# OPTIMAL POWER FLOW SOLUTION WITH STOCHASTIC RENEWABLE ENERGIES USING NATURE INSPIRED ALGORITHM

ABDUL MU'IZ ZULFADLI BIN AB WAHAB

B.ENG (HONS.) ELECTRICAL ENGINEERING (POWER SYSTEM)

UNIVERSITI MALAYSIA PAHANG

**UNIVERSITI MALAYSIA PAHANG**

**DECLARATION OF THESIS AND COPYRIGHT**

Author's Full Name : ABDUL MU'IZ ZULFADLI BIN AB WAHAB

Date of Birth : 23 SEPTEMBER 1997

Title : OPTIMAL POWER FLOW SOLUTION WITH STOCHASTIC RENEWABLE ENERGIES USING NATURE INSPIRED ALGORITHM

Academic Session : SEMESTER 1 ACADEMIC SESSION 2021/2022

I declare that this thesis is classified as:

| | | |
|---|---|---|
| ☐ | CONFIDENTIAL | (Contains confidential information under the Official Secret Act 1997)* |
| ☐ | RESTRICTED | (Contains restricted information as specified by the organization where research was done)* |
| ☑ | OPEN ACCESS | I agree that my thesis to be published as online open access (Full Text) |

I acknowledge that Universiti Malaysia Pahang reserves the following rights:

1. The Thesis is the Property of Universiti Malaysia Pahang
2. The Library of Universiti Malaysia Pahang has the right to make copies of the thesis for the purpose of research only.
3. The Library has the right to make copies of the thesis for academic exchange.

Certified by:

| | |
|---|---|
| _____ | _____ |
| (Student's Signature) | (Supervisor's Signature) |
| 970923-11-5373 | Assoc. Prof. Dr. Mohd Herwan Bin Sulaiman |
| Date: 13 February 2022 | Date: 13February 2022 |

NOTE : * If the thesis is CONFIDENTIAL or RESTRICTED, please attach a thesis declaration letter.

# THESIS DECLARATION LETTER (OPTIONAL)

Librarian,
*Perpustakaan Universiti Malaysia Pahang*,
Universiti Malaysia Pahang,
Lebuhraya Tun Razak,
26300, Gambang, Kuantan.

Dear Sir,

CLASSIFICATION OF THESIS AS RESTRICTED

Please be informed that the following thesis is classified as RESTRICTED for a period of three (3) years from the date of this letter. The reasons for this classification are as listed below.

Author's Name
Thesis Title

Reasons (i)

(ii)

(iii)

Thank you.

Yours faithfully,

_____
(Supervisor's Signature)

Date:

Stamp:

Note: This letter should be written by the supervisor, addressed to the Librarian, *Perpustakaan Universiti Malaysia Pahang* with its copy attached to the thesis.

## SUPERVISOR'S DECLARATION

I hereby declare that I have checked this thesis and in my opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Electrical Engineering (Power System).

_____

(Supervisor's Signature)

Full Name : Assoc. Prof. Dr. Mohd Herwan Bin Sulaiman

Position :

Date :

_____

(Co-supervisor's Signature)

Full Name :

Position :

Date :

## STUDENT'S DECLARATION

I hereby declare that the work in this thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at Universiti Malaysia Pahang or any other institutions.

_____

(Student's Signature)

Full Name : Abdul Mu'iz Zulfadli bin Ab Wahab

ID Number : Ec18091

Date : 13 February 2022

# OPTIMAL POWER FLOW SOLUTION WITH STOCHASTIC RENEWABLE ENERGIES USING NATURE INSPIRED ALGORITHM

ABDUL MU'IZ ZULFADLI BIN AB WAHAB

Thesis submitted in fulfillment of the requirements
for the award of the
B.Eng (Hons.) Electrical Engineering (Power System)

Faculty of Electrical & Electronics Engineering

UNIVERSITI MALAYSIA PAHANG

FEBRUARY 2022

## ACKNOWLEDGEMENTS

All praise to the almighty God with His blesses and chances for me as I can finish this project and thesis writing in order for me to finish my Bachelor in Electrical Engineering.

In this opportunity, I would like to give a big gratitude and thank to my beloved parent. A thousand of thanks for both of you that support me without hesitate until I can finish my project. Their moral support gives me hope and push me to exceed my boundary in making this thesis happen.

Also, I would like to thank and give my gratefulness for my supervisor, Associate Professor Dr. Mohd Herwan Bin Sulaiman who taught me and lead me until I can successfully finish the project given in the meantime. His guidelines based on his experience in researching skills and knowledge help me a lot in doing this project and writing this thesis.

My next gratitude will be for my Academic Advisor, Ts. Dr. Norazlianie Binti Sazali as she helps me a lot during my ups and downs while finishing this project. Her supportive character always calms me and help me in making good decisions throughout this project development. Not to forget all staff in College in Engineering, specifically the Department of Electrical Engineering that gives their knowledge and experience to makes me understands more about my project.

To my friends who is together with me throughout this final year project, our sleepless night, our joy and cry. From the bottom of my heart, I would like to thank all of you for the support and encourage to me, lets pray for our best in future.

# ABSTRAK

Thesis ini menerangkan penggunaan algoritma Moth Flame Optimization (MFO) untuk menyelesaikan aliran kuasa optimum sebagai masalah pengoptimuman objektif dalam pengendalian dan kawalan sistem kuasa. Memandangkan kesan pencemaran alam sekitar daripada loji kuasa bahan api fosil, aliran kuasa optimum yang meminimumkan hanya kos keseluruhan bahan api nampaknya tidak lagi relevan untuk dilihat sebagai kekangan objektif tunggal. Kaedah pengoptimuman, yang berdasarkan model statistik untuk menyelesaikan aliran kuasa optimum dan masalah, hendaklah ditakrifkan sebagai menyelesaikan masalah dengan satu fungsi objektif yang sama. Menggunakan persamaan yang berkaitan, yang tidak melanggar sistem api rama-rama yang telah dicipta sebagai asasnya, simulasi akan dijalankan untuk beberapa lelaran, dan selepas melengkapkan lelaran, simulasi akan mencetak keluaran yang paling optimum. hasil, dan simulasi ini mesti dijalankan beberapa kali untuk mencari output yang mantap untuk pengumpulan data. Kaedah ini telah digunakan pada tiga sistem penjanaan yang berbeza dengan keadaan beban yang berbeza-beza. Keputusan yang diperoleh menggunakan pendekatan yang dicadangkan adalah berkaitan dengan pendekatan lain yang dibincangkan dalam tinjauan literatur. Menjelang akhir penyelidikan ini, algoritma ini harus ditunjukkan sebagai salah satu sistem yang mudah digunakan dan mampu mencari penyelesaian optimum hampir global dengan penumpuan dan prestasi yang ketara jika dibandingkan dengan algoritma lain.

## ABSTRACT

The use of the Moth Flame Optimization (MFO) algorithm to solve optimal power flow as an objective optimization problem in power system operation and control is described in this thesis. Given the environmental consequences of pollution from fossil-fueled power plants, the optimal power flow that minimises only the overall cost of fuel appears to be no longer relevant as a single objective constraint. The optimization method, which is based on statistical models to solve optimal power flow and problems, shall be defined as a method for solving problems with a single identical objective function. Using the relevant equation, which is not violating the moth flame's system that has been developed as their base, the testing will run for a number of iterations, and after achieving the iterations, the testing will print out the output which is at their best optimal outcome, and this testing must run for a number of times to find the steady output for data collection. This method was tested on three different generation systems under varying load conditions. The results obtained using the proposed approach are comparable to those obtained using the other approaches discussed in the literature review. By the end of this study, this algorithm should have been demonstrated to be a process that is simple to use and capable of searching for a near-global optimal solution with significant convergence and effectiveness when compared to other algorithms.

## TABLE OF CONTENT

## LIST OF TABLES

## LIST OF FIGURES

## LIST OF EQUATION

## LIST OF ABBREVIATIONS

| | |
|---|---|
| MFO | Moth-Flame Optimization |
| BMO | Barnacles Mating Optimizer |
| PSO | Particle Swarm Optimization |
| CSA | Cuckoo Search Algorithm |

# CHAPTER 1

# INTRODUCTION

## 1.1 Project Background

The moth-flame optimization technique is a new algorithm that has recently received a lot of attention. This new intelligent application was inspired by the biological behaviour of moths fighting fires in nature. The moth's biological inspiration is its nocturnal flight method. The moth updates its position by swirling around the flame. The moth-flame optimization method (MFO) has the advantages of a minimal number of setup elements, ease of understanding and implementation, and short convergence time. Nonetheless, the literature shows that the moth-flame optimization technique still has room for improvement. As a result, a number of scholars have worked to improve the algorithm in various ways during the previous two years. The method is also widely used in physics, medicine, economics, and other fields. Finally, several academics have provided examples of how the method could be applied to real-world problems in other domains.

The Moth-Flame Optimizer (MFO), a new nature-based optimization method, is applied to the well-known economic dispatch (ED) problem in power system operation. ED is a classic optimization problem that has piqued the interest of power engineers and academics worldwide in order to achieve the lowest cost of power generation while meeting all constraints and demands. Practical constraints will be considered. MFO, on the other hand, is a completely new algorithm based on the light-and-fly mechanism. MFO will be used to determine the best mix of power generation to achieve the lowest cost while staying within any constraints. This project's goal is to identify the least amount of power loss while minimising costs.

## 1.2 Problem Statement

One of the foremost tough optimisation issues for power engineers and researchers to unravel is economic dispatch (ED) problems with sensible constraints. the matter of disfunction is to seek out the optimum power generation that meets demand whereas remaining among constraints to attain the bottom doable operative price. tiny changes in power planning may end up in important cost savings. Nonetheless, managing the multiple native minimum points within the cost function in an exceedingly practical ED problem may be a difficult task. As a result, several techniques and algorithms are enforced in ED to scale back operating costs.

Moth-flame Optimization (MFO), Particle Swarm Optimization (PSO), Cuckoo Search Algorithm (CSA) and Barnacles Mating Optimizer (BMO) have been proposed in the literature for solving ED problems.

In the last decade, there has been a flurry of research activity aimed at developing new algorithms inspired by nature to solve optimization problems. Nature-inspired algorithms are becoming increasingly popular for a variety of reasons, including their ease of use, flexibility, lack of complex mathematical derivation, and ability to avoid the local optima problem. They usually adhere to a simple set of rules that mimic the behaviour or properties of phenomena or animals. Nature-inspired algorithms will treat optimization problems as a black box, and the solution will be found by examining the input and output. As a result, they are extremely adaptable to a wide range of problems.

Many algorithms, including the popular genetic algorithm (GA), artificial bee colony (ABC) algorithm, ant colony optimization (ACO), and others, have been successfully applied to real-world optimization problems. The no free lunch (NFL) theorem states that no algorithm can perform well in all optimization problems. One can perform well in one set of problems and then suddenly perform poorly in another set of problems.

## 1.3 Objective Of Project

The goal of this study is to determine the optimum value output that fulfilled the OPF function. This goal can be achieved by developing a mathematical equation that solves the problem of economic load dispatch, combining the two problems into a single parameter or constraint, and using the Moth Flame Optimization (MFO) algorithm to solve the problem. At the conclusion of the project, this project should be able to:

i. To minimize the cost of the power generation that consist of stochastic solar power generation, thermal generators and wind power generators.

ii. To minimize power losses of the power generation that consist of stochastic solar power generation, thermal generators and wind power generators.

## 1.4 Scope of project

To achieve the objective of this research, there are two scopes that have been identified:

i. To find the optimal control variables such as power generation, transformer setting, generator's voltage, reactive compensation element.

ii. To solve Optimal Power Flow (OPF) in the power system operation and planning.

## 1.5 Thesis Organization

This thesis is divided into five chapters and appendices, which contain the following:

The backdrop of the project in general, the description of the problem, the project goal, the project scopes and limitations, and the thesis organisation are all covered in Chapter 1.

The literature study presented in Chapter 2 discusses the moth flame optimization (MFO) and the nature behind it. There is also a review of the literature on a few algorithms that are used in the optimization area of research.

The study technique utilised to carry out this project, which is moth flame optimization, is described in Chapter 3.

The findings of moth flame optimization applied to one kind of generation, which is 30-units, will be presented in Chapter 4. The end outcome is optimum power flow.

The research effort is summarised in Chapter 5. It will provide a clear conclusion for this research in order to comprehend the optimization for the area study based on particular algorithms used for current solutions that should be able to solve the issue.

# CHAPTER 2

# LITERATURE REVIEW

## 2.1 Introduction

There are numerous papers and journals that have been studied in relation to this project. The majority of these papers are based on experimental research. This chapter will explain the background research and literature review on various articles, research papers, and other similar experiments.

## 2.2 Moth-Flame Optimization (MFO)

The biological principle and basic model of the moth-flame optimization algorithm will be introduced in this chapter. The description of the algorithm's basic situation will be more useful for the operation and implementation of the algorithm's points for improvement in the following chapter.

### 2.2.1 The Biological Basis of the Moth-Flame Optimization Algorithm

While flying at night, moths use particularly unique navigational mechanisms to sustain lateral orientation. The moth flies in this mechanism by having a constant angle of its light corresponding to the moon. Because the moon is so far away, the moth relies on this near-parallel light near the surface to keep its path straight. Although lateral orientation is effective, moths have been observed circling the source repeatedly until they are exhausted. In fact, the abundance of artificial or natural point light sources confuses moths. Typically due to the moo productivity of horizontal situating; as it were when the light source is exceptionally distant absent is it supportive to the moths for keeping up straight flying development, and there's a part of fake or characteristic light other than that coming about from moths being exceptionally near to the moon; when the moths proceed to utilize the light transmitted from a settled point as a light source,

disappointment can lead to route and create, as appeared in Figure 2.1, a dangerous winding flight way.

