Proportional-Integral-Derivative and Fractional Order Proportional-Integral-Derivative Controller Design for Load Frequency Control of A Two Area Automatic Generation Control System Tuned by Optimization Algorithms and Artificial Neural Networks

by

Sheikh Samit Muhaimin (170021024)

Md. Nazmush Shakib Shahi (170021058)

Nabil Anan Orka (170021108)

A Thesis Submitted to the Academic Faculty in Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF SCIENCE IN ELECTRICAL AND ELECTRONIC ENGINEERING

Department of Electrical and Electronic Engineering

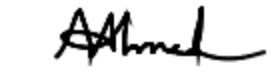
Islamic University of Technology (IUT)

Gazipur, Bangladesh

May 2022

Proportional-Integral-Derivative and Fractional Order Proportional-Integral-Derivative Controller Design for Load Frequency Control of A Two Area Automatic Generation Control System Tuned by Optimization Algorithms and Artificial Neural Networks

Approved by:

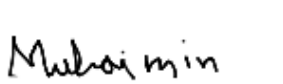


Dr. Ashik Ahmed

Supervisor and Professor,
Department of Electrical and Electronic Engineering,
Islamic University of Technology (IUT),
Boardbazar, Gazipur-1704.

Date: 10. 05. 2022

Signature of the Authors:



Sheikh Samit Muhaimin
ID: 170021024

Md. Nazmush Shakib Shahi
ID: 170021058

Nabil Anan Orka
ID: 170021108

Table of Contents

List of Tables:

List of Figures:

List of Acronyms & Appendix:

LFC = Load Frequency Control

AGC = Automatic Generation Control

PID = Proportional Integral Derivative

FOPID = Fractional Order Proportional Integral Derivative

NN = Neural Network

ANN = Artificial Neural Network

ITAE = Integral of Time multiplied by Absolute Error

GBO = Gradient Based Optimizer

EGBO = Enhanced Gradient Based Optimizer

MPA = Marine Predators Algorithm

MTDE = Multi Trial vector based Differential Evolution

Appendix A.

Nomenclatures used in the studied system

R_1, R_2	Speed regulation constants of the governors of Area 1 and Area 2, respectively
b_1, b_2	Frequency-bias factors of Area 1 and Area 2, respectively
ACE_1, ACE_2	Area Control Errors of Area 1 and Area 2, respectively
u_1, u_2	Outputs of the controllers of Area 1 and Area 2, respectively
T_{sg1}, T_{sg2}	Time-constants for speed governor in seconds of Area 1 and Area 2, respectively
T_{sg}	Time-constant for governor block
x_3, x_6	Per-unit set position changes of governor valves of Area 1 and Area 2, respectively
T_{t1}, T_{t2}	Time-constants for the turbine in seconds of Area 1 and Area 2, respectively
T_t	Time-constant for turbine block
x_2, x_5	Per-unit change in output powers of the turbines of Area 1 and Area 2, respectively
T_{ps1}, T_{ps2}	Time-constants for the generator-load model in seconds of Area 1 and Area 2, respectively
T_{ps}	Time-constant for the generator-load model
K_{ps1}, K_{ps2}	Gains of the generator-load model of Area 1 and Area 2, respectively
K_{ps}	Gain of the generator-load model
T_{12}	Coefficient of synchronization
w_1, w_2	A step-change in load demands of Area 1 and Area 2, respectively
x_7	Tie-line power variation from its targeted value
T_{12}	Coefficient of synchronization in seconds
a_{12}	Constant
x_1, x_4	Frequency deviations from their nominal value of Area 1 and Area 2, respectively

Appendix B.

Nominal values used in the simulations

R_1, R_2	b_1, b_2	T_{sg1}, T_{sg2}	T_{t1}, T_{t2}	T_{ps1}, T_{ps2}	K_{ps1}, K_{ps2}	a_{12}
3	0.425	0.4	0.5	20	100	1

Appendix C.

Various e and f parameters of the studied system:

$$e_1 = 2\pi T_{12} K_{p1} + K_{i1} B_1 - \frac{K_{p1} B_1}{T_{ps1}}$$

$$e_2 = \frac{K_{p1} B_1 K_{ps1}}{T_{ps1}} - \frac{K_{d1} B_1 K_{ps1}}{T_{ps1} T_{t1}} - \frac{K_{d1} B_1 K_{ps1}}{T_{ps1}^{\,2}} + \frac{K_{d1} K_{ps1} 2\pi T_{12}}{T_{ps1}}$$

$$e_3 = \frac{K_{d1} B_1 K_{ps1}}{T_{ps1} T_{t1}}$$

$$e_4 = -K_{p1} 2\pi T_{12} + \frac{K_{d1} B_1 K_{ps1} 2\pi T_{12}}{T_{ps1}} + \frac{K_{d1} 2\pi T_{12}}{T_{ps2}}$$

$$e_5 = -\frac{K_{d1} K_{ps2} 2\pi T_{12}}{T_{ps2}}$$

$$e_6 = 0$$

$$e_7 = K_{i1} - \frac{K_{p1} B_1 K_{ps1}}{T_{ps1}} + \frac{K_{d1} B_1 K_{ps1}}{T_{ps1}^{\,2}} - \frac{K_{d1} K_{ps1} 2\pi T_{12}}{T_{ps1}} - \frac{K_{d1} K_{ps2} 2\pi T_{12} a_{12}}{T_{ps2}}$$

$$e_8 = -\frac{K_{p1} B_1 K_{ps1}}{T_{ps1}} - \frac{K_{d1} 2\pi T_{12} K_{ps1}}{T_{ps1}} + \frac{K_{d1} B_1 K_{ps1}}{T_{ps1}^{\,2}}$$

$$e_9 = \frac{K_{d1} 2\pi T_{12} K_{ps2}}{T_{ps2}}$$

$$f_1 = -K_{p2} 2\pi T_{12} a_{12} + \frac{K_{d2} B_2 K_{ps2} 2\pi T_{12} a_{12}}{T_{ps2}} + \frac{K_{d2} a_{12} 2\pi T_{12}}{T_{ps1}}$$

$$f_2 = -\frac{K_{d2} K_{ps1} 2\pi T_{12} a_{12}}{T_{ps1}}$$

$$f_3 = 0$$

$$f_4 = -\frac{K_{p2} B_2}{T_{ps2}} + K_{p2} 2\pi T_{12} a_{12} + K_{i2} B_2 - \frac{K_{d2} B_2 K_{ps2} 2\pi T_{12} a_{12}}{T_{ps2}} + \frac{K_{d2} B_2}{T_{ps2}^{\,2}} - \frac{K_{d2} 2\pi T_{12} a_{12}}{T_{ps2}}$$

$$f_5 = \frac{K_{p2}B_2K_{ps2}}{T_{ps2}} - \frac{K_{d2}B_2K_{ps2}}{T_{ps2}T_{t2}} - \frac{K_{d2}B_2K_{ps2}}{T_{ps2}^{\ 2}} + \frac{K_{d2}K_{ps2}2\pi T_{12}a_{12}}{T_{ps2}}$$

$$f_6 = \frac{K_{d2}B_2K_{ps2}}{T_{ps2}T_{t2}}$$

$$f_7 = \frac{K_{p2}B_2K_{ps2}a_{12}}{T_{ps2}} - \frac{K_{d2}B_2K_{ps2}a_{12}}{T_{ps2}^{\ 2}} + \frac{K_{d2}K_{ps1}2\pi T_{12}a_{12}}{T_{ps1}} - K_{i2}a_{12} + \frac{K_{d2}K_{ps2}2\pi T_{12}a_{12}^{\ 2}}{T_{ps2}}$$

$$f_8 = \frac{K_{d2}2\pi T_{12}K_{ps1}a_{12}}{T_{ps1}}$$

$$f_9 = -\frac{K_{p2}B_2K_{ps2}}{T_{ps2}} - \frac{K_{d2}2\pi T_{12}K_{ps2}a_{12}}{T_{ps2}} + \frac{K_{d2}B_2K_{ps2}}{T_{ps2}^{\ 2}}$$

Appendix D.

Parametric values used in the optimizers

Optimizer	Values of the control parameters
EGBO (FOPID)	Popsize = 30, MaxIt = 250, dim = 10, x_min=[-18 -35 0.75 -20 0.75 -18 -35 0.75 -18 0.75], x_max=[-6 -12 1 -5 1 -7 -12 1 -5 1]
ChOA	SearchAgents_no = 100, Max_iteration = 500, dim = 6
EGBO	nP = 100, MaxIt = 500, dim = 6, LC = 0.7
GBO	nP = 100, MaxIt = 500, dim = 6, pr = 0.5
GWO	SearchAgents_no = 100, Max_iteration = 500, dim = 6
PSO	noP = 100, maxIter = 500, nVar = 6, wMax = 0.9, wMin = 0.2. c1 = 2, c2 = 2
SCA	SearchAgents_no = 100, Max_iteration = 500, dim = 6, a = 2

Acknowledgements:

First and above all, we convey our heartfelt gratitude to Almighty Allah (SWT), the Most Merciful and Compassionate, for providing us with the health and capacity to carry out this task.

We would like to express our sincere appreciation to Dr. Ashik Ahmed, Professor, Department of Electrical and Electronic Engineering, Islamic University of Technology (IUT), for his unwavering support and direction during this work. Throughout our studies and research, his excellent counsel, support, and guidelines established a research atmosphere in which we were allowed to explore various ideas without restriction. His shared vision and willingness to share his extensive expertise tremendously aided us in conducting this thesis.

We are also extremely grateful to the authors of the citations because of their tremendous research work.

We would also like to extend our thanks to all of the Electrical and Electronic Engineering (EEE) department's faculty members for their cooperation and encouragement throughout the process.

Finally, we thank our parents, family, friends, and well-wishers for their undying love and support.

Abstract:

This paper describes how to tune Proportional Integral Derivative (PID) and Fractional Order PID (FOPID) controllers for Load Frequency Control (LFC) of two area linked thermal power systems using an optimization technique and Neural Network (NN) prediction-based approaches. Generator speed fluctuates due to frequent variations in load demand. As a result, frequency deviates from the specified value. That is why it is critical to employ a correctly tuned controller to change the generator's speed in order to keep the frequency as near to its rated value as feasible. The Enhanced Gradient Based Optimizer (EGBO) is utilized in this study to adjust a PID controller by optimizing an Integral Time multiplied by Absolute Error (ITAE) based fitness function. The EGBO algorithm's results were compared to the Gradient Based Optimizer (GBO), Chimp Optimization Algorithm (ChOA), Sine Cosine Algorithm (SCA), Grey Wolf Optimization (GWO), and Particle Swarm Optimization algorithms (PSO). When compared to alternative optimization approaches, the relevant data reveal that the EGBO algorithm is competitively superior in terms of robustness, accuracy, and latency. The ITAE-based fitness function of the FOPID controller is also optimized using EGBO. In general, the FOPID controller outperforms the PID controller in terms of ITAE since it has a few more configurable parameters. In order to optimize the ITAE, optimization methods require some time. As a result, Artificial Neural Networks (ANNs) are used for tuning with nearly little latency. Datasets are created using the previously described PID controller optimization procedures, which are then utilized to train different hyperparameters.

Chapter 1: Introduction

1.1 General Overview

For a power generation system, the load demand is always changing. As a result, the power generation system is needed to be modified with existing power generation system to fulfill the load demand. But here arises another severe problem: as most of the loads are dynamic in nature, frequent change in electrical power drawn is happening. As a result, speed of the generator is changing. The generator speed changes due to the frequent change in electrical power drawn because, when load increases, the electrical torque becomes higher than the mechanical torque of the generator. As a result, speed of the generator decreases. On the other hand, when load decreases, the electrical torque becomes lower than the mechanical torque of the generator. As a result, speed of the generator increases. Now the relation between generator speed and frequency can be established from the equation below,

$$f_s = N_s * \frac{P}{120} \qquad (1)$$

Where, f_s is the frequency, N_s is the speed of the generator and P is the number of poles of the generator. Basically, in equation (1), P is constant because number of poles is fixed in a generator. So, f_s is directly proportional to N_s. That means, when N_s increases, f_s also increases and vice-versa.

So, in this situation, Automatic Generation Control (AGC) can be used to get rid of this problem. Basically, AGC will control the speed of the generator by using some control mechanism or controller to ensure the frequency value within acceptable limits.

Generally, mechanical torque of the generator is controlled by turbine or engine. The speed of the turbine or engine is basically controlled by fuel injection. And the fuel injection is controlled by the governor. So, actually generator speed is controlled by the governor indirectly. The following figure neatly shows these relations:

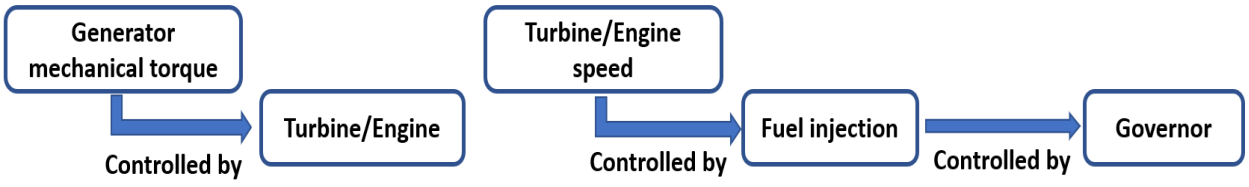


Fig. 1: Generator-governor relationship

1.1.1 System Model:

In this work, a two area AGC is considered for an interconnected non-reheat thermal power system. The system model is as follows:

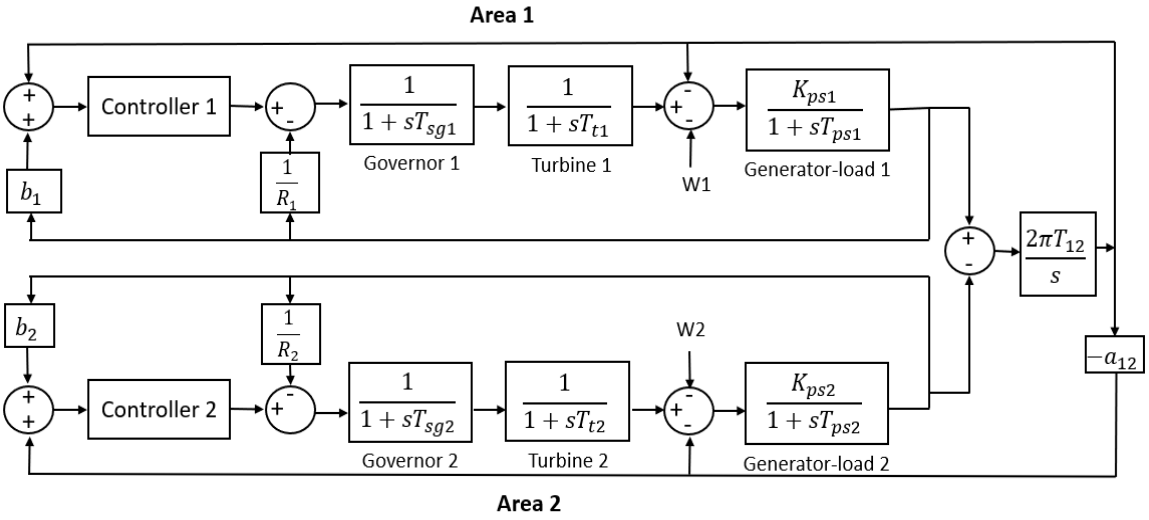


Fig. 2: Two area thermal power system model

Both areas in the AGC system are equipped with identical non-reheat thermal power stations. The in-depth discussion for different components of the adopted thermal power station are displayed in the following subsections.

I. Governor Model:

When the generator electrical load is suddenly increased, the electrical power exceeds the mechanical power input [1]. This power deficiency is supplied by the kinetic energy stored in the rotating system [1]. The reduction in kinetic energy causes the turbine speed and, consequently, the generator frequency to fall [1]. The change in speed is sensed by the turbine governor which acts to adjust the turbine input valve to change the mechanical power output to bring the speed to a new steady-state [1]. The earliest governors were the Watt governors which sense the speed by means of rotating flyballs and provides mechanical motion in response to speed changes [1]. However, most modern governors use electronic means to sense speed changes [1]. Operating a steam turbine with an outdated or obsolete governor can result in reduced efficiency and lost profits [2]. Mechanical systems are sensitive to ambient conditions, have components that wear, and can experience fatigue and failure, increasing downtime and maintenance costs [2]. An electronic governor is virtually maintenance free, provides more precise speed control, and has fewer wearing parts than

mechanical systems [2]. So, governor is responsible for operating valves to control steam flow into the turbine according to the feedback and controller output of the system. The transfer function of the governor is as follows [3]:

$$G_{sg}(s) = \frac{1}{1 + sT_{sg}} \qquad (2)$$

II. Turbine Model:

Turbine is basically working as a prime mover in this model. The source of mechanical power, commonly known as the prime mover, may be hydraulic turbine at waterfalls, steam turbines whose energy comes from the burning of coal, gas, nuclear fuel and gas turbines [1]. The steam turbine is driven by highly pressurized steam which is responsible for driving the generator shaft. The first-order transfer function of steam turbine is as follows [3]:

$$G_{t}(s) = \frac{1}{1 + sT_{t}} \qquad (3)$$

III. Generator-Load Model:

The load on a power system consists of a variety of electrical devices. For resistive loads, electrical power is independent of frequency [1]. However, for inductive load such as motor is very sensitive to frequency as the speed of the motor is directly proportional to the frequency. When the steam turbine rotates, the shaft of the generator also rotates, resulting in the generator output. The simplified transfer function of the generator-load model is as follows [3]:

$$G_{ps}(s) = \frac{K_{ps}}{1 + sT_{ps}} \qquad (4)$$

IV. Tie-Line:

Tie-lines are the transmission links that connect one area to its neighboring regions. As the name implies, LFC manages power flow between different interconnected areas while maintaining a constant frequency. To achieve 0% steady-state error in tie-line power flow, an integral controller is used in this work [3].

Detail of the nomenclatures used in the system under study can be found in appendix A. The nominal parametric values of the system, which are selected for simulations, are provided in appendix B.

1.1.2 Controller overview:

When load fluctuates, the mechanical input to the generator that means the steam turbine speed is controlled by the governor accordingly. But this process cannot be 100% accurate as the mechanical system cannot be adjusted instantaneously. So, some errors are generated in this process. That's why controller is used to minimize this error. In this work two types of controllers are used:

a. Proportional Integral Derivative (PID) controller
b. Fractional Order PID (FOPID) controller [4] [5]

a. Proportional Integral Derivative (PID) controller:

A Proportional Integral Derivative (PID) controller is a control loop mechanism employing feedback that is widely used in industrial control systems and a variety of other applications requiring continuously modulated control [6]. A PID controller continuously calculates an error value, e(t) as the difference between a desired Set Point (SP) and a measured Process Value (PV) and applies a correction based on proportional, integral and derivative terms (denoted P, I, and D respectively), hence the name [6].

Actually, Proportional controller is responsible for decreasing Rise time and Steady-state error. On the other hand, Integral controller is responsible for eliminating the Steady-state error. However, due to both Proportional and Integral controller, Overshoot increases. As a result, Settling time also increases. That's why Derivative controller is used in order to reduce the Overshoot and Settling time.

