

**INTEGRATION OF PARTICLE SWARM OPTIMIZATION (PSO)
TECHNIQUE INTO DC MOTOR CONTROL**

By

**NATHASYA NADJWA BINTI NORDIN
(10887)**

FINAL DISSERTATION

**Submitted to the Electrical & Electronics Engineering Programme
in Partial Fulfilment of the Requirements
for the Degree
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)**

**Universiti Teknologi Petronas
Bandar Seri Iskandar
31750 Tronoh
Perak DarulRidzuan**

**© Copyright 2011
by
Nathasya Nadjwa binti Nordin, 2011**

CERTIFICATION OF APPROVAL

Integration of Particle Swarm Optimization (PSO) Technique into Dc Motor Control

by

Nathasya Nadjwa binti Nordin

A project dissertation submitted to the
Electrical & Electronics Engineering Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

Approved:




AP Dr. Irraivan Elamvazuthi
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
TRONOH, PERAK

September 2011

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



Nathasya Nadjwa binti Nordin

ABSTRACT

Particle Swarm Optimization (PSO), an artificial method to determine the optimal proportional – integral - derivative (PID) controller parameters to be integrated into a brushed DC motor is presented. Particle Swarm Optimization (PSO), developed by Eberhart and Kennedy in 1995 was inspired by swarming patterns occurring in nature such as flocking birds. It was observed that each individual exchanges previous experience, hence knowledge of the “*best position*” attained by an individual becomes globally known. In the study, the problem of identifying the PID controller parameters is considered as an optimization problem. An attempt has been made to determine the PID parameters employing the PSO technique. This technique is used to improve the step response of a second order system. The step response of the given system is defined in rise time, settling time and peak overshoot. The best parameters to be used for PSO that can optimize the performance of a DC Motor (e.g.: population size, acceleration constant and inertia weight factor) is evaluated. First chapter discusses the types of DC motor available in industry nowadays and the origination of Particle Swarm Optimization technique itself. Next, the following chapter continues with the implementation of DC motor control and the tuning available that has been researched before. The usage of Particle Swarm Optimization technique is briefly explained which comprises the 6-steps of selection process. For this study, the software used is MATLAB/Simulink, where the implementation of the chosen DC motor model is represented and Particle Swarm Optimization is integrated into the PID controller of the motor, to observe the performance of chosen parameters. The results of PID controller tuning and also the results for the implementation of PSO based PID controller is presented on the Result & Discussion chapter. Comparison then is made and discussed to see whether the results are as expected. Lastly, recommendation and conclusion pertaining to the completion of this project is presented.

ACKNOWLEDGEMENTS

First and foremost, the author would like to express appreciation to the project supervisor, Associate Professor Dr. Irraivan Elamvazuthi for his valuable input and guidance throughout the completion of this project.

The author would also like to express gratitude to Electrical and Electronics Engineering Department of Universiti Teknologi PETRONAS (UTP) for providing this chance to undertake this remarkable final year project.

Next, the author would like to thank as well the Final Year Project (FYP) committee for arranging several of exhibitions as support and knowledge to assist the students. Thanks to all lecturers and colleges from UTP who had provided untiring guidance and help throughout the period of the project.

Finally, the author would like to apologize if any party was inadvertently excluded from being mentioned above and the author would like to thank all parties that were involved in making this project a success. Thank you.

Contents

ABSTRACT	iv
ACKNOWLEDGEMENTS	v
CHAPTER 1	1
Introduction	1
1.1 Background of Study	1
1.1.1 DC Motor.....	1
1.1.2 PID Controller.....	2
1.2 Problem Statement.....	3
1.3 Objectives.....	3
The objectives of this project include:	3
1.4 Scope of Study.....	3
CHAPTER 2	5
Literature Review	5
2.1 DC Motor Control.....	5
2.2 Evolutionary Computation and Swarm Intelligence Paradigms	6
2.2.1 Evolutionary Computation.....	7
2.2.2 Swarm Intelligence.....	8
2.3 Particle Swarm Optimization	8
CHAPTER 3	11
Methodology.....	11
3.1 Flowchart of PSO.....	11
3.2 Block diagram of PID (Auto-tuning)	12
3.3 Block diagram of PSO based PID	12
3.4 Fitness Function	13
3.5 Experiment Overview.....	15

CHAPTER 4.....	16
Results & Discussion	16
4.1 Transfer Function of DC Motor	16
4.1.1 DC Motor.....	16
4.2 PID Controller Tuning.....	18
4.3 Designing of PID Using Particle Swarm Optimization	20
4.3.1 Initializing the Parameter of the Particle Swarm Optimization	20
4.3.2 PSO based PID with different parameters value.....	31
4.3.3 PSO based PID vs. PID only.....	32
CHAPTER 5.....	34
Recommendation &Conclusion	34
5.1 Recommendation.....	34
5.2 Conclusion.....	34
Appendixes.....	35
Appendix I	36
Appendix II	37
Appendix III	41
Appendix IV	42
Appendix V	43
References	44

List of Figure

Figure 1 Separately Excited DC Motor	5
Figure 2 Flowchart of PSO [9]	11
Figure 3 Block Diagram of Auto tuning PID.....	12
Figure 4 Block Diagram of PSO based PID [2].....	13
Figure 5 Fitness function at main loop	15
Figure 6 Step Response for PID only.....	19
Figure 7 Step response for $c_1=0.6$, $c_2=0.06$	21
Figure 8 Step response for $c_1=0.8$, $c_2=0.08$	22
Figure 9 Step response for $c_1=1.0$, $c_2=0.10$	22
Figure 10 Step response for $c_1=1.2$, $c_2=0.12$	23
Figure 11 Step response for $c_1=1.2$, $c_2=0.25$	23
Figure 12 Step response of $n=10$	25
Figure 14 Step response of $n=20$	25
Figure 14 Step response of $n=30$	26
Figure 15 Step response of $n=40$	26
Figure 16 Step response of $n=50$	27
Figure 17 Step response of $n=100$	27
Figure 18 Step response for $w=0.9$	29
Figure 19 Step response for $w=0.8$	30
Figure 20 Step response for $w=0.7$	30
Figure 21 PSO based PID.....	32

List of Table

Table 1 Parameters of the motor	18
Table 2 Controller Parameters & Performance for PID-only	18
Table 3 Controller Parameters & Performance (Varies c_1 & c_2)	24
Table 4 Controller Parameters & Performance (Varies n).....	28
Table 5 Controller Parameters & Performance (Varies w).....	31
Table 6 Comparison of PSO based PID with different parameters value	31
Table 7 Comparison Result	33

Nomenclature

n	– number of particles in group
d	– dimension
t	– pointer of iterations (generations)
$V_{i,m}^{(t+1)}$	– velocity of particle i at iteration t
w	– inertia weight factor
$c1,c2$	– acceleration constant
$\text{Rand}(n)$	– random number between 0 and 1
$X_{i,d}^{(t)}$	– current position of particles i at iterations
$Pbest_{i,m}$	– best previous position of the i th particle
$Gbest_m$	– best particle among all the particles in the population
V	– applied voltage
E	– back EMF
I	– current
R	– DC resistance
L	– armature inductance
ω	– speed
T	– torque
K_b	– back EMF constant
K_t	– torque sensitivity
J_m	– motor moment of inertia
τ_m	– mechanical time constant
τ_e	– electrical time constant
K_p	– proportional gain
K_i	– integral gain
K_d	– derivative gain
Tr	– rise time
T_s	– settling time
M_p	– peak overshoot

CHAPTER 1

Introduction

1.1 Background of Study

1.1.1 DC Motor

There are mainly two types of dc motors widely used in industry. The first type is the conventional DC motor where the current through the field coil of the stationary pole structure produces flux [9]. The second type is the brushless DC motor where the flux is provided through necessary air gap of permanent magnet instead of the wire-wound field poles [9].

Brushed DC motors are widely used in applications ranging from toys to push-button adjustable car seats [14]. Inexpensive, easy to drive, and are readily available in all sizes and shapes are some of the characteristics of brushed DC (BDC) motors [14]. The basic components BDC motors are: a stator, rotor, brushes and a commutator [14].

In this project, the DC motor chosen are **Axsys Technology 3625V-084 Brush DC Motor** for high acceleration application requiring improved response for rapid start/stop actions.

1.1.2 PID Controller

A proportional-integral-derivative controller (PID controller) is commonly used in industrial control systems as a feedback controller. The difference between a measured process variable (PV) and a desired set point (SP) is an error which is calculated by the PID controller. The controller will minimize the error by regulating the process control inputs.

The PID controller calculation 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 [15]. These gains value 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 [15]. The weighted sum of these three actions is used to adjust the process through a control element such as the position of a control valve, or the power supplied to a heating element [15].

The aim of PID controller tuning is to determine parameters that meet closed loop system performance specifications, and the robust performance of the control loop over a wide range of operating conditions should also been sure [2]. Virtually, it is often difficult to simultaneously achieve all of these desirable qualities.

For instance, if the PID controller is adjusted to provide better transient response to the set point change, it usually results in a sluggish response when under disturbance conditions [2]. In contrast, if the control system is made robust to disturbance by choosing conventional values for the PID controller, it may result in a slow closed loop response to a set point change [2].

Recently, many methodologies of evolutionary algorithms have been widely proposed for PID tuning of DC motor. One of the well-known evolutionary algorithms that have been evolved rapidly for the past few years is the Particle Swarm Optimization (PSO) based technique. PSO is an evolutionary algorithms based on population of potential solutions and motivated by the simulations of social behaviour instead of survival of fittest individual.

1.2 Problem Statement

The main problems of PID controller:

- Sluggish
- Slow closed loop response to a set point change

1.3 Objectives

The objectives of this project include:

- To improve the gain of PID controller by particle swarm optimization integration for speed control of DC motor.
 - Minimize the rise time, minimize the maximum error and minimize the settling time.
- To obtain the dynamic response of speed control problem using MATLAB model.

1.4 Scope of Study

This project focuses on optimizing the PID controller for DC motor system using Particle Swarm Optimization technique. There is no constraint in the searching space of the optimal PID parameters. The new PID tuning algorithm is applied to the speed control of DC motors. The performance measure to be minimized contains the following objectives of the PID controller [1]:

- Minimize the rise time
 - Time required for the system response to rise from: 10% to 90% (Over damped); 5% to 95%; 0% to 100% (Under damped) of the final steady state value of the desired response [1].

- Minimize the maximum overshoot
 - The maximum peak value of the response curve measured from the desired response of the system [1].

- Minimize the settling time
 - Time required for response to reach and stay within 2% of final value [1].

CHAPTER 2

Literature Review

2.1 DC Motor Control

The three most common for speed control of DC motor are resistance control, armature voltage control, and armature resistance control. However, Ayasun and Karbayez, from their study also identified that feedback control system is also implemented of speed control system for DC motor drives [12]. In the armature voltage control technique, the voltage applied to the armature circuit is varied without changing the voltage applied to the field circuit of the motor [12]. However, in the field resistance control method, a series resistance is inserted in the shunt-field circuit of the motor in order to change the flux by controlling the field current [12]. An increase in the armature resistance results in a significant increase in the slope of the torque-speed characteristics of the motor while the no-load speed remains constant for the armature resistance control [12]. These criteria are true for a separately excited DC motor system only.

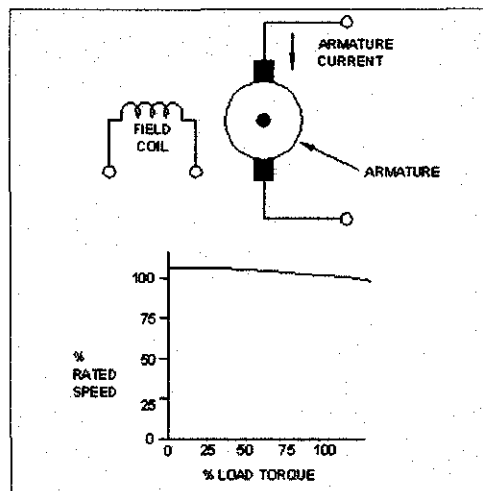


Figure 1 Separately Excited DC Motor

In [9], Mehdi Nasri, Nezamabadi-pour and maliheMoghfoori had found out that the designed PID based PSO has much faster response than the response of a conventional method which is by using Ziegler-Nichols method. With PSO, the response is better in term of rise time, settling time and also the error associated with the methods proven to be lesser than conventional method. It is also observed that PSO performed better than Genetic Algorithm (GA) and Linear Quadratic Regulator (LQR) techniques. All three techniques of optimization are proven to be efficient powerful optimization tools for obtaining optimal solutions of the BLDC motor compare to conventional design procedures. The results show that the proposed controller (PSO) can perform an efficient search for the optimal PID controller. By comparison with LQR and GA methods, it shows that this method can improve the dynamic performance of the system in better way [9].