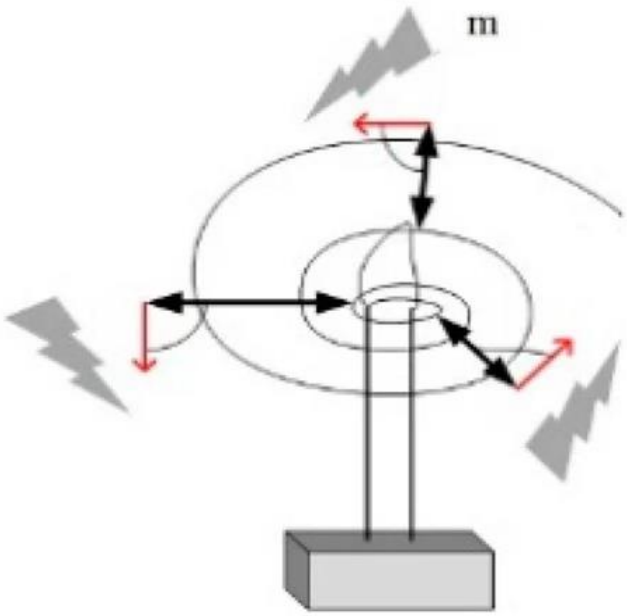


**Figure 2.1** Spiral flight path of moths near point light source

### 2.2.2 The Moth-Flame Optimization Algorithm's Basic Model

The MFO set of rules assumes the moth to be the candidate way to the problem, and the variable to be solved is the moth's role in space. Moths can fly in one, two, three, or even better dimensions with the aid of using converting their role vectors. Because the MFO set of rules is largely a swarm intelligence optimization set of rules, the moth populace withinside the matrix may be represented as follows:

$$M = \begin{bmatrix} m_{1,1} & \cdots & m_{1,d} \\ \vdots & \ddots & \vdots \\ m_{n,1} & \cdots & m_{n,d} \end{bmatrix}, \qquad 2.1$$

where n is that the variety of moths to be resolved and d is the number of management variables to be solved (dimension of the optimisation problem). it's conjointly assumed that there's a corresponding list of fitness price vectors for these moths, that is diagrammatical as follows:

$$OM = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix}, \qquad 2.2$$

To avoid the algorithmic rule falling into the native optimum value, every lepidopteran within the MFO algorithm is needed to update its own position solely with the distinctive flame admire it, that greatly improves the algorithm' world search ability. As a result, the flame and moth positions in the search house are variable matrices of constant dimension.

$$F = \begin{bmatrix} f_{1,1} & \cdots & f_{1,d} \\ \vdots & \ddots & \vdots \\ f_{n,1} & \cdots & f_{n,d} \end{bmatrix}, \tag{2.3}$$

It is additionally assumed that there's a corresponding column of fitness worth vectors for these flames, that is drawn as follows:

$$OF = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix}, \tag{2.4}$$

The update strategy of the variables within the 2 matrices differs throughout the iteration process. Moths are search people who move through the search space, and also the flame is that the best position that iteratively optimized lepidopterous insects are able to do therefore far. every moth is encircled by a corresponding flame, and once a much better answer is discovered, it's updated to the situation of the flame in the next generation. The formula will realize the worldwide optimum solution victimization this mechanism.

The change mechanism for the position of every lepidopteron relative to a flame are often expressed by the subsequent equation so as to hold out mathematical modeling for the flight behavior of a moth to a flame:

$$M_i = S(M_i, F_j), \tag{2.5}$$

where Mi denotes the ith moth, Fj denotes the jth flame, and S denotes the helical function.

The following conditions are met by this function:

1) The initial point of the helical function is chosen from the moth's initial space position.

2) The spiral's end point corresponds to the current flame's position in space.

3) The spiral's fluctuation range should not be greater than its search space.

The helical function of the moth flight path is defined by the above conditions as follows:

$$S(M_i, F_j) = D_i \cdot e^{bt} \cdot \cos(2\pi t) + F_j, \quad 2.6$$

$$t = (a-1) * rand + 1, \quad 2.7$$

$$a = -1 + Iteration * \left(-\frac{1}{T_{max}}\right), \quad 2.8$$

where Di is the linear distance between the ith moth and the jth flame, b is the defined logarithmic helix shape constant, and t is a random number in [-1,1]. The magnitude of t is represented by Eq (2.7), while the magnitude of an is represented by Eq (2.8), and it decreases linearly from -1 to -2. The expression of Di is as follows:

$$D_i = \left|F_j - M_i\right|, \quad 2.9$$

Eq (2.9) simulates the spiral flight of a moth. supported this equation, future position of the moth' renewal within the epoch is decided by the flame it surrounds. within the helix function, the constant t (t∈ [−1,1]) represents the space between the moth' position and therefore the flame in the next optimisation iteration, (t=1) represents the position nearest to the flame, and (t=−1) represents the position farthest from the flame. The spiral equation demonstrates that moths will fly round the flame instead of simply between them, making certain the algorithm' world search and native development capabilities.

When this model is used, the subsequent characteristics are observed:

1) A moth will converge to any field of flame by at random choosing parameters t.

2) The closer the moth is to the flame, the smaller the value of t.

3) As the moth gets closer to the flame, its position around the flame is updated more quickly.

The higher than flame position update mechanism will make sure the lepidopteron' ability to develop domestically round the flame. the most effective answer found within the current generation is employed because the location of succeeding generation of moths around the flame to extend the probabilities of finding a far better solution. As a result, the flame position matrix F typically contains the best solution presently available. throughout the improvement process, every moth updates its position supported the F matrix. within the MFO algorithm, the trail constant r is created of internal random numbers in [r,1], and therefore the variable r decreases linearly with the quantity of iterations within the improvement iteration method in [−1,−2]. because the iteration progresses, the moth can approach the flame a lot of exactly in its corresponding sequence. The flames are reordered supported fitness prices when every iteration. The moth updates its position in the next generation based on the flame that corresponds to that in the updated sequence. the primary moth perpetually updates its position in reference to the flame with the most effective fitness value in the list, whereas the last moth updates its position in relation to the flame with the worst fitness value in the list.

If every location update of n moths is predicated on n completely different locations within the search space, the algorithm' local development capability is reduced. to deal with this issue, associate accommodative mechanism for the amount of flames is proposed, permitting the number of flames to be reduced adaptively throughout the unvarying process, equalization the algorithm' world search capability and native development capability in the search space. the subsequent is that the formula:

$$flame_{no} = round\left(N - 1 * \frac{N-1}{T}\right), \qquad 2.10$$

where l represents the present iteration variety, N the initial maximum number of flames set, and T the utmost number of iterations set Simultaneously, thanks to flame reduction, the flame insect resembling the reduced flame within the sequence updates its position in every generation supported the flame with the worst current fitness value.

## 2.3 Particle Swarm Optimization (PSO)

Several studies on the social behaviour of animal teams were developed within the early 1990s. These studies disclosed that some animals in a very specific group, particularly birds and fishes, are ready to share data among themselves, and this ability provides these animals with a major survival advantage. impressed by these works, Kennedy and Eberhart planned the PSO rule in 1995, a metaheuristic algorithm appropriate for optimising nonlinear continuous functions. The algorithm was inspired by the idea of swarm intelligence, that is usually seen in animal groups akin to flocks and shoals.

A discussion on flock behaviour is given to elucidate however the PSO impressed the formulation of Associate in Nursing improvement rule to resolve advanced mathematical downsides. A swarm of birds flying over a location should realize some extent to land, and decisive wherever the whole swarm ought to land may be a complex problem as a result of it depends on many factors, as well as increasing the supply of food and minimising the danger of predators' existence. during this context, the birds' movement may be understood as a choreography; the birds move synchronously for a amount of your time till the most effective place to land is known and also the entire flock lands at once.

In the given example, the flock moves only when all of the swarm members can share information; otherwise, each animal will more than likely land at a different point and at a different time. According to the animal social behaviour studies described earlier in this study, all birds in a swarm looking for a good place to land can know that the best position until it is discovered by one of the swarm's members. As a result, each swarm member helps to balance their individual and swarm understanding experiences, which is referred to as social knowledge

The described problem of determining the best landing spot includes an optimization problem. In order to maximise the survival conditions of its members, the flock must identify the best point, such as latitude and longitude. To accomplish this, each bird flies around searching for and assessing different points while employing

several surviving criteria at the same time. Each of those has the advantage of knowing where the best location point is until it is discovered by the entire swarm.

Kennedy and Eberhart proposed an algorithm called PSO that could mimic the social behaviour of birds, which provides them with significant survival advantages when solving the problem of finding a safe place to land. The algorithm's inertial version, also known as the classical version, was proposed in 1995. Since then, other variations of the classical formulation have been proposed, such as the linear-decreasing inertia weight, the constriction factor weight, the dynamic inertia, and maximum velocity reduction, in addition to hybrid models or even quantum inspired approach optimization techniques that can be applied to PSO.

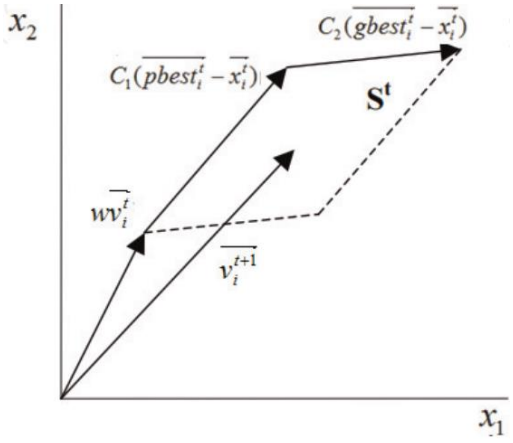


**Figure 2.2** PSO algorithm

## 2.4 Cuckoo Search Algorithm (CSA)

In the wild, cuckoos look for suitable nests at random. To simulate the cuckoo reproductive process, we propose three idealised rules:

1) A solution is an egg for the cuckoos. The cuckoo lays one egg at a time and then chooses a nest at random to place the egg in.

2) In each generation, the cuckoos find numerous nests in which to lay their eggs. Some nests contain high-quality or satisfactory solutions, which are included in the next iteration.

3) The total number of nests of the host bird is constant, and the host bird has a chance of identifying the cuckoo eggs ∈(0,1).

The cuckoo's egg represents a new solution in the cuckoo search algorithm, and that each egg there in nest defines a solution in the cuckoo search algorithm. Subpar methods have been filled with the better or new solutions in the nest. Because more than one egg is located in a nest, the algorithm becomes much more complicated because there are multiple solutions. In this study, however, we only recognise the much more normal case, where a nest includes one egg. As a result, in the CS algorithm, all of the particles, including eggs, nests, as well as cuckoos, represent a solution.

According to the above spawning behaviour hypothesis, the formula for updating the solutions in the CS algorithm is as shown in Eq (2.11),

$$x_i^{t+1} = x_i^t + \alpha \oplus Levy(\beta), \tag{2.11}$$

Where α>0 is used to control step size, its value is usually 0.01, 0.1, or 1, depending on the specific problem. Its value in this paper is 0.1. The term operator refers to entry-wise multiplications, and the Levy flight strategy formula is shown as Eq (2.12),

$$Levy(\beta) = \frac{\varphi . u}{|v|^{1/\beta}}, \tag{2.12}$$

Where v ~N(0, 1) u ~N(0, 1)

$$\varphi = \left( \frac{\Gamma(1+\beta).\sin(\pi.\beta/2)}{\Gamma\left(\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{(\beta-1)/2}\right)} \right)^{1/\beta}, \tag{2.13}$$

Where β is a constant, Its value is on interval [1, 2], and it is usually 1.5,ד (•) denotes to gamma function.

## 2.5 Barnacles Mating Optimizer (BMO)

Barnacles mating optimizer (BMO) may be a novel bioinspired optimisation rule impressed by barnacle mating. The simulation optimization method is completed

through initialization, selection, and reproduction. the subsequent is a elaborated description of the mathematical model.

The barnacle population will be expressed within the following matrix throughout the data format process:

$$\begin{bmatrix} x_1^1 & \cdots & x_1^n \\ \vdots & \ddots & \vdots \\ x_N^1 & \cdots & x_N^n \end{bmatrix}, \tag{2.14}$$

where N denotes the number of barnacles within the population and n the number of management variables the oldsters to be mated are chosen haphazardly from the population in the following choice process. Eqs. (2.15) and (2.16) propose the mathematical forms.

$$barnacle\_d = randperm(N), \tag{2.15}$$

$$barnacle\_m = randperm(N), \tag{2.16}$$

During the reproduction process, BMO primarily produces offspring using the Hardy-Weinberg principle. The barnacle's penis length (pl) is an intriguing fact that influences the exploitation and exploration of the BMO algorithm. When pl equals 7, Fig. 1 shows that barnacle #1 can only mate with one of the barnacles #2-#7. The exploitation process will then begin. Eq. (2.17) is proposed in this case to produce new offspring from parents.

$$x_i^{N\_new} = px_{barnacle\_d}^N + qx_{barnacle\_m}^N, \tag{2.17}$$

Where p is a random number drawn from the standard normal distribution between [0, 1], q = (1 – p), xNbarnacle_d and xNbarnacle_m are barnacle Dad and Mum variables chosen in Eq. (2.15), and (2.16). Furthermore, p and q represent the genotype percentages of Dad and Mum in the new generation. The genotype frequencies p and q of the parents are used to generate the new offspring. If Barnacle #1 mates with Barnacles #8-#10, the offspring undergoes the sperm cast process. The exploration process will then begin. Eq. (2.18) is proposed in this case to produce new offspring from parents.

$$x_i^{n\_new} = rand() \times x_{barnacle\_m}^n, \tag{2.18}$$

Where rand() returns a number between [0, 1]. It should be noted that Eq. (2.18) shows that the new offspring are produced solely on the basis of Mum. In general, barnacle positions are updated in each iteration by Eq. (2.17) or Eq. (2.18) to find the best position (the best solution).