There are some terms of PID controller that are defined below:

- **Set Point (SP):**
  Set Point (SP) is normally a user entered value [7]. In this work, it is the rated steam turbine speed which is basically responsible for rotating the generator shaft to ultimately control the frequency.

- **Process Value (PV):**
  The Process Value (PV) is the value that is being actually controlled [7]. In this work, the actual steam turbine speed that is practically measured is the PV.

- **Output:**
  Output is the controlled value of the PID controller [7]. In case of load frequency control, output is the signal to the governor of how much steam should be fed into the steam turbine by controlling the valve.

- **Error (e(t)):**
  The error (e(t)) is the value used by the PID controller in order to determine how to adjust the Output to bring the PV to SP.

$$e(t) = SP - PV \qquad (5)$$

The gains of the PID controller are explained below:

The gains of the PID controller are nothing but user input values which are P-gain, I-gain and D-gain that are denoted as K_p , K_i and K_d , respectively. Actually, gain is the term used for "multiplication factor". By adjusting the gain settings (or multiplication factor) of the proportional, the integral and the derivative, the user can control how much effect the PID controller has on the output, and how the controller will react to the changes in the PV [7].

Now the PID controller is explained below in details:

- **Proportional (P):**

  The Proportional (P) is calculated by multiplying the P-gain by the error. The purpose of the proportional is to have a large immediate reaction on the output to bring the process value close to the set point. As the error becomes less, the influence of the proportional value on the output becomes less [7].

  The following equation denotes the P controller:

$$P = K_p * e(t) \qquad (6)$$

❖ **Integral (I):**

The Integral is calculated by multiplying the I-gain by the error, then multiplying this by the cycle time of the controller (how often the controller performs the PID calculation) and continuously accumulating this value as the "total integral" [7].

Every time the controller performs the PID calculation, the new calculated integral value is added to the total integral [7]. The integral will normally not have as much immediate influence on the output as the proportional [7]. But as the integral is continuously accumulating over time, the longer it takes for the PV to reach the SP, the more effective the integral becomes over the output [7].

The following equations denote the I controller:

$$I = K_i * e(t) * dt \qquad (7)$$

$$I_t = I_t + I \qquad (8)$$

Where, dt = Controller cycle time or loop time, I_t = Total integral.

❖ **Derivative (D):**

The derivative is calculated by multiplying the D-gain by the ramp rate of the PV [7]. The purpose of the derivative is to "predict" where the PV is going and bias the output in the opposite direction of the proportional and integral controller in order to hopefully prevent the controller from overshooting the SP if the ramp rate is too fast [7].

Explained a bit simpler, if the PV is approaching the SP too fast, the derivative will limit the output to prevent the PV from overshooting the SP [7].

The following equation denotes the D controller:

$$D = K_d * \{| e(t) - e_p(t) |\} / dt \qquad (9)$$

Where, $e_p(t)$ = Previous error.

❖ **Output:**

Actually, summing up the P, I and D controller outputs together give the overall PID controller output value. The gain of each of the controller sets up how much authority that particular controller has over the output.

So, PID controller output = P + I_t + D (10)

Considering equation (6-9), PID controller output can be written as,

Output = K_p * e(t) + I_t + K_i * e(t) * dt + K_d * {| e(t) - $e_p(t)$ |} / dt (11)

b. Fractional Order PID (FOPID) controller [6][7]:

Fractional Order PID (FOPID) controller [4] [5] is exactly similar to the PID controller in terms of K_p, K_i and K_d gains where the individual controller works as similar fashion as the PID controller. The only difference is unlike PID controller, the FOPID controller [6][7] has couple of extra tunable parameters which are the order of the integral and derivative controller transfer function denoted as λ and μ, respectively. Basically, the PID controller has integer order transfer function that means the power of s is always 1 in Laplace domain. On the other hand, the FOPID controller [4] [5] has the power of s equal to -λ for integral controller, μ for derivative controller in Laplace domain and these λ and μ values are fractional. The values of λ and μ always remain in between 0 and 1. That's why it is called Fractional Order PID controller.

So, FOPID controller [4] [5] has total 5 tunable parameters. Because of these extra two tunable parameters, FOPID controller has much higher flexibility compared to PID controller for controlling a device precisely.

The PID controller is very much familiar in industries and any other control applications because it is quite simple, it is widely available and it is quite easy to implement in a system. On the other hand, FOPID controller [4] [5] is not that much popular like PID controller in industrial application. Because it is more complex in structure and it uses higher order integer approximation in order to be operational. However, in case of tunable parameters, PID controller has lesser flexibility and

robustness compared to the FOPID controller [4] [5]. So, FOPID controller [4] [5] is capable of giving far better result than the PID controller if it is tuned properly.

Transfer function of FOPID controller is as follows [4] [5]:

$$C(s) = K_p s + K_i s^{-\lambda} + K_d s^{\mu} \quad (12)$$

Or,

$$C(s) = K_p \left(1 + \frac{1}{T_i s^{\lambda}} + T_d s^{\mu}\right) \quad (13)$$

Where,

$$T_i = \frac{K_p}{K_i} \quad (14)$$

$$T_d = \frac{K_d}{K_p} \quad (15)$$

Although the order of integral and derivative controller is fractional, the transfer function is converted into equivalent higher order (integer) transfer function for ease of calculation. The converted higher order transfer function is basically an approximated integer order transfer function which is equivalent to the main fractional order transfer function. Also, computer program such as Matlab [8] cannot work with the fractional order transfer function. Actually, Matlab [8] needs the approximated integer order transfer function of the fractional order transfer function in order to execute any program. That's why accurate approximation is important in order to get accurate result.

Actually, to get the approximated integer order transfer function, frequency range (ω_l and ω_h) and approximation order (N) are needed to be calculated. Then Oustaloup approximation [4] [9] method can be applied using Fomcon Toolbox [4] in order to get the approximated integer order transfer function. However, FOPID block [4] in Simulink [8] by default approximates the integer order transfer function of the fractional order transfer function when the frequency range and the approximation order is provided.

In [10], the rules for calculating the frequency range and the approximation order are explained quite nicely. The formulas for calculating frequency range and the approximation order are as follows,

$$\omega_l <= \left(\left|\frac{K_p P(0)}{T_i}\right| \epsilon\right)^{\frac{1}{\lambda}} \qquad (16)$$

$$\omega_h >= 10^3 \omega_b \qquad (17)$$

$$N \approx \frac{(1-v)\log\left(\frac{\omega_h}{\omega_l}\right)}{\log\left(\frac{1+cos(0.5\pi v)}{1-cos(0.5\pi v)}\right)} \qquad (18)$$

Where,

P(0) = Plant model at s = 0, K_p = Proportional gain, ϵ = upper bound of the steady-state value of the simulation error, ω_b = Bandwidth of the feedback system, v = Values of λ and μ .

1.1.3 Tuning of the controllers:

Controllers will be completely useless if it is not tuned properly. There are a lot of ways to tune the controllers. The controllers can be tuned manually in trial-and-error method. But it is not easy to tune the controllers in this method manually and it is very much time consuming considering the complex system in this work. Basically, in this work, the following two techniques are adopted in order to tune the controllers:

- **i.** Optimization algorithms
- **ii.** Neural Network prediction-based approaches

i. Optimization algorithm:

Optimization algorithms are mainly designed to optimize a specific objective or fitness function by setting up the system parameters within the given range. The optimization of objective function can be minimization or maximization based; it depends on the system requirements.

Optimization algorithms are really helpful in solving real world problems in real time within a short amount of time where solving those problems manually is an extremely time consuming and difficult task. To deal with the real-world problems with optimization algorithm, the whole system is needed to be converted into mathematical model. Then the optimization algorithm does the analysis using that

mathematical model to find out the optimum solution for the problem. Optimization algorithms are generally divided into two categories which are Deterministic algorithms and Stochastic algorithms [11]. Deterministic algorithms are sub-divided into Linear programming, Non-linear programming, Gradient based and Gradient free algorithms. On the other hand, Stochastic algorithms are categorized into two parts: Heuristic and Meta-heuristic algorithms. Where, Meta-heuristic algorithms are sub categorized into Population based and Trajectory based algorithms. The following figure shows the classifications of optimization algorithms [11]:

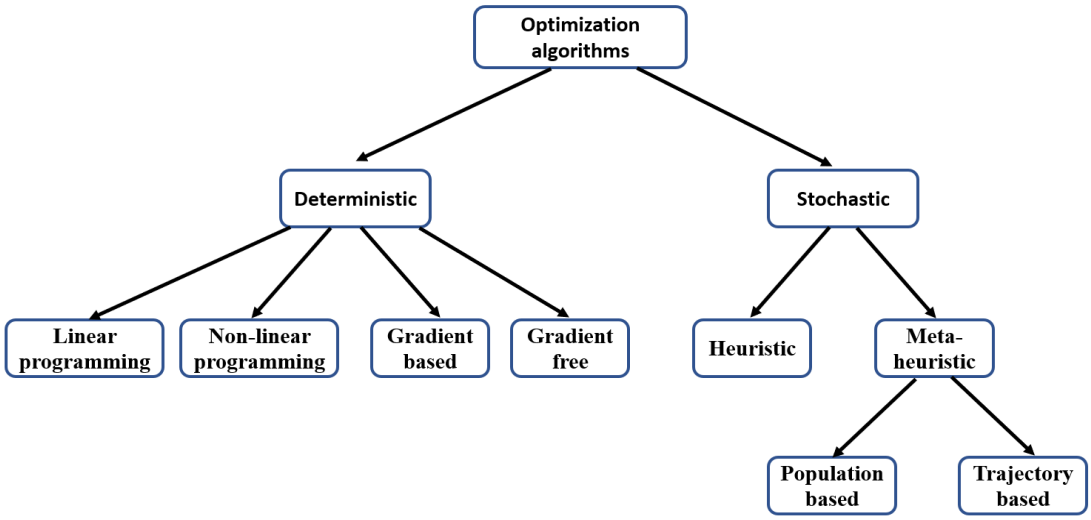


Fig. 3: Classification of optimization algorithm [11]

A wide variety of objective functions are available to be set in an optimization algorithm. the mostly used time domain objective functions are: Integral of Time multiplied by Squared Error (ITSE) [12], Integral of Squared Error (ISE) [12], Integral of Absolute Error (IAE) [12], Integral of Time multiplied by Absolute Error (ITAE) [12] [13]. In this work, ITAE [12] [13] based objective function is chosen for optimization.

For tuning PID controller [13] [14] [15], the following optimization algorithms are used in this work:

- Particle Swarm Optimization (PSO) [14]
- Sine Cosine Algorithm (SCA) [16]
- Grey Wolf Optimizer (GWO) [17] [12]
- Chimp Optimization Algorithm (ChOA) [18]

- Gradient Based Optimizer (GBO) [19]
- Enhanced Gradient Based Optimizer (EGBO) [20]
- Marine Predators Algorithm (MPA) [21]
- Multi Trial vector based Differential Evolution (MTDE) [22]

In this work, GBO [19], EGBO [20], MPA [21] and MTDE [22] are used for generating datasets which are used to train the Neural Network. Some novel algorithms are also investigated by hybridizing two algorithms together. Those are following:

- Enhanced Gradient Based Golden Eagle Optimizer (EGBGEO)
- Gradient Based Golden Eagle Optimizer (GBGEO)
- Gradient Based Sine Cosine Algorithm (GBSCA)

As the novel algorithms could not yield convincing results, the endeavor for hybridizing newer algorithms is not further explored.

For tuning FOPID controller [23], the following algorithms are tried so far:

- Particle Swarm Optimization (PSO) [14]
- Enhanced Gradient Based Optimizer (EGBO) [20]

ii. Neural Network prediction-based approach:

Neural Networks (NN) are neuronal networks that might be biological or artificial [24]. In reality, a neural network is a set of algorithms that attempts to detect underlying links in a piece of data using a technique that is similar to how the human brain works [24].

The human brain is made up of many linked neurons. Neurons are the basic and primary components of the human brain and nervous system. Neurons are in charge of receiving external inputs, which are nothing more than data from human body sensor organs, and then commanding output to the motor nerves based on that data. The motor nerves are in charge of moving any portion of the human body. Artificial Neural Network (ANN) is the foundation of Artificial Intelligence and is inspired by this extraordinary notion of human brain neural network (AI).

Computers or machines do not understand physical things and lack the ability to recognize objects or foresee events in the same way that humans do. Those gadgets only recognize discrete data. That is why those gadgets require some sort of algorithm in their software to run the system in the manner of a human brain. In this example, an ANN duplicates the features of human brain neural networks, giving robots the capacity to recognize and anticipate things.

The base form of ANN consists of 3 layers:

- Input layer
- Hidden layer
- Output layer

In fig. 4, the basic ANN structure for a Multiple Input and Multiple Output (MIMO) system is displayed. Here, x1, x2 and x3 are the 3 inputs which can be compared with the sensor input of the human body. The weights of the inputs are represented by w1, w2 and w3. The hidden layer is basically the neuron which has an activation function. Basically, the neuron processes the inputs based on their weights and afterwards generates outputs. The outputs are y1, y2 and y3.

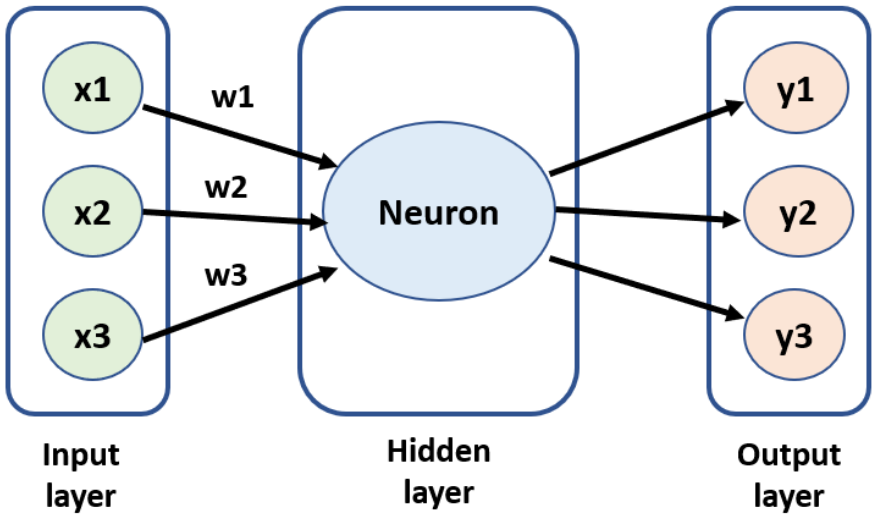


Fig. 4: Basic ANN diagram

The activation function can be of different types according to the application. The most popular functions are:

- **Sigmoid function:** It is basically used for decision making application. The output of this function is either 0 or 1.

- **Linear Unit (LU):**

  - **Rectified LU (ReLU):** It is used to solve forecasting problem. That means for predicting something, this function can be used.

  - **Leaky ReLU (LReLU):** It is used when a lot of negative activation values are there or when ReLU stops learning. As ReLU only considers values greater than 0, for a lot of negative activation values, ReLU starts to stop learning which is known as "Dying ReLU" problem. To solve this problem LReLU is used.

  - **Exponential LU (ELU):** ELUs, like ReLUs and LReLUs, reduce the vanishing gradient effect. Because the positive portion of these functions is the identity, their derivative is one and not contractive, which solves the vanishing gradient problem. ELUs have negative values, bringing the activation mean closer to zero. Faster learning is enabled by mean activations that are closer to zero because they bring the gradient closer to the natural gradient [25].

The classifications of NNs are as follows:

- Feedforward NN or Multi-Layer Perceptron (MLP)
- Convolutional NN (CNN)
- Recurrent NN (RNN)
- Deep NN

Basic ANN is a feedforward NN. MLPs are basically consists of multiple layers such as input layer, output layer and hidden layer. There can be multiple hidden layers. NN with more than one hidden layer is known as Deep NN. Fig. 5 is an example of Deep NN. On the other

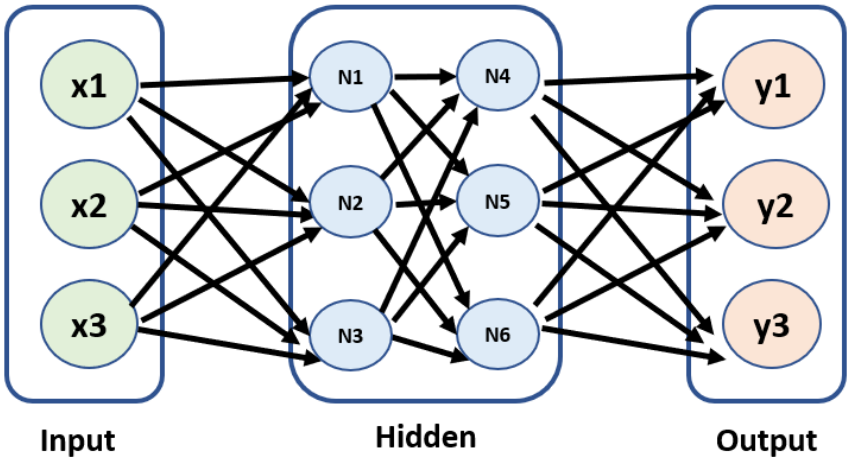


Fig. 5: ANN with multiple hidden layers

hand, CNN is basically used in computer vision and object recognition-based image processing. RNN is very handy in modeling data sequence. Again, RNN has feedback loops which can produce predictive results in sequential data.

In this work, mostly ReLU based ANN algorithm is used in order to predict PID parameters.

ANNs are known for their flexibility to changes in parameter values. Furthermore, their most significant trait is their ability to learn and improve their accuracy over time. The application of ANN controllers in this study is motivated by their short calculation time, ease of implementation, and ability to be used under a variety of Step-Load Perturbation (SLP) circumstances. ANNs have been widely used solving nonlinearities and uncertainties in complex systems as universal approximators [14].

1.2 Literature Survey:

1.2.1 PID Controller Incorporated AGC:

Load Frequency Control (LFC) is the frequency control mechanism of an AGC system, integrating feedback loops of several systems connected through a tie-line. For a stable system response, incorporation of an efficient controller is vital considering the sensitivity and flexibility of the systems. Till date, Proportional Integral Derivative (PID) controllers are one of the vastly adopted techniques due to its simplicity of operation and cost-effectiveness. However, tuning a PID controller manually or in a trial & error method is a difficult and meticulous task. Optimization algorithms are ideal for such cases as they search for optimal parametric values for a certain objective function within specified system constraints.

In [26], a self-tuning Particle Swarm Optimization (PSO) based PID controller for a two-area system is proposed and compared with controllers tuned with the Ziegler-Nichols technique. In [14], a neural network approach has been taken for the LFC problem using PSO and Artificial Neural Network (ANN).

A detailed study of LFC of 4 different test systems is presented in [17], employing Grey Wolf Optimization (GWO), Comprehensive Learning Particle Swarm Optimization (CLPSO), and Ensemble of Mutation and Crossover Strategies and Parameters in Differential Evolution (EPSDE) algorithms in both Proportional-Integral (PI) and PID

controllers, while observing the transients of the frequency fluctuations and keeping track of the ITAE values, settling times and overshoots.

