DEVELOPMENT OF AN IMPROVED CULTURAL ARTIFICIAL FISH SWARM ALGORITHM WITH CROSSOVER

BY

Ahmed Tijani SALAWUDEEN

Department of Electrical and Computer Engineering
Faculty of Engineering
Ahmadu Bello University Zaria, Nigeria

MAY, 2015

DEVELOPMENT OF AN IMPROVED CULTURAL ARTIFICIAL FISH SWARM ALGORITHM WITH CROSSOVER

BY

Ahmed Tijani SALAWUDEEN, B.Eng. (ABU) 2011

MSC/ENG/24418/2012-2013

salawudeenahmedtijani@gmail.com

A THESIS SUBMITTED TO THE SCHOOL OF POSTGRADUATE STUDIES,

AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A

MASTER OF SCIENCE (MSC) DEGREE IN CONTROL ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

MAY, 2015

DECLARATION

I SALAWUDEEN Ahmed Tijani, hereby declare that this Thesis titled **"Development of an Improved Cultural Artificial Fish Swarm Algorithm with Crossover (mCAFAC)"** was carried out by me under the supervision of Professor. M. B. Mu'azu and Dr. T. H. Sikiru as part of the requirements for the award of degree of Master of Science in Control Engineering. To the best of my knowledge, this research work has never been submitted anywhere for the award of any certificate. All literatures used and cited in this reported, are duly acknowledged by means of references.

Ahmed Tijani SALAWUDEEN _____ _____

CERTIFICATION

This Thisistitled “Development of an improved Cultural Artificial Fish Swarm Algorithm with Crossover meets the requirements for the award of degree of Master of Science (MSc) in Control Engineering by Ahmadu Bello University, Zaria and it is approved for its contribution to knowledge and literature.

Chairman, Supervisory Committee_____ _____
(Professor. M. B Mu’azu)**Signature** **Date**

Member, Supervisory Committee
(Dr. T. H. Sikiru)_____ _____
Signature **Date**

Head of Department
(Professor. M. B Mu’azu) _____
_____
Signature **Date**

Dean Postgraduate School
Professor. A. Z. Hassan _____
_____
Signature **Date**

DEDICATION

This research work is dedicated to promoters of peace and those who in one way or the other make a significant difference in an ordinary person's life.

ACKNOWLEDGEMENT

In the name of Allah (SWT), the most beneficent, the most merciful. All praises are due to Allah (SWT), the creator of heaven and earth, the Lord of universe. Peace and blessings be upon His prophet, Muhammad (SAW) and his companions aswell as those who follow the right path until the day of judgement.

I express my gratitude and sincere thanks to my supervisor and role model Professor. M. B. Mu'azu, for his constant motivation and support throughout my undergraduate and postgraduate studies. I truly appreciate and value his esteemed guidance and encouragement from the beginning to the end of this Thesis. I am indebted to him for having helped me shape the problem and providing insights towards the solution. His immeasurable service to humanity will surely be rewarded by God. My thanks also goes to my co-supervisor Dr. T. H. Sikiru for his unending patience, constant encouragement, guidance and expertise without which this work would not have been this special in terms of content and richness. I will never forget the long hours we spent discussing various ideas and methods in Kaduna. My deep appreciation goes to the Control Research Group and all those Thursday meetings, where academic freedom and discourse are at its best.

I am highly indebted to the entire Salawudeen's family, most especially Mr and Mrs RUKAYYA Salawudeen for their special and unequalled love towards all their children. May God bless and protect them hoping that very soon, I will repay them, part of what they really deserve. I also thank my two elder brothers Mal. Salawu Abdulrahman and Dr. Salawu Hameed for their unending love and care throughout my education. I thank God for giving me the best brothers anyone could wish for. To my sisters, Hauwa, Fatima, Khadija and beloved younger sister Amina, your support and love throughout this study would never be forgotten. May God bless and protect all of you.

I would like to express my sincere gratitude to Professor Chung C. J. who spared no time in sending me his PhD dissertation and his constant reply to my emails at the appropriate time. He said and I quote:

"Researchers who are interested in swarm optimization should consider cultural algorithm as it has proven its superiority over others"

"Congratulations!!! Great Work, Please if your papers are published let me know."

It is my pleasure to say ‘Best Wishes’ to him as he always said. I would never forget Professor Muhammad Reza Meybodi for sending me the link to his website where I downloaded some vital journal. May God reward him abundantly.

My appreciation goes to ProfessorB. G. Bajoga, Dr. B. Jimoh, Dr. Tekanyi, Dr. S. M. Sani, Dr. A. D. Usman and Dr. S. Garba for their constructive criticism, valuable contributions and creating time out of their busy schedule to go through the manuscript. I am highly indebted to all my lecturers Dr. Y. Jibril, Dr. Abu-Bilal, Dr. J. Y. Oricha,Engr. Mu’azu, Engr. A. I. Abdullahi, Engr. S. Musa, Engr. E. Okafor, Engr. Sae’ed and most especially, those whose names could not be mentioned. My esteem appreciation goes to Mal. Tukur Lawal for giving me the opportunity and much needed support to work in the Robotics and Embedded System laboratory during the odd hours. My deepest appreciation also goes to Abdullahi Tukur for all the administrative support towards me and the entire students of Department of Electrical and Computer Engineering.

My deepest appreciation goes to my special and beloved friends: A. Y. Zubair, A. M. Kabir, O. A. Abdulrahman, Kashim, Atta, Busayo, Sir Husseini, Sir Adamu, B. O. Sadiq, Sir Mustapha, A. C. Valentine, Abubakar, Arafat, Anatuku Muqtar, Aliyu Okas, I. Yusuf, Y. Abdulmumuni and those whose names could not be mentioned. There continuous support and contributions toward the success of this research would never be forgotten, may God reward them and strengthen our friendship.

Last but not the least, I would like to thank Y. A. Basira, I. L. Maryam, I. Bukola, Samira and T. T. Alonge for their support and encouragement during my stay in school. I cannot end this without mentioning these wonderful peoplePretzious, Rhodiya, Zainab, Haritha, Husseinah and RUKAYYA for making sure I was always in class doing the right thing and those who contributed to the success of this research work. May God reward all of you abundantly.

Ahmed Tijani SALAWUDEEN
ELTEEJAYY

May, 2015.

ABSTRACT

This research was aimed at the development of an improved artificial fish swarm optimization algorithm based on knowledge (normative and situational) in cultural algorithm and crossover operator called the modified Cultural Artificial Fish Swarm Algorithm with Crossover (mCAFAC). The Normative and Situational knowledge inherent in cultural algorithm were utilized to guide the step size as well as the direction of evolution of AFSA at different configurations, in order to combat the ease at which AFSA falls into local minima. An inertial weight selection is adopted 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 called the mCAFAC, in order to enhance its convergence to a global minimal. Four variations of mCAFAC (mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd and mCAFAC_NsNd) were implemented in Matlab R2013b using different configurations of the cultural knowledge. A total of sixteen test functions (Ackley, Cosine Mixture, Rastrigirn etc.) were employed to evaluate the performance of each mCAFAC variant. Simulation results showed that the mCAFAC outperformed the original AFSA with the mCAFAC_NsSd having superior performance over all the other variants.mCAFAC_NsSd produced the best result in 9 out of the 16 test cases (56.25%) while mCAFAC_NsNd produced the best result in 1 out of the 16 test cases (6.25%), mCAFAC_Ns produced the best result in 3 out of the 16 test cases (18.75%) and mCAFAC_Sd produced the best result in 1 out of the 16 test cases (6.25%). All the variants, including the standard AFSA, the modified AFSA and the replicated ABC produced the same result in 2 out of the 16 test cases (12.5%).mCAFAC_NsSd was then applied to determine the optimal values of the weighting matrices (Q and R) of linear quadruple regulator (LQR) controller.This was validated on the quadrupleInverted Pendulum (QIP) model where the obtained LQR was able to stabilize the model in 7.4161s as against 7.5162s when using the conventional trial-and-error LQR method. This showed a convergence of the solution space using both approaches with the LQR (mCAFAC) having a more optimal time-to-solution.

TABLE OF CONTENTS

DEVELOPMENT OF AN IMPROVED CULTURAL ARTIFICIAL FISH SWARM ALGORITHM WITH CROSSOVER

DECLARATION ii

CERTIFICATION iv

DEDICATION v

ACKNOWLEDGEMENT vi

ABSTRACT viii

LIST OF TABLES xv

LIST OF FIGURES xi

LIST OF ABBREVIATIONS xvi

CHAPTER ONE:INTRODUCTION

1.1 Background on Heuristics Optimization 1

1.2 Background on Optimal Control Theory 3

1.3 Aim and Objectives 5

1.4 Statement of Problem 6

1.5 Methodology 7

1.6 Significant Contributions 9

1.7 Theis Organization 9

CHAPTER TWO:REVIEW OF FUNDAMENTAL CONCEPTS

2.1 Introduction 10

2.2 Review of Fundamental Concepts on Heuristic Methods 10

2.2.1 Artificial fish swarm algorithm 10

2.2.2 Cultural algorithm 16

2.2.3 Genetic algorithm 24

2.2.4 Particle swarm optimization 26

2.2.5 Artificial bee colony optimization algorithm 28

2.2.6 Standard optimization test functions 34

2.2.6.1 Ackley function 34

2.2.6.2Cosine-Mixture function 35

2.2.6.3 DeJongF4 function 35

2.2.6.4 Expfun function 35

2.2.6.5 Griewangk function 36

2.2.6.6 Hyperelliptic function 36

2.2.6.7Levy and Montelvo first function 36
2.2.6.8 Levy and Montelvo second function 37
2.2.6.9 Neumaier3 function 37
2.2.6.10 Rastrigin function 37
2.2.6.11 Rosenbrock function 38
2.2.6.12 Sal Function 38
2.2.6.13 Schwefel function 39
2.2.6.14 Schaffer function 39
2.2.6.15 Sphere function 40
2.2.6.16 Bukin N.6 function 40
2.2.7 Determination of optimal values of weighting matrices in LQR problems 40
2.2.7.1 Design choice for determination of LQR weighting matrices 42
2.3 REVIEW OF SIMILAR WORKS 44
CHAPTER THREE: METHODS AND MATERIALS
3.1 Introduction 53
3.2 Modified artificial fish swarm algorithm 53
3.2.1 Modified preying behaviour 53
3.2.2 Modified swarming behaviour 54
3.2.3 Modified chasing behaviour 54
3.3 Modified Cultural Artificial Fish Swarm Algorithm with Crossover 57
3.3.1 mCAFAC Knowledge adjustment 57
3.4 Design Choices for the mCAFAC Influence Function 58
3.4.1 mCAFAC_Ns: mCAFAC Using normative knowledge 58
3.4.2 mCAFAC_Sd: mCAFAC using situational knowledge 60
3.4.3 mCAFAC_NsSd: mCAFAC using normative knowledge and situational knowledge 62
3.4.4 mCAFAC_NsNd: mCAFAC using normative knowledge for step size, visual distance and the direction of evolution 64
3.5 Recombination Operator 66
3.6 Determination of Weighting Matrices Using mCAFAC 68
3.6.1quadruple inverted pendulum stabilization 69
CHAPTER FOUR: RESULTS AND DISCUSSIONS
4.1 Introduction 75
4.2 Visualization of the Optimization Test Function 75
4.2.1 Ackley function 75
4.2.2 CM function 76
4.2.3 DeJongF4 function 76

4.2.4 Expfun function 77
4.2.5 Griewangk function 77
4.2.6 Hyperelliptic function 78
4.2.7 Levy and Montelvo first function 78
4.2.8 Levy and Montelvo second functions 79
4.2.9 Neumaier3 function 79
4.2.10 Rastrigin test function 80
4.2.11 Rosenbrock test fuction 80
4.2.12 Sal test function 81
4.2.13 Schwefel test function 81
4.2.14 Schaffer test function 82
4.2.15 Sphere test function 82
4.2.16 Bukin N.6 test function 83
4.3 Performance Evaluation of Proposed AFSA Algorithms 83
4.3.1 Performance evaluation of AFSA (With and Without Inertial Weight) 83
4.3.2 Performance evaluation of mCAFAC_Ns using normative knowledge. 85
4.3.3 Performance evaluation of mCAFAC_Sd using situational knowledge 86
4.3.4 Performance evaluation of mCAFAC_NsSd using situational knowledge. 86
4.3.5 Performance evaluation of mCAFAC_NsNd using situational knowledge. 87
4.3.6 Performance evaluation of artificial bee colony (ABC) algorithm 88
4.4 Parameter Settings for the Proposed Algorithm 89
4.5 Artificial Bee Colony Parameters 90
4.6 Computer System Specification 90
4.7 Application of the Proposed Controller for Optimal Determination of Controller Parameters 91
CHAPTER FIVE: CONCLUSION AND RECOMMENDATIONS
5.1 Introduction 96
5.2 Summary of Findings 96
5.3 Conclusion 97
5.3 Limitations 6
5.4 Recommendations for Further Work 7
REFERENCES 98
APPENDIX A1
m-FILE Code of AFSA. 105
APPENDIX A2
m-FILE for Artificial Bee Colony (Abc) Algorithm 109
APPENDIX B1

m-FILE for mCAFAC_Ns 112

APPENDIX B2

m-FILE for mCAFAC_Sd 115

APPENDIX B3

m-FILE for mCAFAC_NsSd 118

APPENDIX B4

m-FILE for mCAFAC_NsNd 121

APPENDIX C1

m-FILE for objective function. 125

APPENDIX C2

m-FILE for the 3d plots 134

APPENDIX D

m-FILE for the Q and R Determination 138

APPENDIX E

m-FILE for the Inverted Pendulum Responses 140

LIST OF FIGURES

Figure1.1: Free Body Diagram of Quadruple Inverted Pendulum (QIP) 4

Figure 2.1: FlowChart of Artificial Fish Swarm Algorithm 14

Figure 2.2: FlowChart of Cultural Algorithm 17

Figure 2.3: Flowchart of Artificial Bee Colony Algorithm 31

Figure 3.1: Flowchart of Modified Artificial Fish Swarm Algorithm 53

Figure 3.2: Flowchart of Modified Cultural Artificial Fish Swarm Algorithm with Crossover 64

Figure 3.3: Inverted Quadruple Inverted Pendulum Extracted From (Abdullahi, 2014) 67

Figure 3.4: Pole Zero Map of QIP System 70

Figure 4.1: 3D Visualization of Ackley Function 72

Figure 4.2: 3D Visualization of Cosine Mixture Function 73

Figure 4.3: 3D Visualization of DejongF4 Function 73

Figure 4.4: 3D Visualization of Exponential Function 74

Figure 4.5: 3D Visualization of Greiwangk Function 74

Figure 4.6: 3D Visualization of Hyperellipric Function 75

Figure 4.7: 3D Visualization of Levy and Montalvo (LM1) Optimization test function 75

Figure 4.8: 3D Visualization of Levy and Montalvo (LM2) Optimization Test Function 76

Figure 4.9: 3D Visualization of Neumaier Optimization Test Function 76

Figure 4.10: 3D Visualization of Rastrigin Function 77

Figure 4.11: 3D Visualization of Rosenbrock function 77

Figure 4.12: 3D Visualization of Sal Optimization Test Functions 78

Figure 4.13: 3D Visualization of Schwefel Function 78

Figure 4.14: 3D Visualization of Schaffer Function 79

Figure 4.15: 3D Visualization of Sphere Optimization Test Function 79

Figure 4.16: 3D Visualization of Bukin Optimization Test Function 80

Figure 4.17: Response of the first pendulum 89

Figure 4.18: Response of the Second Pendulum 90

Figure 4.19: Response of the Third Pendulum 91

Figure 4.20: Response of the Fourth Pendulum 92

LIST OF TABLES

Table 4.1: Performance Evaluation of Standard AFSA and mAFSA Algorithm 81

Table 4.2: Performance Evaluation of Standard AFSA, mAFSA and mCAFAC_Ns Algorithms 82

Table 4.3: Performance Evaluation of Standard AFSA, mAFSA, mCAFAC_Ns and mCAFAC_Sd Algorithms 83

Table 4.4: Performance Evaluation of Standard AFSA, mAFSA, mCAFAC_Ns, mCAFAC_Sd and mCAFAC_NsSd Algorithms 84

Table 4.5: Performance Evaluation of Standard AFSA, mAFSA, mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd and mCAFAC_NsNd Algorithms 85

Table 4.6: Performance Evaluation of Standard AFSA, mAFSA, mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd, mCAFAC_NsNd and ABC Algorithms 86

Table 4.7: Parameter Setting for AFSA and mCAFAC 87

Table 4.8: Parameter Setting for ABC 87

Table 4.12: Comparison of Responses for the Fourth Pendulum 92

LIST OF ABBREVIATIONS

3D	Three Dimensional
ABC	Artificial Bee Colony
ACO	Ant Colony Optimization
AF	Artificial Fish
AFA	Artificial Fish Algorithm
AFSA	Artificial Fish Swarm Algorithm
AFS	Artificial Fish Swarm
AFSO	Artificial Fish swarm Optimization Algorithm
AFSAVP	Artificial Fish Swarm Algorithm with Variable Population
AHAFSA	Adaptive Hybrid Artificial Fish Swarm Algorithm
ANN	Artificial Neural network
BFA	Bacterial Foraging Algorithm
BLF	Believe
CA	Cultural Algorithm
CBPSO	Cultural Based Particle Swarm Optimization
CEP	Cultural Evolution Process
CI	Computational Intelligence
CV	Cross Validation
CM	Cosine mixture
DE	Deferential Evolution
EP	Evolutionary Programming
FNN	Fuzzy Neural Network
FFA	Fire Fly Algorithm
GA	Genetic Algorithm
GSO	Glow-warm Swarm Optimization
GSAFSA	Group Search Artificial Fish Swarm Algorithm
LM	Levey and Montelvo
LQR	Linear Quadruple Regulator

Matlab	Matrix Laboratory
mAFSA	modified Artificial Fish Swarm Algorithm
mCAFAC_Ns	modified Cultural Artificial Fish Swarm Algorithm with Crossover (Normative Knowledge for step size)
mCAFAC_Sd	modified Cultural Artificial Fish Swarm Algorithm with Crossover (Situational Knowledge for Visual Distance)
mCAFAC_NsSd	modified Cultural Artificial Fish Swarm Algorithm with Crossover (Normative Knowledge for Step Size and Situational Knowledge for Visual Distance)
mCAFAC_NsNd	modified Cultural Artificial Fish Swarm Algorithm with Crossover (Normative Knowledge for Step Size and Normative Knowledge for Visual Distance)
LTI	Linear Time Invariant System.
MF	Membership Function
MIP	Mixed Integer Programming
NP	Non-Deterministic
OS	Operating System
PSO	Particle Swarm Optimization
POP	Population
QIP	Quadruple Inverted Pendulum
QAFSA	Quantum Artificial Fish Swarm Algorithm
RAM	Read Access Memory
RUL	Remaining Useful Life
RCGA	Real Coded Genetic Algorithm.
SA	Simulated Annealing
SBFC	Swarm Based Fuzzy Controller
SHS	Short-term Hydro Scheduling
SOA	Swarm Optimization Algorithm
TS	Taboo Search

CHAPTER ONE

INTRODUCTION

1.1 Background on Heuristics Optimization

Optimization is a subject that deals with the problem of minimizing or maximizing a certain function in a finite dimensional space over a subset of that space, which is usually determined by functional inequalities (Wang & Li, 2015). During the past century, optimization has been developed into a mature field that includes many branches, such as linear conic optimization, convex optimization, global optimization, discrete optimization, etc. Each of such branches has a sound theoretical foundation and is featured by an extensive collection of sophisticated algorithms and software. Optimization, as a powerful modelling and problem solving methodology, has a broad range of applications in management science, industry and engineering (Nashat et al., 2012)

There is no known single optimization method available for solving all optimization problems. Several Optimization Algorithms have been developed in recent years. These Algorithms includes, Artificial Fish Swarm Algorithm (AFSA) (Li, 2002), Artificial Bee Colony Optimization (ABC) (Karaboga, 2005), Particle Swarm Optimization (PSO) (Eberhart & Kennedy, 1995), Genetic Algorithm (GA) (Holland, 1959), Ant Colony Optimization (ACO) (Dorigo, 1996), Fire-Fly Algorithm (FFA) (Yang, 2010), Bacterial Foraging Algorithm (BFA) (Passino, 2002) and Cultural Algorithm (CA) (Reynolds, 1994).

Some of these optimization methods mimicking evolution, animal behaviour, rules of natural ecology and mechanisms of human culture in nature were developed in order to solve certain categories of complicated scientific, Social and engineering design problems (Chung, 1997):

i) Weak problems, problems with little or no domain knowledge

ii) Non-deterministic Polynomial (NP) complete problems

iii) Problems for which a near optimal solution can be acceptable

iv) Problems with non-smooth (discontinuous; not-differentiable) and noisy search space.

v) Problems whose environments are uncertain and/or dynamic

This research considers artificial fish swarm optimization, a relatively recent addition to the field of natural computational intelligence that has elements inspired by the social behaviours of natural swarms of fish. Since its introduction in 2002, AFSA has found widespread application in complex optimization domains, and is currently a major research topic, offering an alternative to the more established evolutionary computational techniques that may be applied in many of the same domains.

The AFSA algorithm was first introduced by Li Xiao Lei (Li *et al.*, 2002)and its basic idea was inspired by simulation of the social behaviour of fish(Li et al., 2002). It is based on the natural process of group communication to share individual knowledge when a group (Swarm) of fish search for food, although all fish do not know where the best food source is. But from the nature of the social behaviour, if any member can find out a desirable path to go, the rest of the members will follow suit quickly (Neshat et al., 2012). AFSA is modelled using the three (Preying, Swarming and Chasing) basic behaviour of fish. In this algorithm, each fish swarms through the multidimensional search space and adjusts its position in every step with its own experience and the experience of neighbouring fishes toward an optimum solution by the entire swarm (Li, 2002). Its next behaviour depends on its current state and its local environmental state (including the quality of the question solutions at present and the states of nearby companions).

AFSA possess similar attractive features of genetic algorithm (GA) such as independence from gradient information of the objective function, the ability to solve complex nonlinear

high dimensional problems. Moreover, they can achieve faster convergence speed and require few parameters to be adjusted. The system is started first in a set of randomly generated potential solutions, and then performs the search for the optimum solution one interactively (Zhang et al. 2006). Similar to human evolution processes, the evolution of AFSA can be represented in form of Cultural Evolution Process (CEP).

Cultural Algorithms were initially developed by Reynolds (Reynolds, 1994) in order to model the evolution of the cultural component of an evolutionary computational system over time as it accumulates experience. The cultural algorithm framework offers a mechanism supporting the dual inheritance system characterizing both the human culture at the macro evolutionary level, and individual phenotypic evolution at the micro evolutionary level (Chung, 1997). It also provides a framework in which knowledge acquired at the cultural level is isolated and exploited to accelerate the problem solving process. This research focuses on modifying the AFSA, using the cultural algorithm knowledge to accelerate the problem solving process during evolution of AFSA.

1.2 Background on Optimal Control Theory

Optimal control theory is a mathematical optimization method which is an extension of the calculus variation with numerous applications in control engineering (Ata & Coban, 2015). Verifying the control signals that will cause a process to meet the desired physical constraints and also optimization (maximization or minimization) of some performance criteria are the main objectives of optimal control system (Das et al., 2013). A special case of the general nonlinear optimal control problem where the cost function is quadruple and the system dynamics are described by a set of linear differential equations is a linear quadruple optimal control problem (Schildbach et al., 2015).

One of the challenging control and benchmark problems in the field of complex non-linear control system is the swinging of an inverted pendulum. Stabilizing and control of such an unstable system has been a major subject of research for control engineers. A cart-driven inverted pendulum has a structure where the pendulum is hinged to the cart via a pivot and only the cart is actuated (Boubaker, 2013). The motion of the pendulum has to be controlled by moving the cart back and forth within a limited movement of the cart. The linearized quadruple inverted pendulum (QIP) adopted in this research is obtained from the unpublished M.Sc work of Abdullahi Mohammed(Abdullahi, 2014). Figure 1.1 shows the free diagram of a typical QIP inverted pendulum.

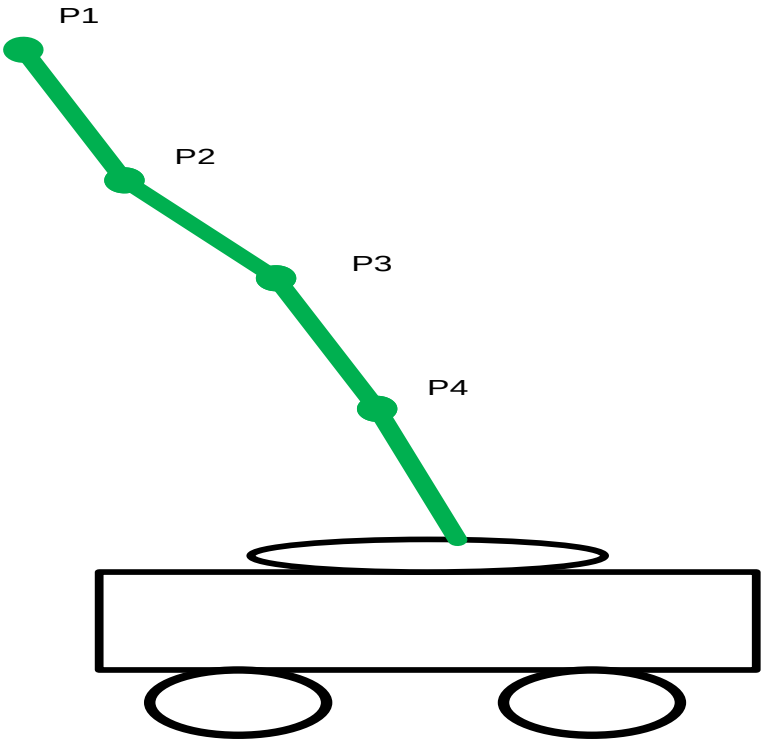


Figure 1.1: Free Body Diagram of Quadruple Inverted Pendulum (QIP)

Where P1, P2, P3 and P4 are the first, second, third and fourth pendulum respectively of the QIP of Figure 1.1.

Linear Quadruple Regulator (LQR) optimal controllers have been predominantly used for the control of such movement as inverted pendulum. The critical design parameters for the LQR are the weighting matrices (Q and R) in the cost function and are normally selected by the designer(Hamidi, 2012). The other parameters, the state and input matrices (A and B), are obtained from the state model of the system. Hence, even though the LQR is an optimal scheme, the method of selecting Q and R is not optimal.

1.3 Aim and Objectives

The aim of this research is to develop an improved artificial fish swarm optimization algorithm based on knowledge in cultural algorithm and crossover operator called modified Cultural Artificial Fish Swarm Algorithm with Crossover (mCAFAC). An inertial weight selection was introduced such that the algorithm can adaptively select its parameters when searching for global solution.

The objectives of the research are as follows:

i) Development of the improved artificial fish swarm optimization algorithm, mCAFAC, using knowledge in culture algorithm, Inertia Weight and genetic crossover operator. These modifications should not significantly complicate the algorithm but should improve its convergence speed, its robustness and its ability to escape local minima.

ii) Evaluation of the performance of the algorithm using sixteen well-known test functions (such as: Ackley, Dejongf4, Schwefel etc.).

iii) Validation of the performance of the algorithm by comparison of the results in item ii) with those obtained using standard AFSA and Artificial Bee Colony (ABC) Algorithm.

iv) Application of the mCAFAC to determine the weighting matrices (Q and R) of Linear Quadruple Regulator (LQR). This is validated using the Quadruple Inverted Pendulum (QIP) model.

1.4 Statement of Problem

The AFSA can search for global optimum effective, and has an adaptive ability for search space. The individual behaviour is to search for local optimum, therefore avoiding individual premature becomes difficult due to slow convergence speed at later stage, a lot of Artificial Fish (AF) performs invalid search which increases the computation time and decreases precision accuracy. In this case, artificial fish can be stuck into local optima when dealing with multi-modal optimization problems. Parameters such as visual distance and step size have demonstrated great influence on the performance of the algorithm. When the visual distance is large, the AFSA has strong global searching ability and when the visual is small, the AFSA has strong local searching ability.Similarly the bigger the step size, the faster the convergence speed and the smaller the step the slower the convergence speed. In order to combat these shortcomings, a modified cultural artificial fish swarm algorithm with inertia weight and crossover operator is developed.

1.5Scope and Limitations

The limitations of this research work are highlighted as follows:

i) This research work did not consider negative cultural knowledge.

ii) This research work did not consider other form of cultural knowledge (such as: Domain, Topographical and Historical knowledge)

iii) This research work did not consider the improvement on artificial bee colony algorithm

5.4 Recommendations for Further Work

The following possible areas of further work are recommended for consideration for future research:

1. Other types of Cultural knowledge such as Historical, Topographical or Domain knowledge can be considered as alternative source of belief space structure instead of the Normative and/or Situational.
2. Modelling cultural based versions of other heuristic algorithms such as Artificial Bee Colony (ABC), FireFly Algorithm (FFA), Bacterial Foraging algorithm (BFA), Cuckoo Search Algorithm (CSA), etc. can be considered.
3. The effect of negative cultural knowledge can also be investigated.
4. Hybridization of AFSA with other heuristic optimization algorithms such as Artificial Bee Colony (ABC), FireFly Algorithm (FFA), Bacterial Foraging algorithm (BFA), Cuckoo Search Algorithm (CSA), etc. can be considered.
5. The algorithm can be applied to areas such as image processing, data clustering, WSN deployment power system stability analysis, constrained and unconstrained optimization problems etc.

1.5 Methodology

The methodology adopted for this research involves developing the artificial fish swarm algorithm using Matlab 2013Rb Control/Optimization toolboxes, then both situational and normative knowledge in cultural algorithm was used to guide the evolution, which will then

be use to initialize the population of the proposed improved algorithm. The steps of the proposed algorithm are highlighted as follows:

1. Initialize the population of artificial fish, which shall be N number of fish in D dimension.
2. Develop the AFSA algorithm as described in Figure 1.1.
    a. Preying Behaviour
    b. Swarming Behaviour
    c. Chasing Behaviour
3. Knowledge (Situational and Normative) in cultural algorithm is used to guide the evolution of AFSA.
4. Inertial weight selection is introduced, such that the algorithm can adaptively select its own parameters during evolution.
5. Crossover criterion is set to judge whether the algorithm falls into local optima or not.
6. Four Variation of Modified Cultural Artificial Fish Swarm Algorithm are developed, which include:
    a) mCAFAC_Ns.
    b) mCAFAC_Sd.
    c) mCAFAC_NsSd
    d) mCAFAC_NsNd
7. The Modified algorithm is validated using a total of sixteen optimization test functions (e.g. Ackley, Cosine Mixture, Rastrigirn etc.)
8. The performance of the modified algorithm is compared with that of the Artificial Bee Colony (ABC) algorithm.
9. Finally, the Algorithm is also used to determine the weighting matrices of the LQR controller used for the quadruple inverted pendulum (QIP) model.

