# ARTIFICIAL BEE COLONY OPTIMIZATION ALGORITHM FOR ENGINEERING APPLICATION

**AININ SOFIYA BINTI ZAKARIA**



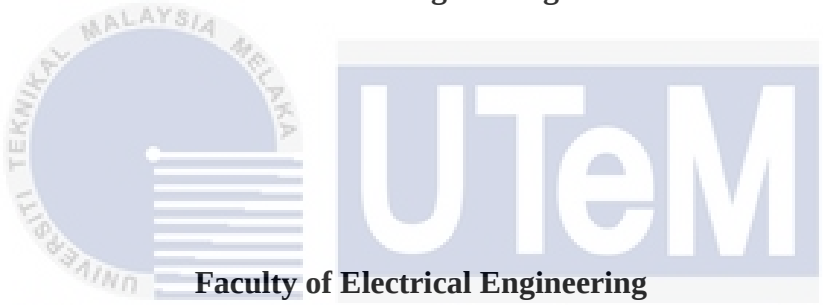
**BACHELOR OF ELECTRICAL ENGINEERING WITH HONOURS**
**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2019**

# ARTIFICIAL BEE COLONY OPTIMIZATION ALGORITHM FOR ENGINEERING APPLICATION

**AININ SOFIYA BINTI ZAKARIA**

**A report submitted**
**in partial fulfillment of the requirements for the degree of**
**Bachelor of Electrical Engineering with Honours**




**UNIVERSITI TEKNIKAL MALAYSIA MELAKA**

**2019**

## DECLARATION

I declare that this thesis entitled "ARTIFICIAL BEE COLONY OPTIMIZATION ALGORITHM FOR ENGINEERING APPLICATION is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature : .....

Name : .....

Date : .....

**APPROVAL**

I hereby declare that I have checked this report entitled “ARTIFICIAL BEE COLONY OPTIMIZATION ALGORITHM FOR ENGINEERING APPLICATION” and in my opinion, this thesis it complies the partial fulfillment for awarding the award of the degree of Bachelor of Electrical Engineering with Honours

Signature : .....

Supervisor Name : .....

Date : .....

**DEDICATIONS**

To my beloved mother and father

**ACKNOWLEDGEMENTS**

In completing this project, they are many people joint help around me who were contributed both mentally and physically. Firstly, I would like to give a sincere appreciation for their positive encouragement and guidance.

Secondly, thanks to Dr. Mohd Ruzaini Bin Hashim, my supervisor for final year project for his guidance and assistant in completing this optimization project. Without his support, this project would not presented here.

I also want to thank to my friends and UTEM's staff member for their assistant and moral support to finish this project. I also want to thank to my best friend, Aifaa for her help and encourage me finishing my report.

Finally, I would like to thank to my dearest family members. They have been encouraged and believed me in completing this project and studies. They sincerely support me in both mentally and financially.

Thank you everyone.

## ABSTRACT

This project presents the optimization technique, the artificial bee colony (ABC), was investigated in the term of finding the best optimal locations [1]. The artificial bee colony (ABC) is a new optimization technique for simulating the honey bee swarms foraging behaviour. These algorithms technique are inspired from nature. This project also proposed their performances assessment on various benchmark functions to see their robustness. In ABC, there are a few control parameter so it easy to investigate their behaviour. In this project, the three analysis are conducted to investigate their performance using 10 benchmark functions. The analysis that have been analyze in this project are number of dimension, number of population and number of iteration. The result of analysis are presented in convergence plot. For the application, flexible manipulator system (FMS) is chosen as a testing platform. The model of FMS is designed in Simulink MATLAB using ABC algorithm in tuning the proportional--integral-derivative (PID), proportional-derivative (PD) and proportional-integral (PI) controller with error criteria to investigate their performance. The most suitable controller is choosing based on overshoot, rise time, settling time and steady state error.

## *ABSTRAK*

Projek ini membentangkan teknik pengoptimuman, lebah koloni (ABC), adalah startegi dalam jangka masa mencari lokasi optimum terbaik [1]. ABC adalah teknik pengoptimuman baru untuk meniru kelakuan madu lebah. Teknik algoritma ini diilhamkan dari alam semula jadi. Projek ini juga mencadangkan penilaian prestasi mereka terhadap pelbagai fungsi penanda aras untuk melihat kekukuhan mereka. Dalam ABC, terdapat beberapa parameter kawalan supaya mudah menyiasat kelakuan mereka. Dalam projek ini, tiga analisis ini dijalankan untuk menyiasat prestasi mereka menggunakan 10 fungsi penanda aras. Analisis yang telah dianalisis dalam projek ini adalah bilangan dimensi, bilangan penduduk dan bilangan lelaran. Hasil analisis dibentangkan dalam plot penumpuan. Untuk aplikasi, sistem manipulator fleksibel (FMS) dipilih sebagai platform ujian. Model FMS direka bentuk dalam Simulink MATLAB menggunakan algoritma ABC dalam mensasarkan pengawal (PID), (PD) and PI untuk menyiasat prestasi mereka. Pengawal dipilih berdasarkan overshoot, masa naik, masa penyelesaian kesilapan keadaan mantap

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF SYMBOLS AND ABBREVIATIONS

| | | |
|---|---|---|
| ABC | - | Artificial Bee Colony |
| Kp | - | Proportional Gain |
| Ki | - | Integral Gain |
| Kd | - | Derivative Gain |
| PID | - | Proportional-Integral-Derivative |
| PI | - | Proportional-Integral |
| PD | - | Proportional-Derivative |
| tr | - | Rise Time |
| tp | - | Peak Time |
| ts | - | Settling Time |
| OS | - | Overshoot |

## LIST OF APPENDICES

# CHAPTER 1

# INTRODUCTION

## 1.1 Overview

The word optimum is in Latin and it means the best. Optimization is everywhere and it is important tool in making decisions by achieving the best possible solution of multiple applications [2]. In mathematical, an optimization approach is the process of finding the maximum or minimum values based on the optimal objective function defined in the system. Optimization can be applied in engineering problems such as optimal control, minimum processing time in production lines or design a aircraft for minimum weight. It is also can implemented in non-engineering problems such as shortest route for travelling salesman and human resource scheduling time. In daily life, we always want to optimize something in engineering field whether to maximize the output, profit, efficiency and performances or to minimize the consumption of fuel. Hence, optimization is the strategy of finding the conditions that give the maximum or the minimum value of a function in the context of mathematics. Even though there are thousands type of optimization methods but there is no single method available in solving the optimization problems efficiently and accurately.



In context of solving a optimization in application, optimization is a strategy to find the optimal solution for real world problems. The strategy to solve the real world problems is more complex and challenging. The solution is obtained from optimization algorithm is a best solution and can be used to solve a real world problems but there is no guarantee the result obtained from optimization is global minimum solution. Therefore, many research is going further to find a better optimization solution. The problem is solved through methodology that contained with data and parameters describing the objective function. Optimal value or optimal solution is a result from optimization algorithm to solve the real world problems. Example of the application in this project is analysis on flexible manipulator system.

Heuristic mean “to find” and meta means “beyond in an upper level” [3]. Metaheuristics is a set of algorithm that define search methods in solving a real world problems. It can solve a complex problems such as quadratic problems, timetable, scheduling and travelling salesman. Metaheuristics is also capable in solving in various sector of engineering such as mechanical, electrical, computer and sivil. Example of metaheuristics algorithms are artificial bee colony (ABC) algorithm, firefly algorithm, tabu search and genetic algorithm. Both exploration and exploitation can also be referred as diversification and intensification strategies. This algorithm is trendy and popular among researchers because accessible to find the optimal solution and solving a real world problems. Besides, the exploration and exploitation are harmony. When there are too much exploitation can lead to fast convergence speed but low accuracy. Meanwhile, when there are too much exploitation can award high accuracy but low convergence speed. The optimal solution is seized from metaheuristics algorithm because it have elitism element that can improve the performance of algorithms. Many researchers are attractive with this algorithm and further the hybrid research in improving a better results.

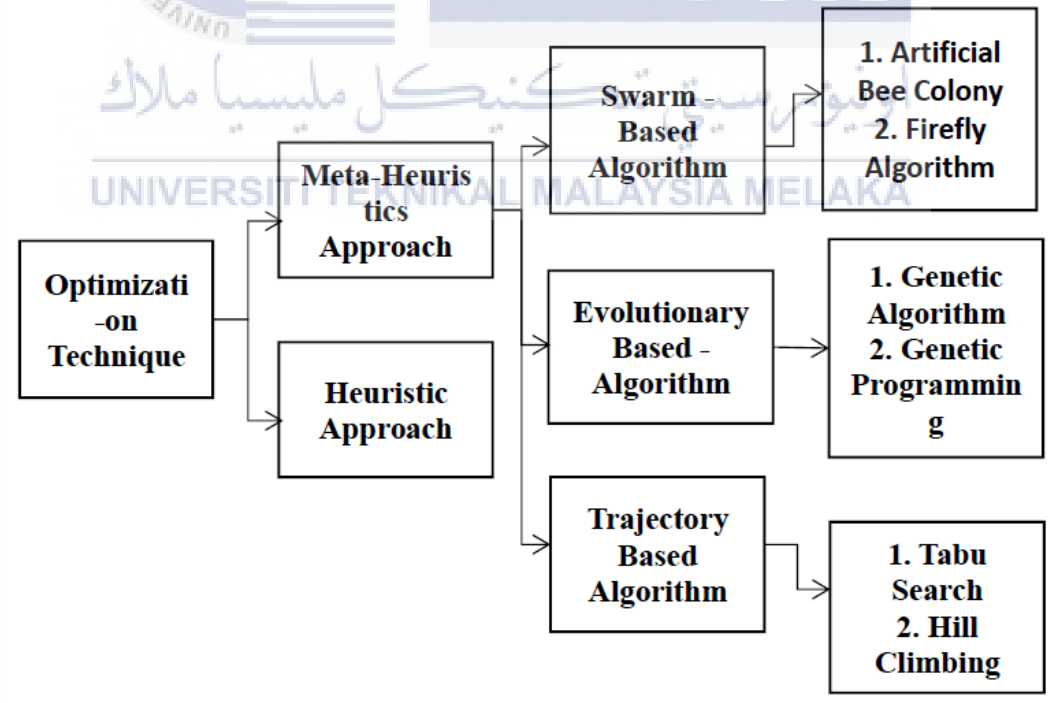



Figure 1.1: Classification of Optimization Technique

Figure 1.1 shows the classification of optimization technique that have been applied in many engineering and non-engineering problems. Swarm Intelligent (SI) is one of the type of optimization that mimic the natural biological or behaviour of species. In early 60's, many SI have been introduced. SI is a self-organized of the intelligence behaviour. Example of SI such as foraging of inserts, nest building of inserts, cooperative transportation and collective sorting. There are several type of swarm intelligent that have been developed in past several years. These algorithms include Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Differential Evolution (DE), Artificial Bee Colony (ABC), Glowworm Swarm Optimization (GSO), and Cuckoo Search Algorithm (CSA) [4]. For this project, artificial bee colony algorithm is chosen to investigate their performance on 10 numerical benchmark functions and engineering applications.

## 1.2 Project Motivation

The main motivation which give drives to investigate Artificial Bee Colony Algorithm are such as:

1. The employer want to maximize the production of products in every parts to maximize the profit but did not exceed the available parts in store.



2. In non-engineering problem, a travelling salesman is also a optimization problems. A travelling salesman need to find the shortest way between the cities and cost to travel from one city to another. This problem can be overcome by using the ABC algorithm.

3. To minimize the payroll costs, human resource department need to schedule park employees for weekly shift which is five works days plus two consecutive days off.

## 1.3 Problem Statement

There are many technique or method to be used in solving a problems such as "try and error" method but it is not guarantee to solve it. There are some of example that we have faced on daily life such as following:

1. Today there are number of optimization techniques in market to solve the several problems in engineering, economic and management. Optimization technique is a way to find the most cost effective, highest performance, highest efficiency, by maximizing desired factors and minimizing undesired other factor. So, using ABC algorithm can solve that problem.

2. In application, the robotic system is important to improve robot performance by keep the rotating angle and eliminate the oscillation angle of end effector. The position and trajectory is controlled by PID controller. So, by using the ABC algorithm can tune the PID controller to get a best performance of robot.

3. Hence, the experiment is repeated to investigate the suitable controller with error criteria in application FMS by to get a better performance between the controllers.



## 1.4 Objective

The objectives of this project as stated below:

1. To investigate the performance of ABC algorithm with parameter settings by using 10 numerical benchmark functions.

2. To investigate the suitable of controller with error criteria for application flexible manipulator system (FMS).

3. To investigate the performance of ABC algorithm with parameter settings in tuning proportional-integral-derivative (PID) with error criteria for application flexible manipulator system (FMS).

## 1.5 Scope of Research

Scopes of this project as stated below:

1. Investigate the performance of ABC algorithm in 10 numerical benchmark function by using MATLAB simulation.
   - The coding is run using a 10 different benchmark function to investigate the performance of ABC algorithm. The performance is measured by using three analysis: number of dimension, number of population and number of iteration. Every of benchmark function is run over 30 independent runs.

2. To investigate the suitable of controller with error criteria for application flexible manipulator system (FMS).
   - The ABC is tune the value of controller in flexible manipulator system. The controllers that used in this project are PI, PD and PID controller. The performance of the controllers based on their value of overshoot, rise time, settling time and steady state error.



3. To investigate the performance of ABC algorithm with parameter settings in tuning proportional-integral-derivative (PID) with error criteria for application flexible manipulator system (FMS).
   - The ABC algorithm is tune the value of PID controller in flexible manipulator system. It is to control the position and vibration of flexible manipulator system. The tuning PID controller is tune with error criteria to see ABC aalgorithm performance.

## 1.6 Report Outline

A brief description of this report is described in this section. Generally, this report contains five chapters and all these chapter will deliver the overall information about this report.

The first chapter of this report contain the introduction of this project. The overall idea of this project is briefly explained in this chapter. The objective and scope of research also explained in this chapter.

The second chapter in this report will explain the literature review of this project. The previous work related to the project will be analyzed in detail. Besides, the background of the research also need to be include.

The third chapter in this report explain the methodology that is being implemented to execute this project. All the formulas and theory used will be explained in this chapter.

The fourth chapter in this report will show the early results of the progress from the methodology used for this project. The data obtained from the results will be analyzed to verify the objective of this project. Also, this chapter show the analysis of the result.



Lastly, the fifth chapter in this report will summarize of the entire work, results of this project. The weakness, shortcomings and strength of the project are presented.

# CHAPTER 2

# LITERATURE REVIEW

This chapter reviews the basic concepts of ABC algorithm. Several ABC algorithm have been investigated. In this chapter, a ABC algorithm is the nature inspired algorithms is presented which are known as swarm intelligence. This algorithm is focused on bee colony behaviour in order to develop some meta-heuristics which has ability to solve the engineering problems. Finally, the behaviour of real honey bees have been inspired many researches to study this algorithm and their application are reviewed.

## 2.1 Theory of ABC Algorithm

Natural metaphors is a famous and trend among scientific community to model and solve the complex optimization problems. A swarm intelligence is a insert behaviour such as honey bee and firefly which can solve the problem using mimic of insert ability. Bonabeau has defined the swarm intelligence as "collective behavior of social insert colony is inspired to design algorithms or problem solving devices" [5]. ABC algorithm is proposed by a Karaboga in a very easy and simple strategy in finding a foods [3]. The ABC algorithm is a swarm based, meta heuristic algorithms proposed on foraging behaviour of real honey bees. This algorithms inspired by the intelligent of the real honey bees in finding food sources known as a nectar. They also sharing the information about the food source among other bee in the nest using dancing language. The model of foraging of the honey bees consists the three important components: food source, employed foragers and unemployed foragers.

From the previous work, ABC algorithm is the algorithms with a good performances to solve many optimization problems. ABC algorithms was initially published by Dervis Karaboga in 2005 [6]. ABC algorithm is motivated by the swarm intelligent foraging behaviour of honey bees. The ABC algorithm able to implemented in wide range of real world problems because its a population-based evolution.