In [26], Sine Cosine Algorithm (SCA) and Non-dominated Sorting Genetic Algorithm (NSGA II) are used for the tuning of a PID with a derivative filter gain (PIDN) controller. Elephant Herding Optimization (EHO), Teacher-Learner-Based Optimization (TLBO), and Whale Optimization Algorithm (WOA) are utilized for a similar type of PID controller in [27]. In [28], [29] respectively, Seeker Optimization Algorithm (SOA) and Quasi-Oppositional Dragonfly Algorithm (QODA) based PID controllers are used for solving the LFC problem of two area systems.

In [14] and [28] ITAE values are not explicitly presented. In [28], [29], [30], [27], PID gains of only one area are optimized by the proposed controllers. In [27], [28], [29], [30] and [17], Step Load Perturbation (SLP) of only one area at a time are studied, ranging from 1 to 10 percent.

In this work, an ITAE based objective function is adopted to minimize the Area Control Error (ACE) of a two-area all thermal system. The overall system model is developed using the state –space approach. Later, generic PID controller blocks are incorporated to enhance the system time-domain characteristics, such as, steady-state error settling time and overshoot.

Different step load perturbations (SLP) ranging from 10% to 25% is applied to both areas as well as separate individual areas to study the robustness and flexibility of the controller. The resulting frequency deviation (FD) and tie-line power deviation (TPD) are tracked to formulate the ITAE based objective function.

For the optimization of the generic PID controller, an Enhanced Gradient-Based Optimizer (EGBO) is adopted in this work, which utilizes Newton's method to search the solution space [16]. It is an update of the Gradient-Based Optimizer (GBO) [19], incorporating new operators and new updating mechanisms [20]. To the best of the authors' knowledge, the usage of EGBO in the LFC problem of multi-area systems is still unexplored. For a comprehensive analysis, widely applied nature-inspired algorithms such as GWO and PSO tested on [17] and [26] respectively, a population-based SCA tested on [30] are used, along with recently developed meta-heuristic Chimp Optimization Algorithm (ChOA) and GBO.

1.2.2 FOPID Controller Incorporated AGC:

According to logical study, FOPID controller should give better result as it has two extra tunable parameters compared to PID controller. A comparative study is done in [51] among different controllers for the same system. It is clearly visible in [31] that FOPID controller gives quite better result than PID controller. That's why it is decided to implement FOPID controller in this work so that more convincing result can be obtained.

Basically, in [31], a two area interconnected Hybrid Power System (HPS) is taken under study where the HPS consists of Renewable Energy (RE) such as Solar and Wind based power generation units, Plug-in Electric Vehicle (PEV) along with thermal power station.

Actually, the large-scale penetration of intermittent RE sources such as wind and solar power generation may cause a problem of frequency aberration of HPS [31]. This occurs when the load frequency control of interconnected system is unable to compensate the power balance between generation and load demand [31]. Also owing to the enhancement of future transport, the PEV plays a significant role to customer at demand side [31]. Thus, the PEV can act as a power control to compensate the power balance in RE integrated power system [31].

Actually, in [31], FOPID controller is used in order to deal with the error. To tune this controller, Atom Search Optimization (ASO) algorithm is implemented. ITAE is used as objective function in order to find out the optimum gains and other parameters of the controller.

The model in [31] is tested for different scenario such including the PEV and excluding the PEV. When the results for those two scenarios are compared, it is clearly noticed that including PEV gives better ITAE. It's because, PEV responds much faster than the governor-generator system of the thermal power system and due to this, less error is generated.

The above-mentioned testing is done for 2.5% Step Load Perturbation (SLP) in area 1 and 1.5% SLP in area 2. The result for including PEV is better and the result for excluding PEV is worse compared to the result of this work. Including PEV scenario gives better result compared to the model of this work because less error is generated due to the incorporation of PEV. However, in both scenarios, the result should be better than the result of this work as the model of this work is simulated with 10% SLPs in both areas which are much higher than the above mentioned SLPs. And it is quite natural that lesser ITAE should be got due to lesser SLPs.

In [32], a three area interconnected power system is studied which is FOPID based AGC and Genetic Algorithm (GA) is used in order to tune the FOPID controller. Basically, among the three area, area 1 is non reheat thermal power system, area 2 is reheat thermal power system and area 3 is hydro power generation system.

Actually, in [32], an extensive analysis on the frequencies of different areas is highlighted. Also, those results are compared with the results of other optimizer (Imperialist Competitive Algorithm (ICA) in this case) and other controller (PID controller in this case) in order to establish superiority of their work.

However, in [32], nothing is mentioned clearly regarding which objective function is used and how much disturbance or SLP is added in each area to get the results. Actually, it is quite unprofessional not to mention the objective function in this kind of optimization-based work. Because objective function is the prime parameter in case of optimizing anything.

In, a hybrid controller design is proposed which is nothing but Fuzzy FOPID controller for both single and two area non reheat thermal power system. Basically, Fuzzy FOPID controller is the integration of Fuzzy controller and FOPID controller where the output of the Fuzzy controller is fed into as the input of the FOPID controller.

ITAE is used as objective function in [33] for optimization. However, nothing is clearly mentioned regarding the optimization algorithm. In order to confirm the effectiveness of the proposed control design approach in [33], numerous simulation tests are performed on the single-area power system [33].

It is mentioned in [33] that the obtained results reveal the superiority of the suggested controller as compared to the recently developed controllers with regard to time response specifications and quantifiable indicators. Additionally, the potential of the suggested controller is also observed by improving the load disturbance rejections under plant parametric uncertainty [33].

However, the optimized result of [33] is not even close to the result of this work in terms of ITAE. In [33], 1% SLP is used where in this work 10% SLP is used in both areas. So, considering the ITAE according to SLP, the result of [33] can not be said superior compared to this work.

In [34], a two area AGC is studied which is equipped with FOPID controller in both areas. Here, two different chaotic maps along with the Uniform Random Number

Generator (RNG) are manifested in the popular Multi Objective Optimization (MOO) algorithm - the Non-dominated Sorting Genetic Algorithm-II (NSGA-II).

Actually, in [34], two FOPID controllers are designed for load frequency control (LFC) of two interconnected power systems. Conflicting time domain design objectives are considered in a MOO based design framework to design the gains and the fractional differential-integral orders of the FOPID controllers in the two area AGC [34]. Different measures of quality for MOO e.g., hypervolume indicator, moment of inertia-based diversity metric, total Pareto spread, spacing metric are adopted to select the best set of controller parameters from multiple runs of all the NSGA-II variants (i.e., nominal and chaotic versions) [34]. The chaotic versions of the NSGA-II algorithm are compared with the standard NSGA-II in terms of solution quality and computational time [34]. In addition, the Pareto optimal fronts showing the trade-off between the two conflicting time domain design objectives are compared to show the advantage of using the FOPID controller over that with simple PID controller [34]. The nature of fast/slow and high/low noise amplification effects of the FOPID structure or the four-quadrant operation in the two inter-connected areas of the power system is also explored [34]. A fuzzy logic-based method has been adopted next to select the best compromise solution from the best Pareto fronts corresponding to each MOO comparison criteria [54].

However, the system design in [34] looks extremely complex and nothing is mentioned regarding the execution time of the system program. Again there is no information of how much SLPs are considered for recording data.

1.2.3 Neural Network Approach for AGC with PID Controller:

Researchers have developed a variety of control strategies for load frequency control (LFC) like classical control strategies where Bode and Nyquist diagrams as well as root locus are used. These techniques show poor dynamic performance, especially in the presence of other destabilizing influences such as parameter fluctuations [35] and nonlinearities [36]. In [37], LFC techniques based on modern optimal control theory have been used.

More advanced techniques were later developed such as variable structure load frequency controllers (VSLFC) [38], combined adaptive control and fuzzy control [39], [40], control principles employing neural networks [41]. In [42], The proportional integral (PI) LFC was created in order to improve frequency management. To enable robust frequency control, the proportional integral differentiation (PID) approach for

LFC is proposed in [43]. In [44], robust control techniques were implemented despite the fact that there are transmission delays. The objectives in the robust control techniques are to develop load frequency controllers that not only satisfy nominal stability and performance requirements, but also ensure robust stability and performance [45]. The linear matrix inequality approach is investigated in [46], in order to fix the characteristic of LFC under various uncertainties and specific time-varying conditions. The use of artificial neural networks (ANNs) was presented in [47] to solve the problem of human inability to govern complex plants.

The use of a Genetic algorithm (GA) to tune the gain of controllers ushered in a new age, which was then extended to a proportional integral controller in an LFC scenario. Particle Swarm Optimization (PSO) [26], Grey Wolf Optimization (GWO) [17], Sine Cosine Algorithm (SCA) [30], Elephant Herding Optimization (EHO) [27], Seeker Optimization Algorithm (SOA) [28], and Quasi-Oppositional Dragonfly Algorithm (QODA) [29] are among the many other metaheuristic algorithms used in the field of power system LFC. PSO and Artificial Neural Network (ANN) were combinedly used in [14] to solve the LFC problem.

In the recent past, new metaheuristic optimization algorithms have been reported, including Newton's method inspired Gradient-Based Optimizer (GBO) [19] and GBO's updated algorithm Enhanced Gradient-Based Optimizer (EGBO) [20], adaptive movement step design inspired Multi-Trial Vector-Based Differential Evolution (MTDE) [22], and nature-based Marine Predators Algorithm (MPA) [21].

The adaptive PI controller provided in [48], [49], [50], [51] is designed utilizing an artificial intelligent method of fuzzy and neural based control system using area-control error and its change. In [52], evolutionary methods like as the genetic algorithm were also utilized to develop and improve the controller's performance. However, because these strategies are fundamentally stochastic procedures, it is difficult to ensure their success in a theoretical sense. The study in [53] built and analysed a novel adaptive feed forward neural network-based PI–derivative (PID) controller for a multi-area linked power system with a hybrid energy storage system. The construction of a non-linear neural network controller utilizing a Generalized Neural Network (GNN) was demonstrated in [54] to solve several problems of Simple neural networks, such as long training times, the need for a high number of neurons, and so on. The GNN controller has also been proven to be particularly effective at controlling plant dynamics in a short amount of time. In [55], The artificial neural network technique is used to control a three-area interconnected power system with tie lines connecting them, and the transient behaviour of each area's frequency and tie-line power deviations in the three-area power system showed that the ANN controller outperformed the conventional integral controller.

The current work presents a comparative study of implementation of different optimization algorithms in the ANN based LFC problem. Basically here, ANN is trained according to the datasets generated using different optimizers for PID controller based AGC. ANN learns from the trends of the datasets and apply its learning capabilities to predict the optimum PID parameters.

1.2.4 Outcomes from Literature Survey:

From the above discussions, it is quite obvious that PID controller is the most tested controller in the field of LFC because of its simplicity and availability. On the other hand, FOPID controller is getting famous nowadays because of its couple of extra tunable parameters. And ANN based prediction approach is not that much common in this LFC field.

Basically, after getting quite satisfactory result from PID controller based AGC model, expecting more satisfactory result is a natural thing. According to [31], FOPID controller is capable of producing much better result than PID controller for the same model. That's why FOPID controller is incorporated into the system. These controllers are not the fastest thing to control the AGC, specially FOPID controller-based system takes a lot of time to find out the optimum tunable parameters. In case of reducing execution time model predictive methods are the most feasible solution [34]. That's why ANN based predictive model is developed in order to reduce the execution time.

According to the aforementioned literature review, several optimization techniques with different goal functions may be employed to tune PID controller based AGC. ITAE, on the other hand, is the most commonly utilized time domain objective function. As a result, ITAE is employed as the objective function in this study for all three scenarios. GBO and EGBO are primarily employed in this study to optimize ITAE-based goal functions in PID controller-based AGC. Actually, those two algorithms are quite new, and they are mostly unexplored in the field of LFC. The MPA and MTDE methods are also utilized to create datasets for the NN model in the same system.

Also, for the FOPID controller based LFC field, EGBO is completely unexplored. That's why this optimization algorithm is used in the FOPID based AGC system in order to optimize the ITAE based objective function.

Although the goal of this project is to reduce ITAE, the execution time is critical in the event of a sensitive system. As a result, this work uses an ANN-based predictive strategy to minimize execution time. In fact, RNN is commonly employed in data prediction. However, there is no convincing proof that RNN can be used in the LFC industry and there is no time-series element on the studied datasets. On the other hand, the literature review shows that ANN-based hybrid algorithms, such as PSO plus ANN, are applied in the LFC sector in some cases. That is why ANN has been chosen for this project.

1.3 Objective of this Work:

From the previous sections and sub-sections, it is quite obvious that: the main objective of this work is to maintain a consistent frequency in the two area non-reheat thermal power system. Consistent frequency is important for a power generation system because most of the household and industrial connected loads are inductive in nature and the performance of these inductive loads directly depends on the frequency.

In order to achieve consistent frequency, the proper controller design is very important which helps to reduce or eliminate the errors. Again, controllers without proper tune is completely useless. That's why appropriate optimization algorithms with appropriate objective function are needed to be chosen in order to tune the controllers.

Chapter 2: Methodology

2.1 Overview of Different Optimization Algorithms:

The following subsections are dedicated for extensive overview of some of the optimization algorithms that are used in this work.

2.1.1 Gradient Based Optimizer (GBO):

GBO is a novel meta-heuristic optimization algorithm [19] that searches the solution space using Newton's method logic [20]. GBO operates with the assistance of two preeminent operators, namely Gradient Search Rule (GSR) and Local Escaping Operator (LEO).

The gradient search rule regulates the movement of vectors to better search in the feasible domain and get better locations using the following equations:

$$GSR = randn * \frac{2\Delta x * x_i}{(y1_i - y2_i + \epsilon)} \qquad (19)$$

Where,

$$y1_i = rand * \left(\frac{u_{i+1} + x_i}{2} + rand * \Delta x\right) \qquad (20)$$

$$y2_i = rand * \left(\frac{u_{i+1} + x_i}{2} + rand * \Delta x\right) \qquad (21)$$

$$u_{i+1} = x_i - randn * \frac{2\Delta x * x_i}{x_{worst} - x_{best} + \varepsilon} + DM \qquad (22)$$

Here, u_{i+1} is the new GBO solution, $randn$ is a random integer with a normal distribution, x_{worst} and x_{best} are the worst and best solutions acquired during the optimization process and ε represents a small-scale number in the range [0, 0.1], which helps the equations to escape from getting zero values in the denominators.

A Direction Movement is denoted by DM, which is written as,

$$DM = rand * F_2 * \left(x_{best} - x_i^{it}\right) \qquad (23)$$

Here, F_2 is an adaptive parameter and it is the iteration number.

Δx is defined as the difference between two vectors, which is formulated as,

$$\Delta x = rand * |\chi| \quad (24)$$

$$\chi = \frac{x_{best} - x_i^{it} + \eta}{2} \quad (25)$$

$$\eta = 2 * rand * \left(\left| \frac{x_{a1}^{it} + x_{a2}^{it} + x_{a3}^{it} + x_{a4}^{it}}{4} - x_i^{it} \right| \right) \quad (26)$$

Here, a1, a2, a3, and a4 denote four random numbers in the range of [1, N], in which, N is the pre-defined population number.

In each iteration of the optimization process, GBO uses the GSR and DM to generate a new solution. As a result, the new solution may be computed as follows:

$$X_{new_1 i}^{it} = x_i^{it} - GSR + DM \quad (27)$$

$$X_{new_1 i}^{it} = x_i^{it} - randn * F_1 * \frac{2\Delta x * x_i}{(y1_i - y2_i + \epsilon)} + DM \quad (28)$$

To improve its local search, GBO generates another new solution vector using the following relation,

$$X_{new_2 i}^{it} = x_{best} - randn * F_1 * \frac{2\Delta x * x_m^{it}}{(y1_i^{it} - y2_i^{it} + \epsilon)} + rand * F_2 * \left(x_{a1}^{it} - x_{a2}^{it} \right) \quad (29)$$

F_1 and F_2 are defined as adaptive scale factors which are used to strike a balance between exploration and exploitation, as well as to discover auspicious regions in the search space.

Combining these two new solution vectors of Eq. (28-29), the solution vector for a certain iteration is obtained as,

$$x_i^{it+1} = r_1 * \left(r_2 * X_{new_1 i}^{it} + (1 - r_2) * X_{new_2 i}^{it} \right) + (1 - r_1) * X_{new_3 i}^{it} \quad (30)$$

Here, r_1 and r_2 denote integers that are randomly chosen from the range [0, 1] and,

$$X_{new_3 i}^{it} = x_i^{it} - F_1 * \left(X_{new_1 i}^{it} - X_{new_2 i}^{it} \right) \quad (31)$$

In the GBO, LEO is employed to boost the capacity to escape local solutions. A suitable solution can be constructed with the help of the following pseudo-code,

```
if rand < 0.5
    if rand < 0.5
        X_LEO^it = X_i^(it+1) + XL
    else
        X_LEO^it = x_best + XL
    end
    X_i^(it+1) = X_LEO^it
end
```

(32)

where,

$$XL = \mu_1 \times (\theta_1 \times x_{best} - \theta_2 \times x_r^{it}) + \mu_2 \times F_1 \times (\theta_3 \times (Xnew2_m^{it} - Xnew1_m^{it}) + \theta_2 \times (x_{a1}^{it} - x_{a2}^{it}))/2 \quad (33)$$

The μ_1 and μ_2 are two randomly generated integers that resides in the range of [−1, 1]. θ_1, θ_2, and θ_3 are also three random values which are formulated under the condition that a randomly generated number in the range of [0, 1], $rand_1$ is less than 0.5, otherwise, their designated value is 1. So, they can be written as,

$$\theta_1 = \begin{Bmatrix} 2 * rand \; ; \; if \; rand_1 < 0.5 \\ 1 \quad ; \; otherwise \end{Bmatrix} \quad (34)$$

$$\theta_2 = \begin{Bmatrix} rand \; ; \; if \; rand_1 < 0.5 \\ 1 \; ; \; otherwise \end{Bmatrix} \quad (35)$$

$$\theta_3 = \begin{Bmatrix} rand \; ; \; if \; rand_1 < 0.5 \\ 1 \; ; \; otherwise \end{Bmatrix} \quad (36)$$

Here, $rand_1$ is a number in the range of [0, 1].

