DEVELOPMENT OF A MODIFIED FRUIT FLY OPTIMIZATION ALGORITHM BASED LINEAR QUADRATIC REGULATOR CONTROLLER FOR AIRCRAFT PITCH CONTROL SYSTEM

BY

SAFIYA ALIYU

DEPARTMENT OF COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

FEBRUARY, 2018

DEVELOPMENT OF A MODIFIED FRUIT FLY OPTIMIZATION ALGORITHM BASED LINEAR QUADRATIC REGULATOR CONTROLLER FOR AIRCRAFT PITCH CONTROL SYSTEM

BY

Safiya ALIYU, B.Eng (BUK) 2008

P14EGEE8020

safiyyaham@yahoo.com

A DISSERTATION SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE (MSc) DEGREE IN CONTROL ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING
FACULTY OF ENGINEERING
AHMADU BELLO UNIVERSITY, ZARIA
NIGERIA

February, 2018

DECLARATION

I declare that this dissertation titled "**Development of a Modified Fruit Fly Optimization Algorithm Based Linear Quadratic Regulator for the Aircraft Pitch Control System (PCS)**" was carried out by me in the Department of Computer Engineering, Ahmadu Bello University, Zaria as part of the requirements for the award of degree of Master of Science in Control Engineering. All literatures used and cited are duly acknowledged in the reference pages. The information derived from literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other institution.

Safiya ALIYU _____ _____

(Student) Signature Date

DEDICATION

This research work is dedicated to Almighty Allah (S.W.T), His beloved Prophet Muhammad (S.A.W) and then to my beloved parents.

CERTIFICATION

This Dissertation titled "DEVELOPMENT OF A MODIFIED FRUIT FLY OPTIMIZATION ALGORITHM BASED LINEAR QUADRATIC REGULATOR CONTROLLER FOR AIRCRAFT PITCH CONTROL SYSTEM" by Safiya ALIYU meets the regulations governing the award of degree of Master of Science (M.Sc) in Control Engineering of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

_____	_____	_____
Chairman, Supervisory Committee (Prof. M.B.Mu'azu)	(Signature)	Date
_____	_____	_____
Member, Supervisory Committee (Dr. A.A.Mati)	(Signature)	Date
_____	_____	_____
Head of Department (Prof. M. B.Mu'azu)	(Signature)	Date
_____	_____	_____
Dean. School of Postgraduate Studies (Prof. S. Z. Abubakar)	(Signature)	Date

ACKNOWLEDGMENT

In the name of Allah (SWT), the most beneficent, the most merciful. All praises are due to Allah (SWT). Peace and blessings be upon His Prophet, Muhammad (SAW), his companions and those that follow their right path until the day of resurrection.

My limitless appreciation goes to the chairman of my supervisory committee, Prof. M.B Mu'azu, for his tireless effort, support, patience, constant guidance and encouragement from the beginning to the end of this dissertation. My appreciation also goes to my co-supervisor Dr. A.A. Mati for his guidance and contributions. May Allah reward them with Jannatul Firdaus. To all lecturers of Electrical and Computer Engineering namely, Prof. B. G. Bajoga, Prof B. Jimoh, Dr. E.A. Adedokun, Dr. Y. Jibril, Dr. S.M. Sani, Dr. K.A. Abubilal, Dr. A.D. Usman, Dr. A.M.S. Tekanyi, Dr. I.J. Umoh, Dr. H.A. Abubakar, Engr. A.S. Musa, Engr A.I. Abdu, Engr. M.J. Muazu, Engr. J.O. Haruna, Engr A. Mohammed, Engr. S.T. Ajayi, and those too numerous to mention, I thank you for the valuable contributions and encouragement.

Special thanks to my parents Engr Aliyu Mohammed Ibrahim and Hajiya Maryam Aliyu for their support and prayers. Words alone cannot express my gratitude.

My profound appreciation goes to my hubby Alhaji Halliru Abdullahi for his understanding, care, kindness, patience and words of encouragement. I say thanks to my lovely children Ja'afar and Aliyu Halliru Ja'afar for their patience and understanding.

Gratefulness to my siblings; Nafisa, Salisu, Alhassan, Muhibbah, Harisu, Ahmad, Umar and Sani, my niece; Suhaila, my nephews; Haidhar and Khalil, my uncle and aunt Baba Bashir & family, my cousins; Aisha and Fatima Usman, my grandmothers; Hajiya Fatima Umar and Amina Abubakar, my in-laws especially Mama Hafsatu, Aisha I. Zubair, Hadiza and Fatima Ja'afar, Lawal S. Halliru and Zainab Usman, for all the care and prayers.

I am grateful to Tijjani A. Salawudeen for his guidance and immense contribution to the completion of this research work. I am grateful to my friends and my colleagues; Jamilu Yusuf, Zainab Mukhtar, Ibrahim Mansur, Ahmad Rufa'i, Basira, Minister, Mal. Tukur Abdullahi, the entire members of Computer/Control research group and people of CERT.

Safiya ALIYU

February 2018

ABSTRACT

This research is aimed at the development of a modified fruit fly optimization algorithm (mFFOA) for the determination of optimized weighting matrices (Q which is a positive definite matrix that penalizes the states and R which is also a positive definite matrix that penalizes the control inputs) of the linear quadratic regulator (LQR) to be used for the aircraft pitch control system (PCS).The standard fruit fly optimization algorithm (FFOA) is an optimization algorithm inspired by intelligent smell and vision behaviour of flies towards fruit. The FFOA suffers from the problem of lack of balance between exploration and exploitation as it has a higher rate of exploitation than exploration leading to a high probability of it being trapped in some local optimal. The mFFOA was developed by modifying the iteration factor of the search radius to a decreasing function to improve the exploration capability of the algorithm and then by introducing a linearly decreasing inertial weight in order to provide an efficient balance between exploration and exploitation of the FFOA. The mFFOA was benchmarked against the FFOA using ten optimization test functions (Ackley, Alpine, Eggcrate, Griewank, Pathologic, Rastrigrin, Rosenbrock, Schaffer, Sphere, and Whitley) and showed a 20% improvement in its convergence to global optima. The optimized values of the Q and R weighting matrices are obtained for ten test runs using mFFOA within a time of 126.9538s when compared with the 140.7819s taken using the FFOA approach. The proposed method reduced the time taken by the FFOA by 13.8281s.These matrices used to determine the LQR controller for the PCS showed a settling time of4.4456s when compared to 4.4764**s** obtained using FFOA. This showed a convergence of the solution search space using the LQR (mFFOA) having a more optimal time-to-solution.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

LIST OF ABREVIATIONS

ABC	Artificial Bee Colony
AFSA	Artificial Fish Swarm Algorithm
ANN	Artificial Neural Network
ARM	Adaptive Radius Mechanism
CI	Computational Intelligence
CMAES	Covariance Matrix Adaptation Evolution Strategy
CPU	Central Processing Unit
DFOA	Fruit Fly Optimization Algorithm based on Differential Evolution
EM	Electromagnetic
FA	Firefly Algorithm
FFO /FOA /FFOA	Fruit Fly Optimization Algorithm
FOA-FA	Hybridized Fruit Fly Optimization Algorithm and Firefly Algorithm
FOAGRNN	FOA-optimized General Regression Neural Network
GA	Genetic Algorithm
G-FOA	Improved Fruit Fly Optimization Algorithm using Bivariable Function.
GRNN	Generalized Regression Neural Network
HP	Hewlett-Packard
IFFO	Improved Fruit Fly Optimization Algorithm with a new control parameter

JDK	Java SE Development Kit
LGMS-FOA	Improved Fruit Fly Optimization Algorithm based on Linear Generation Mechanism of candidate Solution
LQR	Linear Quadratic Regulator
MATLAB	Matrix Laboratory
mFFOA	modified Fruit Fly Optimization Algorithm
MFOA	Modified Fruit Fly Optimization Algorithm using random search m and adaptive population size
MIV	Mean Impact Value
NLPPs	Nonlinear Programming Problems
OLS_LR	Least Squares Linear Regression
PCS	Pitch Control System
PID	Proportional Integral Derivative
PSO	Particle Swarm Optimization
PSOGRNN	Generalized Regression Neural Network model with Particle Swarm Optimization
PSOGRNN	PSO-optimized General Regression Neural Network
RAM	Read Access Memory
RBF	Radial Basis Neural Network
RMS	Root Mean Square
RMSE	Root-Mean-Square Error
SaDE	Self-adaptive Differential Evolution Algorithm

SALSSVM	Least Squares Support Vector Machine with Simulated Annealing Algorithm
SEDI-FOA	Improved Fruit Fly Optimization Algorithm based on Selecting Evolutionary Direction Intelligently
TSP	Traveling Salesman Problem
TSPLIB	International standard library for Traveling Salesman Problems
wAFSA	weighted Artificial Fish Swarm Algorithm

CHAPTER ONE

INTRODUCTION

1.1 Background of Research

The linear quadratic regulator (LQR) is a linear optimal controller which ensures that a feedback control system returns to its state of equilibrium in the presence of disturbances or perturbations. The weighting matrices (Q which is a positive definite matrix that penalizes the states and R which is also a positive definite matrix that penalizes the control inputs) are the critical design parameters for the LQR and are usually selected by the designer (Hamidi, 2012) which may be time consuming and not necessarily lead to optimal solution. Recently, researches have been carried out for the determination of the optimized weighting matrices of the LQR controller using different computational intelligence (CI) methods which are useful in addressing these issues, such as: genetic algorithm (GA) (Nagarkar & Patil, 2016 ;Gupta & Tripathi, 2014; Abdulla *et al.*, 2013; Ghoreishi *et al.*, 2011); particle swarm optimization (PSO) (Ghoreishi & Nekoui, 2012; artificial bee colony (ABC) (Ata & Coban, 2015); weighted artificial fish swarm algorithm (wAFSA) (Mu'azu *et al.*, 2015; Salawudeen & Mu'azu, 2015), etc. For the purpose of this research work mFFOA is used to determine the weighing matrices of the LQR.

Pan (2011) developed the fruit fly optimization algorithm (FFOA), which is a metaheuristic search method that works based on the food finding behaviour of the fruit fly (*Drosophila*). The main inspiration of FFOA is that the fruit fly is superior to other species in sensing and perception, especially in osphresis and vision it can smell food source 40km away and it has the following advantages: simple structure, immediately accessible for practical applications, ease of implementation and speed to acquire solutions (Pan *et al.*,

2014). Studies have shown that it can be used to solve real world and engineering optimization problems (Pan *et al.*, 2014).

However, the FFOA suffers from the problem of lack of balance between exploration and exploitation in that it has a higher rate of exploitation than exploration leading to a higher probability of it being trapped into some local optimal (Rizk-Allah, 2016). It also has the drawback of having a fixed value of search radius (Pan *et al.*, 2014). In order to address these issues, an iteration factor was introduced so that the search radius can be adaptively changed for different evolution phases (Pan *et al.*, 2014). However, the iteration factor introduced by Pan *et al.*, (2014) resulted in a larger step size which limits the exploration capability of the algorithm. This work intends to modify the adaptive behaviour of the FFOA by modifying the iteration factor of search radius introduced by (Pan *et al.*, 2014), to a decreasing function in order to improve the exploration capability and also by introducing a linearly decreasing inertial weight in order to provide an efficient balance between exploration and exploitation of the FFOA.This research aims to apply the modified FFOA (due to its simplicity and its ability to improve the convergence time) to determine the optimized parameters of LQR (Q and R) for the pitch control system (PCS) of an aircraft in order to enhance its time-to-solution.

The PCS helps a pilot to control an aircraft by keeping the pitch attitude constant, that is, make the aircraft return to desired attitude in a reasonable length of time after a disturbance of the pitch angle, or make the pitch follow a given command as fast as possible (Ju & Mohamed, 2007). Many research works have been reported on controlling the pitch or longitudinal dynamic of an aircraft for the purpose of flight stability using LQR (Jisha &

Aswin, 2015; Stefanescu *et al.*, 2013; Wahid & Hassan, 2012; Wahid & Rahmat, 2010). The approach developed in this work improved the settling time of the PCS.

1.2 **Problem Statement**

The FFOA lacks balance between exploration and exploitation as such it has higher rate of exploitation than exploration leading to a higher probability of it being trapped into some local optimal. It also has the drawback of having a fixed value of search radius. In order to address this issue, an iteration factor was introduced (Pan *et al.*, 2014) so that the search radius can be adaptively changed for different evolution phases. However, the iteration factor introduced results in a large step size which limits the exploration capability of the algorithm. Several CI methods have been used to determine the weighting matrices of the LQR controller for complex engineering benchmark system PCS in literature, but these methods require high computational time and are complex.

Thus this research developed an algorithm to determine the weighting matrices of the LQR controller for PCS using decreasing function iteration factor for the search radius to improve the exploration capability of the FFOA and linearly decreasing inertial weight to provide an efficient balance between exploration and exploitation for the FFOA. This improved the time-to-solution in the determination of the mFFOA-based LQR weighting matrices. Also, the effectiveness of the mFFOA-based LQR on the complex PCS system was investigated.

1.3 Aim and Objectives

The aim of this study is to develop a modified fruit fly optimization algorithm (mFFOA) for the determination of the optimized weighting matrices (Q and R) of the LQR controller using the aircraft pitch control system (PCS) as a case study.

In order to achieve this aim, the following objectives were set:

1. Replication of the standard FFOA and development of the modified FFOA by the introduction of a linearly decreasing inertial weight and a decreasing iteration function.
2. Comparison of the performances of the standard FFOA and the modified FFOA using ten (10) benchmark optimization test functions (Ackley, Alpine, Eggcrate, Griewank, Pathologic, Rastrigrin, Rosenbrock, Schaffer, Sphere, and Whitley) using the convergence to the global optima as the performance metric.
3. Application of the mFFOA for the determination of the optimized weighting matrices (Q and R) of LQR controller for aircraft PCS model adopted from Jisha & Aswin (2015) and evaluating its performance using time-to-solution and settling time.

1.4 Methodology

The methodology adopted for this research work is as follows:

1. Replicate the standard FFOA, using the following steps:
    a. Initialize parameters such as: maximum generation; maximum population; location; step and random reference
    b. Initialize the fruit fly group i.e. random direction and distance of searching for food using the sense of smell of an individual fruit fly.

c. Generate several fruit flies randomly around the fruit using osphresis for foraging.

d. Evaluate the population to obtain the smell concentration values (fitness value).

e. Determine the fruit fly with the maximum smell concentration among the fruit fly swarm.

f. Retain the best smell concentration value and x, y coordinates, the fruit fly swarm flies toward the position by vision.

2. Develop the modified FFOA:

   a. Repeat step 1 (a) with random reference =

$$\lambda = \lambda_{\max} \times \exp\left(\log\left(\frac{\lambda_{\min}}{\lambda_{\max}} \right) \times \left(\frac{Iter_{\max} - Iter}{Iter_{\max}} \right) \right)$$

   where, λ is the search radius in each iteration, $\lambda_{\max}$ is the maximum radius, $\lambda_{\min}$ is the minimum radius, $Iter$ is the iteration number and $Iter_{\max}$ is the maximum iteration number.

   b. Repeat step 1(b)

   c. Introduction of linearly decreasing inertial weight

$$w_k = w_{\max} - \frac{w_{\max} - w_{\min}}{iter_{\max}} \times k \text{ to step 1(c)}$$

   where, w_k is the current inertial weigh, $w_{\max}$ is the maximum generated inertial weight, $w_{\min}$ is the minimum generated inertial weight and k is the kth position of inertial weight

   d. Repeat Step 1(d-f)

3. Compare the performance of the proposed mFFOA with that of the standard FFOA using ten benchmark optimization test functions (Ackley, Alpine, Eggcrate, Griewank, Pathologic, Rastrigrin, Rosenbrock, Schaffer, Sphere, and Whitley)
4. Determine the optimized LQR weighting matrices (Q and R) using the modified FFOA
5. Implement the LQR controller obtained in step (4) for the PCS model and evaluate the performance of the PCS performance using settling time and time-to-solution as performance metric when the system is subjected to step signal. All simulations will be carried out in MATLAB R2015a.

1.5 Dissertation Organization

The general introduction of this research has been presented in Chapter One. The remaining chapters are organized as follows: Firstly, a comprehensive review of related literature and important fundamental concepts about Fruit Fly Optimization Algorithm, LQR, PCS, Standard Optimization Test Functions, are carried out in Chapter Two. Secondly, a detailed approach and important mathematical models describing the development of the modified fruit fly optimization algorithm are presented in Chapter Three. Thirdly, the analysis, performance and discussion of the results are shown in Chapter Four. Finally, conclusion and recommendations of further work makes up the Chapter Five. The list of references pertinent to the research and MATLAB codes in the appendices are presented at the end of this dissertation.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

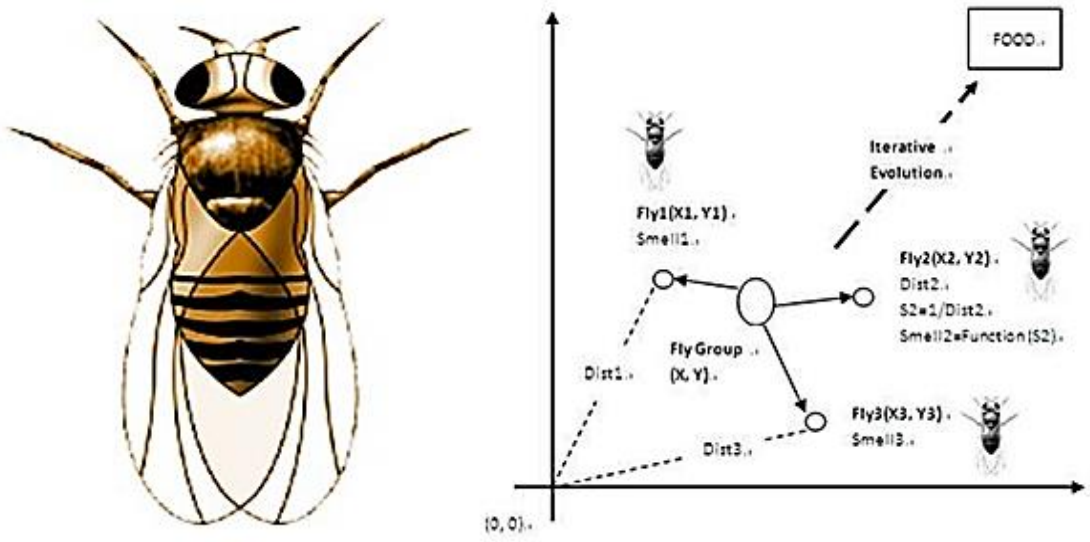
The literature review comprises of the review of fundamental concepts and the review of similar works. In the review of fundamental work, some of the existing works and the fundamental theories of all the algorithms that will be used for the realization of this research are reviewed, and then similar works are reviewed.

2.2 Review of Fundamental Concepts

In this section, concepts fundamental to the research such as: fruit fly optimization algorithm, LQR, and pitch control system of an aircraft, amongst others are reviewed.

2.2.1 The fruit fly optimization algorithm (FFOA)

Wen-Tsao Pan in 2011 developed a new algorithm known as fruit fly optimization algorithm (FFOA), which was based on the inspiration of the fruit fly, which is superior to other species in sensing and perception, especially in osphresis and vision. They feed chiefly on rotten fruits. In the process of finding food, their osphresis organs smell all kinds of scents in the air, it can even smell food source from 40 km away (Pan, 2011). They then fly towards the corresponding locations. When they get close to the food locations, they find foods using their visions and then fly towards that direction. FFOA has many advantages such as a simple structure, ease of implementation and speed to acquire solutions. The foraging process of a fruit fly group is shown in Fig. 2.1 (Li *et al.*, 2013):



(a) The Body Structure (b) Iterative Search for Food Diagram

Fig 2.1: Fruit Fly Structure and Search Mechanism (Li *et al.*, 2013)

Despite these advantages, the FFOA however suffers from the problem of lack of balance between exploration and exploitation in that it has a higher rate of exploitation than exploration leading to a higher probability of it being trapped in some local optima (Rizk-Allah, 2016). It also has the drawback of having a fixed value of search radius (Pan *et al.*, 2014) leading to poor exploration capability. In order to address this issue, an iteration factor was introduced so that the search radius can be adaptively changed for different evolution phases (Pan *et al.*, 2014). However, the iteration factor introduced by Pan *et al.* (2014) resulted in an increase of step size as the iteration number increases which limits the exploration capability of the algorithm.