In ABC algorithm, it is inspired from the intelligent behaviour of real honey bees in finding a food source that known as nectar. The algorithm has a three groups in foraging which are employed bees, onlooker bees and scouts. They have given a specific task in foraging activities. Honey bees used a dancing language as a communication behaviour among them. ABC algorithm is a attractive than other optimization algorithms because it has a following characteristics. This algorithm has a few control parameters. The control parameter that ABC algorithm are the population size, limit and maximum cycle number. Besides, it also flexible, simple and fast convergence speed. This algorithm also easy to hybrid with other optimization algorithms.

### 2.1.1 The Flow Chart of ABC Algorithm

The figure 2.1 shows the flow chart of ABC algoirthm. The phases of employee bee, onlooker and scout bee will expain in detail in the next sub-section.



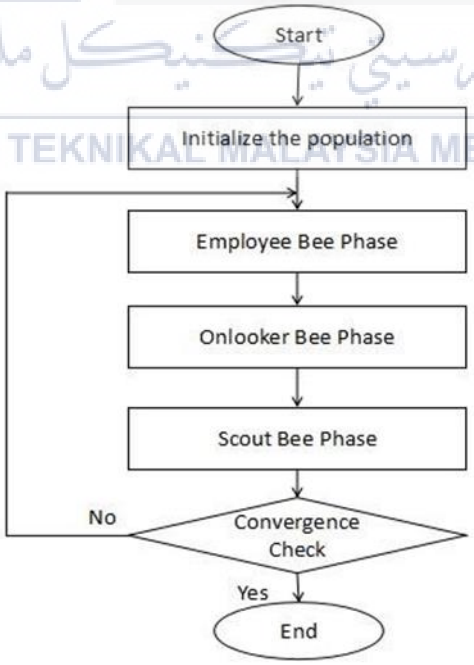



Figure 2.1: The Flow Chart of ABC Algorithm

### 2.1.2 Phases of ABC Algorithm

The working principle have been approached. The investigation process of ABC has three steps:

a) Employed bee as a search agent. Sending the employed bees to a food source and memorize the amount the food source which is nectar.

b) Employed bee will share the information of the food source with the onlooker bees in the nest. The onlooker bees will calculate the approximately their quality of nectar.

c) Scout as a replace agent. Sending the scout bees to the possible food sources.

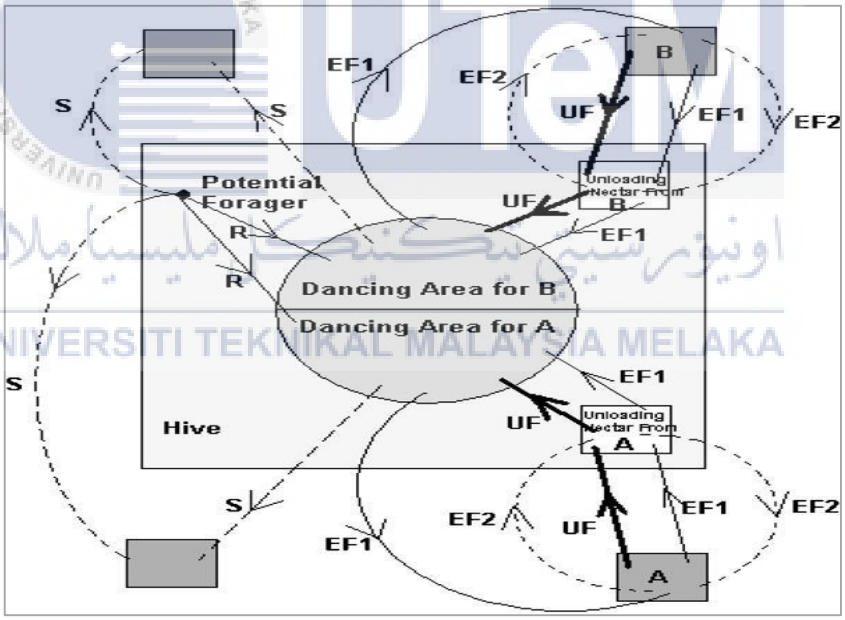



Figure 2.2: Foraging of Honey Bees [7]

### 2.1.3 Food Source

Food source for a honey bees represent the possible solution of the optimization problem. The concentration of its energy, proximity to the nest and extracting the

energy are the factors that effect the value of food source. As for a test case, nectar amount correspond to the quality (fitness) of the solution. The ABC algorithms that have a extraordinary foraging behaviour of real honey bees has three important constraints. The first constraint is the population which is the number of food source. The second constraint is a limit which is number of tries when the employed bee reject the foos source. The third constrain is the iteration, the maximum number is the criteria to stop the process of foraging. The location of food sources are randomly initialize according to the equation:

$$x_y = x_j^{min} + rand(0,1)\left(x_j^{min} - x_j^{min}\right) \quad (2.1)$$

Where i = 1 and j = 1; y = ij; $x_j^{min}$ is lower bound and $x_j^{max}$ is upper bound of xij.

### 2.1.4 Employed Bee Phase

The second phase of ABC algorithm is employed bee phase. The employed with them the information of nectar about amount of nectar, distance and direction from their nest. They shared the information with onlooker bees using the dancing language. The food source depends on the employed bees. In other words, the number of food source equal to the employed bees for the hive. The probability to be selected by employed bees increases with the increases the nectar quality. After sharing the information of food source with onlooker bees, employed bees return to the food source from the previous cycle to select the another food source. The employed bees that have a information of food source with the highest nectar quality recruits the onlooker to get the nectar. The resulting food source is generated according to the equation below.

$$v_{ij} = x_{ij} + \emptyset\left(x_{ij} - x_{kj}\right) \quad (2.2)$$

Where: k is a neighbour of i, i ≠ k; ɵ is a the range [-1,1] as a control the production of neighbour solutions; vij is a new solution for xij. The fitness of the new food source as followed:

$$fit = \begin{cases} \frac{1}{1+f_i}, f_i \geq 0 \\ a + abs(f_i), f_i \leq 0 \end{cases} \quad (2.3)$$

Where fi is the objective function with each food source while fiti is the fitness value.

Figure 2.3: Dancing Language of Honey Bees (Waggle) [8]



The waggle dance as a communication system between the bees have been adapted to scientific problems for optimization. These communication contributes to the social bee colonies. By performing this dance, the forager which is employed bee successful share with the onlooker bees the amount of nectar and the information of direction the food source from hive. From the previous studies on waggle dancing, the direction of the waggle refers to the direction of the food source in relation to the sun and the duration of the dance refers to the amount of nectar on related food source.

### 2.1.5 Onlooker Bee Phase

The third phase of ABC algorithms is onlooker bee phase. The onlooker bees have the same number as a employed bees: 50% of employed bees and 50% of onlooker bees. The task that given to onlooker bees are to observe the dance of the employed bees to select the information of food source. The information that onlooker received from employed bee are food source position and nectar quality. The below show the equation of onlooker bees select the food source:

$$p_i = \frac{fit_i}{\sum_{j=1}^{N} fit_j} \tag{2.4}$$

Where $fit_i$ is the best fitness value of i-th solution and $p_i$ is selection probability of i-th solution.

### 2.1.6 Scout Bee Phase

Scout bees are free bees responsible to find the food sources and even to evaluate their nectar. The scout bee become the employed bee when they find the food source. In this phase, the scout is chose from one of the employed bees. The limit is a control parameter to control the selection of the scout. The bees find the new food source when there is no improvement in quality of nectar which is fitness and they abandoned old food source. When the employed bee turned to a scout, the number of releasing of food source is equal to the value of limit. The limit is one of the important control parameter in ABC algorithm. This algorithm assume that number of employed bee equal to number of food source around their hive.

## 2.2 Flexible Manipulator System (FMS)

### 2.2.1 Introduction to FMS

A flexible manipulator system or FMS is an arrangement of machines interconnected by a transport system. A schemetic diagram of a FMS rig is shown in Figure 2.4. The system consists of a single flexible link made of aluminium beam and attached to an electro-mechanical motor. The function of U9M4AT type printed circuit board is to rotate the motor shaft in both counterclockwise and clockwise direction [9]. The system have the three outputs. To measure the end-point acceleration, piezoelectric accelerometer is placed at the tip of the beam. The advantages of the sensor are light, small, has high voltage sensitivity and low impedance outputs. The low impedance outputs prevent the losses of signal and distortion problems. An encoder with 2,048 resolution is attached to measured the hub angle. The tachometer also attached to measure the hub-velocity.

Figure 2.4 shows the schematic representation for the mechanical structure of the FMS. The *(t)* is the hub angle or angular displacement, is the input torque *mp* is the payload attached at the tip of the flexible beam and *u (x, t)* is the elastic deflection of an arbitrary point along the flexible beam with a distance of *x* mm from the hub.



Manipulators are modelled as rigid robots, flexible joint robots. It is challenging task in modelling the flexible manipulator system. The challenge task is due to the vibration behaviour and non-linear characteristics and lead to complex mathematical models but it is important because the information about behaviour of system is essential. The mathematical model of flexible manipulator system divided into analytical and system identification approaches.

Flexible arm robot systems are motivated for better arm weight to payload ratios, shorter time moving and lower energy consumption. Some of the advantages are have small actuators, safe operation and have low cost. The positioning of flexible manipulator is difficult to maintain their accuracy. The some problem are because of the vibration of the system, precise positioning requirements and modelling the

system. The controller is used to track a reference signal, reduce the vibration and maintain accurate end-point positioning.

In this project, PID controller is used to overcome the main weakness of the controlling methods to ensure robustness against parameters by using ABC algorithm. The main of the optimization is to confirm the stability of a system. The PID controller is regulated the control loop.

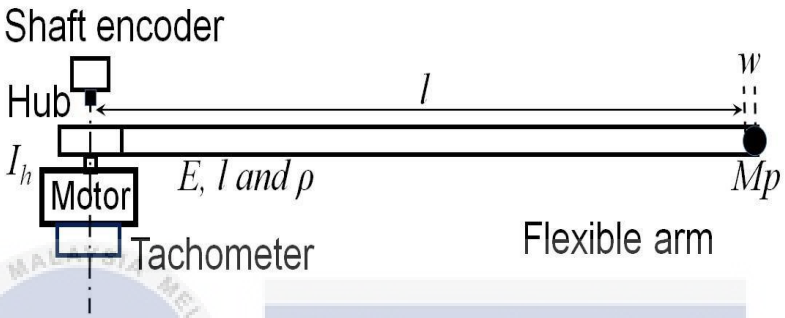



Figure 2.4 Schematic Diagram of FMS Rig

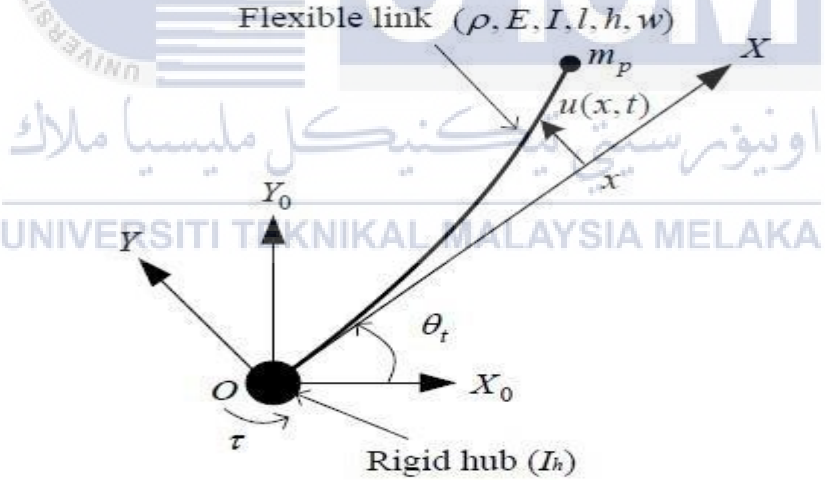



Figure 2.5: Schematic Representation Of The Mechanical Structure (Adapted From: Azad, 1994)

Table 2.1: Physical Parameters of Flexible Manipulator System

| Physical parameters | Value |
|---|---|
| Width (*w*) | 19.008 mm |
| Length (*l*) | 960 mm |
| Thickness (*h*) | 3.2004 mm |
| Hub inertia (*Ih*) | 5.86 x 10-4 Kgm2 |
| Moment of inertia (*Ib*) | 0.04862 Kgm2 |
| Second moment of inertia (*I*) | 5.1924 x10-11 m4 |
| Young modulus (*E*) | 71x109 Nm-2 |
| Mass density per unit volume (*p*) | 2,710 Kgm-3 |

### 2.2.2 Artificial Bee Colony Optimization

The artificial bee colony optimization is a evolutionary that showed the social behaviour of honey bees seeking the richest food source [10]. The communication and memories between honey bees are employed to boost search efficiency and robustness. This algorithm initialize the population randomly to detect the global minimum to achieve objective function. The advantage of ABC algorithm is this algorithm is stuck at the local minimum.



In this project, ABC algorithm is used to explore the performance of flexible manipulator system. Figure 2.6 shows the PID tuning by using ABC algorithm. The purpose of ABC algorithm chosen to tune the parameters of the PID controller by using objective function. Furthermore, PI and PD controller also tune the parameter with error criteria in order to see their performance.

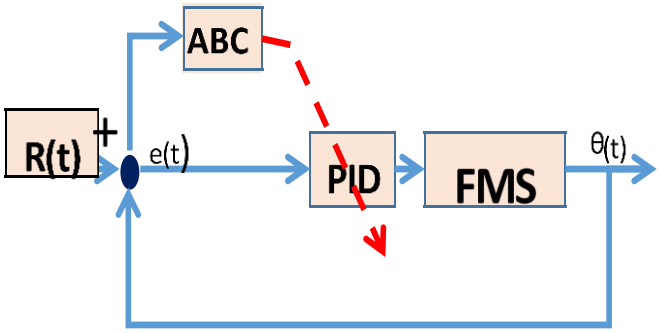



Figure 2.6: PID Tuning by using ABC Algorithm

### 2.2.3 Control of FMS

To control this FMS, the output is hub angle and end-point acceleration. Distributed parameter system of infinite order, vibrational and non-minimum phase behaviour is challenging because of the distributed flexibility of its link. The FMS has the ability to control unwanted signal and unwanted disturbance or noise in the operation of the system. The hybrid control technique is famous approach to control FMS.

## 2.3 PID Characteristics Parameters

PID controller is generally used in industrial control systems and any other real world applications. A proportional, integral and derivative are applied to correct error value e (t). Error value is the difference between a reference value r (t) and a system output y (t). The figure 2.7 shows the parallel PID controller that widely used for control system such as high order systems.

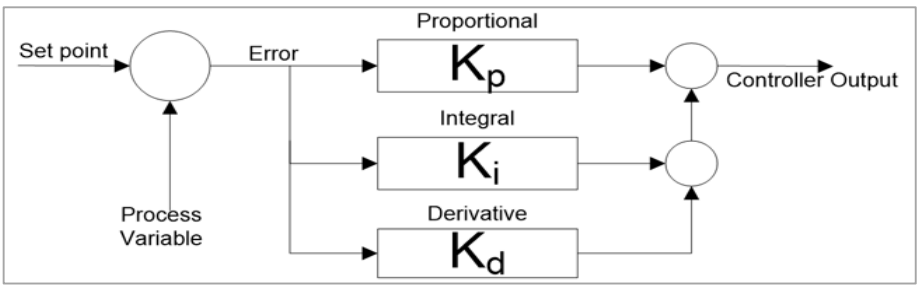



Figure 2.7: Block Diagram of a PID Controller

The proportional controller is the one to control error. By increasing the value of Kp, the steady state error will reduce and time rise also reduce. Hence, there a disadvantage by increasing the Kp. The system will oscillate due to large overshoot produced by large Kp. Integral control is added to the system to eliminate the steady state error. The elimination of steady state error will make the transient response of the system worse. In order to stable the system, derivative is added that effects in reducing overshoot, increasing damping and improving transient response.

Table 2.2 shows the effect of changing the parameter of controller. The table also shows the value of transient response of different controller is changing based on value of parameter of controller. The main characteristics of the output of the system are rise time, overshoot, settling time and steady state error.

