1(i+1,3)  = (Error1(i+1,3) + Error1(i,3))*dt/2; % integration of the error
   I1(i+1,3)    = sum(Int1(:,3)); % the sum of the integration of the error

   PID1(i+1,3)  = Kp*Prop1(i,3) + Ki*I1(i+1,3)+ Kd*Der1(i,3); % the three PID terms


end
[fmin0,index0]=min(PID1);
pbest=x0; % initial pbest
gbest=x0(index0,:); % initial gbest
% pso initialization----------------------------------------------------end
% pso algorithm---------------------------------------------------------start
ite=1;
tolerance=1;
%while ite<=maxite && tolerance>10^-12
w=wmax-(wmax-wmin)*ite/maxite; % update inertial weight
% pso velocity updates
for i=1:n
for j=1:m
v(i,j)=w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))...
+c2*rand()*(gbest(1,j)-x(i,j));
end
end
% pso position update
for i=1:n
for j=1:m
x(i,j)=x(i,j)+v(i,j);
end
end
% handling boundary violations
for i=1:n
```

```
for j=1:m
if x(i,j)<LB(j)
x(i,j)=LB(j);
elseif x(i,j)>UB(j)
x(i,j)=UB(j);
end
end
end
% evaluating fitness
Prop(n+1,3) = 0; Der(n+1,3) = 0; Int(n+1,3) = 0; I(n+1,3) = 0;
PID(n+1,3) = 0;
FeedBack(n+1,3) = 0;
Output(n+1,3) = 0;
Error(n+1,3) = 0;
state1(n+1,3) = 0; STATE1(n+1,3) = 0;
state2(n+1,3) = 0; STATE2(n+1,3) = 0;
for i=1:n
 Error(i+1,3) = desired - FeedBack(i,3); % error entering the PID controller

    Prop(i+1,3) = Error(i+1,3);% error of proportional term
    Der(i+1,3)  = (Error(i+1,3) - Error(i,3))/dt; % derivative of the error
    Int(i+1,3)  = (Error(i+1,3) + Error(i,3))*dt/2; % integration of the error
    I(i+1,3)    = sum(Int(:,3)); % the sum of the integration of the error

    PID(i+1,3)  = Kp*Prop(i,3) + Ki*I(i+1,3)+ Kd*Der(i,3); % the three PID terms

end
% updating pbest and fitness
for i=1:n
if PID(i,:)<PID1(i,:)
   pbest(i,:)=x(i,:);
PID1(i,:)=PID(i,:);
end
end
[fmin,index]=min(PID1(:,1)); % finding out the best particle
ffmin(ite,run)=fmin; % storing best fitness
ffite(run)=ite; % storing iteration count
% updating gbest and best fitness
if fmin<fmin0
gbest=pbest(index,:);
fmin0=fmin;
end
% calculating tolerance
if ite>100;
tolerance=abs(ffmin(ite-100,run)-fmin0);
end
% displaying iterative results
```

```
if ite==1
disp(sprintf('Iteration Best particle Objective fun'));
end
disp(sprintf('%8g %8g %8.4f',ite,index,fmin0));
ite=ite+1;
end
% pso algorithm----------------------------------------------------end
gbest;
            % start timer to calculate CPU time
Prop(n+1,3) = 0; Der(n+1,3) = 0; Int(n+1,3) = 0; I(n+1,3) = 0;
PID1(n+1,3) = 0;
FeedBack(n+1,3) = 0;
Output(n+1,3) = 0;
Error(n+1,3) = 0;
state1(n+1,3) = 0; STATE1(n+1,3) = 0;
state2(n+1,3) = 0; STATE2(n+1,3) = 0;
for i = 1:n
      Error(i+1,3) = desired - FeedBack(i,3); % error entering the PID controller

   Prop(i+1,3) = Error(i+1,3);% error of proportional term
   Der(i+1,3)  = (Error(i+1,3) - Error(i,3))/dt; % derivative of the error
   Int(i+1,3)  = (Error(i+1,3) + Error(i,3))*dt/2; % integration of the error
   I(i+1,3)    = sum(Int(:,3)); % the sum of the integration of the error

   PID(i+1,3)  = Kp*Prop(i,3) + Ki*I(i+1,3)+ Kd*Der(i,3); % the three PID terms



   %% You can replace the follwoing five lines with your system/hardware/model

end
fff(run)=PID(run);
rgbest=gbest;
disp(sprintf('-----------------------------------'));
%end
% pso main program----------------------------------------------------end
disp(sprintf('\n'));
disp(sprintf('************************************************************'));
disp(sprintf('Final Results-------------------------'));
[bestfun,bestrun]=min(fff)
best_variables=rgbest(bestrun,:)
disp(sprintf('************************************************************'));
% PSO convergence characteristic
```