People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University M'Hamed BOUGARA - Boumerdes

Institute of Electrical and Electronic Engineering

Department of Power and Control

Final Year Project Report Presented in Partial Fulfillment of the Requirements of the Degree of:

'Master'

In Electrical and Electronic Engineering

Option: **Control Engineering**

Title:

Optimization Techniques Based PID Tunning for AVR System: Comparative Study

---

Presented By:

- **RECHID Mohammed Sadek**
- **DJEDI Lahcen**

Supervisor:

Dr. F. Recioui

Abstract

This work presents a several metaheuristic methods employed to enhance the capability of traditional techniques tuning of a Proportional-Integral-Derivative (PID) controller for an Automatic Voltage Regulator (AVR) system. The presented approaches referred to as Particle Swarm optimization (PSO) algorithm, Cuckoo Search optimization (CSO) algorithm, Moth Flame optimization (MFO) algorithm, Water Cycle optimization (WCO) algorithm, Teaching-Learning Based optimization (TLBO) algorithm and Hill Climbing optimization (HCO) algorithm. In order to achieve optimal transient response and improved stability of the considered AVR system, a conventional and modified objective functions are employed to obtain optimized PID controller gains. After that, the step response compared with some approaches in literature to show the superiority of our optimized PID controllers, and the root locus, bode plot, robustness and disturbance rejection ability analysis are performed to test the stability of the optimized AVR system. According to the comparison and analysis results, the proposed optimization algorithms based PID controller improve the tracking behaviors of the AVR system, making it suitable for synchronous generator terminal voltage stability.

Contents

List of Figures

List of Tables

General Introduction

One of the major control challenges of power systems with a direct relationship to grid security and reliability is keeping a constant voltage level under various circumstances in an electric power network. When the grid voltage level fluctuates, it may cause significant changes in the system dynamics, which can result in shorter system component lifetimes, more expensive grid operations, and in the worst-case situation the possibility of partial system collapse. Furthermore, any fluctuation in voltage from the nominal value demands the flow of reactive power, which increases line losses and as a result, leads to a loss of economy.[3]

To prevent the aforementioned issues, the supply voltage is kept constant utilizing various technologies throughout the generation, transmission, and distribution levels of the power system. Excitation control of the synchronous generator is the most used technique to maintain the voltage level steady among various voltage regulation devices, such as serial and parallel capacitor banks, synchronous compensators, tap-changing transformers, reactors, and Static VAR Compensators (SVC). The automatic voltage regulator is the most essential element of the excitation system (AVR).

The automatic voltage regulator (AVR) is widely used in electric power systems and industrial applications to provide stability and good regulation of various apparatus. The AVR's primary task is to maintain the terminal voltage by adjusting the generator exciter voltage. The AVR must regulate the generator terminal voltage at all times and under all operating conditions in order to keep it within pre-established limits. An AVR system without a controller, on the other hand, will have slow responses and may cause instability. As a result, appropriate generator regulation by an AVR system is essential to guarantee that generators remain within safe and stable conditions for normal operations and respond quickly under various types of disturbances.

Several control techniques based on optimal and robust control have been studied in the literature in the hopes of implementing and improving the dynamic response of an AVR system; however, the classical PID controller remains the most preferred controller for AVR systems due to its simple structure and robustness to variations in system parameters. Because the optimal tuning of the controller is critical for providing the desired voltage response of the AVR system, many metaheuristic optimization algorithms have been proposed for the setting of PID controller parameters in the AVR system, rather than using traditional tuning methods that fail to operate effectively under a wide range of operating conditions and perform poorly in terms of system performance.

The aim of this project is to study and develop a PID controller for AVR system. The design is presented as an optimization problem where the cost functions include mainly in time domain performance parameters. Particle Swarm Optimization (PSO), Cuckoo search Optimization (CSO), Moth Flame Optimization (MFO), Water Cycle Optimization, Teaching-Learning Based Optimization (TLBO), and Hill Climbing Optimization (HCO) were used to minimize the cost functions. Each method is referred to a specific category in the metaheuristic optimization methods.

The report is structured into four main chapters. Chapter 1 deals with power system control and the

main objective of AVR system. A linearized AVR system with a PID controller is used to measure the performance. Then, Chapter 2 discusses different proposed algorithms, namely Particle Swarm Optimization (PSO), Cuckoo search Optimization (CSO), Moth Flame Optimization (MFO), Water Cycle Optimization(WCO), Teaching Learning Based Optimization (TLBO), and Hill Climbing Optimization (HCO). Chapter 3, includes the basic concepts of proportional integral derivative (PID) controllers and their tuning process has been discussed as well. In the last chapter, the complete system is constructed in Matlab where simulation is performed. Validation of the developed AVR system and the results of the closed-loop dynamic performance of the controlled system are provided and discussed. Finally, the project is end up by a general conclusion.

Chapter 1

Automatic voltage regulator

1.1 Introduction

The quality of electrical energy is the main consumers demand in the power system. Since voltage and frequency are an important indicators for system quality, these parameters must be maintained at the desired level at any moment[1]. So most of the equipment's are designed to work in predetermined values of voltage and frequency of operation. Thus maintaining constant voltage at the output terminal of generator is essential for satisfactory mains power supply .Therefore , a small deviation in voltage can cause considerably large variation in reactive power flow and this may leads to huge power loss which consequently results in loss of economy.

The load on the system is not constant all time, it's always varying and can be affected by disturbances, such as unpredictable change in the load voltage,temperature, speed...etc. the deviation of the voltage from the nominal case generally leads to a decrease in performance and reduction in life time of the power system equipment[2]. To avoid this the automatic voltage regulator (AVR) system is used in power generation site to ensure voltage stability at the generator terminal. The main objective of AVR system is to maintain the terminal voltage of the alternator in the generating station through excitation system.

This chapter will provide a brief summary of the automatic voltage regulator (AVR) and the required to build an AVR system.

1.2 Construction of the synchronous generator

A synchronous generator (alternator) is a rotating apparatus that converts mechanical energy into electrical energy , which is usually a part of power system. It is essentially comprised of two elements: a set of field structure or rotor and an armature coil or stator in relative motion. The stator is an annular structure made up of slots and the core. Insulated coils are placed in the slots, and these coils are connected to obtain a three-phase winding. The rotor has electromagnets which are known as field poles. The rotor is placed within the stator.

For convenience and economics reasons, the winding of armature is normally located in the stationary portion of the machine, and the second is contained in rotating portion of the machine. When we energize the rotor of the machine with the help of direct current from prime mover, alternating voltages are induced in the armature coils. A three-phase machine is most commonly used in practice and

because most electric power is generated as a three-phase power.

1.3 Power System Control

Electricity generation, transmission, distribution, and consumption are the four subsystems that compose up a power system. Control systems are complex and hierarchical in nature. That is, there are numerous nested control loops in the system that govern various quantities. Control loops at lower system levels have a smaller time constant than control loops at higher system levels. The automatic voltage regulator (AVR), for example, responds in a fraction of a second or less, whereas secondary voltage control (transformer stations) functions in a fraction of a second or minutes. The total control system is complicated, but because of the decoupling, the complexity is minimized.

1.4 Transient Stability analysis

In an electrical system reliability there is a demand at synchronous generators to satisfy the load demand. At certain cases, the generator losses synchronise with the rest of the system because of that another demand is important in order to maintain the power system integrity. The high voltage transmission system connects the generation sources to the load center, which requires the power system study, where all electrical systems are connected to each other. At each occurrence of disturbances an unbalance is created between the system generation and load, so a new operation point will be established and consequently there will be voltage adjustments. The system adjustment to its new operation condition is called “transient period” and the system behavior during this period is called “dynamic performance”. It can be said that the system oscillatory response during the transient period, short after a disturbance is damped and the system goes in a definite time to a new operating condition, so the system is stable. This means that the oscillations are damped. In a power system, the instability can be shown in different ways, according to its configuration and its mode of operation, but it can also be observed without synchronism loss. Automatic devices control generators in its voltage output and frequency, in order to keep them constant according to pre-established values.[3]

The automatic devices are:

- Automatic voltage regulator
- Governor

The main objective of the automatic voltage regulator is to control the terminal voltage by adjusting the generators exciter voltage. It must keep track of the generator terminal voltage all time and under any load condition, working in order to keep the voltage within pre-established limits. Based on this it can be said that AVR also controls the reactive power generated.[3]

1.5 Reactive Power and Voltage control

The problem of maintaining voltage within the required limits is complicated because of the fact that the power system supplies power to vast number of loads and is fed from many generating units, as the loads vary the reactive power requirements of the transmission system vary as well. Since the reactive

power cannot be transmitted over a long distance, the voltage has to be maintained by special devices, the proper selection and coordination of equipment for controlling reactive power and voltage are among the major challenges of power systems engineering.[4]

The main method used to control the voltage level in the generating unit is the automatic voltage regulator (AVR) which controls the field excitation to maintain the terminal voltage of the generator constant.

1.6 Excitation System

The system that is used to provide a direct current to the synchronous generator field winding (rotor winding) ,such type of system is called as excitation system. This current is varied in order to regulate the synchronous generator terminal voltage. The excitation system is important to control the output voltage. It should be capable of responding rapidly to any disturbance to enhance transient stability. The amount of excitation required is depend upon load power factor, load current and speed of the machine. The dc field current is obtained from a separate source called an exciter. The excitation systems have taken many forms over the years of their evolution

(a) **DC excitation system:** the excitation system of this type uses DC generator as a source of excitation power and provide current to the rotor of synchronous machine through slip rings. This type of DC excitation system has slow response. For this reason, DC excitation systems are gradually disappearing.