1.6 Significant Contributions

A lot of research works have been done on improving the precession accuracy and convergence speed of Artificial Fish Swarm Algorithm. Many researches have also been conducted on different area of its application. The significant contributions of this research work are as follows:

i. Development of a modified Cultural Artificial Fish Swarm Algorithm with Crossover (mCAFAC) and inertial weight with particular emphasis on visual distance and step size. This has added adaptive capabilities to the standard AFSA

ii. Four variants of the mCAFAC using different configurations of cultural knowledge were developed and validated on several test functions. All the variants (mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd and mCAFAC_NsNd) outperformed the standard AFSA by 18.75%, 6.25%, 56.25% and 6.25% respectively

iii. mCAFAC was used to obtain the optimal weighting matrices (Q and R) of Linear Quadruple Regulator (LQR) controller, which are critical in stabilizing and optimally controlling systems. This was validated using the QIP as a case study and the values obtained using the settling time as a metric. The LQR (mCAFAC) settled the QIP in 7.4161s as against 7.5162s obtained using the LQR (conventional trial-and-error method)

1.7 Thesis Organization

The general introduction has been presented in Chapter One. The rest of the chapters are structured as follows: Detail review of related literatures and relevant fundamental concept

about AFSA, CA, PSO, GA, ABC and LQR is carried out in Chapter Two. In-depth approach and relevant mathematical models describing the modification of an improved cultural artificial fish swam algorithm with crossover alongside the application LQR weighting matrices determination and QIP stabilization analysis were presented in Chapter Three. The analysis, performance and discussion of the result were analysed in Chapter Four. Conclusion and recommendations makes up the Chapter Five. Quoted references and Appendices are also provided at the end of this Thesis.

CHAPTER TWO

REVIEW OF FUNDAMENTAL CONCEPTS

2.1 Introduction

This chapter consists of the review of the fundamental concepts and the review of similar works which are relevant to the scope of this research.

2.2 Review of Fundamental Concepts on Heuristic Methods

In this subsection, concepts that are fundamental to AFSA, Cultural Algorithm, Genetic Algorithm, Particle Swarm Optimization, Artificial Bee Colony Algorithm and Linear Quadruple Regulator are presented.

2.2.1 Artificial fish swarm algorithm

The optimization algorithms which are inspired by intelligent behaviour of school of fishes (Artificial fish swarm - AFS) are among the most recently introduced techniques. Several approaches have been proposed to model the specific intelligent behaviours of fishes and since its invention (Lei, 2002), AFSA has been successfully applied to many kinds of problems (Mobayen *et al.*, 2011; Orivuori *et al.*, 2010; Peng *et al.*, 2013; Xu *et al.*, 2014; Xu and Long (2014); Yi *et al.*, 2014; Yu & Li, 2014; Zeng & Xu, 2012; Zhou *et al.*, 2010; Zhu &

Lin, 2013) and (Hassan *et al.*, 2015). According to the various applications mentioned, AFS algorithm confirmed its good performance.

The basic behaviour of fish in water is to discover regions with more food, by vision or sense; the place with more fish is generally the place with most food. Therefore, the environment where an AF lives is mainly the solution space and is the states of other AFs (Neshat et al., 2012). Its next behaviour depends on its current state and its local environmental state (including the quality of the question solutions at present and the states of nearby companions). The individual behaviour can be represented as follows (Lei 2002):

1. Synergetic move rule: Basic communication is kept between individuals. If an individual in the swarm receives a call sent by the other individuals, it moves forward to the call individual's position by a step with a certain probability.
2. Reconnaissance move rule: If the individual does not receive a call, it implements reconnaissance according to its own swarm historical experiment. It finds a better goal and then moves forward to the position by a step.
3. Stochastic move rule: if the searching individuals does not receive a call or does not find a better goal, it moves a step randomly. If it then finds a better goal during the random move, it sends a call to the searching swarm.

An artificial fish would influence the environment via its own activities and its companions' activities. The system is started first in a set of randomly generated potential solutions, and then performs the search for the optimum one interactively (Zhang et al, 2006).

The idea of Artificial Fish Algorithm (AFA) is to imitate the fish behaviours, such as preying, swarming and chasing. Suppose the problem under consideration has D-dimensions. Initialize the state vector of the Swarm with N artificial fish such that the state of one artificial fish can be formulated as: $X_i = (x_{i1}, x_{i2}, ..., x_{iD})\ for.i = 1,2.....N$ where X_i is the status of the fish, which represent the target variable for the problem under consideration (Wu et al., 2011). The

current food concentration in the position of fish is expressed as $y = f(x_i)$ which is the objective function. The visual distance between the artificial fish is $d_{i,j} = \|X_i - X_j\|$, where i and j is a randomly generated fish. Step is the maximum size of the movement of artificial fish. δ Is the degree of congestion (crowd factor).

2.2.1.1 Preying

Preying is the basic biological behaviour of fish; generally the fish perceives the region with more food in water by vision or sense and moves quickly towards this region.

Suppose the current state of AF is X_i, the artificial fish select a state randomly within its visual distance such that (Wu et al, 2011):

$$X_j = X_i + rand(0,1) \times visual \tag{2.1}$$

Where X_j is the new state and X_i is the previous state

If $f(X_j) < f(X_i)$ in the minimum problem, it goes forward a step towards X_j in the following direction (Wu et al., 2012).

$$X_i^{(t+1)} = X_i^{(t)} + rand(0,1) \times step \times \frac{X_j^{(t)} - X_i^{(t)}}{\left\| X_j^{(t)} - X_i^{(t)} \right\|} \tag{2.2}$$

Where $\left\| X_j^{(t)} - X_i^{(t)} \right\| = \left| \sqrt{((X_j^{(t)})^2 - (X_i^{(t)})^2)} \right|$ is the Euclidean distance between the artificial fish j and artificial fish i.

If $f(X_j) > f(X_i)$, the artificial fish selects another state randomly again. If the artificial fish cannot meet the requirement in a given time, it moves one step randomly as (Wu et al., 2011):

$$X_i^{(t+1)} = X_i^{(t)} + rand(0,1) \times step \tag{2.3}$$

2.2.1.2 Swarming

The fish will assemble and move in groups as a natural mechanism of guaranteeing their existence and avoiding danger. Suppose the current state of the artificial fish is, X_i and nf is the number of its fellows within the visual distance, which is equal to the number of elements in the set of $B = \{X_i \mid d_{ij} \leq visual\}$. If $nf \neq 0$, which means the set of element B is not empty, let X_c be the centre position and Y_c stands for the fitness of the centre position. Let $X_c = \sum_{j}^{nf} X_j / nf$ and $Y_c = f(X_c)$. If $nf \times Y_c < \delta \times Y_i$, this area is not crowded. If $Y_c < Y_i$ the artificial fish moves one step forward towards the companions centre position (Gao et al., 2010).

$$X_i^{(t+1)} = X_i^{(t)} + rand(0,1) \times step \times \frac{X_c^{(t)} - X_i^{(t)}}{\left\| X_c^{(t)} - X_i^{(t)} \right\|} \tag{2.4}$$

Otherwise, it executes the preying behaviour. The crowd factor limits the scale of swarms, and more AF only cluster at the best area, which ensures that AF move to optimum in a wide field.

2.2.1.3 Chasing

When a fish finds food, neighbouring fish will trail and reach the food. Suppose the current state of the artificial fish is X_i, and X_m stands for the best artificial fish individual within $X_i's$ visual distance. nf Is the number of $X_m's$ within the visual distance. $Y_m = f(X_m)$, if $Y_m < Y_i$ and $nf \times Y_m < \delta \times Y_i$ the artificial fish moves one step towards X_m (Gao et al., 2010):

$$X_i^{(t+1)} = X_i^{(t)} + rand(0,1) \times step \times \frac{X_m^{(t)} - X_i^{(t)}}{\left\| X_m^{(t)} - X_i^{(t)} \right\|} \tag{2.5}$$

Otherwise it executes the behaviour of preying.

Swarming makes few fish confined in local extreme values move in the direction of a few fishes tending to global extreme value, which results in AF fleeing from the local extreme values. Chasing accelerates AF moving to better states, and at the same time, accelerates AF moving to the global extreme value field from the local extreme values. The flowchart for the artificial fish swarm algorithm is shown in Figure 2.1 (Yazdani et al. 2013):

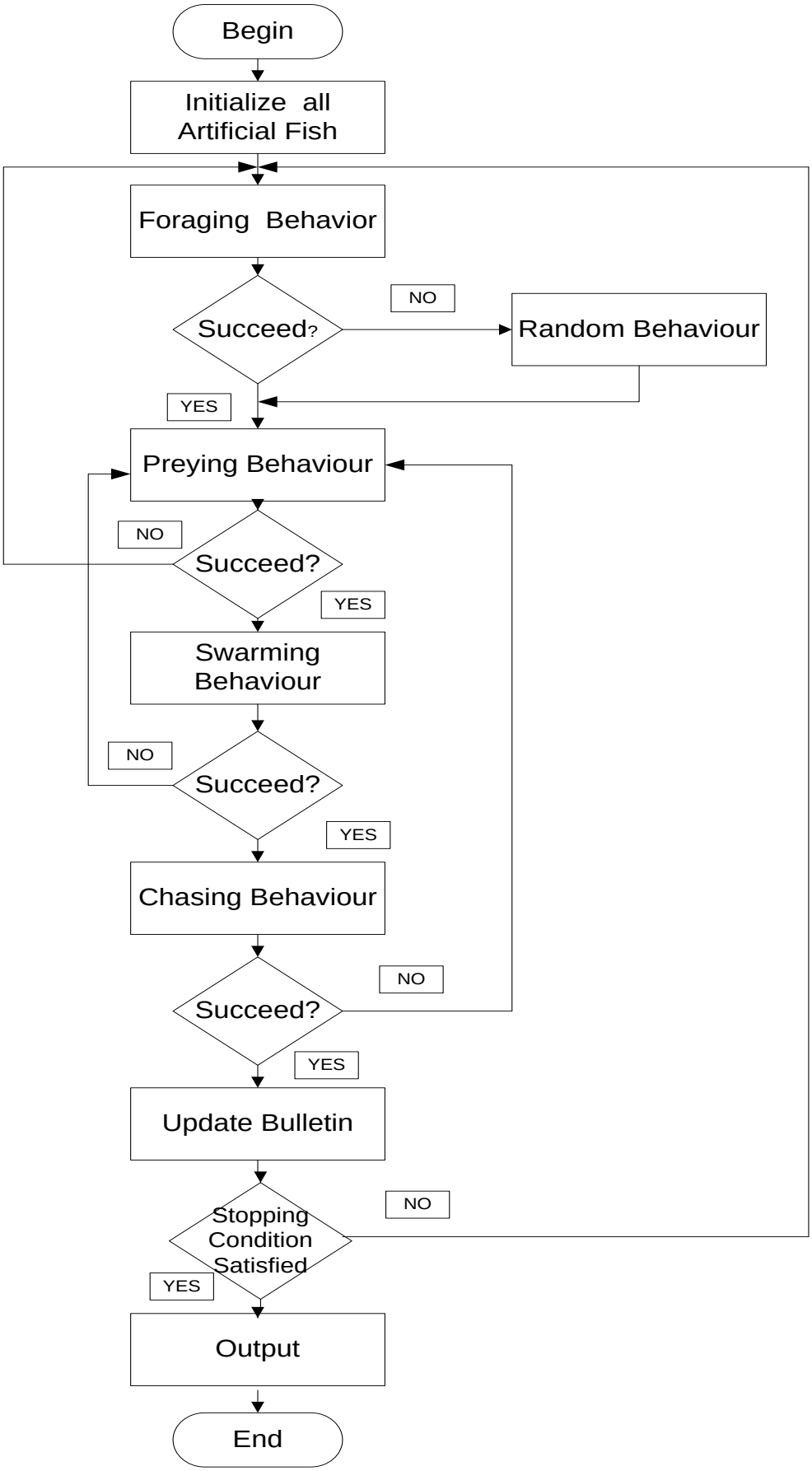


Figure 2.1: Flowchart of Artificial Fish Swarm Algorithm

2.2.2 Cultural algorithm

Cultural Algorithms (CA) was developed by R. G. Reynold (Reynold, 1994). The algorithm consists of a social population and a belief space. Cultural Algorithms (CAs) are a class of computational models derived from models of the cultural evolution process. Experience of individuals selected from the population space by the acceptance function is used to generate problem solving knowledge that resides in the belief space. The belief space stores and manipulates the knowledge acquired from the experience of individuals in the population space (Reynold and Chung, 1997). This knowledge can control the evolution of the population component by means of the influence function. As a result, Cultural Algorithms can provide an explicit mechanism for global knowledge and a useful framework within which to model self-adaptation in an Evolutionary Computation (EC) system (Chung, 1997). The cultural algorithm framework offers a mechanism supporting the dual inheritance system characterizing both the human culture at the macro evolutionary level, and individual phenotypic evolution at the micro evolutionary level. It also provides a framework in which knowledge acquired at the cultural level is isolated and exploited to accelerate the problem solving process. It has been suggested that cultural evolution proceeds at a faster rate than biological evolution (Reynolds and Chung, 1997) due to the guiding influence of the cultural knowledge.

The cultural evolution process can be viewed as a vehicle for amplifying individual or group behaviour and building consensus. In other words, during cultural evolution, “conceptual beacons” that symbolize acceptable and unacceptable behaviour for individuals in a population (society) are accumulated (Chung, 1997).

In this research, cultural algorithm as well as cross-over and inertial weight is used to implement an improvement on Artificial Fish Swarm Algorithm (AFSA). The population

level of the cultural algorithm component is the Artificial Fish Swarm Algorithm. The global knowledge that has been learned by the population is expressed in terms of both normative and situational knowledge(Reynolds & J., 1997).

The basic aspects of the Cultural Algorithm's structure are:

1) The level of self-adaptation
2) Type of knowledge in the belief space
3) Methods to acquire knowledge: acceptance function,
4) Methods to exploit the knowledge to guide the population: influence function.

The performance measures that, this research is focused on are:

a) Accuracy of the solution
b) The execution speed,
c) The rate of convergence, and
d) The probability of finding global optimum solutions.

The pseudo code description of cultural algorithm is given as follows:

```
Begin
   t = 0;
   Initialize POP(t);
   Initialize BLF(t);
   repeat
      Evaluate POP(t);
      Vote(BLF(t), Accept(POP(t)));
      Adjust(BLF(t));
      Evolve(POP(t), Influence(BLF(t)));
      t = t + 1;
      Select POP(t) from POP(t-1);
   until (termination condition achieved)
End
```

Population are first initialized which is done using AFSA in this research. Individuals in the population are then evaluated using some performance (test) functions. The performance information represents the problem solving experience of an individual (Reynold and Chung, 1997). An acceptance function determines which individuals in the current population are able to impact or to be voted to contribute to the current beliefs. The experience of these selected individuals are used to adjust the current group beliefs. These group beliefs are then used to guide and influence the evolution in the population at the next step, where parameters for self-adaptation can be determined by the belief space. Information that is stored in the belief space is suitable for self-adaptation, since it contains generalizations of individual experience which can be used to direct the future behaviour of individuals in the population. The framework of cultural algorithm is given in Figure 2.2

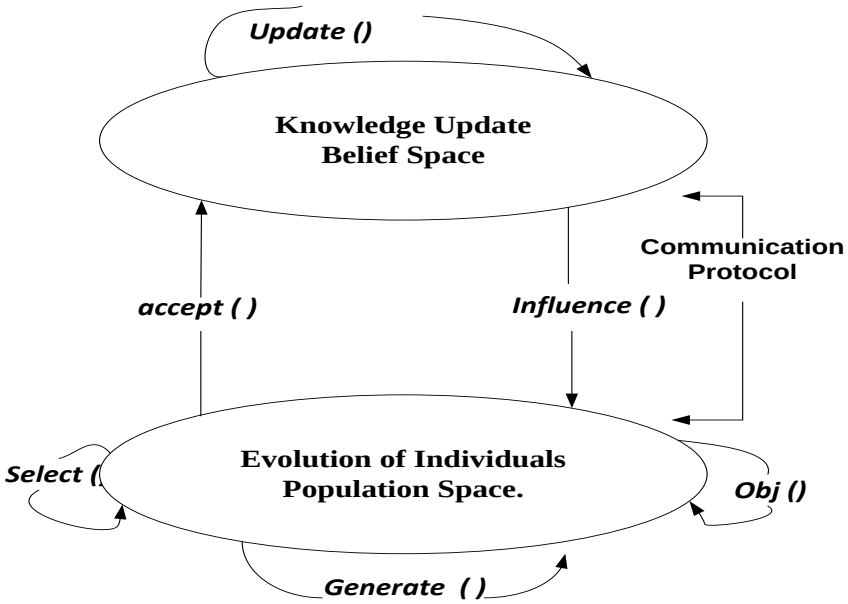


Figure 2.2: Framework of Cultural Algorithm

The Cultural Algorithm can be viewed as a dual inheritance system with evolution taking place at the population level and at the belief space level. The two components interact through a communications protocol. The protocol determines the set of "acceptable" individuals that are able to update the belief space (Chung, 1997). Likewise the protocol determines how the updated beliefs are able to impact and influence the adaptation of the population component.

It is important to know that growth and development of a society is affected by the creative mind of committed minority (Nguyen & Yao). In this research, the creative minority is selected from the population using the acceptance function and then, they are used to generalise the problem solving experience in the belief space.

The knowledge in the belief space can influence the evolution, in two ways:

(1) Determining the size of the changes, step size

(2) Determining the direction of change, e.g. increase or decrease the current value (Visual).

This is then used to guide the evolution of AFSA. Crossover operator was applied to help the artificial fish not only jump out local minima search but also inherits the advantage of its parents (Wu et al., 2011).

2.2.2.1 Knowledge Representation in the Belief Space

In a Cultural Algorithm, the belief space is where the culture knowledge is formed and maintained. Five basic categories of cultural knowledge have been identified: Normative, Situational, Domain, History, and Topographical Knowledge (Jin & Reynolds, 1999). For the purpose of this research, only normative and situational knowledge was considered as they are deemed more fundamental to the operation of cultural system.

2.2.2.2 Normative Knowledge

Normative knowledge is a set of promising variable ranges that provide standards for individual behaviours and guidelines within which individual adjustments can be made (Reynolds & Peng, 2004). The normative knowledge describes the feasible solution space of the optimization problems. It also provides standards for individual behaviour and provides guidelines within which individual adjustments can be made (Chung, 1997). The normative knowledge component in the belief space, N, is formally updated as follows:

$$N = \langle I, U, L, D \rangle, \tag{2.6}$$

Where U, L and D, are n -dimensional vectors, and I_j denotes the closed interval for variable j that is a continuous set of real numbers x represented as a number pair:

$$I_j = [l_j, u_j] = \{ x \mid l_j \le x \le u_j \} \tag{2.7}$$

n is the number of the variables, l_j and u_j are the lower and upper bounds for the j^{th} variable, respectively, L_j and U_j are the values of the fitness function associated with the

bound l_j and u_j. Generally, l_j and u_j are initialized with the lower and upper bounds of individuals. L_j and U_j are usually initialized with positive infinity.

Usually the normative knowledge leads individuals to "jump into the good range" if they are not already there (Reynolds and Peng, 2004). In the normative knowledge, the parameter values for individuals selected by the acceptance function are used to calculate the current acceptable interval for each parameter in the belief space. The idea is to be conservative when narrowing the interval and be progressive when widening the interval (Chung, 1997; Reynolds & Chung, 1996). The interval update rule is formulated as follows:

For the left boundary and its score for parameter j (Reynolds & J., 1997):

$$l_j^{t+1} = \begin{cases} x_{i,j}^t & if \ x_{i,j}^t \le l_j^t \ \ or \ \ f(x_i^t) < L_j^t \\ l_j^t & otherwise \end{cases} \tag{2.8}$$

$$L_j^{t+1} = \begin{cases} f(X_i) & if \ x_{i,j}^t \le l_j^t \ or \ f(x_i^t) < L_j^t \\ L_j^t & otherwise \end{cases} \tag{2.9}$$

Where i^{th} individual affects the lower bound for parameter j. l_j^t Represents lower limit for parameter j at generation t and L_j^t denotes the performance score (fitness) for it.

For the right boundary and its score for parameter j:

$$u_j^{t+1} = \begin{cases} x_{k,j} & if \ \ x_{k,j} \ge u_j^t \ \ \ or \ \ \ f(x_k) < U_j^t \\ u_j^t & otherwise \end{cases} \tag{2.10}$$

$$U_j^{t+1} = \begin{cases} f(x_k) & if \ \ x_{k,j} \ge u_j^t \ \ \ or \ \ \ f(x_k) < U_j^t \\ U_j^t & otherwise \end{cases} \tag{2.11}$$

Where k^{th} individual affects the upper bound for parameter j. u_j^t represents upper limit for

variable j at generation t and U_j^t denotes the performance score for it.

2.2.2.3 Situational Knowledge

Situational knowledge provides a set of exemplary cases that are useful for the interpretation of specific individual experience. Situational knowledge leads individuals to "move towards the exemplars" (Reynolds and Peng, 2004). Here the situational knowledge exemplar set consists of only the best solution found so far (Wu et al, 2011), that is, at iteration t, $S =< S^t \mid S^t = \{s_1^t, s_2^t, ..., s_n^t\} >$, in other words, it can be initialized with the best fish found so far in the initial fish swarm and updated by the following rule (Reynolds & J., 1997):

$$S_j^{t+1} = \begin{cases} X_{best,j}^{t+1} & if \quad f(X_{best,j}^{t+1}) < f(s^t) \\ s_j^t & otherwise \end{cases}, \tag{2.12}$$

Where X_{gbest}^{t+1} denotes the best artificial fish individuals in the swarm at generation $t+1$.

Normative knowledge can be analogous to social standard and situational knowledge can be analogous to social model cases. It is important to note that both normative and situational knowledge can be positive and/or negative. However in this study, only positive knowledge is considered because it is considered as the foundation of cultural knowledge (Reynolds, 2004). Considering negative knowledge can be considered as a future research subject.

2.2.2.4 Domain Knowledge

In this class, the problem domain knowledge is used to guide the evolution of the search. For example, in a functional landscape composed of Ackley function, knowledge about Ackley shape and the related parameters is useful in deciding which direction to move during the search process (Reynolds & Peng, 2004).

2.2.2.5 Historical Knowledge

Historical or Temporal Knowledge monitors the search process and records the important information during the search process. These important events may either be a significant move in the search space or a detection of change in the landscape. Individuals guided by the history knowledge can consult those recorded events for guidance in selecting a direction of movement.

2.2.2.6 Topographical Knowledge

Topographical Knowledge is proposed to reason about region-based functional landscape patterns (Jin & Reynolds, 1999). The whole landscape is divided into cells according to spatial characteristics and each cell keeps track of the best individual in its region. Topographical Knowledge guides individuals to copy the behaviours of the cell-best individual.

2.2.2.7 Acceptance Function

An acceptance function selects individuals and their performances who will impact the forming of the current belief space (Reynolds & Peng, 2004). A number of different classes of acceptance function can be employed. The first class of functions are deterministic in nature and use only information about the absolute ranking (top 10%, top 20%, top 30%, top 40%, and top 50%-median of individuals in the population, respectively) or relative ranking (above average individuals) to select the individuals that can be used to update the belief space (Chung, 1997). The second class of functions are also deterministic in nature but use temporal information about the number of generations used so far. The number of the individuals accepted for the update of the belief space in this class is obtained according to the following function: (Reynolds and Chung, 1994)

$$f_a(N,t) = N.\beta + \lfloor N.\beta / t \rfloor \quad (2.13)$$

Where N is the size of the swarm (population), t is the current number of generation and β is usually chosen to be 0.2 (which means top 20% of the population). However, a higher value is usually selected at the beginning of the search process.

2.2.3 Genetic algorithm

The precursors of genetic algorithms (GA) can be traced back to the early 1950s when several biologists used computer models to simulate biological systems (Xiaoping & Liming, 2002). Then, early 1970s, the genetic algorithm was developed by John Holland (Holland, 1975).

The theoretical foundations of Genetic Algorithms relied on a binary string representation of chromosomes, and on the notion of schema which is a template allowing exploration of similarities among chromosomes (Moradi & Abedini, 2012). In GA, population members are considered analogous to chromosomes that contain sequence of genes (Goldberg & Kuo, 1987). Each structure, which in this case is purely genotypic, is decoded into a phenotypic expression which is in turn evaluated in the light of a performance function. A selection operator acts to replicate the structures in the population in proportion to their relative fitness. The roulette wheel or rank selection methods are often used for the selection operator (Goldberg & Kuo, 1987). New structures are created by applying operators that mimic biological transformations, such as crossover and mutation.

A GA allows a population composed of many individuals to evolve under specified selection rules to a state that optimizes (maximizes or minimizes) the "fitness" function. In that process, crossover operator plays an important role. One of the problems in using genetic algorithms is the choice of crossover operator (Fang et al., 2014). The driving force behind GA is the unique cooperation between selection, crossover and mutation

operator. A genetic operator is a process used in GAs to maintain genetic diversity. The most widely used genetic operators are recombination, crossover and mutation (Fang et al., 2014). In this research, the crossover operator is applied to fuse the mCAFAC and the standard AFSA in order to determine the convergence to local minima. The pseudo code description of basic genetic algorithm is given as follows:

```
Basic Genetic Algorithms
Beginning (1)
    t = 0
    Initialization P(t)
    evaluation P(t)
    While (the stop condition is not verified) do
    Beginning (2)
        t = t + 1
        selection P(t) from P(t-1)
        P(t) ? crossover P(t)
        P(t) ? mutation P(t)
        P(t) ? replacement (P(t-1),P(t))
        evaluation P(t)
    Final(2)
Final(1)
```

Holland's analysis of the sources of power of genetic algorithms has served as guidance for the applications of genetic algorithms for years. Crossover is simply a matter of replacing some of the genes in one parent by the corresponding genes of the other. The technique of applying a recombination operator (crossover) to the population of artificial fish swarm algorithm in this research is vital in determining whether the algorithm falls into local optimum or not (Wu et al, 2011). It is important to note that crossover produces off-springs by recombining the information from two parents.

The crossover operator is applied to determine whether the modified (mCAFAC) falls into local optimum or not. It produced a new off-spring by recombination (fusion) of the information from mAFSA and mCAFAC using the following equation (Gao et al., 2010):

$$\left|\frac{f(X_i^t) - f(X_i^{t-1})}{f(X_i^{t-1})}\right| < 0.1 \tag{2.14}$$

When this condition is satisfied, the crossover operator in equation (2.15) is then applied to the i^{th} artificial fish $X_i (i = 1,2,..., N)$.

$$x_i = x_{r1} + \alpha \times (x_{r2} - x_{r1}) \tag{2.15}$$

Where x_{r1} and x_{r2} are two individuals selected randomly r_1 and r_2 are two integers which are generated randomly in the interval of $[1, N]$ and $r_1 \neq r_2 \neq i$. α Is a random number, uniformly distributed in the interval of $[-d, 1+d]$ and d is choosing as 0.25.

2.2.4 Particle swarm optimization

Particle Swarm Optimization (PSO) is a population based optimization approach which was inspired by observation of the behaviours in bird flocks and fish schools. PSO(Eberhart & Kennedy, 1995) is one of the modern heuristic algorithms. It was developed through simulation of a simplified social system, and has been found to be efficient in solving non-linear optimization problems. Unlike in the GA technique, the selection operation is not performed in PSO. All the particles in PSO are kept as members of the population through the course of a run (Toushmalani, 2013) (a run is defined as the total number of generation of the evolutionary algorithms prior to its termination). It is the velocity of the particle which is updated according to the previous best position of its companions and the particles fly with the updated velocities. The pseudo code representation of PSO is shown below:

```
Begin(1)
t=0;
Initialize pop(t)
evaluate Pi(t) %Pi(t) is the fitness of each particle
while (stopping condition is not verified) do
   if Pi(t) < Pbest(t)
      replace Pbest(t) with Pi(t);
   End
   if Pbest < Gbest
      take Pbest;
   else
      take Gbest;
   End
End
Update velocity
Update position
Repeat (until stopping condition is verified)
Final(1)
```

The idea that is used for the parameter selection is similar to inertial weight selection in Particle Swarm Optimization (PSO). In the first version of PSO, there was no actual control mechanism for the velocity of particles. This could result in inefficient behaviour of the algorithm, especially in the neighbourhood of the global minimum. Shi and Eberhart (1998) introduced a new way of addressing this shortcoming called inertial weight. One of the most widely used improvements is the introduction of inertia weight, which is employed to control the impact of the previous history of velocities on the current one (Arumugam & Rao, 2008). The Inertial weight (w) that is chosen for the purpose of this research is a linear value of iteration weight which is given as (Yazdani et al., 2013):

$$w = \left(\frac{itr_{\max} - itr}{itr_{\max}}\right)^{n} \tag{2.16}$$

Where $itr_{\max}$ correspond to the maximum iteration number and itr correspond to the current iteration number. This weight in equation (2.16) is then utilized to determine visual and step size trend to local or global solutions in the solution space.

$$visual_{itr} = visual \times w \tag{2.17}$$

$$step_{itr} = step \times w \tag{2.18}$$

Equations (2.17) and (2.18) are expected to make significant improvement to the precision accuracy of the modified algorithm.

2.2.5 Artificial bee colony optimization algorithm

Artificial Bee Colony (ABC) (Karaboga, 2005) algorithm is one of the most recently introduced swarm-based algorithms which was inspired by the intelligent foraging behaviour of honey bee. Similar to AFSA, a typical example of swarm optimization algorithm (SOA) is honey bees swarming around their hives.