From all the literature reviews of others work, it has been found out that the brush DC motor model illustrates good electrical and mechanical performances more than other DC motor models [3]. Hence, for this study, the brush DC motor model is chosen for research.

2.2 Evolutionary Computation and Swarm Intelligence Paradigms

Evolutionary computation (EC) and Swarm Intelligence (SI) are examples of artificial intelligence (AI) [4]. EC is initiated upon the principles of biological evolution whereas SI techniques are inspired by swarm behavioural patterns occurring in nature [8]. With the increase of computational power, AI has increasingly been used to solve complex linear and nonlinear control problems [11].

2.2.1 Evolutionary Computation

EC techniques are inspired by biological concepts such as population mutation, self-organizing and survival of the fittest [11]. They are considered as main purpose of stochastic search methods that simulate the process of natural selection and evolution in the biological world [11]. There are four major evolutionary techniques namely:

- *Genetic Programming (GP)*: GP is used to search for the fittest program to solve a specific problem. Individuals are represented as trees and the attention is on the genetic composition of the individual [11].
- *Evolutionary Programming (EP)*: EP is generally used to optimize real-valued continuous functions. EP uses selection and mutation operators and does not use the crossover operator. The focus is on the observed characteristics of the population. The selection operator is used to determine chromosomes (called parents) for mating in order to generate new chromosomes (called offspring.) [11]
- *Evolutionary Strategies (ES)*: ES is used to optimize real-valued continuous functions. ES incorporates selection, crossover and mutation operators. ES optimizes both the population and the optimisation process by evolving the strategy parameters [10].
- *Genetic Algorithms (GA)*: The GA is a commonly used of evolutionary algorithm. PSO is similar to the GA since these two evolutionary heuristics are population-based search methods. The GA and its variants have been popular in academia and the industry mainly because of its intuitiveness, ease of implementation and its ability to solve highly non-linear, mixed integer optimization problems that are typical of complex engineering systems [6].

2.2.2 Swarm Intelligence

The study of collective behaviour in decentralized and self-organized systems is the base of Swarm Intelligence (SI) methods [11]. SI systems are basically made up of a population of simple agents interacting locally with one another and with their environment [11]. There is no centralized control structure explaining how individual agents should behave, however, the local interactions between agents often lead to the growth of a global behaviour. Two of the most successful SI techniques modelled on the behaviour of natural systems are ant colony optimization (ACO) proposed by Dorigo and Gambardella (1997) and particle swarm optimization (PSO) proposed by Kennedy and Eberhart (1995) [11].

2.3 Particle Swarm Optimization

PSO is one of the optimization techniques and a type of evolutionary computation technique. The technique is derived from research on swarm such as bird flocking and fish schooling [9]. In the PSO algorithm, for a d-variable optimization problem, a flock of particles are put into the d-dimensional search space with randomly chosen velocities and positions knowing their best values in contrast of GA which using evolutionary operators such as mutation and crossover to manipulate their algorithms, [9].

This algorithm is proposed by Dr. Eberhart and Dr. Kennedy in 1995 uses a single dimension (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 [2].

To implement this PSO technique, firstly, a group of random potential solution need to be initialized. All the potential solution is defined as “particle” in this technique. Each particle has their own velocity that will direct their direction to fly. At the same time, to optimize the solution, the particles have fitness value which will be

evaluated by fitness function. The particles are flying through the problem space succeeding the current optimum in the population. Each particle will ensure to remember their own best position they have achieved so far which is called Pbest. At the same time, there exist a lot of Pbest for each particle in the swarm, however, the particle with the best value obtain so far as well as the greatest fitness is called the global best (Gbest) of the swarm. After the two best values have been obtained, the particles will update its velocity and position to the current best value.

The summary on the main steps in the particle swarm optimization and selection process is listed in the following [1]:

- i. A population of particles with random positions and velocities in d-dimensions is initialized.
- ii. The fitness of each particle in the swarm is evaluated.
- iii. Each particle's fitness is compared with its previous best fitness (Pbest) in iteration. If the current value is better than Pbest, then Pbest is set equal to the current value and the Pbest location equal to the current location in the d-dimensional space.
- iv. Pbest of each particle is compared and the swarm global best location is updated with the particle with greatest fitness (Gbest).
- v. The velocity and position of the particle is updated according to equations (1) and (2) respectively.
- vi. Steps (ii) to (v) are repeated until its reach maximum iterations.

$$V_{i,m}^{(t+1)} = wV_{i,m}^{(t)} + c_1 \text{rand}(n) (Pbest_{i,m} - x_{i,m}^{(t)}) + c_2 \text{rand}(n) (Gbest_{i,m} - x_{i,m}^{(t)}) \quad (1)$$

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

The PSO search and minimization algorithm has many parameters and these are described as follows: ω is called the inertia weight factor that controls the exploration and exploitation of the search space because it dynamically adjusts velocity. Vmax is the maximum allowable velocity for the particles (i.e. in the case where the velocity of the particle exceeds Vmax, then it is limited to Vmax). Thus, resolution and

fitness of search depends on V_{max} . If V_{max} is too high, then particles will move beyond a good solution. If V_{max} is too low, particles will be trapped in local minima. The constants C_1 and C_2 in (1) and (2) termed as cognition and social components, respectively. These are the acceleration constants which changes the velocity of a particle toward [15].

CHAPTER 3

Methodology

3.1 Flowchart of PSO

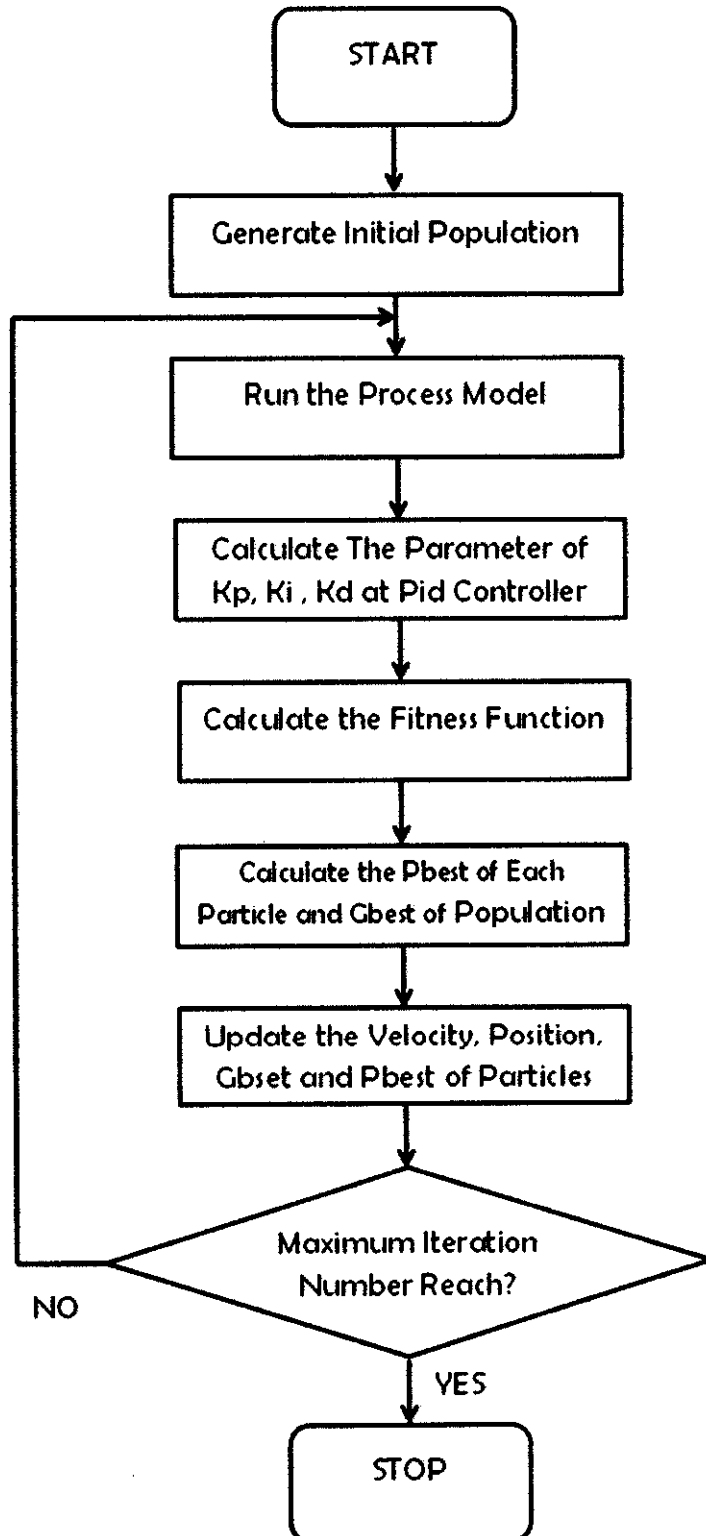


Figure 2 Flowchart of PSO based PID [9]

3.2 Block diagram of PID (Auto-tuning)

The gains of PID controllers can be automatically tuned using Simulink to meet performance requirements. There exists an AutotunerPID Toolkit that allows experimenting with well-established tuning methods and comparing the results of different methods. The AutotunerPID Toolkit simulates a single SISO control loop [17]. The main component of the Toolkit is a Simulink file, which includes a PID with auto tuner, the plant to be controlled and some auxiliary blocks to manage the simulation. Figure 3 shows the block diagram of PID for auto tuning that is applied throughout this study.

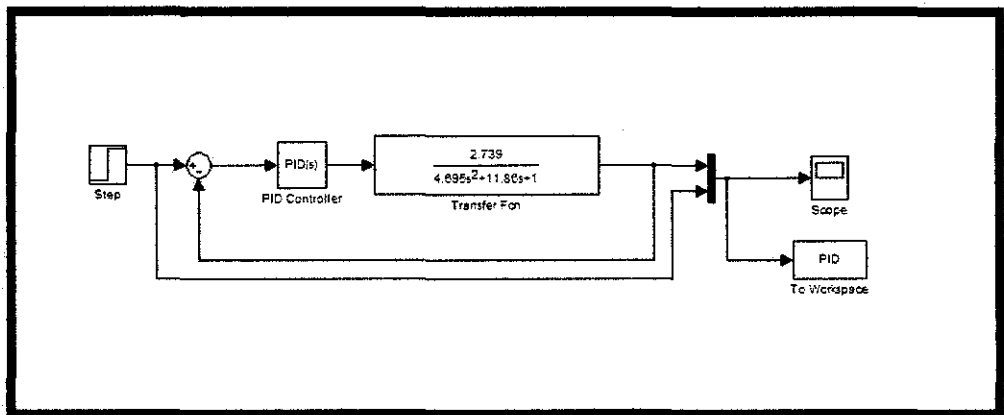


Figure 3 Block Diagram of Auto tuning PID

3.3 Block diagram of PSO based PID

MATLAB with its tool boxes such as Simulink is one of the most popular software packages used to run the simulation or testing of a plant or process system before the real implementation onto the equipment. Thus, the software is utilized by embedding it into the DC motor model system for the optimization technique. The block diagram that is integrated into the system is as shown in Figure 4 .However, for PSO technique to integrate into PID controller, the usage of M-file with programming is required (refer to Appendix II).