The following figure represents the flowchart of GBO algorithm:

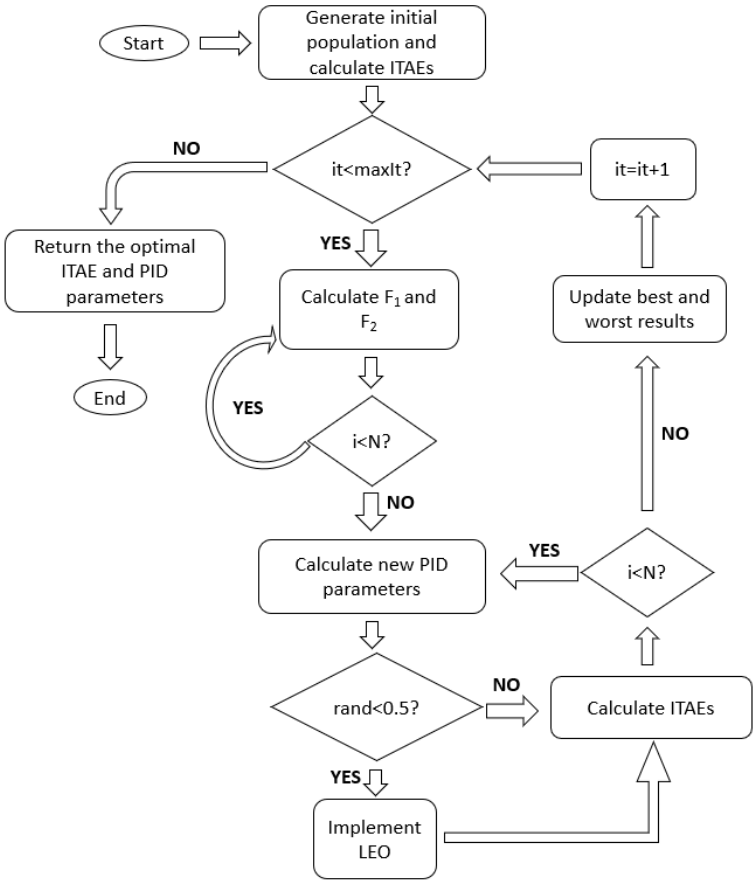


Fig. 6: GBO flowchart

2.1.2 Enhanced Gradient Based Optimizer (EGBO):

Properly setting two parameters F_1 and F_2 in the GBO is critical for improving precision and convergence speed. To increase the efficiency of the process in EGBO, small control parameter values are disproportionately assigned to superior solutions, whereas high values are disproportionately given to worse solutions [20]. A rank-based process is employed to accomplish this, in which all particular entity are sorted in ascending order depending on the values of their objective functions. Then, these parameters are constructed using the following technique,

$$
\begin{aligned}
&if \;\; rand < 0.5 \\
&\quad F_1 = a1/N + 0.1 \times randn \\
&\quad F_2 = a2/N + 0.1 \times randn \\
&else
\end{aligned}
\tag{37}
$$

$F_1 = i/N + 0.1 \times randn$

$F_2 = i/N + 0.1 \times randn$

end

The crossover operator is used to increase population variety by merging the GBO's solution vectors with the existing solution in the following way,

for j = 1 to d

if $rand < pc_r$ *or* $j = j_{rand}$

if $rand < 0.5$

$Xnew_j = Xnew1_j$

else

$Xnew_j = Xnew2_j$ (38)

end

else

$Xnew_j = x_{i,j}$

end

end

Here, pc_r is defined as the crossover probability rate. In this work, pc_r is defined based on the following equation,

$$pc_r = \left(\frac{i}{N}\right) + 0.1 * randn \quad (39)$$

A modified version of the LEO, named MLEO is created to improve the performance of the original LEO. The enhanced MLEO is presented in the pseudo-code below,

if $rand < LC$

if $rand < 0.5 \times (1 - \frac{it}{MaxIt})$

$Xnew = x_{best3} + F_1 \times (x_{best2} - x_r^{it}) + F_2 \times (x_{a1}^{it} - x_{a2}^{it})$

else (40)

$Xnew = x_{best} + F_1 \times (x_{best2} - x_r^{it}) + F_2 \times (x_{a1}^{it} - x_{a2}^{it})$

end

end

In this work, LC denotes a chaotic logistic map, initiated with a value of 0.7. LC is updated in each iteration following the equation below,

$$LC = 4 * LC * (1- LC) \quad (41)$$

The following figure depicts the working flowchart of the EGBO algorithm where the main differences between EGBO and GBO are highlighted:

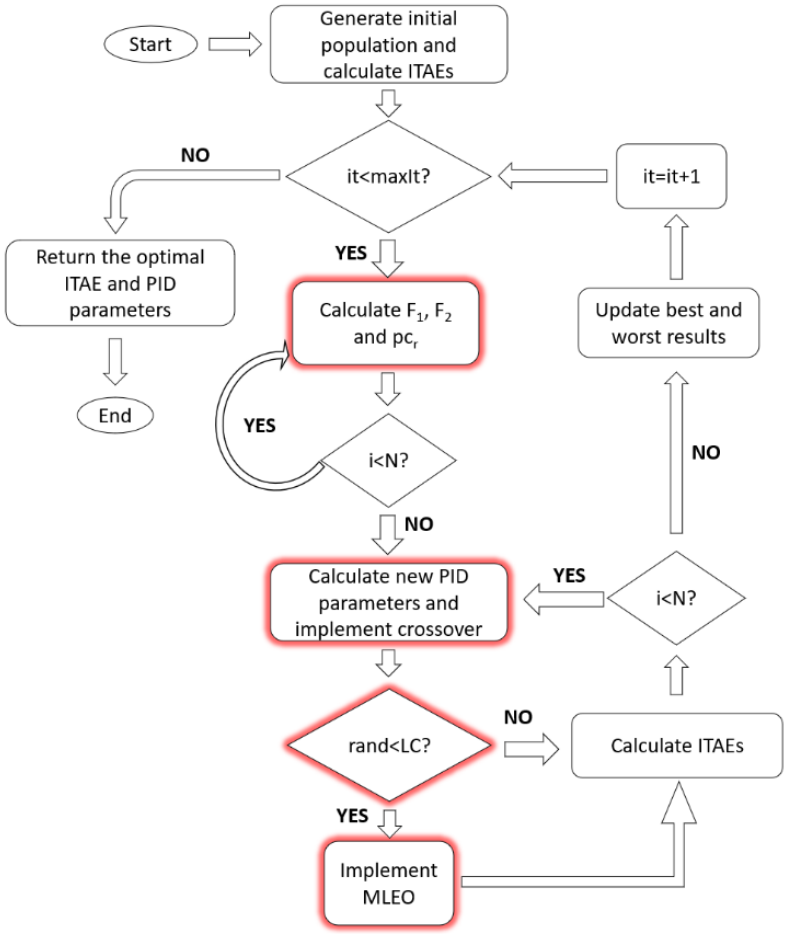


Fig. 7: EGBO flowchart

2.1.3 Chimp Optimization Algorithm (ChOA):

Chimp Optimization Algorithm (ChOA) is a meta-heuristic algorithm which is inspired from the individual intelligence and sexual motivation of chimps during their group hunting [18].

Chimp colony consists of four types of chimps for hunting [18]:

1. Driver
2. Barrier
3. Chaser
4. Attacker

These four individual types are considered as four individual groups and each group follows their own patterns to search local and global problem space [18]. The hunting process of chimps is divided into two main phases: Exploration, which consists of driving, blocking and chasing the prey and exploitation, which consists of attacking the prey [18].

a) Driving and chasing (exploration phase): Mathematical model for driving and chasing the prey is as follows,

$$\vec{d} = \left| \vec{c}.\vec{x}_{prey}(t) - \vec{m}.\vec{x}_{chimp}(t) \right| \quad (42)$$

$$\vec{x}_{chimp}(t+1) = \vec{x}_{prey}(t) - \vec{a}.\vec{d} \quad (43)$$

Where,

$$\vec{a} = 2.\vec{f}.\vec{r}_1 - \vec{f} \quad (44)$$

$$\vec{c} = 2.\vec{r}_2 \quad (45)$$

Here, $\vec{d}$ is the distance vector between chimp and prey, t is the number of current iterations, $\vec{a}, \vec{m} \& \vec{c}$ are the coefficient vectors, $\vec{x}_{prey}$ is the prey position vector, $\vec{x}_{chimp}$ is the chimp position vector, $\vec{m}$ holds a chaotic value based on various chaotic map, $\vec{r}_1 \& \vec{r}_2$ are random vectors in [0,1] range and $\vec{f}$ is reduced non-linearly from 2.5 to 0 through the iteration process (in both exploitation and exploration phase).

b) Attacking method (exploitation phase): To mathematically model the attacking behavior of chimps, two approaches are designed [18]. The chimps are capable of exploring the prey's location (by driving, blocking and chasing) and then encircling it. The hunting process is usually conducted by attacker chimps [18].

In order to mathematically simulate the behavior of the chimps, it is assumed that the first attacker (best solution available), driver, barrier and chaser are informed about the location of potential prey and other chimps are forced to update their positions according to the best chimps' location [18]. The mathematical model is as follows:

$$\vec{d}_{Attacker} = \left|\vec{c}_1.\vec{x}_{Attacker} - \vec{m}_1.\vec{x}\right|,\ \vec{d}_{Barrier} = \left|\vec{c}_2.\vec{x}_{Barrier} - \vec{m}_2.\vec{x}\right| \\ \vec{d}_{Chaser} = \left|\vec{c}_3.\vec{x}_{Chaser} - \vec{m}_3.\vec{x}\right|,\ \vec{d}_{Driver} = \left|\vec{c}_4.\vec{x}_{Driver} - \vec{m}_4.\vec{x}\right| \quad (46)$$

$$\vec{x}_1 = \vec{x}_{Attacker} - \vec{a}_1\left(\vec{d}_{Attacker}\right),\ \vec{x}_2 = \vec{x}_{Barrier} - \vec{a}_2\left(\vec{d}_{Barrier}\right) \\ \vec{x}_3 = \vec{x}_{Chaser} - \vec{a}_3\left(\vec{d}_{Chaser}\right),\ \vec{x}_4 = \vec{x}_{Driver} - \vec{a}_4\left(\vec{d}_{Driver}\right) \quad (47)$$

$$\vec{x}(t+1) = \frac{\vec{x}_1 + \vec{x}_2 + \vec{x}_3 + \vec{x}_4}{4} \quad (48)$$

Here, $\vec{x}_1, \vec{x}_2, \vec{x}_3 \,\&\, \vec{x}_4$ are the best position vectors of attacker, barrier, chaser and driver, respectively.

2.1.4 Marine Predators Algorithm (MPA):

Background:

Brownian motion:

Before going into the steps of the proposed algorithm, it's important to understand the mathematical models for two different types of random walks:

(i) Brownian and
(ii) Lévy motions.

Brownian motion is a stochastic process whose step duration is determined by a probability function defined by a Normal (Gaussian) distribution with zero mean (μ =0) and unit variance (σ^2 =1). For this motion, the controlling Probably Density Function (PDF) is as follows [17]:

$$f_B(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{\sqrt{2\pi}}\exp(-\frac{x}{2}) \quad (49)$$

Lévy flight:

The step sizes of a Lévy flight are governed by a probability function given by the Lévy distribution (power-law tail) [21]:

$$L(x_j) \approx \left| x_j \right|^{1-\alpha} \tag{50}$$

where $1 < \alpha \leq 2$ is the power-law exponent and x_j is the flight length. In integral form, the probability density of a Lévy stable process is defined as [21]:

$$f_L(x;\alpha,\gamma) = \frac{1}{\pi}\int_0^{\infty} \exp(-\gamma q^{\alpha}) \cos(qx) dq \tag{51}$$

Where α determines the distribution index and regulates the process's scale parameters, and γ determines the scale unit. When x owns a large value, the integral solution usually involves utilizing the series expansion method as follows [21]:

$$f_L(x;\alpha,\gamma) \approx \frac{\gamma\ \Gamma(1+\alpha)\sin\left(\frac{\pi\alpha}{2}\right)}{\pi x^{(1+\alpha)}}, x \to \infty \tag{52}$$

Where Γ is the Gamma function, in which for α integers, $\Gamma(1+\alpha)$ equals α. For generating random numbers based on Lévy distribution the following method is used [17]:

$$Levy(\alpha) = 0.05 \times \frac{x}{\left| y \right|^{\frac{1}{\alpha}}} \tag{53}$$

Here, x and y are two normal distribution variables with σ_x and σ_y standard deviations, x and y are defined as [21]:

$$x = Normal(0, \sigma_x^2) \tag{54}$$

$$y = Normal(0, \sigma_y^2) \tag{55}$$

Where σ_x , σ_y and α is defined as,

$$\sigma_x = \left[\frac{\Gamma(1+\alpha)\sin\left(\frac{\pi\alpha}{2}\right)}{\Gamma\left(\frac{(1+\alpha)}{2}\right)\alpha 2^{\frac{(\alpha-1)}{2}}} \right] \quad (56)$$

$$\sigma_y = 1 \text{ and } \alpha = 1.5 \quad (57)$$

While the Lévy flight can efficiently and thoroughly search a close neighbourhood because to its short step lengths and explore other portions of the domain due to its large steps, it cannot cover all areas of a domain. Brownian motion walks, on the other hand, can trace and investigate distant portions of the neighbourhood but cannot search as precisely and deeply as the Lévy approach. Thus, The Marine Predators Algorithm (MPA) takes advantage of the unique qualities of the Lévy strategy, as well as elements of Brownian motion, to optimize exploration and exploitation of a region.

MPA Formulation:

MPA is a population-based approach in which the initial answer is uniformly allocated over the search space as the first trial:

$$X_0 = X_{\min} + rand(X_{\max} - X_{\min}) \quad (58)$$

Here, the lower and upper bounds for variables are $X_{\min}$ and $X_{\max}$ respectively, and *rand* is a uniform random vector in the range of 0 to 1.

According to the *survival of the fittest theory*, apex predators in nature are more skilled foragers. As a result, the fittest solution is designated as a top predator in order to construct the Elite matrix. This matrix's arrays are in charge of looking for and discovering prey based on information about the prey's location.

$$Elite = \begin{bmatrix} X_{1,1}^I & X_{1,2}^I & \cdots & X_{1,d}^I \\ X_{2,1}^I & X_{2,2}^I & \cdots & X_{2,d}^I \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1}^I & X_{n,2}^I & \cdots & X_{n,d}^I \end{bmatrix}_{n \times d} \quad (59)$$

Where $\overrightarrow{X^I}$ the top predator vector, which is multiplied by n to get the *Elite* matrix. The number of search agents is n and the number of dimensions is d.

Prey is another matrix with the same dimension as *Elite*, and it is used by predators to update their positions. In a nutshell, initialization creates the initial *Prey*, from which the fittest predator builds the *Elite*. The following is a representation of the *Prey* matrix:

$$Prey = \begin{bmatrix} X_{1,1} & X_{1,2} & \cdots & X_{1,d} \\ X_{2,1} & X_{2,2} & \cdots & X_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ X_{n,1} & X_{n,2} & \cdots & X_{n,d} \end{bmatrix}_{n \times d} \tag{60}$$

The *j-th* dimension of the *i-th* prey is represented by $X_{i,j}$. It should be noted that these two matrices play a crucial role in the entire optimization process.

MPA Optimization Scenarios:

The MPA optimization method is separated into three primary phases, each of which takes into account a particular velocity ratio while simulating the whole life of a predator and prey: (1) when the predator is going faster than the prey or in high velocity ratio, (2) when the predator and prey are moving at nearly the same speed or in unit velocity ratio, and (3) when the predator is moving faster than the prey or in low velocity ratio.

Phase 1: When the predator is going faster than the prey in a high-velocity ratio, this condition occurs during the earliest optimization stages, when exploration is important. The optimum strategy for a predator in a high-velocity ratio (v ≥ 10) is to not move at all. The mathematical model of this rule is as follows:

$$\begin{aligned} &While\ Iter < \frac{1}{3} Max\_Iter \\ &\overrightarrow{stepsize_i} = \overrightarrow{R_B} \otimes (\overrightarrow{Elite_i} - \overrightarrow{R_B} \otimes \overrightarrow{Prey_i}) \qquad i = 1, ..., n \\ &\overrightarrow{Prey_i} = \overrightarrow{Prey_i} + P.\vec{R} \otimes \overrightarrow{stepsize_i} \end{aligned} \tag{61}$$

Where $\overrightarrow{R_B}$ represents Brownian motion and is a vector containing random values based on the Normal distribution. Entry-by-entry multiplications are shown in the $\otimes$

notation. By multiplying $\overrightarrow{R_B}$ by *Prey*, prey movement is simulated. $\vec{R}$ is a vector of uniform random numbers in the range [0,1], and P=0.5 is a constant value.

Phase 2: When both the predator and the prey are moving at the same speed, this is known as a unit velocity ratio. According to the rule, if prey moves in Lévy at unit velocity ratio (v ≈ 1), the ideal predator approach is Brownian. As a result, prey movements are considered in Lévy, while predator movements are considered in Brownian.

$$\begin{aligned} &While\ \ \frac{1}{3}Max\_Iter < Iter < \frac{2}{3}Max\_Iter \\ &\text{For the first half of the population} \\ &\overrightarrow{stepsize_i} = \overrightarrow{R_L} \otimes (\overrightarrow{Elite_i} - \overrightarrow{R_L} \otimes \overrightarrow{Prey_i}) \qquad i = 1, ..., n/2 \\ &\overrightarrow{Prey_i} = \overrightarrow{Prey_i} + P.\vec{R} \otimes \overrightarrow{stepsize_i} \end{aligned} \tag{62}$$

Where $\overrightarrow{R_L}$ denotes Lévy movement and is a vector of random numbers based on the Lévy distribution. The addition of the step size to the prey position simulates prey movement, while the multiplication of $\overrightarrow{R_L}$ and *Prey* simulates prey movement in a Lévy manner.

$$\begin{aligned} &\text{For the second half of the population} \\ &\overrightarrow{stepsize_i} = \overrightarrow{R_B} \otimes (\overrightarrow{R_B} \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}) \qquad i = 1, ..., n/2 \\ &\overrightarrow{Prey_i} = \overrightarrow{Elite_i} + P.CF \otimes \overrightarrow{stepsize_i} \end{aligned} \tag{63}$$

Where,

$$CF = \left(1 - \frac{Iter}{Max\_Iter}\right)^{\left(2\frac{Iter}{Max\_Iter}\right)} \tag{64}$$

While *CF* is used as an adaptive parameter to adjust predator movement step size.

Multiplication of $\overrightarrow{R_B}$ and *Elite* replicates predator movement in Brownian motion, while prey updates its position in response to predator movement in Brownian motion.

***Phase 3*:** When the predator is going faster than the prey in a low-velocity ratio it is usually accompanied with a high level of exploitation capacity. This scenario occurs in the final stage of the optimization process, and Lévy is the best predator tactic when the velocity ratio is modest (v = 0.1). This phase is described as follows:

$$\begin{aligned} &While\ Iter > \frac{2}{3} Max\_Iter \\ &\overrightarrow{stepsize_i} = \overrightarrow{R_L} \otimes (\overrightarrow{R_L} \otimes \overrightarrow{Elite_i} - \overrightarrow{Prey_i}) \qquad i = 1, ..., n \\ &\overrightarrow{Prey_i} = \overrightarrow{Elite_i} + P.CF \otimes \overrightarrow{stepsize_i} \end{aligned} \tag{65}$$

In the Lévy method, multiplying $\overrightarrow{R_L}$ and *Elite* simulates predator movement, while adding the step size to *Elite* position simulates predator movement to aid in the updating of prey position.