(b) **AC excitation system:** the excitation system of this category utilize alternators as the source of the main generator power. Controlled or non-controlled rectifiers rectify the ac output of this alternator.

(c) **Static excitation system:** a portion of the AC from each phase of synchronous generator output is fed back to the field windings as DC excitations through a system of transformers rectifiers. all the component in the system are static, static rectifiers, controlled or uncontrolled .

(d) **Brushless excitation system:** A brushless excitation system is essentially an inside-out ac generator that delivers its ac voltage to the rotor of the main generator and receives its excitation from the stator of that same generator. The ac voltage of the inside-out generator is transformed to dc by a non-controlled rectifier, this way products the excitation current of the main generator.

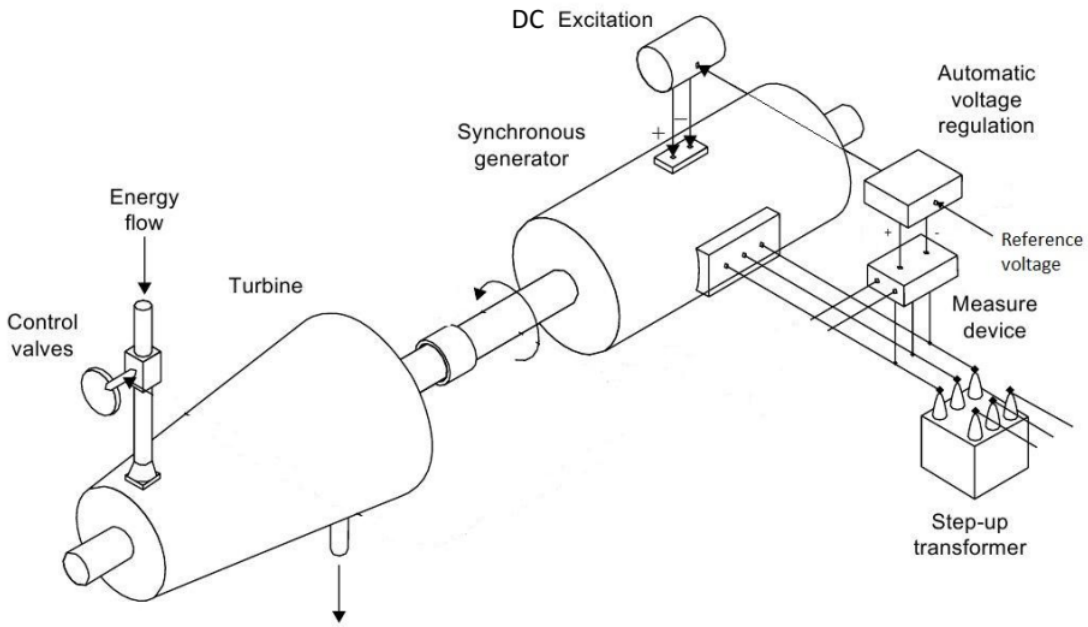


Figure 1.1: Schematic diagram of a synchronous machine with excitation system control

1.7 Automatic Voltage Regulator (AVR)

The primary function of an AVR system is to maintain the terminal voltage of the generator at a constant level through the excitation system. It sense the fluctuation in voltage and changes hem into a stable voltage. The variation in the voltage occurs because of the variation in load. An AVR is work as feedback control system that measures the output voltage of the generator and compares that output to a set point, and generates an error signal that error signal controls the excitation of the generator. As the terminal voltage of the alternator increases as the excitation current in the field winding of the generator increases[5]. the design of an AVR system is a significant and difficult task because it provides such a crucial service. A typical AVR system consists of the following components: controller, amplifier, exciter, generator and sensor.

The object that needs to be controlled in this control scheme is a synchronous generator, whose terminal voltage is measured by the sensor. An error signal, which presents the difference between the desired and the measured voltage value, is formed in the comparator. One of the main components in the AVR system that needs to be chosen carefully is the controller. the controller defines the control signal. Generally the controller output is is deficient. Due to this, the existence of the amplifier is necessary in order amplify of the control signal. Then an amplified signal is used to control the excitation system of the synchronous generator, and therefore, to define the terminal voltage level. The scheme of such a described system is illustrated in Figure 1.2.

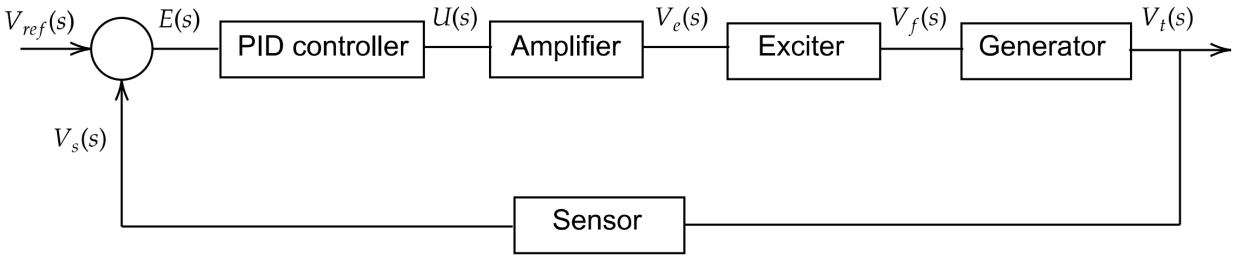


Figure 1.2: Scheme of the AVR

1.8 The need of automatic voltage regulator

The automatic voltage regulator is needed to satisfy many objectives, such as:

- Maintain the voltage magnitude of the synchronous generator at specified safety range, otherwise the performance of lights, motors, etc, will be affected.
- Dealing with the voltage dip and restoring the rated voltage without cause overshooting and regeneration.
- Avoiding malfunction operation that caused by the deviation of voltage in power plant or any load.
- To keep voltage under system fault conditions that ensures rapid operation of protective devices (as: relays, switches...).

1.9 Modelling of the AVR system

AVR Modelling system comprises four components which are : amplifier, exciter, generator and sensor. These components are formed as four transfer functions where saturation and nonlinearity are ignored(they are treated as linear devices)[6] as shown in Figure 1.3 illustrates the terminal voltage of synchronous generator (V_t). This voltage is sensed by sensor to be compared with reference voltage (V_{ref}) through a comparator. The output of the comparator is the error voltage (V_e) which is the difference between reference voltage (V_{ref}) and sensor voltage (V_s). The error voltage is amplified and sent to adjust the excitation current of the exciter. The excitation current regulates the rotor field current of synchronous generator which controls the electromagnetic force (E.M.F) on the stator winding that adjusts and regulates the stator terminal voltage of synchronous generator.[4]

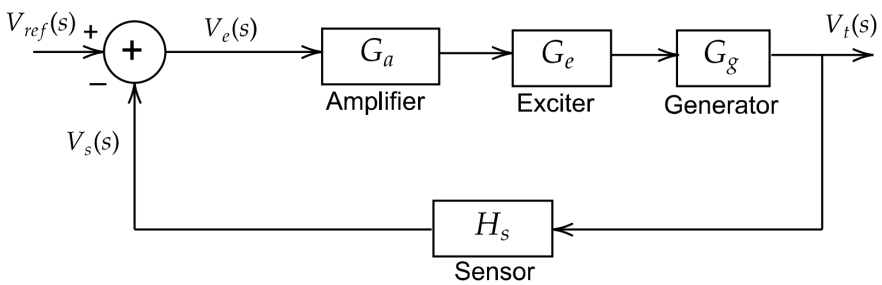


Figure 1.3: Block diagram of AVR system

It is to be noted that error voltage:

$$V_e = |V|_{ref} - |V|_s \tag{1.9.1}$$

(a) **The amplifier:** Transfer function of the amplifier is modeled by a gain and a time constant given by:

$$\Delta V_R(s) = \frac{K_A}{1+sT_A}\Delta V_e(s) \tag{1.9.2}$$

Hence ,the transfer function of the amplifier is :

$$G_A(s) = \frac{K_A}{1+sT_A}$$

Where the typical value of K_A is in the range $[10,40]$, and T_A is very small ranging from 0.02 to 0.1 s.

(b) **The exciter:** Transfer function for the exciter is modeled by a gain and a time constant given by

$$\Delta V_f(s) = \frac{K_e}{1+sT_e}\Delta V_R(s) \tag{1.9.3}$$

Thus, the transfer function of the exciter is :

$$G_e(s) = \frac{K_e}{1+sT_e}$$

Where the typical value of K_e is in the range $[10,400]$, and T_A is very small ranging from 0.5 to 1 s.

(c) **The generator :** Transfer function for the generator is modeled by a gain and a time constant given by:

$$\Delta V_t(s) = \frac{K_G}{1+sT_G}\Delta V_f(s) \tag{1.9.4}$$

the transfer function of the generator field is :

$$G_g(s) = \frac{K_G}{1+sT_G}$$

Where K_G may varies between 0.7 and 1.0, whereas T_G varies from 1.0 to 2.0 s from full load to no load .

(d) **The sensor :**

The sensor model is given by:

$$G_s(s) = \frac{K_R}{1+sT_R} \tag{1.9.5}$$

Where K_R is around 1, and T_R vary between 0.001 to 0.06s

The main components transfer function equations, their ranges, gains and time constants used are represented in Table 1.1.