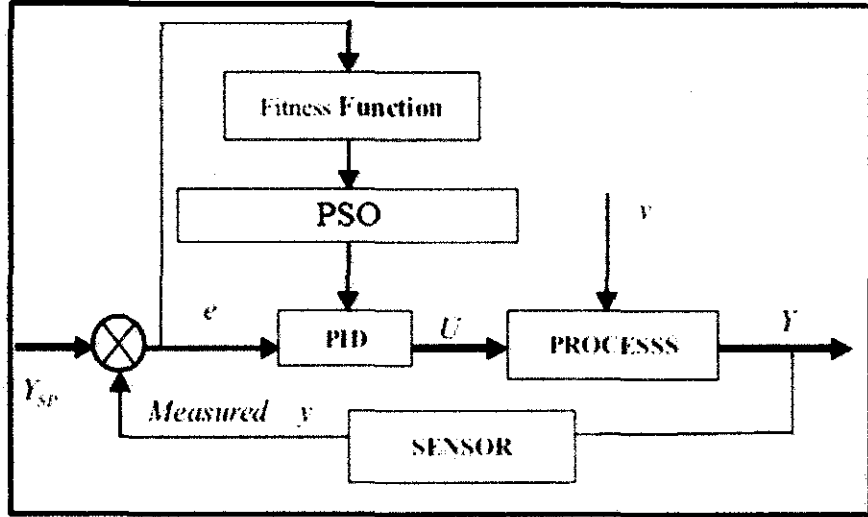


Figure 4 Block Diagram of PSO based PID [2]

3.4 Fitness Function

In PID controller design methods, the most common performance criteria are integrated absolute error (IAE), the integrated of time weight square error (ITSE) and integrated of squared error (ISE) that can be evaluated analytically in the frequency domain [9]. The ISE, IAE and ITSE performance criterion formula are as follows:

ITSE Index:

$$ITSE = \int_0^{\infty} te^2(t)dt \quad (3)$$

This method gives little emphasis on initial errors and heavily penalizes errors occurring late in the transient response to a step input [11].

IAE Index:

$$IAE = \int_0^{\infty} |e(t)|dt \quad (4)$$

Systems based on this index correct the control error [11].

ISE Index:

$$ISE = \int_0^{\infty} e^2(t)dt \quad (5)$$

The upper limit ∞ may be replaced by T which is chosen sufficiently large such that $e(t)$ for $T < t$ is negligible and the integral reaches a steady-state [11]. A characteristic of this performance index is that it corrects large errors heavily and small errors lightly [11]. A system designed by this characteristic tends to show a rapid decrease in a large initial error [11]. Hence the response is fast and oscillatory leading to a system that has poor relative stability [11]. The ISE performance index is selected to be used in this study.

However, a time domain criterion is used for evaluating the PID controller [9]. A set of good control parameters, P, I and D can provide a good step response that will result in performance criteria minimization in time domain [9]. These performance criteria in time domain include the overshoot, rise time and settling time. Performance criteria can be defined as follows:

$$F = e^2\beta + Mp\alpha \quad (6)$$

- To compute the error:

$$e = y_{out} - 1 \quad (7)$$

- To compute system overshoot:

$$Mp = \max(y_{out}) - 1 \quad (8)$$

The optimum selection of α & β depends on the designer's requirement & the characteristic plant under control.

```

for i = 1:n,
current_fitness(i) = objfunction(current_position(:,i)) ;
end

for i = 1 : n
    if current_fitness(i) < local_best_fitness(i)
        local_best_fitness(i) = current_fitness(i);
        local_best_position(:,i) = current_position(:,i) ;
    end
end

```

Figure 5 Fitness function at main loop

Figure 5 calculates the fitness function (objective function) aimed to be minimized at the main loop in the source code. This function is basically just the mean squared error of the difference between reference signal and the output of the system.

3.5 Experiment Overview

The DC motor model is constructed based on its electrical and mechanical characteristics. The block diagram for the DC motor that will be integrated with the PSO technique is modeled as well.

Based on the transfer function that has been computed, a MATLAB Simulink model is developed and few tests case have been simulated using MATLAB/SIMULINK version 7.10.0.499.

Flow of process through completion this study is presented in Appendix III. The Gantt Chart for FYP I and FYP II is attached in Appendix IV and V respectively. The dynamic performance, such as rise time, settling time and maximum overshoot are calculated based on the theory discussed from [19].

CHAPTER 4

Results & Discussion

4.1 Transfer Function of DC Motor

For this project, a 3625V-084 Brush DC Motor from Axsys Technology is chosen based on the criteria given; high energy product Samarium Cobalt (SmCo) magnets combined with optimum motor windings provide the maximum torque and performances available in a broad range of frames sizes. Using the standard equation for the brushed type DC motor given in the Axsys Technology manual book and the reference from datasheet (see Appendix I), the motor transfer function is obtained.

4.1.1 DC Motor

The basic equations for DC motor are:

$$V = E + IR \quad (9)$$

$$E = K_b \omega \quad (10)$$

$$T = K_t I \quad (11)$$

Substitute equation (10) & (11) into (9) will resulting to speed-torque characteristic for DC motor:

$$V = K_b \omega + \frac{TR}{K_t} \quad (12)$$

The first portion in the equation (12) indicates the voltage required overcoming the back EMF of the motor at desired speed and the second term specifies the voltages required to produce the desired torque.

4.1.1.1 Motor Transfer Function

Equation (7) is the simplest transfer function for a DC motor, neglecting the motor induction, friction and shaft resonance.

$$\frac{\omega}{v} = \frac{1/k_b}{\tau_m s + 1} \quad (13)$$

However, in this project, the effect of motor inductance is included. Hence, the transfer function is modified to include an additional term.

$$\frac{\omega}{V} = \frac{1/K_b}{(\tau_m s + 1)(\tau_e s + 1)} \quad (14)$$

Where the mechanical time constant and electrical time constant are:

$$\tau_m = \frac{R J_m}{K_t K_b} \quad (15)$$

$$\tau_e = \frac{L}{R} \quad (16)$$

The transfer function in equation (14), assumes that the mechanical time constant is much larger than the electrical time constant and the friction is negligible.

The parameters of the DC motor used for simulation are as follows:

Table 1 Parameters of the motor

Parameters	Values and units
Resistance, R	2.17Ω
Inductance, L	0.89mH
Back EMF Constant, Kb	0.230 V/(rad/sec)
Torque Sensitivity, Kt	32.6 oz-in/A
Rotor Inertia, Jm	0.0396
Electrical time constant, τ_e	0.41 ms
Mechanical time constant, τ_m	11.45ms

Using the parameters from table 1 and equation (14), the transfer function is calculated. Thus, the transfer function for this specific motor is:

$$G(s) = \frac{2.793}{4.695s^2 + 11.86s + 1} \quad (17)$$

4.2 PID Controller Tuning

Applying the DC Motor transfer function into the PID controller block diagram, the controller is tuned using auto tuned tool from MATLAB to produce the best performance for the motor.

Table 2 Controller Parameters & Performance for PID-only

Response	PID Parameters			Dynamic Performance		
	Kp	Ki	Kd	Tr (s)	Ts(s)	Mp (%)
Original	5.7986	0.15701	1.4953	2.06	11.7	2.42
Tuned	0.5959	0.11309	-3.7994	9.11	28.2	9.49

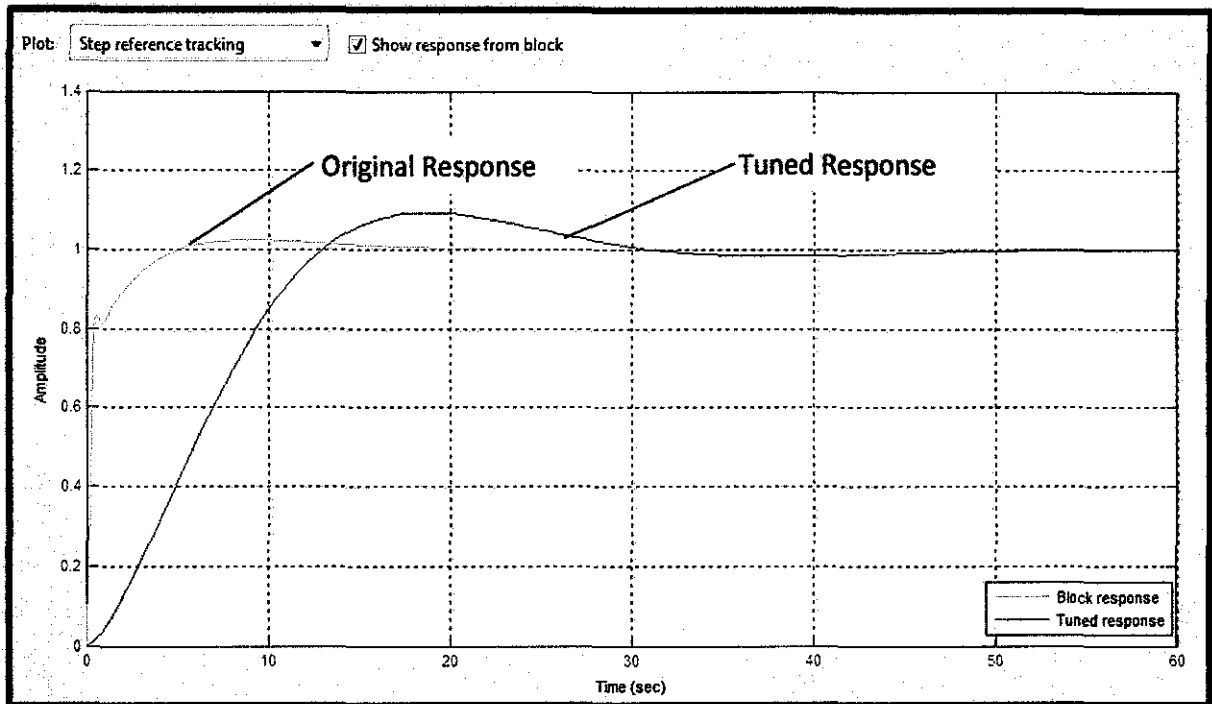


Figure 6 Step Response for PID only

Figure 6 shows the response for the transfer function when applying PID controller only. Grey responds indicates the PID response of the block without being optimized yet. The blue responds shows the optimize response for the PID after being automatically tuned by the MATLAB/Simulink.

Table 2 on the other hand, shows the value for K_p , K_i and K_d value as well as the dynamic performance after being automatically tuned. The value for K_p , K_i and K_d of tuned response is used to integrate with the PSO, to get a new value of PID and optimize this current performance. The performance of tuned response based PID only will be compared with PSO-PID performance, to analyse which is better method. This concludes the simulation of the motor in closed loop configuration.

4.3 Designing of PID Using Particle Swarm Optimization

In a PSO system, a swarm of individuals (called *particles*) fly through the search space. Each particle represents a potential solution to the optimization problem. The position of a particle is affected by the best position visited by itself (i.e. its own experience) and the position of the best particle in its entire population. The best position obtained is referred to as the *global* best particle. The performance of each particle (i.e. how close the particle is from the global optimum) is measured using a fitness function that varies depending on the optimization problem [11].

Swarm particles display distinct behavioural characteristics, namely swarm convergence and particle explosion, as they traverse a system's space searching for an optimal solution [11]. Variations of the swarm's behaviour are achieved by adjusting four parameters of the PSO algorithm, namely: *acceleration constant ($c1$ & $c2$)*, *population size (n)*, *inertia weight factor (w)* and *dimension (d)*. These four parameters are set at the beginning of each trial and remain constant throughout.

In this proposed PSO method, each particles contains three members; P, I and D. It means that the search space has three dimension and particles must 'fly' in a three dimensional space [9]. However, for acceleration constant, population size and inertia weight factor, the parameters need to be identified heuristically.

4.3.1 Initializing the Parameter of the Particle Swarm Optimization

The main essential part for PSO to be integrated into PID is to initialize its parameter. The parameter gives a huge impact on the output response. The initializations that are vital include the population size, dimension, acceleration constant, and inertia weight factor. All the parameters stated need to be identified heuristically accept for the dimension. The dimension is defined to the objective to optimize. In this study, the dimension is 3.

4.3.1.1 *C1 and C2 parameters*

With the weighted factor remain constant $w=0.9$ and size of population of 50, Figure 7 till Figure 11 shows the step responses with different accelerating constant, $c1$ and $c2$ at each trials.