Table 2.2: Effect of Changing the Parameter of Controller

| **Parameter** | **Rise Time** | **Overshoot** | **Settling Time** | **Steady State Error** |
|---|---|---|---|---|
| Kp | Decrease | Increase | Small Change | Decrease |
| Ki | Decrease | Increase | Increase | Eliminate |
| Kd | Small Change | Decrease | Decrease | No change |



- Rise time (tr) is defined as time required for the step response to rise from 10% to 90% of the set point.
- Settling time (ts) is defined as the time required for the response to stay within 2% of the set point.
- Overshoot (Mp) is defined as a maximum peak over the set point.
- Steady state error is defined as different between the process variable and the set point.

**Error criteria**

Table 2.3 shows the error criteria used in tuning controller in this project. The error is calculated by different between input step and output system.

Table 2.3: Error Criteria and Their Formula

| Error Criteria | Formula |
|---|---|
| Integral Absolute Error (IAE) | $\int_0^\infty \|e(t)\|dt$ |
| Integral Time Absolute Error (ITAE) | $\int_0^\infty t\|e(t)\|\,dt$ |
| Integral Square Error (ISE) | $\int_0^\infty e(t)^2 dt$ |
| Mean Square Error (MSE) | $\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{y})^2$ |
| Root Mean Square Error (RMSE) | $\sqrt{\frac{1}{n}\sum_{i=1}^{n}\|e_i^2\|}$ |



## 2.4 Related Work

### 2.4.1 The Performance of Artificial Bee Colony (ABC) Algorithm

Author in the paper [10] is explaining the foraging behaviour of honey bees. The scope of this paper to compares the performance of ABC algorithms with of differential evolution (DE), particle swarm optimization (PSO) and evolutionary algorithms (EA) for numeric problems. The simulation is result is compared with other algorithm mentioned before. The ABC algorithm can solve the optimization problems. It also simulate the ABC algorithm with different benchmark functions such as Schaffer, Sphere, Griewank, Rastrigin and Rosenbrock.

### 2.4.2 Algorithm for Solving Constrained Optimization Problem

The paper [11] write by Dervis Karaboga and Bahriye Basturk focused on performances ABC algorithm for constrained optimization problems. The ABC algorithm is successful solving the constrained optimization problems in this paper. They used 13 different numericalbenchmark functions to see the performance of ABC algorithm. The benchmark function includes linear, non-linear and quadratic. The colony size that have been used is 40 and maximum cycle number is 6000. The experiment were repeated 30 times. They concluded that ABC algorithm can be used for solving constrained optimization problems.

### 2.4.3 Optimization of Benchmark Functions Using Artificial Bee Colony (ABC) Algorithm

The paper [12], written by Dr. Dharmender Kumar and Balwant Kumar presented a topology of ABC algorithm (RABC). The result of RABC is successful improve in term of robustness, convergence speed and quality of the solution. They used population size is 100 and number of iteration is 100 for this experiment. The RABC and original ABC were compared using 5 benchmark functions. The benchmark function mentioned included Sphere, Rosenbrock, Griewank, Schwefel and Rastrigin. From the table 2.1, RABC algorithm is better than ABC on all functions. They found that RABC algorithm has the ability to far from local minima and trap to global optimum.



Table 2.2 Comparison result between ABC and RABC algorithm

| Function | Statistic | ABC | RABC |
|---|---|---|---|
| Sphere | mean | 7.07666 | 0.000216354 |
| Griewank | mean | 0.953976 | 0.038093 |
| Rosenbrock | mean | 8237.68 | 158.428 |
| Schwefel | mean | 7302.69 | 6451.49 |
| Rastrigin | mean | 131.212 | 59..6629 |

# CHAPTER 3

# METHODOLOGY

Chapter 3 is explained about methodology used in this project. For achieving the objectives mentioned before in chapter 1, the flow chart are added to show the method. The analysis of this project are completed using the MATLAB Simulink. Gantt chart for the whole project is showed in appendix

## 3.1 Methodology to Achieve First Objective

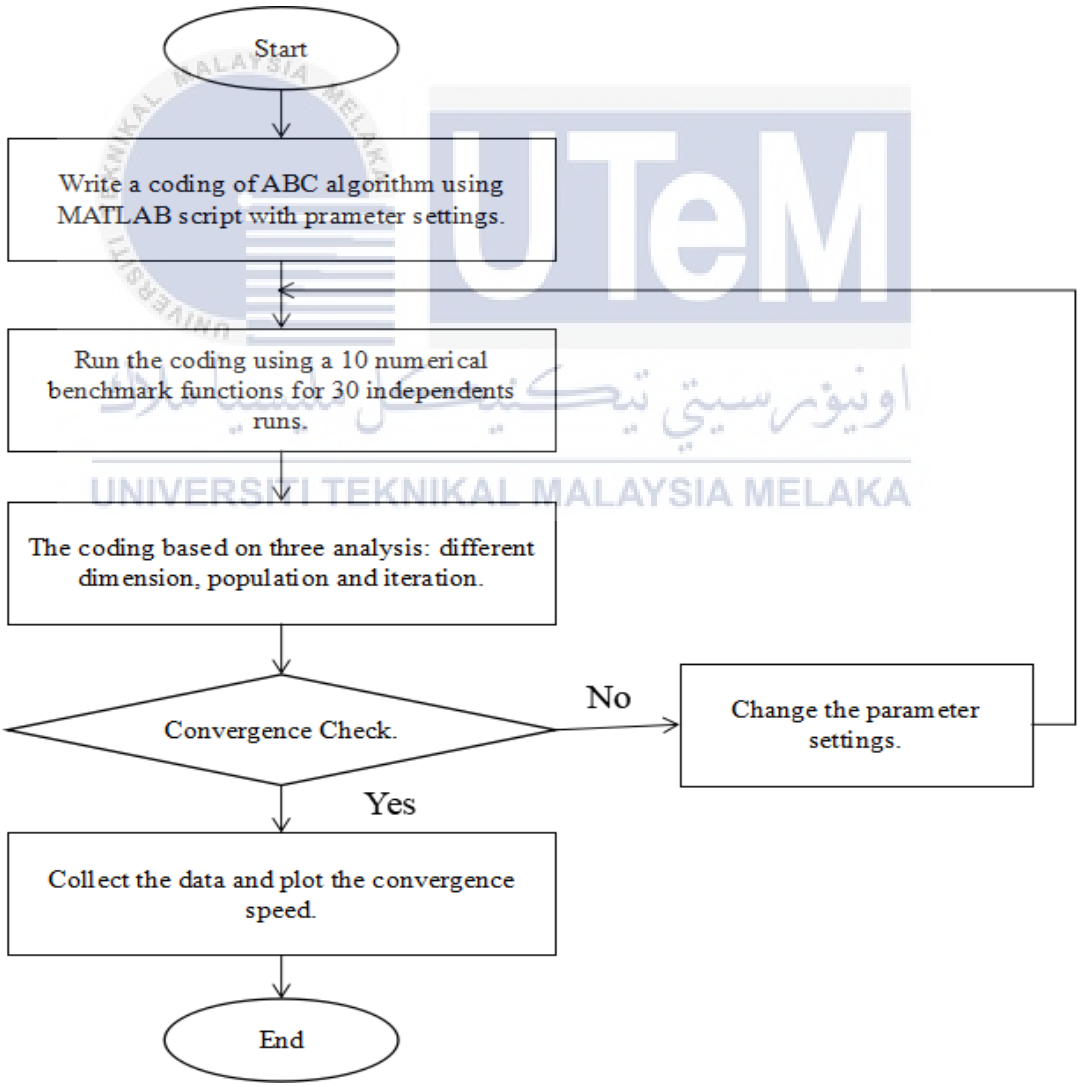



Figure 3.1: Methodology of First Objective

The figure 3.1 showed the flow chart of methodology in achieving the first objective of this project. The purpose of this methodology is to investigate the performance of ABC algorithm using 10 numerical benchmark functions. The benchmark function act as a test performance platform to investigate the robustness. The benchmark functions that used in this project are booth, bukin, six-hump camel, cross-in-tray, easom, eggholder, holder, matyas, shubert, and sum of different powers. The step involved are as below:

i. Write a coding of ABC algorithm in MATLAB script.
ii. 10 numerical benchmark function as a objective function is chosen randomly to see the convergent speed.
iii. The performance of ABC algorithm is measured and analyzed with three parameter setting which are different number of dimension, population (bee) and iteration. Each of analysis with parameter setting is run about 30 independent runs.
iv. After finishing 30 independent runs, the data is collected and presented in graph convergence speed to see their performance.

### 3.1.1 Benchmark Function

Benchmark function is the platform to investigate the performance optimization approach. The table 3.1 shows the 10 benchmark function used in this experiment. The table also shows the formula, range and minimum of the functions.

Table 3.1: Benchmark Function and Their Parameters

| Function | Formula | Range |
|---|---|---|
| Booth | $f_1(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$ | [-10,10] |
| Bukin | $f_2(x) = 100\sqrt{\lvert x_2 - 0.01x_1^2 \rvert} + 0.01\lvert x_1 + 10 \rvert$ | [-15,-5] |
| Six-Hump Camel | $f_3(x) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2$ | [-3,3] |
| Cross-in-Tray | $f_4 = -0.001(\left\lvert \sin(x_1)\sin(x_2)\exp(\left\lvert 100 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right\rvert\right\rvert + 1$ | [-10,10] |
| Easom | $f_5 = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)2 - (x_2 - \pi)2)$ | [-100,100] |
| Eggholder | $f_6 = -(x_2 + 47)\sin(\sqrt{\left\lvert x_2 + \frac{x_1}{2} + 47\right\rvert} - x_1\sin(\sqrt{(x_1 - (x_2 + 47)}$ | [-512,512] |
| Holder | $f_7 = -\left\lvert \sin(x_1)\cos(x_2)\exp(\left\lvert 1 - \frac{\sqrt{x_1^2 + x_2^2}}{\pi}\right\rvert\right\rvert$ | [-10.10] |
| Matyas | $f_8 = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$ | [-10.10] |
| Shubert | $f_9 = (\sum_{i=5}^{5} icos((i+1)x_1 + i))(\sum_{i=5}^{5} icos((i+1)x_2 + i))$ | [-10,10] |
| Sum of Different Powers | $f_{10} = \sum_{i=1}^{d} \lvert x_i \rvert^{i+1}$ | [-1,1] |



### 3.1.2 Parameter Settings

Parameter setting is control parameter used to see the performance of ABC algorithm. Table 3.2 shows the parameter settings in order to achieve second objective of this project. For this project, there are three analysis to investigate the performance of ABC algorithm in term of convergence speed. The control parameter used are different number of dimension, population and iteration.

Table 3.2: Parameter Setting Used For Analysis

| Variables | Dimension | Population | Iteration |
|---|---|---|---|
| 1. Dimension | 2 | 10 | 1000 |
| | 10 | | |
| | 30 | | |
| | 50 | | |
| 2. Population | 2 | 10 | 1000 |
| | | 20 | |
| | | 40 | |
| | | 80 | |
| 3. Iteration | 2 | 10 | 50 |
| | | | 100 |
| | | | 500 |
| | | | 1000 |

## 3.2 Methodology to Achieve Second Objective

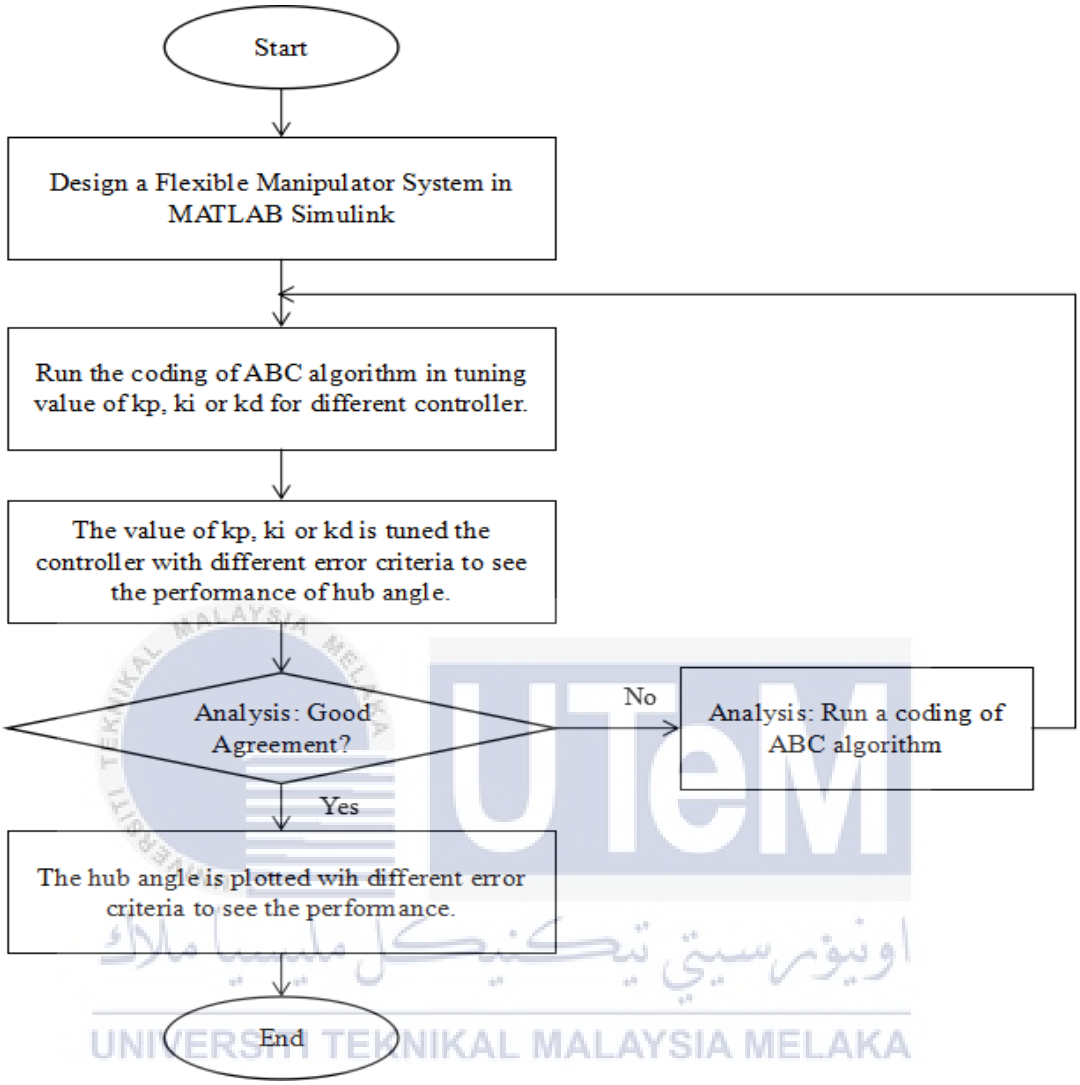





Figure 3.2: Methodology of Second Objective

The figure 3.2 shows the flow chart in achieving the second objective. The purpose of this methodology to investigate the suitable controller for application flexible manipulator system (FMS). The controller that chosen are proportional-integral-derivative (PID), proportional- derivative (PD) and are proportional-integral (PI) controller. The experiment of application FMS is conducted with different error criteria. The best result is chosen by the minimum overshoot, rise time and settling time. The steps involved state as followed:

i. The model of FMS is designed in MATLAB Simulink with error criteria.

ii. Run the coding of ABC algorithm with control parameter such as number of population, dimension and iteration in tuning value of kp, ki and kd.

iii. The value of kp, ki and kd are tuned in controller for application FMS with error criteria to track the output.

iv. The output is obtained from hub angle after run the model of FMS.

v. The value of overshoot, rise time, settling time and steady state error are obtained from the hub angle.