Social insect colonies can be considered as dynamical system gathering information from environment and adjusting its behaviour in accordance to the information acquired (Singh, 2009). While gathering information and adjustment processes, individual insects do not perform all the tasks because of their specializations. Generally, all social insect colonies behave according to their own division of labours related to their morphology (Akay & Karaboga, 2012). Bee system consists of two essential components:

1. Food Source: To select a good food source, a forager bee evaluates some important properties such as: its closeness to the hive, richness of the energy, taste of its nectar, and the ease or difficulty of extracting this energy.
2. Foragers: Foraging is the swarm behaviour of bee that tend to the food source. Foragers are classified into three forms:

a. Unemployed foragers: A forager bee that looks for a food source to exploit is called unemployed. It can either be a scout who searches the environment randomly or an onlooker who tries to find a food source by means of the information given by the employed bee. The two possibilities for an unemployed forager are (Zhu & Kwong, 2010):

   i. Scout Bee: If the bee starts searching spontaneously without any knowledge, it is a scout bee. The main population of scout bees varies from 5% to 30% (Karaboga, 2009) according to the information in the nest.

   ii. Recruit: If the unemployed (Onlooker) forager responds to a waggle dance done by some other bee, the bee will start searching by using the knowledge obtained from the waggle dance.

b. Employed foragers:When the recruit bee finds and exploits the food source, it will raise to be an employed forager who memorizes the location of the food source. An employed bee caries information about a specific food source and shares it with other bees waiting in the hive. The information includes the distance, the direction and the profitability of the food source (Karaboga, 2005). After the employed foraging bee loads a portion of nectar from the food source, it returns to the hive and unloads the nectar to the food area in the hive. There are three possible options related to residual amount of nectar for the foraging bee (Ozturk et al., 2015).

   i. If the nectar amount decreased to a low level or exhausted, foraging bee abandons the food source and become an unemployed bee.

   ii. If there are still sufficient amount of nectar in the food source, it can continue to forage without sharing the food source information with the nest mates.

iii. It can go to the dance area to perform waggle dance to inform the nest mates about the same food source.

c. Experienced foragers:These types of forager use their historical memories for the location and quality of food sources. It can be any of the following (Akay & Karaboga, 2012):

i. Inspector forager: which controls the recent status of the food source already discovered

ii. Reactivated forager: which makes use of the information from waggle dance and tries to explore the same food source discovered by it, if there are some other bees confirming the quality of the same food source

iii. It can be scout bee which searches for new patches if the whole food source is exhausted.

iv. It can be a recruit bee which is searching for a new food source declared in dancing area by another employed bee.

In artificial bee colony algorithm (ABC), the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution (Akay & Karaboga, 2012). The number of the employed bees or the onlooker bees is equal to the number of solutions in the population. At the first step, the ABC generates a randomly distributed initial population $P(C=0)$ of N solutions (food source positions), where N denotes the size of employed bees or onlooker bees. Each solution $x_i(i=1,2,...,N)$ is a D-dimensional vector. Here, D is the number of optimization parameters.

After initialization, the population of the positions (solutions) is subject to repeated cycles, $C=1,2,\ldots,MCN$ of the search processes of the employed bees, the onlooker bees and the scout bees.

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source p_i using the following equation (Karaboga & Basturk, 2007):

$$p_i = \frac{fit_i}{\sum_{n=1}^{N} fit_n} \tag{2.19}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i and N is the number of food sources which is equal to the number of employed bees or onlooker bees.

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression (Karaboga & Basturk, 2007).

$$v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{ik}) \tag{2.20}$$

Where $k \in \{1,2,...,N\}$ and $j \in \{1,2,...,D\}$ are chosen randomly and ϕ_{ij} is a random number between [-1 1].

Assume that the abandoned food source is x_i and $j \in \{1,2,...,D\}$, then the scout bee discovers a new food source to be replaced with x_i as follows(Karaboga, 2005):

$$x_i^j = x_{\min}^j + rand(0,1) \times (x_{\max}^j - x_{\min}^j) \tag{2.21}$$

After each candidate source position v_{ij} is produced and then evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source has an equal or better nectar than the old source, it is replaced with the old one in the memory; otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one (Singh, 2009). Figure 2.3 shows the Flowchart representation of ABC whilethe pseudo code representation of ABCis as follows.

```
%Pseudo Code Representation of ABC
Initialize parameters
Initialize Employed Bee Colony (Population)
Evaluate population
Cycle=1;
Repeat
Generate a new food source for the employed bee
    For i=1: N
        Produce a new food source Vi for the employed bee
        Apply greedy selection process between Vi and Xi
        If solution Xi does not improve triali= triali+1,
        Otherwise triali=0
    End
    Calculate the probability value pi
    Memorize the best solution achieved so far.
    Cycle=cycle+1;
    Until cycle=MCN
```

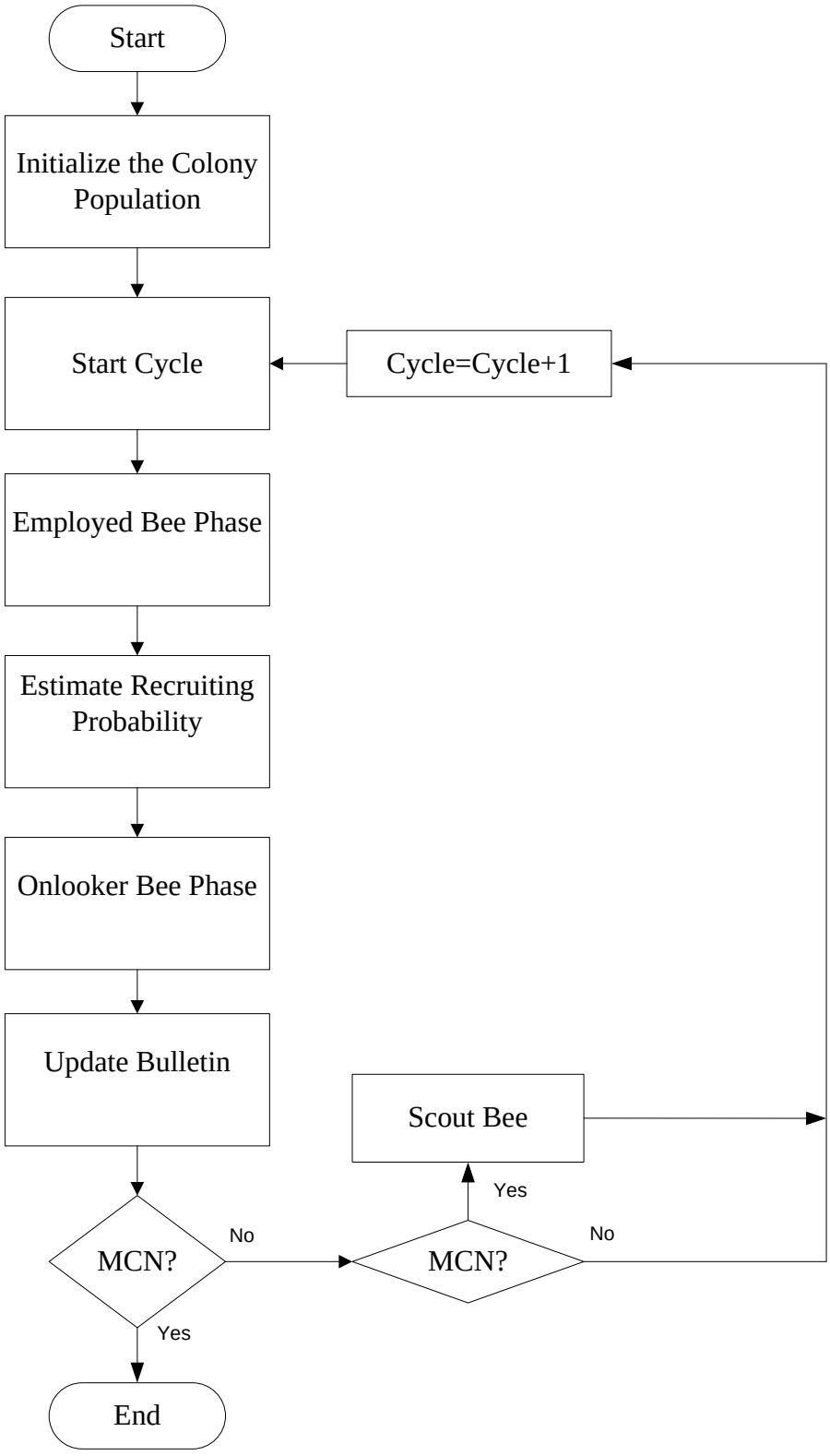


Figure 2.3: Flowchart of Artificial Bee Colony Algorithm

2.2.6 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).

The performance indices include:

i. Convergence speed.
ii. Robustness.
iii. Precision accuracy.
iv. General performance.

There are many test functions reported in the literature, but there is no standard list or set of test functions one has to follow. 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 type of optimization problems efficiently (Tang et al., 2007).

In this research, a total of sixteen (16) multi-modal nonlinear test functions were used to investigate the optimization capability of mCAFAC. All these functions have a lot of local minima around the global optimal. The test functions considered in this research are described in the following sub-sections.

2.2.6.1 Ackley function

This function is one of the commonly used test function which has a lot of local minima around the global minima. This test function is expressed as follows (Wu et al., 2011):

$$f_{Ackley}(x) = -20\exp[-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}] - \exp[\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)] + 20 + e \qquad (2.22)$$

Where $n = 1,2..., and -32.768 \le x_i \le 32.768$ for $i = 1,2,\ldots,n$.

This function has the global minimum $f_* = 0;$ at $x_* = (0,0,...,0)$ (Jamil & Yang, 2013)

2.2.6.2 Cosine-Mixture function

The Cosine Mixture (CM) function is a multimodal optimization test function described using the following (Wu et al., 2011):

$$f(x) = (-0.1) \times \sum_{i=1}^{n} \cos(5\pi x_i) + \sum_{i=1}^{n} x_i^2 \tag{2.23}$$

Where $n = 1,2,\ldots,$ and $i = 1,2,\ldots,n$ with a global minimal of -3 for $f_* = -3$ at $x_* = (0,0,\ldots 0)$ (Jamil & Yang, 2013)

2.2.6.3 DeJongF4 function

Dejong 4th-order function is represented as follows (Wu et al., 2011):

$$f(x) = \sum_{i=1}^{n} i \times x_i^4 \tag{2.24}$$

$$-5.12 \le x_i \le 5.12$$

This function has the global minimal $f_* = 0$ at $x_* = (0,0,\ldots 0)$ (Ben-Tal & Nemirovski, 2001). DeJong4 function is unimodal and convex.

2.2.6.4 Expfun function

Expfun is often described as the simplest form of Dejong function, which is described as follows (Wu et al., 2011):

$$f(x) = \exp(-0.5 \times \sum_{i=1}^{n} x_i^2) \tag{2.25}$$

This function is unimodal and has the global minimal of $f_* = 1$ at $x_* = (0,0,\ldots 0)$ (Jamil & Yang, 2013).

2.2.6.5 Griewangk function

Griewangk's function is similar to Rastrigin's function. It has many widespread local minima (Hansen, 2006). However, the locations of the minima are regularly distributed

$$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1 \qquad (2.26)$$

Where $n = 1,2,\ldots,$ and $i = 1,2,\ldots,n$ for $-600 \le x_i \le 600$.

This function has a global minimum of $f_* = 0$ at $x_* = (0,0,\ldots 0)$ (Tang et al., 2007)

2.2.6.6 Hyperelliptic function

The axis parallel hyper-ellipsoid is similar to De Jong's function. It is also known as the weighted sphere model. It is continuous, convex and uni-modal (Wu et al., 2011):

$$f(x) = \sum_{i=1}^{n} i^2 x_i^2 \qquad (2.27)$$

This function has global minimal $f_* = 0$ at $x_* = (0,0,\ldots, 0)$ (Li et al., 2013)

Where $n = 1,2,\ldots,$ and $i = 1,2,\ldots,n$ for $-5.12 \le x_i \le 5.12$.

2.2.6.7 Levy and Montalvo first function

Levy and Montalvo (LM1) is a multimodal optimization test function which has been widely used to verify the performance of optimization algorithm. This function is represented as follows (Jamil & Yang, 2013):

$$f(x) = \frac{\pi}{n}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + \sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} \qquad (2.28)$$

This function has the same global minimum which is $f_* = 0$ at $x_* = (0,0,...,0)$ (Jamil & Yang, 2013):

2.2.6.8 Levy and Montalvo second function

This is another Levy and Montalvo (LM2) optimization test function which is expressed as follows (Jamil & Yang, 2013):

$$f(x) = 0.1\left(\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_{i+1})\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right) \quad (2.29)$$

This function has the same global minimum as LM1 which is $f_* = 0$ at $x_* = (0,0,...,0)$ (Jamil & Yang, 2013):

2.2.6.9 Neumaier3 function

The Neumaier3 function is expressed as (Wu et al., 2011):

$$f(x) = \sum_{i=1}^{n}(x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1} \quad (2.30)$$

This function has a lot of local minima around the global minima with a global solution of $f_* = -4930$ at $x_* = (0,0,...,0)$ (Wu et al., 2011).

2.2.6.10 Rastrigin function

Rastrigin's function is based on ExpFun with the addition of cosine modulation to produce many local minima. Thus, the test function is highly multimodal. However, the locations of the minima are regularly distributed. The mathematical expression for this function is as follows (Wu et al., 2011):

$$f(x) = 10n + \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i)], \quad (2.31)$$

This function is highly multi-modal and has global minimum $f_* = 0$ at $x_* = (0,0,...,0)$ with a constraint of $-5.12 \le x_i \le 5.12$ (Jamil & Yang, 2013).

2.2.6.11 Rosenbrock function

The Rosenbrock's valley is a classic optimization problem, also known as Banana function (Tang et al., 2007). The global optimum is inside a long, narrow, parabolic shaped flat valley. To find the valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used in assessing the performance of optimization algorithms (Wu et al., 2011).

$$f(x) = \sum_{i=1}^{29} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (2.32)$$

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 $i = 1,2,\ldots,n-1$. In the 2D case, it is often written as(Jamil & Yang, 2013)

$$f(x,y) = (x-1)^2 + 100(y - x^2)^2 \quad (2.33)$$

2.2.6.12 Sal Function

Sal function is another multi-modal optimization test function considered in this research. The function is described as follows (Wu et al., 2011):

$$f(x) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{30} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{30} x_i^2} \quad (2.34)$$

The Sal function has the global minimum $f_* = 0$ at $x_* = (0,0,...,0)$ in the domain $-2\pi \leq x_i \leq 2\pi$ (Li et al., 2013):

2.2.6.13 Schwefel function

Schwefel's function is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction (Tang et al., 2007).

$$f(x) = 418.9829 \times n + \sum_{i=1}^{n} (-x_i \times \sin(\sqrt{|x_i|})) \quad (2.35)$$

This function has global minimum $f_* \approx -418.9829\,n$ occurs at $x_i = 420.9687$ in the domain of $-500 \leq x_i \leq 500$. Where $n = 1,2,\ldots,$ and $i = 1,2,\ldots,n$ (Tang et al., 2007):

2.2.6.14 Schaffer function

Schaffer is a multimodal non-convex optimization test function expressed in the following equation (Wu et al., 2011):

$$f(x) = \sum_{i=1}^{30} (x_i^2 + x_{i+1}^2)^{0.25} \{[\sin 50(x_i^2 + x_{i+1}^2)^{0.1}]^2 + 1\} \quad (2.36)$$

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

2.2.6.15 Sphere function

This is the simplest form of DeJong function. This function is unimodal and convex with an obvious local minimum of $f_* = 0$ at $x_* = (0,0,...,0)$ in a domain of $-5.12 \le x_i \le 5.12$ (Wu et al., 2011).

$$f(x) = \sum_{i=1}^{30} x_i^2 \tag{2.37}$$

2.2.6.16 Bukin N.6 function

The Bukin N.6 function is a multi-modal test function with several local minima around the global minima

The mathematical expression describing this function is as follows(Hansen, 2006):

$$f(x,y) = 100\sqrt{\left|y - 0.01x^2\right|} + 0.01\left|x + 10\right| \tag{2.38}$$

Bukin function has a global optimal of $f(-10,1) = 0$ with the constraint of $-15 \le x \le -5$ and $-3 \le y \le 3$(Hansen, 2006).

2.2.7 Determination of optimal values of weighting matrices in LQR problems

The Linear Quadruple Regulator (LQR) is an optimal method used to design linear controllers for linear systems. 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.

Thus using intelligent optimization methods, like the mCAFAC, for finding Q and R is one of the objectives of this research. The LQR scheme has been used in the control of induction motors (Wang et al., 2014), vehicular drive-shaft control (Changhao & Duan, 2013), Inverted Pendulum Stabilization (Boubaker, 2013; Hasan et al., 2012) etc. Recently, researches have been performed for LQR controller design using Computational Intelligence (CI) method (Boubaker, 2013; Boubaker & M'sahli, 2008; Das et al., 2013; Ghoreishi et al., 2011; Hamidi, 2012; Hasan et al., 2012; Saberi et al., 2012).

The Linear Quadruple Regulator (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 quadruple 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)

$$\dot{X} = Ax + Bu \tag{2.39}$$

$$y = Cx + Du \tag{2.40}$$

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, and C 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):

$$u = -Kx \tag{2.41}$$

Putting equation (2.41) into equation (2.39), the closed-loop system becomes:

$$\dot{X} = (A - BK)x \tag{2.42}$$

and the eigenvalues are obtained from:

$$\det(\lambda I - A + BK) = 0 \tag{2.43}$$

with the appropriate feedback gain (K), the closed loop poles can be shifted to the desired locations. The optimal control law indicated in equation (2.41) minimizes the quadruple performance index (cost function) which consists of state and control energies as follows (Ghoreishi et al., 2011):

$$J = \frac{1}{2}\int_0^{\infty} (x^T Qx + u^T Ru\, dt \tag{2.44}$$

Where $Q \in \Re^{n\times n}$ is a positive, semi-definite and symmetric real matrix $Q \geq 0$, and $R \in \Re^{m\times m}$ is a positive, definite and symmetric control weighting matrix $(R > 0)$. A feature of optimal controllers (such as the LQR) is that there 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)

$$K = R^{-1}B^T P \tag{2.45}$$

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

$$AP + A^T P + Q - PBR^{-1}B^T P = 0 \tag{2.46}$$

Equation (2.46) shows that the choice of Q and R will influence P, the solution of the algebraic Riccati equation.

2.2.7.1 Design choice for determination of LQR weighting matrices

The LQR method guarantees robustness, but only allows pole placement in a specific region, and the poles that give the desired performance may or may not be in these regions (Ghoreishi et al., 2011). Therefore it may not be possible to use this method to achieve both robustness and exact pole placement.

For a controllable and observable LTI system, the designer has the freedom to place the pole at any location in the left hand side of the complex plane (Hasan et al., 2012). The idea used in this research is to first establish that the system is controllable and observable, such that the poles positions can be assigned.

A system is completely state controllable if any initial state x_0, can be shifted to any final state x_t in a finite time. In other words, a feedback matrix exists that can shift a closed loop pole to any desired location. Controllability for the linear system given is determined by forming the controllability matrix (Cuong et al., 2015).

$$M_c = [A \mid AB \mid A^2B \mid \ldots \mid A^{n-1}B] \tag{2.47}$$

Where n is the number of states. The system is controllable if the rank of M_c, is n. Ifthe system is not of full rank, n-rank (M_c) gives the number of poles that cannot be moved by feedback (Das et al., 2013).

A system is said to be completely state observable if all the states x can be determined from measuring the outputs y. If a state is not observable, it does not influence the output and cannot be identified. For the linear system given, the observability matrix (Cuong et al., 2015) is:

$$M_o = [C^T \mid A^TC^T \mid (A^T)^2 \mid \ldots \mid (A^T)^{n-1}C^T] \tag{2.48}$$

can be formed to check if the system is completely observable. Ifthe rank of M_o equals n, the number of states, then the system is completely observable. For an unobservable system, the number modes that can be observed is equal to n-rank (M_o).

A system is detectable if all the unstable (right half plane) poles are observable, and a system is stabilizable if all the unstable (right half plane) poles are controllable.

2.3 REVIEW OF SIMILAR WORKS

In this subsection, most of the relevant literatures directly related to this work were presented. This literatures includes the previous research works on, AFSA, ABC, PSO, GA, CA and optimal design of LQR Controllers.

(Ma & Zhang, 2005) presented an artificial fish swarm neural network in short term forecasting of power systems. In this work, a new artificial neural network (ANN) based forecasting model using AFSA is built in which the weights of ANN are trained by AFSA, then the neural network using AFSA is applied to short term load forecasting. Applying the presented forecasting method to a certain actual power network it is shown that comparing with traditional BP neural network forecasting method the presented forecasting method has better adaptive ability and can give better forecasting result. However, the convergence speed and precision accuracy of AFSA were not considered.

(Mingyan & Dongfeng, 2005) evaluated wavelet optimization by introducing a new leaping behaviour into the AFA which improved the probability to leap out of local extremes. This allowed each AF to select a proper behaviour to execute during evolution without wasting much time on evaluating others. Selecting a suitable parameter (visual and step) during the evolution of AFA was not considered.

(Huang *et al.*, 2006) presented an optimization approach to the location of oilfield multistage stations by fish-swarm algorithm. The continuous real variables in the large-scale Mix Integer Programing (MIP) model for locating optimization of oilfield multistage station model are changed into discrete 0-1 variables so that the nonlinear MIP model is transferred into a pure

0-1 nonlinear IP model in order to provide the convergence using fish swarm algorithm. The application showed that the computing speed and stability of the algorithm are higher, and the optimum solution of the model can be gained on microcomputers. However they did not consider the effect of parameter setting during the evolution of AFSA

(Wang *et al.*, 2008) presented a Quality of Service QoS multicast routing scheme based on TS (Tabu Search) and AFS (Artificial Fish Swarm). The fast searching ability of AFS and global optimization ability of TS was combined to find the best multicast tree with maximum probability of meeting QoS constraints under the given cost and inaccurate QoS information. However, they did not consider the effect of step size and visual distance during the evolution of AFSA.

(Huang *et al.*, 2008) presented a cultural-based particle swarm optimization (CBPSO) which consists of the population space and the belief space. Numerical result shows that CBPSO enhances the global searching ability and has better optimization performance than PSO. Only two cultural influence function considered to model the CBPSO in their research.

(Cheng *et al.*, 2009) proposed a novel clustering algorithm based on improved artificial fish swarm algorithm. In their work, the information of global best AF was added to the behaviours of the initial population, which improved the performance of the AFSA. However, they did not consider the effect of step size and visual distance during the evolution of AFSA.

However, in the course of treatment of multi-peak and the large search space function optimization problems, it has been discovered that due to slow convergence speed of AFSA,

it was easy for it to fall into local optimal solution. At the same time, the experiences of group members are not used for the next moves. In order to improve the optimization performance of the AFSA, many researchers have proposed some improvements on the algorithm.

(Yazdani *et al.*, 2010)proposed a new hybrid algorithm for optimization in continuous and static environments by hybridizing cellular learning automata and artificial fish swarm algorithm. In the proposed algorithm, each dimension of search space was assigned to one cell of cellular learning automata and each cell a swarm of AF was located which have the optimization duty of that specific dimension. The learning automata in each cell was responsible for making diversity in artificial fish's swarm of that dimension and equivalence between global search and local search processes. Experimental results showed that the proposed method had an improved acceptable performance over the standard AFSA. However the effect of group members and parameter selection in AFSA was not considered.

(Zhu & Jiang, 2010)proposed a novel quantum artificial fish swarm algorithm based on the principles of quantum computing, such as the quantum bit and quantum gate. The position of the Artificial Fish (AF) was encoded by the angle in $[0,2\pi]$ based on the qubit's polar coordinate representation in the 2-dimension Hilbert space. The quantum rotation gate was used to update the position of the AF in order to enable the AF to move and the quantum non-gate is employed to realize the mutation of the AF for the purpose of speeding up the convergence. The experimental results proved that the performance of QAFSA is significantly improved compared with that of standard AFSA. However, they did not consider the effect of step size and visual distance during the evolution of AFSA.

(Jiang & Cheng, 2010)reported a novel stochastic approach called simulated annealing-artificial fish swarm algorithm (SA-AFSA) for solving some multimodal optimization problem. This approach obtained much better optimization precision and convergence speed compared with the basic artificial fish swarm algorithm. The hybrid algorithm has the strong local search ability of simulated annealing and the swarm intelligence of AFSA. The individual behaviour of Fish which could cause the AFSA to fall into local minima easily was not considered in this research.

(Wu et al., 2011)proposed a knowledge-based artificial fish swarm algorithm. Both situational and normative knowledge in cultural algorithm was used to direct the step size as well as the evolution of the AFSA. A crossover operator was used to increase the diversification of the artificial fishes based on their parents' characteristics and also to determine whether the algorithm fell into local optima or not. However, they did not consider the effect of step size and visual distance during the evolution of AFSA.

(Guo *et al.*, 2011) reported a new stochastic global optimization algorithm. In this work, chaos optimization was first employed to initialize the position of individual artificial fish and then AFSA was applied to obtain the neighbourhood of the global optimum solution. When there is no change or little change of the function values on bulletin board in successive iterations, chaotic mutation was then executed to help the artificial fish swarm get rid of the local optimum. The algorithm was applied in optimizing cascade hydropower station. The effect of group movement and parameter selection were not considered.

(Jiang *et al.*, 2012) reported a hybrid algorithm using a hybrid of artificial fish swarm algorithm and particle swarm optimization. The following and swarming behaviour of AFSA

are performed on two sub-swarms simultaneously. There proposed algorithm increased the variety of the population and improved the accuracy of the solution. However, the effect of both visual and step size on the evolution of AFSA were not considered.

(De Oliveira & Ludermir, 2012) proposed a Fuzzy Artificial Fish Ensemble Extreme Learning machine. The algorithm was employed for solving Extreme Learning Machine Ensembles. The ensemble selection was performed by the use of a hierarchical clustering algorithm, and two diversity measures. Experimental results showed that the proposed method shows an improvement. The proposed method was not verified using more complex multi-dimensional NP-hard problem and the effect of Visual distance and step size was not considered.

(Hanafy *et al.*, 2012)proposed a development of a technology for car's auto-parking using swarm search-based fuzzy control system. The technique was divided into two main stages. The first stage is the generation of the optimal path trajectory that the car can follow so that it can be parked in a parking lot with the minimum acceptable dimensions, after the discovery of the parking place. The second stage is the path tracking problem, which was solved by the fuzzy control system. The parameters of the fuzzy controller beside the required time for parking achievement all are tuned by artificial fish swarm algorithm (AFSA). The performance of the involved swarm-based fuzzy controller (SBFC) was measured for solving the path tracking problem while operating on the dynamic model of a car-like robot, so that a full practical technique for car's auto parking can be realised. However, the effect of both visual distance and step size on the evolution of AFSA was not considered.

(Zhou *et al.*, 2013) proposed a novel hybrid glow-worm swarm optimization (HGSO). The algorithm embedded predatory behaviour of artificial fish swarm algorithm (AFSA) into glow-worm swarm optimization (GSO) algorithm and combined the improved GSO with differential evolution (DE) on the basis of a two-population co-evolution mechanism. In addition, to overcome premature convergence, the local search strategy based on simulated annealing (SA) was used. However, the effect of both visual and step size on the evolution of AFSA was not considered.

(Yanmin & Zhibin, 2013) proposed a method based on artificial fish swarm algorithm to better solve the optimization problem of fuzzy neural network (FNN). This method was aimed at the structure optimization problem of FNN, AFSA-FNN1 was established and realizes the simplification of fuzzy rules. Aiming at the parameter optimization problem of FNN, AFSA-FNN2 was built and realizes the acquisition of parameters of membership function (MF) automatically. The proposed method used for path planning of the robot, simulation results showed that the optimized FNN can enhance the smoothness of the path better, than the standard AFSA. In this research, the slow convergence speed of AFSA was not considered.

(Chen *et al.*, 2013) developed a quantitative model for battery Remaining Useful Life (RUL) prediction using an adaptive bathtub-shaped function (ABF). ABF was utilised to model the normalised battery cycle capacity prognostic curves, which attempted to predict the remaining battery capacity with given historical test data. An artificial fish swarm algorithm method with a variable population size (AFSAVP) was employed as the optimiser for the parameter determination of the ABF curves, in which the fitness function is defined in the form of a coefficient of determination (R2). A 4 × 2 cross-validation (CV) was devised, and

the results showed that the method can work valuably for battery health management and battery life prediction. However they did not consider the effect of step size and visual distance of AFSA.

(Fang et al., 2014) presented a hybrid algorithm for solving short-term hydrothermal scheduling (SHS) by combining real coded genetic algorithm and artificial fish swarm optimization (RCGA-AFSA). RCGA was applied as global search, which can explore more promising solution space and give a good direction to the global optimal region, and AFSA was applied as local search to obtain the final optimal solution for improving the exploitation capability of algorithm. However they did not consider the effect of step size and visual distance of AFSA.

(Xue, 2014) proposed an adaptive hybrid artificial fish swarm algorithm (AHAFSA), which used the Hooke-Jeeves pattern search method as a local search operator embedded in artificial fish swarm algorithm (AFSA) to speed up the local search. AHAFSA was used to solve the real roots of polynomials. The numerical experiment results showed that the AHAFSA can not only effectively locate the global optimum, but also have a rather high convergence speed. Although, he was able to improve on the convergence speed of the AFSA, but, the effect of individual parameter selection during evolution was not considered.

(Tao & Han, 2014)investigated area coverage enhancement by directional sensors with tuneable sensing orientations. Deployment of directional sensors as a 2D stationary Poisson point process was modelled, and the relationship between the coverage probability and the number of directional sensors was then evaluated. The notion of "sensing centroid," was introduced which is the geometric centre of a sensing sector to simplify the pending problem.

The "sensing centroid" was referred as artificial fish, which searched an optimal solution in the solution space by simulating fish swarm behaviours with a tendency towards high food consistence. However, they did not consider the effect of step size and visual distance in AFSA.

(Wang & Li, 2015)proposed an improved search intelligent algorithm based on the group search algorithm and artificial fish swarm optimization. These two algorithms were integrated to form a new improved algorithm called the Group Search Artificial Fish Swarm Optimization (GS-AFSA). The algorithm was used to optimize 10-bar planar truss structure, 15-bar planar truss structure and 72-bar planar truss structure. They showed that GS-AFSA outperformed both GS and AFSA. However, the effect of step size and visual distance during the evolution of AFSA was not considered.

(Hassan et al., 2015)proposed a community detection algorithm in complex networks based on Artificial Fish Swarm Algorithm. The AFSA was used to automatically determine the number of communities. Experimental results showed that the AFSA can successfully find an optimized community structure based on the quality function used. In the research, they did not consider the effect of step size and visual distance on the evolution of AFSA.

Due to the profound effect of Linear Quadruple Regulator (LQR) controller weighting matrices Q and R on the controller performance, many researches have proposed an intelligent way of determining these matrices (Q and R) using Computational Intelligent algorithms. This method has been applied to stabilization problems such as that of the inverted pendulum.

(Ata & Coban, 2015) proposed a linear quadruple optimal controller design for a nonlinear inverted pendulum. The LQR was used to control a single bulb inverted pendulum as a nonlinear dynamic system and the Artificial Bee Colony (ABC) algorithm was used for selecting the weighting matrices (Q and R) to overcome the design difficulty of LQR design. The result obtained showed that the proposed method performed better than the conventional (trial and error) method. In the research, they did not extend their method to inverted pendulum of higher degree of freedom.