## 2.6 Summary

At the end of this chapter, the details of moth flame, and its nature which lead to a new algorithm called moth flame optimization is explained well together with the system and process of the optimization under this algorithm. Throughout this chapter also, there is few details explanations for other algorithms which is particle swarm optimization, barnacles mating optimizer and Cuckoo Search Algorithm (CSA) that will be used in this project as comparison for their effectiveness compared to the result that will be obtain by the moth flame optimization.

# CHAPTER 3

# METHODOLOGY

## 3.1 Introduction

Methodology is a formal, analytical examination of the processes used in the field of study. This entails conducting a systematic examination of the set of approaches and concepts identified by the information group. Theory, analytical model, process, and quantitative or qualitative techniques are common terms.

The methodology approach is not meant to include solutions, and it is also not the same as the process. Methodology, on the other hand, provides a theoretical framework for explaining which procedure, collection of methods, or best practices should be applied to a specific situation, such as the estimation of a specific outcome.

This chapter will present relevant information about the moth flame optimization algorithm, as well as the equation that will be used to solve optimal power flow.

## 3.2 Project flowchart

The work sequence is depicted using a flowchart system. This flowchart can be used as a guide to ensure that the simulation process is carried out correctly. The procedure requires the user to enter their desired specifications, such as the number of moths, as well as the maximum iteration and function details data for lower bound, upper bound, and variable dimension.

As the user specifies, the system will run the simulation and begin the process outlined in Chapter 2 Part 2.2, without having violated the moth flame's setup that has been created as their base.

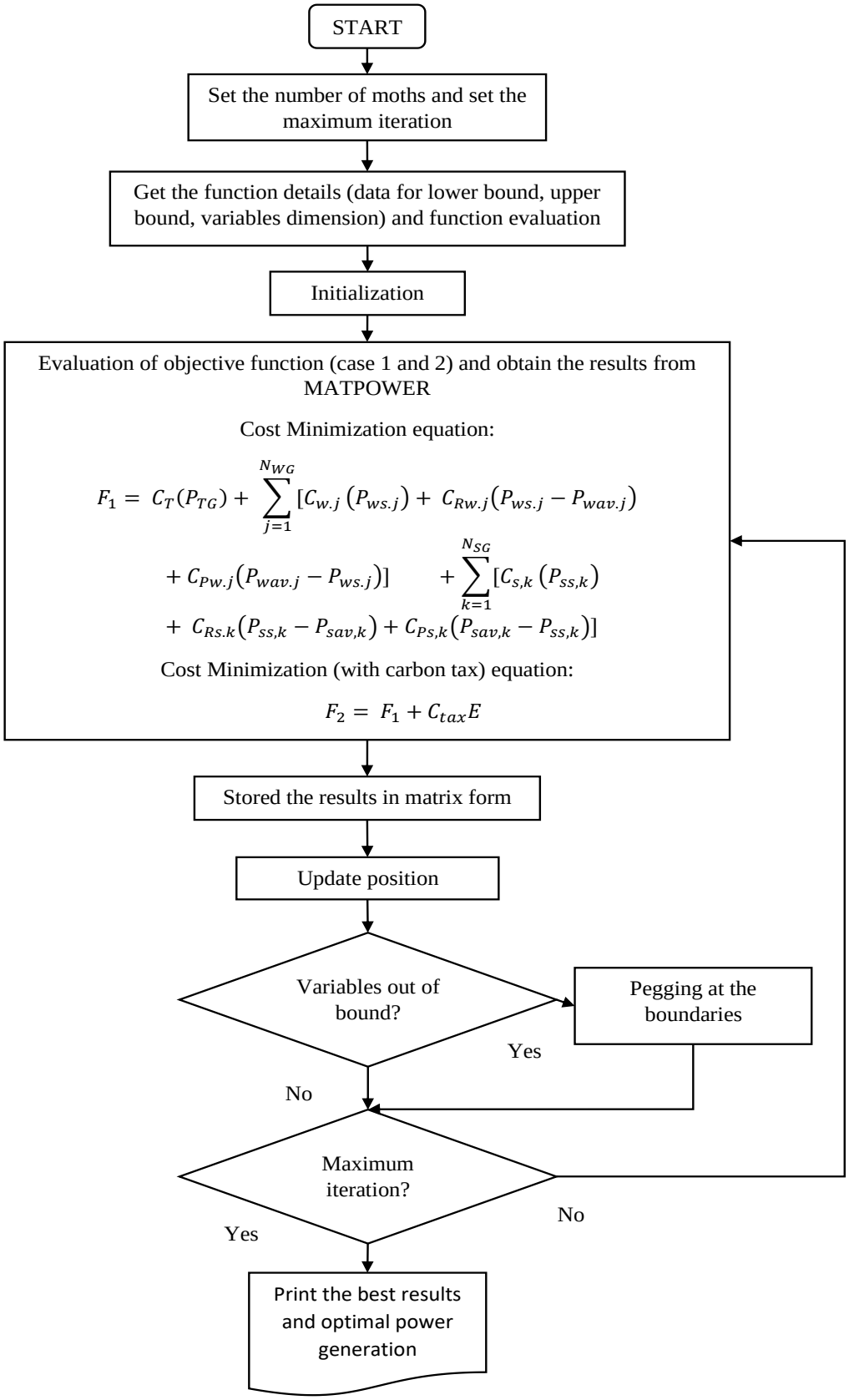


**Figure 3.1** Project flowchart

## 3.3 Pseudocode

Pseudocode is an important factor in determining project success. Pseudocode is an unofficial high-level definition of a computer system or algorithm operating theory. It employs complex conventions of standard computer language, but it is intended for a specific human reading and understanding. Pseudocode generally ignores details that are critical to a system that the computer system recognises from the proposed algorithm, such as conditional statements, system-specific code, and some subprocesses. The proposed MFO algorithm's pseudocode is provided below.

*Update flame no using Eq. 2.10*

*OM = FitnessFunction(M);*

***If** iteration == 1*

    *F = sort(M);*

    *OF = sort(OM);*

***else***

    ***F = sort($M_{t-1}$, $M_t$);***

    ***F = sort($M_t$, $M_{t-1}$);***

***End***

***For** i = 1 : n*

    ***For** j = 1 : d*

        *Update r and t*

        *Calculate D using Eq. (3.13) with respect to the corresponding moth*

        *Update M(I,j) using Eqs. Eqs. (3.11) and (3.12) with respect to the corresponding moth*

        ***end***

***end***

## 3.4 Project Design

This project was created using a statistical method in which the sample taken from the moth flame only includes a small proportion of the population in the search area. The samples have already been collected; the process will follow the steps set out In chapter 2 Part 2.2.

## 3.5 Project Instrument

This project's instrument is MATLAB software version 2019a, and it is being completed using a simulation-based method. The device used for this project has an AMD Ryzen 7 4800H processor with Radeon Graphics 2.90 GHz and 8GB RAM pre-installed.

## 3.6 Scope of Work

For this project, there is use 3 source of power that consist of thermal generator, wind power generator and solar PV unit. This generator source is used to finding the OPF which is 30-Unit Bus System. Table 1 will show the summary of IEEE-30 bus system.

**Table 3.1** Summary of IEEE-30 bus system

| **Items** | **Quantity** | **Details** |
|---|---|---|
| **Buses** | 30 | - |
| **Branches** | 41 | - |
| **Thermal generators (TG1, TG2, TG3)** | 3 | Buses: 1, 2 and 8 |
| **Wind generators (WG1, WG2)** | 2 | Buses: 5 and 11 |
| **Solar PV unit (SPV)** | 1 | Bus: 13 |
| **Control variables** | 11 | - |
| **Load bus voltage range** | 24 | [0.95-1.05] p.u |

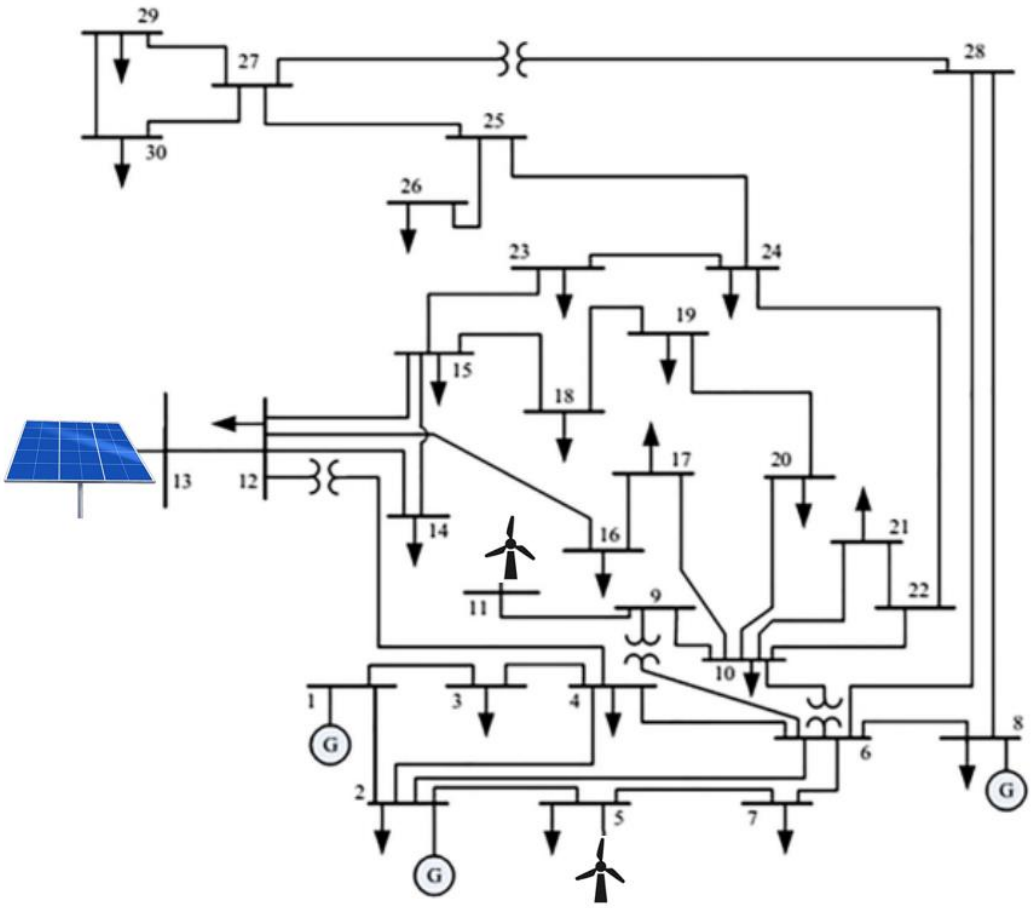


**Figure 3.2** Modified of IEEE-30 bus system

## 3.7 Optimal Power Flow Problem Formulation

The primary goal of OPF is to find the optimal control variable setting in power system components to minimise the selected objective functions while satisfying all equality and inequality constraints. To solve OPF for the system consisting of thermal and stochastic wind solar and wind power generators, two objectives are chosen: (1) cost of generation minimization and (2) loss minimization.

### 3.7.1 Cost of generation minimization

#### 3.7.1.1 Thermal units

The total cost of a thermal generating unit, including valve loading effects, is expressed as $F_{Cost}$:

$$F_{Cost}(P_{TTG}) = \sum_{i=1}^{N_{TG}} \{a_i + b_i P_{TGi} + c_i P_{TGi}^2 + |d_i \cdot \sin[e_i \cdot (P_{TGi}^{min} - P_{TGi})]\}, \quad 3.1$$

where $P_{TTG}$ is the total power output from thermal generators, $a_i$, $b_i$, $c_i$, $d_i$, and $e_i$ are the cost coefficients of the respective generator $P_{TGi}$ with the valve loading effect taken into account, $P_{TGi}^{min}$ is the i-th generator's minimum power setting, and $N_{TG}$ is the number of thermal generators in the system.

#### 3.7.1.2 Wind power generator and Solar PV unit

Certain conditions, such as the uncertain and intermittent nature of solar and wind when integrating into the power grid, had to be considered for solar PV power and wind power. Typically, solar PV farms are owned by private entities that have a power purchase agreement with the Independent System Operator (ISO). As a result, the costs of these power generators are classified into three categories: direct, reserved, and penalty costs.

The following is the direct cost of a solar PV generator:

$$C_{S,k}(P_{SG,j}) = g_{SG,k} P_{SG,k}, \quad 3.2$$

where $g_{SG,k}$ is the direct cost coefficient associated with the *k-th* solar power plant and $P_{SG,k}$ is the solar power plant's scheduled power.

The following are the direct costs of wind power:

$$C_{w.j}(P_{ws.j}) = g_j P_{ws.j}, \quad 3.3$$

where $g_j$ is direct cost coefficient related to j -th wind power plant and $P_{ws.j}$ is scheduled power.