The basic FFOA consists of four continuous phases. These are initialization, osphresis foraging, population evaluation, and vision foraging. Firstly, FFOA sets its control parameters, i.e., population size and a finish criterion, and initializes its fruit fly swarm

location. Then FFOA is repeated with the search process of osphresis foraging and vision foraging until the finish criterion is satisfied. At the osphresis foraging phase, a population of fruit flies randomly searches food sources around the fruit fly swarm location. After that, the smell concentration value or fitness value is evaluated for each of the food sources. In the vision foraging phase, the best food source with the maximum smell concentration value is found and then the fruit fly group flies towards it. The mathematical description of FFOA is as follows (Pan, 2012.):

To implement the FFOA, the following basic steps are used for calculation:

Step 1, confirm the fruit fly group's location by random (X-axis, Y-axis).

Step 2, endow personal fruit fly with random value for looking for food and location(X, Y).

X= X-axis+ Random Value; (2.1)

Y= Y-axis+ Random Value. (2.2)

Step 3, calculating the personal fruit fly's distance from the zero point, and find out the S value of density. This value equals to the inverse value of distance.

$$Dist = \sqrt{X^2 + Y^2}; S = 1/dist \quad (2.3)$$

Step 4, bring the smell density value S to the density judge function, and find out the smell density value of personal fruit fly at confirmed location. Smell=Function(S).

Step 5, repeat step two to step four, calculate the smell density of all the fruit flies in the group, and find the fruit fly with largest and lowest smell density value.

Step 6, keep the smell density value and location (X, Y) of the best fruit fly, and the group fly to that location.

Step 7, enter in the iterative optimization, and repeat step two to step five, and check whether the smell density value is better than the previous one, if yes, carry on the step six. The flowchart of the FFOA is shown in Fig 2.2 (Li *et al.*, 2013)

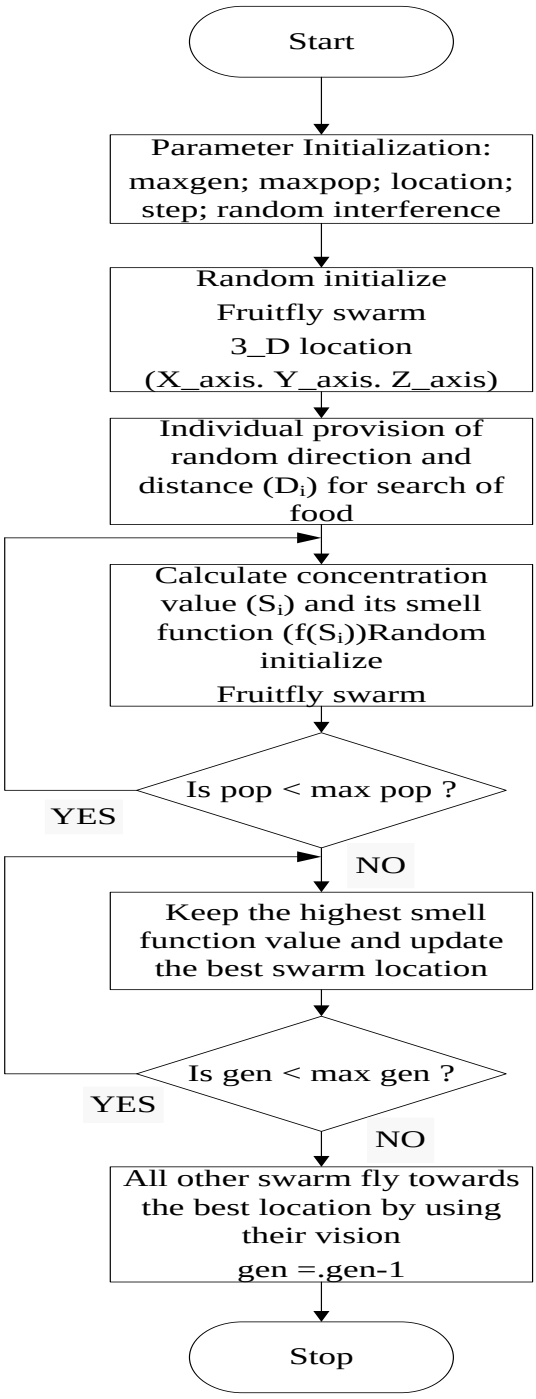


Fig 2.2: Flowchart of Standard Fruit Fly Optimization Algorithm (Li *et al.*, 2013)

2.2.2 Adaptive search radius

The implementation of FFOA basically consists of four (4) major steps. These steps include parameter initialization, osphresis foraging, population evaluation and vision foraging. In the osphresis foraging state, a new solution is evaluated by adding each decision variable of the swarm location and a random value in the range of $[-1,1]$ (Pan *et al.*, 2014). This means that, the swarm location of FFOA is rounded within a fixed radius of 1, which has resulted in an unbalanced behaviour of the FFOA while searching for the optimum solution in a given solution space. For example, at the early stage of the algorithm, the fruit fly swarm location is often far from an optimum solution. Thus, this search radius may be too small and considerable increase in iterations may be required to find a promising region within the solution space. In order to address this, the fixed radius was adaptively changed with iteration factor as introduced in (Pan *et al.*, 2014):

$$\lambda = \lambda_{\max} \times \exp\left(\log\left(\frac{\lambda_{\min}}{\lambda_{\max}} \right) \times \left(\frac{Iter}{Iter_{\max}} \right) \right) \tag{2.4}$$

where λ is the search radius in each iteration, $\lambda_{\max}$ is the maximum radius, $\lambda_{\min}$ is the minimum radius, $Iter$ is the iteration number and $Iter_{\max}$ is the maximum iteration number. The adaptive behaviour given in equation (2.4), significantly improved the performance of the FFOA. However, the $\left(\frac{Iter}{Iter_{\max}} \right)$ iteration factor in equation (2.4) indicates an increasing function, this result in a larger step size which limits the exploration capability of the algorithm. Thus, in this research work, the adaptive behaviour in equation (2.4) was

modified to a decreasing iteration function $\frac{Iter_{\max} - Iter}{Iter_{\max}}$. This decreasing iteration function will improve the exploration capability of FFOA.

2.2.3 Linearly decreasing inertial weight

It has been stated, that, inertial weight strategy provides an efficient balance between exploration and exploitation. Thus, in this research, a linearly decreasing inertial weight was used as it performs better over a large number of optimization problems (Shi & Eberhart, 1999) and is given as:

$$w_k = w_{\max} - \frac{w_{\max} - w_{\min}}{Iter_{\max}} \times k \tag{2.5}$$

where,

w_k is the current inertial weigh, $w_{\max}$ is the maximum generated inertial weight, $w_{\min}$ is the minimum generated inertial weight and k is the kth position of inertial weight

2.2.4 Standard optimization test functions

These are standard optimization test functions used to validate the characteristics of optimization algorithms and to compare their various performances (Li *et al.*, 2013). Many test functions have been reported in literatures, but there is no standard set of test functions needed to be followed. New optimization algorithms should be tested using at least a subset of functions with diverse properties so as to make sure that the tested algorithm can solve certain types of optimization problems efficiently (Tang *et al.*, 2007).

In this work, ten (10) test functions will be used to investigate the optimization capability of mFFOA. These test functions have diverse properties to include a wide variety of problems, such as unimodal, multimodal, regular, irregular, separable, non-separable and

multi-dimensional problems(Jamil & Yang, 2013). The test functions considered in this research are described as follows.

1. **Ackley function:** This function is one of the commonly used test function which is continuous, differentiable, non-separable, scalable and multi-modal. The mathematical equation for this function is expressed as follows(Bäck & Schwefel, 1993):

$$f_{Ackley}(x) = -2o\exp\left[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right] - \exp\left[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right] + 20 + e \qquad (2.6)$$

   where $n = 1, 2, \ldots$ and $-32.768 \le x_i \le 32.768$ for $i = 1, 2, \ldots, n$. This function has global minimum $f_* = 0$ at $x_* = (0, 0, \ldots, 0)$.

   This function has global optimum of $f(x_*) = 0$ at $x_* = 0$ for $-100 \le x_i \le 100$

2. **Alpine function:** Alpine function is continuous, non-Differentiable, separable, non-scalable, and multimodal and is defined as (Rahnamayan *et al.*, 2007):

$$f(x) = \sum_{i=1}^{n} \left| x_i \sin(x_i) + 0.1x_i \right| \qquad (2.7)$$

   This function has global optimum of $f(x_*) = 0$ at $x_* = 0$ for $-10 \le x_i \le 10$

3. **Eggcrate test function:** This function is Continuous, Separable, Non Scalable and is mathematically defined as follows(Yang, 2010c):

$$f(x, y) = x^2 + y^2 + 25(\sin^2 x + \sin^2 y) \qquad (2.8)$$

   This function has global minimum $f_* = 0$ occurs at $x_* = (0, 0)$ in the domain of $(x, y) \in [-2\pi, 2\pi] \times [-2\pi, 2\pi]$.

4. **Griewank test function:** Griewank's function is similar to Rastrigin's function. It has many widespread local minima (Hansen, 2006). However, the location of the minima are regularly distributed and has the following diverse properties: continuous, differentiable, non-separable, scalable, and multimodal The mathematical expression for this function is as follows (Fister Jr *et al.*, 2013):

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{2.9}$$

where $n = 1,2,...,$ and $i = 1,2,...,n$ for $-600 \le x_i \le 600$. This function has a global minimum of $f_* = 0$ at $x_* = (0,0,...0)$.

5. **Pathologic function:** Pathologic function is continuous, differentiable, non-separable, non-scalable and multi-modal and is defined as (Rahnamayan *et al.*, 2007):

$$f(x) = \sum_{i=1}^{n} \left(0.5 + \frac{\sin^2\left(\sqrt{100x_i^2 + x_{i+1}^2}\right) - 0.5}{1 + 0.001(x_i^2 - 2x_i x_{i+1} + x_{i+1}^2)^2} \right)^2 \tag{2.10}$$

6. **Rastrigin test function:** Based on the Sphere's function, the Rastrigin's function adds cosine modulation to create many local minima. Because of this feature, the function is multimodal. The mathematical expression for this function is as follows (Fister *et al.*, 2013):

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)], \tag{2.11}$$

This function has global minimum $f_* = 0$ at $x_* = (0,0,...,0)$ with a constraint of $-15 \le x_i \le 15$.

7. **Rosenbrock test function: :** Rosenbrock's valley is a classic optimization problem, which is also known as Banana function (Tang, K. *et al.*, 2007b). This is continuous, differentiable, non-separable, scalable and unimodal (Jamil & Yang, 2013). The Rosenbrock function is mathematically defined as follows:

$$f(x)=\sum_{i=1}^{n-1}((x_i-1)^2+100(x_{i+1}-x_i^2)^2) \tag{2.12}$$

This function has global minimum $f_*=0$ occurs at $x_*=(1,1,...,1)$ in the domain of $-2.048\le x_i\le 2.048$ where $i=1,2,...,n-1$. In the 2D case, it is often written as:

$$f(x,y)=(x-1)^2+100(y-x^2)^2 \tag{2.13}$$

8. **Schaffer function:** Schaffer is a continuous, differentiable, non-separable, non-scalable, multimodal non-convex optimization test function defined as:

$$f(x)=\sum_{i=1}^{30}\left(x_i^2+x_{i+1}^2\right)^{0.5}\left\{\left[\sin 50\left(x_i^2+x_{i+1}^2\right)^{0.1}\right]^2\right\} \tag{2.14}$$

The Schaffer function has the global minimum $f_*=0$ at $x_*=(0,0,.....0)$ in the domain $-100\le x_i\le 100$ (Li, X. *et al.*, 2013)

9. **Sphere test function:** This is the simplest form of DeJong function. Sphere function is continuous, differentiable, separable, scalable, and multimodal and is mathematically defined as follows (Schumer & Steiglitz, 1968):

$$f(x)=\sum_{i=1}^{d}x_i^2 \tag{2.15}$$

This function is unimodal and convex with an obvious local minimum of $f_* = 0$ at $x_* = (0,0,...,0)$ in a domain of $-15 \le x_i \le 15$.

10. **Whitley test function:** Whitley function is continuous, differentiable, separable, scalable, multimodal (Jamil & Yang, 2013) and is mathematically defined as follows (Pan *et al.*, 2014):

$$\sum_{k=1}^{n} \sum_{j=1}^{n} \left(\frac{y_{jk}^2}{4000} - \cos\left(y_{jk}\right) + 1 \right), \quad y_{jk} = 100\left(x_k - x_j^2\right)^2 + \left(1 - x_j^2\right)^2 \tag{2.16}$$

with global optimum $x^* = (1,1.....,1)$ and $f\left(x^*\right) = 0$ for $-100 \le x_i \le 100$.

2.2.5 Linear quadratic regulator (LQR) controller

The linear quadratic regulator (LQR) is a linear optimal controller that ensures that a feedback control system returns to its state of equilibrium in the presence of disturbances or perturbations. The weighting matrices (Q which is a positive definite matrix that penalizes the states and R which is also a positive definite matrix that penalizes the control inputs) are the critical design parameters for the LQR and are usually selected by the designer (Hamidi, 2012). The cost function for the LQR, which is the performance index, is given in:

$$J = \frac{1}{2} \int_0^{\infty} \left(X^T Q X + U^T R U \right) dt \tag{2.17}$$

where J is the cost function, Q and R are positive-definite Hermitian or real matrices (weighting matrices) used to determine the relative importance of the state variables and

control inputs (expenditure of energy) respectively. X^T and U^T are the transpose matrices of *X* and *U* (state variables and input function) respectively.

The other parameters, the state and input matrices (A and B), are obtained from the state model of the system.

2.2.5.1 Determination of optimal values of weighting matrices in LQR problems

As stated earlier, the critical design parameters for LQR are the weighting matrices in the objective function (cost function) and are normally selected by the designer. These weighting matrices Q and R have profound effect on controller performance. Determining the best Q and R is usually time-consuming due to its trial-and-error approach requiring several computer iterations and simulations. The LQR is an optimal feedback control approach that minimizes the excursion in state trajectories of a system while requiring minimum control effort (Cuong *et al.*, 2015). Before the linear quadratic regulator theory is introduced, some definitions that are essential to the theory are presented. Consider the state space representation of a linear time invariant (LTI) system (Monfort *et al.*, 2015)

$$\dot{x} = Ax + Bu \tag{2.18}$$

$$y = Cx + Du \tag{2.19}$$

where x is an n-dimensional state vector, u is an m-dimensional input vector and y is a p dimensional output vector. A, B, C, and D are constant matrices of the appropriate size. Assuming that, the system is controllable, the LQR approach determines a linear state feedback law (Monfort *et al.*, 2015):

$$u = -Kx \tag{2.20}$$

From equation (2.18) to equation (2.20), the closed-loop system becomes:

$$\dot{X} = (A - BK)x \tag{2.21}$$

and the closed loop eigenvalues are obtained from equation (2.21):

$$\det(\lambda I - A + BK) = 0 \tag{2.22}$$

with the appropriate feedback gain (K), the closed loop poles can be shifted to the desired locations. The optimal control law minimizes the quadratic performance index (cost function) which consists of state and control (inputs) energies as in equation (2.17) (Cuong *et al.*, 2015).

A feature of optimal controllers (such as the LQR) is that their introduction to systems will stabilize such systems irrespective of the open-loop stability status. The optimal solution is the control gain matrix (Ghoreishi *et al.*, 2011):

$$K = R^{-1}B^{T}P \tag{2.23}$$

Where P is the unique symmetric, positive and semi-definite matrix solution to the algebraic Riccati equation given as (Ata & Coban, 2015):

$$AP + A^{T}P + Q - PBR^{-1}B^{T}P = 0 \tag{2.24}$$

Equation (2.24) shows that the choice of Q and R will influence P, the solution of the algebraic Riccati equation.

2.2.5.2 Methods of determining weighting matrices (Q and R) of LQR

The problem of selecting weighting matrices has been investigated by several researchers using different methods (standard and intelligent).

1. **The Standard Methods**

In this, normal methods are used to determine the state Q and control R weighting matrices these includes: conventional, Bryson's, pole placement etc.

A) Conventional Method: Generally, weighting matrices are determined by a trial-and-error method in which an expert adjusts the weighting matrices intuitively, and then refines them iteratively to obtain a satisfying performance. The trial-and-error method is not feasible for high dimensional systems and even for simpler systems; it is labour -intensive and time consuming approach.

Fig 2.3 shows the flowchart for the determination of Q and R using standard method.

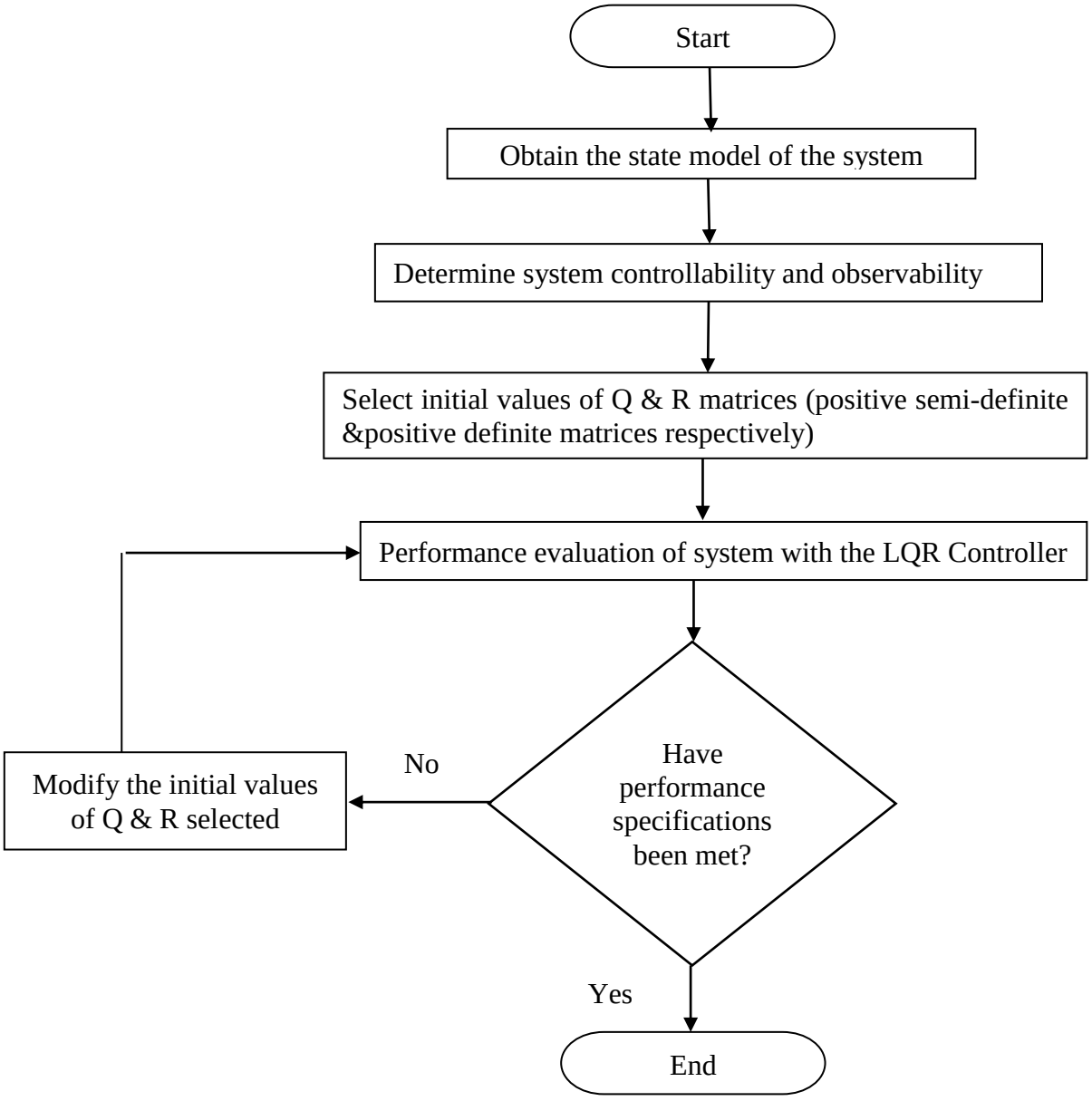


Fig 2.3: Flowchart for Determining the Q and R of LQR Controller using Standard Method

B) Bryson's Method: Bryson's method (Johnson & Grimble, 1987) is another iterative method. In this technique, initially, state and feedback variables are normalized with respect to their largest permissible, and utilized to initialize the weighting matrices. Then, similar to trial-and-error method, the weighting matrices are gradually refined to approach the minimum index value.