Eddy formation and FADs' effect:

Environmental factors such as eddy formation or the effects of Fish Aggregating Devices (FADs) are another factor that causes behavioural changes in marine predators [21]. Local optima are referred to as FADs, and their influence is described as "trapping" in these points in search space. Simulating these larger jumps prevents local optima from becoming stagnant. As a result, the FADs effect is represented numerically as:

$$\begin{aligned} &if\ \ r \le FADs, \\ &\overrightarrow{Prey_i} = \overrightarrow{Prey_i} + CF\left[\overrightarrow{X_{\min}} + \vec{R} \otimes \left(\overrightarrow{X_{\max}} - \overrightarrow{X_{\min}}\right)\right] \otimes \vec{U} \end{aligned} \tag{66}$$

$$\begin{aligned} &if\ \ r > FADs \\ &\overrightarrow{Prey_i} = \overrightarrow{Prey_i} + \left[FADS(1-r) + r\right]\left(\overrightarrow{Prey_{r1}} - \overrightarrow{Prey_{r2}}\right) \end{aligned} \tag{67}$$

The following figure represents the flow chart of MPA algorithm:

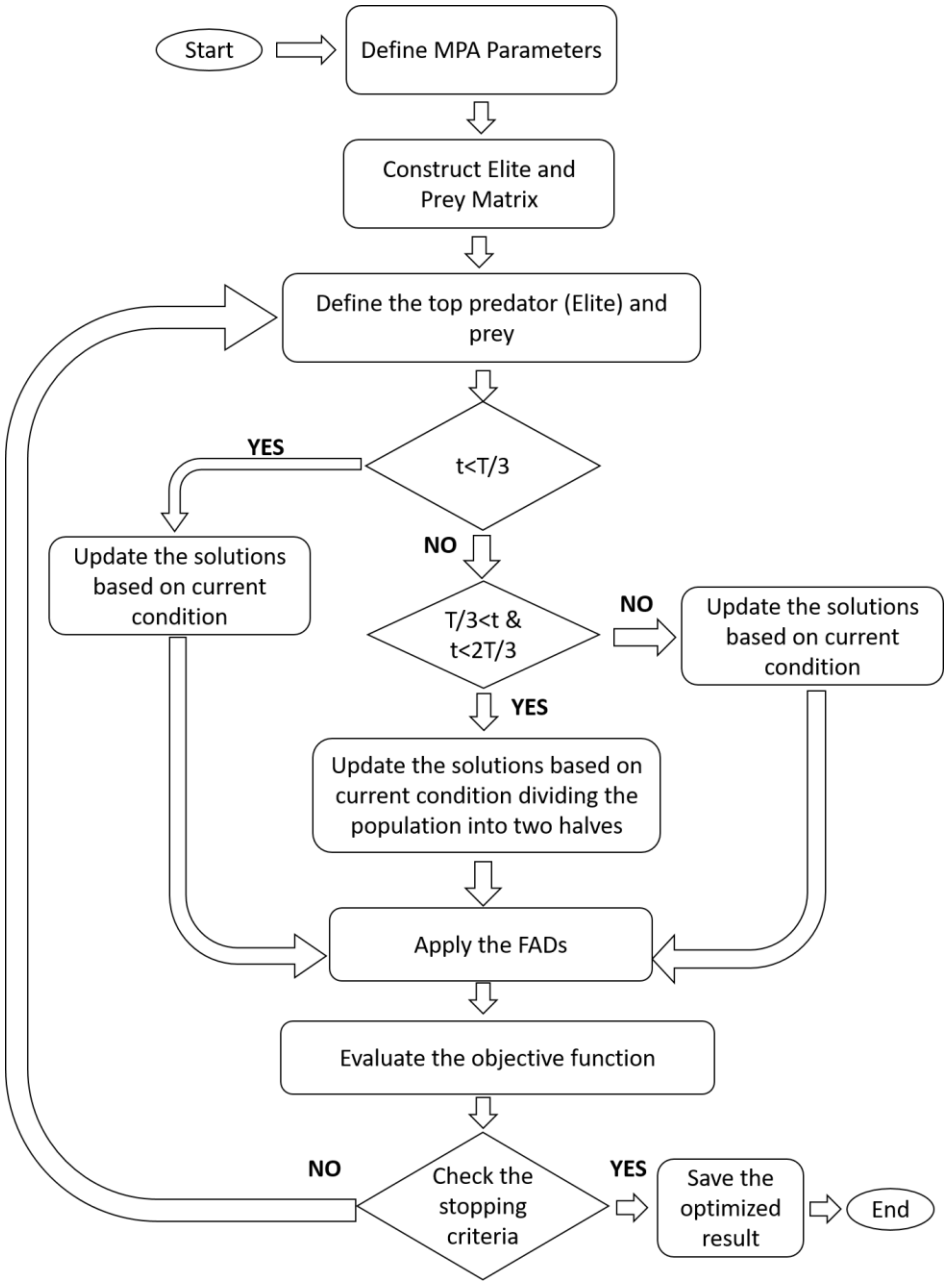


Fig. 8: MPA flowchart

The probability of FADs effect on the optimization process is *FADs* = 0.2. $\vec{U}$ is a binary vector that contains arrays of zero and one. This is made by creating a random vector in the range [0,1] and setting its array to zero if it is less than 0.2 and one if it is more than 0.2. In the range [0,1], r is the uniform random number. The lower and higher bounds of the dimensions are represented by the vectors $\overrightarrow{X_{\min}}$ and $\overrightarrow{X_{\max}}$. Random indexes of the prey matrix are denoted by the *r1* and *r2* subscripts.

Marine memory:

Predators of the sea have a good memory for remembering where they have been successful in foraging. Memory saving in MPA simulates this capability. After updating the *Prey* and including the FADs impact, *Prey* matrix is examined for fitness to update the *Elite*. Each current iteration's solution is compared against its counterpart from the previous iteration, and the current one replaces the previous one if it is more suited. The MPA pseudo-code is listed below:

```
Initialize search agents (Prey) populations
i=1,…,n

While termination criteria are not met
Calculate the fitness and construct the Elite
matrix

If Iter<Max_Iter/3
Update prey based on Eq. 61

Else if Max_Iter/3<Iter<2*Max_Iter/3
For the first half of the populations
(i=1,…,n/2)
Update prey based on Eq. 62

For the other half of the populations
(i=n/2,…,n)
Update prey based on Eq. 63

Else if Iter>2*Max_Iter/3
Update prey based on Eq. 65
End (if)

Accomplish memory saving and Elite update

Applying FADs effect and update based on Eq. 66
& 67

Accomplish memory saving and Elite update

End while
```

2.1.5 Multi Trial vector based Differential Evolution (MTDE):

The no-free lunch theorem, which states that different search techniques are required to tackle diverse optimization problems, is addressed by the multi-trial vector-based differential evolution (MTDE) algorithm [22]. For all global optimization problems with varied properties, no single algorithm can show superior performance. In other words, different search strategies might be effective on different problems and at different points of the search process, depending on several criteria like as the multimodality of the problem, the chosen EA, and the global exploration/local exploitation stages of the search process. The MTV method is used in the MTDE algorithm, which has two key steps: initializing and movement. *N* individuals x_1, ..., x_N are randomly dispersed in the search space in the initializing stage to assume a constrained boundary between lower bound (l) and upper bound (u) by utilizing the following equation:

$$x_{ij} = l_j + (u_j - l_j) \times rand(0.1) \quad (68)$$

Where, x_{ij} is the i-th individual's location in the j-th dimension, and l_j and u_j are the j-th dimension's lower and upper bounds. Individual x_i is represented by the vector $xi = \{x_{i1}, x_{i2},..., x_{iD}\}$ where *D* specifies the number of dimensions of the search space and N individuals are stored in the matrix $X_{N\times D}$.

Movement step using MTV approach:

The MTDE movement phase is based on the previously mentioned MTV method, which takes into account three trial vector producers: R-TVP, L-TVP, and G-TVP. The purpose of combining these three trial vector generators is to give MTDE for a variety of issues. R-TVP is used to avoid local optima trapping and maintain diversity, L-TVP benefits from fast convergence and a proper balance of exploration and exploitation, and G-TVP represents a significant ability in exploitation and escaping local optima. As indicated in the flowchart, the MTDE algorithm's movement step has four sub-steps: winner-based distributing, multi-trial vector producing, evaluating, and life-time archiving. The winner-based distributing sub-step is based on the following Definitions:

Definition 1:

Given *k WinIter*, each with *n* iterations, and only one winner TVP denoted by Win-TVP, there is only one winner TVP for each *WinIter*. R-TVP is regarded Win-TVP in the first *WinIter*, while Win-TVP in subsequent *WinIters* is the TVP with the highest improved rate in the previous *WinIter*. The enhanced rate of X-TVP (where X stands for R, L, or G) denoted by $_{IRX\text{-}TVP}$ is determined using the equation below.

$$IR_{X-TVP} = \frac{IF_{X-TVP}}{FE_{X-TVP}} \quad (69)$$

Where, the number of enhanced finesses by the X-TVP and the number of function evaluations in a *WinIter* are represented by IFX-TVP and FEX-TVP, respectively. The winner-based distribution sub-step distributes the population amongst the trial vectors R-TVP, L-TVP, and G-TVP after determining the Win-TVP with the help of the distribution policy based on Definition 2.

Definition 2:

$N_{R\text{-}TVP}$, $N_{L\text{-}TVP}$, and $N_{G\text{-}TVP}$ are the sizes of the subpopulations of $X_{R\text{-}TVP}$, $X_{L\text{-}TVP}$, and $X_{G\text{-}TVP}$, respectively. The size of these subpopulations is set by two reward and punishment procedures defined based on the Win-TVP. *N* individuals are then allocated randomly amongst these subpopulations.

Reward rule:

If R-TVP or L-TVP is Win-TVP,

$$\text{then } N_{Win-TVP} = 0.6 \times N \text{ and } N_{Loser-TVPs} = 0.2 \times N \quad (70)$$

Penalty rule:

If G-TVP is Win-TVP,

$$\text{then } N_{G-TVP} = 0.2 \times N \text{ and } N_{Loser-TVPs} = 0.4 \times N \quad (71)$$

The next paragraphs provide basic preliminaries and essential concepts before describing how three introduced TVPs use their movement strategy.

In order to construct the final produced trial vectors, both R-TVP and G-TVP cross their produced trial vectors across their corresponding individuals. A scale factor is calculated for each individual *xi* using the Cauchy distribution, with

$$F_i = randc_i\left(\mu_f, \sigma\right) \quad (72)$$

where μ_f is the mean value of improved scale factors and σ is the variance with constant value 0.2. F_i must be between 0 and 1, and if $F_i > 1$, it is considered by 1,

and if $F_i \leq 0$ it will be calculated again, with the parameter f set to 0.5. If any improved individual in the population exists in subsequent iterations, μ_f is determined using the weighted Lehmer mean, and if there is no improved individual, the scale factors remain unaltered.

$$\mu_f = \frac{\sum_{f_i \in S_f} w_{f_i} \times f_i^2}{\sum_{f_i \in S_f} w_{f_i} \times f_i} \tag{73}$$

Where S_f is the set of all *Fs* in the current iteration for which $f(u_i) < f(x_i)$, and the weight is derived according to the previous equation,

$$w_{f_i} = \frac{\Delta f_i}{\sum_{f_i \in S_f} \Delta f_i} \tag{74}$$

Where, $\Delta f_i = f(x_i) - f(u_i)$.

Using the locations of a random individual x_{u_pop} from the unionPopulation and two individuals, x_{i_best} and x_{i_worst}, R-TVP shifts individual x_i from its subpopulation. The i-th elements of the ascending and descending ordered representation of the fitness values of the subpopulation $X_{R\text{-}TVP}$ are x_{i_best} and x_{i_worst}, respectively. Then, using the transformation matrix *M* and its binary inverse $\overline{M}$ and v_i calculated and the trial vector u_i in R-TVP is created.

$$u_i = M \times x_i + \overline{M} \times v_i \tag{75}$$

$$v_i = x_i + F_i \times \left(x_{i_{best}} - x_i \right) + F_i \times \left(x_{i_{worst}} - x_i \right) + a_1 \times \left(x_{u\_pop} - x_i \right) \tag{76}$$

Where $a_1 = 2 - iter \times \left(\frac{2}{MaxIter} \right)$ (77)

The difference between the positions of two randomly picked people x_{r1} and x_{r2} from XL-TVP and the difference of x_i from a randomly selected individual x_{u_pop} derive from the unionPopulation is considered by L-TVP to modify the position of x_i. As demonstrated in the equation below,

$$v_i = x_i + F_i \times \left(x_{r1} - x_{r2} \right) + a_2 \times \left(x_{u\_pop} - x_i \right) \quad (78)$$

Where,

$$a_2 = initial - (initial - final) \times \left(((MaxIter - iter) / MaxIter)^{Mu} \right) \quad (79)$$

The flowchart of MTDE algorithm is presented in the following figure:

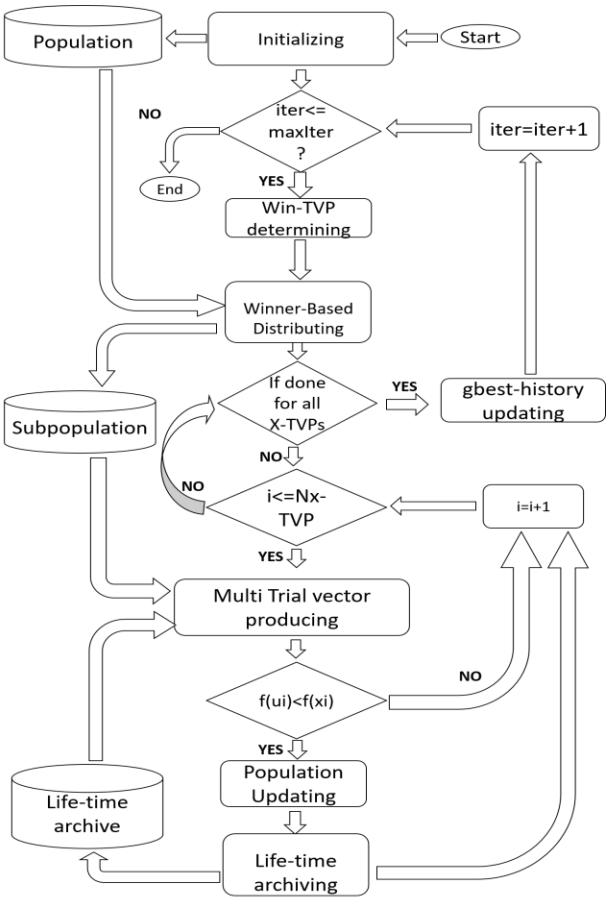


Fig. 9: MTDE flowchart

The initial and final values of control parameter a_2 are *initial* and *final*, respectively, and *Mu* is a dimension dependent value. The final trial vector u_i, unlike the R-TVP, is based on individual learning rather than evolution, hence it does not utilize a crossover and $u_i = v_i$.

G-TVP is developed to address the problem of trapping in local optima by utilizing a *gbest* history dubbed *gbest-history*. The *gbest-history* is initialized with simply the *gbest* in the first iteration. The *gbest* of each iteration is then added to the *gbest-history* in the subsequent iterations. When *gbest-history* is full, the current *gbest* is replaced by a previous *gbest* with a better fitness value from the past. A matrix $M_{gb\text{-}h}$ with $N_{G\text{-}TVP}$ rows and D columns is also created from the *gbest-history* for the G-TVP. The $M_{gb\text{-}h}$ is built in each iteration by reproducing the *gbest-history* for (N/m) times where m is the number of gbest-history members. The trial vector u_i is then computed which is analogous to the R-TVP utilizing the M and $\overline{M}$ matrixes and v_i computation. Here, $x_{i\_gb\text{-}h}$ is the i-th row of the $M_{gb\text{-}h}$, and x_{r1} and x_{r2} are two people from the $X_{G\text{-}TVP}$ chosen at random.

$$u_i = M \times x_i + \overline{M} \times v_i \tag{80}$$

$$v_i = x_{i\_gb-h} + a_2 \times \left(x_{r1} - x_{r2} \right) \tag{81}$$

The quality of each trial vector u_i from $U_{X\text{-}TVP}$ is compared to its matching individual x_i from $X_{X\text{-}TVP}$ in the assessing sub-step. If $f\left(u_i\right) \geq f\left(x_i\right)$ then xi in the population remains unaffected. If $f\left(u_i\right) < f\left(x_i\right)$, the population updating mechanism advances x_i by u_i.

Additionally, the value of F_i is saved, and the number of individuals benefitted by each TVP is increased by one. Then, as an inferior solution, the life-time archiving sub-step adds x_i to the life-time archive. The life-time archive is initially empty, and it can store the position and lifetime of N individuals. The lifetime of archived individuals is increased by one at the conclusion of each iteration, while N younger archived individuals with shorter lifetimes are kept. Finally, the iteration counter (*iter*) is raised by one, and the evolutionary search can be repeated until the maximum number of iterations (*MaxIter*) is reached.

2.2 Two Area AGC with PID Controller:

2.2.1 General Representation of the Model:

In this work, an interconnected two area non reheat thermal power system is considered under study. The system model with PID controller is as follows:

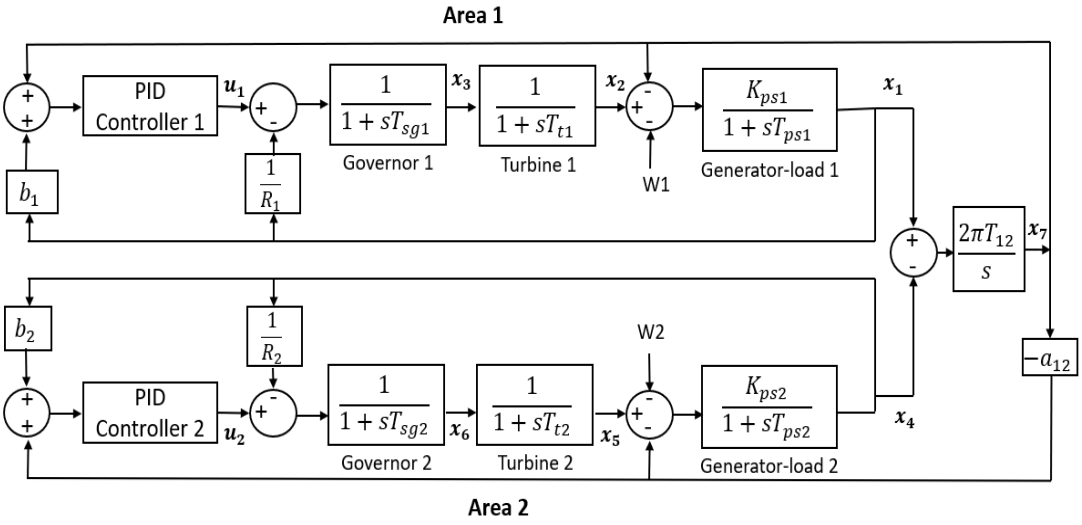


Fig. 10: System model with PID controller

Here, PID controller is used in both areas. In the model, u_1 and u_2 are the outputs of the PID controllers of area1 and area 2, respectively.