## 3.3 Methodology to Achieve Third Objective

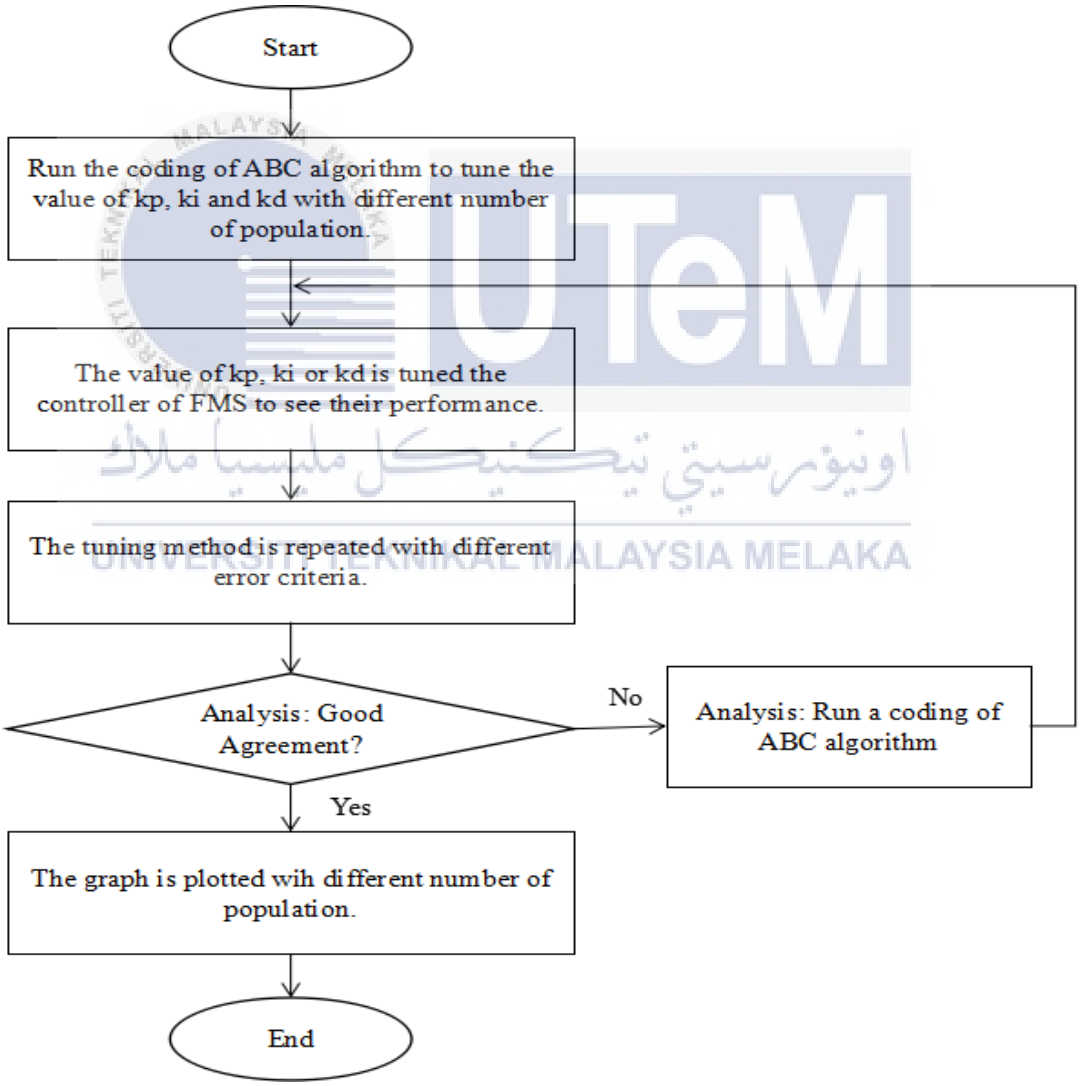



Figure 3.3: Methodology of Third Objective

The figure 3.3 shows the flow chart in achieving the third objective. The purpose of this methodology is to investigate the performance of ABC algorithm in tuning PID controller for FMS. The steps involved as followed:

i. Run the coding of ABC algorithm with different number of population to tuned the value of kp, ki and kd.
ii. The values that obtained is tuned in PID controller to see the performance.
iii. The experiment is repeated with different error criteria.
iv. The graph of different number of population is plotted in the same graph to compare their performance for each error criteria.

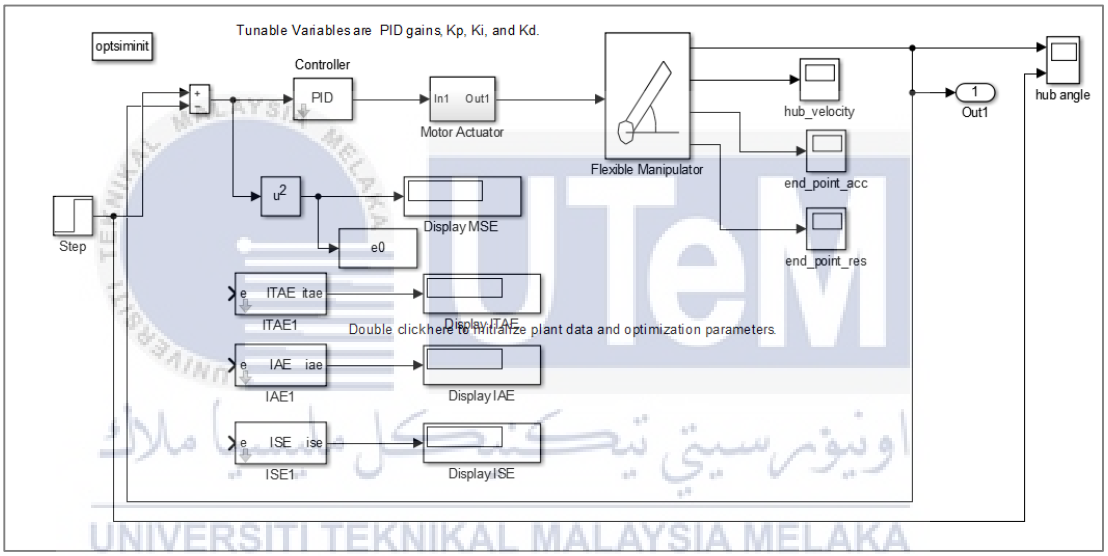



Figure 3.4: Simulink of Tuning Variable of PID

# CHAPTER 4

# RESULTS AND DISCUSSIONS

In this chapter, the results from first, second and third objectives are showed and discussed in detail. For first objective, the performance of ABC algorithm is investigated using 10 benchmark functions to measure their robustness. For second objective, the PID controller with criteria is tuned repeatly to see their performance. For third objective, the PI controller, PD controller and PID controller with error criteria are tuned for application flexible manipulator system (FMS). MATLAB Simulink is used in completing all the objective that mentioned before in Chapter 1.

## 4.1 Performance of ABC Algorithm with Parameter Settings using 10 Benchmark Function

The performance of ABC algorithm is investigated using 10 different benchmark function with parameter settings. The benchmark function with uni-modal and multi-modal features are used as test performance to see their robustness. The performance of ABC algorithm is tested by three analysis. The parameter settings that used to see their performance are using different number of dimension, number of population and number of iteration. The graph is represented the convergence speed of the algorithm. The mean values showed in the table are represented the best solutions of the algorithm. The graph with minimum value which is zero values is the best solutions for different parameter settings used in this test performance.



### 4.1.1 First Analysis: Number of Dimension

Different number of dimension is chosen as a first analysis to investigate the performance of ABC algorithm. The different dimensions for this experiment are 2, 10, 30 and 50. The different dimensions are chosen based on previous works by Karaboga. For other parameter used for this analysis is number population is 10 and number of iteration is 1000. The limit control is 100. The experiments is repeated 30

independent runs in MATLAB. The mean values referred as best solution and standard deviation have been recorded obtain from the independent runs. The time showed in the table refer to time processing for each cycle in unit second. The result showed by incresing the number of dimension, the longer the time will take to finish a cycle.

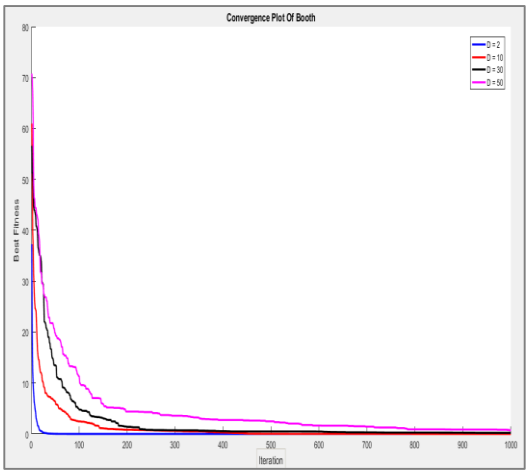
Table 4.1 showed the result of different dimension using 10 numerical benchmark function. The data of mean values, standard deviation, best value, worse value and time processing in second are recorded in table to show their performance. The analysis is showed that by increasing the dimension, the mean values are become larger. It is because of slow in convergence speed and away from local minimum. Hence, the ABC algorithm with dimension of 2 is faster in convergence speed than a dimension of 50 for all bencmark functions. *F10* which is Sum of Different Powers function is the accurate and fast in term of convergence speed. The mean and standard deviation of *f1* are 1.864E-18 and 2.092E-18 respectively for dimension of 2. The *f7* which is holder function is the worst bencmark function as a test performance because it is away from local minimum and slow convergence speed. The mean and standard values are -9.139E18 and 5742.4E00 respectively for dimension of 2. The figure 4.1 showed the convergence speed of different dimension using 10 numerical benchmark function.



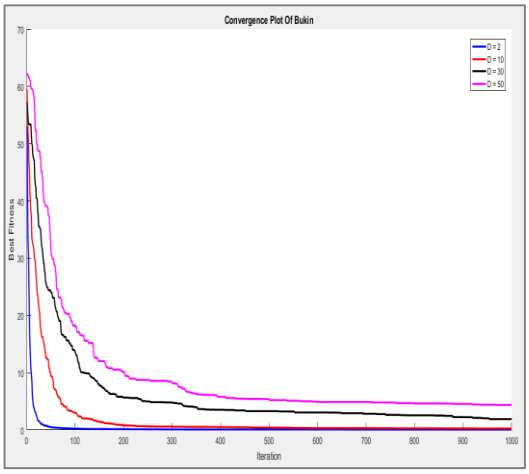
Table 4.1: The Result of Different Dimension using 10 Benchmark Function

| **Function** | **Accuracy** | **Number of Dimension** | | | |
|---|---|---|---|---|---|
| | | **2** | **10** | **30** | **50** |
| *f1* | Mean | 1.542E-16 | 2.117E-03 | 2.225E-01 | 1.736E-01 |
| | Stan. Dev | 1.643E-16 | 8.157E-03 | 1.942E-01 | 1.275E00 |
| | Best | 7.510E-18 | 1.780E-09 | 8.260E-03 | 1.111E-03 |
| | Worse | 7.650E-17 | 0.192E-02 | 8.534E-01 | 5.944E00 |
| | Time (s) | 1.008E00 | 9.541E-01 | 8.239E-01 | 8.125-01 |
| *f2* | Mean | 7.989E-02 | 2.689E-01 | 1.894E00 | 4.369E00 |
| | Stan. Dev | 1.669E-02 | 1.812E-01 | 1.434E00 | 3.113E00 |
| | Best | 7.000E-02 | 1.073E-01 | 2.490E-01 | 8.178E-01 |
| | Worse | 1.114E-01 | 7.062E-01 | 6.137E00 | 13.439E00 |
| | Time (s) | 8.939E-01 | 1.778E-01 | 8.266E-01 | 8.139E-01 |
| *f3* | Mean | -1.031E00 | -1.031E00 | -1.031E00 | -1.018E00 |
| | Stan. Dev | 0.000E00 | 8.187E-10 | 2.059E-03 | 1.396E-02 |
| | Best | -1.031E00 | -1.031E00 | -1.029E00 | -0.967E-01 |

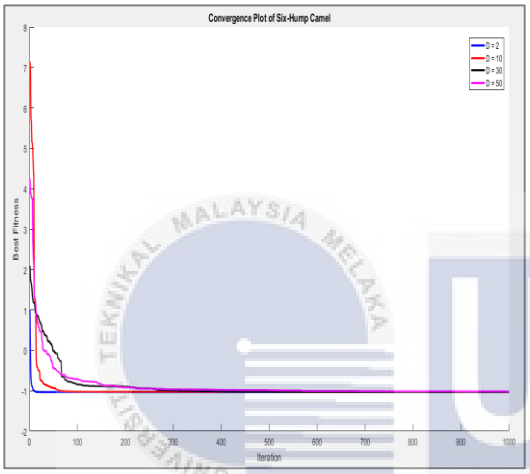
| | | | | | |
|---|---|---|---|---|---|
| | Worse | -1.031E00 | -1.031E00 | -1.031E00 | -1.030E00 |
| | Time (s) | 9.443E-01 | 0.987E-01 | 1.198E00 | 1.396E00 |
| *f4* | Mean | -2.062E00 | -2.062E00 | -2.062E00 | -2.062E00 |
| | Stan. Dev | 0.000E00 | 1.316E-11 | 4.056E-04 | 3.137E-03 |
| | Best | -2.062E00 | -2.062E00 | -2.062E00 | -2.060E00 |
| | Worse | -2.062E00 | -2.062E00 | -2.0632E00 | -2.063E00 |
| | Time (s) | 8.507E-01 | 8.243E-01 | 8.907E-01 | 9.271E-01 |
| *f5* | Mean | -8.178E-01 | -3.336E-01 | -5.040E-02 | -8.157E-02 |
| | Stan. Dev | 3.484E-01 | 4.174E-01 | 1.576E-01 | 2.018E-01 |
| | Best | -2.130E-15 | 0.000E00 | 0.000E00 | 0.000E00 |
| | Worse | -1.000E00 | -9.626E-01 | -3.580E-05 | -3.040E-05 |
| | Time (s) | 8.494E-01 | 9.558E-01 | 8.975E-01 | 9.046E-01 |
| *f6* | Mean | -943.0E00 | -931.6E00 | -887.9E00 | -842.5E00 |
| | Stan. Dev | 25.268E00 | 24.09E00 | 42.61E00 | 78.27E00 |
| | Best | -888.6E00 | -894.5E00 | -820.6E00 | -660.2E00 |
| | Worse | -890.5E00 | -955.8E00 | -959.2E00 | -950.3E00 |
| | Time (s) | 1.031E00 | 1.019E00 | 1.154E00 | 1.290E00 |
| *f7* | Mean | -9.139E18 | -9.139E18 | -9.005E18 | -8.099E18 |
| | Stan. Dev | 5742.4E00 | 5.923E12 | 2.881E17 | 1.278E18 |
| | Best | -9.140E+18 | -9.140E+18 | -7.620E+18 | -5.700E+18 |
| | Worse | -9.140E+18 | -9.140E+18 | -9.140E+18 | -9.140E+18 |
| | Time (s) | 1.336E00 | 1.406E00 | 1.420E00 | 1.203E00 |
| *f8* | Mean | 3.232E-09 | 2.567E-03 | 5.071E-01 | 1.716E-02 |
| | Stan. Dev | 5.798E-09 | 2.957E-03 | 6.448E-03 | 1.981E-02 |
| | Best | 4.060E-15 | 4.850E-06 | 1.228E-02 | 6.510E-05 |
| | Worse | 1.440E-08 | 3.779E-03 | 2.318E-04 | 8.507E-02 |
| | Time (s) | 1.052E00 | 1.108E00 | 1.278E00 | 1.586E00 |
| *f9* | Mean | -186.7E00 | -186.6E00 | -180.3E00 | -157.8E00 |
| | Stan. Dev | 0.000E00 | 9.670E-02 | 6.916E00 | 25.82E00 |
| | Best | -186.7E00 | -186.3E00 | -156.1E00 | -87.93E00 |
| | Worse | -186.7E00 | -186.7E00 | -186.4E00 | -186.6E00 |
| | Time (s) | 1.183E00 | 1.209E00 | 1.124E00 | 1.309E00 |
| *f10* | Mean | 1.864E-18 | 4.063E-17 | 1.427E-10 | 6.434E-08 |
| | Stan. Dev | 2.092E-18 | 1.621E-17 | 5.170E-10 | 1.756E-07 |
| | Best | 5.820E-20 | 8.650E-17 | 2.020E-15 | 1.030E-11 |
| | Worse | 1.050E-18 | 2.420E-17 | 1.870E-10 | 9.010E-07 |
| | Time (s) | 9.975E-01 | 1.187E00 | 1.199E00 | 1.209E00 |



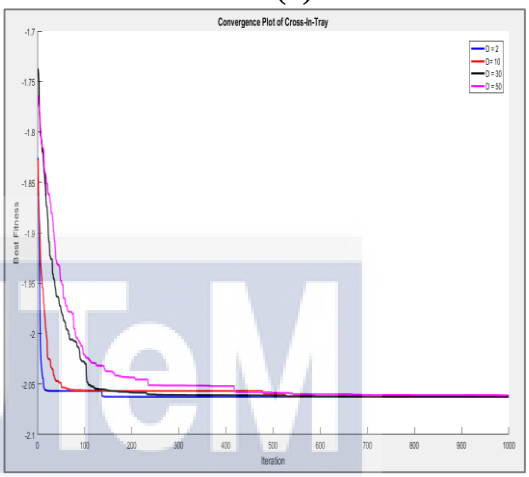

(a)