Table 1.1: Models of components in AVR system

	Transfer function	Parameters Ranges	Used Parameters Values
Amplifier	$G_A(s) = \frac{K_A}{1+sT_A}$	$10 \leq K_A \leq 40$ $0.02 \leq T_A \leq 0.1$	$K_A = 10$ $T_A = 0.1$
Exciter	$G_e(s) = \frac{K_e}{1+sT_e}$	$1 \leq K_e \leq 2$ $0.4 \leq T_e \leq 1$	$K_e = 1$ $T_e = 0.4$
Generator	$G_g(s) = \frac{K_f}{1+sT_G}$	$0.7 \leq K_f \leq 1$ $1 \leq T_G \leq 2$	$K_f = 1$ $T_G = 1$
Sensor	$G_s(s) = \frac{K_R}{1+sT_R}$	$K_R = 1$ $0,001 \leq T_R \leq 0.06$	$K_R = 1$ $T_R = 0.01$

1.10 The simplified model of an AVR system with PID Controller

1.10.1 PID controller

It consists of three gains parameters: proportional gain K_p, integral gain K_i and derivative gain K_d. Proportional gain K_p reduces rise time. Integral gain K_i improves steady-state error. Derivative gain K_d reduces overshoot and improves stability margin. The PID equation is expressed in Laplace transform as follow:

$$G(s) = K_p + \frac{K_i}{s} + sK_d \tag{1.10.1}$$

- K_p: Proportional gain.
- K_i: Integral gain.
- K_d: Derivative gain.

The error voltage resulting from the comparator is used by the PID controller to generate a control signal which is then amplified to control the field windings of the generator by means of exciter. Figure 1.4 shows the schematic diagram of the AVR system with PID controller.

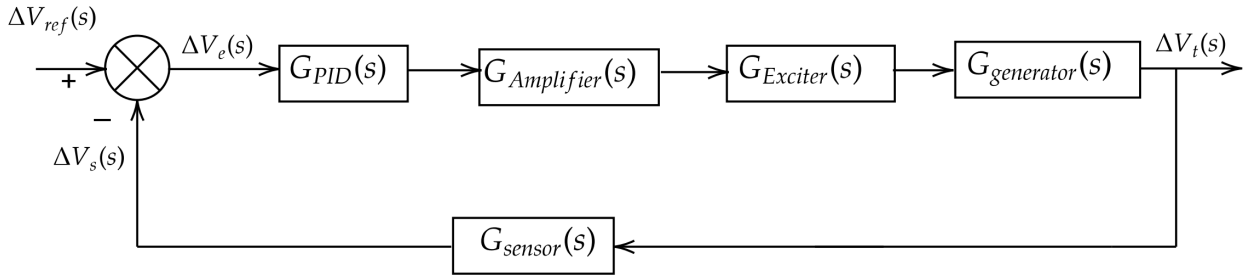


Figure 1.4: AVR system with PID controller

1.11 Conclusion

In this chapter, we have discussed the AVR system concepts and its importance for voltage regulation also it has been seen its closed control loop using PID controller.
The following chapter will go through the methods that will be utilized in the design of the AVR's PID parameters, such as PSO, CSO, MFO, WCO, TLBO, and HCO optimization algorithms.

Chapter 2

The proposed optimization algorithms

2.1 Introduction

The process of optimization is used to improve the output of a system so that the desired performances such as quality, cost, and efficiency can be achieved. Optimization is always desirable in many fields such as engineering design, computer science, operations research, economics, biomedicine ....

In order to accomplish the the aforesaid design process, this chapter will present all the algorithms that will be used for the sake of tuning the parameters of PID controller. In the following sections, will introduce the Particle Swarm Optimization(PSO) ,Cuckoo Search (CS), Moth Flame Optimization (MFO), Water Cycle Optimization (WCO), Teaching Learning-based Optimization, Hill Climbing, respectively.

2.2 Stochastic methods Classification

Stochastic methods divided into two main families called Heuristic (problem dependent) and Meta-heuristic algorithms (problem independent).

1. Heuristics Algorithms: are design as solution methods for specific problems.
2. Meta-heuristics Algorithms: are design to solve different type of problems.

2.2.1 Meta-heuristics Algorithms

Meta-heuristics Algorithms itself are divided into two main families.

2.2.1.1 Individual-based:

Tabu Search (TS), hill climbing, Iterated Local Search (ILS), Simulated Annealing (SA).

2.2.1.2 Population-based:

which can be classified to four categories based on the source of inspiration.

- **Evolution-based:** Inspired from the evolutionary phenomena in nature using the main operators including selection ,reproduction , recombination, and mutation[7]. Some of the recently proposed evolutionary algorithms are Biogeography-based Optimization (BBO) algorithm, evolutionary membrane algorithm, human evolutionary model, and Cuckoo Search Optimization (CS).

- **Swarm-based:** Inspired by the collective behavior of swarms in nature .Researchers have observed such behaviors of animals, plants, or humans, analyzed the driving force behind the phenomena, and then proposed various types of algorithms[8]. Some of the most recent ones are Glowworm Swarm Optimization (GSO) Bees Algorithm (BA), Artificial Bee Colony (ABC) algorithm, Bat Algorithm (BA), Firefly Algorithm (FA), Grey Wolf Optimizer (GWO). Dolphin Echolocation (DE, Hunting Search (HS), and Fruit Fly Optimization Algorithm (FFOA).

- **Physics-based:** Originates from physical laws in real-life and typically describes the communication of search solutions based on controlling rules ingrained in physical methods[9]. The most recent algorithms in this category are Gravitational Search Algorithm (GSA), Chemical Reaction Optimization (CRO), Artificial Chemical Reaction Optimization Algorithm (ACROA), Charged System Search (CSS) algorithm, Ray Optimization (RO), Black Hole (BH) algorithm, Central Force Optimization (CFO), Kinetic Gas Molecules Optimization algorithm (KGMO), and Gases Brownian Motion Optimization (GBMO).

- **Human-based:** Which motivated by human co-operations and human behavior in communities[9]. One of the most used algorithms in this group are Teaching Learning-Based Algorithm (TLBA) Imperialist Competitive Algorithm (ICA), and Harmony Search (HS).

2.3 Advantages and disadvantages of several stochastic optimization methods

There are several advantages and disadvantages for each of these families, for example, Individual-based algorithms need less computational cost and function evaluation but suffer from premature convergence (because of the local optima). In contrary, population-based algorithms have high ability to avoid local optima since a set of solutions are involved during optimization. In addition, information can be exchanged between the candidate solutions and assist them to overcome different difficulties of search spaces. However, high computational cost and the need for more function evaluation are two major drawbacks of population-based algorithms[10].

As the above paragraphs show, there are many algorithms in this field, which indicate the popularity of these techniques but if we consider the hybrid, multi-objective, discrete, and constrained methods, the number of publications will be increased dramatically.

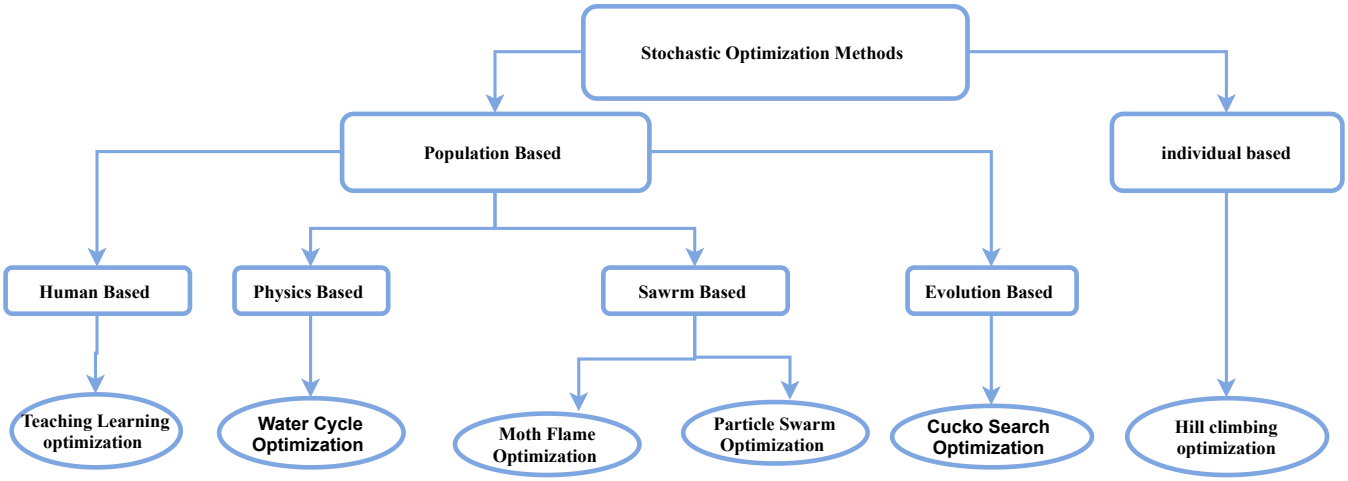


Figure 2.1: Classification of the proposed meta-heuristic algorithms

2.4 The proposed optimization techniques

2.4.1 Particle swarm optimization (PSO) algorithm

2.4.1.1 Introduction

The particle swarm optimization is (PSO) is a new metaheuristic population–based stochastic algorithm , based on paradigm of swarm intelligence and it is inspired by animal's social behavior such as including insects, herds, birds and fishes. PSO algorithm was firstly introduced by Russell Eberhart (electrical engineer) and James Kennedy (social psychologist) in 1995[11], a modified PSO was then developed in 1998 to improve the performance of the original PSO . It is now one of the most commonly used optimization techniques that have been widely used in many different fields.

The algorithm is based on the natural process of group communication to share individual knowledge when a group of birds or insects search food or migrate and so forth in a searching space, although all birds or insects do not know where the best position is. But from the nature of the social behavior, if any member can find out a desirable path to go, the rest of the members will follow quickly.