C) Pole placement Method: This is another popular technique for determining the weighting matrices. However, in pole placement technique; the weighting matrices are based on the given poles, and thus cannot guarantee the performance and constraints of the system.

However, these techniques only aim to minimize the quadratic performance index and control objectives were not considered such as minimizing the overshoot, rise time, settling time, and steady state error. Computational intelligence paradigms have been exploited to enhance the accuracy, design process and performance of the system, and lessen the shortcomings of classic techniques.

2. Computational Intelligence Methods

Computational intelligence methods are methods used in solving various problems of artificial intelligence with the use of computers to perform numerous calculations (Er & Oentaryo, 2011). These calculations contain the applications of six major techniques: neural networks, fuzzy logic, evolutionary algorithms, rough sets, uncertain variables, and probabilistic methods (Er & Oentaryo, 2011).

Evolutionary algorithm are inspired from the process of natural evolution(Al-salami, 2009). Among them, are: genetic algorithm (GA), particle swarm optimization (PSO)

algorithm, weighted artificial fish swarm algorithm (wAFSA), artificial bee colony, firefly algorithm, fruit fly optimization algorithm (FFOA) etc.:

A) Genetic algorithm (GA): This method (Haupt & Haupt, 2004) is one of the evolutionary algorithms inspired by natural evolution and it is the most frequent applied algorithm. In GA, the proposed responses for an optimization problem are considered as living creatures. The most important operators and mechanisms used in GA with their equivalencies in nature are:

   a. Selection operator which is equivalent to natural selection phenomenon
   b. Crossover operator which is equivalent to reproduction phenomenon
   c. Mutation operator which is equivalent to genetic mutation phenomenon

   By using these operators on current population, new population emerges which is the average of them and not worse than the current population, nevertheless the average is quite often better. Thus, as time goes the GA gives better response for optimization problem.(Abdulla *et al.*, 2013; Ghoreishi *et al.*, 2011; and Wongsathan & Sirima, 2009) applied GA to improve the performance of LQR. Although GA is well-founded and indicates high exploration capability, it suffers from two pitfalls: low exploitation and convergence speed.

B) Particle swarm optimization (PSO) algorithm: This method was first introduced by James Kennedy and Rassel Eberhart (Kennedy & Eberhart, 1995). In PSO, a number of simple entities, namely the particles, are placed in the search space of some problem or function, and each evaluates the objective function at its current location. The algorithm, searches a space by adjusting the trajectories of particles as they are conceptualized as moving points in multidimensional space. The

individual particles are drawn stochastically toward the positions of their own previous best performance and the best previous performance of their neighbors. This method is able to reach the globally optimal solution within a few iterations. It has been experimentally shown that the PSO is scalable and its processing time grows at a linear rate with respect to the size of the problem (Akay, 2013).

C) Artificial fish swarm algorithm (AFSA): The basic idea of fish in water is to discover regions with more food, by vision or by sense through its intelligent behaviours (*Preying*, *Swarming* and *chasing)*. Hence the environment where an artificial fish (AF) lives is mainly the solution space of the optimization problem (Wang & Li, 2015) and is usually the state of other fishes.

D) Weighted artificial fish swarm algorithm (wAFSA) proposed by Salawudeen & Mu'azu, (2015) adopted an inertial weight selection such that the algorithm can adaptively select its parameters (visual and step size) when searching for global solution. Crossover operator was applied to fuse the AFSA and the modified cultural artificial fish swarm algorithm (mCAFAC), in order to enhance its convergence to a global minimal. The wAFSA was then applied to determine the optimal values of the weighting matrices (Q and R) of linear quadratic regulator (LQR) controller. This was tested on the quadruple inverted pendulum (QIP) model and was able to stabilize the model in less time compared to using the conventional pole placement LQR and standard AFSA methods.

E) Artificial Bee Colony (ABC) Algorithm: The ABC algorithm was introduced by Karaboga in 2005. The method is based on the intelligent behavior of honey bee swarms finding nectar and sharing the information of food resources with each

other in the field of swarm intelligence to solve optimize numeric benchmark functions (Karaboga, 2005.). The general procedure of the ABC algorithm contains four phase; initialization phase, employed bees phase, onlooker bees phase, and scout bees phase. The ABC algorithm has the advantages of strong robustness, fast convergence and high flexibility, fewer control parameters and also it can be used for solving multidimensional and multimodal optimization problems (Karaboga & Akay, 2009; Karaboga & Basturk, 2007).

F) Firefly algorithm (FA**):** The FA was developed by Xin-She (Yang, 2008) and it is based on idealized behavior of the flashing characteristics of fireflies. For simplicity, we can summarize these flashing characteristics as the following three rules:

   a. All fireflies are unisex, so that one firefly is attracted to other fireflies regardless of their sex.

   b. Attractiveness is proportional to their brightness, thus for any two flashing fireflies, the less bright one will move towards the brighter one. The attractiveness is proportional to the brightness and they both decrease as their distance increases. If no one is brighter than a particular firefly, it will move randomly.

   c. The brightness of a firefly is affected or determined by the landscape of the objective function to be optimized (Yang, 2008).

G) Fruit fly optimization algorithm (FFOA): The FFOA was developed by Pan in 2011, which works based on the foraging behaviour of the fruit fly on food, the fruit fly is superior to other species in sensing and perception, especially in

osphresis and vision. It has been used to solve real world and engineering optimization problems (Pan *et al.*, 2014).

For the purpose of this research work mFFOA is used to determine the weighing matrices of the LQR.

2.2.6 Pitch control system of an aircraft

Pitch is defined as a rotation around the lateral or transverse axis, which is parallel to the wings of an aircraft, and is measured as the angle between the direction of speed in a vertical plane and the horizontal line. The deflection of the elevator brings about changes of pitch, which raises or lowers the nose and tail of the aircraft. When the elevator is raised (defined as negative value), the force of the airflow will push the tail down, thereby causing the nose of the aircraft to rise and the altitude of the aircraft to increase. One of the aims of a pitch control system is to help a pilot to control an aircraft by keeping the pitch attitude constant, that is, make the aircraft return to desired attitude in a reasonable length of time after a disturbance of the pitch angle, or make the pitch follow a given command as fast as possible(Ju & Mohamed, 2007).

The three primary ways through which an aircraft change its direction relative to the passing air is shown in Fig 2.4.

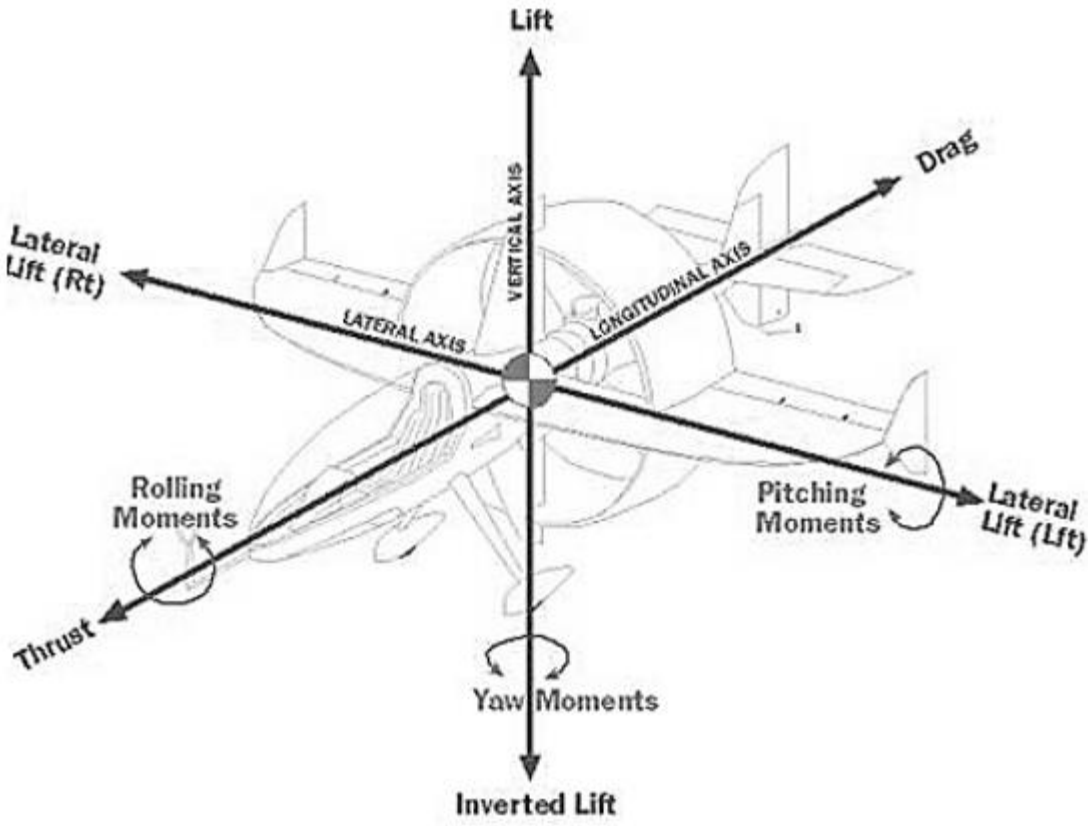


Fig 2.4: Forces, and Moments acting on an Aircraft (Jisha & Aswin, 2015)

Fig 2.4 shows the forces, moments and velocities acting on an aircraft, and the three ways through which an aircraft changes its direction are:

A) Pitch: This is the movement of the aircraft about the lateral axis causing the nose and tail of the aircraft to move up or down.

B) Roll: This is the movement of the aircraft about the longitudinal axis, i.e., the axis which runs along the length of the aircraft

C) Yaw: This is the movement of the aircraft left or right.

Longitudinal motion: are those movements through which the aircraft has to move on within the x-z plane and rotation about y-axis (pitch). The three longitudinal equations of motions are derived from the X-force, Z-force and y –moments of an aircraft (Jisha & Aswin, 2015) and the pitch control system is shown in Fig 2.5.

$$X - mg\sin\theta = m\left(\dot{u} - rv + qw\right) \tag{2.25}$$

$$M = I_Y \dot{q} + I_{XZ}\left(p^2 - r^2\right) + rp\left(I_X - I_Z\right) \tag{2.26}$$

$$Z + mg\cos\theta\cos\phi = m\left(\dot{w} - pv + qu\right) \tag{2.27}$$

Equations (2.25, 2.26 and 2.27) represents the longitudinal (PCS) equations of motions of an aircraft.

where, X and Z is are the aerodynamics force components, p, q and r are the angular velocities along x, y and z axes respectively, u, v and w are the forward(x), pitch (y) and vertical (z) linear velocity and Θ, ϕ and δe represent the orientation of aircraft (pitch angle), roll angle in the earth axis system and elevator deflection angle respectively.

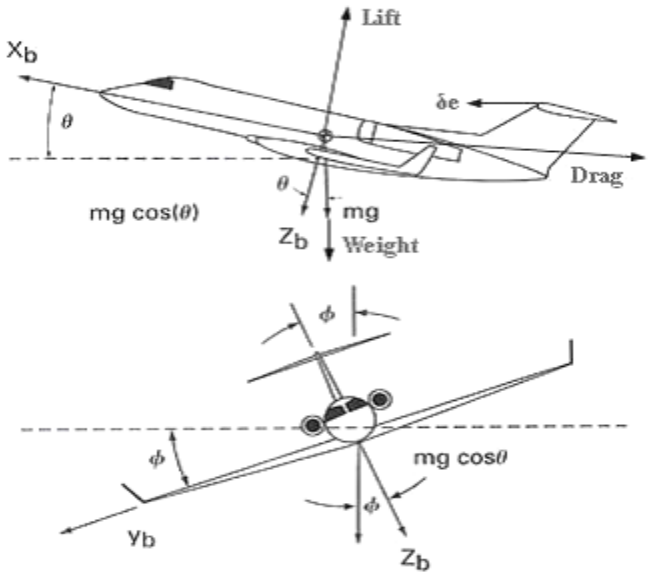


Fig 2.5: Pitch Control System of Aircraft (Jisha & Aswin, 2015)

Fig 2.5 shows the pitch control system of aircraft, where, Xb and Zb are the aerodynamics force components, p, q and r are the angular velocities along x, y and z axes respectively, u, v and w are the forward(x), pitch (y) and vertical (z) linear velocity and Θ, ϕ and δe represent the orientation of aircraft (pitch angle), roll angle in the earth axis system and elevator deflection angle respectively.

2.2.7 Performance metrics

The improvement, effectiveness, efficiency, and appropriate levels of internal controls are determined by performance metric of a system (Alagoz *et al.*, 2015).To determine the time response behaviour of the LQR controller, time-domain analysis was carried out.

The response of a dynamic system to an input can be expressed as a function of time. The time response of systems can be calculated if the nature of the input is known and the mathematical model of the system can be determined in time-domain analysis. The time response basically has two components: transient response and the steady-state response.

1. **Transient response**: This is defined as that part of the system response that goes to zero as time goes to infinity and is dependent upon the system poles only and not on the type of input. Therefore, the transient response can be analyzed using a step input. The steady-state response shows where the system output ends as time approaches infinity and depends on system dynamics and the input quantity (Burns, 2001).

   The transient-response characteristics of a control system to a unit step input are identified by the following: Peak overshoot, M_p, peak time, t_p, settling

time, t_s, steady-state error, e_{ss}, rise time, t_r and delay time, t_d as in Figure 2.6 (Rahim, 2009).

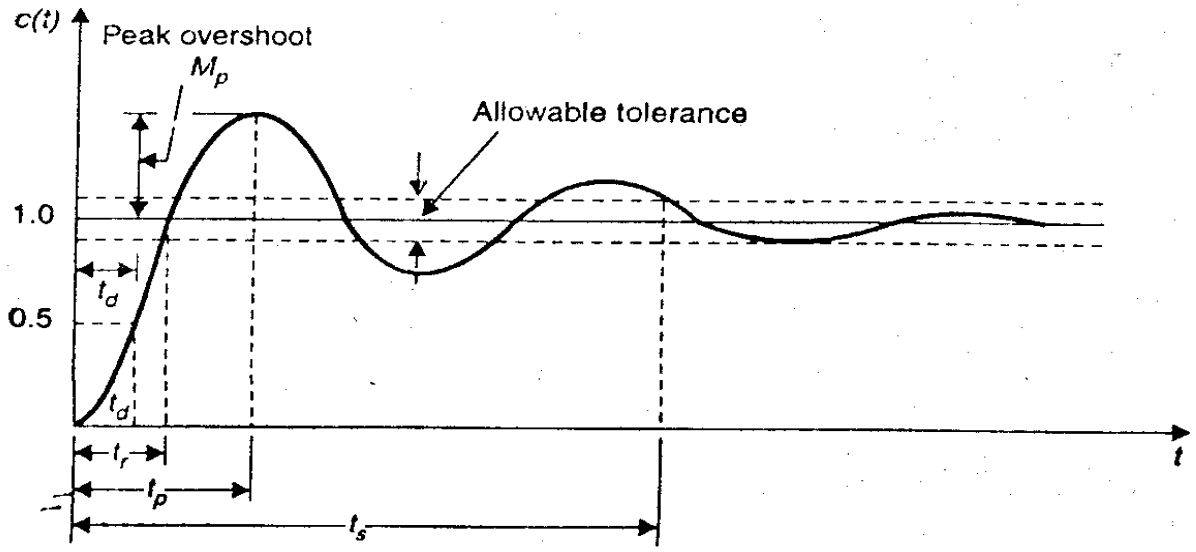


Fig 2.6: Transient Response Characteristics of a System (Rahim, 2009).

A) **Delay time (t_d):** Time required for the response to reach within 50% of the final steady-state value

B) **Rise time (t_r):** Time required for the response to rise from 0 to 90% of the final steady-state value

C) **Peak time (t_p):** Time required for the peak overshoot of the response to occur

D) **Settling time (t_s):** Time required for the response to reach and stay within 2% to 5% of its final steady-state value.

E) **Peak overshoot (M_p):** Normalized difference between the time response peak and the steady output and is defined as:

$$\%M_p = \frac{c(t_p) - c(\infty)}{c(\infty)} \times 100\% \tag{2.28}$$

where $c(t_p)$ is the time response peak and c(∞) is the steady output

Equation (2.28) shows how peak/maximum overshoot is calculated from transient response of a system

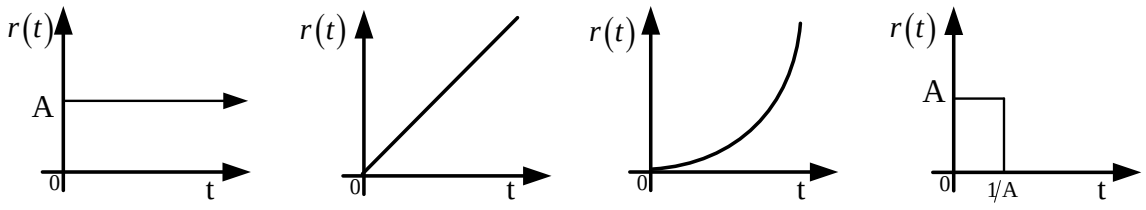
F) **Steady-state error (e_{ss}):** Error between the actual output and desired output as 't' tends to infinity and is defined as:

$$e_{ss} = \lim_{t \to \infty}[r(t) - c(t)] \tag{2.29}$$

where, r(t) and c(t) are the actual input and desired output respectively.

Equation (2.29) shows how steady-state-error is calculated from transient response of a system

Robustness: This is the low sensitivity to disturbances or perturbations not necessarily considered in the design and analysis of systems. The process/plant dynamics of a robust system can change but, its performance is not expected to weaken to an intolerable level, i.e. the system should be able to bear those effects when executing the tasks for which it was designed. The disturbances and the dynamic behaviour of a system are simulated and analysed using the standard test signals: impulse (sudden shock), step (sudden change), ramp (constant velocity) and parabolic (constant acceleration) as shown in Fig 2.7 (Rahim, 2009).



a) Step b) Ramp c) Parabolic d) Impulse

Fig 2.7: Standard Test signals (Rahim, 2009)

2. **Time-to-solution:** As stated earlier, the conventional method of determining the weighting matrices of LQR (Q and R) is time consuming. In some cases, it may take up to hours or even days before the right combination of this matrices are obtained. In practical system, it is often required that the action of the controller on the system should be as fast and robust as possible since the stability of physical system is of paramount importance.

Though the LQR is an optimal controller, its performance strongly depend on how fast and well the appropriate value of this weighting matrices is able to achieve stability. Therefore, the time to obtain these matrices is crucial and will be employed as a performance metric in this research.

2.3 Review of Similar Works

In this subsection, some of the relevant literatures directly related to the study are reviewed and presented in this section based on two groups: the first part of the review is focused on works based on the modification of the FFOA and the second part is focused on the use of metaheuristics search algorithm to determine the LQR weighting matrices (Q & R).

2.3.1 Review of Research Works Based on Modification of FFOA

The review of research works done in the area of evolution of the fruit fly optimization algorithm (FFOA) is presented in this section.