It is evident from the literatures that, improving the convergence speed and optimization precision has been given a significant research attention. However, developing an algorithm that will outperform other optimization algorithm in terms of global optimization result is still a serious subject of concern for researchers. It is on this evidence that this research was inspired. The research is an extension of Wu et al (2011), as they did not consider the effect of such parameters visual distance, step and population size during the generation of AFSA. Similarly, their work was parameter specific, for example, population size was selected as 60 which will not be the case in all optimization problems. In this research, an inertial weight selection procedure is introduced such that every generation will have its own suitable visual distance and step size. Other parameters, such as population size, problem dimension, degree of crowdedness, acceptance percentage, visual distance and step size were made user dependent. The proposed algorithm is used to determine the weighting matrices of LQR controller which is then used to stabilize the Quadruple Inverted Pendulum

CHAPTER THREE

MATERIALS ANDMETHODS

3.1 Introduction

In this chapter, the detailed procedures for the successful completion of this research are discussed and four variants of improved mCAFAC were developed.

3.2 Modified artificial fish swarm algorithm

In this research, an inertial weight is applied as an adaptive parameter in order to improve the AFSA. The improved AFSA is then used to initialize the population of mCAFAC. The following equations describe the modifications to the behaviours (Preying, Swarming and Chasing) of AFSA.

3.2.1 Modified preying behaviour

Suppose the current state of AF is X_i, the modified artificial fish select a state randomly within its visual distance such that:

$$X_j = X_i + rand(0,1) \times visual \times (\frac{itr_{\max} - itr}{itr_{\max}})^n \quad (3.1)$$

Where X_j is the new state and X_i is the previous state.

If $f(X_j) < f(X_i)$ in the minimum problem, it goes forward a step towards X_j in the following direction.

$$X_i^{(t+1)} = X_i^{(t)} + rand(0,1) \times step \times (\frac{itr_{\max} - itr}{itr_{\max}})^n \times \frac{X_j^{(t)} - X_i^{(t)}}{\left\| X_j^{(t)} - X_i^{(t)} \right\|} \quad (3.2)$$

If $f(X_j) > f(X_i)$, the artificial fish selects another state randomly again. If the artificial fish cannot meet the requirement in a given time, it moves one step randomly as:

$$X_i^{t+1} = X_i^t + rand(0,1) \times step \times (\frac{itr_{\max} - itr}{itr_{\max}})^n \quad (3.3)$$

3.2.2 Modified swarming behaviour

The swarming behaviour of AFSA described in section two is modified as follows:

$$X_i^{(t+1)} = X_i^{(t)} + rand(0,1) \times step \times (\frac{itr_{\max} - itr}{itr_{\max}})^n \times \frac{X_c^{(t)} - X_i^{(t)}}{\left\| X_c^{(t)} - X_i^{(t)} \right\|} \quad (3.4)$$

All other information are as described in subsection 2.2.1.2

3.2.3 Modified chasing behaviour

The swarming behaviour of AFSA described in section two is modified as follows:

$$X_i^{(t+1)} = X_i^{(t)} + rand(0,1) \times step \times (\frac{itr_{\max} - itr}{itr_{\max}})^n \times \frac{X_m^{(t)} - X_i^{(t)}}{\left\| X_m^{(t)} - X_i^{(t)} \right\|} \quad (3.5)$$

Otherwise it executes the behaviour of preying.

The complete code for the modified AFSA algorithm is given in Appendix A1 and the code for the ABC algorithm is given in Appendix A2.

Figure 3.1 shows the flowchart implementation for the modified mAFSA

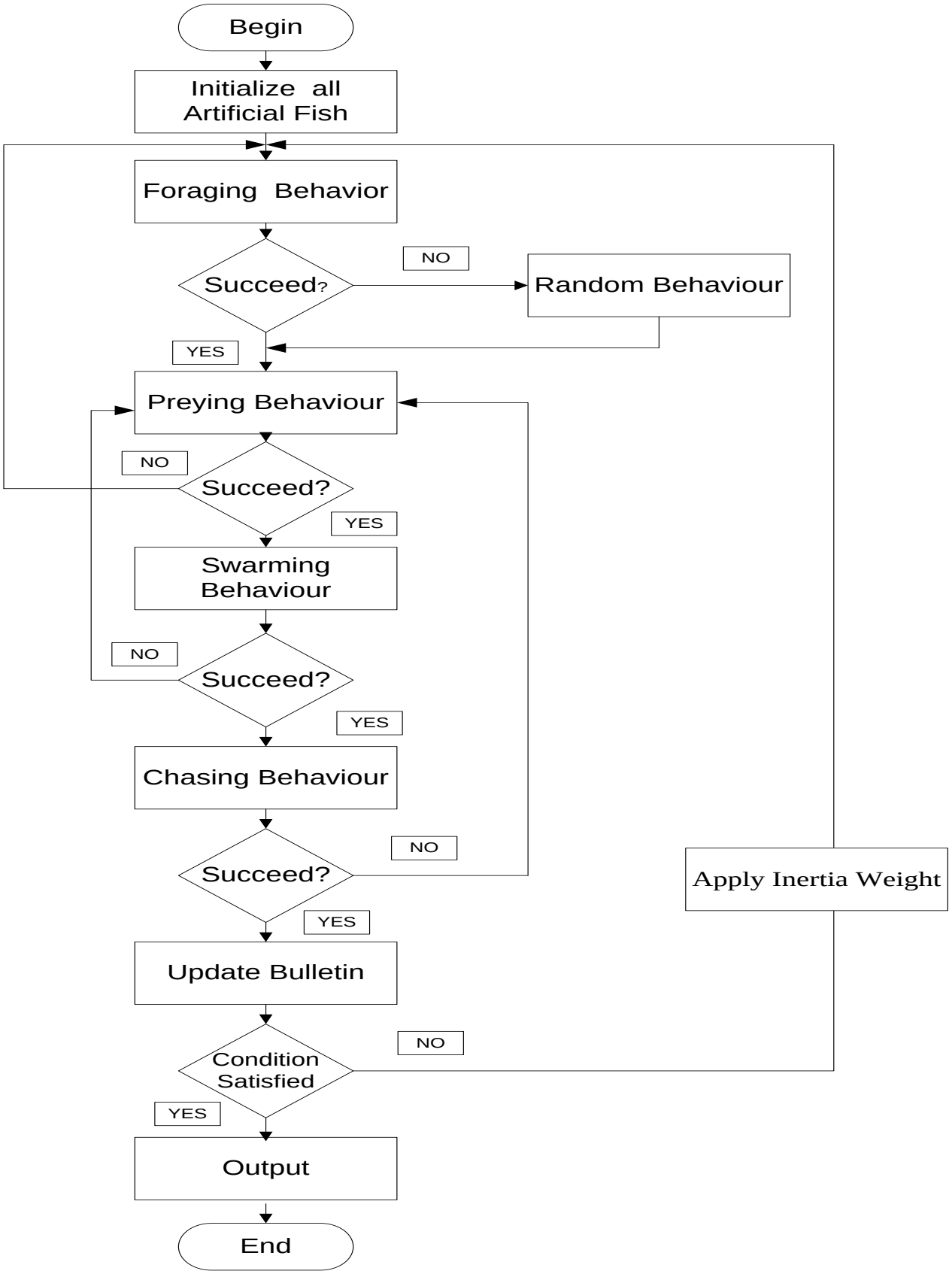


Figure 3.1: Flowchart of Modified Artificial Fish Swarm Algorithm

3.3 Modified Cultural Artificial Fish Swarm Algorithm with Crossover

Here, the result of the fish swarm is regarded as the population space, where the domain knowledge which was used to initialize the population of mCAFAC is extracted from. Therefore, the domain knowledge is formed and stored in belief space so as to model and impact the evolution of the population at generation. Four Versions mCAFACs (mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd and mCAFAC_NsNd) was developed using the different influence function.

3.3.1 mCAFAC Knowledge adjustment

In cultural algorithm, each knowledge found in the belief space has its own update rule. The update rule (function) for situational and normative knowledge are described in Sections 3.3.1.1 and 3.3.1.2 respectively.

3.3.1.1 Situational knowledge adjustment in belief space

The formal syntax of the belief space, is a pair structure, $< S, N_{[n]} >$. The situational knowledge is used to create a version of mCAFAC, which consists of the best current individual found so far and this is updated as described in section 2.2.2.3

3.3.1.2 Normative knowledge adjustment in belief space

Here, the normative knowledge provides more information about a set of feasible solution of the optimization problem. It is a set of information about each variables and is updated as described in subsection 2.2.2.2

3.4 Design Choices for the mCAFAC Influence Function

The belief space can influence the population in three different ways (Wu et al., 2011).

1. Determining the step size of the evolution.
2. Determining the direction of the evolution.
3. Determining the visual distance of the AFA.

The shape of the influence rule will depend on the type of knowledge maintained in the belief space, whether it is normative, situational, or both. The influences considered in this research are the direction of evolution, step size and visual distance. Each mCAFAC system influenced by belief space will now be described as follows:

3.4.1 mCAFAC_Ns: mCAFAC Using normative knowledge

In this variant, the simplest form of cultural influence, which employs only the normative knowledge to determine the step size of evolution and visual distance in AFSA was employed.

3.4.1.1 Influence function for preying mCAFAC_Ns

In this variant, the situational knowledge was used to guide the preying behaviour of AFSA. A selection operator was first introduced to initialize the mCAFAC, this operator is given as follows (Wu, et al 2011):

$$X_i^{t+1} = X_i^t + rand(0,1) \times size(I_i) \tag{3.6}$$

Where $size(I_i) = u_i - l_i$ is the size of the belief space interval for parameter i, which is decided by the normative knowledge for the i^{th} variable (Chung, 1997) and $rand(0,1)$ is a normally distributed random variable with a mean of zero and a standard deviation of 1

(Jones et al., 1995).

The mathematical expression describing the preying behaviour of mCAFAC_Ns is given as follows:

$$x_{ik}^{t+1} = x_{ik}^{t} + \frac{size(I_k) \times rand(0,1) \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_i^{t+1} - X_i^{t} \right\|} \tag{3.7}$$

Where $size(I_k) = u_k - l_k$ is the size of the belief space interval for parameter k which is decided by the normative knowledge for the k[th]variable. The other parameters are described in Section 2.2.1.1 Equation (3.7) is the influence function for preying mCAFAC when the i^{th} individual generation is less or equal to the maximum number of trial ($i \le try_num$).

When the i^{th} individual generation is greater than maximum number of trial $i > try_num$, the preying behaviour of mCAFAC_Ns is influenced as follows (Wu et al 2011):

$$x_{ik}^{t+1} = x_{ik}^{t} + size(I_k) \times rand(0,1) \tag{3.8}$$

Equations (3.7) and (3.8) are to ensure the evolution is guided towards the best solutions.

3.4.1.2 Influence function for swarming mCAFAC_Ns

The swarming behaviour of AFSA is guided using the simplest form of cultural influence, in this case only the normative knowledge is used in order to determine the step size of evolution and visual distance in AFSA.

$$x_{ik}^{t+1} = x_{ik}^{t} + \frac{size(I_k) \times rand(0,1) \times (x_{ck}^{t} - x_{ik}^{t})}{\left\| X_c^{t+1} - X_i^{t} \right\|} \tag{3.9}$$

Where $X_c = \sum_{j=1}^{nf} X_j / nf$ the centre position as is discussed in section 2.2.1.2 and nf is the number of its fellow within the visual distance, x_{ck} is the centre position associated with the upper bound for variable *j*.

3.4.1.3 Influence function for chasing mCAFAC_Ns

The chasing behaviour of AFSA is modelled using the normative knowledge to determine the step size of evolution and visual distance as follows:

$$x_{ik}^{t+1} = x_{ik}^{t} + \frac{size(I_k) \times rand(0,1) \times (x_{mk}^{t} - x_{ik}^{t})}{\left\| X_m^t - X_i^t \right\|} \tag{3.10}$$

Where X_m stands for the best artificial fish individual within X_i's visual distance as discussed in section 2.2.1.3. The program for this variant (mCAFAC) can be found in Appendix B1

3.4.2 mCAFAC_Sd: mCAFAC using situational knowledge

mCAFAC_Sd is another variant of mCAFAC, where the situational knowledge of the current exemplar or best solution found so far, was used to decide the direction of evolution. The modified variations are influenced as in the following sub-sections.

3.4.2.1 Influence function for preying mCAFAC_Sd

The preying behaviour of AFSA is modelled using only the situational knowledge follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{rand(0,1) \times step_{itr} \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_i^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} < S_k^{t} \\ x_{ik}^{t} - \left| \dfrac{rand(0,1) \times step_{itr} \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_i^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} > S_k^{t} \\ x_{ik}^{t} + \dfrac{rand(0,1) \times step_{itr} \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_i^{t+1} - X_i^{t} \right\|} & otherwise \end{cases} \quad (3.11)$$

Where $step_{itr} = step \times (\frac{itr_{\max} - itr}{itr})^n$ and S_k^t is the best exemplar parameter value of the situational knowledge for variable *j* in the belief space. If an individual's parameter value is less than that of the current best, probably it would be better to go toward the current best. The current best itself is mutated using a random direction, which is the otherwise case in equation (3.11) (Chung, 1997).

Equation (3.11) is the influence function for preying mCAFAC_Sd when the i^{th} individual generation is less or equal to the maximum number of trial ($i \le try_num$).

When the i^{th} individual generation is greater than maximum number of trial $i > try_num$, the preying behaviour of mCAFAC_Sd is influenced as follows (Wu et al 2011):

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| rand(0,1) \times step_{itr} \right| & if\ \ x_{ik}^{t} < S_k^{t} \\ x_{ik}^{t} - \left| rand(0,1) \times step_{itr} \right| & if\ \ x_{ik}^{t} > S_k^{t} \\ x_{ik}^{t} + rand(0,1) \times step_{itr} & otherwise \end{cases} \quad (3.12)$$

3.4.2.2 Influence function for swarming mCAFAC_Sd

In this variant, the swarming behaviour of AFSA is guided using the situational knowledge in order to determine the step size of evolution and visual distance as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{rand(0,1) \times step_{itr} \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_{c}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} < S_{k}^{t} \\ x_{ik}^{t} - \left| \dfrac{rand(0,1) \times step_{itr} \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_{c}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} > S_{k}^{t} \\ x_{ik}^{t} + \dfrac{rand(0,1) \times step_{itr} \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_{c}^{t+1} - X_{i}^{t} \right\|} & otherwise \end{cases} \quad (3.13)$$

3.4.2.3 Influence function for chasing mCAFAC_Sd

In this variant, the chasing behaviour of AFSA is modelled using the situational knowledge to determine the step size of evolution and visual distance as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{rand(0,1) \times step_{itr} \times (x_{mk}^{t+1} - x_{ik}^{t})}{\left\| X_{m}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} < S_{k}^{t} \\ x_{ik}^{t} - \left| \dfrac{rand(0,1) \times step_{itr} \times (x_{mk}^{t+1} - x_{ik}^{t})}{\left\| X_{m}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} > S_{k}^{t} \\ x_{ik}^{t} + \dfrac{rand(0,1) \times step_{itr} \times (x_{mk}^{t+1} - x_{ik}^{t})}{\left\| X_{m}^{t+1} - X_{i}^{t} \right\|} & otherwise \end{cases} \quad (3.14)$$

The complete program for this variant (mCAFAC_NsSd) is given in Appendix B2

3.4.3 mCAFAC_NsSd: mCAFAC using normative knowledge and situational knowledge

In this variant, both the normative and situational knowledge are integrated. The normative knowledge is used to guide the step size and visual distance while the situational knowledge is used to determine the direction of evolution.

3.4.3.1 Influence function for preying mCAFAC_NsSd

In this case, the preying behaviour of AFSA was modified as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_i^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} < S_k^{t} \\ x_{ik}^{t} - \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_i^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} > S_k^{t} \\ x_{ik}^{t} + \dfrac{size(I_k) \times rand(0,1) \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_i^{t+1} - X_i^{t} \right\|} & otherwise \end{cases} \quad (3.15)$$

Where S_k^t is the best exemplar parameter value of the situational knowledge for variable k in the belief space. Equation (4.15) is the influence function for preying mCAFAC_NsSd when the i^{th} individual generation is less or equal to the maximum number of trials ($i \le try_num$).

When the i^{th} individual generation is greater than maximum number of trial $i > try_num$, the preying behaviour of mCAFAC_NsSd is influenced as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| size(I_k) \times rand(0,1) \right| & if\ \ x_{ik}^{t} < S_k^{t} \\ x_{ik}^{t} - \left| size(I_k) \times rand(0,1) \right| & if\ \ x_{ik}^{t} > S_k^{t} \\ x_{ik}^{t} + size(I_k) \times rand(0,1) & otherwise \end{cases} \quad (3.16)$$

3.4.3.2 Influence function for swarming mCAFAC_NsSd

In this case, the swarming behaviour of AFSA is modified as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_c^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} < S_k^{t} \\ x_{ik}^{t} - \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_c^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} > S_k^{t} \\ x_{ik}^{t} + \dfrac{size(I_k) \times rand(0,1) \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_c^{t+1} - X_i^{t} \right\|} & otherwise \end{cases} \quad (3.17)$$

Where S_k^t is the best exemplar parameter value of the situational knowledge for variable k in the belief space.

3.4.3.3Influence function for chasing mCAFAC_NsSd

The Chasing behaviour of AFSA in this version is modified using the following expression:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{mk}^{t} - x_{ik}^{t})}{\left\| X_{m}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} < S_k^{t} \\ x_{ik}^{t} - \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{mk}^{t} - x_{ik}^{t})}{\left\| X_{m}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} > S_k^{t} \\ x_{ik}^{t} + \dfrac{size(I_k) \times rand(0,1) \times (x_{mk}^{t} - x_{ik}^{t})}{\left\| X_{m}^{t+1} - X_{i}^{t} \right\|} & otherwise \end{cases} \tag{3.18}$$

The complete program for this variant (mCAFAC_NsSd) is given in Appendix B3.

3.4.4 mCAFAC_NsNd: mCAFAC using normative knowledge for step size, visual distance and the direction of evolution

In this variant, only normative knowledge is used to guide the step size, direction of evolution and visual distance. The influence functions for this variant are expressed as in the following sub-sections.

3.4.4.1 Influence function for preying mCAFAC_NsNd

In this case, the Preying behaviour of AFSA is modified using normative knowledge as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_{i}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} < l_k^{t} \\ x_{ik}^{t} - \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_{i}^{t+1} - X_{i}^{t} \right\|} \right| & if\ x_{ik}^{t} > u_k^{t} \\ x_{ik}^{t} + \dfrac{size(I_k) \times rand(0,1) \times (x_{ik}^{t+1} - x_{ik}^{t})}{\left\| X_{i}^{t+1} - X_{i}^{t} \right\|} & otherwise \end{cases} \tag{3.19}$$

Equation (3.19) is the influence function for preying mCAFAC_NsNd when the i^{th} individual generation is less or equal to the maximum number of trial ($i \le try_num$).

When the i^{th} individual generation is greater than maximum number of trial $i > try_num$, the preying behaviour of mCAFAC_NsSd is influenced as follows (Wu et al 2011):

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| size(I_k) \times rand(0,1) \right| & if \quad x_{ik}^{t} < l_k^{t} \\ x_{ik}^{t} - \left| size(I_k) \times rand(0,1) \right| & if \quad x_{ik}^{t} > l_k^{t} \\ x_{ik}^{t} + size(I_k) \times rand(0,1) & otherwise \end{cases} \tag{3.20}$$

3.4.4.2 Influence function for swarming mCAFAC_NsNd

In this case, the Swarming behaviour of AFSA is modified using the normative knowledge as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_c^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} < l_k^{t} \\ x_{ik}^{t} - \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_c^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} > u_k^{t} \\ x_{ik}^{t} + \dfrac{size(I_k) \times rand(0,1) \times (x_{ck}^{t+1} - x_{ik}^{t})}{\left\| X_c^{t+1} - X_i^{t} \right\|} & otherwise \end{cases} \tag{3.21}$$

3.4.4.3 Influence function for chasing mCAFAC_NsNd

In this case, the Chasing behaviour of AFSA is modified using the normative knowledge as follows:

$$x_{ik}^{t+1} = \begin{cases} x_{ik}^{t} + \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{mk}^{t+1} - x_{ik}^{t})}{\left\| X_m^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} < l_k^{t} \\ x_{ik}^{t} - \left| \dfrac{size(I_k) \times rand(0,1) \times (x_{mk}^{t+1} - x_{ik}^{t})}{\left\| X_m^{t+1} - X_i^{t} \right\|} \right| & if\ x_{ik}^{t} > u_k^{t} \\ x_{ik}^{t} + \dfrac{size(I_k) \times rand(0,1) \times (x_{mk}^{t+1} - x_{ik}^{t})}{\left\| X_m^{t+1} - X_i^{t} \right\|} & otherwise \end{cases} \quad (3.22)$$

The complete program for this variant is given in Appendix B4

3.5 Recombination Operator

The crossover operator described in section 2.2.3 was applied to determine whether the modified (mCAFAC) falls into local optimum or not. It produced a new improved off-spring by recombination (fusion) of the information from AFSA and mCAFAC. The flowchart for the modified algorithms is shown in Figure 3.2

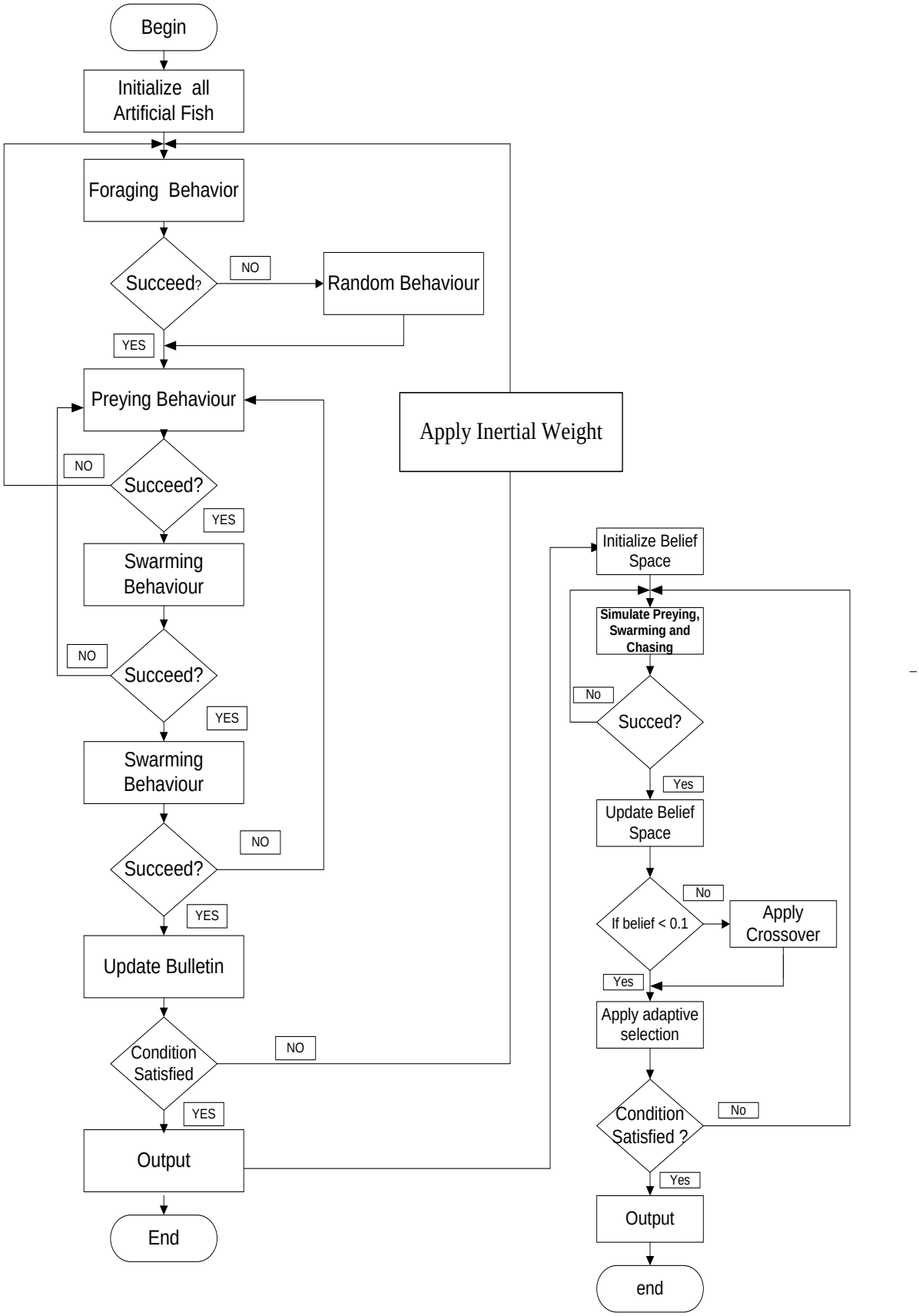


Figure 3.2: Flowchart of Modified Cultural Artificial Fish Swarm Algorithm with Crossover

3.6 Determination of Weighting Matrices Using mCAFAC

The variant of mCAFAC with superior performance over all the other variants developed in this research is used for the determination of optimal values of the weighting matrices (Q & R).

The most important points in Q and R matrices are the constraints on them. As mentioned before, the Q and R matrices should be positive and semi-definite and positive and definite respectively. However, these constraints cannot be easily satisfied. It is hard to define a simple relation between matrix elements and their positive definiteness.

Thus, in this research, a method, described by equations (3.23) and (3.24) is used to make Q and R positive semi-definite matrices. It is known that for any real matrix G, $H = G^T G$ is non-negative, i.e., it can be said that matrix H is positive-semi-definite (Wang et al., 2014). The above method is used in this research to make initial responses and to code the responses of the problem. Instead of using Q and R matrices as unknown variables, the matrices Mand N that satisfy equations (3.23) and (3.24) are used as unknown variables. Thus mCAFAC_NsSd is used to first find the optimal values of M and N, then Q and R matrices are calculated as follows:

$$Q = M^T M \tag{3.23}$$

$$R = N^T N \tag{3.24}$$

The matrices M and N are initialized such that $M = diag(m1, m2, ... m_n)$(i.e. an identity matrix) and N = 1. Then the corresponding cost function given in equation (2.44) becomes

$$J = \int_0^\infty (q_1 x_1^T + q_2 x_2^T + \ldots + q_n x_n^T)^2 \, dt \tag{3.25}$$

3.6.1 Quadruple inverted pendulum stabilization

The linearized state space model extracted from the Double Quadruple Inverted Pendulum (DQIP) by truncating the mathematical modelling of the DQIP at the fourth link is given as follows (Abdullahi, 2014):

$$\dot{x} = \begin{bmatrix} 0_{5x5} & I_{5x5} \\ -M^{-1}(0)\dfrac{\partial D(0)}{\partial \theta} & -M^{-1}(0)\,\mathrm{N}(0) \end{bmatrix} x + \begin{bmatrix} 0_{5x5} \\ M^{-1}(0)\,\mathrm{G} \end{bmatrix} u \tag{3.26}$$

Equation (3.26) shows the compact state space model of the extracted QIP system which corresponds to the standard state space described in equation (2.39)

Where

$$A = \begin{bmatrix} 0_{5x5} & I_{5x5} \\ -M^{-1}(0)\dfrac{\partial D(0)}{\partial \theta} & -M^{-1}(0)\,\mathrm{N}(0) \end{bmatrix} \tag{3.27}$$

$$B = \begin{bmatrix} 0_{5x5} \\ M^{-1}(0)\,\mathrm{G} \end{bmatrix} \tag{3.28}$$

Equation (3.27) and (3.28) are the state and input matrices of the extracted QIP system respectively.

Figure 3.3 shows the diagram of quadruple inverted pendulum extracted from (Abdullahi, 2014) DQIP model.

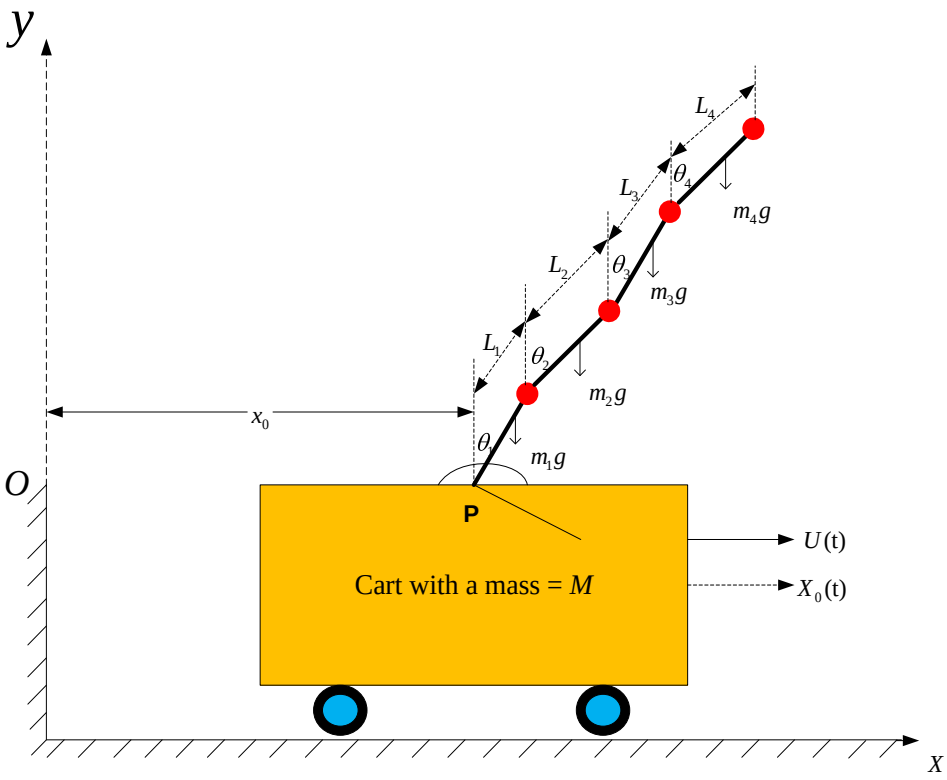


Figure 3.3: Inverted Quadruple Inverted Pendulum Extracted From (Abdullahi, 2014)

The state and input matrices obtained from the linearized state space model given in equation (2.36) are as follows:

$$A=\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -6.5881 & 1.2417 & -0.1983 & 0.0219 & -16.4652 & 0.0194 & -0.079 & 0.0022 & -0.0005 \\ 0 & 124.9542 & -99.3947 & 15.8763 & -1.7537 & 42.6129 & -0.4480 & 0.3344 & -0.1759 & 0.0406 \\ 0 & -141.6903 & 199.4814 & -76.9531 & 8.5003 & -11.4491 & 0.6004 & -0.6732 & 0.5519 & -0.1969 \\ 0 & 39.3968 & -133.9553 & 129.7706 & -32.2476 & 3.1834 & -0.2505 & 0.05552 & -0.8295 & 0.4463 \\ 0 & -14.9856 & 50.9534 & -111.0453 & 73.9500 & -1.2109 & 0.0.953 & -0.3255 & 0.9378 & -0.6778 \end{bmatrix} \quad (3.29)$$

$$B=\begin{bmatrix} 0 & 0 & 0 & 0 & 0.7185 & -1.8596 & 0.4996 & -0.1389 & 0.0528 \end{bmatrix}^T \quad (3.30)$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (3.31)$$

$$D = 0 \quad (3.32)$$

Where A is the state transition matrix, B is the input matrix, C is the output matrix and D is the disturbance matrix(Abdullahi, 2014). The initial state of the system is set as(Abdullahi, 2014):

$$X_0 = \begin{bmatrix} 2 & -0.04 & 0.02 & -0.01 & 0.005 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad (2.33)$$

Control test was performed to determine the controllability and observability of the system (QIP), before the mCAFAC is then used to obtain the Q and R weighting matrices. The snippet of the control test code is given as follows.

```
%Determine 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)
```

The output of the above program is as follows:

```
ans =

System is Controlable

CONTR_RANK =

    10

ans =

System is Observable

OBSV_RANK =

    10
```

The results obtained from this test confirmed 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. A program is written to first of all, determine the initial positions of the eigenvalues as follows:

```
[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*max(max(A))*100)^
0.125)*max(max(B)))^0.5;
    eigenvalue(i,i)=realeigen+1i*(-1)^i*imageigen;
end
eigenvalue=real(eigenvalue);
```

Where A and B are the state and input matrices obtained from the linearized LQR model (Abdullahi, 2014).