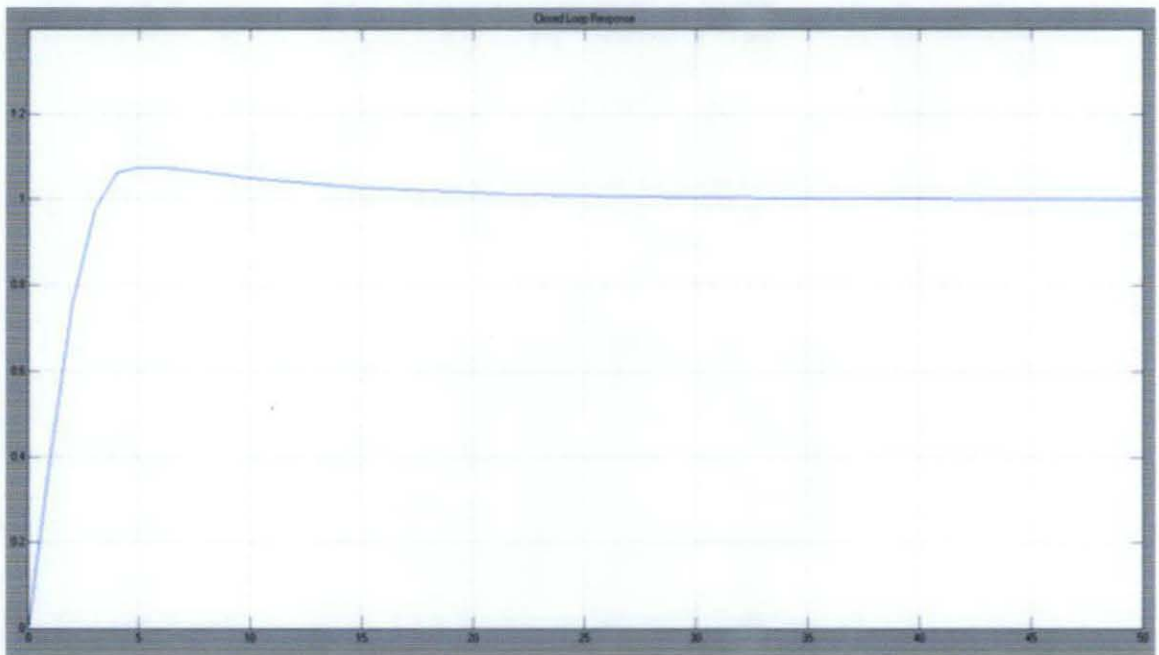


Figure 7 Step response for $c1=0.6$, $c2=0.06$

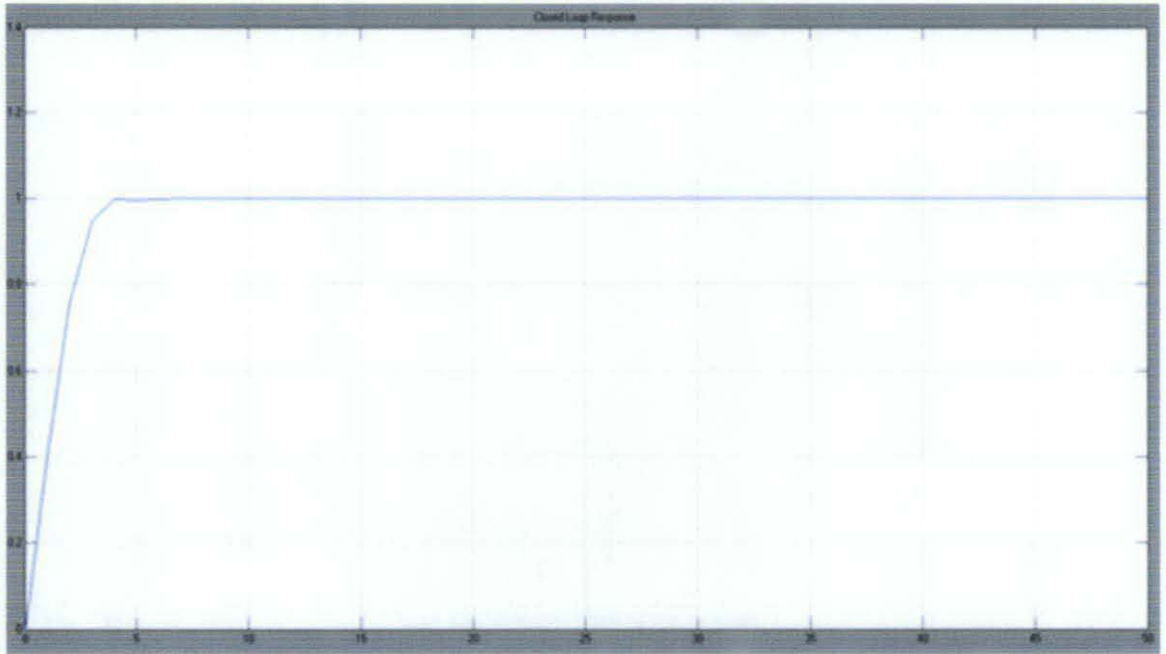


Figure 8 Step response for $c_1=0.8$, $c_2=0.08$

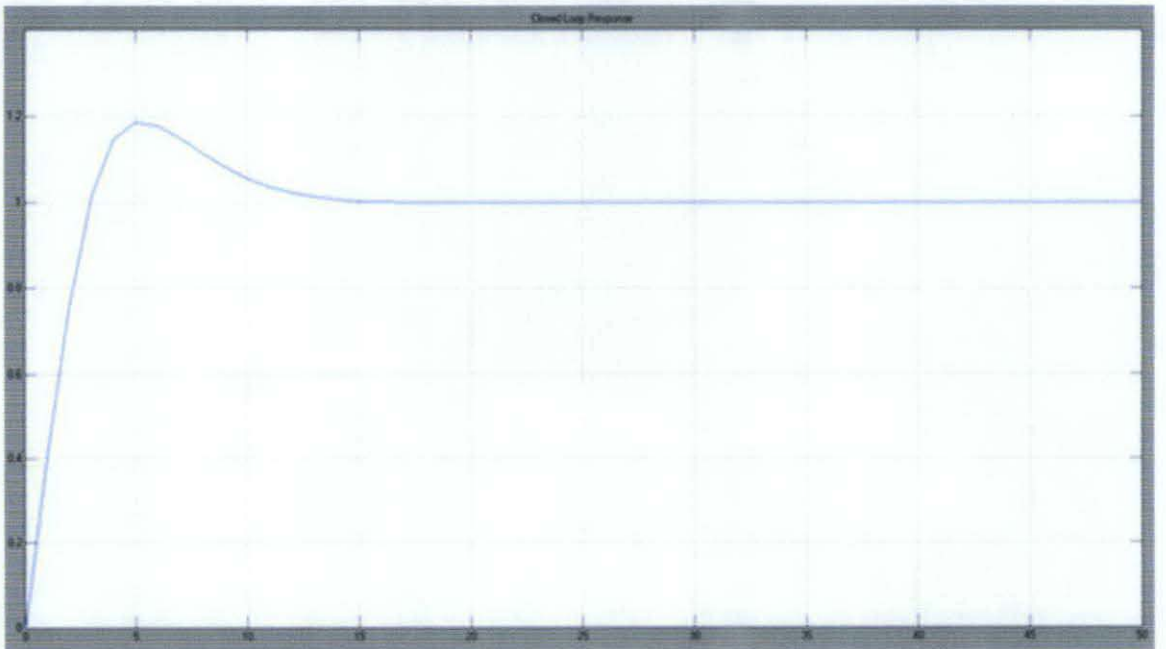


Figure 9 Step response for $c_1=1.0$, $c_2=0.10$

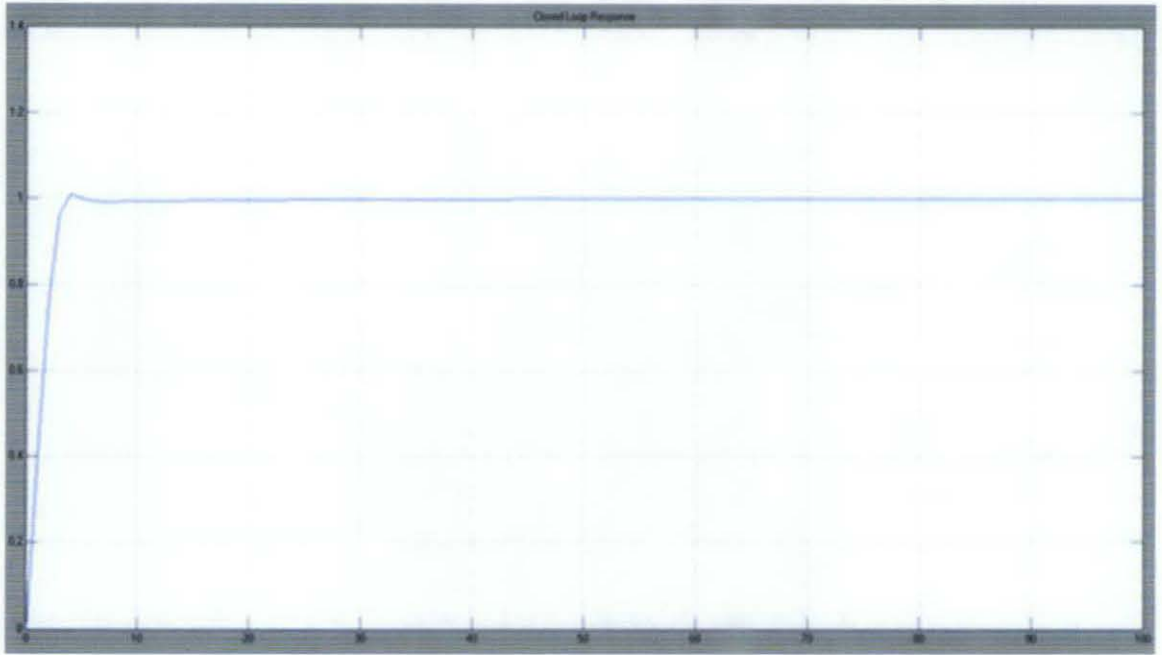


Figure 10 Step response for $c_1=1.2$, $c_2=0.12$

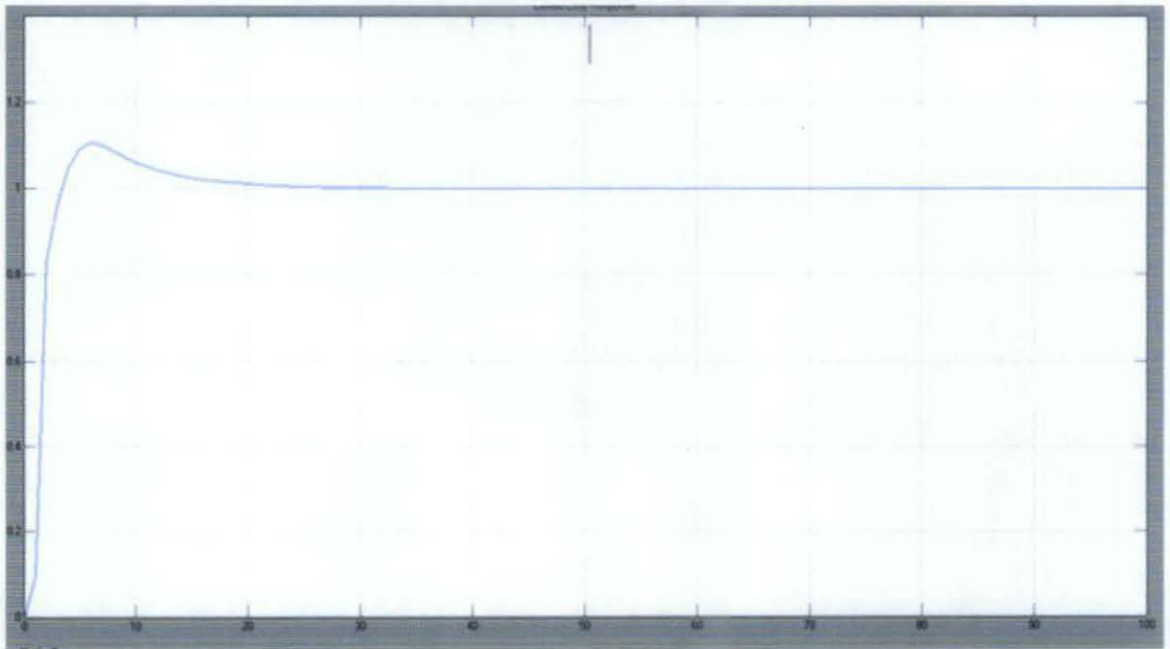


Figure 11 Step response for $c_1=1.2$, $c_2=0.25$

Table 3 Controller Parameters & Performance (Varies c1 & c2)

Trial	c1	c2	Kp	Ki	Kd	Tr(s)	Ts (s)	Mp(%)
1	0.06	0.6	3.2706	0.4437	2.5273	2.25	2.4	7.4
2	0.08	0.8	3.9456	0.2212	0.1789	2.30	6.5	0.0
3	0.10	1.0	4.0725	0.7134	-0.189	2.25	15.0	19.0
4	0.12	1.2	3.4742	0.4997	-1.087	2.50	9.0	0.8
5	0.25	1.2	4.2678	0.1204	0.7073	3.50	25.0	10.0

Table 3 shows the dynamic performance when varying the accelerating constant value. Trial 2 ($c1 = 0.8$ and $c2 = 0.08$) performance are better when compared to the performance of other accelerating constant. There are some overshoot for the system when applying for other value of $c1$ and $c2$, however, there are no overshoot when the value of accelerating constant are changed to 0.8 and 0.08. The settling time is the lowest and the rise time is closed to other trials. The simulation shows that these are the optimum values of accelerating constant for this plant.