The PSO algorithm basically learned from animal's activity or behavior to solve optimization problems. In PSO, each member of the population is called a particle and the population is called a swarm. Starting with a randomly initialized population and moving in randomly chosen directions, each particle goes through the searching space and remembers the best previous positions of itself and its neighbors. Particles of a swarm communicate good positions to each other as well as dynamically adjust their own position and velocity derived from the best position of all particles. The next step begins when all particles have been moved. Finally, all particles tend to fly towards better and better positions over the searching process until the swarm move to close to an optimum of the fitness function.

The PSO method is becoming very popular because of its simplicity of implementation as well as ability to swiftly converge to a good solution .

2.4.1.2 PSO's principle

Particle Swarm Optimization (PSO) is used to investigate the search space of some optimization function and test the function coefficients and parameters in order to minimize the fitness function values. The PSO method aims to optimize a problem iteratively, starting with a set, or population, of candidate solutions, called a swarm of particles, in which each particle knows the global best position within the swarm (and its corresponding value), along with its individual best position (and its fitness value) found so far during the search process in the problem's solution space.

At each iteration, the velocity and the position of each particle in the swarm, represented by d-dimensional vectors, are influenced by the individual and the collective knowledge, which direct the repeated flights of the particles over the space of possible solutions to the problem in search of the optimum, until a suitable stopping criterion is satisfied.

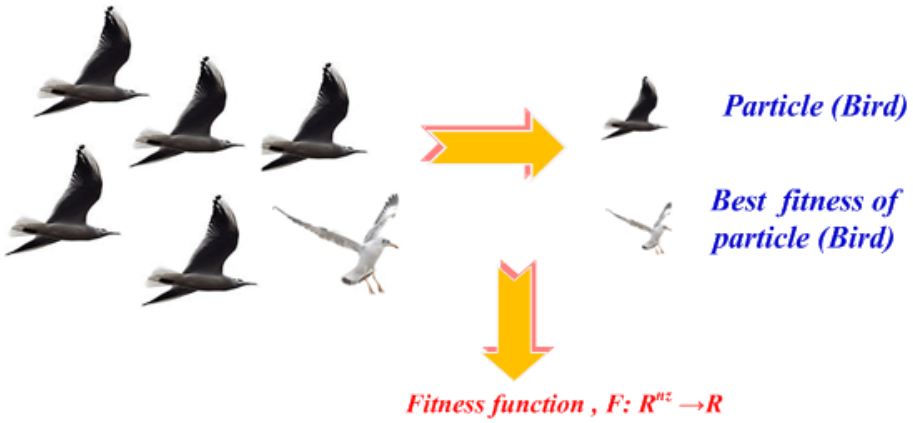


Figure 2.2: Swarm of birds move into better regions

2.4.1.3 Main Parameters of the PSO

Population Size The total number of particles utilised in the swarm is the population size. A big swarm generates larger parts of the search space to be covered per iteration . In the first paper on PSO Kennedy and Eberhart[11] referred to the zoological studies in which the movements of flocks composed of 15–30 birds were simulated. This seems to be an initial guess as the authors of [12] use 20 particles in their first PSO tests . Discussion on the population size of PSO in all relatively recent reviews seems to either be skipped, or based on suggestions given in the early studies highlighted above, without getting into detail about doubts that could easily come up when the results are focused on in depth[12], but "in most cases it has been demonstrated that when the number of individuals is larger than 50, PSO is not sensitive to the size of the population". Finally, Wang et al.[13] simply stated that the common selection is between 20 and 50 particles.

Personal and Global Best Determining global and personal best recorded positions is held important for the algorithm process operation. The global best is the position where the best solution (fitness value) is achieved among all particles so far, whereas the another best value is the personal best which is the best position of the particle among its all positions visited so far, these two parameters are formally defined as:

$$P_{i,best}(t) = \begin{cases} P_{i,best}(t-1) & \text{if } f[x(t_i)] \geq f[P_{i,best}(t-1)] \\ x_i(t) & \text{otherwise} \end{cases} \tag{2.4.1}$$

$$G_{best}(t) = argmin\left\{f[P_{1,best}(t)], f[P_{2,best}(t)], f[P_{3,best}(t)], \ldots, f[P_{N,best}(t)]\right\} \tag{2.4.2}$$

where:

- $P_{i,best}(t)$ is the personal best position of particle i at iteration t.
- $f[x_i(t)]$ is fitness function evaluation at position $x_i(t)$.
- $x_i(t)$ is position of particle i at iteration t.
- $G_{best}(t)$ is the global best position discovered by the swarm from beginning till iteration t.
- *argmin* is a function that returns the argument that causes the minimum image of function f.

Particle's velocity The concept of PSO includes changing the velocity of each particle to its best position P_{best} and the global best G_{best} position at each time step (iteration). For example, the jth particle is represented as $x_j = \left(x_{j,1}, x_{j,2}, \ldots, x_{j,g}\right)$ in the g-dimensional space. The best previous position of the th particle is recorded and represented as $pbest\,j = \left(pbest_{j,1}, pbest_{j,2}, \ldots, pbest_{j,g}\right)$. The index of best particle among all of the particles in the group is represented by the $gbest_g$. The rate of the position change (velocity) for particle is represented as $v_j = \left(v_{j,1}, v_{j,2}, \ldots, v_{j,g}\right)$. The modified velocity and position of each particle can be calculated using the current velocity and the distance from $pbest_{j,g}$ to $gbest_g$. As shown in the following formulas:

$$v_{i,g}^{(t+1)} = \omega v_{i,g}^{(t)} + c_1.rand()\left(pbest_{i,g} - x_{i,g}^{(t)}\right) + c_2.Rand()\left(gbest_g - x_{i,g}^{(t)}\right) \tag{2.4.3}$$

$$x_{i,g}^{(t+1)} = x_{i,g}^{(t)} + v_{i,g}^{(t+1)}\ ;\ j = 1,2,\ldots,n\ ;\ g = 1,2,\ldots,m \tag{2.4.4}$$

where:

- n: number of particles in a group
- m: number of members in a particle
- t: pointer of iterations (generations)
- $v_{i,g}^{(t)}$: velocity of particle at iteration
- ω: inertia weight factor
- c_1, c_2: acceleration constant
- $rand(), Rand()$: random numbers between 0 and 1
- $x_{i,g}^{(t)}$: current position of particle at iteration

The constants c_1 and c_2 represent the weighting of the stochastic acceleration terms that pull each particle toward and positions. Low values allow particles to roam far from the target regions before being tugged back. On the other hand, high values result in abrupt movement toward, or past, target regions. Hence, the acceleration constants and were often set to be 2.0 according to past experiences.

Suitable selection of inertia weight provides a balance between global and local explorations, thus requiring less iteration on average to find a sufficiently optimal solution[14]. As originally developed, often decreases linearly from about 0.9 to 0.4 during a run. In general, the inertia weight ω is set according to the following equation:

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{iter_{max}} \times iter \tag{2.4.5}$$

where $iter_{max}$ is the maximum number of iterations (generations), and is the current number of iterations.

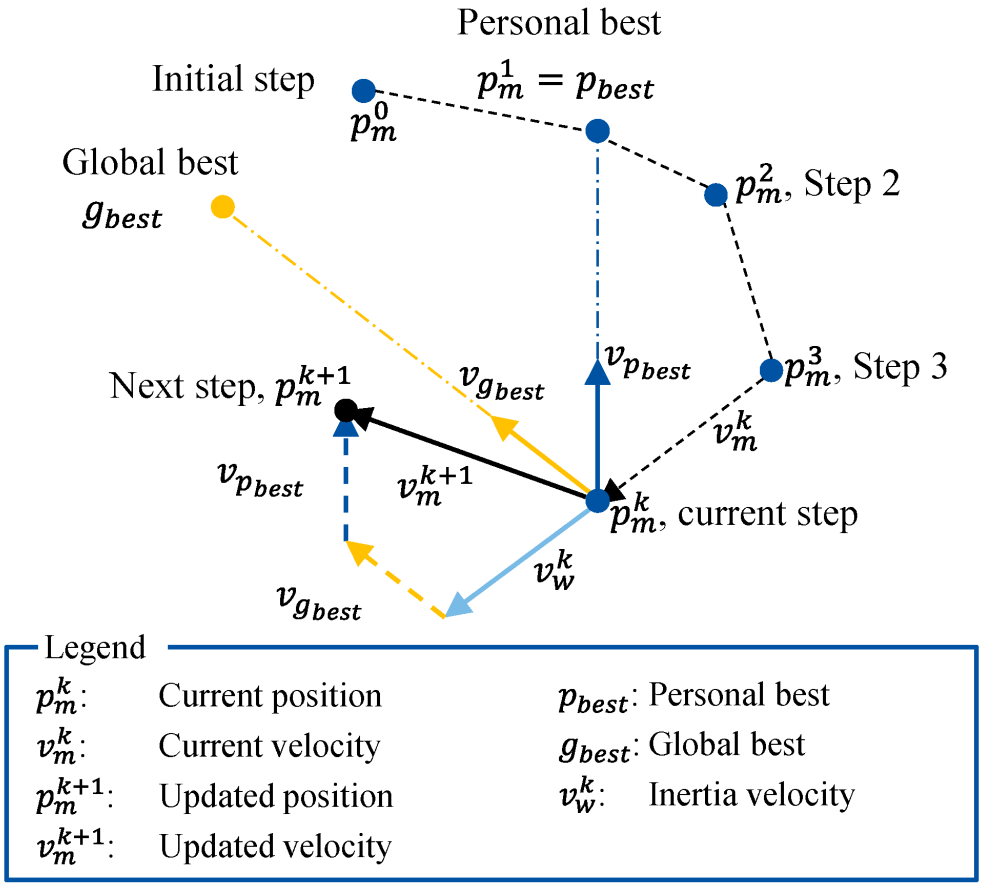


Figure 2.3: velocity and position update for a particle in a two-dimensional search space

Iteration numbers The number of iterations to obtain a good result is also problem-dependent. A too low number of iterations may stop the search process prematurely, while too large iterations has the consequence of unnecessary added computational complexity and more time needed[15].