Equation (2.43) is used to determine the appropriate gain that satisfies the locations of the poles and mCAFAC is used to randomly determine the optimal values of Q and R which satisfies equations (3.23) and (3.24) respectively. The complete program can be found in Appendix A. The program in Appendix A is called from mCAFAC as a fitness function using the following code.

```
if fitness==17
    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
    [Q,R]=lqrQR(A,B)
end
```

The stability of the system using the values of Q and R weighting matrices determined using mCAFAC was confirmed using the pole zero map plot (pzmap) in matlab.

```
[K P E]=lqr (A,B,Q,R);%Q and R used here are obtained from
                        mCAFA_NsSd
sys_LQ=ss(A-B*K,B,C,D);
pzmap(sys_LQ);% Used to plot the system response and
                Confirming the pole location
sgrid
```

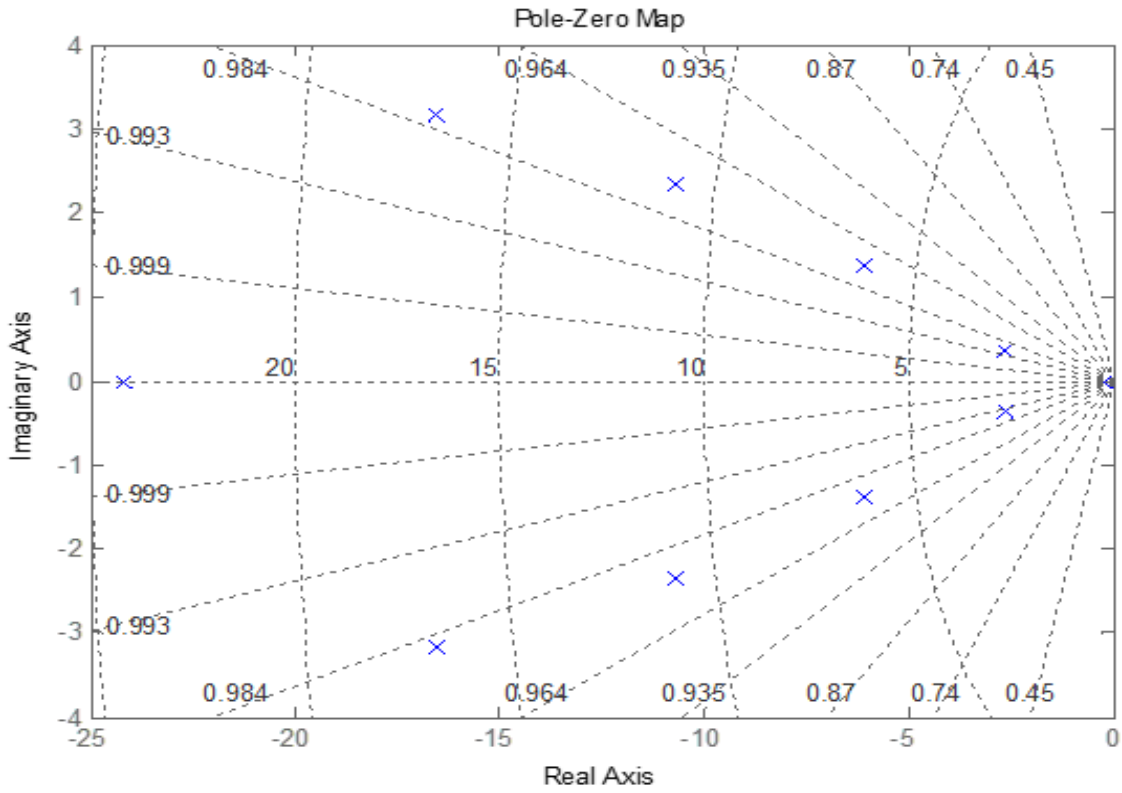


Figure 3.4 Pole-Zero Map of the QIP System

Figure 3.3 shows the locations of the eigenvalues in the complex s-plane. It can be observed that all the eigenvalues (marked ‘x’) are on the negative half of the complex s-plane, which is a confirmation of stability.

The step-by-step approach to obtaining the Q and R weighting matrices are as follows.

1. Initialize all the parameters of the mCAFAC.
2. Defined the problem by specifying A, B and initialize the pole location
3. Provide an initial guess for M and N.
4. Calculate Q and R from equations (3.23) and (3.24) respectively using mCAFAC
5. Repeat step 4 until the best value of Q and R is obtained
6. Solve the Riccati equation and determine the gain matrix K.

CHAPTER FOUR

RESULTS AND DISCUSSIONS

4.1 Introduction

In this Chapter, the performances of the AFSA, mAFSA, mCAFAC_Ns, CAFAC_Sd, mCAFAC_NsSd and mCAFAC_NsNd are evaluated using the optimization test functions discussed in subsection 2.3. Three dimensional (3D) plots were presented to enhance the understanding of all the test function. The performances of the proposed algorithms were compared with the performance of the replicated artificial bee colony (ABC) algorithm for further validation. The LQR weighting matrices determined using mCAFAC and the responses of the QIP were also presented in this chapter.

4.2 Visualization of the Optimization Test Function

To enhance the understanding and visualization of the local minimal point of most of the optimization test functions described in section 2.2.6, Figures 4.1- 4.16 were generated in Matlab program to show the surfaces and shapes of the test functions.

4.2.1 Ackley function

The 3D visualization of Ackley function is as shown in Figure 4.1.

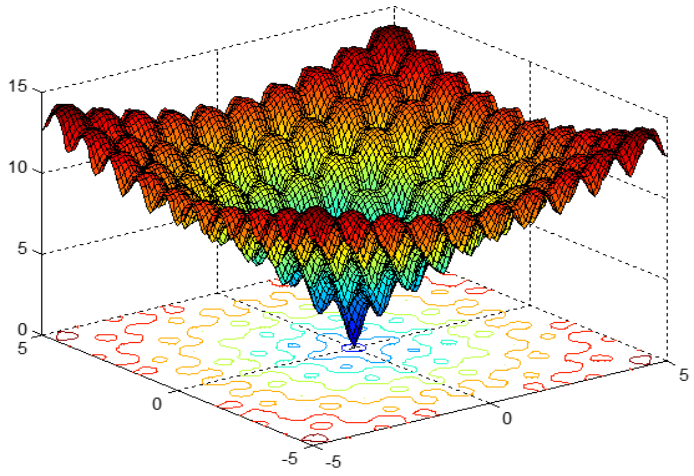


Figure 4.1: Visualization of Ackley Function

From Figure 4.1, it is obvious that Ackley function has many local minima, but the most minima (global minima) of this function can be observed at point (0, 0) as can been seen from Figure 4.1.

4.2.2 CM function

Just like Ackley function, it obvious that CM function has a lot local minimal around the global minimal. As specified in section 2.2.6.2, the global minimal of this test function is at -3.0. The 3D visualization of Ackley function is as shown in Figure 4.2.

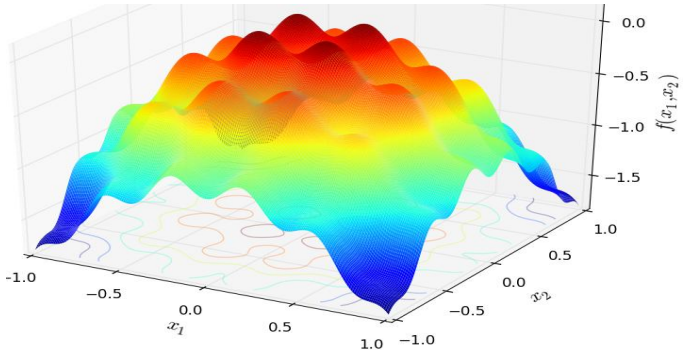


Figure 4.2: 3D Visualization of Cosine Mixture Function

4.2.3 DeJongF4 function

The global minimal of DejongF4 is 0 which can be observed in the 3D visualization of the function in Figure 4.3. This function is described in section 2.2.6.3

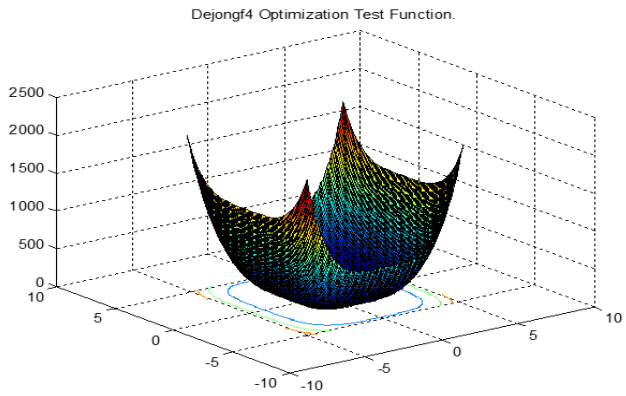


Figure 4.3: 3D Visualization of DejongF4 Function

4.2.4 Expfun function

It can be observed from the 3D visualization of Figure 4.4, that the global maximal of this function is located at point 0. This function is described in section 2.2.6.4

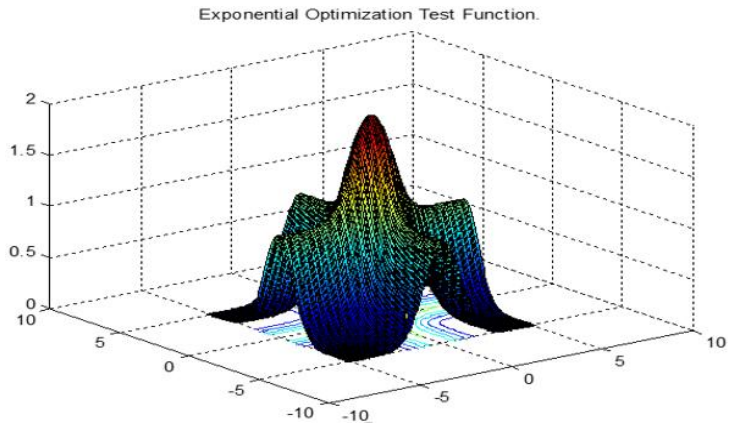


Figure 4.4: 3D Visualization of Exponential Function

4.2.5 Griewangk function

Griewangk's function is similar to Rastrigin's function. It has many widespread local minima as seen in the 3D visualization of Figure 4.5.

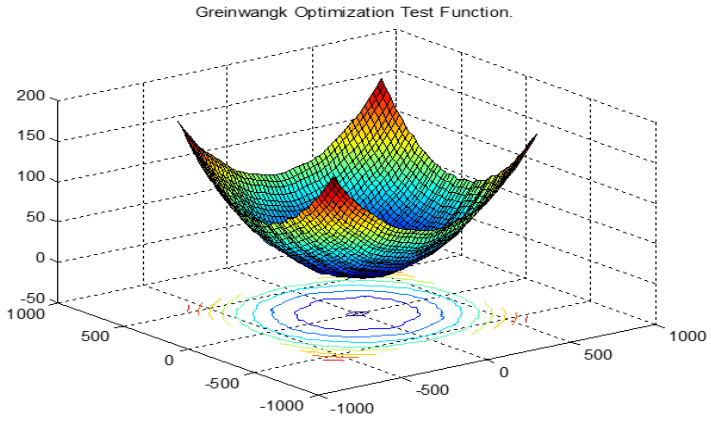


Figure 4.5: 3D Visualization of Greiwangk function

It can be observed that, the global minimal of this function is at point 0. The function is described in section 2.2.6.5

4.2.6 Hyperelliptic function

Hyperelliptic function has it global minimal located at point zero as described in section 2.2.6.6

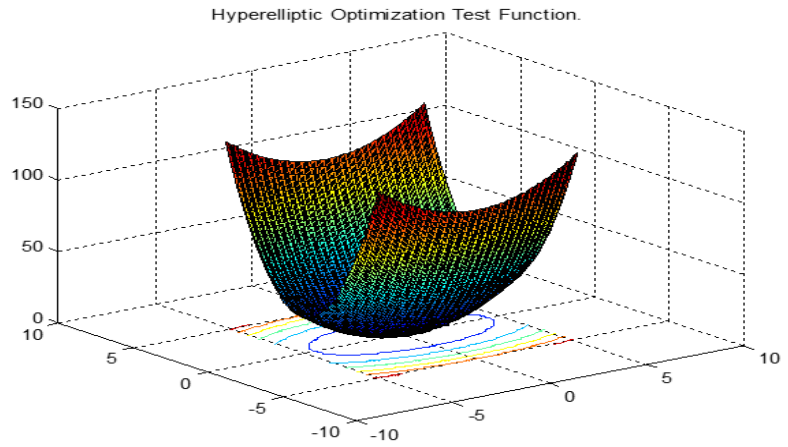


Figure 4.6: 3D Visualization of Hyperelliptic Function

From the 3D visualization of Figure 4.6, it can be observed that the global minimal of Hyperelliptic function is located at 0.

4.2.7 Levy and Montalvo first function

The Levy and Montalvo LM1 function has a global minimum which is $f_* = 0$ at $x_* = (0,0,...,0)$ as described in section 2.2.6.7

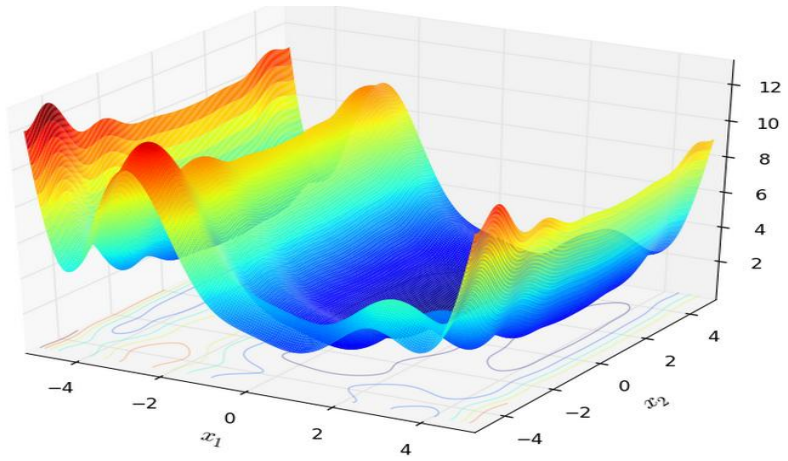


Figure 4.7: 3D Visualization of Levy and Montalvo (LM1) Function

From the 3D visualization of Figure 4.7, it is observed that the global minimal of this function is located at point 0.

4.2.8 Levy and Montalvo second functions

Just like the LM1 function, the LM2 function has its local minimal at 0. The detail of this function is described section 2.2.6.8

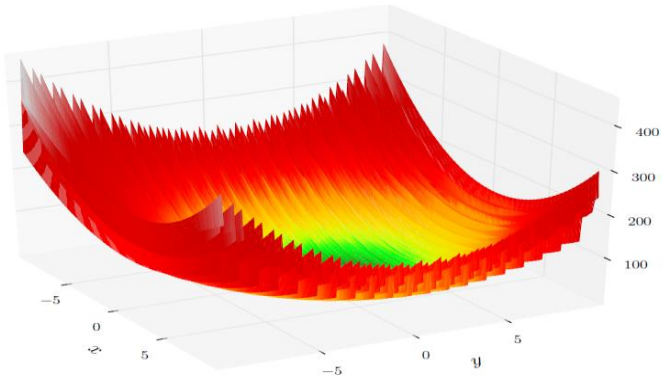


Figure 4.8: 3D Visualization of Levy and Montalvo Optimization Test Function

Figure 4.8 shows the 3D Visualization of the function and it can be seen that the global minimal of this function is located at point 0.

4.2.9 Neumaier3 function

The Neumaier3 function is as described in section 2.2.6.8 and its 3D visualization is as shown in Figure 4.9.

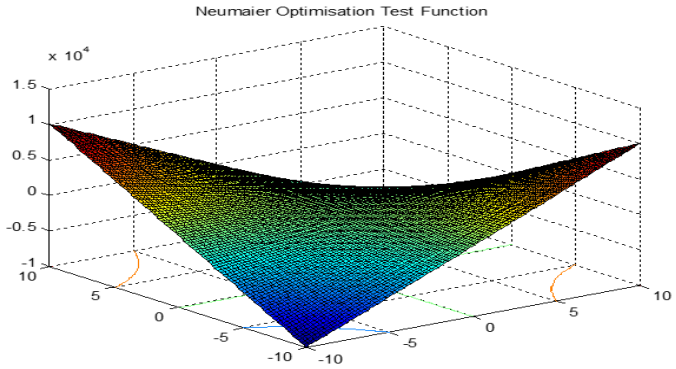


Figure 4.9: 3D Visualization of Neumaier Function

The neumaier third function has it global optimal located at point zero (0). This can be observed from the Figure 4.9.

4.2.10 Rastrigin test function

As discussed in section 2.2.8.10, Rastrigin's function is based on ExpFun with the addition of cosine modulation to produce many local minima. The 3D visualization of Rastrigin function is as shown in Figure 4.10 with a global minimal located at 0.

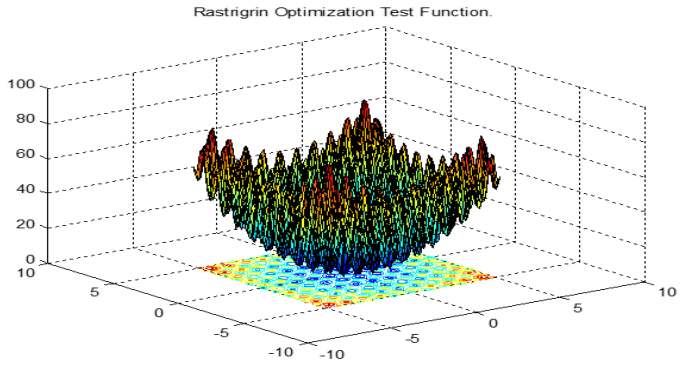


Figure 4.10: 3D Visualization of Rastrigin Function

4.2.11 Rosenbrock test fuction

Rosenbrock function has its global optimum inside a long, narrow, parabolic shaped flat valley which makes it very difficult for optimization algorithm to converge towards the global optimum.

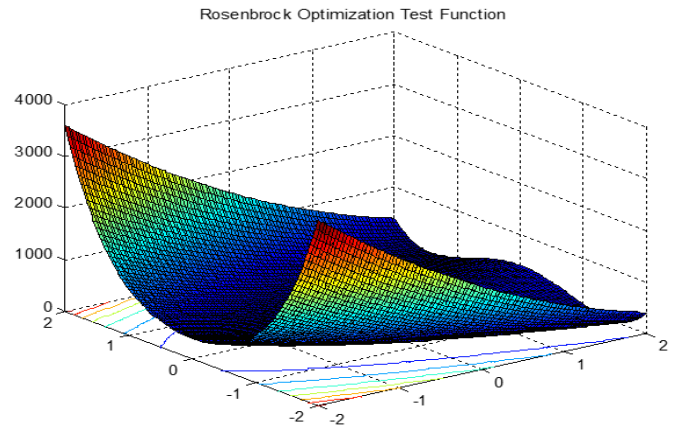


Figure 4.11: 3D Visualization of Rosenbrock function

As shown in the 3D visualization of Figure 4.11, the global minimal of Rosenbrock Valley function can be carefully observed at point 0.

4.2.12 Sal test function

Sal, function is a multi-modal optimization test function which was used to evaluate the performance of the algorithms. The global optimal of this functions is located at point zero as can be observed from the 3D visualization of Figure 4.12.

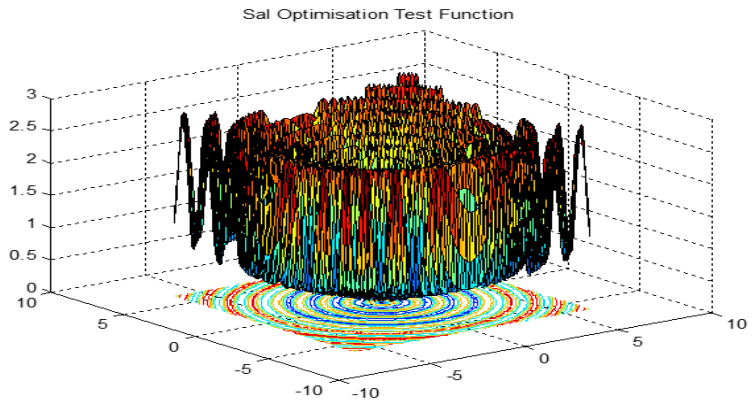


Figure 4.12: 3D Visualization of Sal Function

4.2.13 Schwefel test function

This function is deceptive, with a lot of local minimal spread around the global minimal. The global point of this function is at 0, which is shown in the 3D visualization of Figure 4.13. The detailed description of this function can be found in section 2.2.6.13

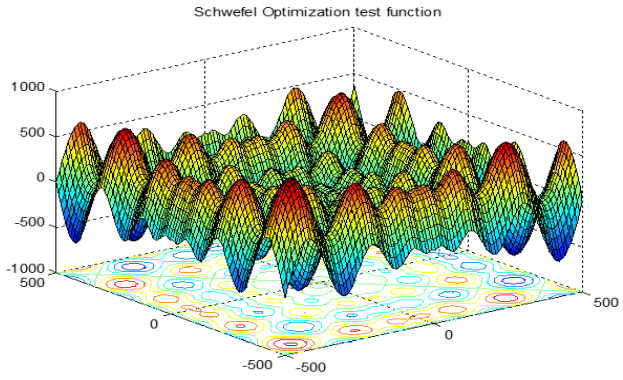


Figure 4.13: 3D Visualization of Schwefel Function

4.2.14 Schaffer test function

Schaffer is a multimodal non-convex optimization test function described in section 2.2.6.14. Figure 4.14 shows the 3D visualization of Schaffer optimization test function and it can be seen that the global minimal of the Schaffer function is 0.

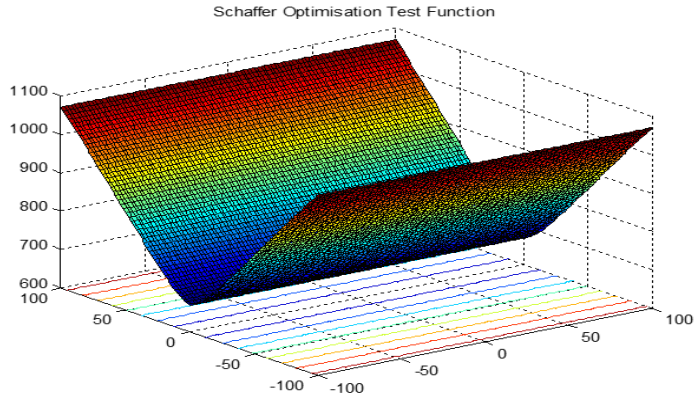


Figure 4.14: 3D Visualization of Schaffer Function

4.2.15 Sphere test function

The sphere function is sometimes called the simplest form of Dejong function as described in section 2.2.6.15. It is uni-modal and non-convex. The local minimal of this function is 0 as can be seen in the 3D visualization of Figure 4.15.

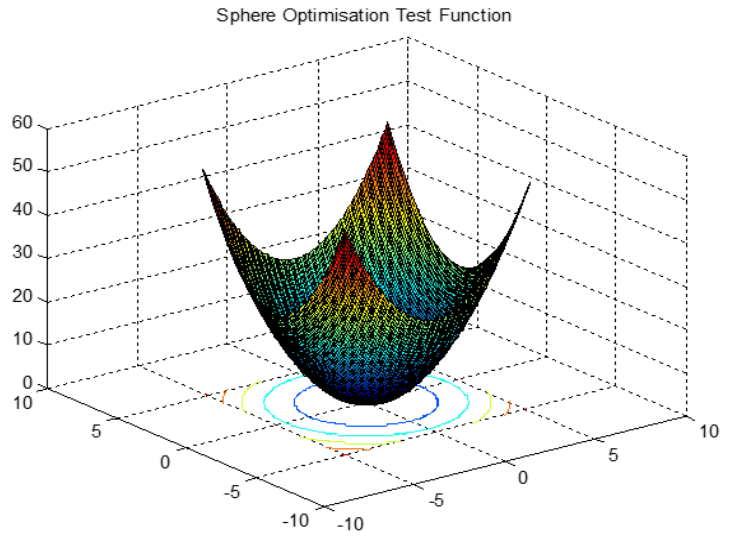


Figure 4.15: 3D Visualization of Sphere Function

4.2.16 Bukin N.6 test function

The Bukin N.6 function is a multi-modal test function with several local minima around the global minima located at point (-10, 1). The detailed description of this function can be found in section 2.2.6.16. Figure 4.16 shows the 3D visualization of Bukin function.

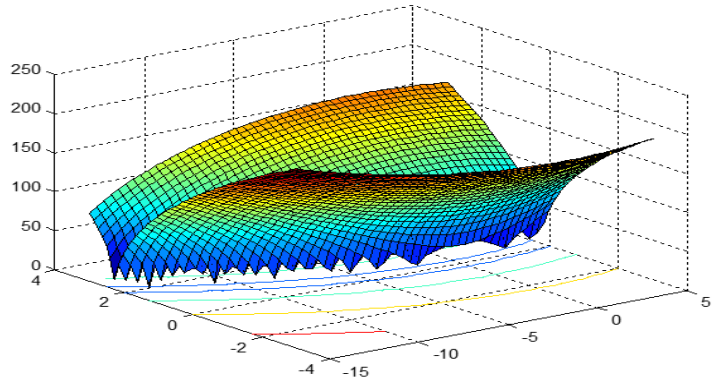


Figure 4.16: 3D Visualization of Bukin Function

The Matlab program for all the test functions (objective function) and the programme used to plot all the 3D graphs can be found in Appendix C1 and C2 respectively.

4.3 Performance Evaluation of Proposed AFSA Algorithms

The performance of the developed algorithm evaluated using the sixteen multi-modal high dimensional optimization test function are evaluated in this subsection

4.3.1 Performance evaluation of AFSA (With and Without Inertial Weight)

The standard Artificial Fish Swarm Algorithm was first replicated. The performance of the replicated AFSA was evaluated using sixteen most commonly used optimization test functions and the result is compared with the result obtained using the modified Artificial Fish Swarm Algorithm (mAFSA) . The results are shown in Table 4.1

Table 4.1: Performance Evaluation of Standard AFSA and mAFSA Algorithm

S/N:	Test Functions	Global minimal	AFSA	mAFSA
1	Ackley	0.0000E+00	1.6650E-01	**1.0590E-01**
2	CM	-3.0000E+00	-2.7819E+00	**-2.8396E+00**
3	DejongF4	0.0000E+00	1.1540E-04	**1.1918E-04**
4	ExpFun	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*
5	Grienwangk	0.0000E+00	***0.0000E+00***	1.4350E-260
6	Heperelliptic	0.0000E+00	**4.9310E-01**	7.2590E-01
7	LM1	0.0000E+00	9.7000E-03	**3.5000E-03**
8	LM2	0.0000E+00	1.2490E-01	**7.7700E-02**
9	Neumaier3	-4.9300E+03	**-8.2196E+02**	-2.6676E+02
10	Rastrigin	0.0000E+00	3.9448E+00	**2.9670E+00**
11	Rosenbrock	0.0000E+00	3.4313E+00	**3.2519E+00**
12	Sal	0.0000E+00	1.1350E-01	**1.2000E-03**
13	Schwefel	0.0000E+00	4.1500E+03	**4.1359+003**
14	Schaffer	0.0000E+00	4.4972E+00	**4.3354E+00**
15	Sphere	0.0000E+00	1.8300E-02	**1.7300E-02**
16	Bukin	0.0000E+00	3.6973E-03	**1.3883E-03**

It is obvious from Table 4.1, that, mAFSA outperformed the AFSA in almost all the optimization test function except in test case 5, 6 and 9. However, AFSA did well for this class of functions (low-dimensional and relatively easy functions), but did poorly on others (high-dimensional and more complex functions). The superiority of mAFSA over AFSA is expected since inertial weight was introduced which guided its evolution towards the global optimal. Table 4.1 will form the basis for the evaluation of the mAFSA and the modified cultural artificial fish swarm algorithm with crossover (mCAFAC). As mAFSA has shown an improvement over AFSA, it is expected that mCAFAC should show a superior overall improvement. The performance of the improved mCAFAC is compared with results obtained using ABC.

Note that the algorithms with the best and similar performances with respect to the global solution are shown in bold and italic characters in the tables respectively.

4.3.2 Performance evaluation of mCAFAC_Ns using normative knowledge.

As discussed in Chapter Two, the performance of the mCAFAC when the normative knowledge was used to determine the step size and the visual distance of evolution is evaluated. The result obtained from mCAFAC_Ns is compared with that of AFSA and mAFSA. The results are tabulated as in Table 4.2.