4.3.1.2 Population number of the particles

According to the theory of Particle Swarm Optimization technique, the number of particles usually ranges from 20 to 40. However, for some difficult or special problems, the population can be increase to 100 even 200 particles. Figure 12 till Figure 17 shows the responses when the number of particles is varies and other parameters are set to remain constant ($c1=0.08$, $c2=0.8$, $w=0.7$).

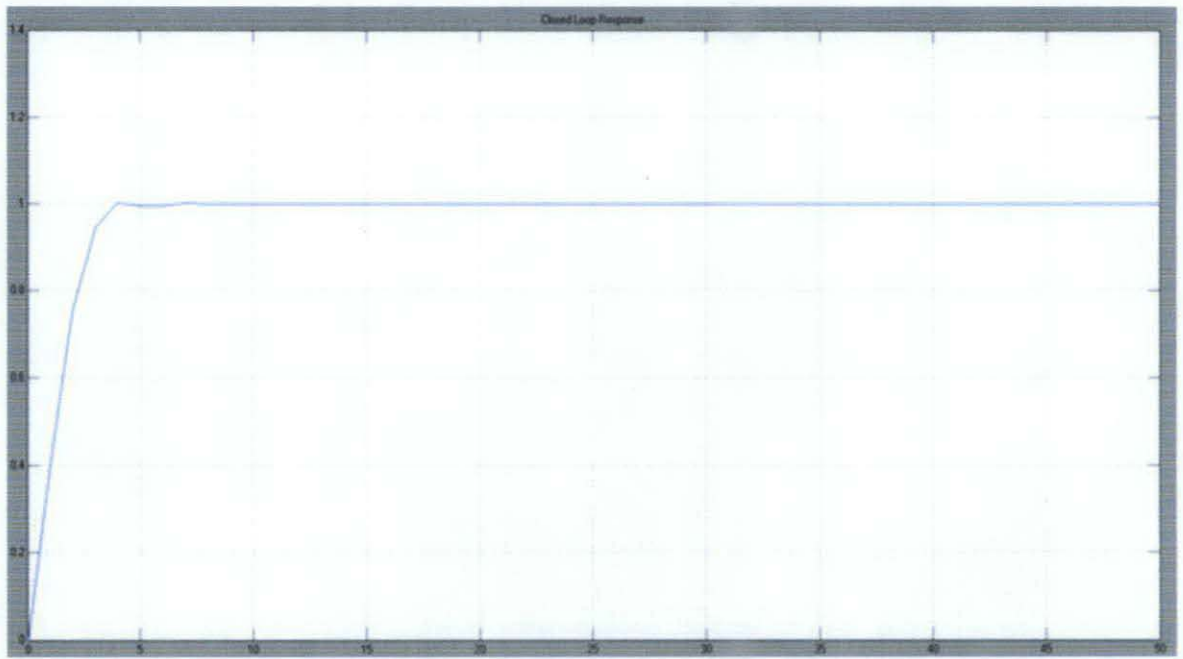


Figure 12 Step Response of $n=10$

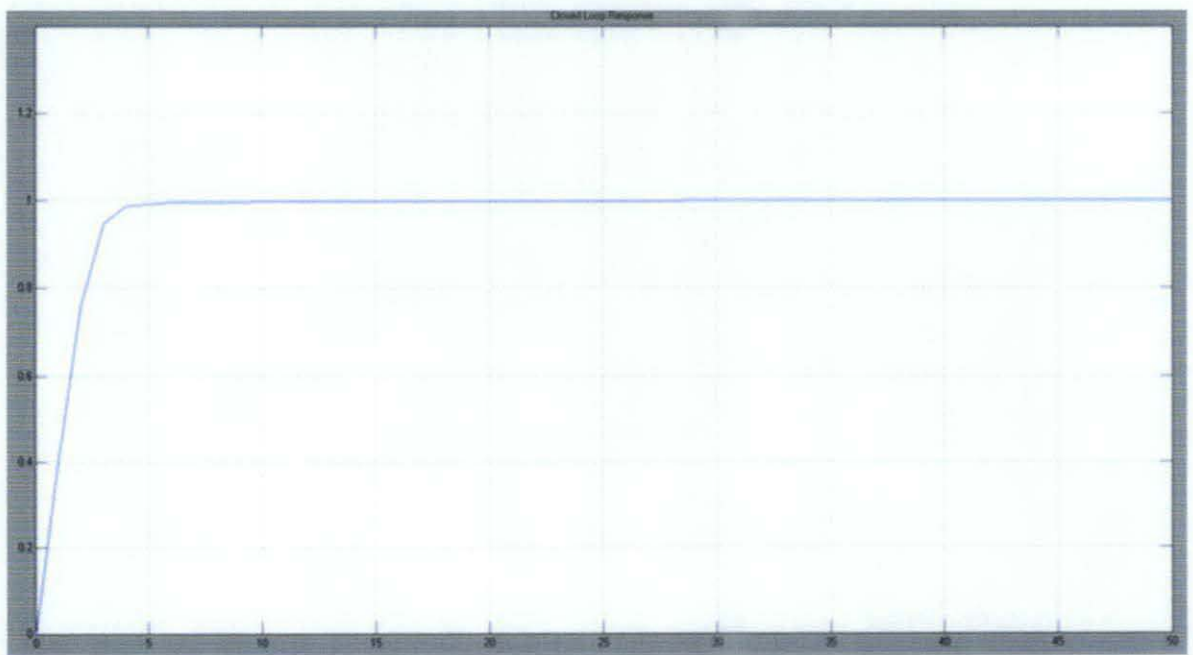


Figure 13 Step Response of $n=20$

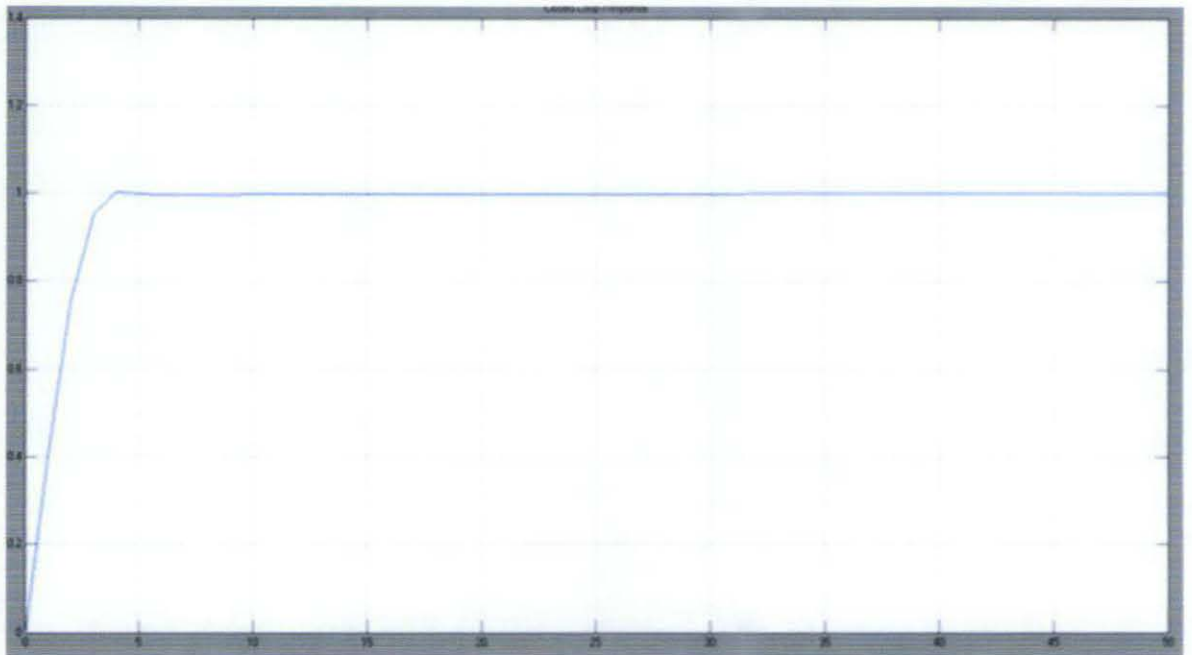


Figure 14 Step Response of $n = 30$

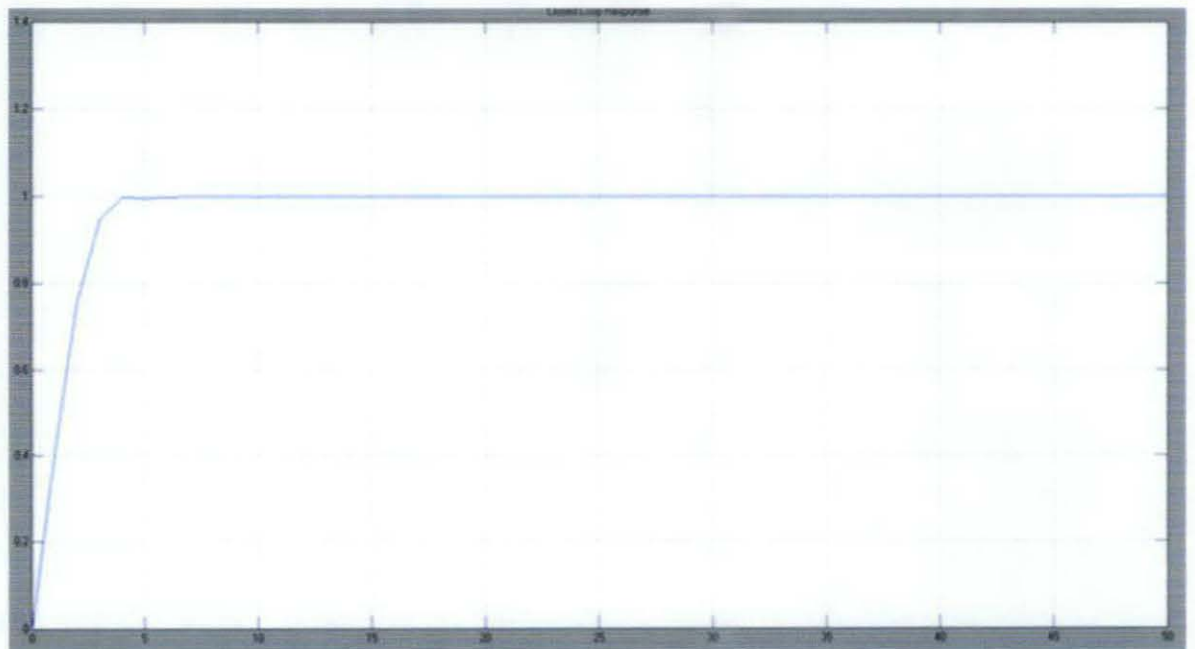


Figure 15 Step Response of $n = 40$

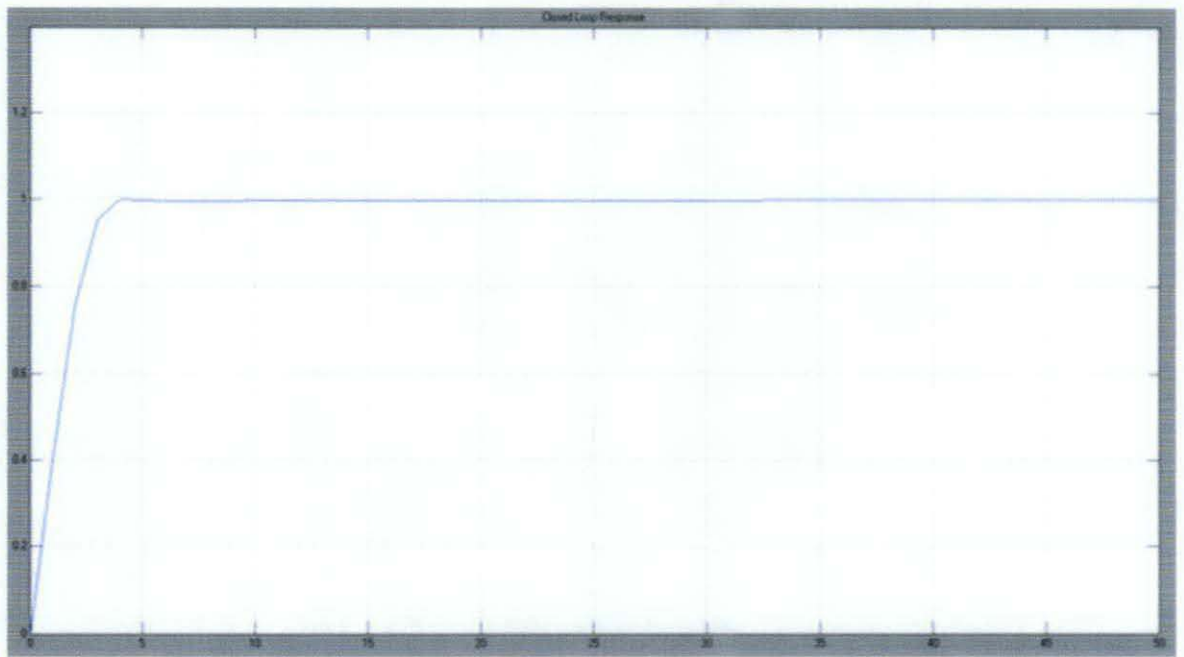


Figure 16 Step Response $n=50$

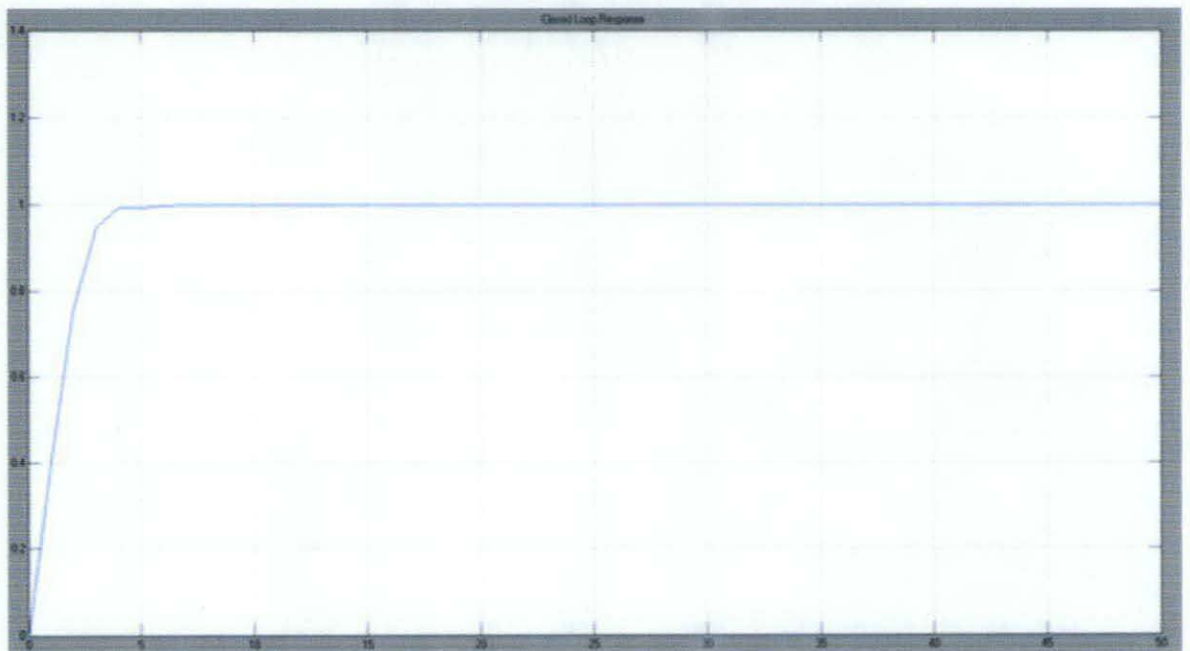


Figure 17 Step Response of $n = 100$

Table 4 Controller Parameters & Performance (Varies n)

Trial	n	Kp	Ki	Kd	Tr(s)	Ts (s)	Mp(%)
1	10	3.3972	-1.754	0.2986	2.3	9.4	0.0
2	20	5.3073	0.1964	1.5744	2.3	7.5	0.0
3	30	3.9456	0.2212	0.1789	2.3	6.5	0.0
4	40	4.0014	0.2185	0.1683	2.3	4.8	0.0
5	50	4.1109	0.2256	1.2895	2.3	6.5	0.0
6	100	3.9259	0.2188	0.1760	2.3	5.5	0.0

A reasonably large population is necessary for a good convergence. If the size of population is too small, the particles could converge to sub-optimal front, but with limited number of non-dominated solutions, which is insufficient in terms of solution spread and coverage [15]. This is because a smaller number of particles do not sufficiently sample the search space, and as a result, certain existing particles could quickly become too dominant early on, and they would prevent other potentially good particles from being produced [15]. A large initial population size would allow for a better sampling of the search space and from there onwards allow PSO to better use domination comparisons operations to find a wide spread of solutions.[15]

Table 4 shows the performance of the output response of different population number. The performance are varies as the population varies. Trial 4 (highlighted) indicates the best performance when the population is set to 40. The settling time is faster which is 4.8s compared to others. The performance of 100 populations is quite good; however the time taken for the simulation to be completed is about half an hour (30 minutes). Thus, taking into account the simulation time, the best population for this plant to be iterated by the swarm is 40.

4.3.1.3 Inertia weight factor

Another parameter to be heuristically identified is the inertia weight factor, w . The use of inertia weight is to control the velocity to give high efficiency results of PSO. Suitable selection of inertia weight provides a balance between global and local explorations [15]. Figure 18 till Figure 20 shows the output response, responding to different inertia weight factor and others parameters are kept constant. ($c1=0.08$, $c2=0.8$, $n=40$)