2.2.2 State Space Modeling of the System:

The state-space model of the PID incorporated two-area thermal power system is represented as:

$$\dot{x}_1 = \frac{K_{ps1}}{T_{ps1}}(x_2 - x_7 - w_1) - \frac{1}{T_{ps1}}x_1 \tag{82}$$

$$\dot{x}_2 = \frac{1}{T_{t1}}(x_3 - x_2) \tag{83}$$

$$\dot{x}_3 = \frac{1}{T_{sg1}}(u_1 - \frac{x_1}{R_1} - x_3) \tag{84}$$

$$\dot{x}_4 = \frac{K_{ps2}}{T_{ps2}}(x_5 - a_{12}x_7 - w_2) - \frac{1}{T_{ps2}}x_4 \tag{85}$$

$$\dot{x}_5 = \frac{1}{T_{t2}}(x_6 - x_5) \tag{86}$$

$$\dot{x}_6 = \frac{1}{T_{sg2}}(u_2 - \frac{x_4}{R_2} - x_6) \tag{87}$$

$$\dot{x}_7 = 2\pi T_{12}(x_1 - x_4) \tag{88}$$

$$\dot{u}_1 = x_1e_1 + x_2e_2 + x_3e_3 + x_4e_4 + x_5e_5 + x_6e_6 + x_7e_7 + w_1e_8 \tag{89}$$

$$\dot{u}_2 = x_1f_1 + x_2f_2 + x_3f_3 + x_4f_4 + x_5f_5 + x_6f_6 + x_7f_7 + w_2e_8 \tag{90}$$

The detailed expressions of the coefficients e and f of Eq. (89-90) can be found in appendix C. The state matrix which is derived from Eq. (82-90) can be defined as,

$$A = \begin{bmatrix} \frac{-1}{T_{ps1}} & \frac{K_{ps1}}{T_{ps1}} & 0 & 0 & 0 & 0 & \frac{-K_{ps1}}{T_{ps1}} & 0 & 0 \\ 0 & \frac{-1}{T_{t1}} & \frac{1}{T_{t1}} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{-1}{R_1T_{sg1}} & 0 & \frac{-1}{T_{sg1}} & 0 & 0 & 0 & 0 & \frac{1}{T_{sg1}} & 0 \\ 0 & 0 & 0 & \frac{-1}{T_{ps2}} & \frac{K_{ps2}}{T_{ps2}} & 0 & \frac{K_{ps2}}{T_{ps2}}a_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{-1}{T_{t2}} & \frac{1}{T_{t2}} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{-1}{R_2T_{sg2}} & 0 & \frac{-1}{T_{sg2}} & 0 & 0 & \frac{1}{T_{sg2}} \\ 2\pi T_{12} & 0 & 0 & -2\pi T_{12} & 0 & 0 & 0 & 0 & 0 \\ e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & 0 & 0 \\ f_1 & f_2 & f_3 & f_4 & f_5 & f_6 & f_7 & 0 & 0 \end{bmatrix} \tag{91}$$

2.2.3 Controller & Objective Function:

In the LFC field, application of several kinds of controllers are observed such as PI controllers [18], classical PID controllers [18], [29], [17], and [26], Ziegler-Nichols

tuned PID controllers [29] and [26], PIDN controllers [30] and [27], Fuzzy PID controllers [43], FOPID (Fractional Order PID) controllers [31], ANN-based controllers [14], etc. For this study, a generic PID controller is chosen due to its industry-wide availability and affordability. The transfer function of the proposed controller, which is adopted from [56]:

$$TF_{PID} = K_p + \frac{K_i}{s} + K_d s \tag{38}$$

K_p, K_i, and K_d are the proportional, integral, and derivative gains of the controllers respectively. The inputs of the controllers are ACEs defined as,

$$ACE_1 = (K_{p1} + \frac{K_{i1}}{s} + K_{d1}s)u_1 = x_1 b_1 + x_7 \tag{39}$$

$$ACE_2 = (K_{p2} + \frac{K_{i2}}{s} + K_{d2}s)u_2 = x_4 b_2 - a_{12} x_7 \tag{40}$$

The operational objectives of the LFC are to maintain reasonably uniform frequency, to divide the load between generators, and to control the tie-line interchange schedules [3]. To keep the system frequency deviation within acceptable limits, the controller gains are optimized with the objective of minimization of ITAE, reducing ACEs of both areas, along with the deviation of the tie-line power flow. Hence, the optimization problem is formulated as follows:

Minimize $ITAE = \int_{t=0}^{t_{final}} t(|X_1| + |X_4| + |X_7|)dt$ (41)

Subject to the following controller gain boundaries,

$$K_{p1}^{\min} \le K_{p1} \le K_{p1}^{\max} \tag{42}$$

$$K_{i1}^{\min} \le K_{i1} \le K_{i1}^{\max} \tag{43}$$

$$K_{d1}^{\min} \le K_{d1} \le K_{d1}^{\max} \tag{44}$$

$$K_{p2}^{\min} \le K_{p2} \le K_{p2}^{\max} \tag{45}$$

$$K_{i2}^{\min} \le K_{i2} \le K_{i2}^{\max} \tag{46}$$

$$K_{d2}^{\min} \le K_{d2} \le K_{d2}^{\max} \tag{47}$$

In this work the search space for the parameters of the controller is chosen as,

$$-16 \le K_{p1} \le -6 \tag{48}$$

$$-45 \le K_{i1} \le -15 \quad (49)$$

$$-8 \le K_{d1} \le -3 \quad (50)$$

$$-16 \le K_{p2} \le -6 \quad (51)$$

$$-45 \le K_{i2} \le -15 \quad (52)$$

$$-8 \le K_{d2} \le -3 \quad (53)$$

2.3 Two Area AGC with FOPID Controller:

2.3.1 General Representation of the Model:

The FOPID incorporated system is exactly same as the PID controller incorporated system except the controller part. The system model with FOPID controller is as follows:

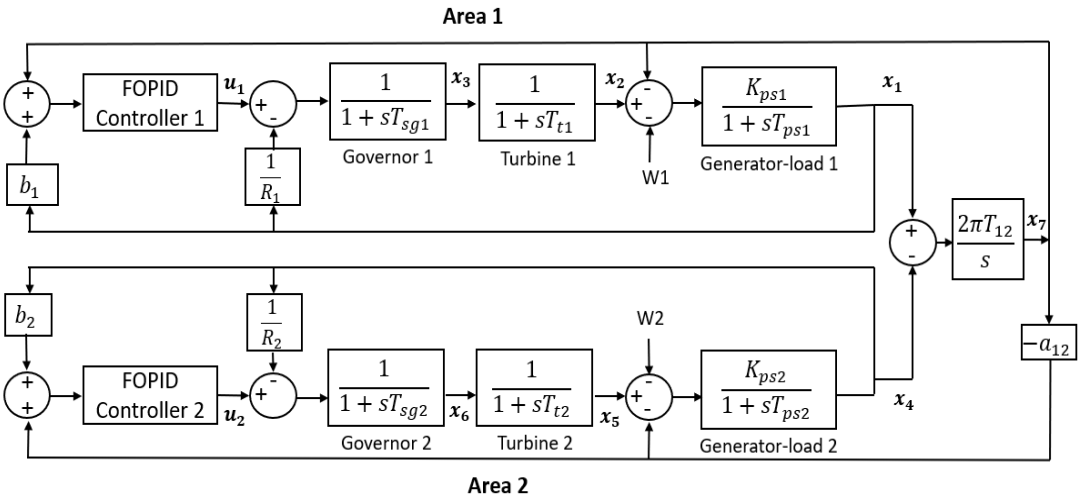


Fig. 11: System model with FOPID controller

As mentioned earlier, FOPID controller has two more extra tunable parameters $(\lambda \; and \; \mu)$ which give the controller more authority to reduce error. The following figure represents the structure of the FOPID controller [23]:

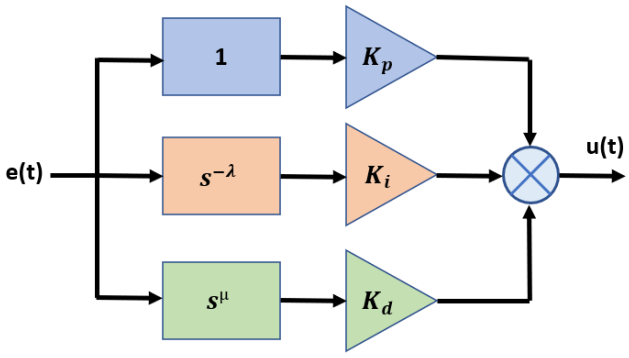


Fig. 12: Basic structure of FOPID controller [23]

2.3.2 Modeling in Simulink:

Initially the system is modeled in Simulink. Then the tunable parameters are assigned in the FOPID block. After that ITAE based objective function is optimized using those assigned parameters in Matlab to get the optimum parameter value.

The following figure is the representation of the Simulink model:

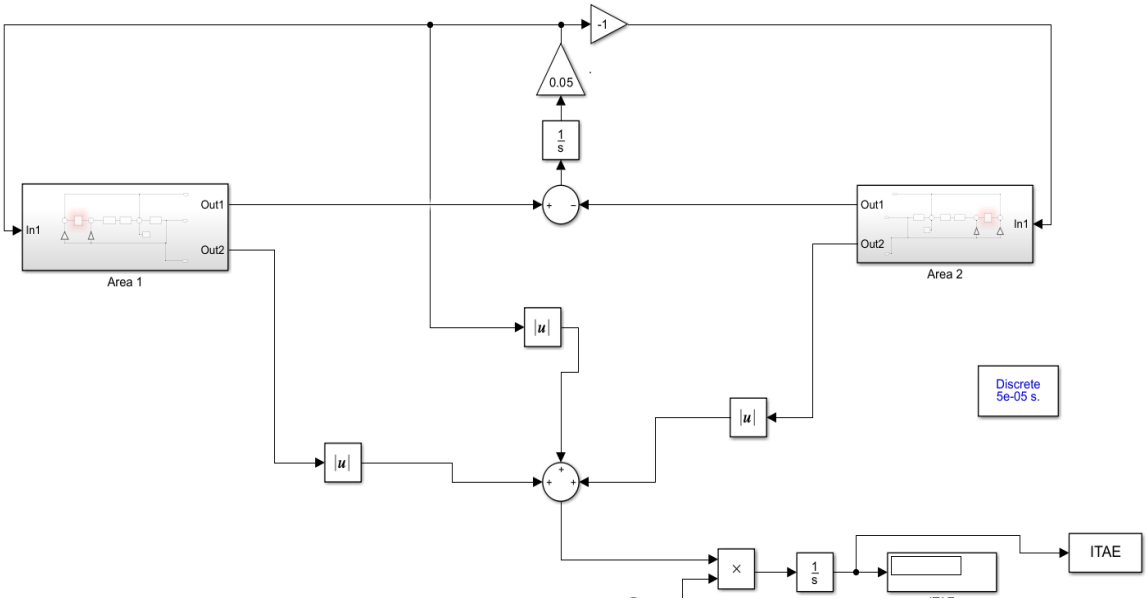


Fig. 13: Simulink modeling of the system

Now, the figures below represent the area 1 and area 2 of the above modelling:

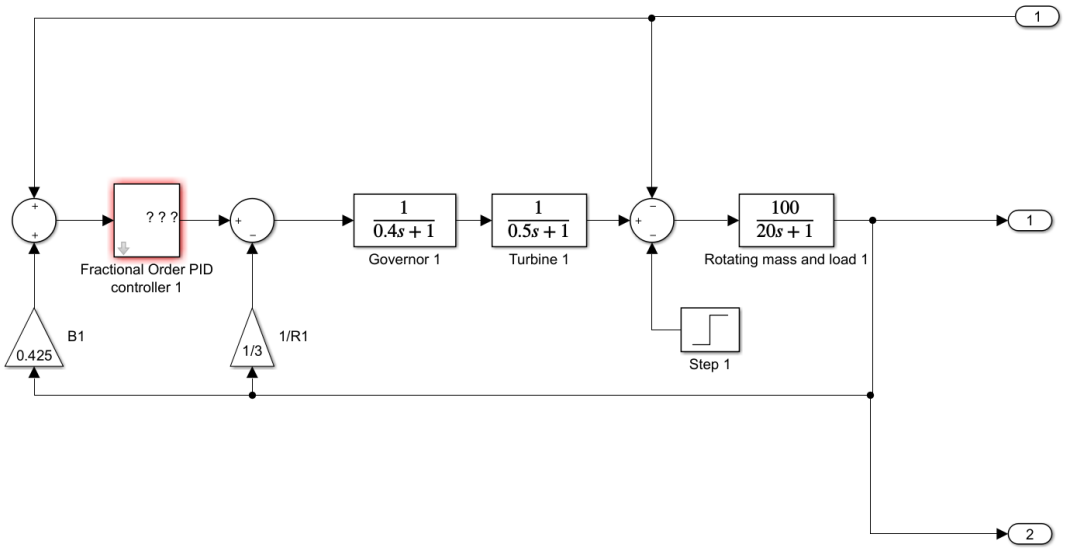


Fig. 14: Area1 in Simulink modeling of the system

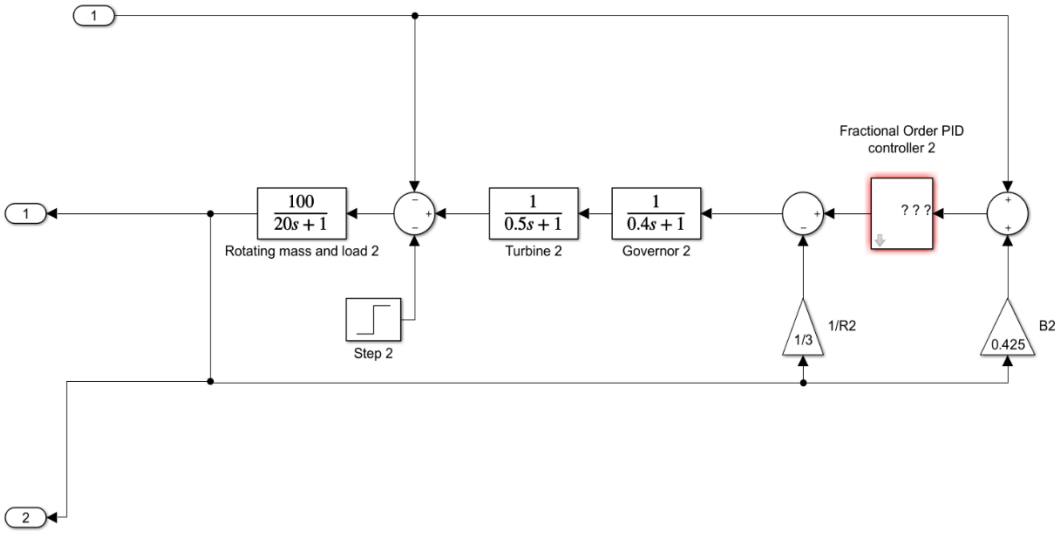


Fig. 15: Area2 in Simulink modeling of the system

Now the following figure is the FOPID controller box parameters for area1:

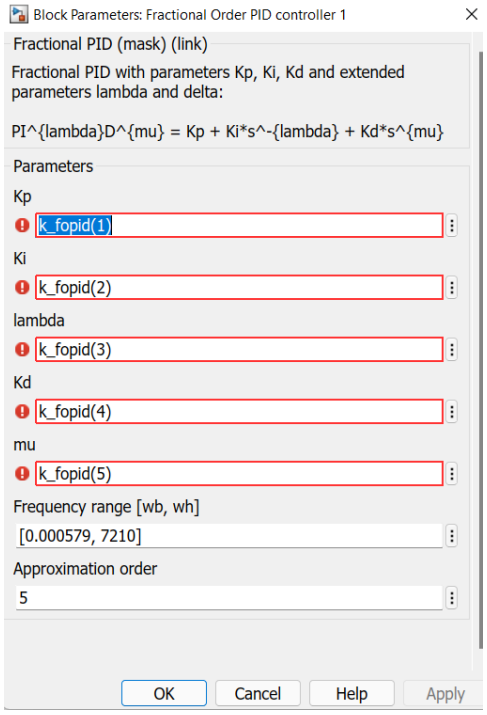


Fig. 16: FOPID controller box parameters

Here, 5 tunable parameters ($K_p, K_i, lamda, K_d \; and \; mu$) are assigned as variable so that the values of those parameters can be set by the optimizer. The frequency range [wb, wh] and approximation order (N) is calculated using eqn. (16-18). The controller parameters for area 2 is exactly similar.

2.4 Neural Network (NN) incorporation into the system:

Due to the dynamic nature of load, we need to pay heed to the time constraints. Widely-used optimizers are able to significantly reduce the ITAE value, however, they are averaging 80-90 seconds per SLPs. To reduce the execution time, the application of neural networks is proposed.

A. Data description and preparation:

By extensive executions and simulations, it is found that for the system under study, EGBO, GBO, MPA, and MTDE algorithms performed better than the 30-odd optimizers tested on this problem. For this reason, four separate datasets are created, with the four optimizers mentioned above. For the simulation, SLPs from 0.005 pu to 0.25 pu are taken for the two areas, which lead to 2500 simulated samples per optimizers, totaling 10000 samples collected in around 278 hours.

In this study, step changes in load demand in two areas are considered as the input feature and the PID parameters of the controllers are considered as the output feature. Each of the datasets contain 168,342 parameters and for each dataset, the data is split 80-10-10 to train, validation, and test sets.

B. Model preparation:

For training, ANN (Artificial Neural Network) is chosen due to its worldwide popularity and ease of use. For the four datasets, the number of layers and activation functions have been kept the same for fair comparisons. The hyperparameters of the models have been set manually after extensive trial and error runs. All the trainable layers except the output layer are L2 regularized (0.0001) to avoid over-fitting. Dropouts are not introduced to reduce the unpredictability of the models. Other general parameters are:

- Optimizer: Adam
- Learning Rate: 0.0001
- Loss functions: Mean Squared Error, Cosine Similarity
- Batch Size: 32
- Epoch Determiner: Early Stopping (Patience: 250)

Table 1: Layer Based Hyperparameters

Layer Type	No. of Units	Activation Function
Batch Normalization		
Fully Connceted	144	ReLU
Fully Connceted	144	ReLU
Fully Connceted	144	ReLU
Fully Connceted	144	ReLU
Fully Connceted	144	ReLU
Fully Connceted	144	ReLU

Fully Connceted	144	ReLU
Fully Connceted	144	ReLU
Fully Connceted	144	ReLU
Fully Connceted	6	Linear

Chapter 3: Results & Outcomes

3.1 Optimization of PID Controller incorporated Two Area AGC:

For a comprehensive comparative analysis, along with EGBO, five other optimization algorithms, namely GBO GWO, PSO, SCA, and ChOA are adopted in the PID controller gain optimization process. For the sake of fair competition, the number of population or search agents for each algorithm is chosen to be 100, with a maximum of 500 iterations. All of the algorithms are initialized with random PID gain parameters. Other relevant parameters for the aforementioned algorithms are provided in appendix D.