Pan (2011) presented a new metaheuristic fruit fly optimization-inspired algorithm based on the food finding behavior of the fruit fly. The fruit fly itself is superior to other species in sensing and perception, especially in osphresis and vision. The osphresis organs of fruit flies can find all kinds of scents floating in the air; it can even smell food source from 40 km away. Then, after it gets close to the food location, it can also use its sensitive vision to find food and the company's flocking location, and fly towards that direction too. The new fruit fly algorithm optimized general regression neural network and multiple regression were adopted to construct a financial distress model for a Taiwan enterprise. The results showed that the RMSE value of the Fruit Fly Optimization Algorithm optimized general regression neural network model had very good convergence classification and prediction capability. However, the algorithm easily got trapped in local optima due to its poor rate of exploration of the solution search space.

Li *et al.*, (2013) proposed a hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. In this work the generalized regression neural network (GRNN) was used for annual power load forecasting. The FFOA was used automatically to select the appropriate spread parameter value for the GRNN power load forecasting model. The effectiveness of the proposed hybrid model was analyzed using two experiment simulations. The results obtained were compared with some other GRNN hybrids, both showed that the proposed hybrid model

outperforms the GRNN model with default parameter, GRNN model with particle swarm optimization (PSOGRNN), least squares support vector machine with simulated annealing algorithm (SALSSVM), and the ordinary least squares linear regression (OLS_LR) forecasting models in the annual power load forecasting. However, the algorithm being trapped in local optima due to its poor rate of exploration of the solution search space still existed.

Lin (2013) carried out analysis of service satisfaction in web auction logistics service using a combination of Fruit fly optimization algorithm and general regression neural network. In the work fruit fly optimization algorithm (FOA) was adopted to optimize artificial neural network (ANN) model. Principal component regression was carried out on the results data of a questionnaire survey on logistics quality and service satisfaction of online auction sellers to form the logistics quality and service satisfaction detection model. Important principal components in the principal component regression analysis results were selected for independent variables, and overall satisfaction level toward auction sellers' logistics service as indicated in the questionnaire survey was selected as a dependent variable for sample data. The FOA-optimized general regression neural network (FOAGRNN), PSO-optimized general regression neural network (PSOGRNN), and other data mining techniques for ordinary general regression neural network were used to form a logistics quality and service satisfaction detection model. Four-Six principal components in principal component regression analysis were selected as independent variables of the model. Analysis of the results showed that out of the four data mining techniques, FOA-optimized GRNN model has the best detection capacity. However, this work only applied

the FFOA to optimize the ANN model, the problem of poor exploration capability associated with the FFOA still existed.

Xu & Tao (2013a) presented variables screening method based on the algorithm of combining fruit fly optimization algorithm and radial basis neural network (RBF). In their work an RBF neural network was optimized using the standard FFOA and established the network model. This method with mean impact value (MIV) were adopted in variables selection. The strength of this model was tested using two examples. Though the method was found to be, simple to learn, more stable and practical. However, this work only applied the FFOA to optimize the RBF model in variable selection, the problem of the FFOA getting trapped in local optima still existed.

Xu & Tao (2013b) proposed an improved fruit fly optimization algorithm using bivariable function. In the proposed modified FFOA (G-FOA) the modification was carried out on the density (S) calculation in order to avoid the limitation of the standard FFOA having a non-negative S value which is not possible in many problems. The performance of the proposed method was tested on three binary nonlinear function using only one classic nonlinear multimode state Rastrigin test function. The results was compared with particle swarm optimization, genetic algorithm and simulated annealing algorithm. The proposed method have the least time-to-solution. However, the problem of fixed search radius still existed.

Choubey (2014) proposed fruit fly optimization algorithm for travelling salesperson problem. The work proposed two variants of sensitive smell and three variants for vision functions. The two different variant of the neighborhood search method were proposed for

the implementation of the smell function: Best and Incremental Neighbor Method. The experiment was conducted with JDK 1.6 on an Intel Core™2 CPU with 2.66 GHz and 2 GB RAM. From the experimental results it was found that the smell function best neighbor method converges earlier as compared to the incremental neighbor method. While the sensitive vision function V1 and V3 were found to be comparatively effective in exploration of the search space and finding global optima respectively. The method proposed was applied to the instances of small number of cities. However the work did not consider the problem associated with the fixed search radius of the FFOA as such this problem still existed.

Iscan & Gunduz (2014) investigated a constant parameter concerning the direction of the FFO using grid search algorithm. The proposed method automatically adjusted the coefficient for each numeric benchmark function using grid search, and adapted the behavior of the method according to the problem. The performance of the proposed approach was evaluated on optimizing 2 different numeric functions. The experimental results showed that proposed FFO was better than the basic FFO in terms of solution quality. However, the work used the grid search algorithm only as an automated approach for finding the related parameter, as such the unbalanced exploration and exploitation behaviour of the FFOA still existed.

Hengyu *et al.*, (2014) proposed an improvement of fruit fly optimization algorithm for solving traveling salesman problems (TSP). The modified algorithm adopted the strategy of adaptive variable step size in order to increase search efficiency effectively. Mutation operator was introduced into the FFOA during the exploration so as to improve the diversity of the population and avoids premature convergence. The performance of the

proposed algorithm was compared with the standard fruit fly optimization algorithm and particle swarm optimization algorithm using nine (9) test functions in TSPLIB international standard library. The simulation results showed that the proposed method performed better. However, the best solution was easily corrupted by the mutation operator which in turn affected the diversification of the solution search space of the proposed algorithm.

Pan *et al.*, (2014) proposed improved FFO to solve high-dimensional functions. In the work, the search radius was changed dynamically with iteration number such that the search radius was adjusted for different evolution phases. Weights were generated randomly from a uniform distribution based on the number of points on Pareto fronts, and a weighted sum was used to combine all the objectives into a single objective. A new solution generating method was developed to enhance accuracy and convergence rate of the algorithm. The proposed approach was tested on 30 benchmark functions. The computational results showed that the proposed IFFO not only improved the basic fruit fly optimization algorithm but also performed better than five state-of-the-art harmony search algorithms. However, the work only improved the FFO to solve high-dimensional functions, as such the problem of poor exploration rate of the FFOA still existed.

Mhudtongon *et al.*, (2015) proposed modified fruit fly optimization algorithm for analysis of large antenna array. In this work the modified fruit fly optimization algorithm (MFOA) was used to analyze the radiation pattern of the large antenna array. The MFOA was improved by incorporating random search of two groups of swarm and self-adaptive population size feature into the conventional FOA in other to achieve wide search space. The results obtained was tested on three nonlinear test function and the modified

algorithm was found to be effective in solving the three nonlinear test functions and EM problems for the large antenna array. Though the work was able to overcome the problem of getting trapped into local optima, but does not converge to the optimum solution.

Rizk-Allah (2016) proposed a hybrid optimization algorithm FOA-FA to solve the nonlinear programming problems (NLPPs) based on FOA and fire fly algorithm. The work integrated the strength of a variation of original fruit fly optimization algorithm (FOA) in handling continuous optimization by employing a new adaptive radius mechanism (ARM) for exploring the whole scope around the fruit flies locations to overcome the drawbacks of original FOA and the merit of firefly algorithm (FA) in achieving robust exploration by updating the previous best locations of fruit flies to avoid premature convergence. The hybrid algorithm speeds up the convergence and improves the algorithm's performance. The proposed FOA-FA algorithm was tested on eleven benchmark problems and two engineering applications. The experimental simulation result showed that the hybrid algorithm speeds up the convergence and improved the algorithm's performance. However, the work did not consider the problem of diversification of the solution search space of the FA because it focused on the problem of weak local search ability and premature convergence of the FFO.

Wu *et al.*, (2016) proposed an improved fruit fly optimization algorithm (SEDI-FOA), based on selecting evolutionary direction intelligently. In their work a mechanism called selecting the evolutionary direction intelligently was introduced to overcome the problem of random search direction and distance of population individuals. The proposed algorithm was tested on twelve standard benchmark functions and the experimental results showed that the proposed algorithm performed well when compared with some of these extended

FFOA: fruit fly optimization algorithm based on differential evolution (DFOA), improved fruit fly optimization algorithm (IFFO) and improved fruit fly optimization algorithm (LGMS-FOA) and three widely-used evolutionary algorithms: PSO algorithm, the covariance matrix adaptation evolution strategy (CMAES) and the self-adaptive differential evolution (SaDE) algorithm. However, realization of quick convergence in practical problems still existed.

It is evident from the literatures that improving the diversification capability of FFOA has been given a significant research attention leading to the development of the improved algorithms. However, developing a modified FFOA using the concept of modifying the iteration factor of search radius introduced by (Pan *et al.*, 2014) to a decreasing function in order to improve the exploration capability of the FFOA and also introduced a linearly decreasing inertial weight in order to provide an efficient balance between exploration and exploitation of the FFOA was not done to the best of my knowledge.

2.3.2 Review of Research Works Based on the use of Metaheuristics Search Algorithm for Determination of LQR Weighting Matrices (Q and R)

The review in this section is focused on the use of metaheuristics search algorithms in the determination of LQR weighting matrices (Q and R)

Ghoreishi& Nekoui (2012) presented optimal weighting matrices design for LQR based on genetic algorithm and PSO. In this work, some important indices such as closed-loop pole locations, speed of response, and maximum level of control effort were considered and combined into an objective function. The genetic algorithm (GA) and particle swarm

optimization (PSO) were used to solve this problem. The proposed method was applied to a nonlinear flexible robot manipulator model. The results obtained from these algorithms showed that the PSO performed better than the GA. However, this algorithm has higher iteration number as such an algorithm with less computational time will be better.

Abdulla *et al.*, (2013) presented genetic algorithm (GA) based optimal feedback control weighting matrices computation. The work attempted to solve the difficulty associated with the selection of the LQR weight matrices using genetic algorithm GA thereby avoiding the trial and error involved in the state feedback technique. The proposed solution was used in the design of position controller of a robot arm. The results obtained showed that the proposed solution satisfied the specifications, for minimum overshoot, settling and rise times. However, it requires higher iterations before reaching the optimal value.

Ohri (2014) designed linear quadratic regulator (LQR) and proportional integral derivative (PID) controllers for pitch control system of an aircraft using genetic algorithm (GA) for tuning the parameter of the LQR and PID controllers. The result obtained showed that the PCS with LQR performed better than the PCS with the PID. However, it requires higher iterations before reaching the optimal value.

Ata & Coban (2015) presented an artificial bee colony algorithm based linear quadratic optimal controller design for a nonlinear inverted pendulum. The LQR was used to control an inverted pendulum as a nonlinear dynamical system and the Artificial Bee Colony (ABC) algorithm was used for selecting weighting matrices. The result obtained showed that the proposed method performed better than the conventional (trial and error) method. However, this method has the limitations of complexity and high computational time.

Mu'azu *et al.*, (2015) and Salawudeen & Mu'azu (2015) proposed weighted artificial fish swarm algorithm with adaptive behaviour based linear controller design for nonlinear inverted pendulum. In this work, first an approach called inertial weight into the standard artificial fish swarm algorithm (AFSA) was introduced to adaptively select its parameters (visual and step sizes) after which, the modified algorithm was used to determine the optimal values of LQR weighting matrices. The proposed method was used to stabilize a non-linear inverted pendulum. The result obtained was efficient in determining the weighting matrices of LQR in comparison with the conventional trial-and error approach. Simulation results showed the efficiency of the method proposed in this work. The complexity of implementing the preying, swarming and chasing behavior of fish is the limitation of this method.

Nagarkar & Patil (2016) presented the optimization of the linear quadratic regulator (LQR) control quarter car suspension system using genetic algorithm (GA). In this research, GA was used to determine the optimum weighting matrix parameters of the LQR. A Macpherson strut quarter car suspension system was implemented for ride control application. The GA was implemented with single objective (minimizing root mean square (RMS) controller force) and the analysis was extended to multi-objective optimization with objectives: minimization of RMS controller force and RMS sprung mass acceleration and minimization of RMS controller force, RMS sprung mass acceleration and suspension space deflection. Though the result obtained gave an acceptable value, but the method required a high computational time.

It is evident from the reviewed literatures that several metaheuristic search algorithms have been given relevant research attention leading to their application in the determination of the weighting matrices (Q and R) of LQR for different control engineering benchmark systems. However, the issues of costs still remain valid research areas. In order to address the issues of complexity and high computational time in determining the Q & R of LQR controller associated with most of the reviewed literature, this research developed a modified FFOA that use the concept of linearly decreasing inertial weight for the determination of optimized weighting matrices (Q and R) of LQR for a PCS. The mFFOA reduced the unbalanced behaviour of the FFOA which improved the exploration process of the standard FFOA.

CHAPTER THREE

MATERIALS AND METHODS

3.1 Introduction

In this chapter, the methods, materials and procedures employed for the successful completion of this research are discussed. The standard FFOA and the modified FFOA were replicated and developed respectively.

3.2 Computer System Specification

For each algorithm and for each test case, the best of the ten (10) tests were obtained using MATLABR2015a on Hewlett-Packard HP G62 with an Intel(R) Pentium(R) 2GHz processor and 3GB RAM with 64-bit Windows 7 pro Operating System (OS).

3.3 Initialization of FFOA, LQR and PCS Parameters

As discussed in subsection 2.2.1, the performance of FFOA depends on the appropriate selection of its control parameters (maximum generation, maximum population, location, step, and random interference). The appropriate values selected for these parameters are presented in Table 3.1

Table 3.1: FFOA Simulation Parameters

S/No	Simulation Parameters	Value
1	Population	1000
2	Generation	50
3	Step	Adaptive

The simulation parameters presented in Table 3.1 are used in determining the performance of the standard algorithm. The MATLAB script for the FFOA algorithm can be found in Appendix B.

The LQR controller used in this research consists of two weighting matrices (Q and R). The Q and R matrices are considered as smell function of the FFOA, performing the ospheresis foraging behaviour of FFOA described in subsection (2.2.1) and are determined using standard methods as shown in Fig 2.3 and equation (2.17).The appropriate values selected for the parameters of the mFFOA used in determining the weighting matrices (Q and R) of the LQR are presented in Table 3.2

Table 3.2: mFFOA (Q and R) Simulation Parameters

S/No	Simulation Parameters	Value
1	Population	500
2	Generation	50
3	Step	Adaptive

The simulation parameters presented in Table 3.2 are used in determining the weighting matrices (Q and R) of the LQR using the developed algorithm.

The PCS model equations (2.25), (2.26) and (2.27) adopted from (Jisha & Aswin, 2015) was obtained after substituting the data from the General Aviation airplane.Table 3.3 shows the data from the General Aviation airplane.

Table 3.3: Longitudinal Derivative Stability Parameters (Jisha & Aswin, 2015)

Longitudinal Derivatives	**Components**		
	Dynamics Pressure and Dimensional Derivative Q = 36.8lb/ft^2,QS= 6771lb, QS c = 38596ft.lb, (c / 2u_0) = 0.016s		
	Z-Force, (F^{-1})	Pitching Moment, (FT^{-1})	Pitching Moment, (FT^{-1})
Rolling velocities	$X_u = -0.045$	$Z_u = -0.369$	$M_u = -0.369$
Yawing velocities	$X_W = 0.036$	$Z_W = -2.02$	$M_W = -0.05$
	$X_{\dot{W}} = 0$	$X_W = 0$	$M_{\dot{W}} = 0$
Angle of attack	$X\alpha = 0$	$Z_\alpha = -355.42$	$M_\alpha = -8.8$
	$X_{\dot{\alpha}} = 0$	$Z_{\dot{\alpha}} = 0$	$M_{\dot{\alpha}} = -0.8976$
Pitching rate	$X_a = 0$	$Za = 0$	$M_a = -2.05$
Elevator deflection	$X_{\delta e} = 0$	$Z_{\delta e} = -28.15$	$M_{\delta e} = -11.874$

To solve the aircraft completely the following model assumptions are made (Jisha & Aswin, 2015):

1. The aircraft is steady state cruise at constant altitude and velocity
2. The change in pitch angle does not change the speed of the aircraft under any situation

3.4 Standard Fruit Fly Optimization Algorithm

In this research, the standard FFOA was first replicated as described in Fig 2.2. The ospheresis and vision steps are described in subsection 2.2.1. The MATLAB command implementation of the FFOA is given as in Appendix B.

3.5 Development of the Modified Fruit Fly Optimization Algorithm

As stated earlier, the main shortcoming of the standard FFOA is lack of balance between exploration and exploitation in that it has a higher rate of exploitation than exploration leading to a higher probability of it being trapped in some local optimal and the drawback of having a fixed value of search radius. In order to develop the mFFOA, the following processes were added to the oshphresis foraging step of the standard FFOA: a decreasing iteration factor for an adaptive search radius and a linearly decreasing inertial weight.

3.5.1 Decreasing iteration function for an adaptive search radius

As discussed in section 2.2.2 an iteration factor was introduced to the fixed search radius so that the search radius can be adaptively changed for different evolution phases given in equation (2.4). However, the iteration factor used resulted to a large step size which limits the exploration capability of the algorithm. In this research, the iteration factor in equation (2.4) was changed to a decreasing function given in equation (2.5) in order to improve the the exploration capability of the standard FFOA and equation (2.4) was modified as:

$$\lambda = \lambda_{\max} \times \exp\left(\log_{10}\left(\frac{\lambda_{\min}}{\lambda_{\max}} \right) \times \left(\frac{Iter_{\max} - Iter}{Iter_{\max}} \right) \right) \tag{3.1}$$

where λ is the search radius in each iteration, $\lambda_{\max}$ is the maximum radius, $\lambda_{\min}$ is the minimum radius, $Iter$ is the iteration number and $Iter_{\max}$ is the maximum iteration number.

The iterative function in equation (3.1), ensured that the effect of the fixed radius was reduced by always taking smaller step size as the algorithm moves towards the optimum solution and improved the exploration capability of the standard FFOA.

3.5.2 Linearly decreasing inertial weight

As discussed in section (2.2.3) this was introduced to the Ospheresis stage of the FFOA in order to have an efficient balance between exploration and exploitation that is the decreasing iteration function was substituted in equation (2.4) and used to determine the contribution of the previous fruit fly position to the fruit fly position in the current time step. The MATLAB code description of the modified algorithm is given in Appendix A.

Based on the MATLAB code in Appendix A, the flow chart implementation is presented in Fig 3.1:

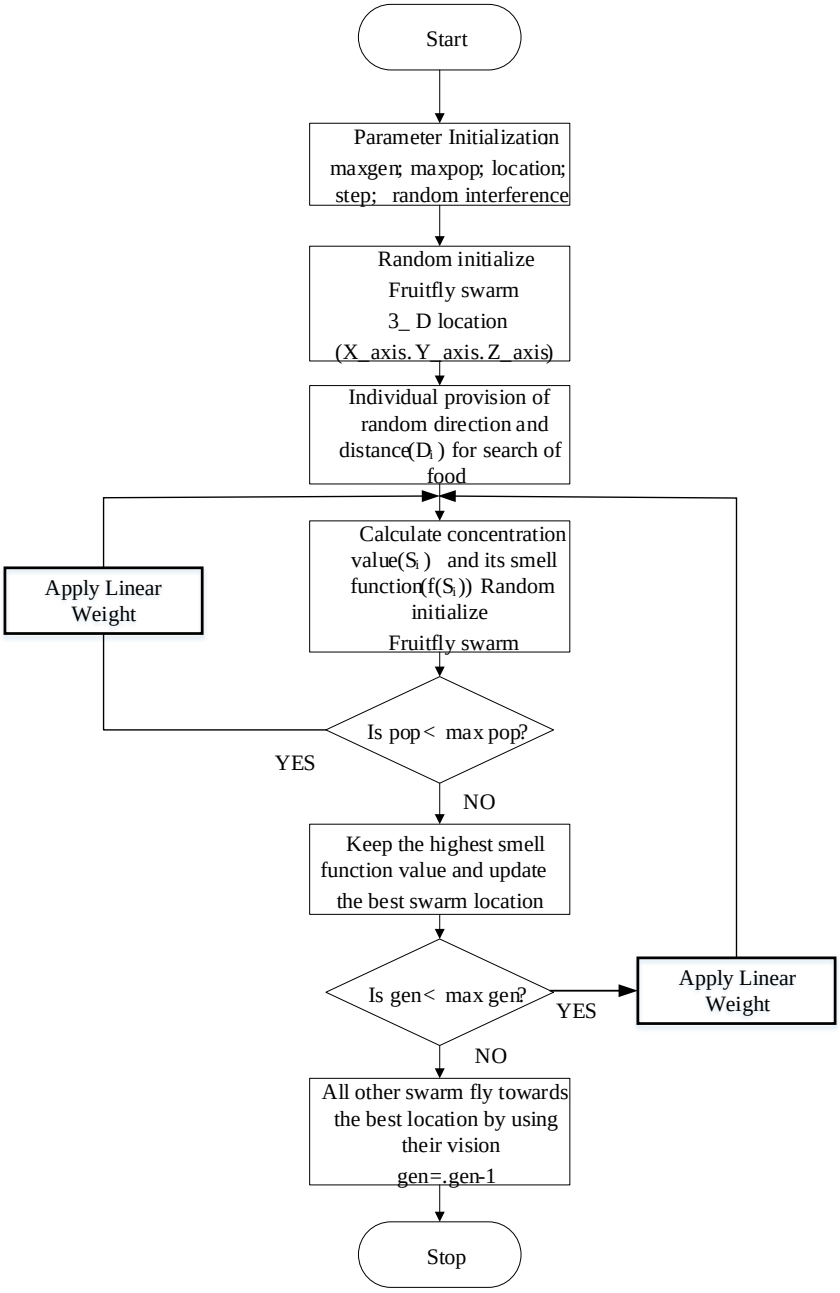


Fig 3.1: Modified FFOA Flowchart

Fig 3.1 shows the flowchart for the modified FFOA. From Fig 3.1, it can be observed that the contribution of the previous fruit fly position to the fruit fly position in the current time step, reduced the unbalanced exploitation and exploration behaviour of the standard FFOA given in Fig. 2.2.