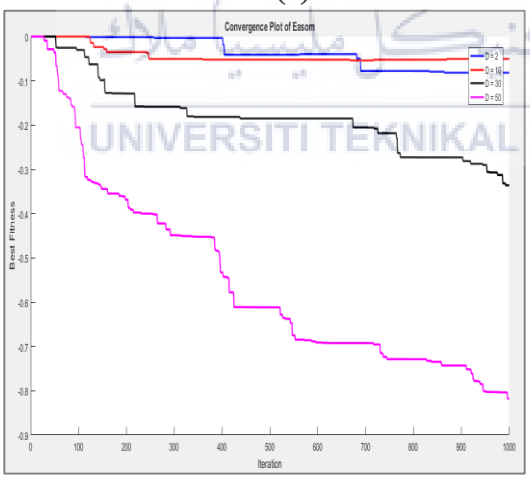

(b)



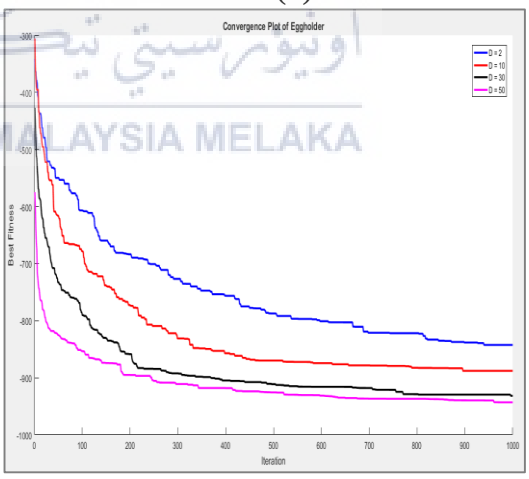

(c)



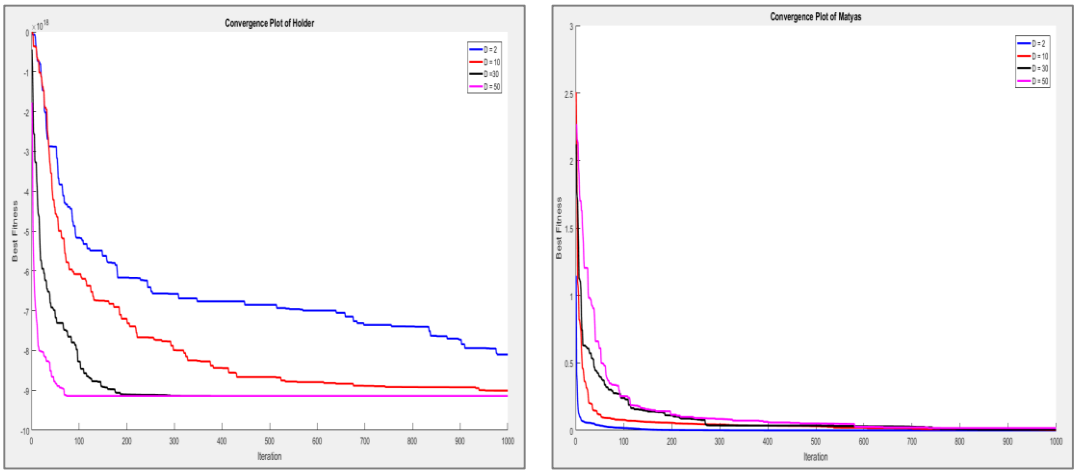

(d)



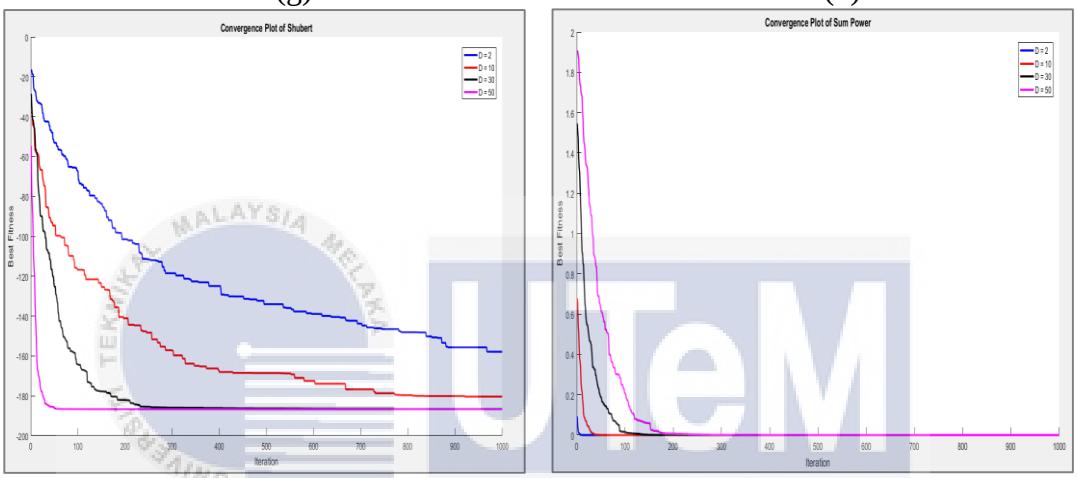

(e)




(f)




(g) (h)




(i) (j)

Figure 4.1: (a) Booth, (b) Bukin, (c) Six-Hump Camel, (d Cross-in-Tray, (e) Easom, (f) Eggholer, (g) Holder, (h) Matyas, (i) Shubert, (j) Sum of Different Powers



### 4.1.2 Second Analysis: Number of Population

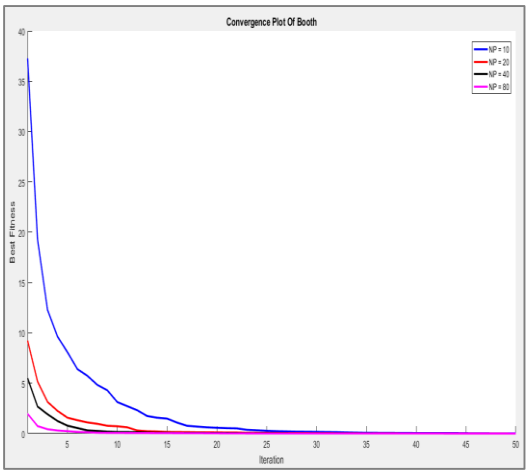
Different number of population is chosen as a second analysis to investigate the performance of ABC algorithm. The different population for this experiment are 20, 40, 60 and 80. For other parameter used for this analysis is number dimension is 2 and number of iteration is 1000. The limit control is 100. The experiments is repeated 30 independent runs in MATLAB. The mean values referred as best solution and standard deviation have been recorded obtain from the independent runs. The time showed in the table refer to time processing for each cycle in unit second. The result showed by incresing the number of bee, the longer the time will take to finish a cycle.

Table 4.2 showed the result of different population using 10 numerical benchmark function. The data of mean values, standard deviation, best value, worse value and time processing in second are recorded in table to show their performance. The analysis is showed that by increasing the population, the mean values are become smaller. The performance of convergence speed is fast and located at local minimum. Hence, the ABC algorithm with population of 80 is faster in convergence speed than a popluation of 10 for all bencmark functions. *F10* which is Sum of Different Powers function is the accurate and fast in term of convergence speed. The mean and standard deviation of *f1* are 9.596E-20 and 1.058E-19 respectively for population of 80. The *f7* which is holder function is the worst bencmark function as a test performance because it is away from local minimum and slow convergence speed. The mean and standard values are -9.139E+18 and 5134E00 respectively for population of 80. The figure 4.2 showed the convergence speed of different population using 10 numerical benchmark function.

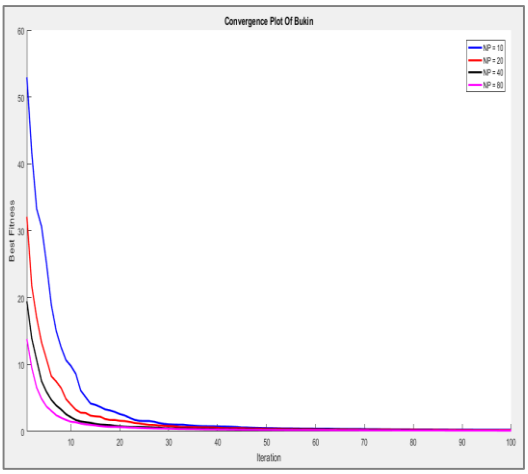
Table 4.2: The Result of Different Population using 10 Benchmark Function

| **Function** | **Accuracy** | **Number of Bees** | | | |
|---|---|---|---|---|---|
| | | **10** | **20** | **40** | **80** |
| *f1* | Mean | 1.543E-16 | 5.587E-17 | 2.633E-17 | 1.028E-17 |
| | Stan. Dev | 1.643E-16 | 5.756E-17 | 1.858E-17 | 7.815E-18 |
| | Best | 7.510E-18 | 4.290E-18 | 3.730E-18 | 4.850E-19 |
| | Worse | 4.840E-16 | 1.680E-16 | 7.420E-17 | 8.180E-18 |
| | Time (s) | 1.008E00 | 1.351E00 | 0.259E00 | 0.094E00 |
| *f2* | Mean | 0.080E00 | 0.070E00 | 0.070E00 | 0.070E00 |
| | Stan. Dev | 0.016E00 | 1.494E-07 | 0.000E00 | 0.000E00 |
| | Best | 0.104E00 | 0.070E00 | 0.070E00 | 0.070E00 |
| | Worse | 0.070E00 | 0.070E00 | 0.070E00 | 0.070E00 |
| | Time (s) | 0.893E00 | 1.303E00 | 2.252E00 | 3.453E00 |
| *f3* | Mean | -1.031E00 | -1.031E00 | -1.031E00 | -1.031E00 |
| | Stan. Dev | 0.000E00 | 0.000E00 | 0.000E00 | 0.000E00 |
| | Best | -1.031E00 | -1.031E00 | -1.031E00 | -1.031E00 |
| | Worse | -1.031E00 | -1.031E00 | -1.031E00 | -1.031E00 |
| | Time (s) | 0.944E00 | 1.624E00 | 2.407E00 | 3.921298 |
| *f4* | Mean | -2.062E00 | -2.062E00 | -2.062E00 | -2.062E00 |
| | Stan. Dev | 0.000E00 | 0.000E00 | 0.000E00 | 0.000E00 |
| | Best | -2.062E00 | -2.062E00 | -2.062E00 | -2.062E00 |
| | Worse | -2.062E00 | -2.062E00 | -2.062E00 | -2.062E00 |
| | Time (s) | 0.856E00 | 0.132E00 | 2.269E00 | 3.812E00 |

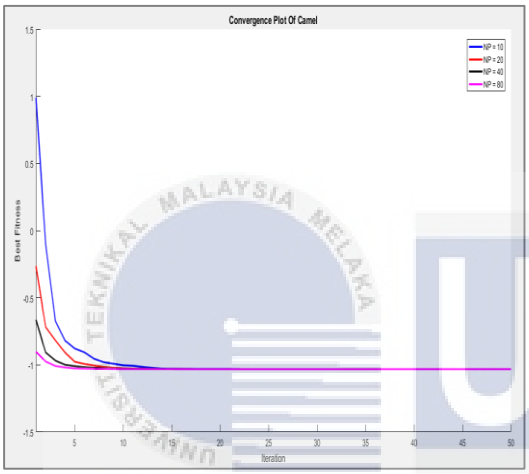
| | | | | | |
|---|---|---|---|---|---|
| *f5* | Mean | -0.818E00 | -0.989E00 | -0.999E00 | -0.999E00 |
| | Stan. Dev | 0.348E00 | 0.036E00 | 1.232E-04 | 3.018E-05 |
| | Best | -2.130E-15 | -0.805E00 | -0.999E00 | -0.999E00 |
| | Worse | -1.000E00 | -1.000E00 | -1.000E00 | -1.000E00 |
| | Time (s) | 0.849E00 | 1.792E00 | 2.789E00 | 4.002E00 |
| *f6* | Mean | -943.0E00 | -958.7E00 | -959.6E00 | -959.6E00 |
| | Stan. Dev | 25.26E00 | 3.218E00 | 0.125E00 | 1.454E-03 |
| | Best | -888.9E00 | -943.2E00 | -959.6E00 | -959.6E00 |
| | Worse | -959.6E00 | -959.6E00 | -959.6E00 | -959.6E00 |
| | Time (s) | 0.132E00 | 1.589E00 | 2.568E00 | 0.104E00 |
| *f7* | Mean | -9.139E+18 | -9.139E+18 | -9.139E+18 | -9.139E+18 |
| | Stan. Dev | 5742E00 | 6283E00 | 5134E00 | 5134E00 |
| | Best | -9.139E+18 | -9.139E+18 | -9.139E+18 | -9.139E+18 |
| | Worse | -9.139E+18 | -9.139E+18 | -9.139E+18 | -9.139E+18 |
| | Time (s) | 1.336E00 | 2.240E00 | 3.260E00 | 5.264E00 |
| *f8* | Mean | 3.232E-09 | 4.281E-12 | 2.450E-12 | 1.431E-12 |
| | Stan. Dev | 5.798E-09 | 9.448E-12 | 4.466E-12 | 2.676E-12 |
| | Best | 1.390E-15 | 6.560E-15 | 2.380E-14 | 7.700E-15 |
| | Worse | 1.440E-08 | 4.450E-11 | 1.500E-11 | 1.130E-11 |
| | Time (s) | 1.052E00 | 1.623E00 | 2.608E00 | 3.916E00 |
| *f9* | Mean | -186.7E00 | -186.7E00 | -186.7E00 | -186.7E00 |
| | Stan. Dev | 0.000E00 | 0.000E00 | 0.000E00 | 0.000E00 |
| | Best | -186.7E00 | -186.7E00 | -186.7E00 | -186.7E00 |
| | Worse | -186.7E00 | -186.7E00 | -186.7E00 | -186.7E00 |
| | Time (s) | 1.183E00 | 1.875E00 | 2.783E00 | 4.021E00 |
| *f10* | Mean | 1.864E-18 | 3.580E-19 | 3.9563E-19 | 9.596E-20 |
| | Stan. Dev | 2.093E-18 | 3.511E-19 | 5.466E-19 | 1.058E-19 |
| | Best | 2.120E-21 | 5.490E-22 | 1.23E-20 | 4.320E-20 |
| | Worse | 7.210E-18 | 5.380E-19 | 7.18E-19 | 1.170E-19 |
| | Time (s) | 0.997E00 | 1.554E00 | 2.487544 | 2.987E00 |