There is a chance that the actual power delivered by the solar farm will be less than what was estimated. This phenomenon is known as overestimation of power from uncertain sources, and it occurs when the system operator must spin reserved in order to ensure continuous supply to customers. The cost of committing the reserved generating units to meet the overestimated values is known as the reserve cost, and it can be expressed as follows for solar power generation:

$$C_{RS,k}\left(P_{SG,k} - P_{Sav,k}\right) = K_{RS,k}(P_{SG,k} - P_{Sav,k}), \quad 3.4$$

$$= K_{RS,k} * f_s(P_{Sav,k} < P_{SG,k}) * [P_{SG,k} - E(P_{Sav,k} < P_{SG,k})]'$$

where $K_{RS,k}$ is the reserve cost coefficient for the k-th solar PV plant, $P_{Sav,k}$ is the actual available power from the same plant, $f_s(P_{Sav,k} < P_{SG,k})$ is the probability of a solar power shortage occurrence compared to the scheduled power ($P_{SG,k}$), and $E(P_{Sav,k} < P_{SG,k})$ is the expectation of solar PV power less than $P_{SG,k}$.

The following is the reserve cost of wind power:

$$C_{Rw,j}\left(P_{ws,j} - P_{wav,j}\right) = K_{Rw,j}(P_{ws,j} - P_{wav,j} \quad 3.5$$

$$= K_{Rw}\int_0^{P_{ws,j}} (P_{ws,j} - p_{w,j}) f_w(p_{w,j}) dp_{w,j}$$

where $K_{Rw,j}$ denotes the reserve cost coefficient for the j-th wind power plant, and $P_{wav,j}$ denotes the actual available power from the same plant. The wind power probability density function for the j-th wind power plant is denoted by $f_w(p_{w,j})$.

In the event that, contrary to the overestimation of power, the actual power delivered is greater than the estimated values, resulting in surplus power, it must be catered for by introducing a penalty cost corresponding to the surplus amount of power, which can be expressed as follows:

$$C_{PS,k}\left(P_{Sav,k} - P_{SG,k}\right) = K_{PS,k}\left(P_{Sav,k} - P_{SG,k}\right), \quad 3.6$$

$$= K_{PS,k} * f_s\left(P_{Sav,k} > P_{SG,k}\right) * \left[E\left(P_{Sav,k} > P_{SG,k} - P_{SG,k}\right)\right]$$

here, $K_{PS,k}$ denotes the penalty cost coefficient for the k-th solar PV plant, $f_s\left(P_{Sav,k} > P_{SG,k}\right)$ denotes the probability of solar PV power exceeding the scheduled power ($P_{SG,k}$), and $E\left(P_{Sav,k} > P_{SG,k} - P_{SG,k}\right)$ denotes the expectation of solar PV power exceeding $P_{SG,k}$.

The following are the penalties for using wind power:

$$C_{Pw,j}\left(P_{wav,j} - P_{ws,j}\right) = K_{Pw,j}\left(P_{wav,j} - P_{ws,j}\right), \tag{3.7}$$

$$= K_{Pw,j}\int_{P_{ws,j}}^{P_{wr,j}} (p_{w,j} - P_{ws,j}) f_w(p_{w,j}) dp_{w,j}$$

where $K_{Pw,j}$ is the penalty cost coefficient for the j-th wind power plant and $P_{wr,j}$ is the rated output power of the same windfarm.

It is well understood that generating power from traditional energy sources emits harmful gases into the environment. The emission of SOx and NOx increases with the increase in generated power (in p.u. MW) from thermal power generators, as shown in Eq (3.8). Emissions in tonnes per hour (t/h) are calculated as follows:

$$Emission, E = \sum_{i=1}^{nTG} [\alpha_i + \beta_i P_{TGi} + \gamma_i P_{TGi}^2) + \omega_i e^{(\mu_i P_{TGi})}] \tag{3.8}$$

As a result, for the first objective function with and without carbon tax, all of the costs associated with thermal, wind and solar generation discussed above are aggregated and shown as follows:

Without Carbon tax:

$$F_1 = C_T(P_{TG}) + \sum_{j=1}^{N_{WG}} [C_{w,j}\left(P_{ws,j}\right) + C_{Rw,j}\left(P_{ws,j} - P_{wav,j}\right) + C_{Pw,j}\left(P_{wav,j} - P_{ws,j}\right) \tag{3.9}$$
$$+ \sum_{k=1}^{N_{SG}} [C_{S,k}\left(P_{SG,k}\right) + C_{Rs.k}\left(P_{SG,k} - P_{Sav,k}\right) + C_{PS.k}\left(P_{Sav,k} - P_{SG,k}\right)]$$

Within Carbon tax:

$$F_2 = F_1 + C_{tax}E \tag{3.10}$$

### 3.7.2 Loss minimization

The second goal of OPF is to minimise total real power loss in the transmission system, FLoss, which can be expressed as follows:

$$F_{Loss} = \sum_{i=j}^{nl} \sum_{j \neq i}^{nl} G_{ij} \left[ V_i^2 + V_j^2 - 2V_i V_j (\delta_i - \delta_j) \right], \quad 3.11$$

where $V_i$ and $V_j$ represent the sending and receiving end voltages at bus i and j, respectively, $G_{ij}$ represents the conductance at the transmission line i-j, and nl represents the number of power transmission lines.

### 3.7.3 Voltage deviation minimization

The voltage deviation minimization at each bus of the power system network, $F_{VD}$ which can be expressed as follows:

$$F_{VD} = \sum_{m=1}^{nL} |V_{Lm} - 1.0|, \quad 3.12$$

where $V_{Lm}$ denotes the voltage at load bus m and $nL$ denotes the number of load buses.

### 3.7.4 Generation and emission cost minimization

The part that must be considered is the reduction of generation and emission costs through the imposition of a carbon tax to reduce greenhouse gas emissions, which can be defined as follows:

$$F_{CE} = F_{Cost} + c_t E_{Emission} \quad 3.13$$

where $c_t$ denotes the carbon tax, which is set at $20 per hour ($/h).

### 3.7.5 Constraints

In order to solve the OPF problem, all feasible solutions must satisfy all equality and inequality constraints. The power balance equation for real and reactive power must be satisfied in order to satisfy the equality constraint, which is expressed as follows:

$$P_{Gi} - P_{Di} - V_i \sum_{j=1}^{nB} V_j \left[ G_{ij} \cos(\delta_{ij}) + B_{ij} \sin(\delta_{ij)} \right] = 0 \; \forall i \in nB, \quad 3.14$$

$$Q_{Gi} - Q_{Di} - V_i \sum_{j=1}^{nB} V_j \left[ G_{ij} sin\,(\delta_{ij}) + B_{ij} cos\,(\delta_{ij)} \right] = 0 \; \forall i \in nB, \qquad 3.15$$

where $\delta_{ij}$ is the voltage angle difference between buses i and j, $P_{Gi}$ and $Q_{Gi}$ are the real and reactive power generation at bus (including wind and solar power), $P_{Di}$ and $Q_{Di}$ are the real and reactive load at bus $i$ and $nB$ is the total number of buses in the system. The inequality constraints, on the other hand, are the operating limits of the power system components, which can be represented as follows:

$$P_{TGi}^{min} \leq P_{TGi} \leq P_{TGi}^{max} \; i = 1, \dots, N_{TG,} \qquad 3.16$$

$$P_{SG,k}^{min} \leq P_{SG,k} \leq P_{SG,k}^{max} \; k = 1, \dots, N_{TG,} \qquad 3.17$$

$$Q_{TGi}^{min} \leq Q_{TGi} \leq Q_{TGi}^{max} \; i = 1, \dots, N_{TG,} \qquad 3.18$$

$$Q_{SG,k}^{min} \leq Q_{SG,k} \leq Q_{SG,k}^{max} \; k = 1, \dots, N_{TG,} \qquad 3.19$$

$$V_{Gi}^{min} \leq V_{Gi} \leq V_{Gi}^{max} \; i = 1, \dots, N_{G,} \qquad 3.20$$

$$V_{Li}^{min} \leq V_{Li} \leq V_{Li}^{max} \; i = 1, \dots, N_{Load,} \qquad 3.21$$

$$Q_{C,k}^{min} \leq Q_{C,k} \leq Q_{C,k}^{max} \; k = 1, \dots, N_{QC,} \qquad 3.22$$

$$T_i^{min} \leq T_i \leq T_i^{max} \; i = 1, \dots, N_{T,} \qquad 3.23$$

The real and reactive power generation limits for thermal, wind power generators, and solar PV generation are represented by Equations (3.16) and (3.17) and Equations (3.18) and (3.19), respectively. Equation (3.20) defines the voltage constraints imposed on generator buses, whereas Equation (3.21) defines the voltage constraints imposed on load buses, where $N_G$ denotes the number of generators and $N_{Load}$ denotes the number of load buses. Equations (3.22) and (3.23) define the injected MVAR limitation and transformer tap setting, respectively, where $N_{QC}$ is the total number of injected MVAR and $N_T$ is the number of transformers.

It is worth noting that all of these constraints are met by employing the power flow programme (MATPOWER) to ensure that accurate results are obtained throughout the studies.

## 3.8 Summary

This chapter presents the overall methodology involved in doing this project, where the main point is about the project flowchart, pseudocode used, project design, project instrument, scope of work and optimal power flow problem formulation. Mathematical process and equation indeed are well explained in this chapter on how this project will work and what method and strategy is best to be used. Throughout these methodology chapter, it is best described as the framework for the research where it contains elements of work, based on the objective and scope of the research. This chapter intended to elaborate on the theoretical process into simulation process under certain circumstances and conditions.

# CHAPTER 4

# RESULTS AND DISCUSSION

## 4.1 Introduction

This chapter explained about the analyse project of optimal power flow by using moth flame optimizer algorithm. Result generated by the MATLAB programming tools to get the result for the best suitable value for algorithm, cost minimization, power losses power in exposed and proposed with algorithm. This result generated by MATPOWER and get by using IEEE 30 bus system.

## 4.2 Result

Table 4.1 show that the cost and emission that setting for the generators (TG1, TG2 and TG3). And then, table 4.2 show that the PDF parameter of solar PV unit and wind power. The figure 4.1 and 4.2 shows that the solar irradiance distribution for solar PV unit at bus 13 and real power distribution for solar PV at bus 13.

**Table 4.1** Cost and emission of coefficients for modified IEEE-30 bus

| Generator | Bus | a | b | c | d | e | $\alpha$ | $\beta$ | $\gamma$ | $\omega$ | $\mu$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| TG1 | 1 | 0 | 2 | 0.00375 | 18 | 0.037 | 4.091 | -5.554 | 6.49 | 0.0002 | 6.667 |
| TG2 | 2 | 0 | 1.75 | 0.0175 | 16 | 0.038 | 2.543 | -6.047 | 5.638 | 0.0005 | 3.333 |
| TG3 | 8 | 0 | 3.25 | 0.00834 | 12 | 0.045 | 5.326 | -3.55 | 3.38 | 0.002 | 2 |

**Table 4.2** PDF parameter of solar PV unit and wind power

| **Wind power generating plants** | | | | | **Solar PV plant** | | |
|---|---|---|---|---|---|---|---|
| Windfarm# | No. of turbines | Rated power, $P_{wr}$ (MW) | Weibull PDF parameter | Weibull mean, $M_{wbl}$ | Rated power, $P_{sr}$ (MW) | Lognormal PDF parameters | Lognormal mean, $M_{lgn}$ |
| 1 (bus 5) | 25 | 75 | c = 9<br>k = 2 | v = 7.976 m/s | 50 (bus 13) | $\mu = 6\,\sigma = 0.6$ | G = 483 W/m$^2$ |
| 2 (bus 11) | 20 | 60 | c = 10<br>k = 2 | v = 8.862 m/s | | | |