3.6 Performance Evaluation

The performance of the standard and modified FFOA was evaluated using performance metrics listed in subsection 2.2.4

3.7 Comparison of the Results obtained from the Developed Modified FFOA with the Results of the Standard FFOA

The results obtained from the developed modified algorithm were compared with that of the replicated FFOA. The MATLAB code for the comparison is found in Appendix C.

3.8 Determination of Weighting Matrices Using mFFOA

The mFFOA developed in this research is used for the determination of optimized values of the weighting matrices (Q and R) of LQR controller. The most important points in Q and R matrices are the constraints on them. As mentioned in subsection 2.2.5, the Q and R matrices should be positive and semi-definite and positive and definite respectively. However, these constraints cannot be achieved easily. A simple relationship between matrix elements and their positive definiteness is indescribable. It is known that for any real symmetric matrix N is positive definite if there exists a real nonsingular matrix M, $N = M^T M$ is nonnegative, i.e., it can be said that matrix N is positive definite (Ayres 1962). Thus, in this research, mFFOA is used to first find the optimized values of L and K, then Q and R matrices are calculated as:

$$Q = L^T L \tag{3.2}$$

$$R = K^T K \tag{3.3}$$

Equations (3.2) and (3.3) is used to make Q and R positive semi-definite matrices. (Wang *et al*., 2014). This method was used in this research to make initial responses and to code the responses of the problem. The matrices $\boldsymbol{J}$ and K which satisfies equations (3.2) and (3.3) are used instead of using Q and R matrices as unknown variables.

Then the corresponding cost function given in equation (2.17) becomes

$$J = \int_0^{\infty} \left(x_1 q_1 x_1^T + x_2 q_2 x_2^T + x_3 q_3 x_3^T + u\ ru^T \right) dt \tag{3.4}$$

In this case, *n* represent the size of the matrix, which in this work is the matrix of pitch control (*n*=3), J represent the cost function, *q* and *r* are weighting matrices used to determine the relative importance of the state variables and control inputs respectively. $x^T_{1.....n}$ and u^T are the transpose matrices of $x_{1.....n}$ and *u* (state variables and input function) respectively.

The MATLAB code for the determination of the weighting matrices of LQR using mFFOA is given in Appendix A:

Based on the MATLAB code in Appendix A, the flow chart implementation is presented in Fig. 3.2.

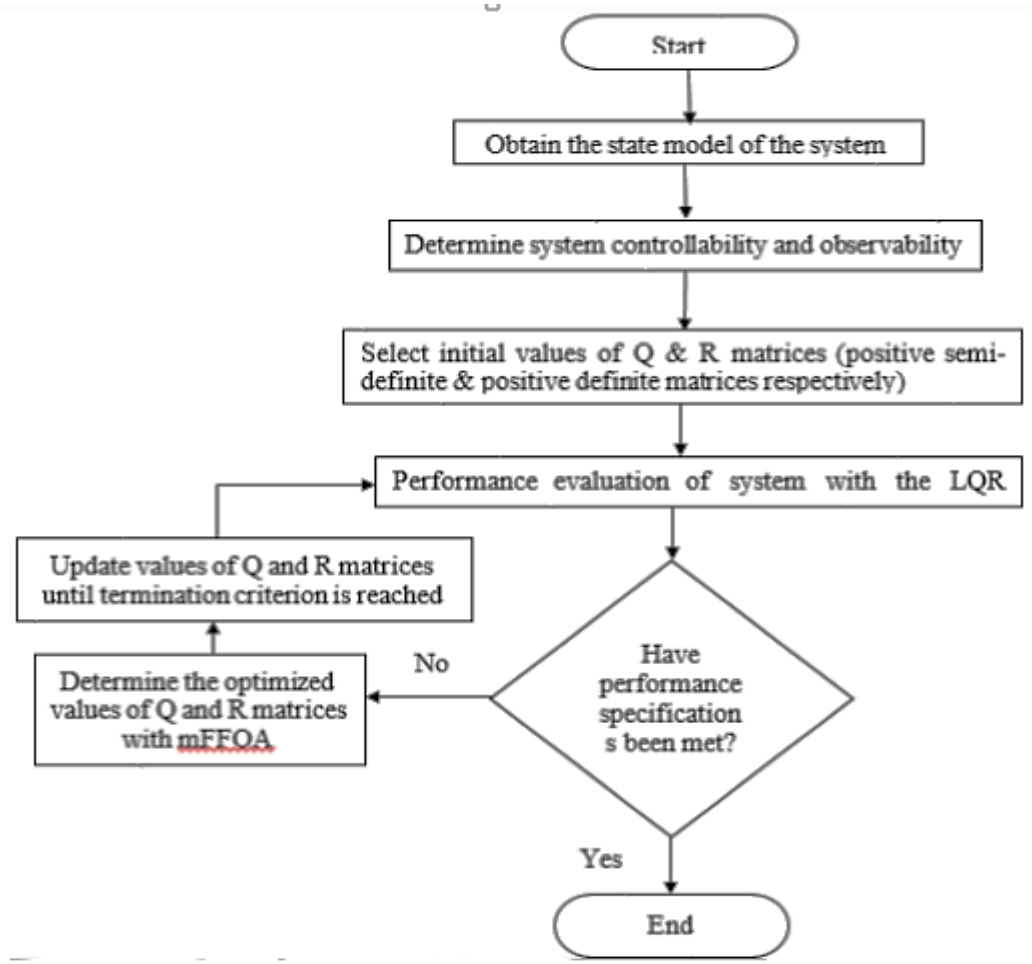


Fig 3.2: Flowchart for the Determination of Optimized Q and R using mFFOA

Fig 3.2 shows the flowchart for the determination of optimized weighting matrices (Q & R) of the LQR using mFFOA. At the initial stage, the state model of the pitch control system is inputted and control test was performed to establish the controllability and observability of the system. Then, an initial value is randomly assigned to the weighting matrices (satisfying constraints) and the cost function value is evaluated. If the optimum cost is obtained, the values of Q and R which produce that cost is selected as the optimized values of the weighting matrices. Otherwise the values of Q and R are adjusted using the mFFOA or the FFOA depending on the algorithm to be run. Then the value Q and R are

updated and the cost function is evaluated again. This procedure is repeated until the optimum cost is obtained.

3.9 Pitch Control System Stabilization

The linearized state space model adopted from Jisha & Aswin (2015)for the PCS of an aircraft was obtained by linearizing (using small-signal disturbance theory) equations (2.26) to (2.28), and substituting the data in Table 3.2 is given as follows (Jisha & Aswin, 2015):

$$\begin{bmatrix} \Delta\dot{\alpha} \\ \Delta\dot{q} \\ \Delta\dot{\theta} \end{bmatrix} = \begin{bmatrix} -2.02 & 1 & 0 \\ -6.9868 & -2.9476 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta q \\ \Delta\theta \end{bmatrix} + \begin{bmatrix} 0.16 \\ 11.7304 \\ 0 \end{bmatrix} [\Delta\delta e] \tag{3.5}$$

$$y = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \Delta\alpha \\ \Delta q \\ \Delta\theta \end{bmatrix} + [0][\Delta\delta e] \tag{3.6}$$

Equations (3.5) and (3.6) show the state space model of the PCS which corresponds to the standard state space model representation described in equations (2.18) and (2.19) respectively.

Where,

$$\dot{x} = \begin{bmatrix} \Delta\dot{\alpha} \\ \Delta\dot{q} \\ \Delta\dot{\theta} \end{bmatrix} \tag{3.7}$$

$$A = \begin{bmatrix} -2.02 & 1 & 0 \\ -6.9868 & -2.9476 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.8}$$

$$x = \begin{bmatrix} \Delta\alpha \\ \Delta q \\ \Delta\theta \end{bmatrix} \tag{3.9}$$

$$B = \begin{bmatrix} 0.16 \\ 11.7304 \\ 0 \end{bmatrix} \tag{3.10}$$

$$u = \begin{bmatrix} \Delta\delta e \end{bmatrix} \tag{3.11}$$

Equations (3.8) and (3.10) are the state and input matrices respectively of the PCS model where α, q, θ and δe are angle of attack, pitch rate, pitch angle and elevator deflection angle.

Furthermore, a few of design specification have to be set to investigate the performance of the control strategies using both mFFOA and FFOA algorithms to determine the Q and R of the LQR which was applied to the PCS. In this work, two considerations have to be met which are settling time specification of PCS less than 5 second and time to solution less than 150 second so as to achieve fast control.

Control test was performed to determine the controllability and observability of the system (PCS), before the mFFOA is used to obtain the Q and R weighting matrices. The MATLAB command of the control test code is given as follows:

```
% Determine Controllability and Observability of PCS
OBV=obsv(A,C);
CTR=ctrb(A,B);
R_OBV=rank(OBV)
R_CTR=rank(CTR)
if rank(CTR)==size(A)
    'System is Controllable'
else
    'System is NOT Controllable'
end
if rank(OBV)==size(A)
    'System is Observable'
else
    'System is NOT observable'
end
```

A snippet of the output from the above program is given as in Fig 3.3:

```
R_OBV =

     3


R_CTR =

     3


ans =

System is Controllable


ans =

System is Observable
```

Fig 3.3: A Snippet of the Output for Controllability and Observability Test Program

From Fig 3.3 it is evident that the state controllability and observability of the system are in full rank. Hence the closed loop poles can then be initialized anywhere in the left-hand side of the complex s-plane. This assertion is necessary in order to establish that the model can be stabilized before designing the controller. First, a program is written to determine the initial positions of the eigenvalues as follows:

```
function [Q R K P E eigenvalue kk]=lqrQR(A,B)
[a1,a2]=size(A);
[b1,b2]=size(B);
eigenvalue=eye(a1);
for i=1:a1;
    realeigen=-1*rand;

imageigen=randi(5,1)*rand*((min(min(A))+max(max(A))*(rand*ma
x(max(A))*100)^0.125)*max(max(B)))^0.5;
    eigenvalue(i,i)=realeigen+1i*(-1)^i*imageigen;
end
```

where A and B are the state and input matrices obtained from the linearized LQR model (Jisha & Aswin, 2015).

To determine the appropriate gain that satisfies the locations of the poles, equation (2.22) was used and mFFOA is randomly used to determine the optimized values of Q and R which satisfies equations (3.2) and (3.3) respectively. The complete program can be found in Appendix A. The fitness function is called from program in Appendix A (mFFOA) using the following code:

```
if testfunction ==0
    A=input('Please Enter The A matrix:\n Such That:\n
dx/dt=Ax(t)+Bu(t).\n\nA = ');
    clc
    B=input('Please Enter The B matrix:\n Such That:\n
dx/dt=Ax(t)+Bu(t).\n\nB = ');
    clc
    t=0;
    while t<QR_Iteration
    [Q,R]=lqrQR(A,B)
    t=t+1;
    end
```

The stability of the system using the optimized values of the mFFOA weighting matrices Q and R was confirmed using the pole zero map plot (pzmap) in MATLAB as shown in Fig3.3.

```
[KmFOA PmFOA EmFOA]=lqr(A,B,Q_mFOA,R_mFOA)%Q and R used here
are obtained from mFOA
sys_LQmFOA =ss(A-B*KmFOA ,B,C,D);
pzmap(sys_LQmFOA)
sgrid
```

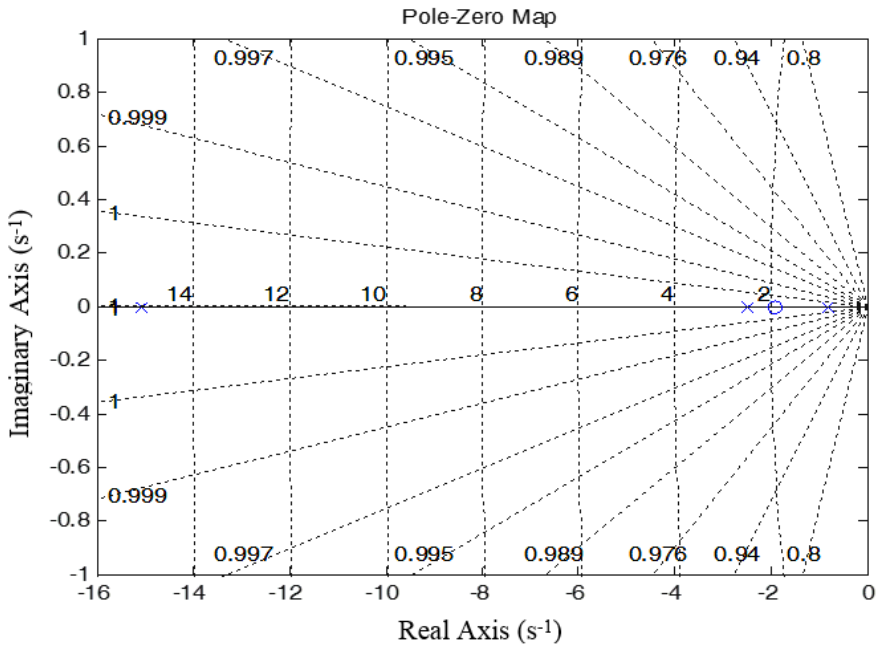


Fig 3.4 Pole-Zero Map of the PCS

Fig 3.4 shows the locations of the eigenvalues in the complex s-plane. It is clear that all the eigenvalues (marked 'x') are on the negative half of the complex s-plane, thereby, indicating stability.

Step-by-step approach for determining the Q and R weighting matrices are as follows.

1. Initialize all the parameters of the mFFOA
2. Define the problem by specifying A, B and initialize the pole location
3. Provide an initial guess for L and K.
4. Calculate Q and R from equations (3.2) and (3.3) respectively using mFFOA
5. Repeat step 4 until the best value of Q and R is obtained
6. Solve the Riccati equation (2.24) and determine the gain matrix K equation (2.23).

The block diagram of the PCS system with the LQR controller whose weighting matrices are obtained using the mFFOA is as shown in Fig. 3.5.

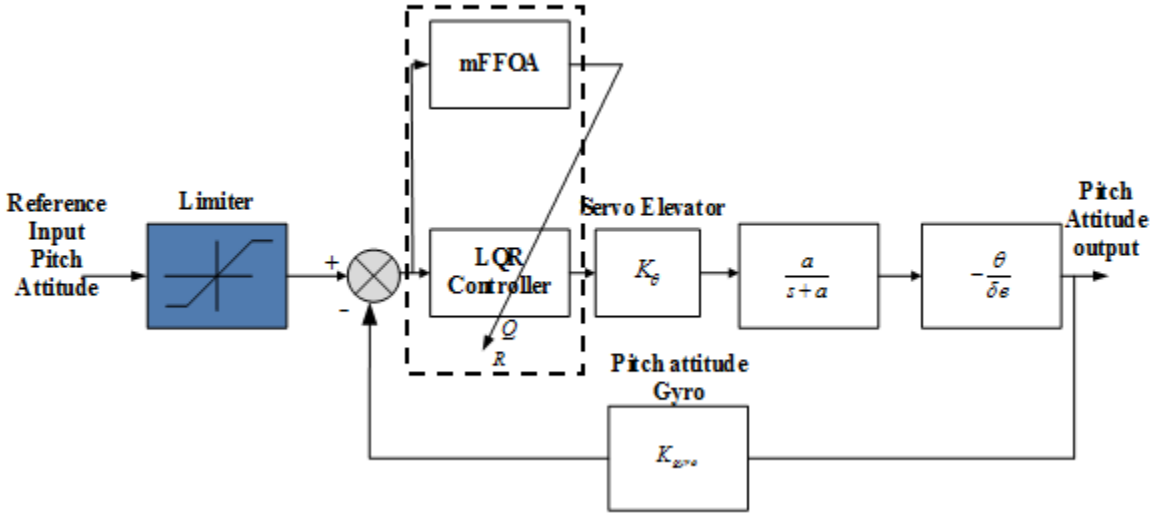


Fig 3.5: Block Diagram of the PCS with the mFFOA-Based LQR

Fig 3.4 shows the implementation of the modified FFOA based LQR optimized weighting matrices (Q & R) on the PCS. The optimized values of Q and R, are determined using the smell and vision intelligent behaviour of fruit fly towards fruits. The MATLAB code for the step response of the PCS with the mFFOA-Based LQR can be found in Appendix C.

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 Introduction

In this chapter, the performances of the modified FFOA are evaluated using the optimization test functions discussed in subsection 2.2.4. Two dimensional (2D) plots were presented for the understanding of all the test function. The performances of the proposed algorithm was compared with the performance of the replicated standard FFOA for validation. The LQR weighting matrices determined using mFFOA and the responses of the PCS were also presented.

4.2 Performance evaluation of mFFOA on the Optimization Test Function

First the standard Fruit Fly Optimization Algorithm was replicated. The performance of the replicated FFOA was evaluated using ten optimization test functions selected for the purpose of this research work, the MATLAB code used for the evaluation is found in Appendix A and the result is compared with the result obtained using the mFFOA found in Appendix B. The results are shown in Table 4.1

Table 4.1: Results obtained for mFFOA, FFOA and Global for the Ten Optimization Test Functions.

SN	Test Function	FFOA	mFFOA	Global
1	Ackley	0.0034	**0.0003**	0.0000
2	Alpine	0.0001	0.0001	0.0000
3	Eggcrate	0.0000	0.0000	0.0000
4	Grienwank	1.0000	1.0000	0.0000
5	Pathologic	0.0000	0.0000	0.0000
6	Rastrigin	0.0001	0.0001	0.0000
7	Rosenbrock	0.0000	0.0000	0.0000
8	Schaffer	0.0000	0.0000	0.0000
9	Sphere	0.0000	0.0000	0.0000
10	Whitely	0.4600	**0.0000**	0.0000

It is clear from Table 4.1, that, mFFOA outperformed the FFOA on ackley test function and Whitely (as indicated in bold) indicating a 20% improvement. Whereas, both algorithm obtained the same solution which is equal or approximate to the global value in the (7) test functions indicating 70% of the test functions. The superiority of mFFOA over FFOA is expected since a decreasing inertial weight was introduced which provided a much better balance between exploration and exploitation thereby reducing the ability of the algorithm being trapped into some local optimal.

In order to show the optimization process of both the algorithms the minimization plot (fitness/smell function plot) was generated usingMATLAB commands. Figs. 4.1 to 4.10 show the superimposed plots for the mFFOA and FFOA on the minimization plots for the test functions used for the purpose of this research.