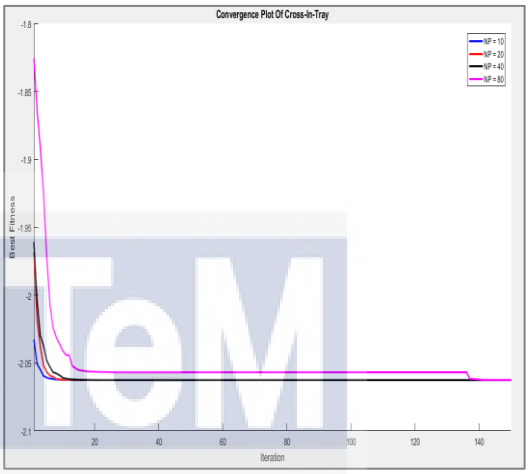

(a) Booth



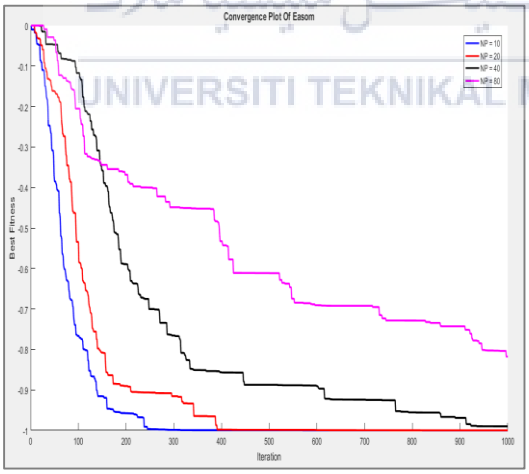

(b) Bukin6



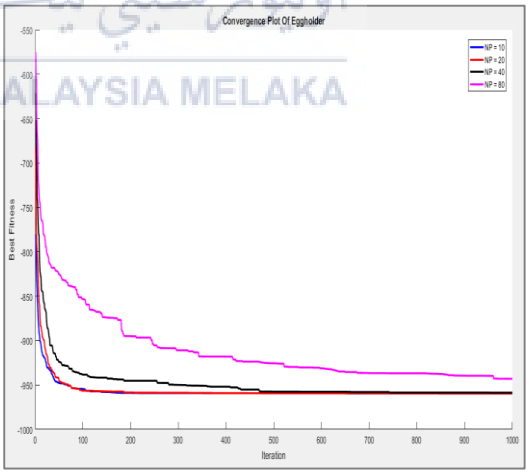

(c) Camel6



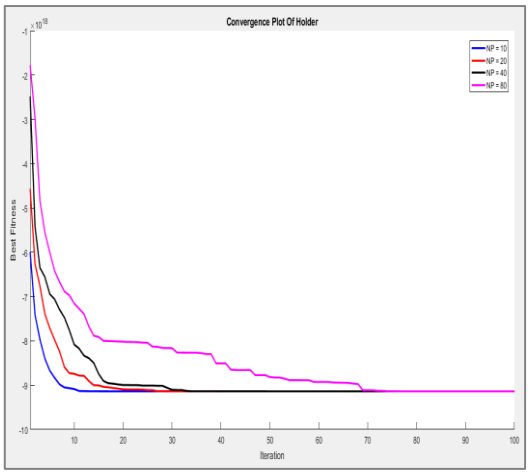

(d) Cross-it



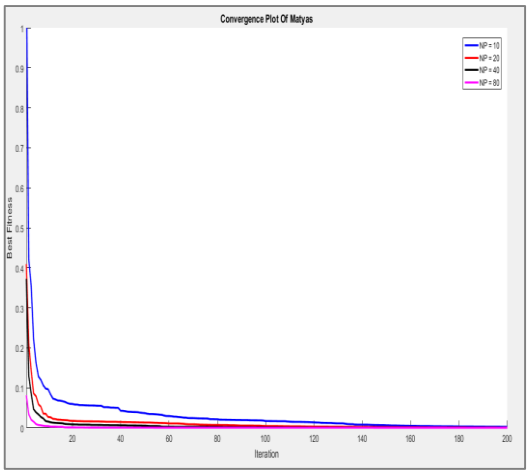

(e) Easom



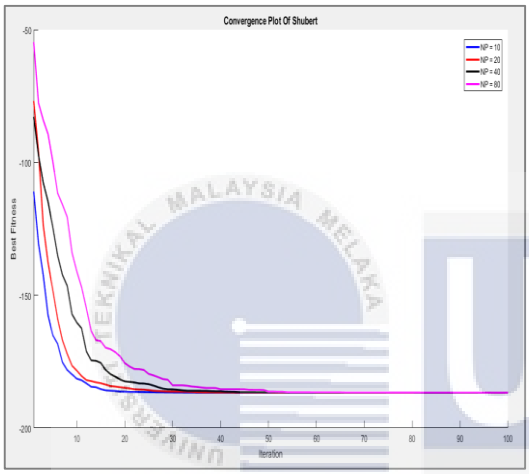

(f) Egg



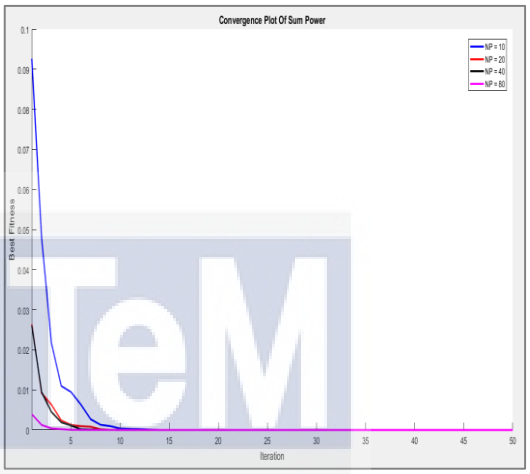

(g) Holder




(h) Matyas




(i) Shubert




(j) SumPow

Figure 4.2: (a) Booth, (b) Bukin, (c) Six-Hump Camel, (d Cross-in-Tray, (e) Easom, (f) Eggholer, (g) Holder, (h) Matyas, (i) Shubert, (j) Sum of Different Powers



### 4.1.3 Third Analysis: Number of Iteration

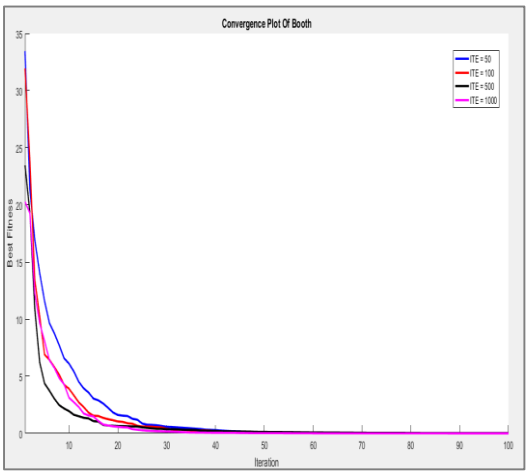
Different number of iteration is chosen as a third analysis to investigate the performance of ABC algorithm. The different iteration for this experiment are 50, 100, 500 and 1000. For other parameter used for this analysis is number dimension is 2 and number of population is 10. The limit control is 100. The experiments is repeated 30 independent runs in MATLAB. The mean values referred as best solution and standard deviation have been recorded obtain from the independent runs. The time showed in the table refer to time processing for each cycle in unit second. The result showed by incresing the number of bee, the longer the time will take to finish a cycle.

Table 4.2 showed the result of different population using 10 numerical benchmark function. The data of mean values, standard deviation, best value, worse value and time processing in unit second are recorded in table to show their performance. The analysis is showed that by increasing the ietraion, the mean values are become smaller. The performance of convergence speed is fast and located at local minimum. Hence, the ABC algorithm with iteration of 1000 is faster in convergence speed than a iteration of 50 for all bencmark functions. *F10* which is Sum of Different Powers function is the accurate and fast in term of convergence speed. The mean and standard deviation of *f10* are 1.864E-18 and 2.093E-18 respectively for iteration of 1000. The *f7* which is Holder function is the worst bencmark function as a test performance because it is away from local minimum and slow convergence speed. The mean and standard values are -9.139E18 and 5742.4E00 respectively for iteration of 1000. The figure 4.3 showed the convergence speed of different iteration using 10 numerical benchmark function.

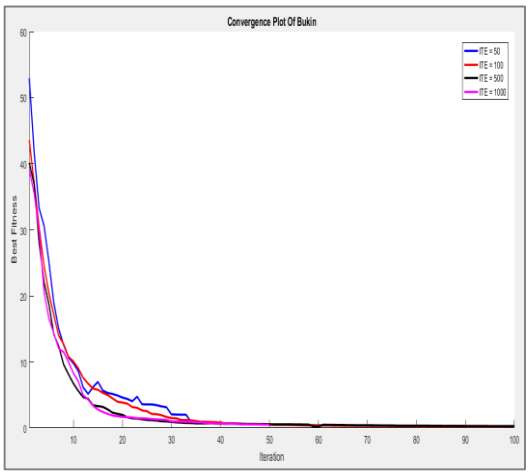
Table 4.3: The Result of Different Iteration using 10 Benchmark Function

| Function | Accuracy | Number of Iteration | | | |
|---|---|---|---|---|---|
| | | 50 | 100 | 500 | 1000 |
| *f1* | Mean | 7.947e-02 | 1.067e-03 | 1.071E-09 | 1.542E-16 |
| | Stan. Dev | 1.456e-01 | 2.178e-03 | 5.869E-09 | 1.643E-16 |
| | Best | 6.700E-05 | 3.080E-09 | 3.700E-18 | 1.100E-16 |
| | Worse | 3.232e-02 | 7.237e-03 | 3.210E-08 | 9.910E-17 |
| | Time (s) | 1.496e-01 | 1.879e-01 | 4.583e-01 | 1.789e-01 |
| *f2* | Mean | 4.856e-01 | 2.797E-01 | 9.107E-02 | 7.989E-02 |
| | Stan. Dev | 2.915e-01 | 1.938E-01 | 4.138E-02 | 1.667E-2 |
| | Best | 1.539E-01 | 8.858E-01 | 7.000E-02 | 1.267E-01 |
| | Worse | 1.317E00 | 7.887E-02 | 1.300E-01 | 1.1414E-01 |
| | Time (s) | 1.561E-01 | 3.908E-01 | 4.751E-01 | 8.934E-01 |
| *f3* | Mean | -1.031E00 | -1.031E00 | -1.031E00 | -1.031E00 |
| | Stan. Dev | 1.282E-05 | 1.119E-07 | 1.896E-08 | 0.00E00 |
| | Best | -1.031E00 | -1.031E00 | -1.031E00 | -1.031E00 |
| | Worse | -1.032E00 | -1.032E00 | -1.032E00 | -1.031E00 |
| | Time (s) | 1.604E-01 | 2.043E-01 | 6.539E-01 | 9.444E-01 |
| *f4* | Mean | -2.063E00 | -2.063E00 | -2.063E00 | -2.063E00 |
| | Stan. Dev | 1.658E-05 | 1.902E-11 | -2.063E00 | -2.063E00 |
| | Best | -2.062E00 | -2.062E00 | -2.063E00 | -2.063E00 |

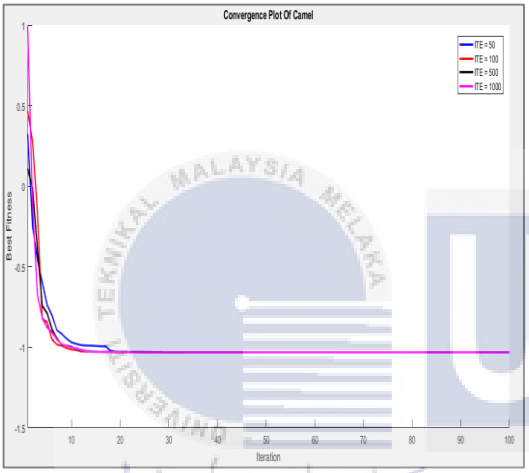
| | | | | | |
|---|---|---|---|---|---|
| | Worse | -2.063E00 | -2.063E00 | 0.000E00 | 0.000E00 |
| | Time (s) | 1.474E-01 | 1.841E-01 | 4.915E-01 | 8.560E-01 |
| *f5* | Mean | -8.537E-02 | -7.910E-02 | -6.116E-01 | -8.178E-01 |
| | Stan. Dev | 2.340E-01 | 2.131E-01 | 4.556E-01 | 3.485E-01 |
| | Best | 0.00E00 | 0.00E00 | -4.40E-208 | -2.130E-15 |
| | Worse | -9.865E-01 | -0.616E-01 | -1.000E00 | -1.000E00 |
| | Time (s) | 1.697E-01 | 2.219E-01 | 5.722-01 | 8.494E-01 |
| *f6* | Mean | -881.0E00 | -915.2E00 | -934.9E00 | -943.0E00 |
| | Stan. Dev | 58.27E00 | 64.37E00 | 29.01E00 | 25.27E00 |
| | Best | -959.5E00 | -959.6E00 | -935.3E00 | -959.6E00 |
| | Worse | -643.6E00 | -657.9E00 | -888.9E00 | -888.6E00 |
| | Time (s) | 1.706E-01 | 2.094E-01 | 5.843E-01 | 1.031E00 |
| *f7* | Mean | -8.718E18 | -9.139E18 | -9.139E18 | -9.139E18 |
| | Stan. Dev | 1.063E+18 | 1.063E14 | 4.4521E14 | 5742.4E00 |
| | Best | -5.700E+18 | -9.140E+18 | -9.140E+18 | -9.140E+18 |
| | Worse | -9.140E+18 | -9.140E+18 | -9.140E+18 | -9.140E+18 |
| | Time (s) | 1.831E-01 | 2.360E-01 | 6.361E-01 | 1.336E00 |
| *f8* | Mean | 4.049E-02 | 2.999E-02 | 4.156E-05 | 3.232E-09 |
| | Stan. Dev | 6.403E-02 | 9.504E-02 | 6.465E-05 | 5.798E-09 |
| | Best | 8.570E-05 | 1.088E-03 | 6.770E-08 | 1.390E-15 |
| | Worse | 1.932E-01 | 5.098E-01 | 1.029E-04 | 1.820E-09 |
| | Time (s) | 1.572E-01 | 1.953E-01 | 5.041E-01 | 1.052E00 |
| *f9* | Mean | -184.6E00 | -186.6E00 | -186.7E00 | -186.7E00 |
| | Stan. Dev | 6.075E00 | 5.199E-01 | 6.109E-06 | 0.000E00 |
| | Best | -153.7E00 | -184.6E00 | -186.7E00 | -186.7E00 |
| | Worse | -186.7E00 | -186.7E00 | -186.7E00 | -186.7E00 |
| | Time (s) | 1.703E-01 | 2.111E-01 | 5.285E-01 | 1.183E00 |
| *f10* | Mean | 3.598E-12 | 1.540E-17 | 4.662E-18 | 1.864E-18 |
| | Stan. Dev | 8.962E-12 | 1.358E-17 | 5.725E-18 | 2.093E-18 |
| | Best | 9.930E-18 | 1.200E-19 | 8.310E-21 | 2.121E-21 |
| | Worse | 2.600E-11 | 4.470E-17 | 1.270E-17 | 1.694E-18 |
| | Time (s) | 1.669E-01 | 2.060E-01 | 5.799E-01 | 9.975E-01 |



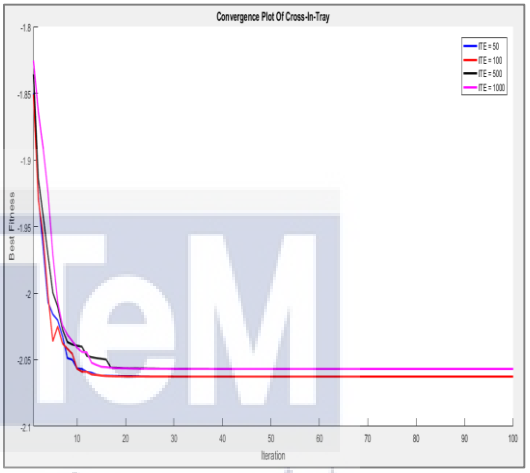

(a)



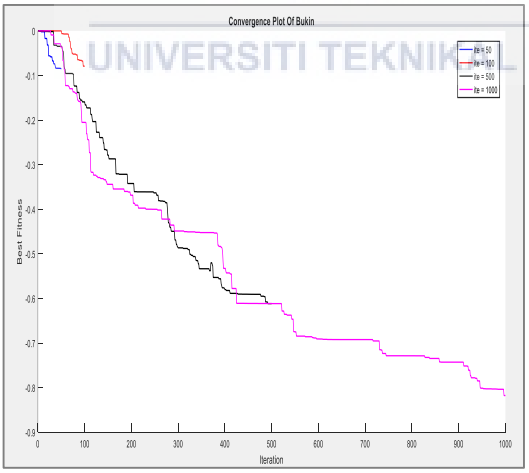

(b)



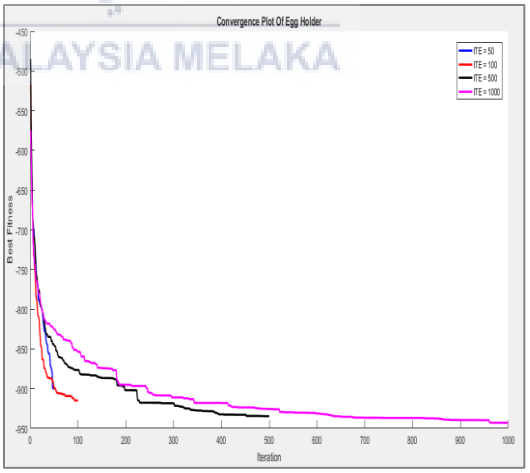

(c)