Table 4.2: Performance Evaluation of AFSA, mAFSA and mCAFAC_Ns

S/N:	Test Functions	Global minimal	AFSA	mAFSA	mCAFAC_Ns
1	Ackley	0.0000E+00	1.6650E-01	1.0590E-01	**8.8818E-12**
2	CM	-3.0000E+00	-2.7819E+00	-2.8396E+00	**-2.9360E+00**
3	DejongF4	0.0000E+00	1.1540E-04	1.1918E-04	**2.2532E-05**
4	ExpFun	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*
5	Grienwangk	0.0000E+00	*0.0000E+00*	1.4350E-260	*0.0000E+00*
6	Heperelliptic	0.0000E+00	4.9310E-01	7.2590E-01	**5.0055E-46**
7	LM1	0.0000E+00	9.7000E-03	3.5000E-03	**3.3000E-03**
8	LM2	0.0000E+00	1.2490E-01	7.7700E-02	**9.0000E-03**
9	Neumaier3	-4.9300E+03	**-8.2196E+02**	-2.6676E+02	-5.6179E+02
10	Rastrigin	0.0000E+00	3.9448E+00	2.9670E+00	**1.9320E+00**
11	Rosenbrock	0.0000E+00	3.4313E+00	**3.2519E+00**	4.0827E+00
12	Sal	0.0000E+00	1.1350E-01	1.2000E-03	**9.0000E-04**
13	Schwefel	0.0000E+00	4.1500E+03	**4.1359+003**	4.1504E+03
14	Schaffer	0.0000E+00	4.4972E+00	4.3354E+00	**4.3235E+00**
15	Sphere	0.0000E+00	1.8300E-02	1.7300E-02	**2.5000E-03**
16	Bukin	0.0000E+00	3.6973E-03	1.3883E-03	**2.9825E-16**

Notice that the normative knowledge based mCAFAC performed much better in most of the optimization test functions. In test case nine (9), the AFSA produced a better result when compared with mCAFAC_Ns and in test cases eleven (11) and thirteen (13), the mAFSA produced the best results. The introduction of knowledge in the belief space ensured a significant improvement on the precision accuracy of AFSA. In test case four (4) and five (5), all the algorithms produced the same result, which is an indication of unimodal nature of these functions.

4.3.3 Performance evaluation of mCAFAC_Sd using situational knowledge

The performance of the mCAFAC when the situational knowledge of the current exemplar or best solution found so far was used to determine the step size and the visual distance of evolution is evaluated in this section. The result obtained is compared with the results obtained from AFSA, mAFSA and mCAFAC_Ns. The results are tabulated as in Table 4.3.

Table 4.3 Performance Evaluation of AFSA, mAFSA, mCAFAC_Ns and mCAFAC_Sd

S/N:	Test Functions	Global minimal	AFSA	mAFSA	mCAFAC_Ns	mCAFAC_Sd
1	Ackley	0.0000E+00	1.6650E-01	1.0590E-01	**8.8818E-12**	7.1200E-02
2	CM	-3.0000E+00	-2.7819E+00	-2.8396E+00	**-2.9360E+00**	-2.7769E+00
3	DejongF4	0.0000E+00	1.1540E-04	1.1918E-04	**2.2532E-05**	1.4889E-04
4	ExpFun	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*
5	Grienwangk	0.0000E+00	*0.0000E+00*	1.4350E-260	*0.0000E+00*	1.7260E-189
6	Heperelliptic	0.0000E+00	4.9310E-01	7.2590E-01	**5.0055E-46**	3.4510E-08
7	LM1	0.0000E+00	9.7000E-03	3.5000E-03	**3.3000E-03**	8.7000E-03
8	LM2	0.0000E+00	1.2490E-01	7.7700E-02	**9.0000E-03**	1.2520E-02
9	Neumaier3	-4.9300E+03	-8.2196E+02	-2.6676E+02	-5.6179E+02	**-9.3858E+02**
10	Rastrigin	0.0000E+00	3.9448E+00	2.9670E+00	**1.9320E+00**	2.4909E+00
11	Rosenbrock	0.0000E+00	3.4313E+00	3.2519E+00	4.0827E+00	**3.2104E+00**
12	Sal	0.0000E+00	1.1350E-01	1.2000E-03	**9.0000E-04**	9.9900E-02
13	Schwefel	0.0000E+00	4.1500E+03	**4.1359+003**	4.1504E+03	4.1505E+03
14	Schaffer	0.0000E+00	4.4972E+00	4.3354E+00	**4.3235E+00**	4.6513E+00
15	Sphere	0.0000E+00	1.8300E-02	1.7300E-02	**2.5000E-03**	1.7500E-02
16	Bukin	0.0000E+00	3.6973E-03	1.3883E-03	2.9825E-16	**3.7531E-18**

It can be observed that, the variant that employed normative knowledge produced the best result for ten (10) out of sixteen (16) optimization test cases considered while the variant that employed the situational knowledge produced the best result in three (3) cases. Thus, normative knowledge appeared to be the dominant knowledge source for the optimization functions.

4.3.4 Performance evaluation of mCAFAC_NsSd using situational knowledge.

The performance of the mCAFAC when the normative knowledge was used to guide the step size and the situational knowledge of the current exemplar or best solution found so far was

used to determine the direction of evolution is evaluated in this section. The result obtained is compared with the results obtained from AFSA, mAFSA, mCAFAC_Ns and mCAFAC_Sd. The results are tabulated as in Table 4.4.

Table 4.4 Performance Evaluation of AFSA, mAFSA, mCAFAC_Ns, mCAFAC_Sd and mCAFAC_NsSd

S/N:	Test Functions	Global minimal	AFSA	mAFSA	mCAFAC_Ns	mCAFAC_Sd	mCAFAC_NsSd
1	Ackley	0.0000E+00	1.6650E-01	1.0590E-01	8.8818E-12	7.1200E-02	**3.0933E-16**
2	CM	-3.0000E+00	-2.7819E+00	-2.8396E+00	-2.9360E+00	-2.7769E+00	**-2.9370E+00**
3	DejongF4	0.0000E+00	1.1540E-04	1.1918E-04	**2.2532E-05**	1.4889E-04	2.9161E-04
4	ExpFun	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*
5	Grienwangk	0.0000E+00	*0.0000E+00*	1.4350E-260	*0.0000E+00*	1.7260E-189	*0.0000E+00*
6	Heperelliptic	0.0000E+00	4.9310E-01	7.2590E-01	5.0055E-46	3.4510E-08	**5.0342E-58**
7	LM1	0.0000E+00	9.7000E-03	3.5000E-03	3.3000E-03	8.7000E-03	**1.2374E-50**
8	LM2	0.0000E+00	1.2490E-01	7.7700E-02	9.0000E-03	1.2520E-02	**3.9114E-49**
9	Neumaier3	-4.9300E+03	-8.2196E+02	-2.6676E+02	-5.6179E+02	-9.3858E+02	**-1.0126E+03**
10	Rastrigin	0.0000E+00	3.9448E+00	2.9670E+00	**1.9320E+00**	2.4909E+00	4.6819
11	Rosenbrock	0.0000E+00	3.4313E+00	3.2519E+00	4.0827E+00	**3.2104E+00**	4.3120E+00
12	Sal	0.0000E+00	1.1350E-01	1.2000E-03	**9.0000E-04**	9.9900E-02	0.00999
13	Schwefel	0.0000E+00	4.1500E+03	4.1359+003	4.1504E+03	4.1505E+03	**3.5538e+003**
14	Schaffer	0.0000E+00	4.4972E+00	4.3354E+00	4.3235E+00	4.6513E+00	**0.2004**
15	Sphere	0.0000E+00	1.8300E-02	1.7300E-02	2.5000E-03	1.7500E-02	**2.9811E-12**
16	Bukin	0.0000E+00	3.6973E-03	1.3883E-03	2.9825E-16	3.7531E-18	**4.3593E-19**

Notice, that the variant that employed both the normative and situational knowledge produced the best result in ten (10) out of sixteen (16) cases. However, in test cases four (4) and five (5) the population-only AFSA and three of the self-adaption mCAFAC produced the same global result. This confirmed the regularity and unimodal nature of the test functions. Only mAFSA and mCAFAC_Sd produced different results in these cases. In this research, such results are considered as optimal, since the divergence away from the global result is so small and as such, can be neglected.

4.3.5 Performance evaluation of mCAFAC_NsNd using situational knowledge.

The performance of the mCAFAC when the normative knowledge is used to guide the step size, visual distance and direction of evolution is evaluated in this section. The result obtained is

compared with the results obtained from AFSA, mAFSA, mCAFAC_Ns, mCAFAC_Sd and mCAFAC_NsSd. The results are tabulated as in Table 4.5.

Table 4.5 Performance Evaluation of AFSA, mAFSA, mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd and mCAFAC_NsNd

S/N:	Test Functions	Global minimal	AFSA	mAFSA	mCAFAC_Ns	mCAFAC_Sd	mCAFAC_NsSd	mCAFAC_NsNd
1	Ackley	0.0000E+00	1.6650E-01	1.0590E-01	8.8818E-12	7.1200E-02	**3.0933E-16**	8.8818E-16
2	CM	-3.0000E+00	-2.7819E+00	-2.8396E+00	-2.9360E+00	-2.7769E+00	**-2.9370E+00**	-2.8650E+00
3	DejongF4	0.0000E+00	1.1540E-04	1.1918E-04	**2.2532E-05**	1.4889E-04	2.9161E-04	3.3725E-04
4	ExpFun	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*
5	Grienwangk	0.0000E+00	*0.0000E+00*	1.4350E-260	*0.0000E+00*	1.7260E-189	*0.0000E+00*	*0.0000E+00*
6	Heperelliptic	0.0000E+00	4.9310E-01	7.2590E-01	5.0055E-46	3.4510E-08	**5.0342E-58**	7.8235E-58
7	LM1	0.0000E+00	9.7000E-03	3.5000E-03	3.3000E-03	8.7000E-03	**1.2374E-50**	5.0929e-050
8	LM2	0.0000E+00	1.2490E-01	7.7700E-02	9.0000E-03	1.2520E-02	**3.9114E-49**	4.5400E-48
9	Neumaier3	-4.9300E+03	-8.2196E+02	-2.6676E+02	-5.6179E+02	-9.3858E+02	**-1.0126E+03**	-7.7103E+02
10	Rastrigin	0.0000E+00	3.9448E+00	2.9670E+00	**1.9320E+00**	2.4909E+00	4.6819	4.0191
11	Rosenbrock	0.0000E+00	3.4313E+00	3.2519E+00	4.0827E+00	**3.2104E+00**	4.3120E+00	1.8244E+01
12	Sal	0.0000E+00	1.1350E-01	1.2000E-03	**9.0000E-04**	9.9900E-02	0.00999	3.0070E-01
13	Schwefel	0.0000E+00	4.1500E+03	4.1359+003	4.1504E+03	4.1505E+03	**3.5538e+003**	4.0521E+03
14	Schaffer	0.0000E+00	4.4972E+00	4.3354E+00	4.3235E+00	4.6513E+00	**0.2004**	6.5510E-01
15	Sphere	0.0000E+00	1.8300E-02	1.7300E-02	2.5000E-03	1.7500E-02	**2.9811E-12**	6.4954e-009
16	Bukin	0.0000E+00	3.6973E-03	1.3883E-03	2.9825E-16	3.7531E-18	4.3593E-19	**1.1406E-19**

The variant that employed normative knowledge for step size, visual distance and direction of evolution (mCAFAC_NsNd) has similar performance when compared with mCAFAC_NsSd. This was expected since the normative knowledge appears to be the dominant knowledge for the test cases under consideration. It can be observed that mCAFAC_NsSd is the most successful of all the variants, although it barely edged out mCAFAC_NsNd with significant margin in most of the test cases.

4.3.6 Performance evaluation of artificial bee colony (ABC) algorithm

In order to further validate and confirm the optimization capability of the developed algorithms,

the results were compared with the results obtained using the replicated Artificial Bee Colony Algorithm (ABC) as in Table 4.6.

Table 4.6 Performance Evaluation of AFSA, mAFSA, mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd, mCAFAC_NsNd and ABC

S/N:	Test Functions	Global minimal	AFSA	mAFSA	mCAFAC_Ns	mCAFAC_Sd	mCAFAC_NsSd	mCAFAC_NsNd	ABC
1	Ackley	0.0000E+00	1.6650E-01	1.0590E-01	8.8818E-12	7.1200E-02	**3.0933E-16**	8.8818E-16	2.0950E-01
2	CM	-3.0000E+00	-2.7819E+00	-2.8396E+00	-2.9360E+00	-2.7769E+00	**-2.9370E+00**	-2.8650E+00	-2.2905E+00
3	DejongF4	0.0000E+00	1.1540E-04	1.1918E-04	**2.2532E-05**	1.4889E-04	2.9161E-04	3.3725E-04	6.8000E-03
4	ExpFun	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*	*1.0000E+00*
5	Grienwangk	0.0000E+00	*0.0000E+00*	1.4350E-260	*0.0000E+00*	1.7260E-189	*0.0000E+00*	*0.0000E+00*	*0.0000E+00*
6	Heperelliptic	0.0000E+00	4.9310E-01	7.2590E-01	5.0055E-46	3.4510E-08	**5.0342E-58**	7.8235E-58	1.6936E+00
7	LM1	0.0000E+00	9.7000E-03	3.5000E-03	3.3000E-03	8.7000E-03	**1.2374E-50**	5.0929e-050	7.1000E-03
8	LM2	0.0000E+00	1.2490E-01	7.7700E-02	9.0000E-03	1.2520E-02	**3.9114E-49**	4.5400E-48	1.4300E-01
9	Neumaier3	-4.9300E+03	-8.2196E+02	-2.6676E+02	-5.6179E+02	-9.3858E+02	**-1.0126E+03**	-7.7103E+02	-1.9603E+02
10	Rastrigin	0.0000E+00	3.9448E+00	2.9670E+00	**1.9320E+00**	2.4909E+00	4.6819	4.0191	5.2458E+00
11	Rosenbrock	0.0000E+00	3.4313E+00	3.2519E+00	4.0827E+00	**3.2104E+00**	4.3120E+00	1.8244E+01	6.8439E+00
12	Sal	0.0000E+00	1.1350E-01	1.2000E-03	**9.0000E-04**	9.9900E-02	0.00999	3.0070E-01	9.9900E-02
13	Schwefel	0.0000E+00	4.1500E+03	4.1359+003	4.1504E+03	4.1505E+03	**3.5538e+003**	4.0521E+03	3.6081E+03
14	Schaffer	0.0000E+00	4.4972E+00	4.3354E+00	4.3235E+00	4.6513E+00	**0.2004**	6.5510E-01	6.3624E+00
15	Sphere	0.0000E+00	1.8300E-02	1.7300E-02	2.5000E-03	1.7500E-02	**2.9811E-12**	6.4954e-009	2.6900E-02
16	Bukin	0.0000E+00	3.6973E-03	1.3883E-03	2.9825E-16	3.7531E-18	4.3593E-19	**1.1406E-19**	4.3883E-02

It is obvious from Table 4.6 that, AFSA, mAFSA and all the mCAFAC variants outperformed the ABC in all the optimization test cases except in test case 4 and 5 where all the algorithms produced the same result. This superiority of AFSA over ABC is expected because the AFSA has more parameters that guide its evolution towards the global solution than the ABC.

Note that the algorithms with the best and similar performance with respect to the global solution are shown in bold and italic characters in the tables respectively.

4.4 Parameter Settings for the Proposed Algorithm

Although the algorithm was made user dependent, but for the purpose of validation and comparison, the parameters used by (Wu et al., 2011) were adopted. These parameters are

shown in Table 4.7 as follows:

Table 4.7: Parameter Settings for AFSA and mCAFAC

SN:	Parameters	Values
1	Population (N)	40
2	Dimension	30
3	Visual	250
4	Step	225
5	Crowdedness	24
6	Try_num	10

4.5 Artificial Bee Colony Parameters

Similarly, Artificial Bee Colony Algorithm was made user dependent, but for the purpose of validation, the following parameters were used. These parameters are shown in Table 4.8

Table 4.8: Parameter Settings for ABC

SN:	Parameters	Values
1	Population	80
2	Dimension	60
3	max_cycle	3000
4	Food Number	40

4.6 Computer System Specification

For each system and for each test case, the average of ten (10) tests were performed on COMPAQ Presario CQ56 with an Intel(R) Celeron(R) 2.20GHz processor and 3.00GB RAM with 64-bit Windows 8 pro Operating System (OS).

Some of the important trends are:

i) Variants that used knowledge to control only step size or direction did not do well, in general, when compared with systems that coordinated both step size and direction.

ii) Variants that employed some normative knowledge won for eleven (11) out of the 16 test cases. Thus, normative knowledge appears to be the dominant knowledge source for the

optimization functions.

iii) Variants that employed situational knowledge alone were less successful than those that employed normative knowledge alone. The performance of the systems can be improved when the situational knowledge is used to control direction of evolution and normative knowledge is used to guide the step size.

4.7 Application of the Proposed Controller for Optimal Determination of Controller Parameters

The optimal values of Q and R determined using mCAFAC_NsSd variant are given as follows:

$$Q = \begin{bmatrix} 7.0982 & 0.0625 & 0.0544 & 0 & 0 & 0 & 0 & 0.0204 & 0 & 0.0523 \\ 0.0625 & 7.0139 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.0544 & 0 & 7.0725 & 0.0020 & 0 & 0.0279 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.0020 & 7.0438 & 0 & 0.0279 & 0.0068 & 0.0204 & 0 & 0 \\ 0 & 0 & 0 & 0 & 6.9772 & 0 & 0 & 0.0204 & 0 & 0 \\ 0 & 0 & 0.0279 & 0.0279 & 0 & 7.0737 & 0 & 0 & 0.0673 & 0.0523 \\ 0 & 0 & 0 & 0.0068 & 0 & 0 & 7.0968 & 0 & 0 & 0.0523 \\ 0.0204 & 0 & 0 & 0.0204 & 0.0204 & 0 & 0 & 7.0927 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.0673 & 0 & 0 & 6.9786 & 0.0523 \\ 0.0523 & 0 & 0 & 0 & 0 & 0.0523 & 0.0523 & 0 & 0.0523 & 6.9636 \end{bmatrix} \quad (4.1)$$

$$R = 0.2785 \quad (4.2)$$

Equations (4.1) and (4.2) was determine within an average simulation time of 15 seconds.

The values of Q and R that stabilized the QIP system using the conventional method as described by Abdullahi's (Abdullahi, 2014)approach are given as follows:

$$Q_e = \begin{bmatrix} 200 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 250 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 200 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 200 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 200 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (4.3)$$

$$R_e = 1 \quad (4.4)$$

In order to confirm the optimality of Q and R weighting matrices obtained from the modified cultural artificial fish swarm algorithm with crossover (mCAFAC_NsSd), the responses of each pendulum ball were plotted and superimposed with the responses obtained using conventional trial-and-error method. These responses are represented in Figures 4.17- 4.20. Notice that the blue lines show the responses obtained for the LQR (mCAFAC) while the red lines show the responses obtained for the LQR (conventional trial-and-error method) in all the cases.

The response of the first inverted pendulum is shown in Figure 4.17

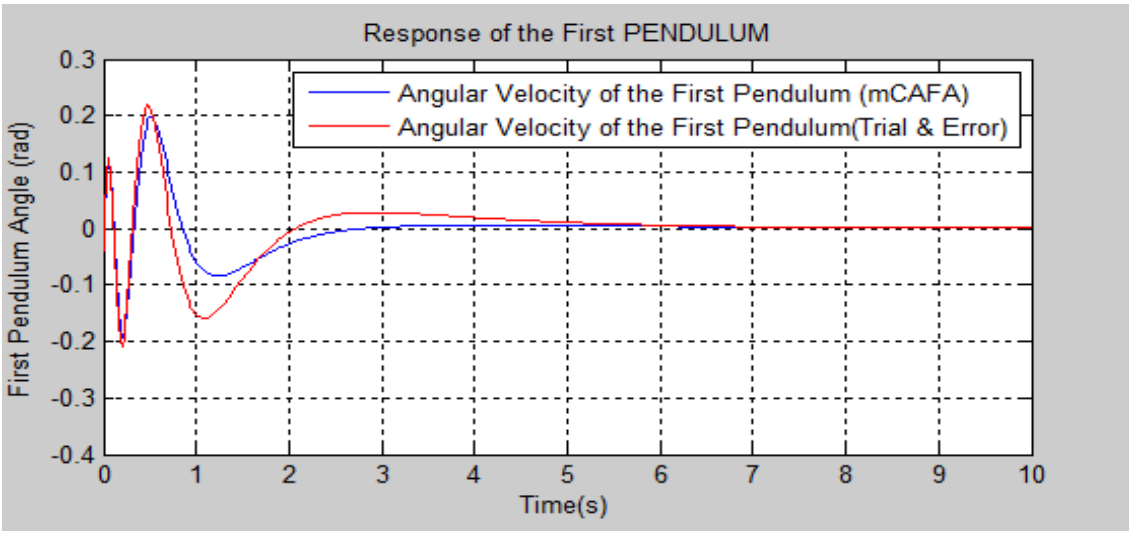


Figure 4.17: Response of the First Pendulum

It can be observed from Figure 4.17 that both LQR (mCAFAC) and LQR (conventional trial-and-error method) stabilized the first pendulum with a settling time of 2.6590seconds and 6.3157seconds respectively.

The response of the second pendulum based on the LQR (mCAFAC) and LQR (conventional trial-and-error method)is as shown in Figure 4.18

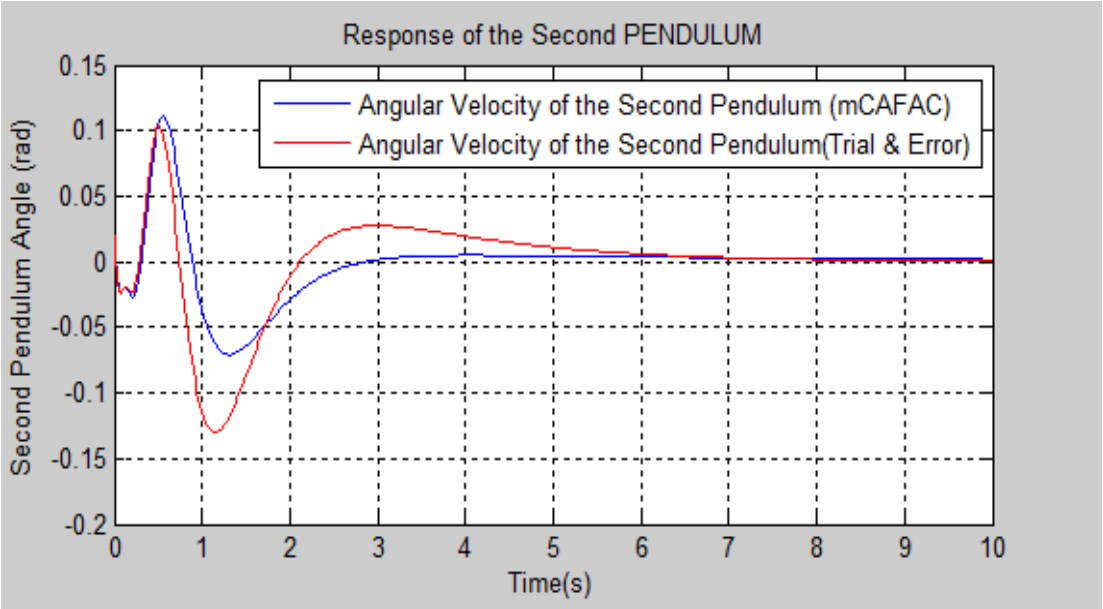


Figure 4.18: Response of the Second Pendulum

It can be observed from Figure 4.18 that both LQR (mCAFAC) and LQR (conventional trial-and-error method) stabilized the second pendulum ball successfully. The settling time of LQR (mCAFAC) and LQR (Conventional trial-and-error)for the second pendulum are 5.7106seconds and 7.0590seconds respectively.

The LQR (mCAFAC) and LQR (conventional trial-and-error method)stabilized the third pendulum ball as shown in Figure 4.19.

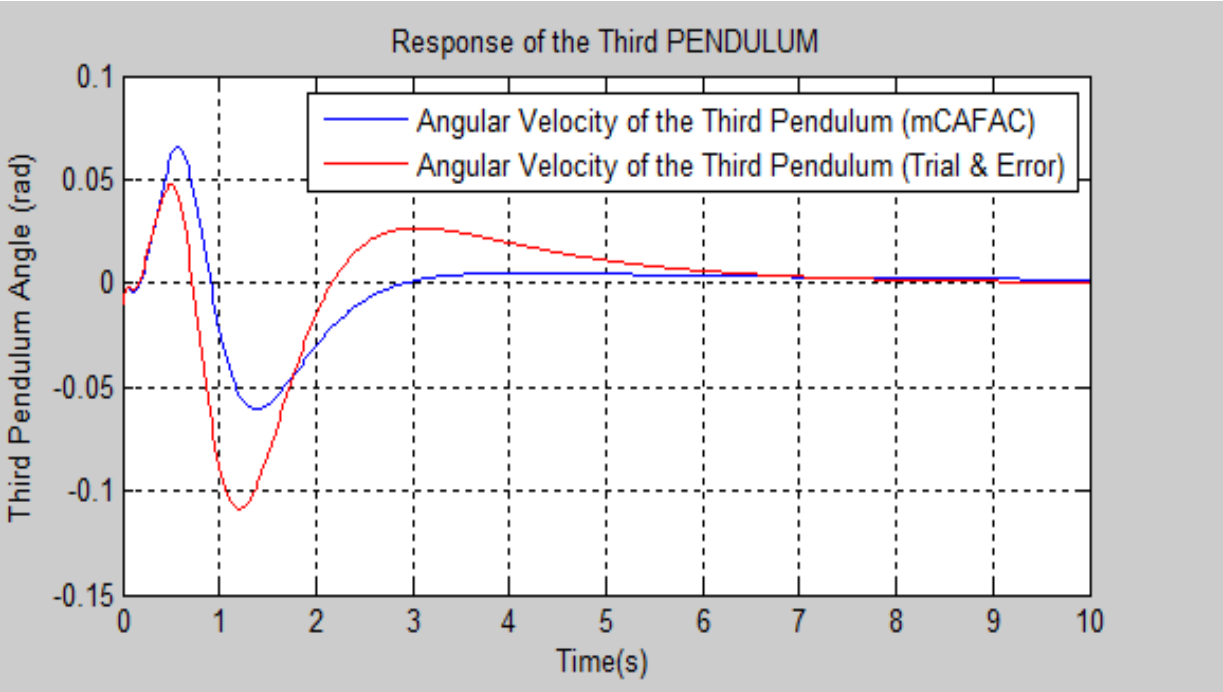


Figure 4.19: Response of the Third Pendulum

It can be observed from Figure 4.19 that both LQR (mCAFAC) and LQR (conventional trial-and-error method) stabilized the third pendulum ball successfully within the same range. With a settling time of 7.1137seconds and 7.3121seconds respectively.

The LQR (mCAFAC) and LQR (conventional trial-and-error method)stabilized the fourth pendulum ball as shown in Figure 4.20 with settling times of 7.4161s and 7.5162s respectively.

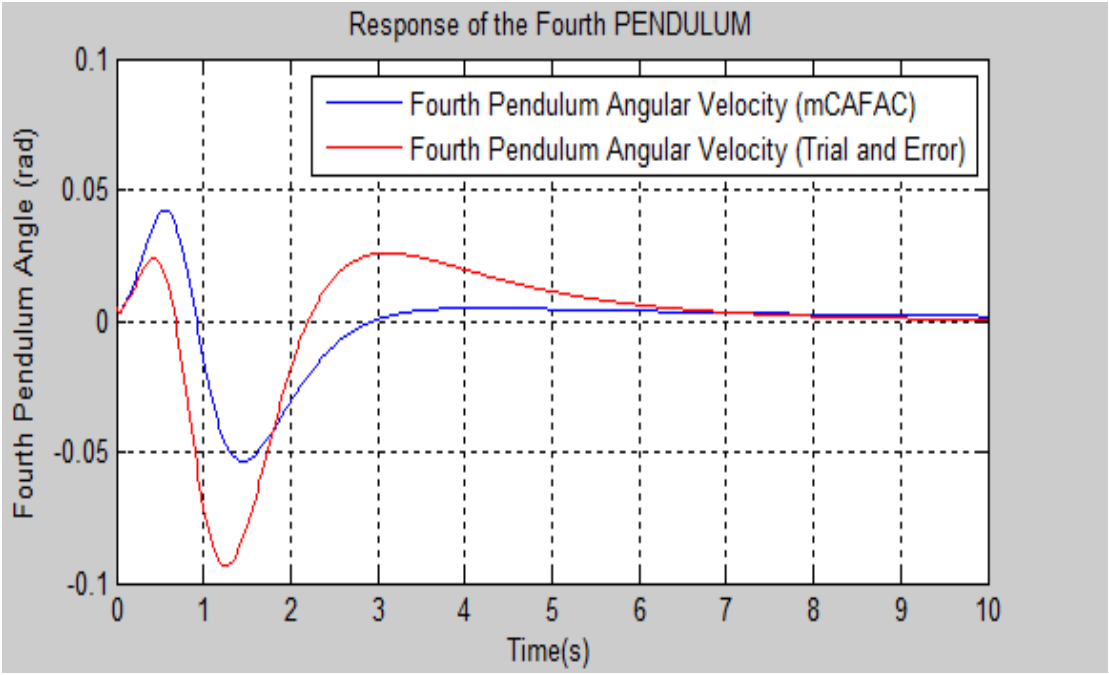


Figure 4.20: Response of the Fourth Pendulum

Detail results obtained from the step response of Figure 4.20 are described in Table 4.12.

Table 4.9 Comparison of Responses for the Fourth Pendulum

SN:	Variable	mCAFAC_NsSd	Trial and Error
1	RiseTime	0.9031	0.6664
2	SettlingTime	7.4161	7.5162
3	SettlingMin	-0.0535	-0.0933
4	SettlingMax	0.0049	0.0259
5	Overshoot (%):	1.457	3.160
6	Peak	0.0535	0.0933
7	PeakTime	1.4600	0.0933

As expected, the settling times (time taken for the QIP to stabilize) using LQR (mCAFAC) and LQR (conventional trial-and-error method) nearly converged showing the convergence of the solution space using either approach. The advantage of the LQR (mCAFAC) approach is that it saves cost in terms of time in arriving at that solution when compared with the conventional approach. This has eliminate the enormous time it takes, trying various values

of Q and R manually. The program used to plot the responses of the QIP can be found in Appendix B.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Introduction

The whole thesisis summarized in this chapter. Conclusion, recommendation and limitation encountered during the course of this research are presented. Areas of future research have also been suggested.

5.2 Summary of Findings

This research has proposed an improved artificial fish swarm algorithm using cultural algorithm, inertial weight and crossover operator, which was aimed at improving the precision accuracy and improved the rate at which the standard algorithm moves towards the global optimal. The performance of the algorithm was verified using sixteen optimization test function and the following findings were reported.

i) The visual and step size has great influence on the performance of the standard AFSA

ii) The introduction of inertial weight (*w*) improved the standard AFSA.

iii) All the four variations of mCAFAC performed better than the standard AFSA with the following performance percentage:

   a) mCAFAC_Ns 18.75%

   b) mCAFAC_Sd 6.25%

   c) mCAFAC_NsSd 56.25%

   d) mCAFAC_NsNd 6.25%

iv) The standard AFSA outperformed, by 78.57% to 21.43%, of the standard ABC over the test functions.

v) The proposed algorithm determined efficiently the optimal values of Linear Quadruple Regulator controllers weighting matrices (Q and R). This has helped in overcoming the trial-and-error approach, which requires a lot of manual iteration and time consuming.

vi) The LQR controller stabilized the Quadruple Inverted Pendulum system efficiently.