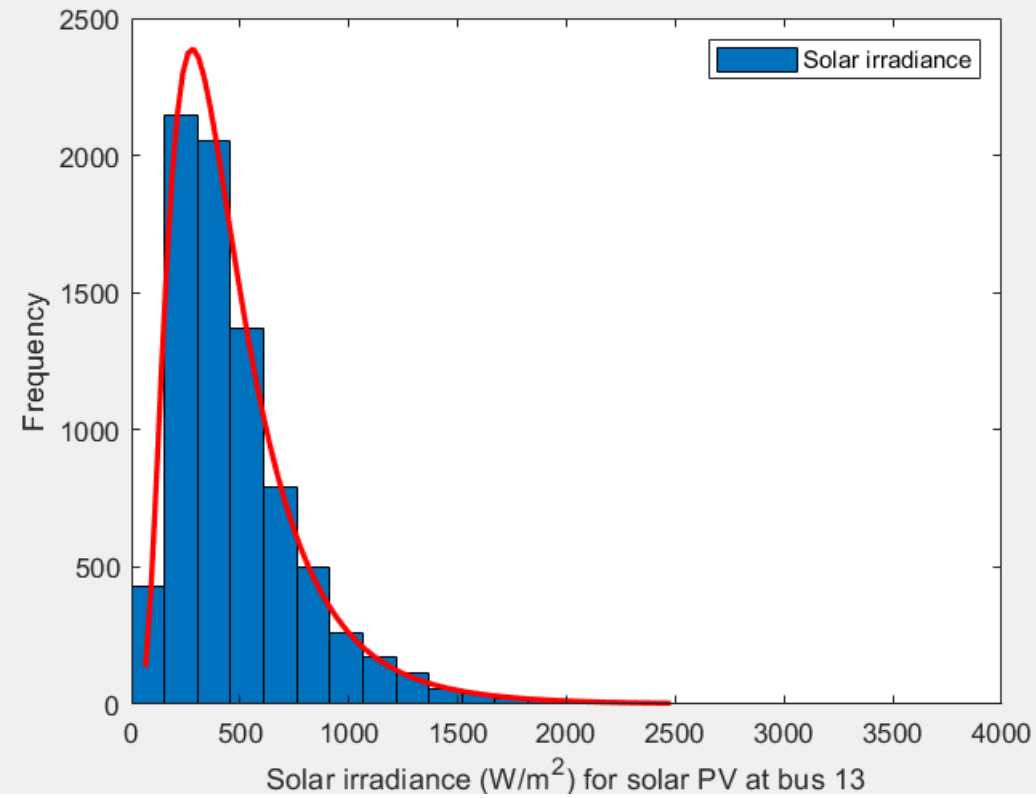


**Figure 4.1** Solar irradiance distrinution for solar PV unit at bus 13

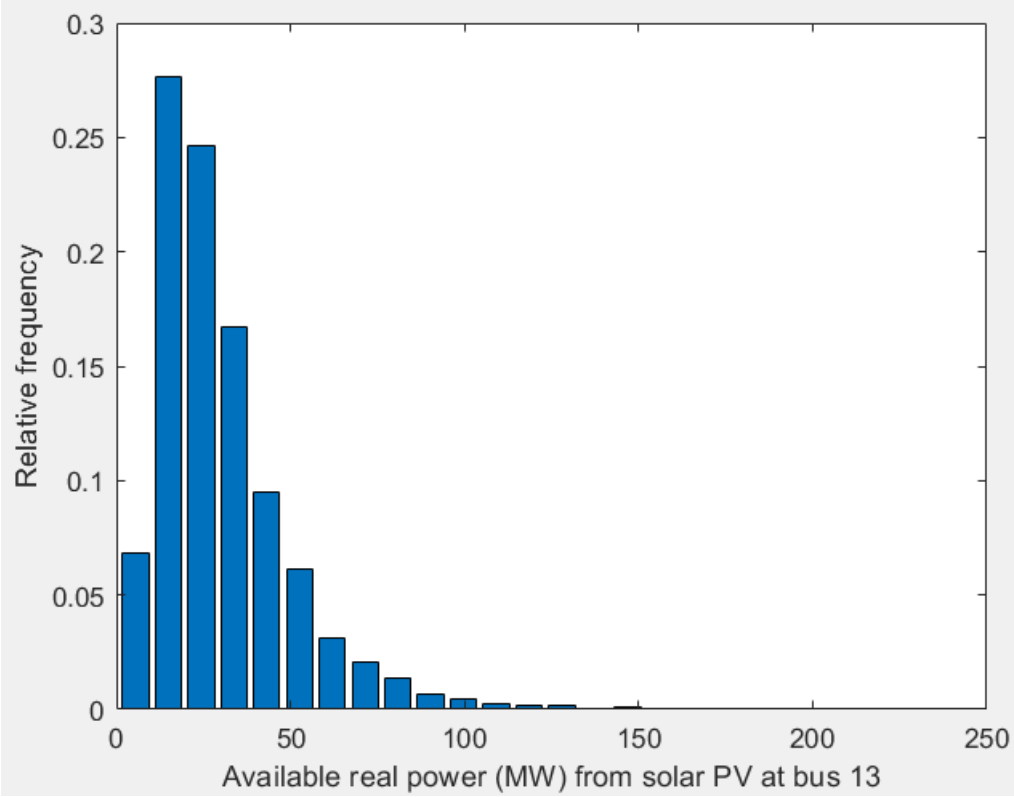


**Figure 4.2** Real power distribution for solar PV at bus 13

### 4.2.1 Cost minimization

**Table 4.3** Statistical result for case 1

| Algorithm | Max | Min | Mean | Std Dev. |
|---|---|---|---|---|
| MFO | 798.4958 | 791.2366 | 793.85425 | 1.354766161 |

**Table 4.4** Statistical result for case 2

| Algorithm | Max | Min | Mean | Std Dev. |
|---|---|---|---|---|
| MFO | 909.638 | 878.445 | 892.02 | 10.36354 |

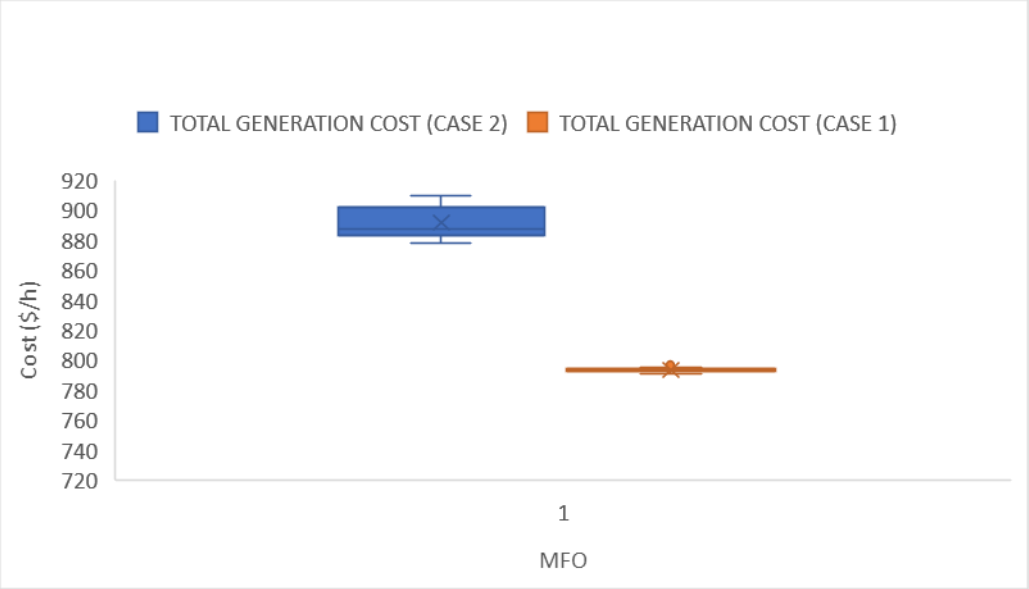


**Figure 4.3** Boxplot cost minimization for 30 runs of simulation for case 1 and case 2

### 4.2.2 Loss minimization

**Table 4.5** Statistical result for case 1

| Algorithm | Max | Min | Mean | Std Dev. |
|---|---|---|---|---|
| MFO | 5.9945 | 5.0609 | 5.390733 | 0.1619278 |

**Table 4.6** Statistical result for case 2

| Algorithm | Max | Min | Mean | Std Dev. |
|---|---|---|---|---|
| MFO | 2.2145 | 2.0818 | 2.13809 | 0.038792 |

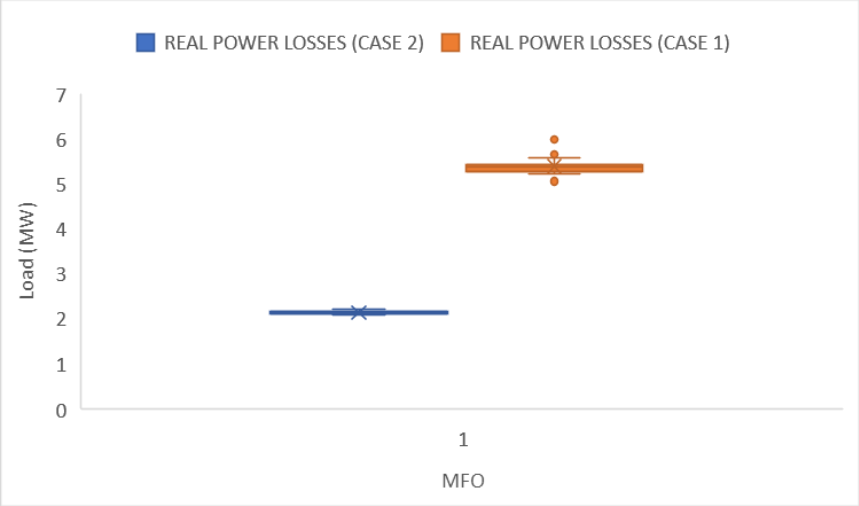


**Figure 4.4** Boxplot power losses for 30 runs of simulation of cae 1 and case 2

### 4.2.3 Voltage load bus

**Table 4.7** Result for voltage profiles of load bus for case 1 and case 2

| Voltage(p.u.) ca | Upper Lim | Lower Lim | Voltage (p |
|---|---|---|---|
| 1.05E+00 | 1.05 | 0.95 | 1.05E+00 |
| 1.04E+00 | 1.05 | 0.95 | 1.05E+00 |
| 1.04E+00 | 1.05 | 0.95 | 1.05E+00 |
| 1.04E+00 | 1.05 | 0.95 | 1.04E+00 |
| 1.04E+00 | 1.05 | 0.95 | 1.05E+00 |
| 1.02E+00 | 1.05 | 0.95 | 1.03E+00 |
| 1.03E+00 | 1.05 | 0.95 | 1.04E+00 |
| 1.01E+00 | 1.05 | 0.95 | 1.03E+00 |
| 1.01E+00 | 1.05 | 0.95 | 1.02E+00 |
| 1.02E+00 | 1.05 | 0.95 | 1.03E+00 |
| 1.01E+00 | 1.05 | 0.95 | 1.02E+00 |
| 9.99E-01 | 1.05 | 0.95 | 1.01E+00 |
| 9.97E-01 | 1.05 | 0.95 | 1.01E+00 |
| 1.00E+00 | 1.05 | 0.95 | 1.01E+00 |
| 1.00E+00 | 1.05 | 0.95 | 1.01E+00 |
| 1.00E+00 | 1.05 | 0.95 | 1.01E+00 |
| 9.97E-01 | 1.05 | 0.95 | 1.01E+00 |
| 9.89E-01 | 1.05 | 0.95 | 9.98E-01 |
| 9.92E-01 | 1.05 | 0.95 | 1.00E+00 |
| 9.74E-01 | 1.05 | 0.95 | 9.82E-01 |
| 1.00E+00 | 1.05 | 0.95 | 1.01E+00 |
| 1.04E+00 | 1.05 | 0.95 | 1.04E+00 |
| 9.83E-01 | 1.05 | 0.95 | 9.90E-01 |
| 9.71E-01 | 1.05 | 0.95 | 9.79E-01 |

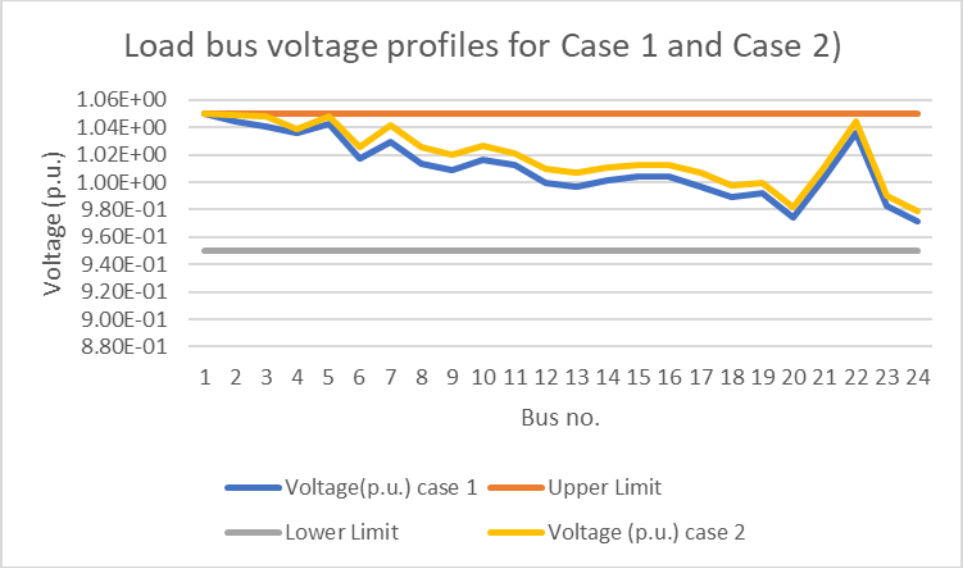


**Figure 4.5** Voltage profiles of load buses for moth flame algorithms for case 1 and case 2

## 4.3 Discussion

Minimization of generation cost performs optimization of generation schedule for all thermal and renewable energy source generator to minimize total generation cost given by Eq. (3.9). cost coefficient is shown in Table 4.1. All the optimum setting will be summarized into appendix that consist of all control variable, generator reactive power, total generation cost and other useful calculated parameter. Power loss and Voltage deviation calculated by Eq. (3.11) and (3.12).

Based on the table 4.3, the data that we got from the running programme for the case 1 which is cost minimization without carbon tax used equation from Eq (3.9). For the cost minimization, we got that the minimum value that can be achieve from running for 30 time by using MATLAB was 791.2366($/h), the maximum was 798.4958($/h), mean was 793.85425($/h) and standard deviation was 1.3548. Next, based on table 4.4 for case 2, that use equation from Eq. (3.10) we got that the value for the minimum was 878.4454($/h), maximum was 909.6375($/h), mean was 892.0202($/h) and standard deviation was 10.363542. As can see, the bit difference between case 1 and case 2 because for the case 1 need to add with carbon tax. For Carbon tax rate, $C_{tax}$ is assumed $20/tonne.

For the second objective is to minimize power losses of the power generation that consist of stochastic solar power generation, thermal generators and wind power generator. This objective use got from the Eq. (3.11). First, based on table 4.5 for the case 1 which is without carbon tax, we found that the minimum value was 5.0609MW, maximum was 5.9945MW, mean was 5.390733MW and standard deviation was 0.161927. And then, based on the table 4.6 for the case 2, the minimum value was 2.0818MW, maximum 2.2145MW, mean 2.138093MW and standard deviation was 0.0387925.

Figure (4.2) shows that the result for the load bus voltage profiles for case 1 and case 2. In optimal power flow problem, constraints on load bus is the one critical thing as operating voltage of load buses are often found be close their limits. For this research the lomit need to be maintained within 0.95p.u to 1.05p.u. If the limit can be reach upper and lower limit, need to give the penalty on the system because of violating the system. Based on the result, this project success without any violating the system which is power cant be reach more than or lower than set values. The value that set was 0.95 p.u until 1.05 p.u.