2.4.1.4 Concept of PSO algorithm

The PSO algorithm works by assigning several particles to move in the search space and update particle parameters during each iteration. Each particle contains the information of its positions in

the search space and velocities about its moving speed which indicates its next position in the search space. Each particle also has the information about its best fitness value about the objective function and the global best fitness value based on the entire particle's information.

The steps of PSO algorithm is listed below:

- Calculate each particle's fitness value based on its positions and the objective function
- Update global best fitness value among all the particles and notice each particle and update responding parameters inside each particle.
- Update each particle's velocities and positions based on PSO velocity equation and position equation.

The following pseudo code summarizes the main steps of particle swarm optimization algorithm:

```
for Each Particle do
    Initialize particle;
end
for Each Particle do
    Calculate fitness value ;
    if the fitness value is better than its peronal best_set then
        current value as the new p_Best;
    end
end
Choose the particle with the best fitness value of all as g_Best ;
for Each Particle do
    Calculate particle velocity ;
    Update particle position ;
end
```

Algorithm 1: Pseudo code for the PSO algorithm

The swarm is initialized, then the position and velocity of particles are randomly initialized within the search space. After that, the objective values of particles are calculated. The first objective values and positions are automatically the personal best values and the personal best positions. The global best value and the global best position are set to the objective value and position of the particle with the best objective value in the entire swarm. After the initial step, all particles are moved to their new positions using Equation 2.4.4. All objective values are evaluated again. Personal best positions are updated for particles that have a new objective value that is better than the old personal best value. The global best position is updated if there is any particle with an objective value that is better than the old global best value. Again, all particles are moved to their new positions using Equation 2.4.3. The algorithm continues with evaluating the objective values and updating the positions, the personal best values, the personal best positions, the global best value, and the global best position. The algorithm stops if a termination criterion, such as a limit on the number of iterations, is reached. The PSO algorithm can be given by the flowchart shown in Algorithm 1.

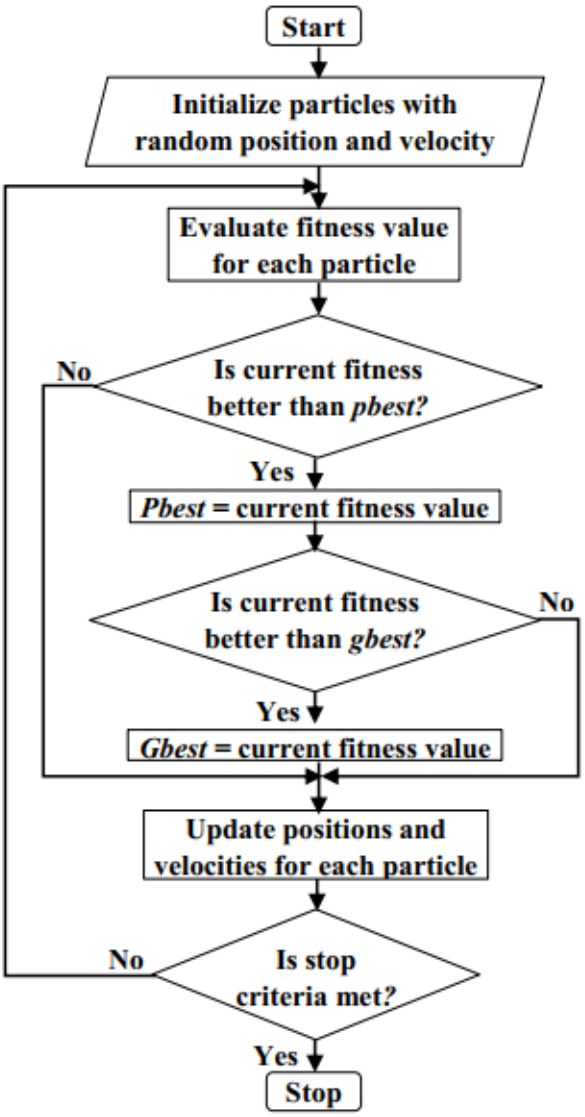


Figure 2.4: Flow chart of Particle Swarm Optimization

2.4.1.5 Advantages and Disadvantages of PSO

It is said that PSO algorithm is the one of the most powerful methods for solving the global optimization problems while there are some disadvantages of the PSO algorithm. The advantages and disadvantages of PSO are discussed below.

Advantages of the PSO algorithm

1. It is easy to implementation, so it can be applied both in scientific research and engineering problems.
2. It has a limited number of parameters and the impact of parameters to the solutions is small compared to other optimization techniques.
3. The calculation in PSO algorithm is very simple.

4. There are some techniques which ensure convergence and the optimum value of the problem calculates easily within a short time.

5. PSO is less dependent of a set of initial points than other optimization techniques.

Disadvantages of the PSO algorithm

1. Slow convergence in refined search stage (weak local search ability).

2. The method can not work out the problems of non-coordinate system , such as the solution to the energy field.

2.4.2 Cuckoo Search Optimization

2.4.2.1 Introduction

Cuckoo search (CS) is one of the recent evolutionary nature-inspired metaheuristic algorithms, developed in 2009 by Xin-She Yang and Suash Deb[16]. It is inspired based on the unique behavior of the bird cuckoo. these are the "Brood parasites" birds. It never builds its own nest and lays their eggs in the nest of another host bird nest. This aggressive reproduction brooding parasitism acts as follows. A female cuckoo typically lays 16 to 22 eggs that can be different in color and can match the eggs of the host. Different species may produce different colors, and they often target at different host birds. The host takes care of the eggs presuming that the eggs are its own. However, some of host birds are able to combat with this parasites behavior of cuckoos and throw out the discovered alien eggs or build their new nests in new locations.

CS ere firstly proposed as a tool for numerical function optimization and continuous problems. Researchers tested this algorithm on some well-known benchmark functions and compared with PSO and GA. Since then, the original developers of this algorithm and many researchers have also applied this algorithm to engineering optimization, where Cuckoo search also showed promising results. Nowadays, cuckoo search has been applied in almost every area and domain of function optimization, engineering optimization, image processing, scheduling, planning, feature selection, forecasting, and real-world applications.

Figure 2.5: A cuckoo egg (green) in a host nest

2.4.2.2 CS's Principle

Looking at the parasitic lifestyle of cuckoo species and tackling of host birds against them, there is an analogy for developing a population-based metaheuristic inspiring by this biological system . The initial population taken is the number of cuckoos and its eggs.

Cuckoo will search and lays its eggs in the nests of other host species. It will seek for the best nest from the accessible nests. It relies on three attitudes namely it lays one egg at a time, and the nest with best eggs can be agitated over next bearing for hatching, the accessible amount of host nests is fixed. Based on this cuckoo hatching address the optimized solution is acquired for the problem. All the nests or eggs whether they belong to the cuckoos or host birds represent the candidate solutions in the search space. Cuckoos and host birds try to breed their own generation.

Authors[17] defined the CS algorithm by setting three rules that idealize behavior of cuckoos in order to become appropriate for implementation as an computer algorithm:

- Each cuckoo lays one egg at a time, and dumps it in a randomly chosen nest.
- The best nests with high-quality eggs will be carried over to the next generations.
- The number of available host nests is fixed and the egg laid by a cuckoo may be discovered by the host bird with a probability $p_a \in (0,1)$. In this case, the host bird can either get rid of the egg, or simply abandon the nest and build a completely new nest.

Regarding the mentioned rules, the CS was implemented as follows:

- Each egg in a nest represents a candidate solution. Thus, each cuckoo can lay only one egg into a nest in original form although each nest can has multiple eggs representing a set of solutions, in general. The task of CS is to generate the new via levy flight and potentially better solutions that will replace the worse solutions in the current nest population. The quality of solutions is evaluated with the objective function of the problem to be solved. Normally, this function needs to be minimized.
- Mathematically, these kinds of problems are trivial to transform from minimization to maximization problems regarding equation $min(f(x)) = max(-f(x))$. In contrast objective function, such transformed function is now named as a fitness function.
- Furthermore, the last rule is approximated by an additional parameter p_a named the switching probability that determine when the worst of the n host nests is replaced by a new randomly generated nest. In fact, this parameter balances two components of the CS process, i.e., exploration and exploitation as identified by [18] where too much exploitation induces premature convergence, while too much exploration slows down the convergence[14].

2.4.2.3 Levy Flight

Levy flight is a random walk; the steps are defined regarding the step-lengths, which have a certain probability distribution, with the directions being random. This random walk can be observed in animals and insects. The next movement is based on the current position.[19]

$$X_i(t+1) = X_i(t) + \alpha \oplus Levy(\lambda)(2.6) \tag{2.4.6}$$

Where $\alpha > 0$ is the step size. In most of the cases assume that α is equal to one. The product $\oplus$ means entry-wise multiplication i.e. Exclusive OR operation Levy flight is a random walk with random step size following a levy distribution

$$Levy \sim u = t - \lambda (1 < \lambda \leq 3) \tag{2.4.7}$$

In the case of CS, the use of Levy Flight improves and optimizes the search: new solutions are generated by a random walk of Levy around the best solution obtained until now, which accelerates the global search.