4.2.1 Ackley function

The superimposed minimization plot for Ackley function using the mFFOA and the standard FFOA is as shown in Fig 4.1.

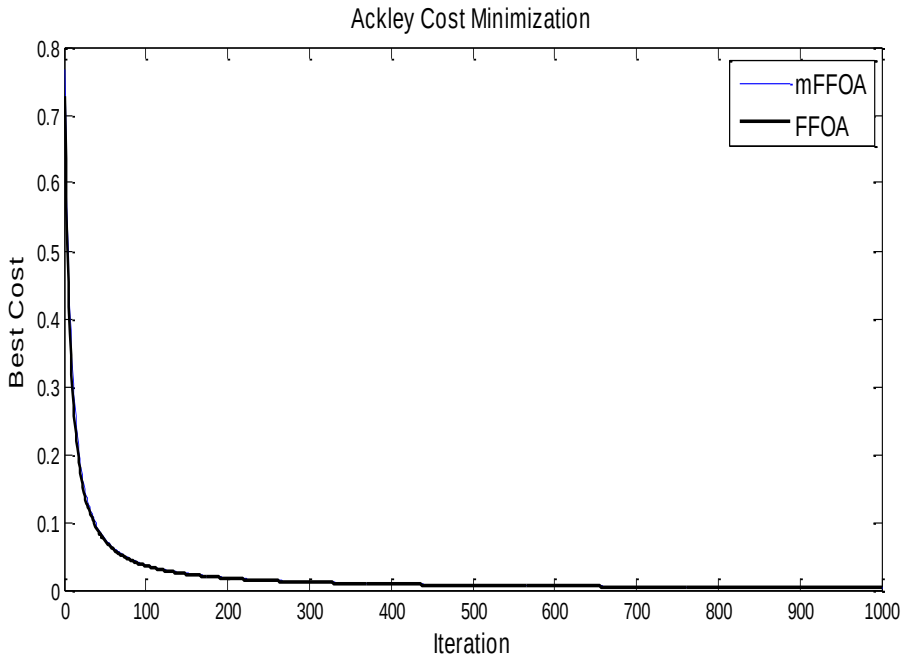


Fig 4.1: mFFOA and FFOA Ackley Cost Minimization Process

From Fig 4.1, it is clear that both mFFOA and FFOA minimize the Ackley function efficiently to a value of 0.0003 and 0.0034 respectively which is the optimum value both can obtain for the function. Thereafter, the result remains constant throughout the optimization process.

4.2.2 Alpine function

The superimposed minimization plot for Alpine function using the mFFOA and the standard FFOA is as shown in Fig 4.2.

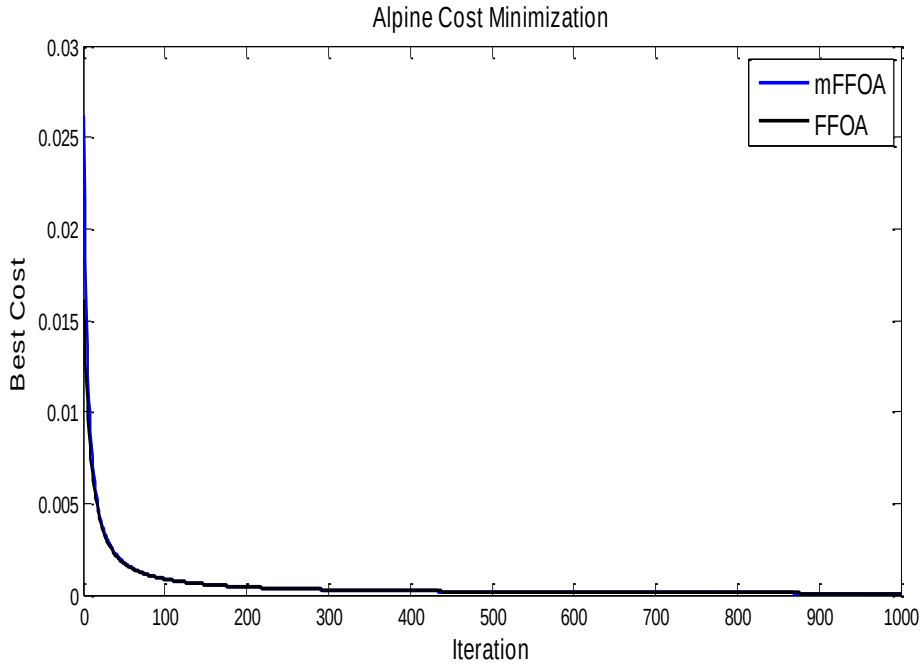


Fig 4.2: mFFOA and FFOA Alpine Cost Minimization Process

From Fig 4.2, it is clear that both mFFOA and FFOA minimize the Alpine function efficiently to a value of 0.0001 and 0.0001 respectively. Thereafter, the result remain constants throughout the optimization process.

4.2.3 Eggcrate function

The superimposed minimization plot for Eggcrate function using the mFFOA and the standard FFOA is as shown in Fig 4.3.

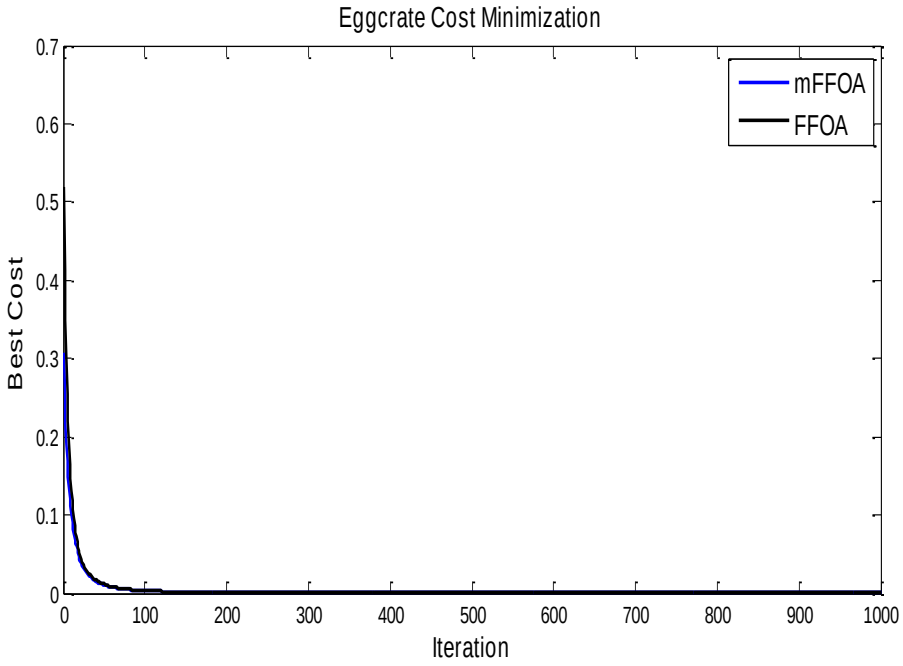


Fig 4.3: mFFOA and FFOA Eggcrate Cost Minimization Process

From Fig 4.3, it is clear that both mFFOA and FFOA minimize the Eggcrate function efficiently to a value of 0 and 0 respectively. This remains constant throughout the optimization process.

4.2.4 Grienwank function

The superimposed minimization plot for Grienwank function using the mFFOA and the standard FFOA is as shown in Fig 4.4

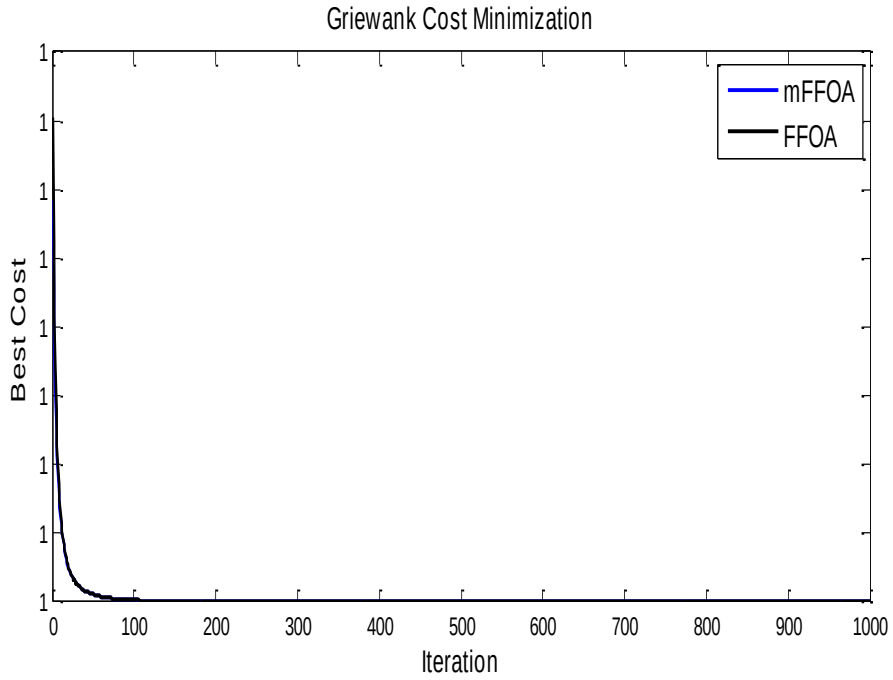


Fig 4.4: mFFOA and FFOA Grienwank Cost Minimization Process

From Fig 4.4, it is clear that both mFFOA and FFOA for the Grienwangk function got trapped into some local optima to a value of 1, whereas the global optimal solution is a value of 0.

4.2.5 Pathologic function

The superimposed minimization plot for Pathologic function using the mFFOA and the standard FFOA is as shown in Fig 4.5.

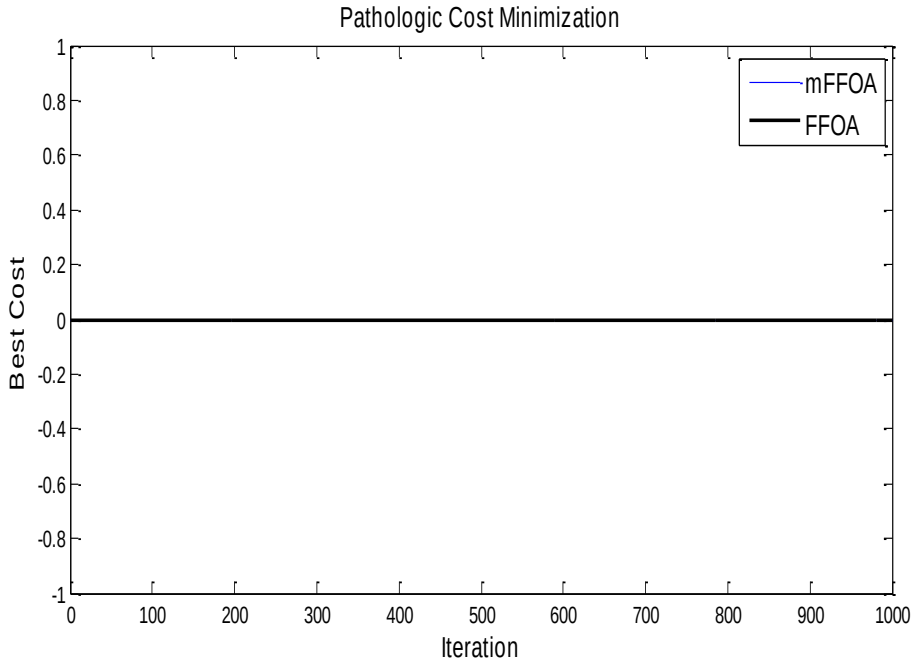


Fig 4.5: mFFOA and FFOA Pathologic Cost Minimization Process

From Fig 4.5, it is clear that both mFFOA and FFOA minimize the Pathologic function efficiently to a value of 0 and 0 respectively. This remains constant throughout the optimization process.

4.2.6 Rastrigin function

The superimposed minimization plot forRastrigin function using the mFFOA and the standard FFOA is as shown in Fig 4.6.

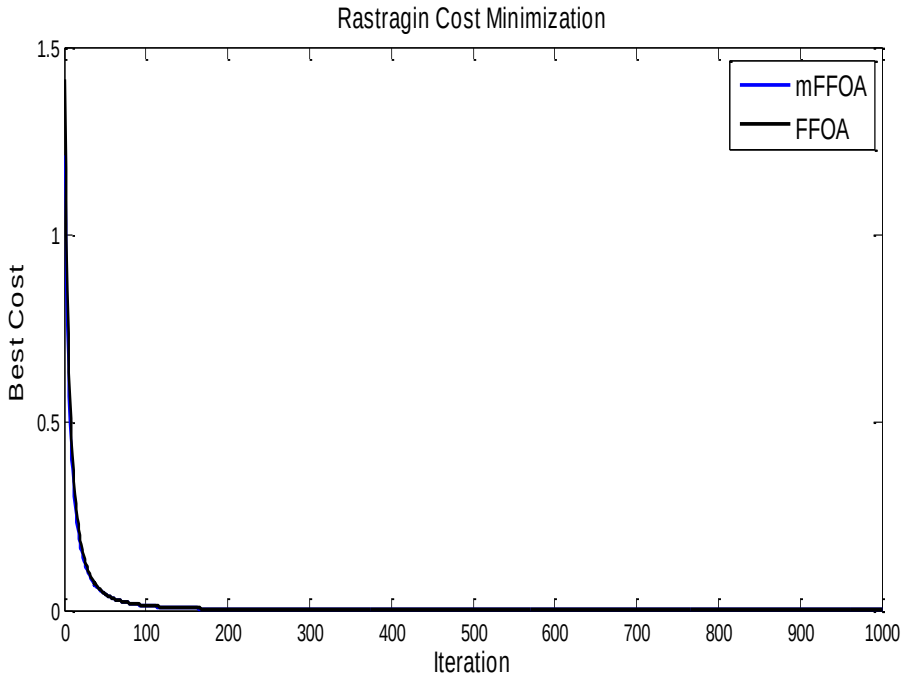


Fig 4.6: mFFOA and FFOA Rastrigin Cost Minimization Process

From Fig 4.6, it is clear that both mFFOA and FFOA minimize theRastrigin function efficiently to a value of 0.0001 and 0.0001 respectively. Thereafter, the result remains constant throughout the optimization process.

4.2.7 Rosenbrock function

The superimposed minimization plot for Rosenbrock function using the mFFOA and the standard FFOA is as shown in Fig 4.7.

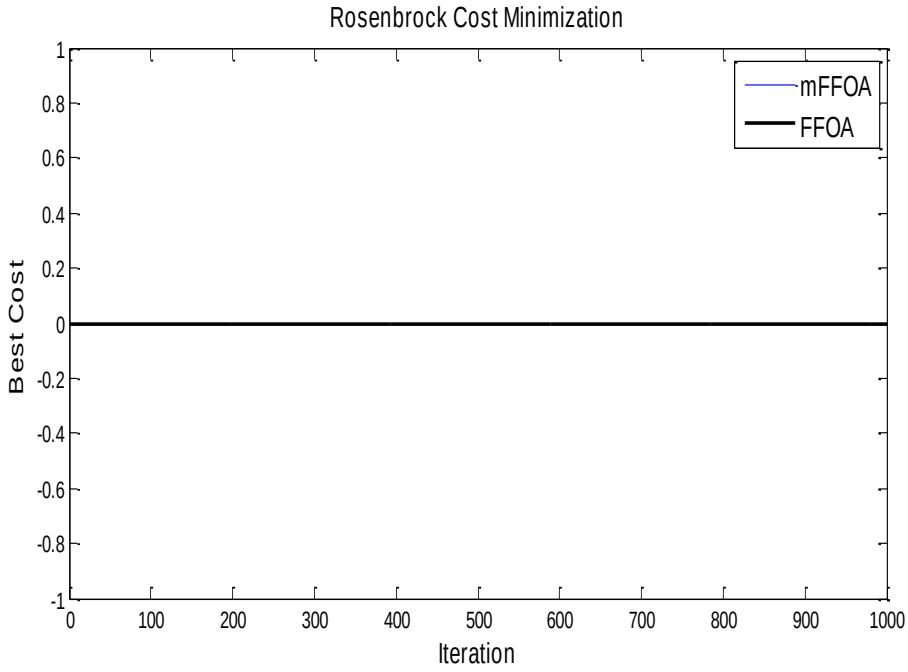


Fig 4.7: mFFOA and FFOA Rosenbrock Cost Minimization Process

From Fig 4.7, it is clear that both mFFOA and FFOA minimize the Pathologic function efficiently to a value of 0 and 0 respectively. This remains constant throughout the optimization process.

4.2.8 Schaffer function

The superimposed minimization plot for Schaffer functionusing the mFFOA and the standard FFOA is as shown in Fig 4.8.

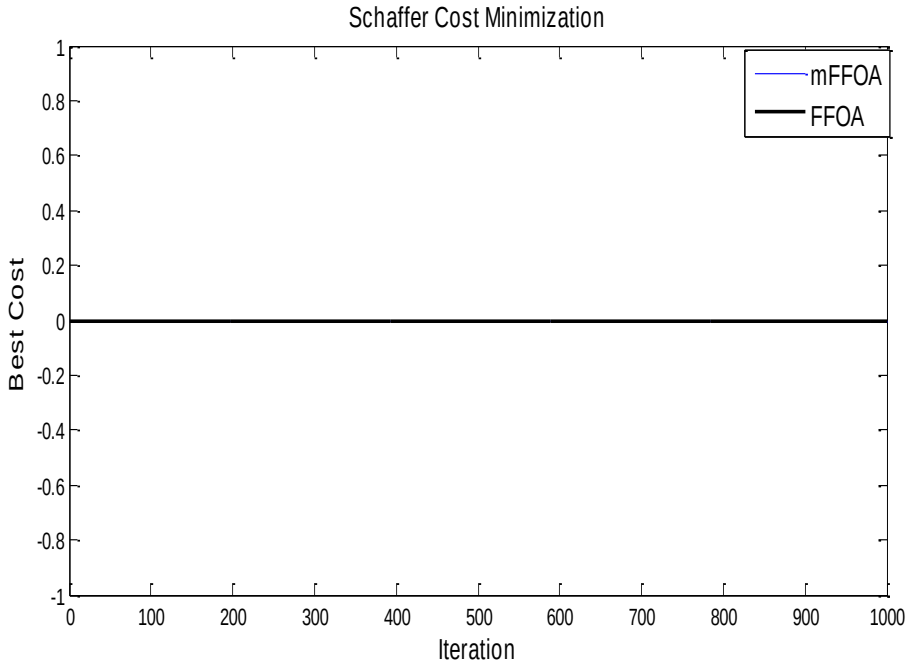


Fig 4.8: mFFOA and FFOA Schaffer Cost Minimization Process

From Fig 4.8, it is clear that both mFFOA and FFOA minimize the Schaffer function efficiently to a value of 0 and 0 respectively. This remains constant throughout the optimization process.

4.2.9 Sphere function

The superimposed minimization plot for Sphere functionusing the mFFOA and the standard FFOA is as shown in Fig 4.9.