5.3 Conclusion

Artificial Fish Swarm Optimization Algorithm (AFSOA) is a class of swarm optimization algorithm which was model based on the preying, swarming and chasing behaviours of fish. In order to overcome the common problems (local minima and poor convergence) associated with AFSA, a modified Cultural Artificial Fish Swarm Algorithm with Crossover (mCAFAC) has been developed, using Cultural Knowledge, Inertia weight and crossover operator. Four versions (mCAFAC_Ns, mCAFAC_Sd, mCAFAC_NsSd and mCAFAC_NsNd) was developed in matlab R2013b and the performance of the developed algorithms was evaluated using sixteen high-dimensional and multi-peak applied mathematical optimization test functions (such as: Ackley, Cosine Mixture, Rastrigin). The result obtained shows that mCAFAC_NsSd has a better precession accuracy and higher ability of escaping local minimal when compared with the standard AFSA, ABC and all the other variations of mCAFAC. The mCAFAC_NsSd was then used to determine the weighting matrices (Q and R) of LQR controller applied in QIP stabilization analysis and the result obtained demonstrated that the algorithm can indeed stabilized the quadruple inverted pendulum. Which is a better alternative to the conventional method of selecting Q and R

input and control weighting matrices respectively. This is an indication of the validity of the mCAFAC.

REFERENCES

Abdullahi, M. (2014). *Modeling, Simulation and Stabilization of a Double Quad Inverted Pendulum System (DQIP) Using a Linear Quadratic Regulator (LQR) Controller.* (Master of Science Degree Thesis), Ahmadu Bello University Zaria, Nigeria., Unpublished.

Akay, B., & Karaboga, D. (2012). A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences, 192*, 120-142.

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.

Ben-Tal, A., & Nemirovski, A. (2001). *Lectures on modern convex optimization: analysis, algorithms, and engineering applications* (Vol. 2): Siam.

Boubaker, O. (2013). The Inverted Pendulum Benchmark in Nonlinear Control Theory: A Survey. *Int J Adv Robotic Sy, 10*(233).

Boubaker, S., & M'sahli, F. (2008). Solutions Based on Particle Swarm Optimization for Optimal Control Problems of Hybrid Autonomous Switched Systems. *International Journal of Intelligent Control and Systems, 13*(2), 128-135.

Changhao, S., & Duan, H. (2013). Artificial bee colony optimized controller for unmanned rotorcraft pendulum. *Aircraft Engineering and Aerospace Technology, 85*(2), 104-114.

Chen, Y., Miao, Q., Zheng, B., Wu, S., & Pecht, M. (2013). Quantitative analysis of lithium-ion battery capacity prediction via adaptive bathtub-shaped function. *Energies, 6*(6), 3082-3096. doi: 10.3390/en6063082

Cheng, Y., Jiang, M., & Yuan, D. (2009). *Novel clustering algorithms based on improved artificial fish swarm algorithm.* Paper presented at the 6th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2009, Tianjin.

Chung, C.-J. (1997). *Knowledge-based approaches to self-adaptation in cultural algorithms.* (Ph.D Dessertation), Weyne State University, Unpublished. E-mail Copy.

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 Vol, 3*(2).

Das, S., Pan, I., Halder, K., & Gupta, A. (2013). *Optimum weight selection based LQR formulation for the design of fractional order PI λ D μ controllers to handle a class of fractional order systems.* Paper presented at the Computer Communication and Informatics (ICCCI), 2013 International Conference on.

De Oliveira, J. F. L., & Ludermir, T. B. (2012). *An evolutionary extreme learning machine based on fuzzy fish swarms.* Paper presented at the 2012 International Conference on Artificial Intelligence, ICAI 2012, Las Vegas, NV.

Eberhart, R. C., & Kennedy, J. (1995). *A new optimizer using particle swarm theory.* Paper presented at the Proceedings of the sixth international symposium on micro machine and human science.

Fang, N., Zhou, J., Zhang, R., Liu, Y., & Zhang, Y. (2014). A hybrid of real coded genetic algorithm and artificial fish swarm algorithm for short-term optimal hydrothermal scheduling. *International Journal of Electrical Power and Energy Systems, 62*, 617-629. doi: 10.1016/j.ijepes.2014.05.017

Gao, X. Z., Wu, Y., Zenger, K., & Huang, X. (2010). *A knowledge-based Artificial Fish-swarm Algorithm.* Paper presented at the 2010 13th IEEE International Conference on Computational Science and Engineering, CSE 2010, Hong Kong.

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), 63-74.

Goldberg, D. E., & Kuo, C. H. (1987). Genetic algorithms in pipeline optimization. *Journal of Computing in Civil Engineering, 1*(2), 128-141.

Guo, W., Fang, G., & Huang, X. (2011). *An improved chaotic artificial fish swarm algorithm and its application in optimizing cascade hydropower stations.* Paper presented at the 2011 International Conference on Business Management and Electronic Information, BMEI 2011, Guangzhou.

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

Hanafy, M., Gomaa, M. M., Taher, M., & Wahba, A. M. (2012). Development of a technology for car's auto-parking using swarm search-based fuzzy control system. *International Journal of Modelling, Identification and Control, 17*(1), 85-97. doi: 10.1504/IJMIC.2012.048643

Hansen, N. (2006). Compilation of results on the 2005 CEC benchmark function set. *Online, May*.

Hasan, M., Saha, C., Rahman, M. M., Sarker, M. R. I., & Aditya, S. K. (2012). Balancing of an inverted pendulum using pd controller. *Dhaka University Journal of Science, 60*(1), 115-120.

Hassan, E. A., Hafez, A. I., Hassanien, A. E., & Fahmy, A. A. (2015). Community Detection Algorithm Based on Artificial Fish Swarm Optimization *Intelligent Systems' 2014* (pp. 509-521): Springer.

Holland, J. (1975). Genetic Algorithms, computer programs that evolve in ways that even their creators do not fully understand. *Scientific American*, 66-72.

Huang, G. Q., Lu, Q. Q., & Zheng, Y. Q. (2006). Optimization approach to the location of oilfield multistage stations by fish-swarm algorithm. *Xi'an Shiyou Daxue Xuebao (Ziran Kexue Ban)/Journal of Xi'an Shiyou University, Natural Sciences Edition, 21*(4), 98-102.

Huang, Y., Xu, Y., & Chen, G. (2008). *The culture-based particle swarm optimization algorithm.* Paper presented at the 4th International Conference on Natural Computation, ICNC 2008, Jinan.

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.

Jiang, J., Bo, Y., Song, C., & Bao, L. (2012) Hybrid algorithm based on particle swarm optimization and artificial fish swarm algorithm. & U. Northeastern, S. Institute of Automation of the Chinese Academy of, K. Chinese University of Hong & C. University of Illinois at (Vol. Ed.)*: Vol. 7367 LNCS. 9th International Symposium on Neural Networks, ISNN 2012* (pp. 607-614). Shenyang.

Jiang, M., & Cheng, Y. (2010). *Simulated annealing artificial fish swarm algorithm.* Paper presented at the 2010 8th World Congress on Intelligent Control and Automation, WCICA 2010, Jinan China.

Jin, X., & Reynolds, R. G. (1999). Using Knowledge-Based Evolutionary Computation to Solve Nonlinear Constraint.pdf>. *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*, 1672-1678.

Jones, G., Willett, P., & Glen, R. C. (1995). Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *Journal of molecular biology, 245*(1), 43-53.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization: Technical report-tr06, Erciyes university, engineering faculty, computer engineering department.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization, 39*(3), 459-471.

Karaboga, N. (2009). A new design method based on artificial bee colony algorithm for digital IIR filters. *Journal of the Franklin Institute, 346*(4), 328-348.

Li, X., Shao, Z., & Qian, J. (2002). An optimization searching model based on animal autonomous body: Artificial Fish Swarm Algorithm, system engineering theory and practice.

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, pp 7*, 33.

Ma, J. W., & Zhang, G. L. (2005). Application of artificial fish-swarm neural network in short-term load forecasting of power system. *Power System Technology, 29*(11), 36-39.

Mingyan, J., & Dongfeng, Y. (2005). *Wavelet threshold optimization with artificial fish swarm algorithm.* Paper presented at the 2005 International Conference on Neural Networks and Brain Proceedings, ICNNB'05, Beijing.

Mobayen, S., Rabiei, A., Moradi, M., & Mohammady, B. (2011). Linear quadratic optimal control system design using particle swarm optimization algorithm. *International Journal of Physical Sciences, 6*(30), 6958-6966. doi: 10.5897/IJPS11.726

Monfort, M., Liu, A., & Ziebart, B. D. (2015). Intent Prediction and Trajectory Forecasting via Predictive Inverse Linear-Quadratic Regulation.

Moradi, M., & Abedini, M. (2012). A combination of genetic algorithm and particle swarm optimization for optimal DG location and sizing in distribution systems. *International Journal of Electrical Power & Energy Systems, 34*(1), 66-74.

Neshat, M., Sepidnam, G., Sargolzaei, M., & Toosi, A. N. (2012). Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications. *Artificial Intelligence Review*, 1-33. doi: 10.1007/s10462-012-9342-2

Nguyen, T. T., & Yao, X. Hybridizing Cultural Algorithms and Local Search.pdf. In B. T. University of Birmingham, United Kingdom (Ed.), *The Centre of Excellence for Research in Computational Intelligence and Applications(CERCIA), School of Computer Science,* : School of Computer Science, University of Birmingham, B15 2TT, United Kingdom

Orivuori, J., Zenger, K., & Sinervo, A. (2010). Active control of rotor vibrations by advanced control methods. *Proceedings of the 16th International Congress on Sound and Vibration- Recent Developments in Acoustics, Noice and Vibration*.

Ozturk, C., Hancer, E., & Karaboga, D. (2015). Dynamic clustering with improved binary artificial bee colony algorithm. *Applied soft computing, 28*, 69-80.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *Control Systems, IEEE, 22*(3), 52-67.

Peng, J., Li, X., Qin, Z. Q., & Luo, G. (2013). Robot global path planning based on improved artificial fish-swarm algorithm. *Research Journal of Applied Sciences, Engineering and Technology, 5*(6), 2042-2047.

Reynolds, R. G. (1994). *An introduction to cultural algorithms.* Paper presented at the Proceedings of the third annual conference on evolutionary programming.

Reynolds, R. G., & Chung, C. (1996). *A self-adaptive approach to representation shifts in cultural algorithms.* Paper presented at the Evolutionary Computation, 1996., Proceedings of IEEE International Conference on.

Reynolds, R. G., & J., C. C. (1997). Fuzzy Approaches to Acquiring Experimental Knowledge.pdf. *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*(97), 260-267.

Reynolds, R. G., & Peng, B. (2004). Cultural Algorithms Modeling of How Cultures Learn to Solve Problems.pdf. *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence*.

Saberi, M., Altafi, H., & Alizadeh, S. M. (2012). Control of the Magnetic Suspension System with a Three-degree-of-freedom Using RBF Neural Network Controller. *International Journal of Computer and Electrical Engineering, 4*(2).

Schildbach, G., Goulart, P., & Morari, M. (2015). Linear controller design for chance constrained systems. *Automatica, 51*, 278-284.

Singh, A. (2009). An artificial bee colony algorithm for the leaf-constrained minimum spanning tree problem. *Applied soft computing, 9*(2), 625-631.

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

Tao, Y., & Han, W. (2014) An improved artificial fish-school algorithm based on group behaviors and self-adaptive mutation. & U. Korea Maritime, U. Inha & C. Hong Kong Industrial Technology Research (Vol. Ed.)*: Vol. 462-463. 2013 International Conference on Mechatronics and Information Technology, ICMIT 2013* (pp. 360-364). Guilin.

Toushmalani, R. (2013). Gravity inversion of a fault by Particle swarm optimization (PSO). *SpringerPlus, 2*(1), 1-7. doi: 10.1186/2193-1801-2-315

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*.

Wang, J., Wang, X., & Huang, M. (2008). *Tabu Artificial Fish Swarm algorithm based intelligent QoS multicast routing algorithm.* Paper presented at the 2008 International Conference on Computational Intelligence and Security, CIS 2008, Suzhou.

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.

Wu, Y., Gao, X. Z., & Zenger, K. (2011). *Knowledge-based Artificial Fish-Swarm algorithm.* Paper presented at the 18th IFAC World Congress, Milano.

Wu, Y., Huang, X. L., Gao, X. Z., & Zenger, K. (2012). Cultural artificial fish-swarm optimization algorithm and application in the parameters identification of rotor system. *Dianji yu Kongzhi Xuebao/Electric Machines and Control*, *16*(5), 102-108.

Xiaoping, W., & Liming, C. (2002). Genetic algorithm-theory, application and software implementation [M]. *Xi'an Jiao tong University Press, Xi'an*.

Xu, J., Zhang, Y., & Wang, P. (2014). A kind of turbine fault diagnosis based on fish-swarm algorithm. *Energy Education Science and Technology Part A: Energy Science and Research, 32*(2), 1325-1330.

Xu, Y. N., & Long, J. (2014) Research on game optimized artificial Fish-swarm algorithm of communication signal processing. & E. HongKong Control, Information, a. International Frontiers of science, A. Science Research, A. technology Research & C. X. C. C. L. YaoWenli (Vol. Ed.)*: Vol. 912-914. 2014 International Conference on Frontiers of Advanced Materials and Engineering Technology, FAMET 2014* (pp. 1357-1360): Trans Tech Publications.

Xue, P. (2014) Adaptive hybrid artificial fish school algorithm for solving the real roots of polynomials. & I. C. I. Society, S. International Neural Network & C. National Science Foundation of (Vol. Ed.)*: Vol. 8588 LNCS. 10th International Conference on Intelligent Computing, ICIC 2014* (pp. 48-54). Taiyuan: Springer Verlag.

Yang, X.-S. (2010). Firefly algorithm, stochastic test functions and design optimisation. *International Journal of Bio-Inspired Computation, 2*(2), 78-84.

Yanmin, L., & Zhibin, F. (2013). *The optimization of fuzzy neural network based on artificial fish swarm algorithm.* Paper presented at the 9th IEEE International Conference on Mobile Ad-Hoc and Sensor Networks, MSN 2013, Dalian.

Yazdani, D., Golyari, S., & Meybodi, M. R. (2010). *A new hybrid algorithm for optimization based on artificial fish swarm algorithm and cellular learning automata.* Paper presented at the 2010 5th International Symposium on Telecommunications, IST 2010, Tehran.

Yazdani, D., Saman, B., Sepas-Moghaddam, A., Mohammad-Kazemi, F., & Reza Meybodi, M. (2013). A new algorithm based on improved artificial fish swarm

algorithm for data clustering. *International Journal of Artificial Intelligence, 11*(13 A), 193-221.

Yi, X., Ma, L. X., Cai, W. F., & Sun, J. (2014) Research of multi-objective power network planning based on improved fish swarm algorithm. & U. Chung Hua, S. Harbin University of, Technology, U. Northwest Forestry & U. Harbin Engineering (Vol. Ed.)*: Vol. 494-495. 2013 International Symposium on Vehicle, Mechanical, and Electrical Engineering, ISVMEE 2013* (pp. 1735-1738). Taiwan.

Yu, L., & Li, C. (2014). A global artificial fish swarm algorithm for structural damage detection. *Advances in Structural Engineering, 17*(3), 331-346. doi: 10.1260/1369-4332.17.3.331

Zeng, C., & Xu, H. (2012). *PID controller parameters optimization based on artificial fish swarm algorithm.* Paper presented at the 2012 5th International Conference on Intelligent Computation Technology and Automation, ICICTA 2012, Zhangjiajie, Hunan.

Zhou, Y., Zhou, G., & Zhang, J. (2013). A hybrid glowworm swarm optimization algorithm for constrained engineering design problems. *Applied Mathematics and Information Sciences, 7*(1), 379-388.

Zhou, Y. L., He, Q. Y., & Xu, M. F. (2010). Application of artificial fish-swarm algorithm in gas-solid fluidized bed flow regime identification. *Huaxue Gongcheng/Chemical Engineering (China), 38*(7), 39-42+50.

Zhu, G., & Kwong, S. (2010). Gbest-guided artificial bee colony algorithm for numerical function optimization. *Applied Mathematics and Computation, 217*(7), 3166-3173.

Zhu, K., & Jiang, M. (2010). *Quantum Artificial Fish Swarm Algorithm.* Paper presented at the 2010 8th World Congress on Intelligent Control and Automation, WCICA 2010, Jinan.

Zhu, W., & Lin, M. (2013). Weak signal intelligent detection system based on stochastic resonance and artificial fish swarm algorithm. *Yi Qi Yi Biao Xue Bao/Chinese Journal of Scientific Instrument, 34*(11), 2464-2470.

APPENDIX A1

m-FILE Code of AFSA.

```
tic
if FISHS==0
pop=rand(N,D);
pop=sort(pop);
else
   pop=FISHS;
end
popi=pop;%Initial positions of the AF and start with preying conditions
fitpopi=objfunc(fitness,popi);
for i=1:N
Try=0;
Itr_max=Itr;
iteration = 0;
while Try<try_num
   iteration = iteration+1;
   Try=Try+1;
   popj(i,(1:D))=popi(i,(1:D))+rand*visual;%randomly selected position
   fitpopj(i,1)=objfunc(fitness,popj(i,(1:D)));
   if fitpopj(i,1)<fitpopi(i,1)
      %Execute preying
   popi(i,(1:D))=popi(i,(1:D))+rand*step*(popj(i,(1:D))-
popi(i,(1:D)))/sum((abs(popj(i,(1:D)).^2-popi(i,(1:D)).^2)).^0.5);
   Try=try_num;
   end
visual=ceil(visual*(Itr_max-iteration)/Itr_max);
step=ceil(step*(Itr_max-iteration)/Itr_max);
try_num=ceil(try_num*(Itr_max-iteration)/Itr_max);
end
if fitpopj(i,1)>fitpopi(i,1)
popi(i,(1:D))=popi(i,(1:D))+rand*step;%Execute if preying fails
end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%start
Swarming%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
popii=pop;
fitpopii=objfunc(fitness,popii);
```

```
popk=popii+rand*visual;% represent next positions of AF.
for hh=1:N
for h=1:N
   Xcentre=zeros(1,D);
   nf=0;
   if sum(abs(popk(h,:).^2-popii(h,:).^2).^0.5)<=visual %Conditions for  number of fellows
btw visual distance
      nf=nf+1;
      B=popk(h,:);
      Xcentre=Xcentre+B;
   end
end
Xcentre=Xcentre/nf;
Ycentre=objfunc(fitness,Xcentre);
if nf*Ycentre<zig*fitpopii(hh)
   if Ycentre<fitpopii(hh)
      popii(hh,:)=popii(hh,:)+rand*step*(Xcentre-popii(hh,:))/sum((abs(Xcentre.^2-
popii(hh,:).^2)).^0.5);%Uncrowded area
   else
clear i;
for i=1:N
Try=0;
Itr_max=Itr;
iteration=0;
while Try<try_num
   iteration = iteration+1;
   Try=Try+1;
   popk(i,(1:D))=popii(i,(1:D))+rand*visual;
   fitpopk(i,1)=objfunc(fitness,popk(i,(1:D)));
   if fitpopk(i,1)<fitpopii(i,1)
   popii(i,(1:D))=popii(i,(1:D))+rand*step*(popk(i,(1:D))-
popii(i,(1:D)))/sum((abs(popk(i,(1:D)).^2-popii(i,(1:D)).^2)).^0.5);%Execute Swarming
   Try=try_num;
   end
   visual=ceil(visual*(Itr_max-iteration)/Itr_max);
   step=ceil(step*(Itr_max-iteration)/Itr_max);
   try_num=ceil(try_num*(Itr_max-iteration)/Itr_max);
end
if fitpopk(i,1)>fitpopii(i,1)
popii(i,(1:D))=popii(i,(1:D))+rand*step;
end
end
   end
end
```

```
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Start
Chasing%%%%%%%%%%%%%%%%%%%%%%%%%%%%
popiii=pop;
fitpopiii=objfunc(fitness,popii);
popl=popiii+rand*visual;
clear i hh h
for hh=1:N
for h=1:N
   Xmin=[];
   nf=0;
   if  sum(abs(popl(h,:).^2-popiii(hh,:).^2).^0.5)<=visual
      nf=nf+1;
      Xmin=[Xmin;popl(h,:)];
   end
end
clear h;
if nf~=0
for h=1:nf-1
   if  sum(abs(popl(h+1,:).^2-popiii(h+1,:).^2).^0.5)<sum((abs(popl(h,:).^2-
popiii(h,:).^2)).^0.5)
      Xmin=Xmin(h+1,:);
   else
       Xmin=Xmin(h,:); %represent the best artificial fish  individual withing visual distance.
   end
end
end
Ymin=objfunc(fitness,Xmin);
if nf*Ymin<zig*fitpopiii(hh)
   if Ymin<fitpopiii(hh)
      popiii(hh,:)=popiii(hh,:)+rand*step*(Xmin-popiii(hh,:))/sum((abs(Xmin.^2-
popiii(hh,:).^2)).^0.5);
   else
clear i;
for i=1:N
Try=0;
Itr_max=Itr;
iteration=0;
while Try<try_num
   iteration=iteration+1;
   Try=Try+1;
   popl(i,(1:D))=popiii(i,(1:D))+rand*visual;
   fitpopl(i,1)=objfunc(fitness,popl(i,(1:D)));
```

```
    if fitpopl(i,1)<fitpopiii(i,1)
    popiii(i,(1:D))=popiii(i,(1:D))+rand*step*(popl(i,(1:D))-
popiii(i,(1:D)))/sum((abs(popl(i,(1:D)).^2-popiii(i,(1:D)).^2)).^0.5);
    Try=try_num;
    end
    visual=ceil(visual*(Itr_max-iteration)/Itr_max);
    step=ceil(step*(Itr_max-iteration)/Itr_max);
    try_num=ceil(try_num*(Itr_max-iteration)/Itr_max);
end
if fitpopl(i,1)>fitpopiii(i,1)
popiii(i,(1:D))=popiii(i,(1:D))+rand*step;
end
end
    end
end
end
```

APPENDIX A2

m-FILE for Artificial Bee Colony (ABC) Algorithm

```
function ABC
tic
if BEE==0
popBee = rand(FoodNumber,D);
popBee=sort(popBee);
else
   popBee=BEE;
end
ObjVal=objfun(fitness,popBee);
trial=0;
for r=1:runtime
trial=zeros(1,FoodNumber);
BestInd=find(ObjVal==min(ObjVal));
BestInd=BestInd(end);
GlobalMin=ObjVal(BestInd);
GlobalParams=popBee(BestInd,:);
iter=0;
while((iter <= maxCycle)),
   iter=iter+1;
   for i=1:(FoodNumber)
      P_C=fix(rand*D)+1;
      neighbour=fix(rand*(FoodNumber))+1;
         while(neighbour==i)
            neighbour=fix(rand*(FoodNumber))+1;
         end;
      sol=popBee(i,:);
      sol(P_C)=popBee(i,P_C)+(popBee(i,P_C)-popBee(neighbour,P_C))*(rand-0.5)*2;
       ind=find(sol<lb);
       sol(ind)=lb(ind);
       ind=find(sol>ub);
       sol(ind)=ub(ind);
       ObjValSol=objfunc(fitness,sol);
       FitnessSol=calculateFitness(ObjValSol);
      if (FitnessSol>Fitness(i))
```

```
            popBee(i,:)=sol;
            Fitness(i)=FitnessSol;
            ObjVal(i)=ObjValSol;
            trial(i)=0;
         else
            trial(i)=trial(i)+1;
        end;
    end;
prob=(0.9.*Fitness./max(Fitness))+0.1;
i=1;
t=0;
while(t<FoodNumber)
    if(rand<prob(i))
        t=t+1;
        P_C=fix(rand*D)+1;
              neighbour=fix(rand*(FoodNumber))+1;
            while(neighbour==i)
              neighbour=fix(rand*(FoodNumber))+1;
            end;
       sol=popBee(i,:);
       sol(P_C)=popBee(i,P_C)+(popBee(i,P_C)-popBee(neighbour,P_C))*(rand-0.5)*2;
        ind=find(sol<lb);
        sol(ind)=lb(ind);
        ind=find(sol>ub);
        sol(ind)=ub(ind);
        ObjValSol=objfunc(fitness,sol);
        FitnessSol=calculateFitness(ObjValSol);
       if (FitnessSol>Fitness(i))
            popBee(i,:)=sol;
            Fitness(i)=FitnessSol;
            ObjVal(i)=ObjValSol;
            trial(i)=0;
         else
            trial(i)=trial(i)+1;
        end;
    end;

    i=i+1;
    if (i==(FoodNumber)+1)
        i=1;
    end;
end;
         ind=find(ObjVal==min(ObjVal));
         ind=ind(end);
```

```
        if (ObjVal(ind)<GlobalMin)
        GlobalMin=ObjVal(ind);
        GlobalParams=popBee(ind,:);
        end;
ind=find(trial==max(trial));
ind=ind(end);
if (trial(ind)>limit)
   trial(ind)=0;
   sol=(ub-lb).*rand(1,D)+lb;
   ObjValSol=objfunc(fitness,sol);
   popBee(ind,:)=sol;
   Fitness(ind)=FitnessSol;
   ObjVal(ind)=ObjValSol;
end;
fprintf('TJ_iter = %d ObjVal = %g\n',iter,GlobalMin);
end GlobalMins(r)=GlobalMin;
end;
```

APPENDIX B1

m-FILE for mCAFAC_Ns

```
% Start of the Modified Cultural Artificial Fish Swarm Algorithm With
% Crossover
popnew=population;%Obtained from Artificial AFSA
Beta=input('ENTER THE VALUE FOR B (% of the accepted Fish)\n or press zero (0) for
default value.\nBeta =  ');
Try=0;
%Preying Behaviour
if Ns==1
if Sd~=1
if Nd~=1
while Try<try_num
  Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
clear i j
for i=1:aceptance
   for j=1:D
   if fitpopnewpj(i,1)<fitpopnewpi(i,1)
      if ((((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2)))^0.5)))~=0
   popnewpi(i,j)=popnewpi(i,j)+rand*(uk(j)-lk(j))*(popnewpj(i,j)-
popnewpi(i,j))/((((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2)))^0.5)));
      end
   end
   end
end
end
end
end
end
%.....................SWARMING INFLUENCE FUNCTION.....................
Xcentre=zeros(1,D);
while Try<try_num
  Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
```

```
nf=0;
for h=1:N
   if (((((abs((sum(popnewsj(h,:).^2-popnewsi(h,:).^2)))))^0.5)))<=visual
      nf=nf+1;
      centre=popnewsj(h,:);
      Xcentre=Xcentre+centre;
   end
end
Xcentre=Xcentre/nf;
clear i j
for i=1:aceptance
   for j=1:D
      if (((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2)))))^0.5)))~=0
   popnewsi(i,j)=popnewsi(i,j)+rand*(uk(j)-lk(j))*(Xcentre(j)-
popnewsi(i,j))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
      end
   end
end
end
end
end
end
Xcentre=zeros(1,D);

%%%%%%%%%%%%%%%%%%%%Chasing Influence
Function%%%%%%%%%%%%%%%%%%%%
clear i j
Xmin=[];
if Sd==1
if Ns~=1
if Nd~=1
   Try=0;
while Try<try_num
   Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
Xmin=[];
nf=0;
Xmin=[];
for h=1:N
   if  (((abs((sum(popnewcj(h,:).^2-popnewci(h,:).^2))))^0.5))<=visual
      nf=nf+1;
      Xmin=[Xmin;popnewcj(h,:)];
   end
end
```

```
if nf~=0
   if  ((((abs((sum(popnewcj(nf,:).^2-popnewci(nf,:).^2)))))^0.5))<((((abs((sum(popnewcj(nf-
1,:).^2-popnewci(nf-1,:).^2)))))^0.5))
      Xmin=Xmin((nf),:);
      nf=nf-1;
   else
       Xmin=Xmin(1,:);
   end
end
clear i j
for i=1:aceptance
   for j=1:D
      if ((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2)))))^0.5))~=0
   if popnewsj(i,j)<S(j)
   popnewci(i,j)=popnewci(i,j)+abs(rand*step*(Xmin(j)-
popnewci(i,j)))/((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2))))).^0.5));
   end
   if popnewsj(i,j)>S(j)
   popnewci(i,j)=popnewci(i,j)-abs(rand*step*(Xmin(j)-
popnewci(i,j)))/((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2))))).^0.5));
   else
   popnewci(i,j)=popnewci(i,j)+rand*step*(Xmin(j)-popnewci(i,j))/((((abs((sum(Xmin.^2-
popnewci(i,(1:D)).^2))))).^0.5));
   end
      end
   end
end
end
end
end
end
```

APPENDIX B2

m-FILE for mCAFAC_Sd

```
%Preying Behaviour
clear i j
if Sd==1
if Ns~=1
if Nd~=1
   Try=0;
while Try<try_num

  Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
clear i j
for i=1:aceptance
   for j=1:D
      if (((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2)))^0.5))~=0
   if popnewpj(i,j)<S(j)
   popnewpi(i,j)=popnewpi(i,j)+abs(rand*step*(popnewpj(i,j)-
popnewpi(i,j)))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   end
   if popnewpj(i,j)>S(j)
   popnewpi(i,j)=popnewpi(i,j)-abs(rand*step*(popnewpj(i,j)-
popnewpi(i,j)))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   else
   popnewpi(i,j)=popnewpi(i,j)+rand*step*(popnewpj(i,j)-
popnewpi(i,j))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   end
      end
   end
end
end
end
end
end
```

```
%%%%%%%%%%%%%%%%%%%%%Swarming Influence Function
&&&&&&&&&&&&&&&&&&&&&&
while Try<try_num
  Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
nf=0;
for h=1:N
   if (((((abs((sum(popnewsj(h,:).^2-popnewsi(h,:).^2)))))^0.5)))<=visual
      nf=nf+1;
      centre=popnewsj(h,:);
      Xcentre=Xcentre+centre;
   end
end
Xcentre=Xcentre/nf;
clear i j
for i=1:aceptance
   for j=1:D
      if ((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2)))))^0.5))~=0
   if popnewsj(i,j)<S(j)
   popnewsi(i,j)=popnewsi(i,j)+abs(rand*step*(Xcentre(j)-
popnewsi(i,j)))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   end
   if popnewsj(i,j)>S(j)
   popnewsi(i,j)=popnewsi(i,j)-abs(rand*step*(Xcentre(j)-
popnewsi(i,j)))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   else
   popnewsi(i,j)=popnewsi(i,j)+rand*step*(Xcentre(j)-
popnewsi(i,j))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   end
      end
   end
end
end
end
end
end
Xcentre=zeros(1,D);
%%%%%%%%%%%%%%%%%%%% CHASING INFLUENCE FUNCTION
%%%%%%%%%%%%%%%%%%%%
while Try<try_num
  nf=0;
Xmin=[];
for h=1:N
   if  (((abs((sum(popnewcj(h,:).^2-popnewci(h,:).^2))))^0.5))<=visual
```

```
        nf=nf+1;
        Xmin=[Xmin;popnewcj(h,:)];
    end
end
if nf~=0
    if  ((((abs((sum(popnewcj(nf,:).^2-popnewci(nf,:).^2)))))^0.5))<((((abs((sum(popnewcj(nf-
1,:).^2-popnewci(nf-1,:).^2)))))^0.5))
        Xmin=Xmin((nf),:);
        nf=nf-1;
    else
         Xmin=Xmin(1,:);
    end
end
Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
clear i j
for i=1:aceptance
    for j=1:D
        if ((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2)))))^0.5))~=0
    popnewci(i,j)=popnewci(i,j)+rand*(uk(j)-lk(j))*(Xmin(j)-
popnewci(i,j))/((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2))))).^0.5));
        end
    end
end
end
end
    end
end
```