2.4.2.4 Alien Eggs Discovery by the Host Birds

The alien eggs discovery is performed for each component of each solution in terms of the discovering probability matrix (P) such as:

$$P_{i,j} = \begin{cases} 1 & \text{if } rand < p_a \\ 0 & \text{if } rand \geq p_a \end{cases} \tag{2.4.8}$$

where *rand* is a random number in [0, 1] interval and pa is the discovering probability. It should be noted that the P matrix has the same size as the Nest matrix.

2.4.2.5 Cuckoo Search in Applications

Cuckoo search has been applied in many areas of optimization, engineering design,data ming and computational intelligence with promising efficiency. For example, in the engineering design applications, cuckoo search has superior performance over other algorithms for a range of continuous optimization problems such as spring design and welded beam design problems[17] [20] [21].

Among the diverse applications, an interesting performance enhancement has been obtained by using cuckoo search to train neural networks as shown by Valian et al. [22] and reliability optimization problems[23] .Also , Yang and Deb[24] developed a multi objective cuckoo search (MOCS) for design engineering applications .Recent studies have demonstrated that cuckoo search can performance significantly better than other algorithms in many applications.

Objective function $f(x), x = (x1, ..., xd)T$;
Initial a population of n host nests $x_i (i = 1, 2, ..., n)$;
while *(t < Max Generation) or (stop criterion)* **do**
  Get a cuckoo (say i) randomly by Levy flights ;
  evaluate its quality/fitness F_i ;
  Choose a nest among n (say j) randomly ;
end
if $F_i > F_j$ **then**
  Replace j by the new solution ;
end
Abandon a fraction (p_a) of worse nests and build new ones at new locations via Levy flights ;
Keep the best solutions (or nests with quality solutions) ;
Rank the solutions and find the current best ;

Algorithm 2: cuckoo search algorithm

And the CSO algorithm can be given by the flowchart shown in Figure 2.6

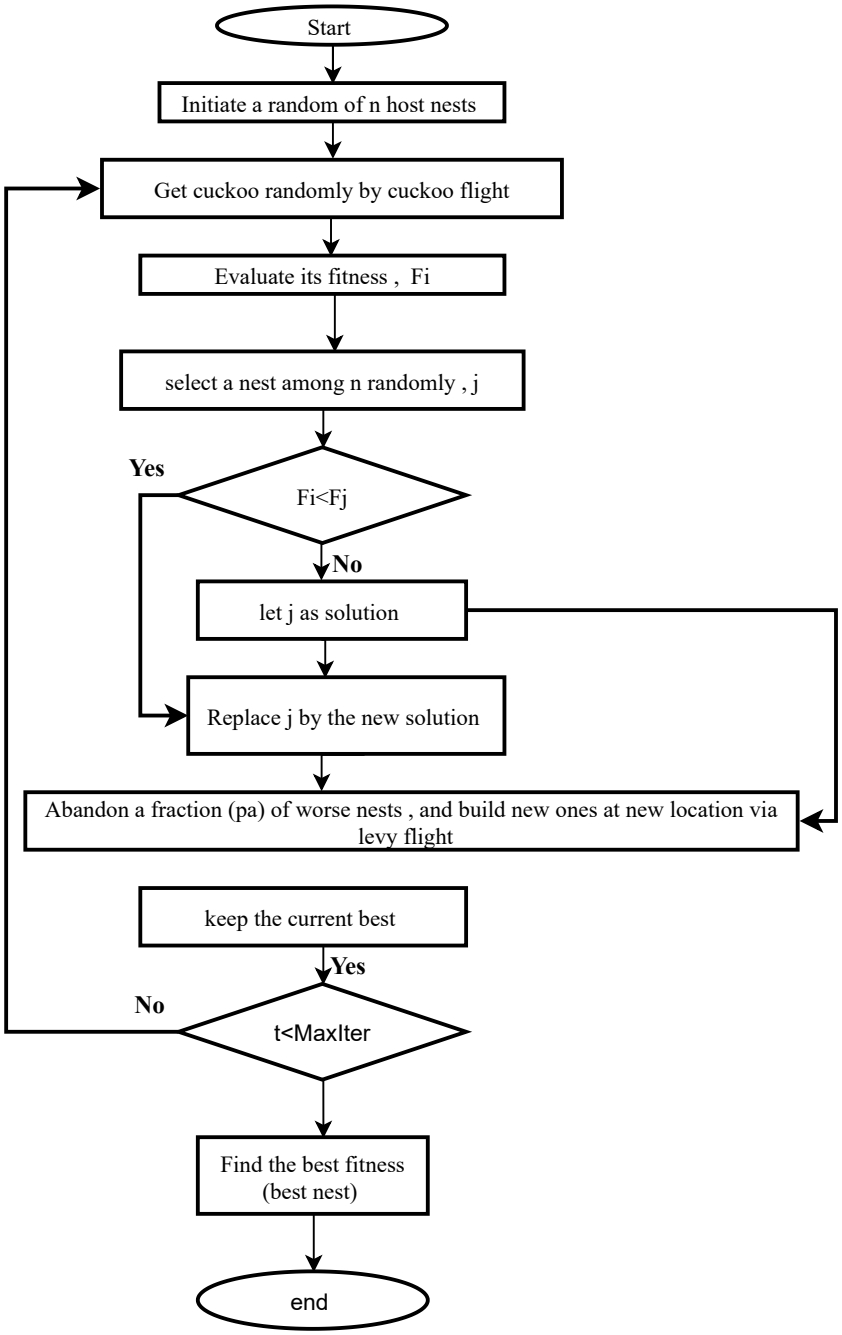


Figure 2.6: Flow chart of Cuckoo Search Optimization

2.4.3 Moth Flame Optimization (MFO) algorithm

2.4.3.1 Introduction

The moth-flame optimization algorithm[10] was recently proposed by Mirjalili in 2015, which is one of the latest algorithms that has gained extensive attention in recent years. This new intelligent algorithm simulates the moths navigation manner in nature (transverse orientation). It is a population-base swarm computation search technique which mimics the behavior of moths their special navigation methods at night.

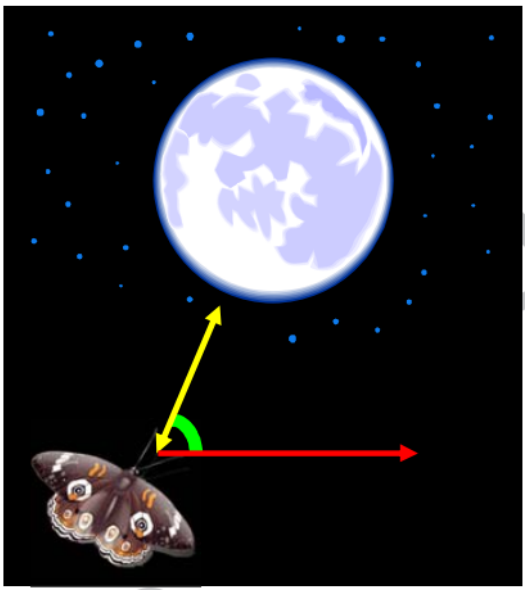


Figure 2.7: Transverse orientation

The idea of the MFO is based on a mechanism called transverse orientation for navigation in night throw the moon light. Using this mechanism, moth flies with a fixed angle with respect to the moon. When moths see a human-made artificial light, they try to maintain a similar angle with the light to fly in straight line. Since such a light is extremely close compared to the moon, maintaining a similar angle to the light source causes a useless or deadly spiral by path for moths[25]

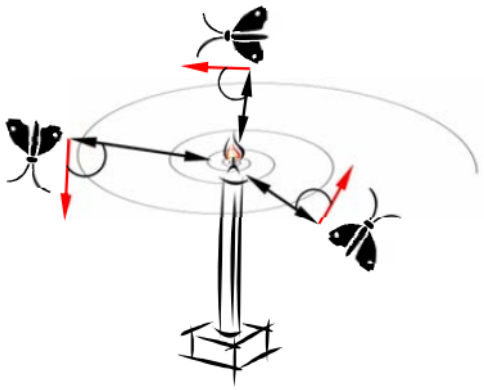


Figure 2.8: Spiral flying path around close light sources

2.4.3.2 Concept of MFO algorithm

In the proposed MFO algorithm, moths are considered as search agents, flames are the solution and the problem's variables are the position of moths in the space. Therefore, the moths can fly in 1-D, 2-D, 3-D, or hyper dimensional space with changing their position vectors since the MFO algorithm is a population-based . It should be noted here that moths and flames are both solutions. The difference between them is the way we treat and update them in each iteration. The moths are actual search agents that move around the search space, whereas flames are the best position of moths that obtains so far. In other words, flames can be considered as flags or pins that are dropped by moths when searching the search space. Therefore, each moth searches around a flag (flame) and updates it in case of finding a better solution. With this mechanism, a moth never lose its best solution[26].

2.4.3.3 MFO Algorithm Parameters

Initially, the MFO algorithm choose the solutions randomly, then evaluated according to the selected fitness function. Evaluation process allows defining best position for each particle and the best position for the whole swarm. The next position of the swarm will be determined based on the previous best position, and evaluated again. During each move (iteration), the best position of each agent is updated and the best position of the whole swarm as well. The following steps are involved in the proposed algorithm :

1. We represent the set of moths in a matrix as follows:

$$M = \begin{bmatrix} m_{1,1} & m_{1,2} & \dots & m_{1,d} \\ m_{2,1} & m_{2,2} & \dots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \dots & m_{1nd} \end{bmatrix} \tag{2.4.9}$$

   Where n is the number of moths and d is the number of variables (dimension).

2. The corresponding fitness values can be sorted in an array form as follows:

$$M = \begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \tag{2.4.10}$$

   Where n is the number of moths. The fitness value is the return value of the objective function for each moth. The position vector of each moth is passed to the fitness function and its output is assigned to the corresponding moth as its fitness value.