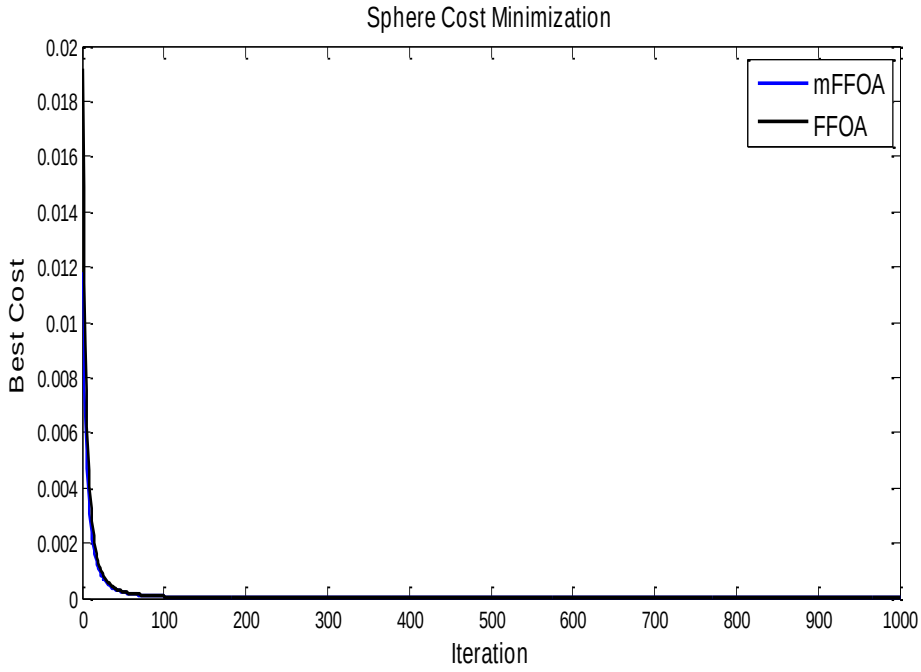


Fig 4.9: mFFOA and FFOA Sphere Cost Minimization Process

From Fig 4.9, it is clear that both mFFOA and FFOA minimize the Sphere function efficiently to a value of 0 and 0 respectively. Thereafter, the result remains constant throughout the optimization process.

4.2.10 Whitely function

The superimposed minimization plot for Whitely function using the mFFOA and the standard FFOA is as shown in Fig 4.10.

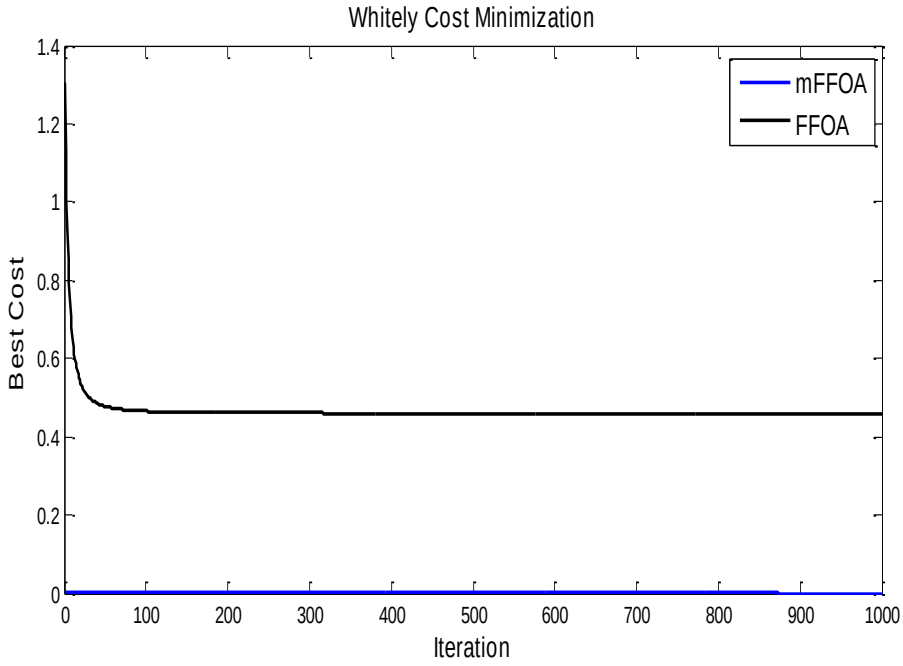


Fig 4.10: mFFOA and FFOA Whitely Cost Minimization Process

From Fig 4.10, it is clear that both mFFOA minimize the Whitely function efficiently to a value of 0 whereas the FFOA got trapped into local optima to a value of 0.46. The results from each of the algorithm remains constant throughout the optimization process.

4.3 Application of the Developed mFFOA Based LQR Controller for Optimal Determination of Controller Parameters

In order to determine the efficiency of the mFFOA in determining the weighting matrices of the LQR controller, both algorithms were ran ten times due to the stochasticity of the algorithm. The optimized Q and R obtained for the mFFOA and FFOA were tabulated as in Table 4.2.

Table 4.2: Comparison of the Q and R Matrices for mFFOA and FFOA

Runs:	mFFOA Q	mFFOA R	FFOA Q	FFOA R
1	$\begin{bmatrix} 5.2741 & 0.0467 & 0.0196 \\ 0.0467 & 5.1876 & 0 \\ 0.0196 & 0 & 5.2227 \end{bmatrix}$	0.4181	$\begin{bmatrix} 6.8266 & 0 & 0.0164 \\ 0 & 6.8684 & 0 \\ 0.0164 & 0 & 6.8943 \end{bmatrix}$	0.1308
2	$\begin{bmatrix} 13.5825 & 0 & 0 \\ 0 & 13.5743 & 0.0319 \\ 0 & 0.0319 & 13.6534 \end{bmatrix}$	0.5656	$\begin{bmatrix} 6.1389 & 0.0120 & 0.0446 \\ 0.0120 & 6.1688 & 0.0446 \\ 0.0446 & 0.0446 & 6.1610 \end{bmatrix}$	0.2504
3	$\begin{bmatrix} 10.0585 & 0 & 0.04249 \\ 0 & 10.0628 & 0.04249 \\ 0.04249 & 0.04249 & 10.0976 \end{bmatrix}$	0.2273	$\begin{bmatrix} 9.5549 & 0.06310 & 0.0307 \\ 0.0631 & 9.5380 & 0 \\ 0.0307 & 0 & 9.4375 \end{bmatrix}$	0.2701
4	$\begin{bmatrix} 3.2476 & 0 & 0 \\ 0 & 3.2380 & 0 \\ 0 & 0 & 3.2542 \end{bmatrix}$	0.1615	$\begin{bmatrix} 6.8266 & 0 & 0.0164 \\ 0 & 6.8684 & 0 \\ 0.0164 & 0 & 6.8943 \end{bmatrix}$	0.1308
5	$\begin{bmatrix} 27.7918 & 0.0163 & 0.0588 \\ 0.0163 & 27.6817 & 0.0588 \\ 0.0588 & 0.0588 & 27.8825 \end{bmatrix}$	0.2898	$\begin{bmatrix} 6.1389 & 0.0120 & 0.0446 \\ 0.0120 & 6.1688 & 0.0446 \\ 0.0446 & 0.0446 & 6.1610 \end{bmatrix}$	0.2504
6	$\begin{bmatrix} 27.5803 & 0 & 0.0583 \\ 0 & 27.6875 & 0 \\ 0.0583 & 0 & 27.6257 \end{bmatrix}$	0.2944	$\begin{bmatrix} 9.5549 & 0.0631 & 0.0307 \\ 0.0631 & 9.5380 & 0 \\ 0.0307 & 0 & 9.4375 \end{bmatrix}$	0.2701
7	$\begin{bmatrix} 24.5498 & 0.0233 & 0 \\ 0.0233 & 24.5218 & 0.0120 \\ 0 & 0.0120 & 24.5446 \end{bmatrix}$	0.1978	$\begin{bmatrix} 9.5890 & 0 & 0.0319 \\ 0 & 9.6145 & 0.0319 \\ 0.0319 & 0.0319 & 9.6329 \end{bmatrix}$	0.1484
8	$\begin{bmatrix} 4.5631 & 0 & 0 \\ 0 & 4.5571 & 0.0081 \\ 0 & 0.0081 & 4.5802 \end{bmatrix}$	0.1630	$\begin{bmatrix} 0.9990 & 0 & 0 \\ 0 & 1.0163 & 0 \\ 0 & 0 & 1.0068 \end{bmatrix}$	0.3138
9	$\begin{bmatrix} 16.5461 & 0 & 0.0346 \\ 0 & 16.5077 & 0 \\ 0.0346 & 0 & 16.5237 \end{bmatrix}$	0.2374	$\begin{bmatrix} 0.6832 & 0 & 0.0011 \\ 0 & 0.6796 & 0 \\ 0.0011 & 0 & 0.6777 \end{bmatrix}$	0.2409
10	$\begin{bmatrix} 21.6173 & 0.0277 & 0 \\ 0.0277 & 21.5658 & 0 \\ 0 & 0 & 21.5299 \end{bmatrix}$	0.2953	$\begin{bmatrix} 17.4394 & 0 & 0 \\ 0 & 17.3370 & 0.0235 \\ 0 & 0.0235 & 17.3230 \end{bmatrix}$	0.4884

Table 4.2 shows the values of the optimized weighting matrices (Q and R) of the LQR controller using the mFFOA and FFOA. From the result it is clear that the mFFOA gives better optimized values of Q's and R'smatrices which was able to stabilize the PCS system with a settling time less than that of the PCS design specification in seven (7) out of the ten (10) runs. The settling time (ts) was obtained from the step response analysis of the LQR, which stabilizes the PCS using the MATLAB code in Appendix C by changing the Q and R values from first to the tenth run. The time taken to determine the weighting matrices were calculated as the time-to-solution/elapsed time (t_2s) using MATLAB command as in Appendix A & B for the FFOA and FFOA respectively. The time to solution (t_2s) and settling time (ts) are tabulated in Table 4.3

Table 4.3 The Settling Time and Time to Solution for mFFOA and FFOA

Runs	**mFFOA**		**FFOA**	
	ts (s)	t_2s(s)	ts (s)	t_2s(s)
1	4.5227	**126.6648**	**4.4551**	126.9090
2	**4.4695**	**125.3113**	4.4920	126.30
3	**4.4633**	**125.3536**	4.5112	126.5801
4	4.4786	**126.0660**	**4.4551**	153.7282
5	**4.4456**	**126.9538**	4.4920	147.7347
6	**4.4616**	**125.9995**	4.5112	139.1311
7	**4.4543**	**126.3670**	4.4586	159.7811
8	**4.4672**	**126.1129**	4.6676	142.1975
9	**4.4611**	**127.1459**	4.6916	134.4642
10	**4.4668**	143.333	4.4764	**140.7819**

From Table 4.3 it is clear that the mFFOA has the least values (in bold) of settling time which is less than 5s in eight (8) runs, while FFOA obtained a value below that of the mFFOA in two test runs. Also in terms of time-to-solution the mFFOA obtained the optimized Q and R matrices within the least time (in bold) which is less than 150s in all the test run, except in the tenth run where the FFOA have a value below that of the mFFOA. This indicates the ability of the mFFOA to achieve fast control, since with any of the best values the PCS can achieve stability.

In order to test the efficiency of these optimized values on the PCS, the best value obtained using both algorithms is selected as follows:

For mFFOA and FFOA the best Q and R value that met the constraint (positive and semidefinite and positive definite matrices respectively) most were selected and given as:

$$Q_{mFFOA} = \begin{bmatrix} 27.7918 & 0.0163 & 0.0588 \\ 0.0163 & 27.6817 & 0.0588 \\ 0.0588 & 0.0588 & 27.8825 \end{bmatrix}$$

$$R_{mFFOA} = 0.2898$$

$$Q_{FFOA} = \begin{bmatrix} 17.4394 & 0 & 0 \\ 0 & 17.3370 & 0.0235 \\ 0 & 0.0235 & 17.3230 \end{bmatrix}$$

$$R_{FFOA} = 0.4884$$

The optimized Q & R values are obtained within a time of 126.9538s when compared with the 140.7819s taken using the FFOA approach. The proposed method reduced the time taken by the FFOA by 13.8281s. These values are used to determine the gain (*K*) matrix as in equation (2.21) and the eigen values for the mFFOA and that of theFFOA method and are given as follows.

$$K_{mFFOA} = \begin{bmatrix} 1.1404 & 9.5947 & 9.8054 \end{bmatrix}$$

$$E_{mFFOA} = \begin{matrix} -114.5877 \\ -2.2550 \\ -0.8568 \end{matrix}$$

$$K_{FFOA} = \begin{bmatrix} 0.4714 & 5.7896 & 5.9556 \end{bmatrix}$$

$$E_{mFFOA} = \begin{matrix} -69.8457 \\ -2.2599 \\ -0.8519 \end{matrix}$$

It is shown above that the mFFOA obtained better result than the FFOA for the gain, and eigenvalues. The result is obtained using the MATLAB code in AppendixC.

In order to confirm the optimality of Q and R weighting matrices obtained from the modified fruit fly optimization algorithm (mFFOA), the responses of the PCS were plotted and superimposed with the best optimized values obtained using standard FFOA method. These response is presented in Fig 4.11. Notice that the blue lines show the responses

obtained for the LQR (mFFOA) and the redlines show the responses obtained for the LQR (FFOA).

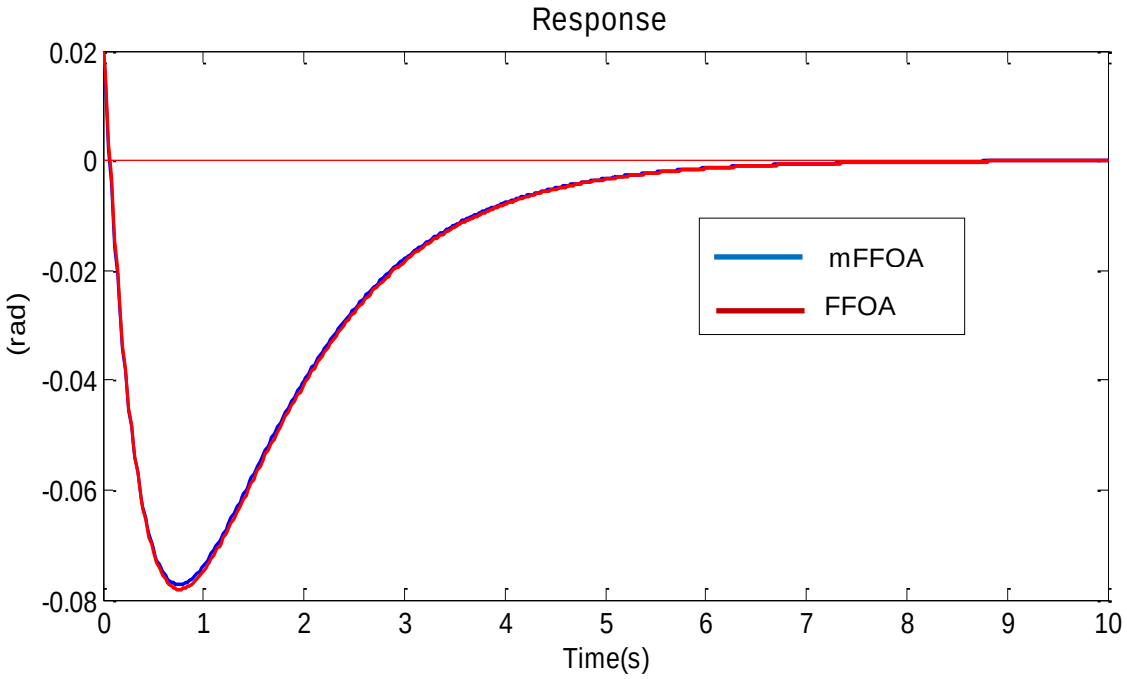


Fig 4.11: Step Response of PCS

As expected, the settling time (time taken for the PCS to stabilize) using LQR (mFFOA)andLQR (FFOA) nearly converged showing the convergence of the solution space using either of the approach. The advantage of the LQR mFFOA over the FFOA approach is that it saves cost in terms of time taken for the PCS to stabilize using LQR based mFFOA which is 4.4456s when compared with that taken using FFOA which is 4.4764s.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Summary

This research has proposed a modified fruit fly optimization algorithm using decreasing inertial weight and adaptive search radius with decreasing iteration function, which was aimed at providing an efficient balance between exploration and exploitationand improved the rate at which the standard algorithm moves towards the global optimal. The performance of the algorithm was confirmed using ten optimization test function and the following findings were reported.

1. The introduction of decreasing inertial weight (w) improved the poor exploration problem of the standard FFOA.
2. The proposed algorithm determined efficiently the optimal values of linear quadratic regulator controllers weighting matrices (Q and R) with the least time when compared with the standard FFOA. This has helped in reducing cost in terms of time.
3. The LQR controller stabilized the Pitch control system of an aircraft efficiently.

5.2 Conclusion

Fruit Fly Optimization Algorithm (FFOA) is a class of evolutionary algorithmwhich was model based on the food finding behaviour of the fruitfly. In order to overcome the common problems (local optima and fixed search radius) associated with FFOA, a

modified version called the mFFOA has been developed, using, decreasinginertial weight and adaptive search radius with decreasing iteration function in MATLAB R2015a and the performance of the developed algorithms was evaluated using ten mathematical optimization test functions with diverse characteristics **(**Ackley, Alpine, Eggcrate, Griewank, Pathologic, Rastrigrin, Rosenbrock, Schaffer, Sphere, and Whitley**)**. The result obtained shows that mFFOA performed better than the standard FFOA with the Ackley and Whitely test functions there by indicating 20% improvement over the standard FFOA. This shows that the mFFOA has a better search radius and higher ability of escaping local optimal when compared with the standard FFOA. Also the proposed algorithm reduced the time taken by the FFOA to obtain the optimized Q and R matrices by 13.8281s. From the PCS step response analysis the result obtained demonstrated that the proposed algorithm can certainly stabilized the pitch control system within a settling time of 4.4456s. This is an indication of the validity of the mFFOA.

5.3 Significant Contributions

A lot of research works have been done on improving the local optimal of Fruit Fly Optimization Algorithm. Many researches have also been conducted on different area of its application. The significant contributions of this research work are as follows:

1. Development of a modified FFOA (mFFOA) with a decreasing inertial weight with particular emphasis on improving exploration capability of the standard FFOA.
2. The optimized weighting matrices (Q and R) of the LQR based aircraft based PCSusing the mFFOA based approach were obtained in 126.954s as against

140.7819s when using the standard FFOA. This represents an improvement of 9.822% in time-to-solution.

3. The Q and R values of the LQR obtained using the mFFOA based approach were able to settle the PCS in 4.4456s as compared to 4.4764s when the values from the standard FFOA based approach were used representing an improvement of 0.69%.

5.4 Recommendation for Further Work

The following possible area is recommended for consideration for future research:

Application of the developed model for optimal determination of the weighting matrices can be extended to other areas in power system control, stability analysis and motor speed control.

REFERENCES

Abdulla, A. I., Ahmed, J. M., & Attya, S. M. (2013). Genetic Algorithm (GA) Based Optimal Feedback Control Weighting Materices Computation. *Al-Rafadain Engineering Journal, 21*(5), 25-33.

Akay, B. (2013). A study on particle swarm optimization and artificial bee colony algorithms for multilevel thresholding. *Applied Soft Computing, 13*(6), 3066-3091.

Al-salami, N. M. (2009). Evolutionary algorithm definition. *American J. of Engineering and Applied Sciences, 2*(4), 789-795.

Ata, B., & Coban, R. (2015). Artificial Bee Colony Algorithm Based Linear Quadratic Optimal Controller Design for a Nonlinear Inverted Pendulum. *International Journal of Intelligent Systems and Applications in Engineering,, 3*(1), 1-6.

Ayres, F. Jr. (1962). *Schaum's Outline of Theory and Problems of Matrices*. New York: Schaum, p. 134.

Bäck, T., & Schwefel, H.-P. (1993). An overview of evolutionary algorithms for parameter optimization. *Evolutionary computation, 1*(1), 1-23.

Choubey, N. S. (2014). Fruit fly optimization algorithm for travelling salesperson problem. *International Journal of Computer Applications (0975–8887), 107*(18), 22-27.

Cuong, N. D., Van Lanh, N., & Van Huyen, D. (2015). Design of LQG Controller Using Operational Amplifiers for Motion Control Systems. *Journal of Automation and Control Engineering 3*(2), 300-306.

Er, M., & Oentaryo, R. (2011). Computational Intelligence: Methods and Techniques. *IEEE Computational Intelligence Magazine, 6*(4), 76-78.

Fister Jr, I., Fister, D., & Yang, X.-S. (2013). A hybrid bat algorithm. *arXiv preprint arXiv:1303.6310, 1-7*.

Ghoreishi, S. A., Nekoui, M. A., & Basiri, S. O. (2011). Optimal Design of LQR Weighting Matrices based on Intelligent Optimization Methods. *International Journal of Intelligent Information Processing,, 2* (1), 1-13.