A. Case Studies:

For the breakdown of the performances of the controller optimized by the aforementioned algorithms, the following case studies are conducted:

- Case-1: 15% constant SLP in both areas
- Case-2: 25% constant SLP in area 1 and no SLP in area 2
- Case-3: 25% constant SLP in area 2 and no SLP in area 1
- Case-4: 20% constant SLP in area 1 and 10% constant SLP in area 2
- Case-5: 20% constant SLP in area 2 and 10% constant SLP in area 1

i. Case-1:

Table 2: Optimal Values of the Controller for Case-1

			Controller 1 Parameters			Controller 2 Parameters		
Algorithms	**Best Optimal ITAE**	**Execution Time (sec)**	K_{p1}	K_{i1}	K_{d1}	K_{p2}	K_{i2}	K_{d2}
ChOA	0.2344	94.3089	-14.9522	-41.6957	-5.6072	-14.7309	-43.1649	-5.618
EGBO	**0.2292**	**74.8221**	-15.1838	-43.5993	-5.7641	-15.1738	-45	-5.761
GBO	0.2293	76.1791	-15.2024	-43.5671	-5.7899	-15.178	-44.9967	-5.7857
GWO	0.2298	95.7718	-15.1871	-43.7646	-5.7986	-15.0911	-45	-5.7799
PSO	0.2296	81.6852	-15.2116	-43.837	-5.8098	-15.1177	-45	-5.7709
SCA	0.2359	97.6668	-16	-45	-6.103	-16	-45	-5.9294

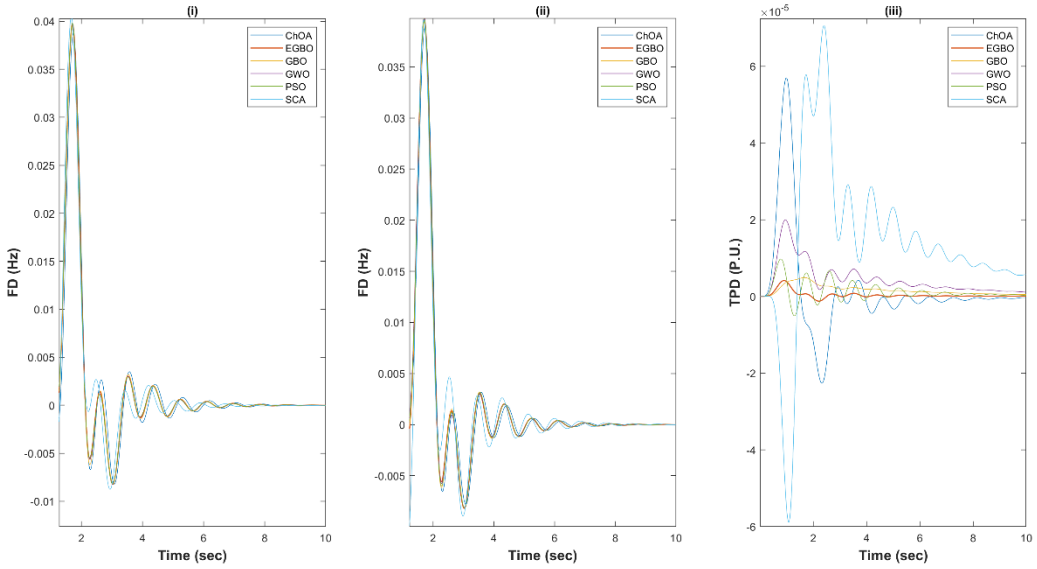


Fig. 17: (i) FD in area 1 (ii) FD in area 2 (iii) TPD for case-1

From table 2, it is evident that EGBO provides the best optimal ITAE while taking the least amount of time. In terms of both best ITAE and execution time, EGBO edges over second-placed GBO and third-placed PSO. In fig. 16, although the algorithms show a similar trend in terms of FD, for the case of TPD, it is evident that EGBO is by far the superior algorithm, boasting very little overshoot and settling in around 6-second.

i. Case-2:

Table 3: Optimal Values of the Controllers for Case-2

			Controller 1 Parameters			Controller 2 Parameters		
Algorithms	**Best Optimal ITAE**	**Execution Time (sec)**	**Kp1**	**Ki1**	**Kd1**	**Kp2**	**Ki2**	**Kd2**
ChOA	0.9384	**95.4597**	-10.9494	-45	-4.7316	-16	-45	-5.0483
EGBO	**0.9378**	98.3608	-10.8375	-45	-4.7069	-16	-39.1146	-4.8704
GBO	**0.9378**	100.6737	-10.8375	-45	-4.7069	-16	-39.1148	-4.8702
GWO	0.9379	96.9174	-10.8392	-45	-4.7091	-15.955	-41.607	-4.9248
PSO	**0.9378**	105.1054	-10.8375	-45	-4.7069	-16	-39.1335	-4.8708
SCA	0.939	100.0173	-10.8832	-45	-4.7175	-15.0735	-21.4369	-5.774

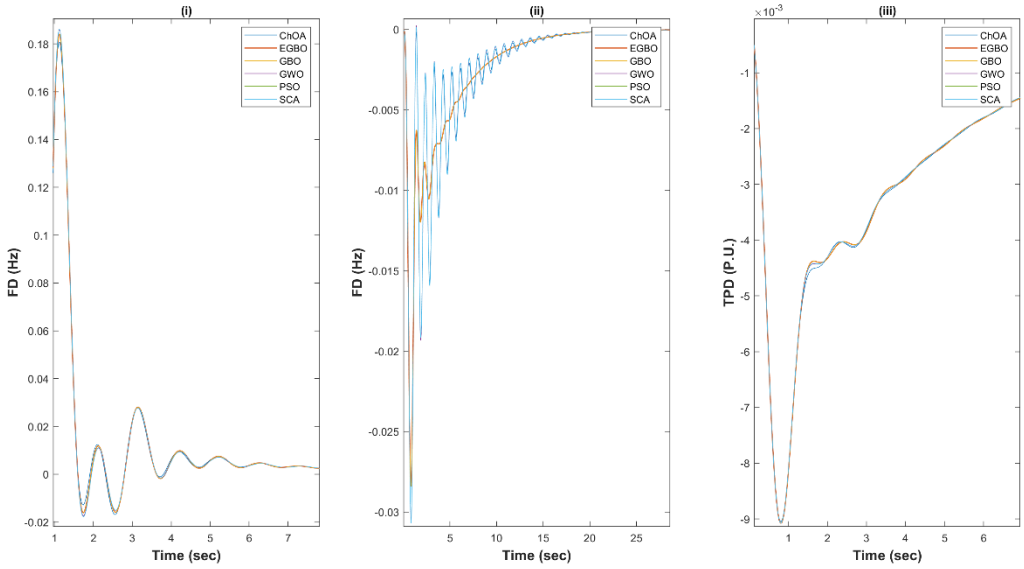


Fig. 18: (i) FD in area 1 (ii) FD in area 2 (iii) TPD for case-2

For case-2, EGBO, GBO, and PSO deliver the same optimal ITAE of 0.9378, while GWO just falls short with an ITAE of 0.9379. Some of the controller parameters obtained from EGBO, GBO, and PSO are almost identical, which resonate with the simulation of FD in area 1 and TPD. However, SLP applied on only area 1 seems to have an adverse effect on FD of area 2. The algorithms seem to find it difficult to minimize the oscillation and to minimize the settling time, albeit EGBO, which provides a smoother response with a faster settling time.

i. Case-3:

Table 4: Optimal Values of the Controllers for Case-3

			Controller 1 Parameters			Controller 2 Parameters		
Algorithms	**Best Optimal ITAE**	**Execution Time (sec)**	**Kp1**	**Ki1**	**Kd1**	**Kp2**	**Ki2**	**Kd2**
ChOA	0.9507	**97.3061**	-16	-43.5345	-5.0029	-10.7151	-45	-4.6522
EGBO	**0.95**	99.9523	-16	-45	-4.9366	-10.6501	-45	-4.6468
GBO	**0.95**	105.6416	-16	-45	-4.9366	-10.6501	-45	-4.6468
GWO	0.9501	103.3018	-16	-45	-4.9314	-10.6555	-45	-4.6481
PSO	**0.95**	113.7967	-16	-45	-4.9366	-10.6501	-45	-4.6468
SCA	0.9528	101.2585	-11.3121	-45	-5.0956	-10.6818	-45	-4.6705

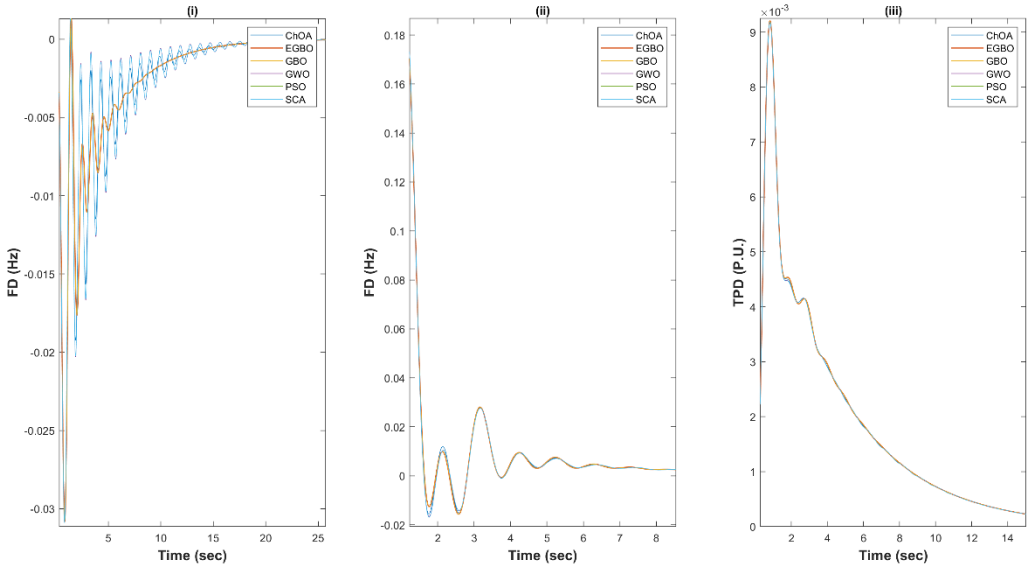


Fig. 19: (i) FD in area 1 (ii) FD in area 2 (iii) TPD for case-3

For case-3, the aforesaid effect of only one area load perturbation on the FD and TPD simulations as explained for case-2 can be seen, as well as similar trends for ITAE and PID gains. For further discussions on case-2 and case-3, the execution time can be accounted for. For both cases, ChOA exhibits the smallest execution times, but not the best optimal ITAEs. Among the algorithms that provide the best optimal ITAEs, EGBO converges to the solution, the quickest.

i. Case-4:

Table 5: Optimal Values of the Controllers for Case-4

			Controller 1 Parameters			Controller 2 Parameters		
Algorithms	**Best Optimal ITAE**	**Execution Time (sec)**	**Kp1**	**Ki1**	**Kd1**	**Kp2**	**Ki2**	**Kd2**
ChOA	0.2805	93.7092	-15.64	-45	-5.931	-16	-23.8575	-5.6133
EGBO	**0.2735**	**82.1557**	-15.292	-45	-5.856	-15.3763	-23.4416	-5.4057
GBO	0.2736	82.6738	-15.309	-45	-5.8478	-15.4755	-23.4111	-5.382
GWO	0.2739	92.401	-15.2249	-45	-5.8318	-15.4385	-23.404	-5.3673
PSO	0.2737	85.5741	-15.2937	-45	-5.863	-15.425	-23.4928	-5.4229
SCA	0.2821	101.5411	-16	-45	-5.9795	-15.478	-22.6949	-5.3067

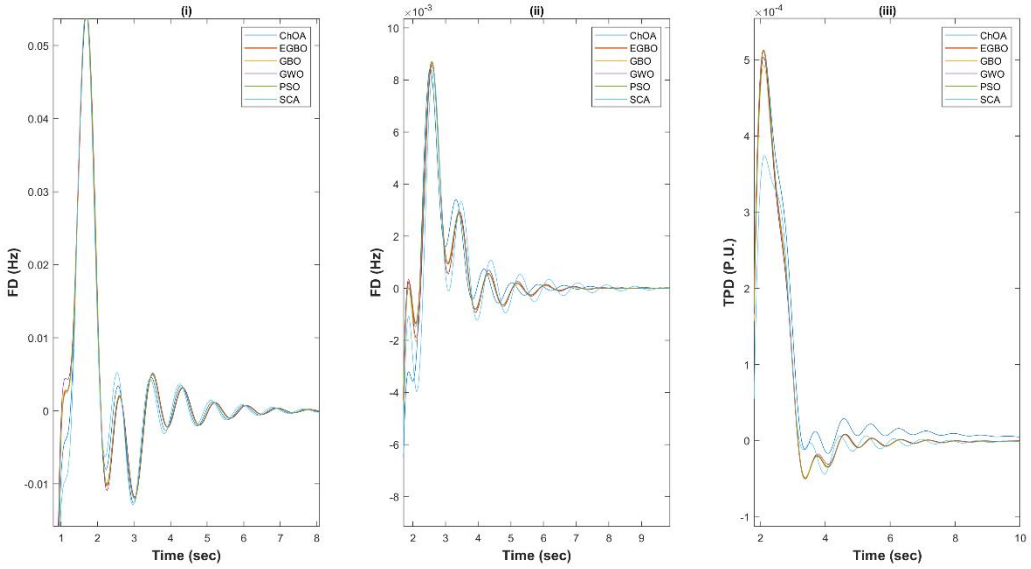


Fig. 20: (i) FD in area 1 (ii) FD in area 2 (iii) TPD for case-4

From table 5, it is clear that EGBO possesses the best optimal ITAE value and execution time among the optimizers, surpassing its parent GBO by a slight margin. SCA shows better overshoot in the case of TPD as seen in figure 5, however, it requires a longer execution time.

i. Case-5:

Table 6: Optimal Values of the Controllers for Case-5

			Controller 1 Parameters			Controller 2 Parameters		
Algorithms	**Best Optimal ITAE**	**Execution Time (sec)**	**Kp1**	**Ki1**	**Kd1**	**Kp2**	**Ki2**	**Kd2**
ChOA	0.285	92.4471	-15.0874	-22.0303	-5.4886	-14.6593	-45	-5.7556
EGBO	**0.2772**	**78.7314**	-15.1578	-21.3754	-5.3152	-15.0066	-45	-5.3152
GBO	0.2773	79.3398	-15.1262	-21.3442	-5.3058	-15.0664	-44.9952	-5.7503
GWO	0.2775	93.1188	-15.1036	-21.3543	-5.3137	-15.0168	-45	-5.7608
PSO	0.2773	84.7991	-15.1633	-21.3759	-5.3172	-15.0122	-45	-5.7557
SCA	0.2892	97.3272	-16	-22.3174	-5.519	-15.3498	-45	-5.6841

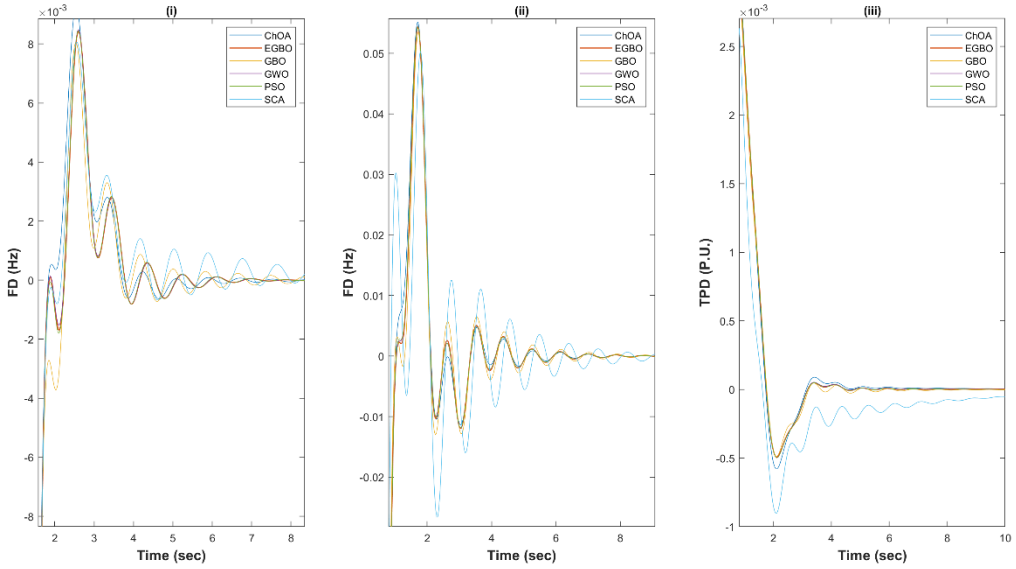


Fig. 21: (i) FD in area 1 (ii) FD in area 2 (iii) TPD for case-5

For case-5, the superiority of EGBO is crystal clear, which is evident from fig. 20 as both FD and TPD simulations exhibit less jitter and faster settling time. Figure 6 is the justification of the lowest optimal ITAE value provided by EGBO among the algorithms. EGBO is also the fastest to converge to such value.

B. Convergence Comparison of the Optimizers:

Table 7: Comparison of the Controllers for Case-5 after 50 iterations

Algorithms	Fitness Value	Optimal ITAE	Execution Time (sec)
ChOA	0.3861	0.285	11.1407
EGBO	**0.2776**	**0.2772**	**8.4618**
GBO	0.2779	0.2773	8.9891
GWO	0.3007	0.2775	10.1287
PSO	0.2793	0.2773	8.8641
SCA	0.3943	0.2892	11.0746

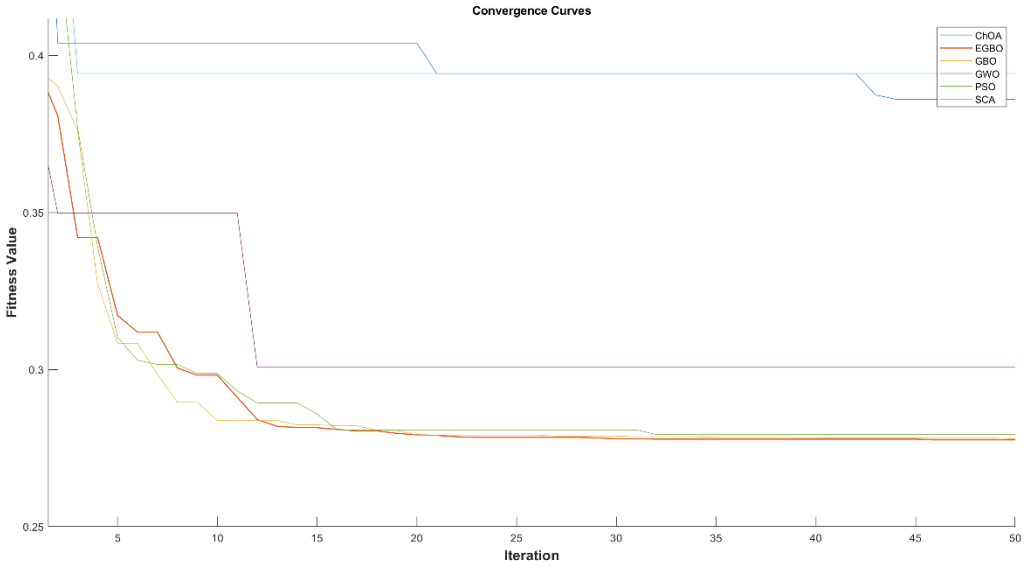


Fig. 22: Convergence curves of the algorithms under comparison

As explained earlier, it is vital to know exactly how fast the algorithm traverses the search space to get to the optimal fitness value. For an industry-wide application, the optimized controllers should be able to cope up with the dynamic load, meaning, coping with the variation in SLP after a certain amount of time. For demonstrating the performance comparison, the first 50 iterations of case-5 are taken, along with the fitness value after 50 iterations, time taken for the algorithms to obtain such fitness value, and the optimal ITAEs found in table 6. From fig. 21, it can be seen that, although GBO performs well for the first few iterations, EGBO provides the lowest ITAE, which is apparent in table 7. EGBO and GBO take around 8 seconds for finding the final fitness value while being 0.14% and 0.22% closer to the optimal ITAE value of each algorithm, respectively.

C. Comparative Statistical Analysis:

This section represents statistical analysis data of all the cases for each algorithm for better comparative study.