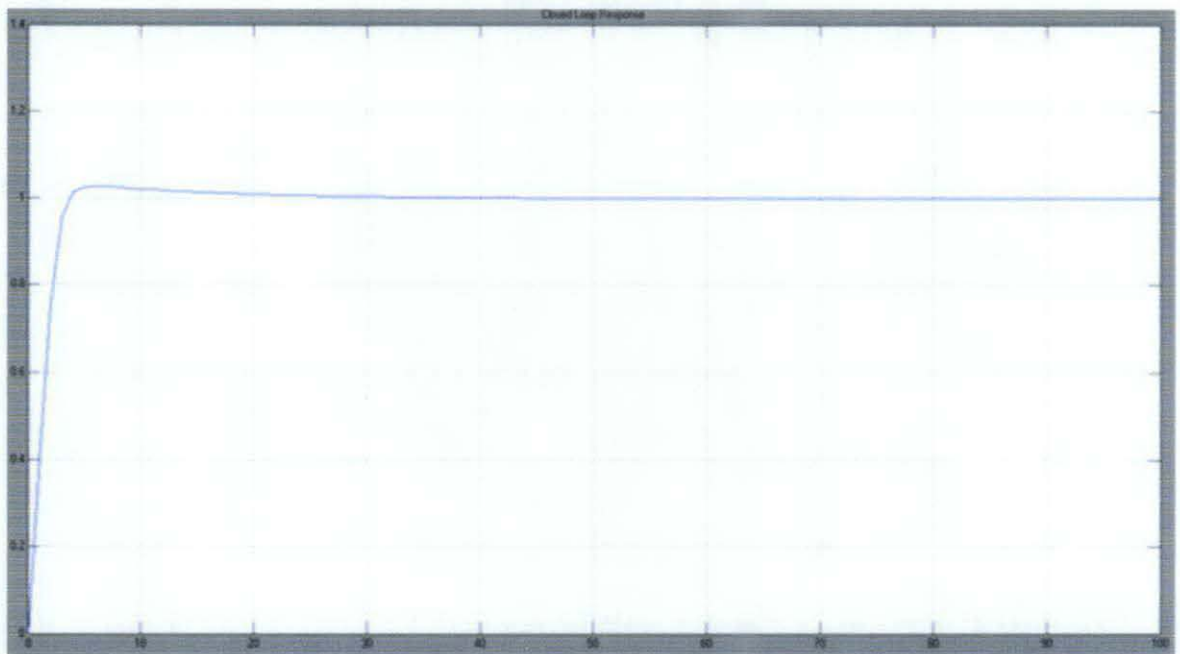


Figure 18 Step Response for $w = 0.9$

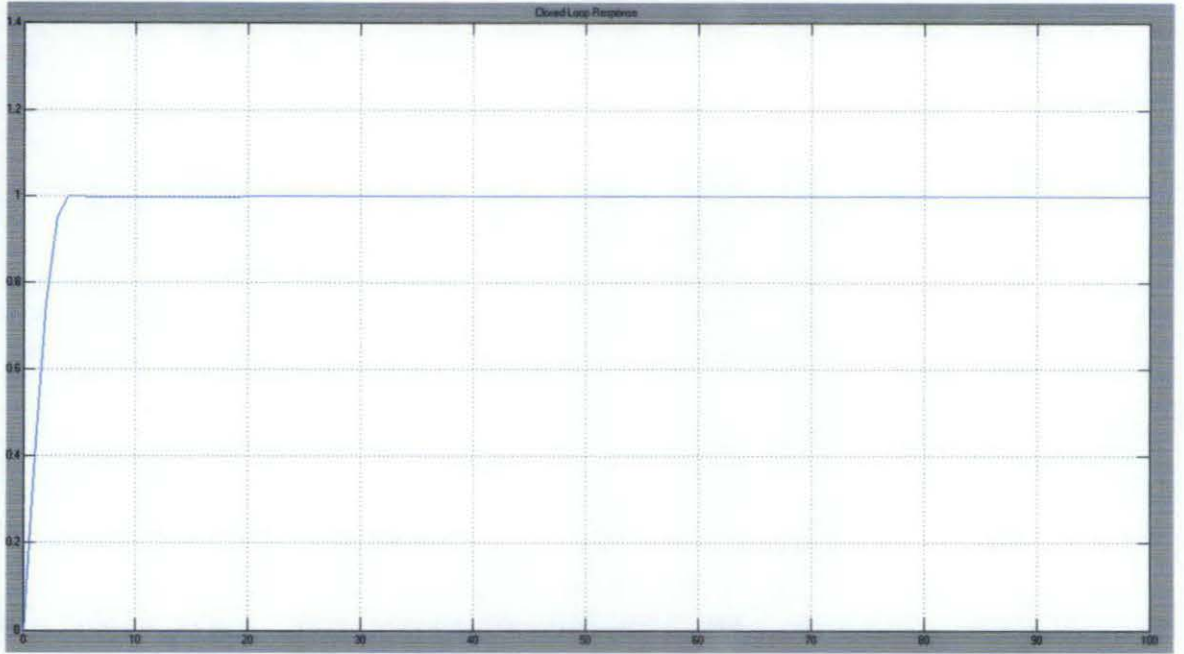


Figure 19 Step Response for $w = 0.8$

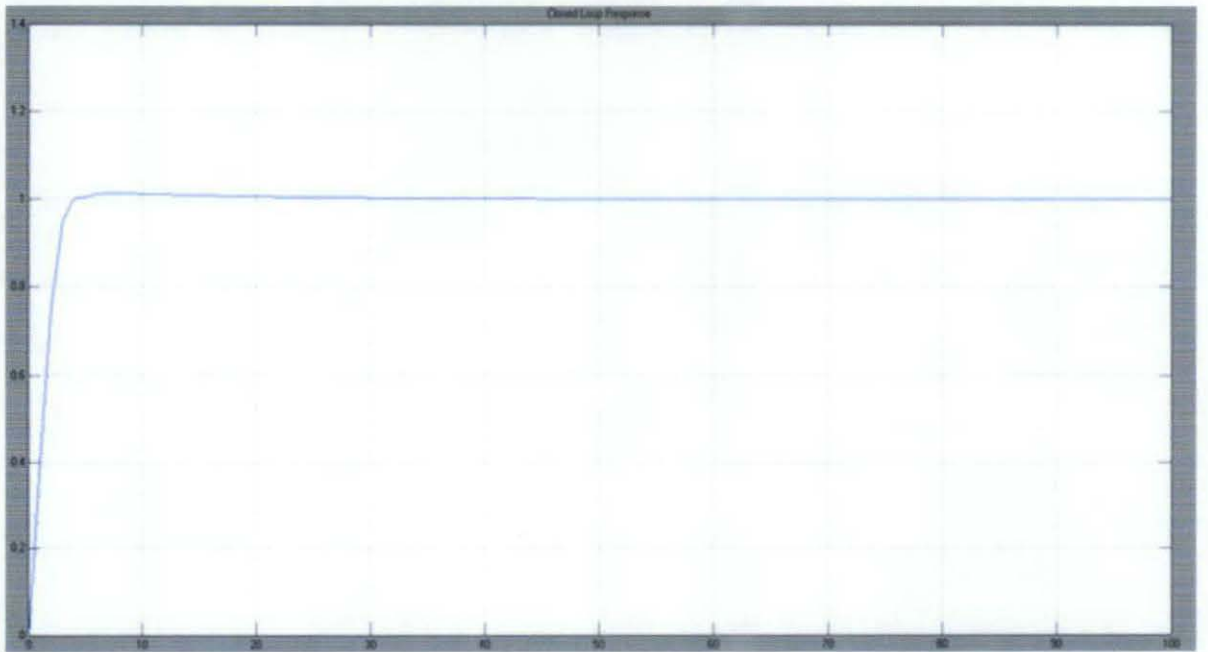


Figure 20 Step Response for $w = 0.7$

Table 5 Controller Parameters & Performance (Varies w)

Trial	w	Kp	Ki	Kd	Tr(s)	Ts (s)	Mp(%)
1	0.7	4.0014	0.2185	0.1683	2.3	4.8	0.0
2	0.8	4.3186	0.2313	1.4994	2.3	14.5	0.02
3	0.9	4.3441	0.0893	0.6136	2.5	30.0	0.75

Table 5 shows the performance of PSO based PID with the variations of inertia weight factors (w). This result shows that varies the inertia weight factor will give a huge impact to the output response. The best inertia weight factor for this plant is 0.7, which give the best performance output. Other values of w, resulting the output respond to have some overshoot and not optimally reach the best plant performance yet.

4.3.2 PSO based PID with different parameters value

After some trial and errors, the performance of different PSO Parameters is compared. Table 6 shows the best result achieved by varying a parameter at one time. Trial 1 is the best result when varying acceleration constant only. Trial 2 is the performance when varying the population number and inertia weight factors. Both population numbers and inertia weight factors give the same performance for their best result.

Table 6 Comparison of PSO based PID with different parameters value

Trial	PSO Parameters				Performance		
	c1	c2	n	w	Tr(s)	Ts (s)	Mp(%)
1	0.08	0.8	50	0.9	2.3	6.5	0.0
2	0.08	0.8	40	0.7	2.3	4.8	0.0

From Table 6, it clearly shows that Trial 2 is the best PSO parameters for this plant. The settling time is faster compare to Trial 1. Hence, these parameters will be initialized and keep constant through entire simulation.

4.3.3 PSO based PID vs. PID only

After several heuristic techniques, the best parameters have been obtained to evaluate the PSO-PID to optimize the DC motor performance. The step for tuning has been mention in section 2.3. The impact of each particles position in iterations within the search space is evaluated according to the ISE function. The PSO source code used for the tuning is given in Appendix II.

PSO parameters:

Size of the swarm "no of birds ", $n = 40$

Maximum number of "birds steps", bird step =40

Dimension of the problem = 3

PSO parameter $C1 = 0.8$

PSO parameter $C2 = 0.08$

Inertia weight factor, $w = 0.7$

After few simulations, the best result obtained as Figure 21 and the comparison of dynamic performance for PSO based PID and PID only is listed in Table 7.

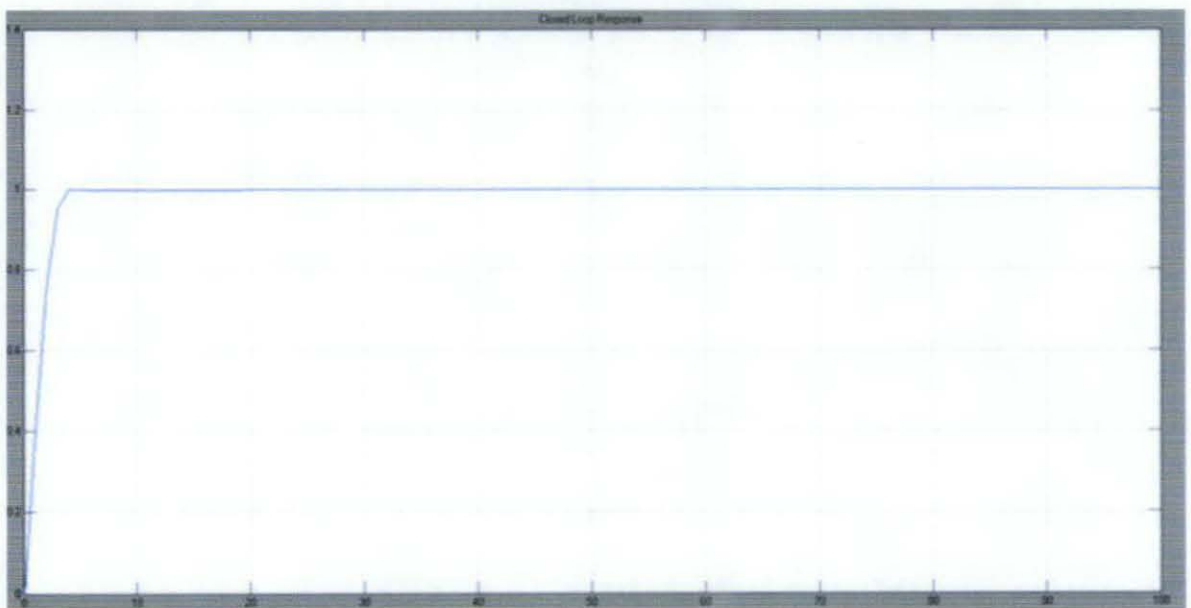


Figure 21 PSO based PID

Table 7 Comparison Result

Tuning Method	PID Parameters			Dynamic Performance Specifications		
	Kp	Ki	Kd	Tr (s)	Ts (s)	Mp (%)
PID Only	0.595	0.11309	-3.7994	9.11	28.2	9.49
PSO based PID	4.0014	0.2185	0.1683	2.3	4.8	0

The objective of this project is to minimize settling time, minimize the rising time as well as the maximum overshoot. Thus, Table 7 compares the performance of the transfer function by PID-only that is simulated using auto tune in the MATLAB, while the PSO-PID result is simulated using full coding integrated with the Simulink block applying the optimum PSO parameters. The performance of PSO based PID is improved compared to PID – only controller. The settling time and the rising time has decrease a lot, making the performance quicker, while the maximum overshoot has reduced to zero.

CHAPTER 5

Recommendation & Conclusion

5.1 Recommendation

The study can be continued to overcome the PSO weaknesses. Further research need to be done to produce an optimize technique of PSO dependable of its own parameter to improve its performance. Other than that, the dynamic performance can also be wider such as to include the steady state error and other performance parameters. An online system also can be tested with this technique to prove the relevancy of the optimization technique online.

5.2 Conclusion

Research was conducted to study the effects of using the PSO algorithm as a tool for PID tuning. From the results presented in the study it was shown that the PSO tuning yielded improved responses. The motor that was used throughout this experiment is the Axsys Technology 3625V-084 Brush DC Motor applicable for high acceleration application requiring improved response for rapid start/stop actions. The provided datasheet from the company are fully utilized to get the transfer function of the motor to be used as the plant model in this simulation

The main objective of this project; to improve the gain of PID controller and improve its performance, has been achieved. The dynamic performance of PSO based PID is much better compare to PID controller only. However, the study also revealed the weakness of PSO based PID control as well. There are too many parameters to be found by trial and error method and the output of the method is too dependable on the parameters. Once the parameters are changed, the output performance will differ and sometimes produced unstable result; the parameters are affecting the output. In a nutshell, PID controller can be integrated with another heuristic technique to improve its performance.

Appendixes

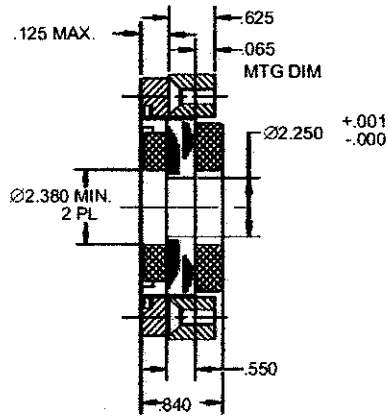
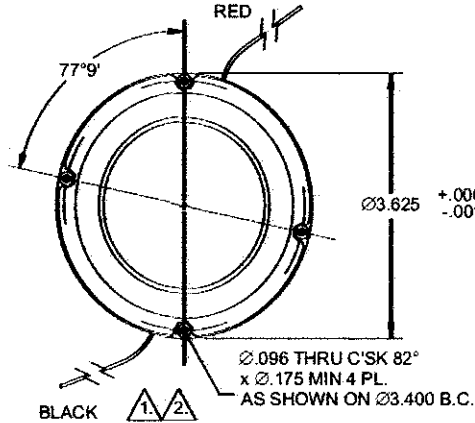
Appendix I

Datasheet for Axsys Technology 3625V-084 Brush DC Motor



Pancake Resolvers >> Brushless Motors >> (800) 777-3393

3625V-084
Brush DC Motors



Size Constants:	(all values at 25° C ambient temperature)		
	Units	Symbol	Value
Peak Torque, stalled @Vp:	oz-in	Tp	300
Power I ² R @Tp:	watts	P	184
Continuous Stall Torque	oz-in	Tcs	56.5
Motor Constant	oz-in/√W	Km	22.13
Electrical Time Constant	ms	Te	0.41
Mechanical Time Constant	ms	Tm	11.45
Damping Factor (zero impedance)	oz-in/(rad/sec)	Fo	3.458
Break Away Torque	oz-in	Tf	6.5
Rotor Inertia	oz-in-sec ²	Jm	0.0396
Theoretical Acceleration @ Tp	rad/sec ²	α _t	7576
Ripple Frequency	cycles/rev	f _R	55
Ripple Torque	% (ave to peak)	T _R	7
Theoretical No Load Speed @ Tp	rad/sec	ω _{NET}	82.5
Weight	oz	WT	15.5
Maximum Allowable Temperature	°C (at winding)	Temp.	155
Thermal Resistance	°C/W	tpr	7.3

NOTES:

- 1. LEADS: #22 AWG TYPE "ET" TEFLON COATED PER MIL-W-16878, 12 INCHES MINIMUM LENGTH.
- 2. MOTOR TO ROTATE CW VIEWED FROM BRUSH END WITH POSITIVE VOLTAGE APPLIED TO RED LEADWIRE WITH RESPECT TO THE BLACK.