3. Similar to the moth matrix , We consider a flame matrix as follows:

$$F = \begin{bmatrix} F_{1,1} & F_{1,2} & \dots & F_{1,d} \\ F_{2,1} & F_{2,2} & \dots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & F_{n,2} & \dots & F_{1nd} \end{bmatrix} \tag{2.4.11}$$

   Where n is the number of moths and d is the number of variables (dimension).

4. For the flames ,we also assume that there is an array for storing the corresponding fitness values as follows:

$$M = \begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \tag{2.4.12}$$

   Where n is the number of moths.

5. In order to mathematically model this behavior, the position of each moth is updated with respect to a flame using the following equation:

$$S\left(M_i, F_j\right) = D_i e^{bt} \cos\left(2\pi t\right) + F_j \tag{2.4.13}$$

Where M_i indicates the ith moth, F_j indicates the jth flame and S is the spiral function , D_i shows the distance of the ith moth for the jth flame, b is a constant for representing the shape of the logarithmic spiral and t is a random number in $[-1,1]$. This function satisfies the following conditions[25].

- The initial point of the helical function is selected from the initial space position of the moth.
- The end point of the spiral is the space position corresponding to the contemporary flame.
- The fluctuation range of the spiral should not exceed its search space.The variable D_i may be calculated as follows:

$$D_i = F_j - M_i \tag{2.4.14}$$

In order to further emphasize exploitation, t is defined as random number in $[r,1]$ where r is linearly decreased from -1 to -2 over the course of iteration. According to equation 2.4.15,each moth is restricted to move towards a flame that may lead to local optimum stagnation. In order to prevent this, at each iteration, a list of flames must be updated and sorted based on their fitness values. The moths then update their positions with respect to their corresponding flames. Since the position updating of moths with respect to n different locations in the search space may degrade the exploitation of the best promising solutions, an adaptive mechanism for the number of flames has been proposed as in the following formula[25]:

$$Flame\ number = round\left(N - \frac{l \times N - 1}{T}\right) \tag{2.4.15}$$

where l is the current number of iteration, N is the maximum number of flames and T indicates the maximum number of iterations.

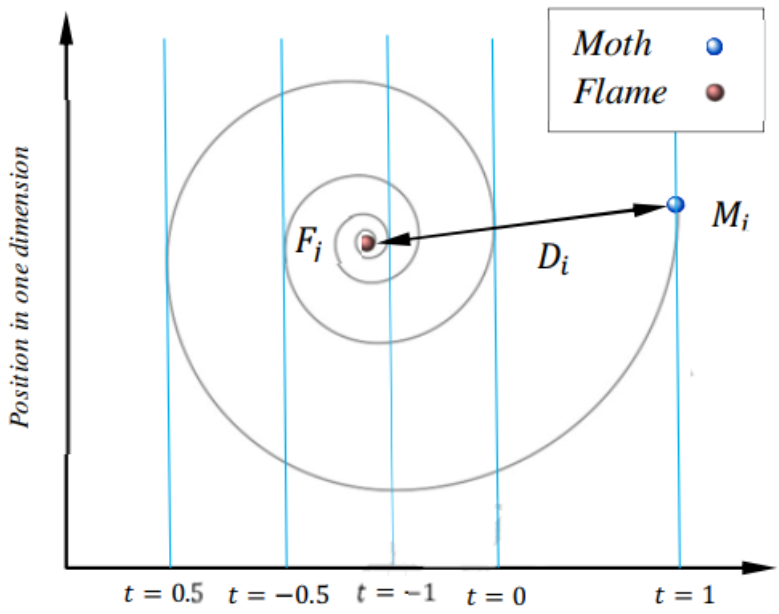


Figure 2.9: Logarithmic spiral, space around a flame, and the position with respect to t

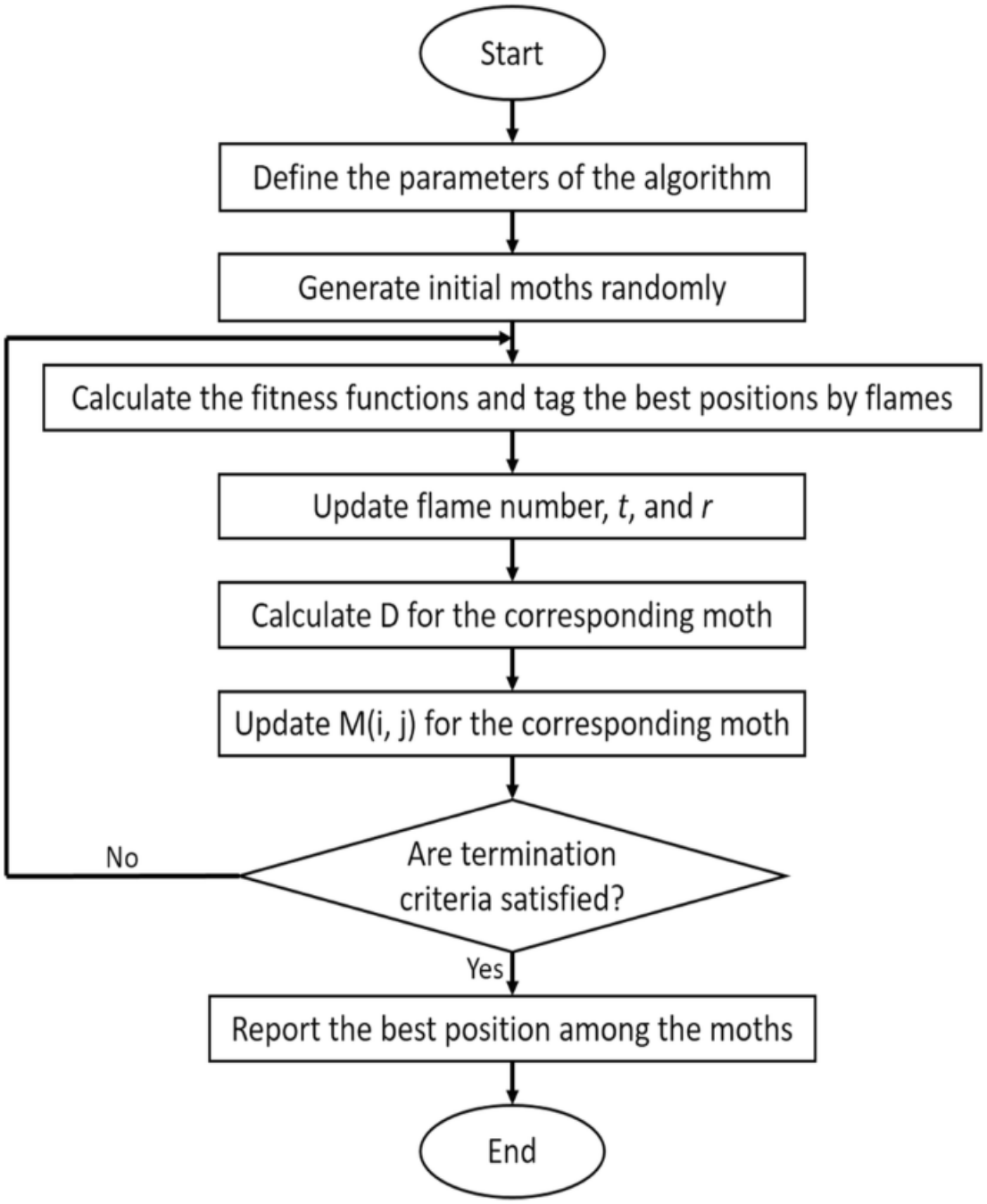


Figure 2.10: Flowchart of the MFO algorithm

The MFO algorithm can be summarized as flowchart and pseudo as follow:

% generate initial solutions and calculate the objective function value ;
for $i = 1 : n$ **do**
  for $j = 1 : d$ **do**
    $M(i, j) = (ub(i) - lb(i)) \times rand() + lb(i)$;
  end
end
$OM =$ Fitness Function(M);
% Update flame n **if** *iteration* $==$ 1 **then**
  $F =$ sort(M) ;
  $OF =$ sort(OM) ;
else
  $F =$ sort($M_t - 1, M_t$); $OF =$ sort($M_t - 1, M_t$);
end
for $i = 1 : n$ **do**
  for $j = 1 : d$ **do**
    % Update r and t ;
    % Calculate D using Equation 2.4.13 with respect to the corresponding moth ;
    %Update $M(i, j)$ using equations 2.4.11 and 2.4.12 with respect to the corresponding moth ;
  end
end

Algorithm 3: Pseudo code for The MFO algorithm

2.4.4 Water Cycle Algorithm

2.4.4.1 Introduction

WCA (also known as the hydrological or the H2O cycle is a met-heuristic optimization method (physics-based) introduced by Eskandar et al.(2012). The fundamental concepts and ideas which underlie the WCA is inspired by nature and based on the observation of water cycle process and how rivers and streams flow to the sea in the real world[27]. It consists of several phases such as evaporation, precipitation, and surface run-off. During H_2O cycle process, which initiates from the highest position in the mountains and terminates in the downhill locations, water continuously flows across the land surface within streams and rivers toward the sea. Streams and rivers take surface-water bodies from the rain and other sub-streams on their journey to the deepest location. The river water evaporation often occurs when the plants emit water during the transpire procedure. Water vapor is progressively transmitted to the atmosphere to create scattered clouds. Then, condensed clouds generate raining drops to create some new stream flows and rivers[28].