Ghoreishi, S. A., & Nekoui, M. A. (2012). *Optimal weighting matrices design for LQR controller based on genetic algorithm and PSO.* Advanced Materials Research, 433-440(2012), 7546-7553.

Gupta, S., & Tripathi, R. K. (2014). Optimal LQR controller in CSC based STATCOM using GA and PSO. *Archives of Electrical Engineering, 63*(3), 469-487.

Hamidi, J. (2012). Control System Design Using Particle Swarm Optimization (PSO). . *International Journal of Soft Computing and Engineering, 1* (6), 116-119.

Hansen, N. (2006). *Compilation of results on the 2005 CEC benchmark function set*. Computational Laboratory (CoLab) Institute of Computational Science ETH Zurich.

Haupt, R. L., & Haupt, S. E. (2004). *Practical Genetic Algorithms* (2nd ed.): John Wiley & Sons, Inc, ISBN: 978-0-471-45565-3.

Hengyu, L., Jiqing, C., Quanzhen, H., Shaorong, X., & Jun, L. (2014). *An improvement of fruit fly optimization algorithm for solving traveling salesman problems.* Paper presented at the IEEE International Conference on Information and Automation (ICIA), 2014 Hailar, China.

Iscan, H., & Gunduz, M. (2014). Parameter Analysis on Fruit Fly Optimization Algorithm. *Journal of Computer and Communications, 2* ,137-141.

Jamil, M., & Yang, X.-S. (2013). A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150-194.

Jisha, S., & Aswin, R. B. (2015). Pitch Control of Aircraft Using LQR &LQG Control. *International Journal of Advanced Research in Electrical,Electronics and Instrumentation Engineering, 4*(8), 6981-6987.

Johnson, M., & Grimble, M. (1987). Recent trends in linear optimal quadratic multivariable control system design. Paper presented at the IEE *Proceedings D-Control Theory and Applications,* 379(1), 53 - 71.

Ju, J., & Mohamed, S. K. (2007). Pitch Control of an Aircraft with Aggregated Reinforcement Learning Algorithms. Paper presented at the IEE *Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA,*.

Karaboga, D. (2005.). An idea based on honey bee swarm for numerical optimization. Technical Report - TR06, *Erciyes University, Kayseri,Turkey,*.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 3*9(3), 459-471.

Karaboga , D., & Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation, 214*(1), 108-132.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. IEEE *Proceedings of IEEE International Conference on Neural Networks,* 1995, Perth, WA, Australia,.

Li, H., Guo, S., Li, C., & J. Sun. (2013). A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm,. *Knowledge-Based Systems, 37*, 378-387.

Li, X., Tang, K., Omidvar, M. N., Yang, Z., Qin, K., & China, H. (2013). Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. *gene, 7*, 1-23.

Lin, S.-M. (2013). Analysis of service satisfaction in web auction logistics service using a combination of fruit fly optimization algorithm and general regression neural network. *Neural Computing and Applications, 22*(3-4), 783-791.

Mhudtongon, N., Phongcharoenpanich, C., & Kawdungta, S. (2015). Modified fruit fly optimization algorithm for analysis of large antenna array. *International Journal of Antennas and Propagation,* 2015(2), 1-11.

Monfort, M., Liu, A., & Ziebart, B. D. (2015). Intent Prediction and Trajectory Forecasting via Predictive Inverse Linear-Quadratic Regulation. *Association for the Advancement of Artificial Intelligence*, 2015, 3672-3678.

Mua'zu, M. B., Salawudeen, A. T., Sikiru, T. H., Abdu, A. I., & Mohammad, A. (2015). Weighted Artificial Fish Swarm Algorithm with Adaptive Behaviour Based Linear Controller Design for Nonlinear Inverted Pendulum. *Journal of Engineering Research, 20*(1), 1-12.

Nagarkar, M., & Patil, G. V. (2016). Optimization of the linear quadratic regulator (LQR) control quarter car suspension system using genetic algorithm. *Ingeniería Investigación, 36*(1), 23-30.

Ohri, J. (2014). *GA tuned LQR and PID controller for aircraft pitch control.* Paper presented at the IEEE 6th India International Conference on Power Electronics (IICPE), 2014.

Pan, Q.-K., Sang, H.-Y., Duan, J.-H., & Gao, L. (2014). An improved fruit fly optimization algorithm for continuous function optimization problems. *Knowledge-Based Systems, 62*, 69-83.

Pan , W. (2012.). A new fruit fly optimization algorithm: taking the financial distress model as an example *Knowledge-Based Systems, 26*, 69-74.

Pan, W. T. (2011). A New Fruit Fly Optimization Algorithm: Taking the Financial Distress Model as an Example. *Knowledge-Based Systems, 26*, 69-74, doi: 10.1016.

Rahnamayan, S., Tizhoosh, H. R., & Salama, M. M. (2007). A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications, 53*(10), 1605-1614.

Rizk-Allah, M. R. (2016). Hybridization of Fruit Fly Optimization Algorithm and Firefly Algorithm for Solving Nonlinear programming Problems. *International Journal of Swarm Intelligence and Evolutionary Computation, 5*(2), 1-10. doi: 10.4172/2090-4908.1000134.

Salawudeen, A. T., & Mu'azu, B. M. (2015). *Stabilization of Inverted Pendulum System Intelligent using Linear Quadratic Regulator Controller*. Paper presented at the International Joint Conference on Computational Intelligence (IJCCI), Lisbon Portugal.

Schumer, M., & Steiglitz, K. (1968). Adaptive step size random search. *IEEE Transactions on Automatic Control, 13*(3), 270-276.

Shi, Y., & Eberhart, R. C. (1999). *Empirical study of particle swarm optimization.* Paper presented at the IEEE Proceedings of the 1999 Congress on Evolutionary Computation, 1999. CEC 99. Washington, DC, USA.

Stefanescu, R. M., Prioroc, C. L., & Stoica, A. M. (2013). *Weighting matrices determination using pole placement for tracking manuevers*. *U.P.B. Sci. Bull., Series D, 75*(2), 31-40.

Tang, K., Yáo, X., Suganthan, P. N., MacNish, C., Chen, Y.-P., Chen, C.-M., & Yang, Z. (2007a). Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. *Nature Inspired Computation and Applications Laboratory, USTC, China*, 1-18.

Tang, K., Yáo, X., Suganthan, P. N., MacNish, C., Chen, Y.-P., Chen, C.-M., & Yang, Z. (2007b). Benchmark functions for the CEC'2008 special session and competition on large scale global optimization. *Nature Inspired Computation and Applications Laboratory, USTC, China*, 153-177.

Wahid, N., & Hassan, N. (2012). *Self-tuning fuzzy pid controller design for aircraft pitch control.* Paper presented at the 2012 Third International Conference on Intelligent Systems Modelling and Simulation, Kota Kinabalu, Malaysia.

Wahid, N., & Rahmat, M. F. a. (2010). *Pitch control system using LQR and Fuzzy Logic Controller*. Paper presented at the IEEE Symposium on Industrial Electronics & Applications (ISIEA), 2010, Penang, Malaysia.

Wang, H., Liao, L., Wang, D., Wen, S., & Deng, M. (2014). Improved Artificial Bee Colony Algorithm and Its Application in LQR Controller Optimization. *Mathematical Problems in Engineering, 2014*, 1-8. doi: 10.1155/2014/695637

Wang, Y., & Li, L. (2015). An improved intelligent algorithm based on the group search algorithm and the artificial fish swarm algorithm. *Int. J. Optim. Civil Eng, 5*(1), 37-52.

Wongsathan, C., & Sirima, C. (2009). *Application of GA to design LQR controller for an Inverted Pendulum System.* Paper presented at the IEEE International Conference on Robotics and Biomimetics, 2008. ROBIO 200, Bangkok, Thailand..

Wu, L., Xiao, W., Zhang, L., Liu, Q., & Wang, J. (2016). An improved fruit fly optimization algorithm based on selecting evolutionary direction intelligently. *International Journal of Computational Intelligence Systems, 9*(1), 80-90.

Xu, F. Q., & Tao, Y. T. (2013a). *Variables Screening Method Based on the Algorithm of Combining Fruit Fly Optimization Algorithm and RBF Neural Network.* Advanced Materials Research,756-759,3225-3230.

Xu, F. Q., & Tao, Y. T. (2013b). *The Improvement of Fruit Fly Optimization Algorithm-Using Bivariable Function as Example.* Advanced Materials Research, 756-759, 2952- 2957.

Yang, X.-S. (2010). A new metaheuristic bat-inspired algorithm *Nature inspired cooperative strategies for optimization. Nature Inspired Cooperative Strategies for Optimization 2010* Springer. 65-74.

APPENDICES

APPENDIX A

```
function mFFOA
%*** Empty Memory
clc
clear
format long
%*** Random initial fruit fly swarm location
X_axis=10*rand();
Y_axis=10*rand();
Radius=rand(1,2);
Lamda_min=min(Radius);
Lamda_max=max(Radius);
Weight=rand(1,2);
wmax=max(Weight);
wmin=min(Weight);
%*** Set parameters
maxgen=500;%input('Provide the Maximum Generation:= '); % Iterations
clc
sizepop=50;%input('Provide the Population of Fruit Flies:='); %
Population size
%*** Optimization started, use the sense of smell to find food.
clc
testfunction=0;%input('Choose:::\n0 to Determin Q&R Matrices\n1 to Run
Ackley\n2 to Run Alpine\n3 to Run Eggcrate\n4 to Run Grienwangk\n5 to
Pathologic\n6 to Run Rastrigin\n7 to Run Rosenbrock\n8 to Run Schaffer\n9
to Run Sphere\n10 to Run Whitely\n\n::=');
QR_Iteration=ceil(maxgen/40);
if testfunction ==0
    A=[-2.02,1,0;-6.9868,-2.9476,0;0,1,0];%input('Please Enter The A
matrix:\n     Such That:\n      dx/dt=Ax(t)+Bu(t).\n\nA = ');
    clc
    B=[0.16;11.7304;0];%input('Please Enter The B matrix:\n     Such
That:\n      dx/dt=Ax(t)+Bu(t).\n\nB = ');
tic

    clc
    t=0;
    while t<QR_Iteration
    [Q,R]=lqrQR(A,B)
    t=t+1;
    end
end

if testfunction ~=0
    ite=0;
for i=1:sizepop
    X(i)=X_axis+2*rand()-1;
    Y(i)=Y_axis+2*rand()-1;
    %***? Since the food location cannot be known, the distance to the
origin is
```

```
% thus estimated first (Dist), then the smell concentration judgment
value (S) is
% calculated, and this value is the reciprocal of distance.
Lamda=Lamda_max*exp(log(Lamda_min/Lamda_max)*(i/sizepop));
w=wmax-((wmax-wmin)/sizepop)*i;

D(i)=Lamda.*((X(i).^2+Y(i).^2)^0.5);
S(i)=(1./D(i)).*w;
%*** Substitute smell concentration judgment value (S) into smell
% concentration judgment function (or called Fitness function) so as to
find the
% smell concentration (Smelli) of the individual location of the fruit
fly.
Smell(i)=objfunc(testfunction,S(i));%sin(S(i))./S(i);
end
%*** Find out the fruit fly with maximal smell concentration (finding the
% maximal value) among the fruit fly swarm.
[bestSmell bestindex]=min(Smell);
%*** Keep the best smell concentration value and x, y coordinate , and at
this
% moment, the fruit fly swarm will use vision to fly towards that
location.
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;
%*** Iterative optimization start
for g=1:maxgen
    %*** Give the random direction and distance for the search of food
using
% osphresis by an individual fruit fly.
for i=1:sizepop
X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;
%***? Since the food location cannot be known, the distance to the origin
is
% thus estimated first (Dist), then the smell concentration judgment
value (S) is
% calculated, and this value is the reciprocal of distance.
D(i)=(X(i).^2+Y(i).^2).^0.5;
S(i)=1./D(i);
%*** Substitute smell concentration judgment value (S) into smell
% concentration judgment function (or called Fitness function) so as to
find the
% smell concentration (Smelli) of the individual location of the fruit
fly.
Smell(i)=objfunc(testfunction,S(i));%Smell(i)=sin(S(i))./S(i);
end
%*** Find out the fruit fly with maximal smell concentration (finding the
% maximal value) among the fruit fly swarm.
[bestSmell bestindex]=min(Smell);
%*** Determine whether the smell concentration better than the previous
% iteration of the concentration, if yes then keep the best smell
concentration
% value and x, y coordinate, and at this moment, the fruit fly swarm will
use
% vision to fly towards that location.
if bestSmell<Smellbest
```

```
    X_axis=X(bestindex);
    Y_axis=Y(bestindex);
    Smellbest=bestSmell;
end
disp(sprintf('%8g %8g %8.4f',maxgen,bestindex,Smellbest));
%*** Each iteration the Smell optimal value, record to an array yy.
yy(g)=Smellbest;
Xbest(g)=X_axis;
Ybest(g)=Y_axis;
end
%***Draw smell concentration of each iteration
figure(1)
yy'
plot(yy,'b')
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('Smell','fontsize',12);
figure(2)
plot(Xbest,Ybest,'b');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
end
toc
end
```

APPENDIX B

```
function FFOA
%*** Empty Memory
clc
clear
%*** Random initial fruit fly swarm location
X_axis=10*rand();
Y_axis=10*rand();
%*** Set parameters
maxgen=input('Provide the Maximum Generation:= '); % Iterations
clc
sizepop=input('Provide the Population of Fruit Flies:='); % Population
size
%*** Optimization started, use the sense of smell to find food.
clc
testfunction=input('Choose:::\n0 to Determin Q&R Matrices\n1 to Run
Ackley\n2 to Run Alpine\n3 to Run Eggcrate\n4 to Run Grienwangk\n5 to
Pathologic\n6 to Run Rastrigin\n7 to Run Rosenbrock\n8 to Run Schaffer\n9
to Run Sphere\n10 to Run Whitely\n\n::=');
QR_Iteration=ceil(maxgen/40);
if testfunction ==0
    A=input('Please Enter The A matrix:\n     Such That:\n
dx/dt=Ax(t)+Bu(t).\n\nA = ');
    clc
    B=input('Please Enter The B matrix:\n     Such That:\n
dx/dt=Ax(t)+Bu(t).\n\nB = ');
tic

    clc
    t=0;
    while t<QR_Iteration
    [Q,R]=lqrQR(A,B)
    t=t+1;
    end
end

if testfunction ~=0
    ite=0;

for i=1:sizepop
    X(i)=X_axis+2*rand()-1;
    Y(i)=Y_axis+2*rand()-1;
    %***? Since the food location cannot be known, the distance to the
origin is
% thus estimated first (Dist), then the smell concentration judgment
value (S) is
% calculated, and this value is the reciprocal of distance.
D(i)=(X(i).^2+Y(i).^2)^0.5;
S(i)=1./D(i);
%*** Substitute smell concentration judgment value (S) into smell
% concentration judgment function (or called Fitness function) so as to
find the
% smell concentration (Smelli) of the individual location of the fruit
fly.
Smell(i)=objfunc(testfunction,S(i));%sin(S(i))./S(i);
```

```
end
%*** Find out the fruit fly with maximal smell concentration (finding the
% maximal value) among the fruit fly swarm.
[bestSmell bestindex]=min(Smell);
%*** Keep the best smell concentration value and x, y coordinate , and at
this
% moment, the fruit fly swarm will use vision to fly towards that
location.
X_axis=X(bestindex);
Y_axis=Y(bestindex);
Smellbest=bestSmell;
%*** Iterative optimization start
for g=1:maxgen
    %*** Give the random direction and distance for the search of food
using
% osphresis by an individual fruit fly.
for i=1:sizepop
X(i)=X_axis+2*rand()-1;
Y(i)=Y_axis+2*rand()-1;
%***? Since the food location cannot be known, the distance to the origin
is
% thus estimated first (Dist), then the smell concentration judgment
value (S) is
% calculated, and this value is the reciprocal of distance.
D(i)=(X(i).^2+Y(i).^2).^0.5;
S(i)=1./D(i);
%*** Substitute smell concentration judgment value (S) into smell
% concentration judgment function (or called Fitness function) so as to
find the
% smell concentration (Smelli) of the individual location of the fruit
fly.
Smell(i)=objfunc(testfunction,S(i));%Smell(i)=sin(S(i))./S(i);
end
%*** Find out the fruit fly with maximal smell concentration (finding the
% maximal value) among the fruit fly swarm.
[bestSmell bestindex]=min(Smell);
%*** Determine whether the smell concentration better than the previous
% iteration of the concentration, if yes then keep the best smell
concentration
% value and x, y coordinate , and at this moment, the fruit fly swarm
will use
% vision to fly towards that location.
if bestSmell<Smellbest
    X_axis=X(bestindex);
    Y_axis=Y(bestindex);
    Smellbest=bestSmell;
end
disp(sprintf('%8g %8g %8.4f',maxgen,bestindex,Smellbest));
%*** Each iteration the Smell optimal value, record to an array yy.
yy(g)=Smellbest;
Xbest(g)=X_axis;
Ybest(g)=Y_axis;
end
%***Draw smell concentration of each iteration
figure(1)
yy'
plot(yy,'-k')
```

```
title('Optimization process','fontsize',12)
xlabel('Iteration Number','fontsize',12);ylabel('Smell','fontsize',12);
figure(2)
plot(Xbest,Ybest,'b.');
title('Fruit fly flying route','fontsize',14)
xlabel('X-axis','fontsize',12);ylabel('Y-axis','fontsize',12);
end
toc
end
```

APPENDIX C

```
function xA= STEP_RESPONSE
clc
clear all
A=inputA;
B=inputB;
C=inputC;
D=inputD;
%Determin the Controllability and Observability of the system
Mcr=ctrb(A,B);
Mob=obsv(A,C);
if rank(Mcr)==size(A)
    'System is Controlable';
else
    'System is NOT Controlable'
end
CONTR_RANK=rank(Mcr)
if rank(Mob)==size(A)
    'System is Observable'
else
    'System is NOT Observable'
end
OBSV_RANK=rank(Mob)
Q_mFOA=inputQ_mFOA;
R_mFOA=inputR_mFOA;
Q_FOA=inputQ_FOA;
R_FOA=inputR_FOA;


% Settings for mFOA.
[K_mFOA P_mFOA E_mFOA]=lqr(A,B,Q_mFOA,R_mFOA)%Q and R used here are
obtained from mFOA
sys_LQ_mFOA =ss(A-B*K_mFOA ,B,C,D);
pzmap(sys_LQ_mFOA)
sgrid
pause
x=[2;-0.04;0.02];
t=0:0.02:10;
x0=initial(sys_LQ_mFOA,x,t);
%ssize(x0);
%size(X1)
X2mFOA=[1;0; 0;]*x0';
% Settings for FOA.
[K_FOA P_FOA E_FOA]=lqr(A,B,Q_FOA,R_FOA)%Q and R used here are obtained
from FOA
sys_LQ_FOA=ss(A-B*K_FOA,B,C,D);
pzmap(sys_LQ_FOA)
sgrid
pause
x=[2;-0.04;0.02];
t=0:0.02:10;
x0=initial(sys_LQ_FOA,x,t);
%size(X1)
X2FOA=[0;1;0]*x0';
% Setting for trial and error
```

```
Q=eye(size(A));
R=1;
[K1 P1 E1]=lqr(A,B,Q,R)
sys_ss=ss(A-B*K1,B,C,D);
x00=initial(sys_ss,x,t);
X11=[1;0;0]*x00';
plot(t,X2mFOA,'b',t,X2FOA,'r')
xlabel('Time(s)')
ylabel('Pitch (rad)')
legend ( 'mFFOA', 'FFOA')
hold on
% mFOA_Info=stepinfo(X2mFOA,t);
%plot(t,X2FOA,'r')
hold on
title('Response ')
xlabel('Time(s)')
ylabel(' (rad)')
%legend ( 'mFFOA', 'FFOA')
disp('mFFOA')
stepinfo(sys_LQ_mFOA)
disp('FFOA')
stepinfo(sys_LQ_FOA)
```