Winding Constants	(all values at 25° C ambient temperature)					
	Units	TOL.	Symbol	-022	-053	-130
Resistance	ohms	+/-12.5%	R	2.17	5.25	13.0
Inductance	mH	+/-30%	L	0.89	2.14	5.32
Torque Sensitivity	oz-in/A	+/-10%	Kt	32.6	50.7	79.8
Back EMF Constant:	V/(rad/sec)	+/-10%	Kb	0.230	0.358	0.564
Peak Voltage @ Tp	Volts	Nominal	Vp	20.0	31.1	48.9
Peak Current @ Tp	Amps	Nominal	Ip	9.20	5.92	3.76

Appendix II

The m.file used to tuned the PID by PSO[18]

```
%% Initialization
clear
clc
n = 50;           % Size of the swarm " no of birds "
bird_step = 50;  % Maximum number of "birds steps"
dim = 3;         % Dimension of the problem

c2 = 0.8;        % PSO parameter C1
c1 = 0.08;       % PSO parameter C2
w = 0.7;         % pso momentum or inertia
fitness=0*ones(n,bird_setp);

%-----%
%   initialize the parameter %
%-----%

R1 = rand(dim, n);
R2 = rand(dim, n);
current_fitness =0*ones(n,1);

%-----%
% Initializing swarm and velocities and position %
%-----%

current_position = 10*(rand(dim, n)-.5);
velocity = .3*randn(dim, n) ;
local_best_position =current_position ;

%-----%
%   Evaluate initial population %
%-----%

for i = 1:n
current_fitness(i) = tracklsq(current_position(:,i));
end

local_best_fitness =current_fitness ;
[global_best_fitness,g] = min(local_best_fitness) ;

for i=1:n
globl_best_position(:,i) = local_best_position(:,g) ;
end
%-----%
% VELOCITYUPDATE %
%-----%
```

```
velocity = w *velocity + c1*(R1.*(local_best_position-
current_position)) + c2*(R2.*(globl_best_position-
current_position));
```

```
%-----%
% SWARMUPDATE %
%-----%
```

```
current_position = current_position + velocity ;
```

```
%-----%
% evaluate anew swarm %
%-----%
```

```
%% Main Loop
```

```
iter = 0 ; % Iterations' counter
```

```
while (iter<bird_step )
```

```
iter = iter + 1;
```

```
for i = 1:n,
```

```
current_fitness(i) = objfunction(current_position(:,i)) ;
```

```
end
```

```
for i = 1 : n
```

```
ifcurrent_fitness(i) <local_best_fitness(i)
```

```
local_best_fitness(i) = current_fitness(i);
```

```
local_best_position(:,i) = current_position(:,i) ;
```

```
end
```

```
end
```

```
[current_global_best_fitness,g] = min(local_best_fitness);
```

```
ifcurrent_global_best_fitness<global_best_fitness
```

```
global_best_fitness = current_global_best_fitness;
```

```
for i=1:n
```

```
globl_best_position(:,i) = local_best_position(:,g);
```

```
end
```

```
end
```

```
velocity = w *velocity + c1*(R1.*(local_best_position-
```

```
current_position)) + c2*(R2.*(globl_best_position-
```

```
current_position));
```

```
current_position = current_position + velocity;
```

```
sprintf('The value of interationiter %3.0f ', iter );
```

```
end% end of while loop its mean the end of all step that the birds
move it
```

```
xx=fitness(:,50);
    [Y,I] = min(xx);
current_position(:,I)
```

m.file for fitness function

```
function F = objfunction(pid)
% Track the output of optsim to a signal of 1

Kp = pid(1);
    Ki = pid(2);
Kd = pid(3);

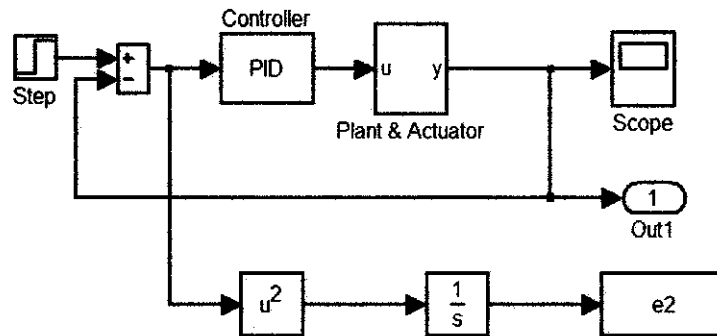
sprintf('The value of interationKp= %3.0f, Ki= %3.0f, Kd= %3.0f',
pid(1),pid(2), pid(3));
% Compute function value
simopt =
simset('solver','ode45','SrcWorkspace','Current','DstWorkspace','Current'); % Initialize sim options
    [tout,xout,yout] = sim('optsim1',[0 300],simopt);
    e=yout-1;% compute the error
sys_overshoot=max(yout)-1;% compute the overshoot

alpha=10;beta=10;
    F=e2*beta+sys_overshoot*alpha;

End
```


Simulink for plant

Tunable Variables are PID gains, Kp, Ki, and Kd.



optsiminit

Double click here to initialize plant data and optimization parameters.

Simulink Block for PSO based PID [18]

Appendix III


No.	Detail/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Selection of Project Topic	■	■												
2	Study about PID controller, Particle Swarm Optimization and DC motor			■	■	■	■	■							
3	Research on PSO concept in recent years						■	■							
4	Familiarize and understand about simulation software						■	■							
5	Submission of Extended Proposal						●								
6	Proposal Defense								■	■					
7	Simulate historic PSO data using MATLAB software										■	■	■		
8	Submission of Interim Draft Report														●
10	Submission of Interim Report														●

Process ■ Suggested milestone ●

Gantt chart for FYPI

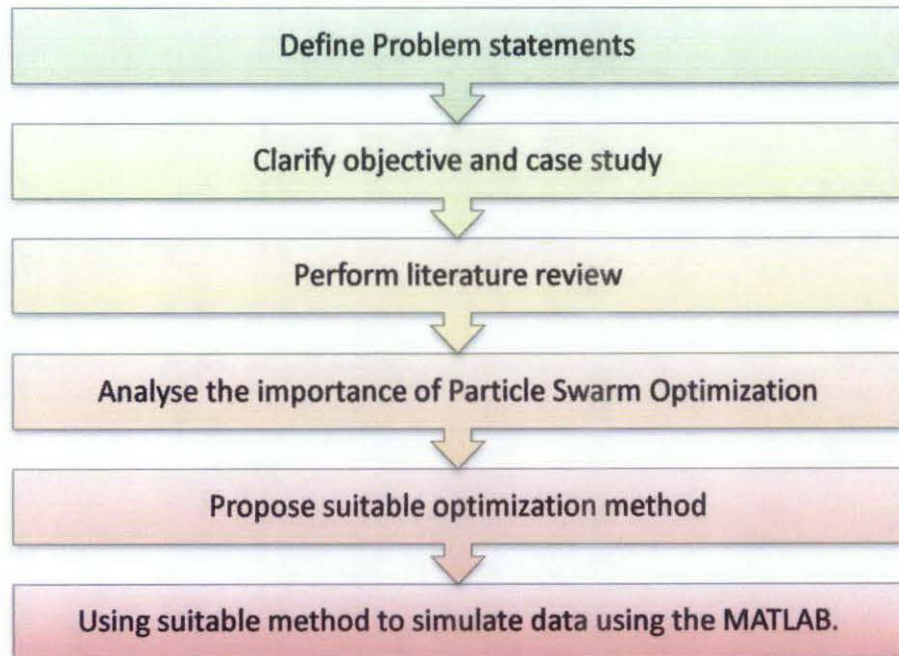
Appendix IV

No.	Detail/Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	Simulation PSO Technique using MATLAB/SIMULINK	■	■												
2	Study how to improve PID-PSO technique			■	■	■	■	■							
3	Submission of Progress Report								●						
4	Implement the improvement of PSO-PID using MATLAB/SIMULINK								■	■	■	■	■		
5	Pre-EDX											●			
6	Submission of Draft Report												●		
7	Submission of Dissertation (softbound)													●	
8	Submission of Technical Paper													●	
9	Oral Presentation														●
10	Submission of Project Dissertation (hardbound)														●

Process  Suggested milestone ●

Gantt chart for FYPII

Appendix V



Project Activity Flow

References

- [1] Adel A. A. El-Gammal and Adel A. El-Samahy, "A Modified Design of PID Controller For DC Motor Drives Using Particle Swarm Optimization PSO" POWERENG 2009 Lisbon, Portugal, March 18-20, 2009, pp.419-424
- [2] B. Nagaraj, P. Vijayakumar, "A Comparative Study of PID Controller tuning using GA, EP, PSO and ACO." Journal of Automation, Mobile robotics & Intelligent Systems, vol. 5, no.2, pp.42-48, 2011.
- [3] BoumedieneAllaoua, AbdellahLaoufi, BrahimGasbaoui, AbdelfatahNasri and AbdessalamAbderrahmani, "Intelligent Controller Design for DC Motor Speed Control based on Fuzzy Logic-Genetic Algorithms Optimization", Leonardo Journal of Sciences, Issue 13, July-December 2008.
- [4] Engelbrecht A.P., "Computational Intelligence", John Wiley and Sons, 2002
- [5] Erick Cantu' Paz et al. (Eds.), "Genetic & Evolutionary Computational." Genetic & Evolutionary Computational Conference Chicago, USA, July 2003.
- [6] Hassan R., Cohanin B., de Weck O. and Venter G., "A Comparison of Particle Swarm Optimization and the Genetic Algorithm", Structural Dynamics & Materials Conference, American Institute of Aeronautics and Astronautics, pp. 1-13, 2005
- [7] J. Kennedy and R. Eberhart, "Particle Swarm Optimization" Proceedings, IEEE International Conference on Neural Networks, vol.4, pp.1942-1948
- [8] Kennedy J., Russell R.C. and Shi Y., "Swarm Intelligence", The Morgan Kaufmann Series in Evolutionary Computation, 2001
- [9] Mehdi Nasri, HosseinNezamabadi-pour and MaliheMaliheMaghfoori, "A PSO-Based Optimum Design of PID Controller for a Linear Brushless DC Motor." Proceedings of world academy of science, Engineering & Technology, vol. 20, pp. 211-215, April 2007.
- [10] Omran M.G.M., "Particle swarm optimization methods for pattern recognition and image processing", PhD Thesis – Department of Computer Science at University of Pretoria, 2005
- [11] Pillay N. and Govender P., "A Particle Swarm Optimization Approach for Model Independent Tuning of PID Control Loop", IEEE Africon 2007.

- [12] Saffet Ayasun, Gultekin Karbeyaz, "DC Motor Speed Control Methods Using MATLAB/Simulink and Their Integration into Undergraduate Electric Machinery Courses", Wiley Periodicals Inc. 2007
- [13] Zhihua Cui, Xingjiun Cai and Jianchao Zeng, "Chaotic Performance Dependent Particle Swarm Optimization." Division of system simulation and computer application Taiyuan University of science and technology".
- [14] <http://www.eetimes.com/design/industrial-control/4014257/Brushed-DC-Motor-Fundamentals>
- [15] https://en.wikipedia.org/wiki/PID_controller#cite_note-0
- [16] <http://www.axsys.com/index.cfm?acronym=axsys-technologies>
- [17] <http://www.mathworks.com/help/toolbox/optim/ug/brn4noo.html>
- [18] http://www.mathworks.com/matlabcentral/fileexchange/20205-particle-swarm-optimization/content/PSO_Wael/PSO/PSO.m
- [19] Control System Engineering 4th Edition by Norman S. Nise, 2004