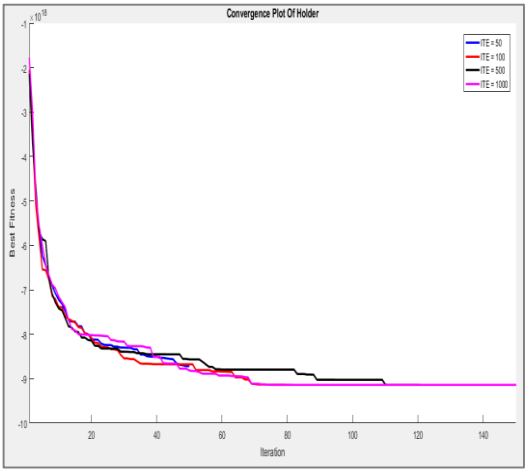

(d)



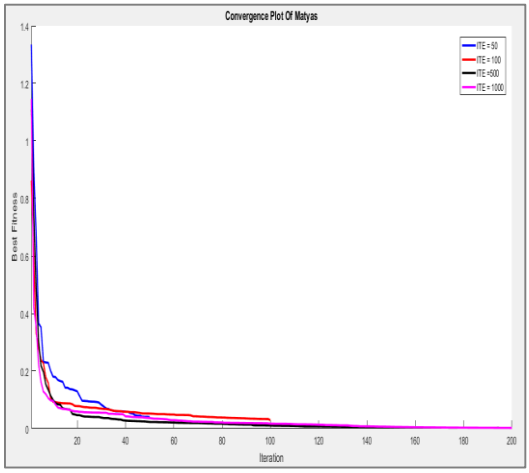

(e)



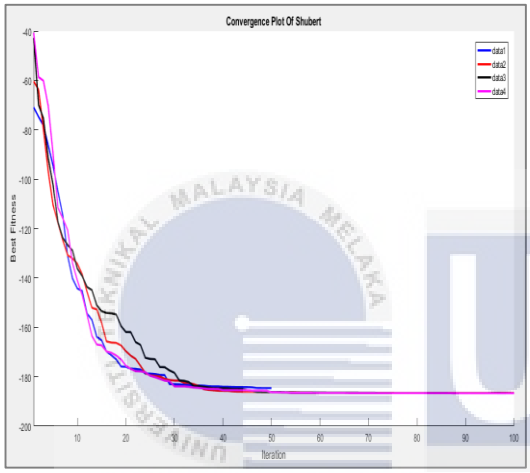

(f)



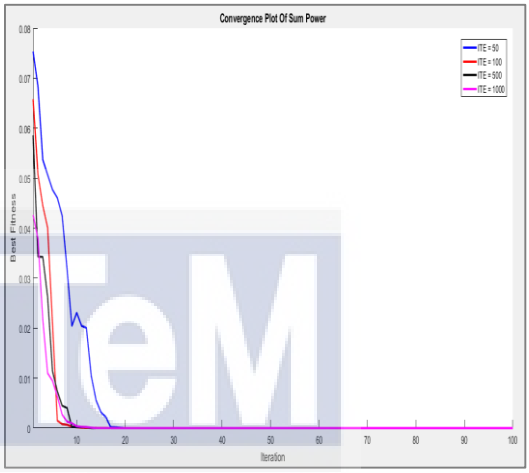

(g)




(h)




(i)




(j)

Figure 4.3: (a) Booth, (b) Bukin, (c) Six-Hump Camel, (d Cross-in-Tray, (e) Easom, (f) Eggholer, (g) Holder, (h) Matyas, (i) Shubert, (j) Sum of Different Powers



## 4.2 Performance of Controller with Error Criteria for FMS

The table 4.4 shows the control parameter values used to find the result of parameter setting and error criteria of PID, PD and PI controllers. The table 4.5 shows the result of parameter settings and error criteria of different controllers. The tuning method for ABC algorithm was using a 50 number of bees for 50 iterations. The search interval for tuning the value of kp, ki and kd is [0, 5]. Figure 4.4 showed the performance of PID, PD and PI controllers using different error criteria. The error criteria were used in this project are IAE, ISE, ITAE, MSE and RMSE. The result of this project is obtained by comparing the performance of controllers. The transient

involved are overshoot, peak time, rise time, settling time and steady state error. The performance index of error also given in the table below. The controller with minimum overshoot and settling time consider as best controller.

For PID controller, ISE had a minimum values of overshoot, peak time, settling time and steady state error which are 9.3410%, 0.7035 seconds, 12.098 seconds and 0.000 respectively. For PD controller, IAE had a minimum value of overshoot, rise time, settling time and steady state error which are 0.478%, 0.798 seconds, 8.829 seconds and 0.00 respectively. For PI controller, RMSE had a minimum overshoot which is 1.992%. PI controller had a minimum overshoot than PID controller but zero in steady state error. The result showed the fluctuated signal and disturbance lead to slow response and poor stability of system. PD also had a minimum overshoot than PID controller but faster in peak time.

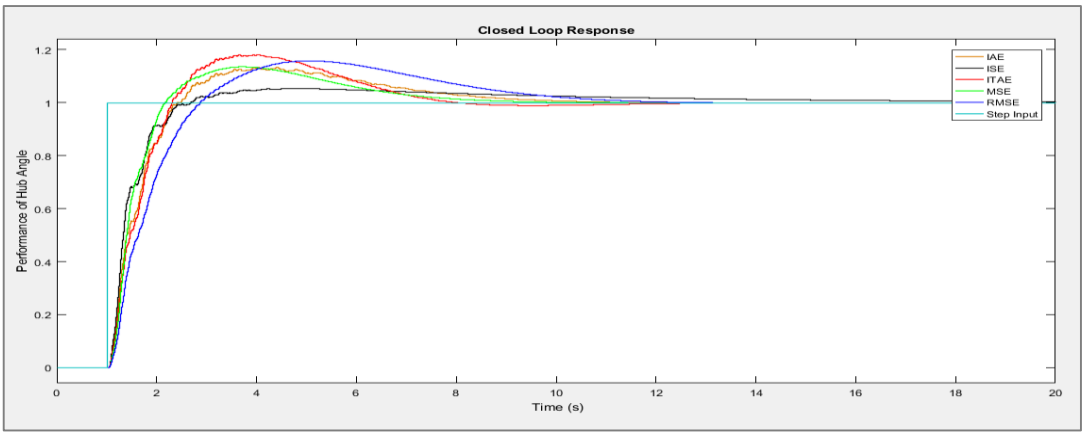
According to simulation results, PID controller is the best controller in tuning the value of kp, ki and kd because of their minimum overshoot, fast peak time and minimum settling time.

Table 4.4: Control Parameter Values

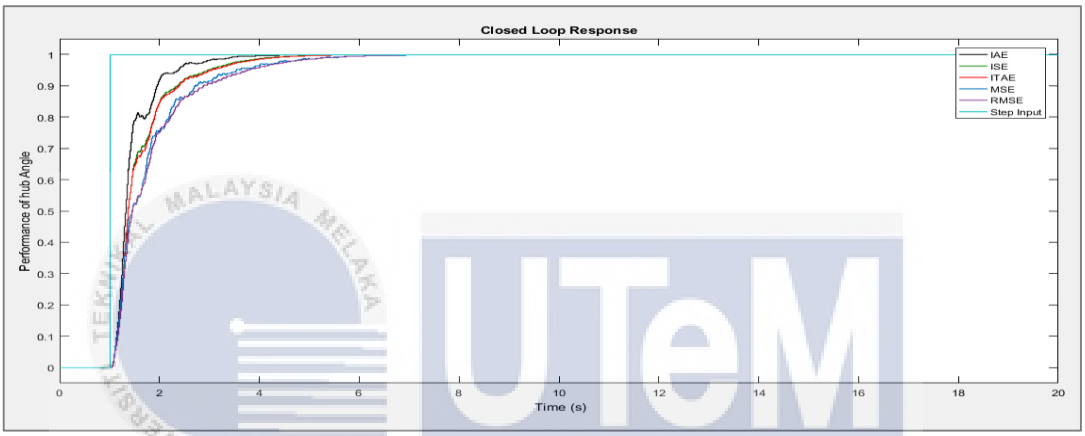
| Control Parameter | Values |
|---|---|
| Number of Bee | 50 |
| Number of Iteration | 50 |
| Search Interval | [0,5] |

Table 4.5: The Result of Parameter Settings and Error Criteria of Different Controllers

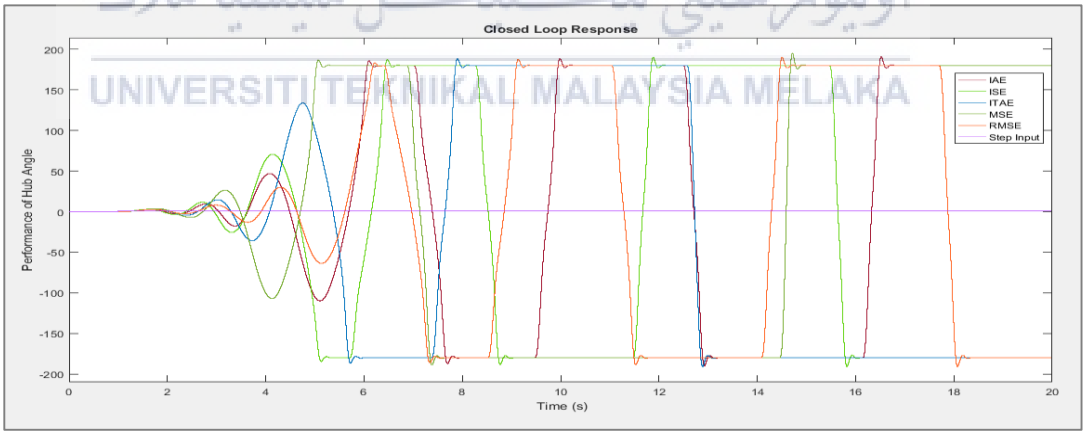
| Controller | Parameters | Error Criteria | | | | |
|---|---|---|---|---|---|---|
| | | IAE | ISE | ITAE | MSE | RMSE |
| PID | Kp | 3.4807 | 3.6710 | 2.9087 | 2.0067 | 1.5894 |
| | Ki | 1.2658 | 0.5413 | 1.6077 | 0.8451 | 0.5321 |
| | Kd | 2.0076 | 1.5388 | 1.9142 | 0.9724 | 1.2047 |
| | Overshoot (%) | 13.068 | 9.3410 | 18.452 | 14.368 | 15.698 |
| | Peak Time (s) | 0.9872 | 0.7035 | 0.9512 | 0.7498 | 1.2570 |
| | Rise Time (s) | 4.210 | 5.739 | 5.536 | 1.573 | 5.060 |
| | Settling Time (s) | 13.695 | 12.098 | 17.658 | 16.906 | 13.719 |
| | Steady State Error | 0.000 | 0.000 | -0.0690 | -0.0010 | 0.001 |
| | Performance Index | 1.0510 | 0.2874 | 3.6129 | 1.843e-12 | 1.408e-08 |
| PD | Kp | 3.041 | 2.4012 | 2.5363 | 3.2760 | 2.4639 |
| | Kd | 1.0344 | 1.0431 | 1.1236 | 1.8431 | 1.4104 |
| | Overshoot (%) | 0.478 | 0.496 | 0.497 | 0.499 | 0.501 |
| | Peak Time (s) | 15.079 | 13.362 | 16.599 | 11.544 | 19.906 |
| | Rise Time (s) | 0.798 | 1.151 | 1.210 | 1.542 | 1.691 |
| | Settling Time (s) | 8.829 | 10.925 | 10.704 | 14.790 | 11.165 |
| | Steady State Error | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | Performance Index | 0.453 | 0.3202 | 1.016 | 0.7687 | 4.441e-16 |
| PI | Kp | 3.4654 | 3.8699 | 2.5576 | 2.0736 | 2.6518 |
| | Ki | 0.9948 | 1.9052 | 2.7598 | 4.4035 | 0.5871 |
| | Overshoot (%) | 2.344 | 2.209 | 2.632 | 3.723 | 1.992 |
| | Peak Time (s) | 16.522 | 10.733 | 7.900 | 14.714 | 10.739 |
| | Rise Time (s) | 0.2883 | 0.4211 | 0.3633 | 0.1292 | 0.3589 |
| | Performance Index | 2494 | 4.522e05 | 3.329e04 | 3.329e04 | 3.276e04 |




(a)




(b)




(c)

Figure 4.4: (a) PID Controller, (b) PD Controller (c) PI Controller

## 4.3 Performance of ABC Algorithm in Tuning PID Controller with Error Criteria for FMS

The table 4.6 showed the result of PID parameter using different number of bees. The purpose of this error analysis was to investigate the performance of PID controller using different number of bees. The Figure 4.7 showed the performance of PID performance using different number of bees and error criteria. For ITAE error, the value of error for number of bees which are 10, 30 and 50 are 22.840, 21.342 and 18.452 respectively. The number of bee was 50 showed the minimum overshoot. The result showed that by increasing the number of bees, the overshoot of the PID controller was decreasing. By comparing error analysis from table below, ISE was the minimum overshoot compared to IAE, ITAE, MSE and RMSE. ISE is the lowest overshoot which is 10.566% when population is 50. From the result above, tuning PID controller with ISE is stable than other error criteria.

Table 4.6: Control Parameter Values

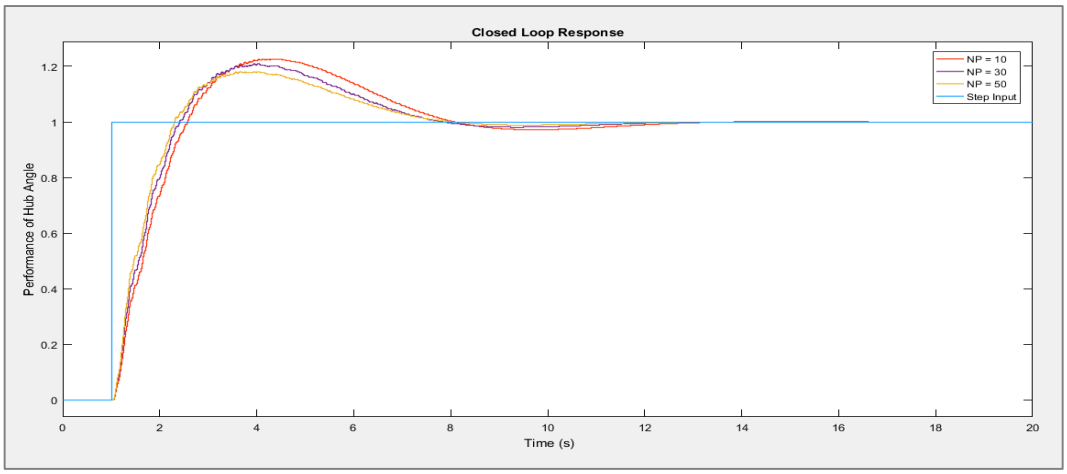
| Control Parameter | Values |
|---|---|
| Number of Dimension | 3 |
| Number of Iteration | 50 |
| Search Interval | [0,5] |