APPENDIX B3

m-FILE for mCAFAC_NsSd

```
%Preying Behaviour
if Ns==1
if Sd==1
if Nd~=1
   Try=0;
while Try<try_num
 Try=Try+1;
aceptance=N*Beta+floor((N*Beta)/Try);
clear i j
for i=1:aceptance
for j=1:D
      if (((((abs((sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2)))))^0.5)))~=0
   if popnewpj(i,j)<S(j)
   popnewpi(i,j)=popnewpi(i,j)+abs(rand*(uk(j)-lk(j))*(popnewpj(i,j)-
popnewpi(i,j)))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   end
   if popnewpj(i,j)>S(j)
   popnewpi(i,j)=popnewpi(i,j)-abs(rand*(uk(j)-lk(j))*(popnewpj(i,j)-
popnewpi(i,j)))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   else
   popnewpi(i,j)=popnewpi(i,j)+rand*(uk(j)-lk(j))*(popnewpj(i,j)-
popnewpi(i,j))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   end
      end
end
end
end
end
end
end
```

```
&&&&&&&&&&&&&&&&&Swarming Influence
function&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
while Try<try_num
  Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
nf=0;
for h=1:N
   if ((((abs((sum(popnewsj(h,:).^2-popnewsi(h,:).^2)))))^0.5))<=visual
      nf=nf+1;
      centre=popnewsj(h,:);
      Xcentre=Xcentre+centre;
   end
end
Xcentre=Xcentre/nf;
clear i j
for i=1:aceptance
   for j=1:D
      if ((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2)))))^0.5))~=0
   if popnewsj(i,j)<lb(j)
   popnewsi(i,j)=popnewsi(i,j)+abs(rand*(uk(j)-lk(j))*(Xcentre(j)-
popnewsi(i,j)))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   end
   if popnewsj(i,j)>ub(j)
   popnewsi(i,j)=popnewsi(i,j)-abs(rand*(uk(j)-lk(j))*(Xcentre(j)-
popnewsi(i,j)))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   else
   popnewsi(i,j)=popnewsi(i,j)+rand*(uk(j)-lk(j))*(Xcentre(j)-
popnewsi(i,j))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   end
      end
   end
end
end
end
end
end
clear i j
Xcentre=zeros(1,D);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%Chasing
Behaviour%%%%%%%%%%%%%%%%%%%%%%%%%%
while Try<try_num
   Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
nf=0;
```

```
Xmin=[];
for h=1:N
   if  (((abs((sum(popnewcj(h,:).^2-popnewci(h,:).^2))))^0.5))<=visual
      nf=nf+1;
      Xmin=[Xmin;popnewcj(h,:)];
   end
end
if nf~=0
   if  ((((abs((sum(popnewcj(nf,:).^2-popnewci(nf,:).^2)))))^0.5))<((((abs((sum(popnewcj(nf-
1,:).^2-popnewci(nf-1,:).^2)))))^0.5))
      Xmin=Xmin((nf),:);
      nf=nf-1;
   else
       Xmin=Xmin(1,:);
   end
end
clear i j
for i=1:aceptance
   for j=1:D
      if ((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2)))))^0.5))~=0
   if popnewsj(i,j)<lb(j)
   popnewci(i,j)=popnewci(i,j)+abs(rand*(uk(j)-lk(j))*(Xmin(j)-
popnewci(i,j)))/((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2))))).^0.5));
   end
   if popnewsj(i,j)>ub(j)
   popnewci(i,j)=popnewci(i,j)-abs(rand*(uk(j)-lk(j))*(Xmin(j)-
popnewci(i,j)))/((((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2)))))).^0.5));
   else
   popnewci(i,j)=popnewci(i,j)+rand*(uk(j)-lk(j))*(Xmin(j)-
popnewci(i,j))/((((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2)))))).^0.5));
   end
      end
   end
end
end
end
end
end
```

APPENDIX B4

m-FILE for mCAFAC_NsNd

```
%Preying Behaviour
if Ns==1
if Sd~=1
if Nd==1
   Try=0;
while Try<try_num
  Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
clear i j
for i=1:aceptance
   for j=1:D
      if (((((abs((sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2)))))^0.5)))~=0
   if popnewpj(i,j)<lb(j)
   popnewpi(i,j)=popnewpi(i,j)+abs(rand*(uk(j)-lk(j))*(popnewpj(i,j)-
popnewpi(i,j)))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   end
   if popnewpj(i,j)>ub(j)
   popnewpi(i,j)=popnewpi(i,j)-abs(rand*(uk(j)-lk(j))*(popnewpj(i,j)-
popnewpi(i,j)))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   else
   popnewpi(i,j)=popnewpi(i,j)+rand*(uk(j)-lk(j))*(popnewpj(i,j)-
popnewpi(i,j))/(((abs(sum(popnewpj(i,(1:D)).^2-popnewpi(i,(1:D)).^2))).^0.5));
   end
      end
   end
end
end
end
```

```
end
end
%%%%%%%%%%%%%%%%%%%%Swarming
Behaviour%%%%%%%%%%%%%%%%%%
while Try<try_num
  Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
nf=0;
for h=1:N
   if ((((abs((sum(popnewsj(h,:).^2-popnewsi(h,:).^2)))))^0.5))<=visual
      nf=nf+1;
      centre=popnewsj(h,:);
      Xcentre=Xcentre+centre;
   end
end
Xcentre=Xcentre/nf;
clear i j
for i=1:aceptance
   for j=1:D
      if ((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2)))))^0.5))~=0
  if popnewsj(i,j)<lb(j)
   popnewsi(i,j)=popnewsi(i,j)+abs(rand*(uk(j)-lk(j))*(Xcentre(j)-
popnewsi(i,j)))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   end
   if popnewsj(i,j)>ub(j)
   popnewsi(i,j)=popnewsi(i,j)-abs(rand*(uk(j)-lk(j))*(Xcentre(j)-
popnewsi(i,j)))/((((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2))))).^0.5));
   else
   popnewsi(i,j)=popnewsi(i,j)+rand*(uk(j)-lk(j))*(Xcentre(j)-
popnewsi(i,j))/(((abs((sum(Xcentre.^2-popnewsi(i,(1:D)).^2)))).^0.5));
   end
      end
   end
end
end
end
end
end
%%%%%%%%%%%%%%%%%%%%Chasing
Behaviour%%%%%%%%%%%%%%%%%%%%
while Try<try_num
    Try=Try+1;
aceptance=N*Beta+floor(N*Beta/Try);
nf=0;
```

```
Xmin=[];
for h=1:N
   if  (((abs((sum(popnewcj(h,:).^2-popnewci(h,:).^2))))^0.5))<=visual
      nf=nf+1;
      Xmin=[Xmin;popnewcj(h,:)];
   end
end
if nf~=0
   if  ((((abs((sum(popnewcj(nf,:).^2-popnewci(nf,:).^2)))))^0.5))<((((abs((sum(popnewcj(nf-
1,:).^2-popnewci(nf-1,:).^2)))))^0.5))
      Xmin=Xmin((nf),:);
      nf=nf-1;
   else
       Xmin=Xmin(1,:);
   end
end
clear i j
for i=1:aceptance
   for j=1:D
      if ((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2)))))^0.5))~=0
  if popnewsj(i,j)<lb(j)
   popnewci(i,j)=popnewci(i,j)+abs(rand*(uk(j)-lk(j))*(Xmin(j)-
popnewci(i,j)))/((((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2))))))^0.5));
   end
   if popnewsj(i,j)>ub(j)
   popnewci(i,j)=popnewci(i,j)-abs(rand*(uk(j)-lk(j))*(Xmin(j)-
popnewci(i,j)))/((((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2))))))^0.5));
   else
   popnewci(i,j)=popnewci(i,j)+rand*(uk(j)-lk(j))*(Xmin(j)-
popnewci(i,j))/((((((abs((sum(Xmin.^2-popnewci(i,(1:D)).^2))))))^0.5));
   end
      end
   end
end
end
end
end
end

clear i j
popnew_Best=[];
for Fish=1:N
   for property=1:3
      if property==1
```

```
            Best=popnewpi(Fish,:);
        end
            if property==2
                if objfunc(fitness,Best)>objfunc(fitness,popnewsi(Fish,:))
                    Best=popnewsi(Fish,:);
                end
            end
            if property==3
                if objfunc(fitness,Best)>objfunc(fitness,popnewci(Fish,:))
                    Best=popnewci(Fish,:);
                end
            end
    end
    popnew_Best=[popnew_Best;Best];
end

%%%%%%%%&&&&&&&&&&&&&&&&%%%Crossover
Operator%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
fitpopnew_Best=objfunc(fitness,popnew_Best);
if abs((fitpopnewi-fitpopnew_Best)/fitpopnew_Best)<0.1
    for kk=1:N
        R=1:N;
        R(kk)=[];
        r=0;
        while r==0
        r=randi((N-1),1,1);
        end
        r1=R(r);
       R(r)=[];
        r=0;
        while r==0
        r=randi((N-2),1,1);
        end
        r2=R(r);
        alpha=1.25;
        popcross(kk,1:D)=popnew_Best(r1,:)+alpha*(popnew_Best(r2,:)-popnew_Best(r1,:));
        if objfunc(fitness, popcross(kk,1:D))<fitpopnew_Best
            popnew_Best(kk,:)=popcross(kk,1:D);
        end
    end
end
visual=ceil(visual*(Itr_max-iteration)/Itr_max);
step=ceil(step*(Itr_max-iteration)/Itr_max);
try_num=ceil(try_num*(Itr_max-iteration)/Itr_max);
```

```
popnewpkk=popnew_Best;
end
```

APPENDIX C1

m-FILE for objective function.

```
function Y=objfunc(n,X)
% Y=objfunc(n,X) is used to determine the fitness of artificial fish
% individuals. n is the serial number of the objective function which is
% selected to be used to estimate the fitness. X is a roundom population of
% artificial fish with dimension D= numbers of columns in the X matrix. the
% number of rows in the X matrix represents the number (population of fish.
%  |FUNCTION|     |n|
%   Ackley          1
%   CM              2
%   DeJongf4        3
%   Expfun          4
%   Griewank        5
%   Hyperelliptic   6
%   LM1             7
%   LM2             8
%   Neumaier3       9
%   Rastrigin       10
%   Rosenbrock      11
%   Sal             12
%   Schwefel        13
%   chaffer         14
%   Sphere          15
%   Bukin           16
[k,l]=size(X);
%Ackley Function
if n==1
   if l<30
      m=30-l;
      X=[X,zeros(k,m)];
   else
   if l>=31
```

```
        X(:,(31:l))=[];
    end
    end
    for i=1:30
        for h=1:k
        x(h)=x(h)+cos(2*pi*X(h,i));
        z(h)=z(h)+(X(h,i)^2);
        end
    end
y=zeros(k,1);
    for ll=1:k
        y(ll)=-20*exp(-0.2*sqrt(z(ll)/30))-exp(x(ll)/30)+20+exp(1);
    end
    Y=y;
End
% Cosine Mixture Function CM
if n==2
     for i=1:l
        for h=1:k
        x(h)=x(h)+cos(5*pi*X(h,i));
        z(h)=z(h)+(X(h,i)^2);
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=-0.1*x(ll)+z(ll);
    end
    Y=y;
End
% DejongF4 Function
if n==3
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];
    else
    if l>=31
        X(:,(31:l))=[];
    end
    end
    x=zeros(k,1);
    for i=1:30
        for h=1:k
        x(h)=x(h)+i*X(h,i)^4;
        end
```

```
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll);
    end
    Y=y;
End
%Exponential Function
if n==5
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];

    else
    if l>=31
        X(:,(31:l))=[];
    end
    end
      for i=1:l
       for h=1:k
       x(h)=x(h)+X(h,i)^2;
       end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=exp(-0.5*sum(x(ll)));
    end
    Y=y;
End
%Griewangk Function
if n==4
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];

    else
    if l>=31
        X(:,(31:l))=[];
    end
    end
    x=zeros(k,1);
    for i=1:30
        for h=1:k
        x(h)=x(h)+X(h,i)^2;
```

```
        end
    end
    for i=1:30
        for h=1:k
        z(h)=z(h)*cos((X(h,i))/sqrt(i));
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll)/4000-z(ll)+1;
    end
    Y=y;
End
%Hyperelliptic Function
if n==6
    if l<20
        m=20-l;
        X=[X,zeros(k,m)];
    else
    if l>=21
        X(:,(21:l))=[];
    end
    end
    x=zeros(k,1);
    for i=1:20
        for h=1:k
        x(h)=x(h)+(i^2)*X(h,i)^2;
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll);
    end
    Y=y;
end
%Levy and Montalvo LM1 Function
if n==7
    x=zeros(k,1);
    for i=1:l-1
        for h=1:k
        x(h)=x(h)+((X(h,i)-1)^2)*(1+(sin(pi*X(h,i+1)))^2)+(X(h,l)-1)^2;
        end
    end
    y=zeros(k,1);
```

```
    for ll=1:k
        y(ll)=(pi/l)*(10*((sin(pi*X(h,1)))^2)+x(ll));
    end
    Y=y;
end
%Levy and Montalvo LM2
if n==8
    x=zeros(k,1);
    for i=1:l-1
        for h=1:k
        x(h)=x(h)+((X(h,i)-1)^2)*(1+(sin(3*pi*X(h,i+1)))^2)+((X(h,l)-
1)^2)*(1+(sin(2*pi*X(h,i)))^2);
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=0.1*(10*((sin(3*pi*X(h,1)))^2)+x(ll));
    end
    Y=y;
End
%Neumiear3 Function
if n==9
    for i=1:l
        for h=1:k
        x(h)=x(h)+(X(h,i)-1)^2;
        if i>1
        z(h)=z(h)+(X(h,i)*X(h,i-1));
        end
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll)-z(ll);
    end
    Y=y;
end
%Rastrigrin Function
if n==10
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];
    else
    if l>=31
        X(:,(31:l))=[];
```

```
    end
    end
    x=zeros(k,1);
    for i=1:30
        for h=1:k
        x(h)=x(h)+((X(h,i)^2)-10*cos(2*pi*X(h,i))+10);
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll);
    end
    Y=y;
end
%Rosenbrock Function
if n==11
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];
    else
    if l>=31
        X(:,(31:l))=[];
    end
    end
    x=zeros(k,1);
    for i=1:29
        for h=1:k
        x(h)=x(h)+100*(X(h,i+1)-X(h,i)^2)^2+(X(h,i)-1)^2;
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll);
    end
    Y=y;
end
%Sal Function
if n==12
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];
    else
    if l>=31
        X(:,(31:l))=[];
```

```
    end
    end
    x=zeros(k,1);
    for i=1:30
        for h=1:k
        x(h)=x(h)+X(h,i)^2;
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=1-cos(2*pi*sqrt(x(ll)))+0.1*sqrt(x(ll));
    end
    Y=y;
end
%Skwefel Function
if n==13
    if l<10
        m=10-l;
        X=[X,zeros(k,m)];
    else
    if l>=11
        X(:,(11:l))=[];
    end
    end
    for i=1:10
        for h=1:k
        x(h)=x(h)+ X(h,i)*sin(sqrt(abs(X(h,i))));
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=418.9828872724339*10-x(ll);
    end
    Y=y;
end
%Schaffer Function
if n==14
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];
    else
    if l>=31
        X(:,(31:l))=[];
    end
```

```
    end
    for i=1:29
        for h=1:k
x(h)=x(h)+(((X(h,i)^2)+(X(h,i+1)^2)^0.25)*(abs((sin(50*(((X(h,i)^2)+(X(h,i+1)^2)^0.1))))^
2)+1));
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll);
    end
    Y=y;
end
%Sphere Function
if n==15
    if l<30
        m=30-l;
        X=[X,zeros(k,m)];
    else
    if l>=31
        X(:,(31:l))=[];
    end
    end
    for i=1:30
        for h=1:k
        x(h)=x(h)+X(h,i)^2;
        end
    end
    y=zeros(k,1);
    for ll=1:k
        y(ll)=x(ll);
    end
    Y=y;
end
%Bukin Function
if n==16
    if l<30
          m=30-l;
          X=[X,zeros(k,m)];
    else
        if l>=31
              X(:,(31:l))=[];
        end
    end
```

```
    for i=1:30
       for h=1:k
          x(h)=x(h)+abs(X(h,i))^0.8+5*sin(X(h,i))^3;
       end
    end
    y=zeros(k,1);
    for ll=1:k
       y(ll)=x(ll);
    end
    Y=y;
end
```

APPENDIX C2

m-FILE for the 3d plots

```
Function xA=plots
clear all
clc
%Ackley Function
str1='-20*exp(-0.2*sqrt(0.5*(x^2+y^2)))';
str2='-exp(0.5*(cos(2*pi*x)+cos(2*pi*y)))+20+exp(1)';
funstr=strcat(str1,str2);
% Converting to an inline function
f=vectorize(inline(funstr));
% range=[xmin xmax ymin ymax];
range=[-5 5 -5 5];
Ngrid=100;
dx=(range(2)-range(1))/Ngrid;
dy=(range(4)-range(3))/Ngrid;
[x,y]=meshgrid(range(1):dx:range(2),...
            range(3):dy:range(4));
z=f(x,y);
% Display the shape of the objective function
figure(1);
surfc(x,y,z);
pause
% 3D Visualization of Cosine Misxture function
x1 = linspace(-5.12,5.12,100);
x2 = linspace(-5.12,5.12,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluateing the function
Z =(-0.5)*cos(5*pi*XX1)+exp(XX1.^2)+(-0.5)*cos(5*pi*XX2)+exp(XX2.^2);
surfc(x1,x2,Z')
title('Cosine Mixture Optimization Test Function.')
pause
% 3D Visualization of Dejong 4th-order function
x1 = linspace(-5.12,5.12,100);
x2 = linspace(-5.12,5.12,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluateing the function
```

```
Z = exp(-0.5*XX1.^2)+exp(-0.5*XX2.^2);
%The 3D plot
surfc(x1,x2,Z')
title('Exponential Optimization Test Function.')
pause
% 3D Visualization of Dejong 4th-order function
x1 = linspace(-5.12,5.12,100);
x2 = linspace(-5.12,5.12,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluateing the function
Z = 1*XX1.^4+2*XX2.^4;
%The 3D plot
surfc(x1,x2,Z')
title('Dejongf4 Optimization Test Function.')
pause
% 3D Visualization of Greinwangk function
x1 = linspace(-600,600,50);
x2 = linspace(-600,600,50);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluateing the function
Z=(1/4000)*XX1.^2-cos(XX1/sqrt(1))+(1/4000)*XX2.^2-
cos(XX2/sqrt(2))*cos(XX2/sqrt(1))
%The 3D plot
surfc(x1,x2,Z)
title('Greinwangk Optimization Test Function.')
pause
% 3D Visualization of Hyperelliptic function
x1 = linspace(-5.12,5.12,100);
x2 = linspace(-5.12,5.12,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluateing the function
Z=XX1.^2+2^2*XX2.^2
%The 3D plot
surfc(x1,x2,Z)
title('Hyperelliptic Optimization Test Function.')
pause
%Levy and Montalvo Optimization First Test Function
%initialize the variables
x1=linspace(-2.12,5.12,10);
x2=linspace(-5.12,5.12,10);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluate the function
Z=(pi/2)*(10*(sin(pi*XX1))^2+((XX1-1)^2)*(1+(sin(XX1*pi)))+(XX1-
1)^2)+(pi/2)*(10*(sin(pi*XX2))^2+((XX2-1)^2)*(1+(sin(XX2*pi)))+(XX2-1)^2);
```

```
%ploting the 3D
surfc(x1,x2,Z)
title('Levey and Motalvo(LM1) Optimisation Test Function')
pause
%Levy and Montalvo Optimization Second Test Function
%initialize the variables
x=linspace(-1,1,50);
y=linspace(-1,1,50);
[X,Y]=meshgrid(x,y);
%Evaluate the function
Z=(sin(3*pi*X))^2+((X-1)^2)*(1+(sin(3*pi*Y)^2))+((Y-1)^2)*(1+(sin(2*pi*Y)))
%ploting the 3D
surfc(x,y,Z')
title('Levey and Motalvo(LM2) Optimisation Test Function');
pause
%Neumaier Optimization Third Test Function
%initialize the variables
x1=linspace(-10,10,100);
x2=linspace(-10,10,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluate the function
Z=(XX2-1).^2-(XX2*XX1);
%ploting the 3D
surfc(x1,x2,Z')
title('Neumaier Optimisation Test Function');
pause
% 3D Visualization of Rastrigrin function
x1 = linspace(-5.12,5.12,100);
x2 = linspace(-5.12,5.12,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluateing the function
Z = (XX1.^2-10*cos(2*pi*XX1)) + (XX2.^2-10*cos(2*pi*XX2))+20
%The 3D plot
surfc(x1,x2,Z')
title('Rastrigin Optimization Test Function.')
pause
%3D Visualization of Rosenbrock function
x1 = linspace(-2,2,100);
x2 = linspace(-2,2,100);
[X1,X2]=meshgrid(x1,x2);
%Evaluating the function
Z=100*(X2-X1.^2).^2+(1-X1).^2;
surfc(x1,x2,Z');
title('Rosenbrock Optimization Test Function');
```

```
pause
%Sal Function
%initialize the variables
x1=linspace(-2*pi,2*pi,100);
x2=linspace(-2*pi,2*pi,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluate the function
Z=1-cos(2*pi*(sqrt((XX1.^2)+(XX2.^2))))+0.1*sqrt(XX1.^2+XX2.^2);
%ploting the 3D
surfc(x1,x2,Z')
title('Sal Optimisation Test Function');
Pause
%Schwefel Function
x1 = linspace(-500,500,100);
x2 = linspace(-500,500,100);
[X1,X2] = meshgrid(x1,x2);
%Evaluating the function
Z = (-1*X1.*sin(sqrt(abs(X1))))+(-1*X2.*sin(sqrt(abs(X2))));
%the 3D plot
surfc(x1,x2,Z);
title('Schwefel Optimization test function')
pause
%Schaffer
x1=linspace(-100,100,100);
x2=linspace(-100,100,100);
[XX1,XX2]=meshgrid(x1,x2);
Z=abs(((XX1.^2+XX2.^2).^(0.25))*(((sin(50*(XX1.^2+XX2.^2))^0.1).^2+1)));
%ploting the 3D
surfc(x1,x2,Z)
title('Schaffer Optimisation Test Function');
pause
%Sphere Function
%initialize the variables
x1=linspace(-5.12,5.12,100);
x2=linspace(-5.12,5.12,100);
[XX1,XX2]=meshgrid(x1,x2);
%Evaluate the function
Z=XX1.^2+XX2.^2;
%ploting the 3D
surfc(x1,x2,Z)
title('Sphere Optimisation Test Function');
pause
%Bukin Function
x=linspace(-15,-5,50);
```

```
y=linspace(-3,3,50);
[X,Y]=meshgrid(x,y);
z=100*(sqrt(abs(Y-0.01*X.^2)))+0.01*(abs(X+10));
surfc(x,y,z)
xlabel('x-axis')
```

Appendix D

m-FILE for the Q and R Determination

```
if n==17
[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*max(max(A)))^0.125)*max
(max(B)))^0.5;
   eigenvalue(i,i)=realeigen+1i*(-1)^i*imageigen;
end
eigenvalue=real(eigenvalue);
kk=[];
bb=[];
for j=1:b1
   if B(j)==0
      bb(j,1)=1;
   else
      bb(j,1)=B(j);
   end
end
for j=1:b1
   M=A-eigenvalue;
   Ki=max(M(:,j)./bb);
   kk=real([kk,Ki]);
end
y=0;
r=zeros(b2);
q=zeros(a1);
while y~=10
   y=y+1;
   t=2*y*randi(2);
r=r+eye(b2).*rand(b2).*randi(2,b2,b2);
W=[];
r=(r+rand*(t-y+1)*r/t)/2;
```

```
R=r;
q=(q+(t*eye(a1)+(q+(sum(sum(r))*(round(rand(a1)).*(randi(100,a1,1)*ones(1,a1))+(eye(a1).
*(randi(100,a1,1)*ones(1,a1))));
q=(tril(q)+(tril(q))')/2;
q=rand*(t-y+1)*q/t;
Q=q;
[K,P,E]=lqr(A,B,Q,R);
q=(abs(A'*P+P*eigenvalue)+(abs(A'*P+P*eigenvalue))');
q=q.*Q/max(max(q));
end
Y=Q;
Z=R;
End
```

APPENDIX E

m-FILE for the Inverted Pendulum Responses

```
function xA= QIP
clc
clear all
%The Quad Inverted Pendulum.
m0=1.3282;%Mass of the Cart
m1=0.2200;%Mass of the First Pendulum
m2=m1;%Mass of the Second Pendulum
m3=m2;%Mass of the Third Pendulum
m4=0.1870;%Mass of the Fourth Pendulum
g=9.81;%Gravitaional Constant
u0=22.9147;
u1=0.007056;
u2=0.002646;
u3=u2;
u4=u3;
% Element of State Matix Computaion
n1=u1+u2;
n2=u2+u3;
n3=u3+u4 ;
L1=0.49;
L2=L1;
L3=L2;
L4=0.452;
a0=m0+m1+m2+m3+m4;
a1=(m1/3+m2+m3+m4)*L1^2;
a2=(m2/3+m3+m4)*L2^2;
a3=(m3/3+m4)*L3^2;
a4=(m4/3)*L4^2;
b0=(m1/2+m2+m3+m4)*L1;
b1=(m2/2+m3+m4)*L2;
b2=(m3/2+m4)*L3;
b3=(m4/2)*L4;
a11=L1*b1;
a12=L1*b2;
a13=L1*b3;
a22=L2*b2;
a23=L2*b3;
```

```
a33=L3*b3;
M0=[a0 b0 b1 b2 b3;b0 a1 b1*L1 b2*L1 b3*L1;b1 b1*L1 a2 b2*L2 b3*L2;
   b2 b2*L1 b2*L2 a3 b3*L3;b3 b3*L1 b3*L2 b3*L3 a4];
N0=[u0 0 0 0 0;0 n1 -u2 0 0;0 -u2 n2 -u3 0;0 0 -u3 n3 -u4;0 0 0 -u3 u4];
D0=[0 0 0 0 0;0 -b0 0 0 0;0 0 -b1 0 0;0 0 0 -b2 0;0 0 0 0 -b3]*g;
G=[1;0;0;0;0];
InvM0=inv(M0);
A1=-InvM0*D0;
A2=-InvM0*N0;
B1=InvM0*G;
A=[zeros(5,5) eye(5,5); A1 A2];
B=[zeros(5,1); B1];
C=eye(size(A));
D=0;
%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=input([provide the value of mCAFAC Q]);
R=input([provide the value of mCAFAC R])
% Settings for mCAFAC.
[K P E]=lqr(A,B,Q,R);%Q and R used here are obtained from mCAFA_NsSd
sys_LQ=ss(A-B*K,B,C,D);
pzmap(sys_LQ)
sgrid
pause
x=[2;-0.04;0.02;-0.01;0.005;0;0;0;0;0];
t=0:0.02:10;
x0=initial(sys_LQ,x,t);
X1=[1 0 0 0 0 0 0 0 0 0]*x0';
X2=[0 1 0 0 0 0 0 0 0 0]*x0';
X3=[0 0 1 0 0 0 0 0 0 0]*x0';
X4=[0 0 0 1 0 0 0 0 0 0]*x0';
```

```
X5=[0 0 0 0 1 0 0 0 0 0]*x0';
% Setting for trial and error
Q0=[200 250 200 200 200 0 0 0 0 0];
Q1=diag(Q0);
R1=1;
[K1 P1 E1]=lqr(A,B,Q1,R1);
sys_ss=ss(A-B*K1,B,C,D);
x00=initial(sys_ss,x,t);
X11=[1 0 0 0 0 0 0 0 0 0]*x00';
X22=[0 1 0 0 0 0 0 0 0 0]*x00';
X33=[0 0 1 0 0 0 0 0 0 0]*x00';
X44=[0 0 0 1 0 0 0 0 0 0]*x00';
X55=[0 0 0 0 1 0 0 0 0 0]*x00';
subplot(2,2,1)
plot(t,X2,'b')
hold on
plot(t,X22,'r')
xlabel('Time(s)')
ylabel('First Pendulum Angle (rad)')
title('Response of the First PENDULUM')
legend('Angular Velocity of the First Pendulum (mCAFA)','Angular Velocity of the First
Pendulum(Trial & Error)')
grid on
mCAFAC_1=stepinfo(X2,t)
TandE_1=stepinfo(X22,t)
subplot(2,2,2)
%Plot for the Second Pendulum
plot(t,X3,'b')
hold on
plot(t,X33,'r')
title('Response of the Second PENDULUM')
xlabel('Time(s)')
ylabel('Second Pendulum Angle (rad)')
legend('Angular Velocity of the Second Pendulum (mCAFAC)','Angular Velocity of the
Second Pendulum(Trial & Error)')
mCAFAC_2=stepinfo(X3,t)
TandE_2=stepinfo(X33,t)
grid on
subplot(2,2,3)
%Plot for the Third Pendulum
plot(t,X4,'b')
hold on
plot(t,X44,'r')
title('Response of the Third PENDULUM')
```

```
xlabel('Time(s)')
ylabel('Third Pendulum Angle (rad)')
legend('Angular Velocity of the Third Pendulum (mCAFAC)','Angular Velocity of the Third Pendulum (Trial & Error)')
grid on
mCAFAC_3=stepinfo(X4,t)
TandE_3=stepinfo(X44,t)
subplot(2,2,4)
%Plot for the Fourth Pendulum
plot(t,X5,'b')
hold on
plot(t,X55,'r')
xlabel('Time(s)')
ylabel('Fourth Pendulum Angle (rad)')
legend('Fourth Pendulum Angular Velocity (mCAFAC)','Fourth Pendulum Angular Velocity (Trial and Error)')
title('Response of the Fourth PENDULUM')
grid on
mCAFAC_4=stepinfo(X5,t)
TandE_4=stepinfo(X55, t)
```