## 4.4 Summary

This project involved more data as stated in this chapter for the overall data in finding OPF value. A total of 30 simulations is made for testing the system, comparison in algorithm and records the obtained data. For data that has been recorded, this project reviewed and analysed to find the reliable correlation for optimal power flow solution.

In addition, this project aims to study the optimal power flow solution how it can give different result when it is combined as a new single objective under the name optimal power flow. From the data that recorded and analysed, researchers found that the hypothesis of the study can be accepted.

# CHAPTER 5

# CONCLUSION

## 5.1 Conclusion

In this report, various of data have been analysed for solving the economic emission load dispatch. This data later being used for calculation on 1 main items, optimal power flow using the MATLAB programming tools.

The application of the nature inspired algorithm namely Moth Flame Optimization in solving practical optimal power flow problems has been proposed in this paper. The possibility and effectiveness of the proposed method BMO was demonstrated on 30-Unit generating systems. From the simulations that have been done, it can be seen that BMO shows the result effectively and yet to be compared with other algorithm to show its effectiveness compared to any other algorithm in solving optimal power flow. Listed to be outperform most of the other algorithms based on the previous researches, MFO can obtain minimum operation cost and obtain close results compared to another algorithm.

Thus, the application of MFO is the best algorithm to solve the OPF problem compare to another algorithm by using different power sources consist of thermal, win and solar PV unit.

## REFERENCES


[1] M. H. Sulaiman, Z. Mustaffa, M. I. M. Rashid, and H. Daniyal, "Economic Dispatch Solution Using Moth-Flame Optimization Algorithm," MATEC Web Conferences, vol. 214, p. 03007, 2018

[2] G. Zwe-Lee, "Closure to "Discussion of 'Particle swarm optimization to solving the economic dispatch considering the generator constraints'"," IEEE Transactions on Power Systems, vol. 19, pp. 2122-2123, 2004

[3] Mohd Herwan Sulaiman, Zuriani Mustaffa, Mohd Mawardi Saari, Ahmad Johari Mohamad, Mohd Rusllim Mohamed, "Optimal power flow with stochastic solar power using barnacles mating optimizer" February 2021 DOI: 10.1002/2050- 7038.12858

[4] Seyedali Mirjalili, "Moth-Flame Optimization Algorithm: A Novel Nature-inspired Heuristic Paradigm" July 2015 Knowledge-Based Systems 89, DOI: 10.1016/j.knosys.2015.07.006

[5] Mohd Herwan Sulaiman, Zuriani Mustaffa, Mohd Mawardi Saari, Ahmad Johari Mohamad, Hamdan Daniyal, "Barnacles Mating Optimizer: A Bio-Inspired Algorithm for Solving Optimization Problems" Jume 2018 DOI: 10.1109/SNPD.2018.8441097

[6] C.-C. Kuo, "A novel string structure for economic dispatch problems with practical constraints," Energy Conversion and Management, vol. 49, pp. 3571-3577, 2008/12/01/ 2008

[7] Ning Ai, Bin Wu, Boyu Li, Zhipeng Zhao, "5G heterogeneous network selection and resource allocation optimization based on cuckoo search algorithm, Computer Communications", Volume 168, 2021, Pages 170-177, ISSN 0140-3664, https://doi.org/10.1016/j.comcom.2020.12.026.

[8] . Zare and T. G. Bolandi, "Modified iteration particle swarm optimization procedure for economic dispatch solving with non-smooth and non-convex fuel cost function," in 3rd IET International Conference on Clean Energy and Technology (CEAT) 2014, 2014, pp. 1-6.

[9] H. W. Dommel, W. F. Tinney, Optimal Power Flow Solutions, IEEE Transactions on power apparatus and systems, Vol. PAS.87, N°.10, pp.1866-1876, October 1968.

[10] M. Sasson, Non linear Programming Solutions for load flow, minimum loss, and economic dispatching problems, IEEE trans. Power apparatus and systems, Vol. Pas-88, N°4, April 1969.

[11] J. Kennedy and R. Eberhart, Particle swarm optimization . In : IEEE Int. Conf on Neural Networks, Perth, Australia, 1942-1948, 1995.

[12] Cao J, Du W, Wang HF, "Weather-Based Optimal Power Flow With Wind Farms Integration", IEEE Transactions on Power Systems, vol.31, no. 4, pp.3073-3081, Jul 2016.

[13] Jadhav HT, Roy R, "Stochastic optimal power flow incorporating offshore wind farm and electric vehicles", International Journal of Electrical Power & Energy Systems, vol.69, pp. 173-187, Jul 2015

[14] R. Fletcher, Practical methods of optimisation, John Willey & Sons, 1986.

[15] Biswas, P. P., Suganthan, P. N., & Amaratunga, G. A. J. (2017, July 4). Optimal Power Flow Solutions incorporating stochastic wind and solar power. Energy Conversion and Management. Retrieved February 10, 2022.

[16] Mishra, S., Yateendra Mishra, and S. Vignesh. Security constrained economic dispatch considering wind energy conversion systems. Power and Energy Society General Meeting, 2011 IEEE. IEEE, 2011.

[17] Chaib AE, Bouchekara HREH, Mehasni R, Abido MA. Optimal power flow with emission and non-smooth cost functions using backtracking search optimization algorithm. Int J Electr Power Energy Syst 2016;81:64–77.

[18] Mohamed AAA, Mohamed YS, El-Gaafary AA, Hemeida AM. Optimal power flow using moth swarm algorithm. Electric Power Sys Res 2017;142:190–206.

[19] Shi L, Wang C, Yao L, Ni Y, Bazargan M. Optimal power flow solution incorporating wind power. IEEE Syst J 2012;6(2):233–41.

[20] Chen Chun-Lung, Lee Tsung-Ying, Jan Rong-Mow. Optimal wind-thermal coordination dispatch in isolated power systems with large integration of wind capacity. Energy Convers Manage 2006;47(18):3456–72.

[21] C S, T A. Optimal power flow using moth swarm algorithm with gravitational search algorithm considering wind power. Futur Gener Comput Syst. 2019;98:708-715.

[22] Joorabian M, Afzalan E. Optimal power flow under both normal and contingent operation conditions using the hybrid fuzzy particle swarm optimisation and Nelder Mead algorithm (HFPSO–NM). Appl Soft Comput. 2014;14:623-633.

[23] Trivedi IN, Jangir P, Parmar SA, Jangir N. Optimal power flow with voltage stability improvement and loss reduction in power system using moth-flame optimizer. Neural Comput & Applic. 2018;30(6):1889-1904.

[24] Taher MA, Kamel S, Jurado F, Ebeed M. An improved moth-flame optimization algorithm for solving optimal power flow problem. Int Trans Electr Energy Syst. 2019;29(3):e2743.

# APPENDIX A

**Table A1** Simulation result for optimization for case 1 and case 2

| Control variables | Min | Max | Case 1 | Case 2 | Parameters | Min | Max | Case 1 | Case 2 |
|---|---|---|---|---|---|---|---|---|---|
| P_TG1(MW) | 50 | 140 | 123.8 | 50.01 | Q_TG1(MVAr) | -20 | 150 | 25.76 | -20 |
| P_TG2(MW) | 20 | 80 | 31.48 | 22.14 | Q_TG2(MVAr) | -20 | 60 | -20 | 22.19 |
| P_TG3(MW) | 10 | 35 | 10 | 35 | Q_TG3(MVAr) | -15 | 40 | 40 | 37.92 |
| P_ws1(MW) | 0 | 75 | 45.83 | 75 | Q_ws1(MVAr) | -30 | 35 | 35 | 18.13 |
| P_ws2(MW) | 0 | 60 | 38.97 | 60 | Q_ws2(MVAr) | -25 | 30 | 18.23 | 30 |
| P_ss(MW) | 0 | 50 | 38.77 | 43.36 | Q_ss(MVAr) | -20 | 25 | 16.67 | 21.37 |
| V_1(p.u.) | 0.95 | 1.1 | 1.078 | 1.054 | Total cost ($/h) | | | **796.9648** | **887.4973** |
| V_2(p.u.) | 0.95 | 1.1 | 1.052 | 1.055 | Emission (t/h) | | | 0.80746 | 0.9925 |
| V_5(p.u.) | 0.95 | 1.1 | 1.044 | 1.042 | Carbon tax ($/h) | | | - | 20 |
| V_8(p.u.) | 0.95 | 1.1 | 1.04 | 1.051 | P_loss(MW) | | | **5.3456** | **2.11** |
| V_11(p.u.) | 0.95 | 1.1 | 1.06 | 1.098 | VD (p.u.) | | | 0.455 | 0.55685 |
| V_13(p.u.) | 0.95 | 1.1 | 1.046 | 1.067 | | | | | |

## APPENDIX B

- **MFOOPF.m**

```
global  nfeval VD PGS emission ploss VI fuelvlvcost Qgen wgencost sgencost
cumcost
format long e;
tic
problem_size = 11;
pop_size = 30;
N = pop_size;
dim = problem_size;
Xmin = [20 0 10 0 0 0.95 0.95 0.95 0.95 0.95 0.95];
Xmax = [80 75 35 60 50 1.1 1.1 1.1 1.1 1.1 1.1];
lb = Xmin;
ub = Xmax;
Max_FES = 24000;
Max_Gen = 100; Max_iteration = Max_Gen; maxrun = 1; runs = maxrun; gn =
4; % gn is the no. of constraints
feval = []; contvar = zeros(maxrun,problem_size);
Re = zeros(maxrun,3); W = zeros(Max_Gen,3);

fobj = @pflow;

%  for runs=1:maxrun
  weibullplot(); % for plotting Weibull PDF
  lognplot(); % for plotting lognormal PDF
  fprintf('Run %d',runs);
  nfeval=0;

 %%  parameter settings for MFO
%Initialize the positions of moths
Moth_pos=initialization(N,dim,ub,lb);

Convergence_curve=zeros(1,Max_iteration);

 %% Main loop

for gen=1:Max_Gen
Iteration = gen;
    % Number of flames Eq. (3.14) in the paper
   Flame_no=round(N-Iteration*((N-1)/Max_iteration));
```

```
    for i=1:size(Moth_pos,1)

        % Check if moths go out of the search spaceand bring it back
        Flag4ub=Moth_pos(i,:)>ub;
        Flag4lb=Moth_pos(i,:)<lb;

Moth_pos(i,:)=(Moth_pos(i,:).*(~(Flag4ub+Flag4lb)))+ub.*Flag4ub+lb.*Flag4l
b;

        % Calculate the fitness of moths
        Moth_fitness(1,i)=fobj(Moth_pos(i,:));

    end

    if Iteration==1
        % Sort the first population of moths
        [fitness_sorted I]=sort(Moth_fitness);
        sorted_population=Moth_pos(I,:);

        % Update the flames
        best_flames=sorted_population;
        best_flame_fitness=fitness_sorted;
    else

        % Sort the moths
        double_population=[previous_population;best_flames];
        double_fitness=[previous_fitness best_flame_fitness];

        [double_fitness_sorted I]=sort(double_fitness);
        double_sorted_population=double_population(I,:);

        fitness_sorted=double_fitness_sorted(1:N);
        sorted_population=double_sorted_population(1:N,:);

        % Update the flames
        best_flames=sorted_population;
        best_flame_fitness=fitness_sorted;
    end

    % Update the position best flame obtained so far
    Best_flame_score=fitness_sorted(1);
    Best_flame_pos=sorted_population(1,:);

    previous_population=Moth_pos;
```

```
    previous_fitness=Moth_fitness;

    % a linearly dicreases from -1 to -2 to calculate t in Eq. (3.12)
    a=-1+Iteration*((-1)/Max_iteration);

    for i=1:size(Moth_pos,1)

        for j=1:size(Moth_pos,2)
            if i<=Flame_no % Update the position of the moth with respect to its
corresponsing flame

                % D in Eq. (3.13)
                distance_to_flame=abs(sorted_population(i,j)-Moth_pos(i,j));
                b=1;
                t=(a-1)*rand+1;

                % Eq. (3.12)

Moth_pos(i,j)=distance_to_flame*exp(b.*t).*cos(t.*2*pi)+sorted_population(i,j
);
            end

            if i>Flame_no % Upaate the position of the moth with respct to one
flame

                % Eq. (3.13)
                distance_to_flame=abs(sorted_population(i,j)-Moth_pos(i,j));
                b=1;
                t=(a-1)*rand+1;

                % Eq. (3.12)

Moth_pos(i,j)=distance_to_flame*exp(b.*t).*cos(t.*2*pi)+sorted_population(Fl
ame_no,j);
            end

        end

    end

    Convergence_curve(Iteration)=Best_flame_score;
    plot(Convergence_curve)
    grid on
```

```
    % Display the iteration and best optimum obtained so far
    if mod(Iteration,50)==0
        display(['At iteration ', num2str(Iteration), ' the best fitness is ',
num2str(Best_flame_score)]);
    end
    Iteration=Iteration+1;

 end

toc

% Print results
[~,IND] = min(Re(:,1));
% disp('Control Variables');
% fprintf('\t %0.4f',contvar(IND,:));
% Func(contvar(IND,:));
fprintf('\n Swing generator power %0.5f MW',PGS);
fprintf('\n Cumulative voltage drop %0.5f p.u.',VD);
fprintf('\n Emission %0.5f ton/h',emission);
fprintf('\n Real power loss %0.4f MW',ploss);
fprintf('\n Fuelvlvcost %0.4f \n',fuelvlvcost);
fprintf('\n Wind gen cost %0.4f \n',wgencost);
fprintf('\n Solar gen cost %0.4f \n',sgencost);
fprintf('\n Total generation cost %0.4f \n',cumcost);
disp(' Load bus voltage');
fprintf('\t %0.5f',VI);
fprintf('\n Generator reactive power');
fprintf('\t %0.5f',Qgen);
```