Table 8: Statistical study of the algorithms for the studied five cases

Cases	Statistical Measures	ChOA	EGBO	GBO	GWO	PSO	SCA
Case-I	Mean	0.2358	**0.2296**	0.23	0.2302	0.2303	0.2471
	Best	0.2344	**0.2292**	0.2293	0.2298	0.2296	0.2359
	Worst	0.2383	**0.2301**	0.2303	0.2304	0.2321	0.2673
Case-II	Mean	0.9395	**0.9378**	0.9379	0.9384	0.9379	0.94

	Best	0.9384	**0.9378**	**0.9378**	0.9379	**0.9378**	0.939
	Worst	0.9408	**0.9378**	0.9391	0.9393	0.9391	0.9416
Case-III	Mean	0.9544	**0.95**	**0.95**	0.9506	**0.95**	0.9575
	Best	0.9507	**0.95**	**0.95**	0.9501	**0.95**	0.9528
	Worst	0.9582	**0.95**	0.9501	0.9532	0.9501	0.9641
Case-IV	Mean	0.2896	**0.2737**	0.2754	0.2741	0.2738	0.3021
	Best	0.2805	**0.2735**	0.2736	0.2739	0.2737	0.2821
	Worst	0.303	**0.2738**	0.2979	0.2743	0.2748	0.3222
Case-V	Mean	0.2961	**0.2773**	0.2792	0.2778	0.2784	0.3077
	Best	0.285	**0.2772**	0.2773	0.2775	0.2773	0.2892
	Worst	0.3088	**0.2775**	0.3043	0.278	0.3032	0.3215

For each case, the algorithms are run for 30 times. The case study section of the paper is comprised of the best cases found while simulating the algorithms. For the comparative statistical analysis, all of the 30 runs are considered and the best, worst and mean values of each of the algorithms for the five cases are calculated and presented in table 8. EGBO is the superior of the studied algorithms, performing efficiently all across the board. For case-2 and case-3, EGBO provides the same value for the considered three statistical measures, showcasing the robustness of the optimizer.

3.2 Optimization of FOPID Controller incorporated Two Area AGC:

As EGBO gives superior result in the previous section, most priority is given to EGBO in terms of optimization algorithm in this section. Basically, in this section, ITAE based objective function is optimized by EGBO in order to tune the FOPID controllers. The optimization is run for 250 iterations and 30 population size. The SLPs for both areas are kept 10%. The other parameters for optimization algorithm are mentioned in appendix D.

The following table represents the results of the FOPID controller for the above-mentioned scenario:

Table 9: Optimal Values of the FOPID Controller for the mentioned scenario

Iterations	Exe. Time (s)	ITAE	K_{p1}	K_{i1}	λ_1	K_{d1}	μ_1	K_{p2}	K_{i2}	λ_2	K_{d2}	μ_2
250	21855.722	**0.0835**	-17.9989	-30.062	0.9999	-12.9772	0.9999	-17.9997	-30.0626	0.9999	-12.9782	0.9999

The convergence curve for the EGBO FOPID controller is as follows:

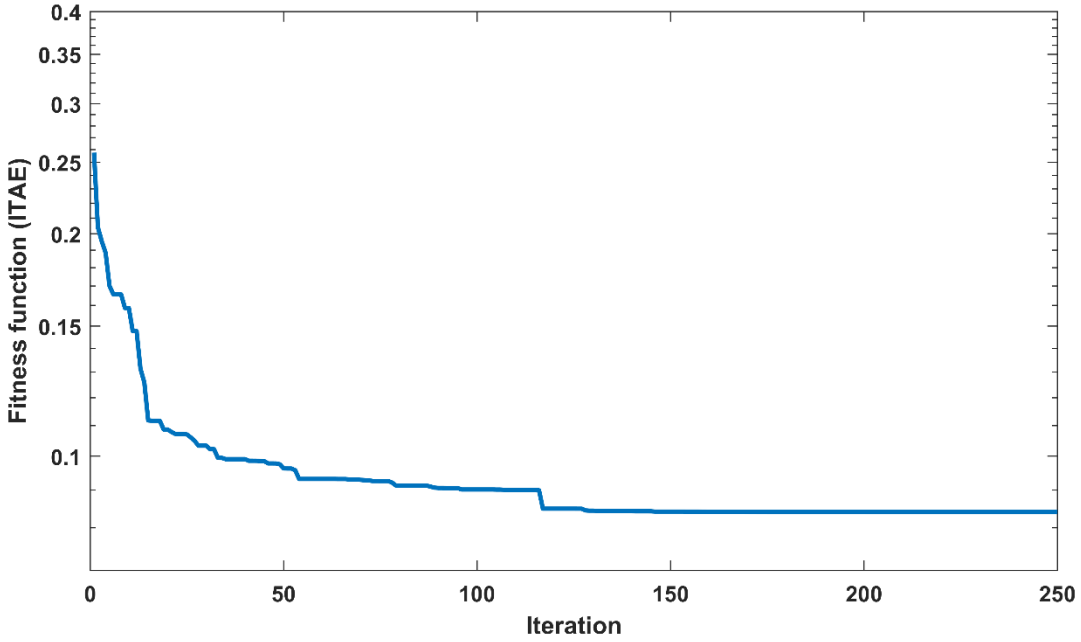


Fig. 23: Convergence curve of the EGBO FOPID controller

From fig. 22, it can be clearly noticed that after 130 iterations, the curve becomes completely flat. And the ITAE is 0.0835 which exceptionally good result for this kind of scenario.

The following table shows the comparison between EGBO based FOPID controller and EGBO, GBO, MPA & MTDE based PID controller results for the same model with 10% SLPs on the both areas. It can be seen that the FOPID-EGBO result is superior compared to others in terms of ITAE.

Table 10: Comparison of the controllers' results

Controller	ITAE	Exe. Time (s)
PID-EGBO	0.15428	74.881
PID-GBO	0.15487	88.288
PID-MPA	0.15431	408.409
PID-MTDE	0.15458	94.068
FOPID-EGBO	**0.08348**	21855.722

3.3 Neural Network Prediction Based Approach for Two Area AGC:

A. Error Metrics:

The models have been tested based on three different error parameters - RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) and the coefficient of determination, R^2.

B. Experimental Results:

Table 11: Experimental results of 4 optimizers

R^2	RMSE	Epochs	MSE	MAE	Optimizer
0.9289	0.4107	2634	0.2149	0.1929	EGBO
0.9639	0.3352	1487	0.1644	0.1511	GBO
0.9748	0.2606	1000	0.1206	0.1110	MTDE
0.9744	0.2531	1750	0.1154	0.1009	MPA

Table 12: Comparison of neural network based controllers and optimized controllers for w_1,w_2 = 0.1

Controller	ITAE	Exe. Time (s)
PID-EGBO	0.1543	74.8808
PID-GBO	0.1549	88.2879
PID-MPA	0.1544	408.4092
PID-MTDE	0.1546	94.0683
ANN-EGBO	0.1577	**0.1591**
ANN-GBO	0.1558	**0.1330**
ANN-MPA	0.1560	**0.1705**
ANN-MTDE	0.1577	**0.1623**

C. Discussion:

It is clear from the experimental results table that, the MPA model possesses less error in terms of RMSE and MAE, which signifies the ability of the model to update the weights meticulously, catering to the provided outputs. However, MTDE yields the highest coefficient of determination, with a staggering 97.48% of the data fitted into the approximated regression model. Before applying ANNs, the primary goal is to reduce the execution to the extent where the controllers will be able to perform real-time. From the comparison table, it is safe to say that the goal is reached as the ANN-based controllers could predict the PID gain parameters in real-time, without sacrificing vital amount of ITAE value.

Chapter 4: Future Works & Conclusion

4.1 Future Works:

Although the FOPID controller gives very good result in terms of ITAE [13], the execution time is extremely high which should be reduced. That's why reducing the execution time of the FOPID controller is one of the future works. It can be done by approximating the integer order of the fractional order transfer function and executing the whole program in Matlab [8].

Tilt-Integral Derivative (TID) controllers and Fuzzy logic controller can be tried as new controllers in this work to compare the results with existing ones.

Weather-based day ahead load forecasting can be done as future works by applying various Recurrent Neural Network (RNN) architectures such as LSTM and GRU.

4.2 Conclusion:

By improving PID and FOPID controllers using EGBO algorithm and a model predictive method utilizing ANN, this work presents an effective and optimum solution to the LFC problem. This project is separated into three sections: PID controller, FOPID controller, and ANN-based controller. So, if a good tuning of PID controllers in a two-area AGC is required, our study can assist with a good ITAE optimization of the tunable parameters. On the other hand, if exceptionally low errored control of the two area AGC is required, this work can also assist by creating extremely low ITAE utilizing its FOPID controller optimization portion. Again, if tweaking a PID controller with nearly no waiting time is required, the predictive technique is the only option. In this situation, an ANN-based model may almost instantly anticipate the adjustable parameters.

References

[1] H. Saadat, Power System Analysis, New York: McGraw-Hill, 2002.

[2] Elliot Group, "Electronic Governor Upgrades," Elliot-Turbo, 2014. [Online]. Available: www.elliott-turbo.com/ElectronicGovernor. [Accessed May 2022].

[3] K. &. J. Nagrath, Modern Power System Analysis, Tata McGraw- Hill Education, 2003.

[4] K. a. K. D. Oprzędkiewicz, "A tuning of a fractional order PID controller with the use of particle swarm optimization method," in *International Conference on Artificial Intelligence and Soft Computing. Springer, Cham*, 2017.

[5] A. e. a. Tepljakov, "FOPID controllers and their industrial applications: A survey of recent results," *IFAC-PapersOnLine,* vol. 51.4, pp. 25-30, 2018.

[6] Wikipedia contributors, "PID Controller," Wikipedia, 30 March 2022. [Online]. Available: en.wikipedia.org/wiki/PID_controller. [Accessed May 2022].

[7] Explained-PID , "PID Controller Explained," PID Explained, 23 July 2020. [Online]. Available: pidexplained.com/pid-controller-explained. [Accessed May 2022].

[8] The Math Works, Inc., "The Math Works, Inc. MATLAB. Version 2018b," The Math Works, Inc., 2018. [Online]. Available: www.mathworks.com/. [Accessed 2022].

[9] K. W. M. a. E. G. Oprzędkiewicz, "An estimation of accuracy of Oustaloup approximation," in *International Conference on Automation. Springer, Cham*, 2016.

[10] F. Merrikh-Bayat, "Rules for selecting the parameters of Oustaloup recursive approximation for the simulation of linear feedback systems containing PIλDμ controller," *Communications in Nonlinear Science and Numerical Simulation,* vol. 17.4, pp. 1852-1861, 2012.

[11] O. ÖZKARACA, "A review on usage of optimization methods in geothermal power generation," *Mugla Journal of Science and Technology,* vol. 4.1, pp. 130-136, 2018.

[12] N. L. S. a. M. P. Paliwal, "Application of grey wolf optimization algorithm for load frequency control in multi-source single area power system," *Evolutionary Intelligence,* pp. 1-22, 2020.

[13] F. G. Martins, "Tuning PID controllers using the ITAE criterion," *International Journal of Engineering Education,* vol. 21.5, p. 867, 2005.

[14] A. F. B. a. M. F. Safari, "A load frequency control using a PSO-based ANN for micro-grids in the presence of electric vehicles," *International Journal of Ambient Energy,* vol. 42.6, pp. 688-700, 2021.

[15] M. H. Khooban, "A new load frequency control strategy for micro-grids with considering electrical vehicles," *Electric Power Systems Research,* vol. 143, pp. 585-598, 2017.

[16] S. Mirjalili, "SCA: a sine cosine algorithm for solving optimization problems," *Knowledge-based systems,* vol. 96, pp. 120-133, 2016.

[17] D. P. K. R. a. S. B. Guha, "Load frequency control of interconnected power system using grey wolf optimization," *Swarm and Evolutionary Computation,* vol. 27, pp. 97-115, 2016.

[18] M. a. M. R. M. Khishe, "Chimp optimization algorithm," *Expert systems with applications,* vol. 149, p. 113338, 2020.

[19] I. O. B.-H. a. X. C. Ahmadianfar, "Gradient-based optimizer: A new metaheuristic optimization algorithm," *Information Sciences,* vol. 540, pp. 131-159, 2020.

[20] I. Ahmadianfar, "Gradient-based optimization with ranking mechanisms for parameter identification of photovoltaic systems," *Energy Reports,* vol. 7, pp. 3979-3997, 2021.

[21] A. Faramarzi, "Marine Predators Algorithm: A nature-inspired metaheuristic," *Expert Systems with Applications,* vol. 152, p. 113377, 2020.

[22] M. H. e. a. Nadimi-Shahraki, "MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems," *Applied Soft Computing,* vol. 97, p. 106761, 2020.

[23] C. a. K. R. S. Bhavanisankar, "An adaptive technique to control the load frequency of hybrid distributed generation systems," *Soft Computing,* vol. 23.23, pp. 12385-12400, 2019.

[24] Investopedia, "Neural Network Definition," Investopedia.com, 8 December 2021. [Online]. [Accessed May 2022].

[25] D.-A. T. U. a. S. H. Clevert, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint,* vol. 1511.07289, 2015.

[26] R. N. a. P. R. K. R. Rao, "PSO based tuning of PID controller for a Load frequency control in two area power system," *International Journal of Engineering Research and Applications (IJERA),* vol. 1.3, pp. 1499-1505, 2015.

[27] S. T. P. a. V. P. S. Dewangan, "Design and performance analysis of elephant herding optimization based controller for load frequency control in thermal interconnected power system," *Optimal Control Applications and Methods,* vol. 42.1, pp. 144-159, 2021.

[28] H. e. a. Parvaneh, "Load frequency control of a multi-area power system by optimum designing of frequency-based PID controller using seeker optimization algorithm," in *2016 6th Conference on Thermal Power Plants (CTPP)*, 2016.

[29] B. e. a. Vedik, "Renewable Energy-Based Load Frequency Stabilization of Interconnected Power Systems Using Quasi-Oppositional Dragonfly Algorithm," *Journal of Control, Automation and Electrical Systems,* vol. 32.1, pp. 227-243, 2021.

[30] S. S. G. a. A. Y. Mishra, "Design and application of controller based on sine-cosine algorithm for load frequency control of power system," in *International Conference on Intelligent Systems Design and Applications. Springer, Cham*, 2018.

[31] A. X. R. e. a. Irudayaraj, "A Matignon's theorem based stability analysis of hybrid power system for automatic load frequency control using atom search optimized FOPID controller," *IEEE Access,* vol. 8, pp. 168751-168772, 2020.

[32] N. e. a. Bayati, "Fopid design for load-frequency control using genetic algorithm," *Sci. Int ,* vol. 27.4, pp. 3089-3094, 2015.

[33] R. a. A. S. Kumar, "Parameter identification for load frequency control using fuzzy FOPID in power system," *COMPEL-The international journal for computation and mathematics in electrical and electronic engineering,* 2021.

[34] I. a. S. D. Pan, "Fractional-order load-frequency control of interconnected power systems using chaotic multi-objective optimization," *Applied Soft Computing,* vol. 29, pp. 328-344, 2015.

[35] R. a. P. K. Shankar, Power system stability and control II, New York: McGraw-Hill Books, 1994.

[36] D. e. a. Das, "Automatic generation control of a hydrothermal system with new area control error considering generation rate constraint," *Electric Machines & Power Systems,* vol. 18.6, pp. 461-471, 1990.

[37] Y. R. K. a. G. N. Hain, "Identification-based power unit model for load-frequency control purposes," *IEEE Transactions on Power Systems,* vol. 15.4, pp. 1313-1321, 2000.

[38] Z. M. a. Y. L. A.-M. Al-Hamouz, "Variable structure load frequency controllers for multiarea power systems," *International Journal of Electrical Power & Energy Systems,* vol. 15.5, pp. 293-300, 1993.

[39] D. K. R. U. a. O. P. M. Chaturvedi, "Adaptive polar fuzzy logic based load frequency controller," *International Journal of Electrical Power & Energy Systems,* vol. 66, pp. 154-159, 2015.

[40] M. M. A.-R. a. M. A. Zribi, "Adaptive decentralized load frequency control of multi-area power systems," *International Journal of Electrical Power & Energy Systems,* vol. 27.8, pp. 575-583, 2005.

[41] L. C. e. a. Saikia, "Automatic generation control of a multi area hydrothermal system using reinforced learning neural network controller," *International Journal of Electrical Power & Energy Systems,* vol. 33.4, pp. 1101-1108, 2011.

[42] L. e. a. Jiang, "Delay-dependent stability for load frequency control with constant and time-varying delays," *IEEE Transactions on Power systems,* vol. 27.2, pp. 932-941, 2011.

[43] H. X. W. a. J. X. Li, "Adaptive event-triggered load frequency control for interconnected microgrids by observer-based sliding mode control," *IEEE access,* vol. 7, pp. 68271-68280, 3019.

[44] I. J.-S. K. a. H. S. Rosyiana Fitri, "High-gain disturbance observer-based robust load frequency control of power systems with multiple areas," *Energies,* vol. 10.5, p. 595, 2017.

[45] M. a. Y. S. M. Azzam, "Robust controller design for automatic generation control based on Q-parameterization," *Energy conversion and management,* vol. 43.13, pp. 1663-1673, 2002.

[46] P. a. M. R. Ojaghi, "LMI-based robust predictive load frequency control for power systems with communication delays," *IEEE Transactions on Power Systems,* vol. 32.5, pp. 4091-4100, 2017.

[47] N. S. S. H. L. Z. A. Demiroren, "Automatic generation control by using ANN technique," *Electric Power Components and Systems,* vol. 29.10, pp. 883-896, 2001.

[48] A. Abdennour, "Adaptive optimal gain scheduling for the load frequency control problem," *Electric Power Components and Systems,* vol. 30.1, pp. 45-56, 2002.

[49] I. a. E. Ç. Kocaarslan, "Fuzzy logic controller in interconnected electrical power systems for load-frequency control," *International Journal of Electrical Power & Energy Systems,* vol. 27.8, pp. 542-549, 2005.

[50] C.-F. a. C.-F. L. Juang, "Load-frequency control by hybrid evolutionary fuzzy PI controller," *IEE Proceedings-generation, transmission and distribution,* vol. 153.2, pp. 196-204, 2006.

[51] R. a. I. A. C. Arivoli, "CPSO based LFC for a two-area power system with GDB and GRC nonlinearities interconnected through TCPS in series with the tie-line," *Int. J. Comput. Applications,* vol. 38.7, pp. 1-10, 2012.

[52] J. a. F. A.-B. Talaq, "Adaptive fuzzy gain scheduling for load frequency control," *IEEE Transactions on power systems,* vol. 14.1, pp. 145-150, 1999.

[53] D. e. a. Xu, ""A novel adaptive neural network constrained control for a multi-area interconnected power system with hybrid energy storage," *IEEE Transactions on Industrial Electronics,* vol. 65.8, pp. 6625-6634, 2017.

[54] D. K. P. S. S. a. P. K. K. Chaturvedi, "Load frequency control: a generalised neural network approach," *International Journal of Electrical Power & Energy Systems,* vol. 21.6, pp. 405-415, 1999.

[55] A. H. L. Z. a. N. S. S. Demiroren, "The application of ANN technique to load-frequency control for three-area power system," *IEEE Porto Power Tech Proceedings (Cat. No. 01EX502),* vol. 2, 2001.

[56] University of Arkansas at Little Rock, "Book: Introduction to Control Systems (Iqbal)," 27 November 2020. [Online]. Available: https://eng.libretexts.org/@go/page/24380. [Accessed May 2022].