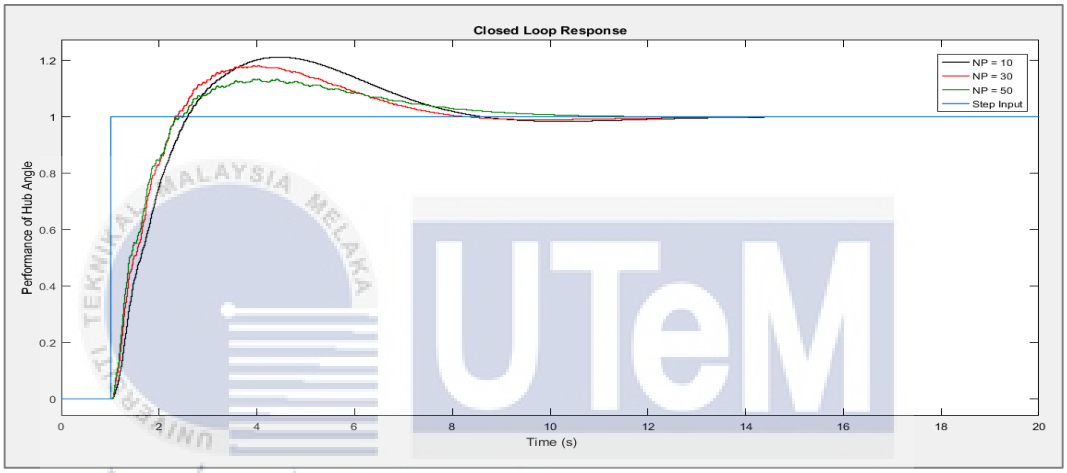
Table 4.7: The Result of PID Parameter Using Different Number of Bees

| Error Criteria | Parameters | Number of Bee(s) | | |
|---|---|---|---|---|
| | | 10 | 30 | 50 |
| ITEA | Kp | 2.1885 | 2.6998 | 2.9087 |
| | Ki | 1.3483 | 1.6077 | 1.6077 |
| | Kd | 1.9142 | 1.9142 | 1.9142 |
| | Overshoot (%) | 22.840 | 21.342 | 18.452 |
| | Rise Time (s) | 1.1080 | 1.0120 | 0.9512 |
| | Settling Time (s) | 18.344 | 17.627 | 17.658 |
| | Steady State Error | 0.000 | -0.001 | -0.069 |
| | Performance Index | 5.5050 | 4.3220 | 3.6120 |
| IAE | Kp | 1.4123 | 2.9076 | 3.4807 |
| | Ki | 0.7433 | 1.5486 | 1.2658 |
| | Kd | 1.1241 | 1.9613 | 2.0076 |
| | Overshoot (%) | 21.341 | 18.452 | 13.068 |

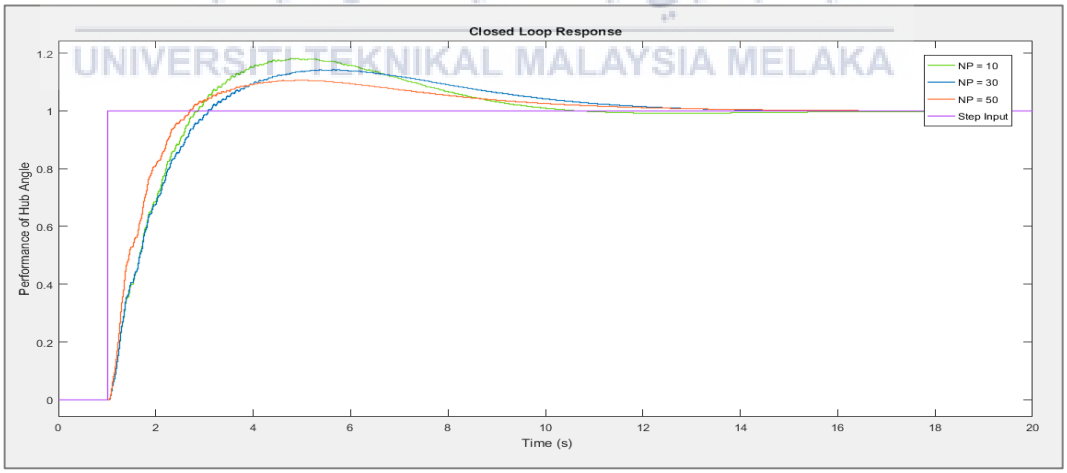
| | | | | |
|---|---|---|---|---|
| | Rise Time (s) | 1.0930 | 0.9682 | 0.9872 |
| | Settling Time (s) | 15.470 | 13.956 | 13.695 |
| | Steady State Error | -0.001 | -0.001 | 0.000 |
| | Performance Index | 1.4480 | 1.1740 | 1.0510 |
| ISE | Kp | 2.4245 | 2.3561 | 2.8473 |
| | Ki | 0.9685 | 0.6653 | 0.7493 |
| | Kd | 2.0971 | 1.9453 | 1.7054 |
| | Overshoot (%) | 18.452 | 14.368 | 10.556 |
| | Rise Time (s) | 1.3170 | 1.4180 | 1.0860 |
| | Settling Time (s) | 10.858 | 17.466 | 15.257 |
| | Steady State Error | 0.067 | 0.067 | 0.001 |
| | Performance Index | 0.5581 | 0.5396 | 0.3970 |
| MSE | Kp | 2.9168 | 1.7814 | 2.0067 |
| | Ki | 0.3174 | 0.6991 | 0.8451 |
| | Kd | 0.5115 | 1.1093 | 0.97242 |
| | Overshoot (%) | 29.221 | 15.698 | 14.368 |
| | Rise Time (s) | 0.2613 | 0.9675 | 0.7498 |
| | Settling Time (s) | 5.0080 | 9.7570 | 16.906 |
| | Steady State Error | -0.008 | 0.001 | -0.001 |
| | Performance Index | 9.419e-06 | 1.157e-10 | 1.843e-12 |
| RMSE | Kp | 1.2893 | 1.7983 | 1.5894 |
| | Ki | 0.8321 | 0.8557 | 0.5321 |
| | Kd | 1.2040 | 1.1265 | 1.2047 |
| | Overshoot (%) | 25.949 | 17.059 | 15.698 |
| | Rise Time (s) | 1.147 | 0.9138 | 1.2570 |
| | Settling Time (s) | 12.233 | 9.1070 | 13.719 |
| | Steady State Error | -0.007 | 0.004 | 0.001 |
| | Performance Index | 1.921e-07 | 9.159e-11 | 1.408e-08 |



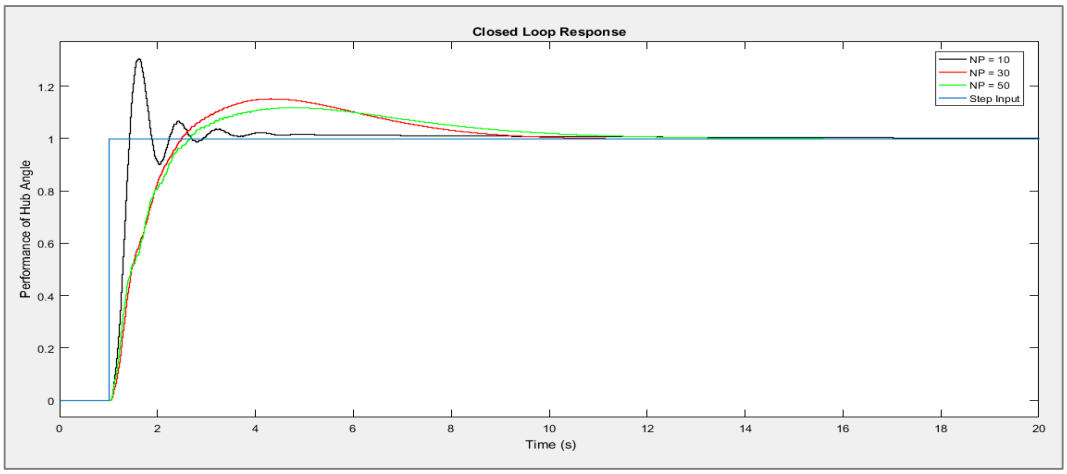

(a)



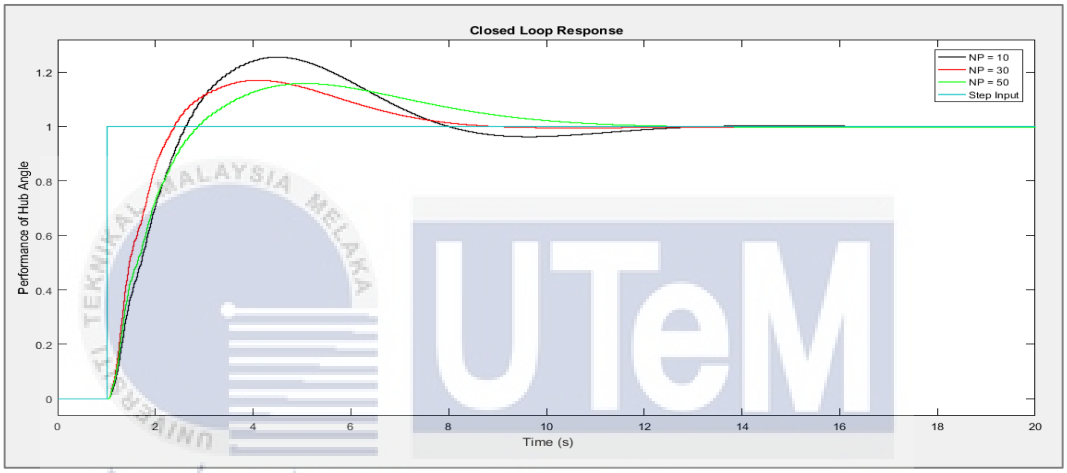

(b)




(c)




(d)




(e)

Figure 4.5: (a) ITAE, (b) IAE, (c) ISE, (d) MSE, (e) RMSE

# CHAPTER 5

# CONCLUSION AND RECOMMENDATIONS

## 5.1 Conclusion

A study of ABC algorithm has been presented in this project. The paper written D. Karaboga (2005) is the best reference as completing this project. The aim for this project have been achieved to investigate the performance of ABC algorithm using 10 numerical benchmark functions and choose the best controller with error criteria for application flexible manipulator system (FMS). The performance of ABC algorithm using benchmark function showed that function with global minimum is the best solution in term of fast convergence speed. As for application FMS, the PID controller is chosen as the best controller than PI and PD controller because of minimum overshoot, rise time and settling time. For PID controller, ISE has least of overshoot compared to other error criteria.

The ABC algorithm is successful implemented for application and it can be used to solve many real engineering problems. Overall, the performance of ABC algorithm is excellent in term fast convergence speed.



## 5.2 Recommendations

Although the objective has been successfully archive of this project, there are a few recommendation for improvement ABC algorithm to solve real engineering problems. The recommendation are stated as below:

1. Hybrid the original ABC algorithm which consists of many control parameter to compare the performance between the hybrid and original of ABC algorithm.
2. Testing the application with high order plants with high dimensionality.

## REFERENCES


[1] D. Karaboga and B.Akay, ‘‘*Artificial Bee Colony (ABC), Harmony Search and Bees Algorithms on Numerical Optimization*’’,Erciyes University, The Dept. of Computer Engineering, 38039, Melikgazi, Kayseri, Turkiye, 2009.

[2] E. Bonabeau, M. Dorigo, G. Theraulaz, “*Swarm Intelligence: From Natural to Artificial Systems*”, New York, NY: Oxford University Press, 1999.

[3] D Karaboga, ‘*An Idea Based on honey bee swarm for numerical optimization*,’ Techn, 2005.

[4] Adil Baykaso, Lale Ozbakir and Pinar Tapkan, “*Artificial bee colony algorithm and its application to generalized assignment problem*,” University of Gaziantep, Department of Industrial Engineering 2Erciyes University, Department of Industrial EngineeringTurkey.

[5] Bagis, A, ‘*Determination of the PID controller parameters by modified genetic algorithm for improved performance*’. Journal of Information Science and Engineering, 2007.

[6] KARABOGA, D., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Journal of global optimization, 39:459-471.

[7] Yujiang Yi and Renjie He, “*A novel artificial bee colony algorithm*,” College of Information System and Management National University of Defense Technology, Changsha, China.

[8] D.Karaboga, “A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC)’’, Springer Science+Business Media, 2007.

[9] N.Ahmad, ‘‘*Bacterial Foraging and Spiral Dynamics Based Metaheuristic Algorithms for Global Optimisation with Engineering Applications*’’, Department of Automatic Control and Systems Engineering The University of Sheffield Mappin Street Sheffield, S1 3JD United Kingdom, 2014.

[10] D.Karaboga and B.Basturk, “*On the performance of artificial bee colony (ABC) algorithm*,” Erciyes University, Engineering Faculty, Computer Engineering Department, TR-38039 Kayseri, Turkey, 2007.

[11] D.Karaboga and B.Basturk, “*Algorithm for solving constrained optimization problems*,” Erciyes University, Engineering Faculty, The Department of Computer Engineering, 2007.

[12] Dr.Dharmender Kumarand Balwant Kumar, “*Optimization of benchmark functions using artificial bee colony (ABC) algorithm*,” 1(Assistant professor, CSE dept., GJUS&T), 2(CSE dept., GJUS&T).

# APPENDICES

## APPENDIX A CODING OF ABC ALGORITHM

```
tic
clear all
close all
clc

%/* Control Paramweters of ABC algorithm*/
NP=50; %/* The number of colony size (employed bees+onlooker bees)*/
FoodNumber=NP/2; %/*The number of food sources equals the half of the
colony size*/
limit=100; %/*A food source which could not be i.mproved through "limit"
trials is abandoned by its employed bee*/
maxCycle=50; %/*The number of cycles for foraging {a stopping criteria}*/


%/* Problem specific variables*/
objfun='tracklsq'; %cost function to be optimized
D=3; %/*The number of parameters of the problem to be optimized*/
ub=ones(1,D)*5; %/*lower bounds of the parameters. */
lb=ones(1,D)*0;%/*upper bound of the parameters.*/

runtime=1;%/*Algorithm can be run many times in order to see its
robustness*/

ObjVal(:,:)=0*ones(1,FoodNumber);
GlobalMins=zeros(1,runtime);

for r=1:runtime

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

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


for i=1:FoodNumber

ObjVal(1,i)=feval(objfun,Foods(i,:));

end

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

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

%/*The best food source is memorized*/
```

```
BestInd=find(ObjVal==min(ObjVal));
BestInd=BestInd(end);
GlobalMin=ObjVal(BestInd);
GlobalParams=Foods(BestInd,:);

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

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

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

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

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

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

       %  /*if generated parameter value is out of boundaries, it is
shifted onto the boundaries*/
        ind=find(sol<lb);
        sol(ind)=lb(ind);
        ind=find(sol>ub);
        sol(ind)=ub(ind);

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

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


    end;
```

```
%%%%%%%%%%%%%%%%%%%%%%%% CalculateProbabilities
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%/* A food source is chosen with the probability which is proportioal to
its quality*/
%/*Different schemes can be used to calculate the probability values*/
%/*For example prob(i)=fitness(i)/sum(fitness)*/
%/*or in a way used in the method below
prob(i)=a*fitness(i)/max(fitness)+b*/
%/*probability values are calculated by using fitness values and
normalized by dividing maximum fitness value*/

prob=(0.9.*Fitness./max(Fitness))+0.1;

%%%%%%%%%%%%%%%%%%%%%%%% ONLOOKER BEE PHASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

i=1;
t=0;
while(t<FoodNumber)
    if(rand<prob(i))
        t=t+1;
        %/*The parameter to be changed is determined randomly*/
        Param2Change=fix(rand*D)+1;

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

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

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

       %  /*if generated parameter value is out of boundaries, it is
shifted onto the boundaries*/
        ind=find(sol<lb);
        sol(ind)=lb(ind);
        ind=find(sol>ub);
        sol(ind)=ub(ind);

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

       % /*a greedy selection is applied between the current solution i
and its mutant*/
       if (FitnessSol>Fitness(i)) %/*If the mutant solution is better
than the current solution i, replace the solution with the mutant and
reset the trial counter of solution i*/
            Foods(i,:)=sol;
            Fitness(i)=FitnessSol;
            ObjVal(i)=ObjValSol;
            trial(i)=0;
```

```
        else
            trial(i)=trial(i)+1; %/*if the solution i can not be
improved, increase its trial counter*/
        end;
    end;

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


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


%%%%%%%%%%% SCOUT BEE PHASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%/*determine the food sources whose trial counter exceeds the "limit"
value.
%In Basic ABC, only one scout is allowed to occur in each cycle*/

ind=find(trial==max(trial));
ind=ind(end);
if (trial(ind)>limit)
    trial(ind)=0;
    sol=(ub-lb).*rand(1,D)+lb;
    ObjValSol=feval(objfun,sol);
    FitnessSol=calculateFitness(ObjValSol);
    Foods(ind,:)=sol;
    Fitness(ind)=FitnessSol;
    ObjVal(ind)=ObjValSol;
end;


costvalue(:,iter)=GlobalMin;
fprintf('Ýter=%d ObjVal=%g\n',iter,GlobalMin);
iter=iter+1;

end % End of ABC


GlobalMins(r)=GlobalMin;
end; %end of runs

toc
save all
```