- **pflow.m**

```
function [f1,error] = pflow(x)

global VD PGS emission ploss VI scale shape fuelvlvcost Qgen mcarlo SP1
wgencost sgencost nbins cumcost
%x = [27.0699 42.9420 10.0000 36.2679 38.0055
1.0720 1.0570 1.0348 1.0396 1.0982 1.0556];
%x = [20.58 60 35 60 60 1.0543 1.04764 1.03343 1.04372 1.0802 1.07762];
%x = [62.84 50 24.64 40 40 1.0591 1.05273 1.03364 1.0355 1.071 1.08075];

% scale = [9 10 11]; % Enter shape parameters of 3 windfarms for Weibull dist
% shape = [2 2 2]; % Enter shape parameters of 3 windfarms for Weibull dist
NT = [25 20]; % No. of turbines in the 2 farms
Vin = 3; Vout = 25; Vr = 16; Pr = 3; % Cut-in, cut-out, rated speed and rated
power of turbine
Ctax = 20; % Carbon tax $/ton

data = loadcase(case30);
data.gen(2:6,2) = x(1:5);
data.gen(1:6,6) = x(6:11);

mpopt = mpoption('pf.enforce_q_lims',2,'verbose',0,'out.all',0);
result = runpf(data,mpopt);

thpowgen = [result.gen(1,2),x(1),x(3)];
%thpowgen = [94.7753,x(1),x(3)];
thgencoeff = vertcat(data.gencost(1:2,5:7),data.gencost(4,5:7));

thgencost =
sum(thgencoeff(:,1)+thgencoeff(:,2).*thpowgen'+thgencoeff(:,3).*(thpowgen.^2
)'); % thermal generator cost

%Find wind generator related parameters
%windgen parameter sl no. bus costcoeff
wgenpar = [1   5   1.60;
        2   11  1.75];
Crwj = 3; Cpwj = 1.5; % wind power penalty and reserve cost coefficients
schwpow = [x(2),x(4)]';

%stochastic wind power cost
%meanwpow = [26;30]; wp = 35;
Prw0 = 1-exp(-(Vin./scale).^shape)+exp(-(Vout./scale).^shape);
Prwwr = exp(-(Vr./scale).^shape)-exp(-(Vout./scale).^shape);
```

```
count1 = 1;
wovest = zeros(); wundest = zeros();

for ii = 1:2
   Prww1 = (shape(ii)*(Vr-Vin))/((scale(ii)^shape(ii))*(NT(ii)*Pr));
   Prww = @(wp)((schwpow(ii)-wp)*Prww1*((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))^(shape(ii)-1))*(exp(-((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))/scale(ii))^shape(ii))));
   wovest2 = integral(Prww,0,schwpow(ii),'ArrayValued',true);
   wovest(count1) = schwpow(ii)*Prw0(ii)*Crwj+Crwj*wovest2;

   Prww = @(wp)((wp-schwpow(ii))*Prww1*((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))^(shape(ii)-1))*(exp(-((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))/scale(ii))^shape(ii))));
   wundest2 = integral(Prww,schwpow(ii),NT(ii)*Pr,'ArrayValued',true);
   wundest(count1) = (NT(ii)*Pr-
schwpow(ii))*Prwwr(ii)*Cpwj+Cpwj*wundest2;
   count1 = count1+1;
end

wgencost = sum(wgenpar(:,3).*schwpow)+sum(wovest)+sum(wundest); %
wind generator cost

%solargen parameter sl no. bus costcoeff
sgenpar = [1   13  1.60];
Crsj = 3; % Reserve cost for solar power overestimation ($/MW)
Cpsj = 1.5; % Penalty cost for solar power underestimation ($/MW)
schspow = x(5); % solar generator schedule power

% Segregate over and underestimated power on the power histogram

[histy1,histx1] = hist(SP1,nbins);

Lowind1 = histx1<schspow;
Highind1 = histx1>schspow;
allP1und = schspow-histx1(histx1<schspow);
allP1over = histx1(histx1>schspow)-schspow;
ProbP1und = histy1(Lowind1)./mcarlo;
ProbP1over = histy1(Highind1)./mcarlo;

% Finding under and over estimation cost
C1und = sum(Crsj*(ProbP1und.*allP1und));
C1over = sum(Cpsj*(ProbP1over.*allP1over));
sovundcost = [C1und,C1over];
```

```
sgencost = sum(sgenpar(:,3).*schspow)+sum(sovundcost); % solar generator
cost

%Constraint finding
Vmax = data.bus(:,12);
Vmin = data.bus(:,13);
genbus = data.gen(:,1);

Qmax = data.gen(:,4)/data.baseMVA;
Qmin = data.gen(:,5)/data.baseMVA;
Qgen = result.gen(:,3);
QG = result.gen(:,3)/data.baseMVA;

PGSmax = data.gen(1,9);
PGSmin = data.gen(1,10);
PGS = result.gen(1,2);
PGSerr = (PGS<PGSmin)*(abs(PGSmin-PGS)/(PGSmax-
PGSmin))+(PGS>PGSmax)*(abs(PGSmax-PGS)/(PGSmax-PGSmin));

blimit = data.branch(:,6);
Slimit = sqrt(result.branch(:,14).^2+result.branch(:,15).^2);
Serr = sum((Slimit>blimit).*abs(blimit-Slimit))/data.baseMVA;

% TO find the error in Qg of gen buses- inequality constraint
Qerr = sum((QG<Qmin).*(abs(Qmin-QG)./(Qmax-
Qmin))+(QG>Qmax).*(abs(Qmax-QG)./(Qmax-Qmin)));
% TO find the error in V of load buses-inequality constraint
VI = result.bus(:,8);  %V of load buses-inequality constraint
VI(genbus)=[];
Vmax(genbus)=[];
Vmin(genbus)=[];
VIerr = sum((VI<Vmin).*(abs(Vmin-VI)./(Vmax-
Vmin))+(VI>Vmax).*(abs(Vmax-VI)./(Vmax-Vmin)));
VD = sum(abs(VI-1));

% Emission : Of thermal generating unit
% bus_no. alpha beta gama omega miu d e Pmin
 emcoeff = [
        1       0.04091 -0.05554 0.06490 0.000200 6.667 18 0.037 50;
        2       0.02543 -0.06047 0.05638 0.000500 3.333 16 0.038 20;
        8       0.05326 -0.03550 0.03380 0.002000 2.000 12 0.045 10];

% VALVE EFFECT
```

```
valveff = sum(abs(emcoeff(:,7).*sin(emcoeff(:,8).*(emcoeff(:,9)-thpowgen'))));
% if all have valve effects

% OBJECTIVE FUNCTIONS
emission =
sum(emcoeff(:,2)+emcoeff(:,3).*thpowgen'/100+emcoeff(:,4).*(thpowgen.^2/10
0^2)'...
    +emcoeff(:,5).*exp(emcoeff(:,6).*thpowgen'/100));

ploss = sum(result.branch(:,14)+result.branch(:,16));

fuelvlvcost = thgencost+valveff;
cumcost = fuelvlvcost+wgencost+sgencost;
error = [Qerr,VIerr,Serr,PGSerr];
if Qerr~=0
   PF1=1000;
else, PF1=0;
end
if VIerr~=0
   PF2=1000;
else, PF2=0;
end
if Serr~=0
   PF3=1000;
else, PF3=0;
end
if PGSerr~=0
   PF4=1000;
else, PF4=0;
end
f1 = ploss+PF1+PF2+PF3+PF4; % CASE 1: fuel cost only
% f1 = cumcost+(Ctax*emission)+PF1+PF2+PF3+PF4; % CASE 2:
fuelcost+carbon tax
```

- **pflow_plot.m**

```
function [f1,error,result] = pflow_plot(x)

global VD PGS emission ploss VI scale shape fuelvlvcost Qgen mcarlo SP1
wgencost sgencost nbins cumcost
%x = [27.0699 42.9420          10.0000          36.2679          38.0055
1.0720  1.0570           1.0348           1.0396           1.0982           1.0556];
%x = [20.58 60 35 60 60 1.0543 1.04764 1.03343 1.04372 1.0802 1.07762];
%x = [62.84 50 24.64 40 40 1.0591 1.05273 1.03364 1.0355 1.071 1.08075];

% scale = [9 10 11]; % Enter shape parameters of 3 windfarms for Weibull dist
% shape = [2 2 2]; % Enter shape parameters of 3 windfarms for Weibull dist
NT = [25 20]; % No. of turbines in the 2 farms
Vin = 3; Vout = 25; Vr = 16; Pr = 3; % Cut-in, cut-out, rated speed and rated
power of turbine
Ctax = 20; % Carbon tax $/ton

data = loadcase(case30);
data.gen(2:6,2) = x(1:5);
data.gen(1:6,6) = x(6:11);

mpopt = mpoption('pf.enforce_q_lims',2,'verbose',0,'out.all',0);
result = runpf(data,mpopt);

thpowgen = [result.gen(1,2),x(1),x(3)];
%thpowgen = [94.7753,x(1),x(3)];
thgencoeff = vertcat(data.gencost(1:2,5:7),data.gencost(4,5:7));

thgencost =
sum(thgencoeff(:,1)+thgencoeff(:,2).*thpowgen'+thgencoeff(:,3).*(thpowgen.^2
)'); % thermal generator cost

%Find wind generator related parameters
%windgen parameter sl no. bus costcoeff
wgenpar = [1   5   1.60;
        2   11  1.75];
Crwj = 3; Cpwj = 1.5; % wind power penalty and reserve cost coefficients
schwpow = [x(2),x(4)]';

%stochastic wind power cost
%meanwpow = [26;30]; wp = 35;
Prw0 = 1-exp(-(Vin./scale).^shape)+exp(-(Vout./scale).^shape);
Prwwr = exp(-(Vr./scale).^shape)-exp(-(Vout./scale).^shape);
```

```
count1 = 1;
wovest = zeros(); wundest = zeros();

for ii = 1:2
   Prww1 = (shape(ii)*(Vr-Vin))/((scale(ii)^shape(ii))*(NT(ii)*Pr));
   Prww = @(wp)((schwpow(ii)-wp)*Prww1*((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))^(shape(ii)-1))*(exp(-((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))/scale(ii))^shape(ii))));
   wovest2 = integral(Prww,0,schwpow(ii),'ArrayValued',true);
   wovest(count1) = schwpow(ii)*Prw0(ii)*Crwj+Crwj*wovest2;

   Prww = @(wp)((wp-schwpow(ii))*Prww1*((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))^(shape(ii)-1))*(exp(-((Vin + (wp/(NT(ii)*Pr))*(Vr-
Vin))/scale(ii))^shape(ii))));
   wundest2 = integral(Prww,schwpow(ii),NT(ii)*Pr,'ArrayValued',true);
   wundest(count1) = (NT(ii)*Pr-
schwpow(ii))*Prwwr(ii)*Cpwj+Cpwj*wundest2;
   count1 = count1+1;
end

wgencost = sum(wgenpar(:,3).*schwpow)+sum(wovest)+sum(wundest); %
wind generator cost

%solargen parameter sl no. bus costcoeff
sgenpar = [1   13  1.60];
Crsj = 3; % Reserve cost for solar power overestimation ($/MW)
Cpsj = 1.5; % Penalty cost for solar power underestimation ($/MW)
schspow = x(5); % solar generator schedule power

% Segregate over and underestimated power on the power histogram

[histy1,histx1] = hist(SP1,nbins);

Lowind1 = histx1<schspow;
Highind1 = histx1>schspow;
allP1und = schspow-histx1(histx1<schspow);
allP1over = histx1(histx1>schspow)-schspow;
ProbP1und = histy1(Lowind1)./mcarlo;
ProbP1over = histy1(Highind1)./mcarlo;

% Finding under and over estimation cost
C1und = sum(Crsj*(ProbP1und.*allP1und));
C1over = sum(Cpsj*(ProbP1over.*allP1over));
sovundcost = [C1und,C1over];
```

```
sgencost = sum(sgenpar(:,3).*schspow)+sum(sovundcost); % solar generator
cost

%Constraint finding
Vmax = data.bus(:,12);
Vmin = data.bus(:,13);
genbus = data.gen(:,1);

Qmax = data.gen(:,4)/data.baseMVA;
Qmin = data.gen(:,5)/data.baseMVA;
Qgen = result.gen(:,3);
QG = result.gen(:,3)/data.baseMVA;

PGSmax = data.gen(1,9);
PGSmin = data.gen(1,10);
PGS = result.gen(1,2);
PGSerr = (PGS<PGSmin)*(abs(PGSmin-PGS)/(PGSmax-
PGSmin))+(PGS>PGSmax)*(abs(PGSmax-PGS)/(PGSmax-PGSmin));

blimit = data.branch(:,6);
Slimit = sqrt(result.branch(:,14).^2+result.branch(:,15).^2);
Serr = sum((Slimit>blimit).*abs(blimit-Slimit))/data.baseMVA;

% TO find the error in Qg of gen buses- inequality constraint
Qerr = sum((QG<Qmin).*(abs(Qmin-QG)./(Qmax-
Qmin))+(QG>Qmax).*(abs(Qmax-QG)./(Qmax-Qmin)));
% TO find the error in V of load buses-inequality constraint
VI = result.bus(:,8);  %V of load buses-inequality constraint
VI(genbus)=[];
Vmax(genbus)=[];
Vmin(genbus)=[];
VIerr = sum((VI<Vmin).*(abs(Vmin-VI)./(Vmax-
Vmin))+(VI>Vmax).*(abs(Vmax-VI)./(Vmax-Vmin)));
VD = sum(abs(VI-1));

% Emission : Of thermal generating unit
% bus_no. alpha beta gama omega miu d e Pmin
 emcoeff = [
        1       0.04091 -0.05554 0.06490 0.000200 6.667 18 0.037 50;
        2       0.02543 -0.06047 0.05638 0.000500 3.333 16 0.038 20;
        8       0.05326 -0.03550 0.03380 0.002000 2.000 12 0.045 10];

% VALVE EFFECT
```

```
valveff = sum(abs(emcoeff(:,7).*sin(emcoeff(:,8).*(emcoeff(:,9)-thpowgen'))));
% if all have valve effects

% OBJECTIVE FUNCTIONS
emission =
sum(emcoeff(:,2)+emcoeff(:,3).*thpowgen'/100+emcoeff(:,4).*(thpowgen.^2/10
0^2)'...
    +emcoeff(:,5).*exp(emcoeff(:,6).*thpowgen'/100));

ploss = sum(result.branch(:,14)+result.branch(:,16));

fuelvlvcost = thgencost+valveff;
cumcost = fuelvlvcost+wgencost+sgencost;
error = [Qerr,VIerr,Serr,PGSerr];
if Qerr~=0
   PF1=1000;
else, PF1=0;
end
if VIerr~=0
   PF2=1000;
else, PF2=0;
end
if Serr~=0
   PF3=1000;
else, PF3=0;
end
if PGSerr~=0
   PF4=1000;
else, PF4=0;
end

f1 = ploss+PF1+PF2+PF3+PF4; % CASE 1: fuel cost only
% f1 = cumcost+(Ctax*emission)+PF1+PF2+PF3+PF4; % CASE 2:
fuelcost+carbon tax
```

- **initialization.m**

```
%_____
_____
%  Moth-Flame Optimization Algorithm (MFO)
%_____
_____

% This function creates the first random population of moths

function X=initialization(SearchAgents_no,dim,ub,lb)

Boundary_no= size(ub,2); % numnber of boundaries

% If the boundaries of all variables are equal and user enter a signle
% number for both ub and lb
if Boundary_no==1
   X=rand(SearchAgents_no,dim).*(ub-lb)+lb;
end

% If each variable has a different lb and ub
if Boundary_no>1
   for i=1:dim
      ub_i=ub(i);
      lb_i=lb(i);
      X(:,i)=rand(SearchAgents_no,1).*(ub_i-lb_i)+lb_i;
   end
end
```

- **lognplot.m**

```
%_____________________________________________________________
_____
%  Moth-Flame Optimization Algorithm (MFO)
%_____________________________________________________________
_____

% This function creates the first random population of moths

function X=initialization(SearchAgents_no,dim,ub,lb)

Boundary_no= size(ub,2); % numnber of boundaries

% If the boundaries of all variables are equal and user enter a signle
% number for both ub and lb
if Boundary_no==1
   X=rand(SearchAgents_no,dim).*(ub-lb)+lb;
end

% If each variable has a different lb and ub
if Boundary_no>1
   for i=1:dim
      ub_i=ub(i);
      lb_i=lb(i);
      X(:,i)=rand(SearchAgents_no,1).*(ub_i-lb_i)+lb_i;
   end
end
```

- **weinullplot.m**

```
function weibullplot()
global scale shape
scale = [9 10]; % Enter shape parameters of 2 windfarms for Weibull dist
shape = [2 2]; % Enter shape parameters of 2 windfarms for Weibull dist

W1 = wblrnd(scale(1),shape(1),8000,1);
figure(1)
histfit(W1,30,'weibull')
xlabel('Wind speed (m/s) for wind generator at bus 5')
ylabel('Frequency')
legend ('Wind distribution')

W2 = wblrnd(scale(2),shape(2),8000,1);
figure(2)
histfit(W2,30,'weibull')
xlabel('Wind speed (m/s) for wind generator at bus 11')
ylabel('Frequency')
legend ('Wind distribution')
```

- **boundConstraint.m**

```
function vi = boundConstraint (vi, pop, X_max, X_min)

% if the boundary constraint is violated, set the value to be the middle
% of the previous value and the bound


[NP, D] = size(pop);  % the population size and the problem's dimension

%% check the lower bound
xl = repmat(X_min, NP, 1);
pos = vi < xl;
vi(pos) = (pop(pos) + xl(pos)) / 2;

%% check the upper bound
xu = repmat(X_max, NP, 1);
pos = vi > xu;
vi(pos) = (pop(pos) + xu(pos)) / 2;
```

- **case30.m**

```
function mpc = case30
%CASE30    Power flow data for 30 bus, 6 generator case.
%   Please see CASEFORMAT for details on the case file format.
%
%   Based on data from ...
%     Alsac, O. & Stott, B., "Optimal Load Flow with Steady State Security",
%     IEEE Transactions on Power Apparatus and Systems, Vol. PAS 93, No. 3,
%     1974, pp. 745-751.
%   ... with branch parameters rounded to nearest 0.01, shunt values divided
%   by 100 and shunt on bus 10 moved to bus 5, load at bus 5 zeroed out.
%   Generator locations, costs and limits and bus areas were taken from ...
%     Ferrero, R.W., Shahidehpour, S.M., Ramesh, V.C., "Transaction analysis
%     in deregulated power systems using game theory", IEEE Transactions on
%     Power Systems, Vol. 12, No. 3, Aug 1997, pp. 1340-1347.
%   Generator Q limits were derived from Alsac & Stott, using their Pmax
%   capacities. V limits and line |S| limits taken from Alsac & Stott.

%   MATPOWER

%% MATPOWER Case Format : Version 2
mpc.version = '2';

%%-----  Power Flow Data  -----%%
%% system MVA base
mpc.baseMVA = 100;

%% bus data
%       bus_i   type    Pd      Qd      Gs      Bs      area    Vm      Va
        baseKV          zone    Vmax    Vmin
mpc.bus = [
        1       3       0       0       0       0       1       1       0       135     1
        1.1     0.95;
        2       2       21.7    12.7    0       0       1       1       0       135
1       1.1     0.95;
        3       1       2.4     1.2     0       0       1       1       0       135     1
        1.05    0.95;
        4       1       7.6     1.6     0       0       1       1       0       135     1
        1.05    0.95;
        5       2       94.2    19      0       0.19    1       1       0       135     1
        1.1     0.95;
        6       1       0       0       0       0       1       1       0       135     1
        1.05    0.95;
```

```
        7       1       22.8    10.9    0       0       1       1       0       135
1       1.05    0.95;
        8       2       30      30      0       0       1       1       0       135     1
        1.10    0.95;
        9       1       0       0       0       0       1       1       0       135     1
        1.05    0.95;
        10      1       5.8     2       0       0       3       1       0       135     1
        1.05    0.95;
        11      2       0       0       0       0       1       1       0       135     1
        1.10    0.95;
        12      1       11.2    7.5     0       0       2       1       0       135
1       1.05    0.95;
        13      2       0       0       0       0       2       1       0       135     1
        1.10    0.95;
        14      1       6.2     1.6     0       0       2       1       0       135     1
        1.05    0.95;
        15      1       8.2     2.5     0       0       2       1       0       135     1
        1.05    0.95;
        16      1       3.5     1.8     0       0       2       1       0       135     1
        1.05    0.95;
        17      1       9       5.8     0       0       2       1       0       135     1
        1.05    0.95;
        18      1       3.2     0.9     0       0       2       1       0       135     1
        1.05    0.95;
        19      1       9.5     3.4     0       0       2       1       0       135     1
        1.05    0.95;
        20      1       2.2     0.7     0       0       2       1       0       135     1
        1.05    0.95;
        21      1       17.5    11.2    0       0       3       1       0       135
1       1.05    0.95;
        22      1       0       0       0       0       3       1       0       135     1
        1.05    0.95;
        23      1       3.2     1.6     0       0       2       1       0       135     1
        1.05    0.95;
        24      1       8.7     6.7     0       0.04    3       1       0       135     1
        1.05    0.95;
        25      1       0       0       0       0       3       1       0       135     1
        1.05    0.95;
        26      1       3.5     2.3     0       0       3       1       0       135     1
        1.05    0.95;
        27      1       0       0       0       0       3       1       0       135     1
        1.05    0.95;
        28      1       0       0       0       0       1       1       0       135     1
        1.05    0.95;
```

```
        29      1       2.4     0.9     0  0       3       1       0       135     1
        1.05    0.95;
        30      1       10.6    1.9     0       0       3       1       0       135
1       1.05    0.95;
];

%% generator data
%       bus     Pg      Qg      Qmax    Qmin    Vg      mBase status  Pmax    Pmin
        Pc1     Pc2     Qc1min          Qc1max          Qc2min          Qc2max
        ramp_agc        ramp_10         ramp_30         ramp_q          apf
mpc.gen = [
        1   99.211    -3.99     150.0   -20     1.0     100     1       140     50 0
0 0 0 0 0 0 0 0 0;
        2   80.00     50.0      60.0    -20     1.0     100     1       80      20 0
0 0 0 0 0 0 0 0 0;
        5   50.00     37.0      35.0    -30     1.0     100     1       60      10 0
0 0 0 0 0 0 0 0 0;
        8   20.00     37.3      40.0    -15     1.0      100    1       35      10 0
0 0 0 0 0 0 0 0 0;
   11   20.00     37.3    30.0    -25     1.0     100     1       60      10 0 0 0 0 0
0 0 0 0 0;
   13   20.00     37.3    25.0    -20     1.0     100     1       60      10 0 0 0 0 0
0 0 0 0 0;
];

%% branch data
%       fbus    tbus    r       x       b       rateA   rateB   rateC   ratio   angle
        status  angminangmax
mpc.branch = [
1       2       0.0192  0.0575  0.0528  130     130     130     0       0       1
        -360    360;
1       3       0.0452  0.1652  0.0408  130     130     130     0       0       1
        -360    360;
2       4       0.057   0.1737  0.0368  65      65      65      0       0       1
        -360    360;
3       4       0.0132  0.0379  0.0084  130     130     130     0       0       1
        -360    360;
2       5       0.0472  0.1983  0.0418  130     130     130     0       0       1
        -360    360;
2       6       0.0581  0.1763  0.0374  65      65      65      0       0       1
        -360    360;
4       6       0.0119  0.0414  0.009   90      90      90      0       0       1
        -360    360;
```

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 7 | 0.046 | 0.116 | 0.0204 | 70 | 70 | 70 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 6 | 7 | 0.0267 | 0.082 | 0.017 | 130 | 130 | 130 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 6 | 8 | 0.012 | 0.042 | 0.009 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 6 | 9 | 0.00 | 0.208 | 0.00 | 65 | 65 | 65 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 6 | 10 | 0.00 | 0.556 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 9 | 11 | 0.00 | 0.208 | 0.00 | 65 | 65 | 65 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 9 | 10 | 0.00 | 0.11 | 0.00 | 65 | 65 | 65 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 4 | 12 | 0.00 | 0.256 | 0.00 | 65 | 65 | 65 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 12 | 13 | 0.00 | 0.14 | 0.00 | 65 | 65 | 65 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 12 | 14 | 0.1231 | 0.2559 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 12 | 15 | 0.0662 | 0.1304 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 12 | 16 | 0.0945 | 0.1987 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 14 | 15 | 0.221 | 0.1997 | 0.00 | 16 | 16 | 16 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 16 | 17 | 0.0524 | 0.1923 | 0.00 | 16 | 16 | 16 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 15 | 18 | 0.1073 | 0.2185 | 0.00 | 16 | 16 | 16 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 18 | 19 | 0.0639 | 0.1292 | 0.00 | 16 | 16 | 16 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 19 | 20 | 0.034 | 0.068 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 10 | 20 | 0.0936 | 0.209 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 10 | 17 | 0.0324 | 0.0845 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 10 | 21 | 0.0348 | 0.0749 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 10 | 22 | 0.0727 | 0.1499 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |
| 21 | 22 | 0.0116 | 0.0236 | 0.00 | 32 | 32 | 32 | 0 | 0 | 1 |
| | -360 | 360; | | | | | | | | |

```
15      23      0.10    0.202   0.00    16      16      16      0       0       1
        -360    360;
22      24      0.115   0.179   0.00    16      16      16      0       0       1
        -360    360;
23      24      0.132   0.27    0.00    16      16      16      0       0       1
        -360    360;
24      25      0.1885  0.3292  0.00    16      16      16      0       0       1
        -360    360;
25      26      0.2544  0.38    0.00    16      16      16      0       0       1
        -360    360;
25      27      0.1093  0.2087  0.00    16      16      16      0       0       1
        -360    360;
28      27      0.00    0.396   0.00    65      65      65      0       0       1
        -360    360;
27      29      0.2198  0.4153  0.00    16      16      16      0       0       1
        -360    360;
27      30      0.3202  0.6027  0.00    16      16      16      0       0       1
        -360    360;
29      30      0.2399  0.4533  0.00    16      16      16      0       0       1
        -360    360;
8       28      0.0636  0.20    0.0428  32      32      32      0       0       1
        -360    360;
6       28      0.0169  0.0599  0.013   32      32      32      0       0       1
        -360    360;
];

%%-----  OPF Data  -----%%
%% generator cost data
%       1       startup shutdown        n       x1      y1      ...     xn      yn
%       2       startup shutdown        n       c(n-1)  ...     c0
mpc.gencost = [
        2       0       0       3       0.0     2.00    0.00375;
        2       0       0       3       0.0     1.75    0.0175;
        2       0       0       3       0.0     3.00    0.025;
        2       0       0       3       0.0     3.25    0.00834;
        2       0       0       3       0.0     3.00    0.025;
        2       0       0       3       0.0     3.00    0.025;
];
```