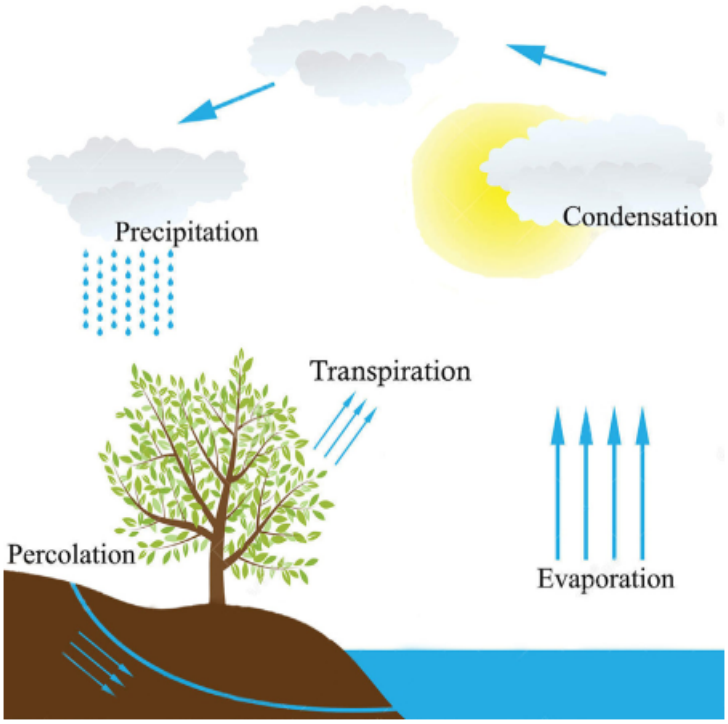


Figure 2.11: A schematic view of the hydrologic cycle (water cycle process)

2.4.4.2 WCO algorithm Concept

he algorithm initiates with the rain equation 2.4.10 or precipitation phenomena by generation of a random population of design variables so called the raindrops. The best individual (best raindrop) is chosen as a sea. Then, a number of good raindrops are chosen as equation 2.4.10 river and the rest of the raindrops are considered as streams, which flow to the rivers and sea.

Depending on their magnitude of flow (objective function value), each river absorbs water from the streams. In fact, the amount of water in a stream entering a rivers and/or sea varies from other streams. In addition, rivers flow to the sea, which is the most downhill location.

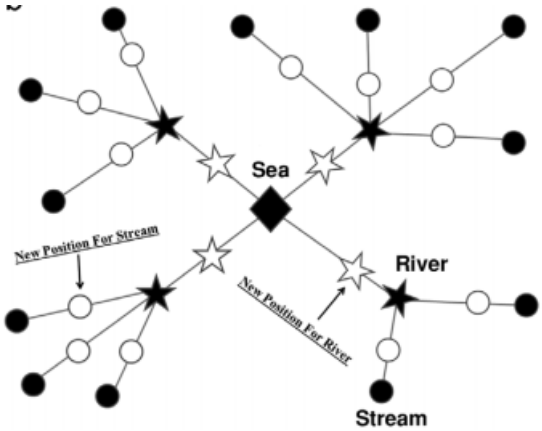


Figure 2.12: Schematic View of the WCA Optimization Process

Opposite to PSO, WCA has been designed based on the indirect movement concept from streams to the rivers and from the rivers to the sea as the temporal optimum solution. It is significant to note

that during optimization procedure, stream movements exploit vicinity of these partial solutions, but exploring the search space is done using stochastically driven evaporation and raining operations[28].

2.4.4.3 WCA Algorithm Parameters

In order to implement WCA algorithm, the following steps are involved in the proposed algorithm:

1. Generating the basic population randomly over the search space and forming the initial sea, rivers, and streams.

   Where stream, river, and the sea show a $1 \times N$-dimensional array representing a particular solution defined as: $stream = [x_1, x_2, x_3 \ldots x_N]$, and Nsr indicates the total number of rivers and sea. $Nstream$ represents number of streams, which directly or indirectly flow to rivers and sea.

$$Total population = \begin{bmatrix} sea \\ River_1 \\ River_2 \\ River_3 \\ \vdots \\ Stream_{N_{s+1}} \\ Stream_{N_{s+2}} \\ Stream_{N_{s+3}} \\ \vdots \\ Stream_{N_{pop}} \end{bmatrix} = \begin{bmatrix} x_1^1 & x_2^1 & x_3^1 & \ldots & x_N^1 \\ x_1^2 & x_2^2 & x_3^2 & \ldots & x_N^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_2^{N_{pop}} & x_2^{N_{pop}} & x_3^{N_{pop}} & \ldots & x_N^{N_{pop}} \end{bmatrix} \tag{2.4.16}$$

$$N_{sr} = textNumberofrivers + 1 \tag{2.4.17}$$

$$N_{streams} = N_{pop} - N_{sr} \tag{2.4.18}$$

2. Calculating the cost of every population member.

3. Computing the corresponding flow intensity of rivers and sea.

$$C_n = Cost_n - Cost_{Nsr+1} \tag{2.4.19}$$

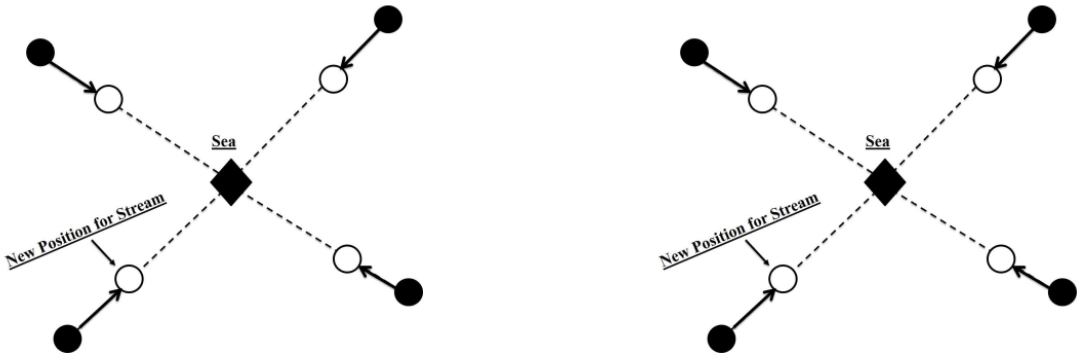
$$NS_n = round\left\{ \left| \frac{Cost_n - Cost_{Nsr+1}}{\sum_{n=1}^{N_{sr}} c_n} \right| \times N_{streams} \right\} \; ; \; n = 1, 2, 3, \ldots, N_{sr} \tag{2.4.20}$$

4. Where NSn stands for the number of streams, which are flowing into the related rivers or sea and Cost is the fitness value of every population number.

5. The streams flow into rivers and the sea according to :

$$X_{Stream}(t+1) = X_{Stream}(t) + rand \times C \times (X_{Sea}(t) - X_{Stream}(t)) \tag{2.4.21}$$

$$X_{Stream}(t+1) = X_{Stream}(t) + rand \times C \times (X_{River}(t) - X_{Stream}(t)) \tag{2.4.22}$$



(a) New Position of the Streams, which Flow toward the Rivers **(b)** New Position of the Streams, which Flow toward the Sea

Figure 2.13: Schematic View of the streams flowing into rivers and the sea Process

A value between 1 and 2 (near to 2).When the C value is restricted to be greater than 1, streams can move toward the rivers from various directions.

6. Each river flows into the downhill place or sea based on the following equation:

$$X_{River}(t+1) = X_{River}(t) + rand \times C \times (X_{sea}(t) - X_{River}(t)) \tag{2.4.23}$$

We Exchange the positions of river with a stream that generates a better solution. Similar to if a river explores a better solution compared to the sea; the position of them should be exchanged.

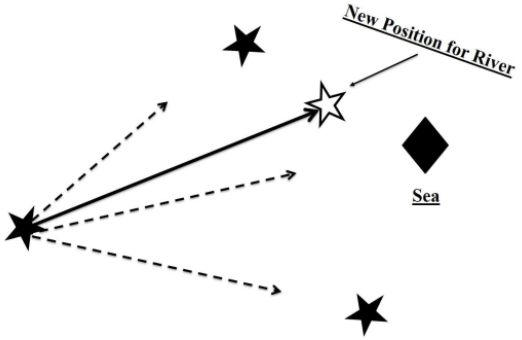


Figure 2.14: New Position of Every River by Considering the Influence of Sea and all other Rivers

7. Checking the evaporation condition based on the following pseudo-code :

if $\left|X_{sea} - X^i_{River}\right| < d_{max}$ *or* $rand < 0.1$ **then**
  for $i = 1;2;3;\ldots;N_{sr-1}$ **do**
    perform raining process
  end
end
if $\left|X_{sea} - X^i_{Stream}\right| < d_{max}$ *or* $rand < 0.1$ **then**
  for $i = 1;2;3;\ldots;N_{sr-1}$ **do**
    perform raining process
  end
end

Where d_{max} is a very small number. d_{max} is determined by user to control the intensification level in the proximity of best solutions. This assumption is proposed in order to prevent searching procedure from premature convergence in tackling constrained tasks[28].

8. If the evaporation process satisfied, the raining process occurs according to the following equation:

$$X_{sream}^{new}(t+1) = LB + rand \times (UB - LB) \tag{2.4.24}$$

here LB and UB represent lower and upper boundaries of the given problem, respectively.

9. Decreasing the value of controlling parameter d_{max} via equation 2.4.18:

$$d_{max}(t+1) = d_{max}(t) \left(\frac{d_{max}}{max(iteration)} \right) \tag{2.4.25}$$

10. Checking the convergence criteria. If the termination condition is fulfilled, terminate; otherwise, return to step 4.

The WCO algorithm can be summarized as Flow char and pseudo code in Figure 2.15 and Algorithm 5 respectively.

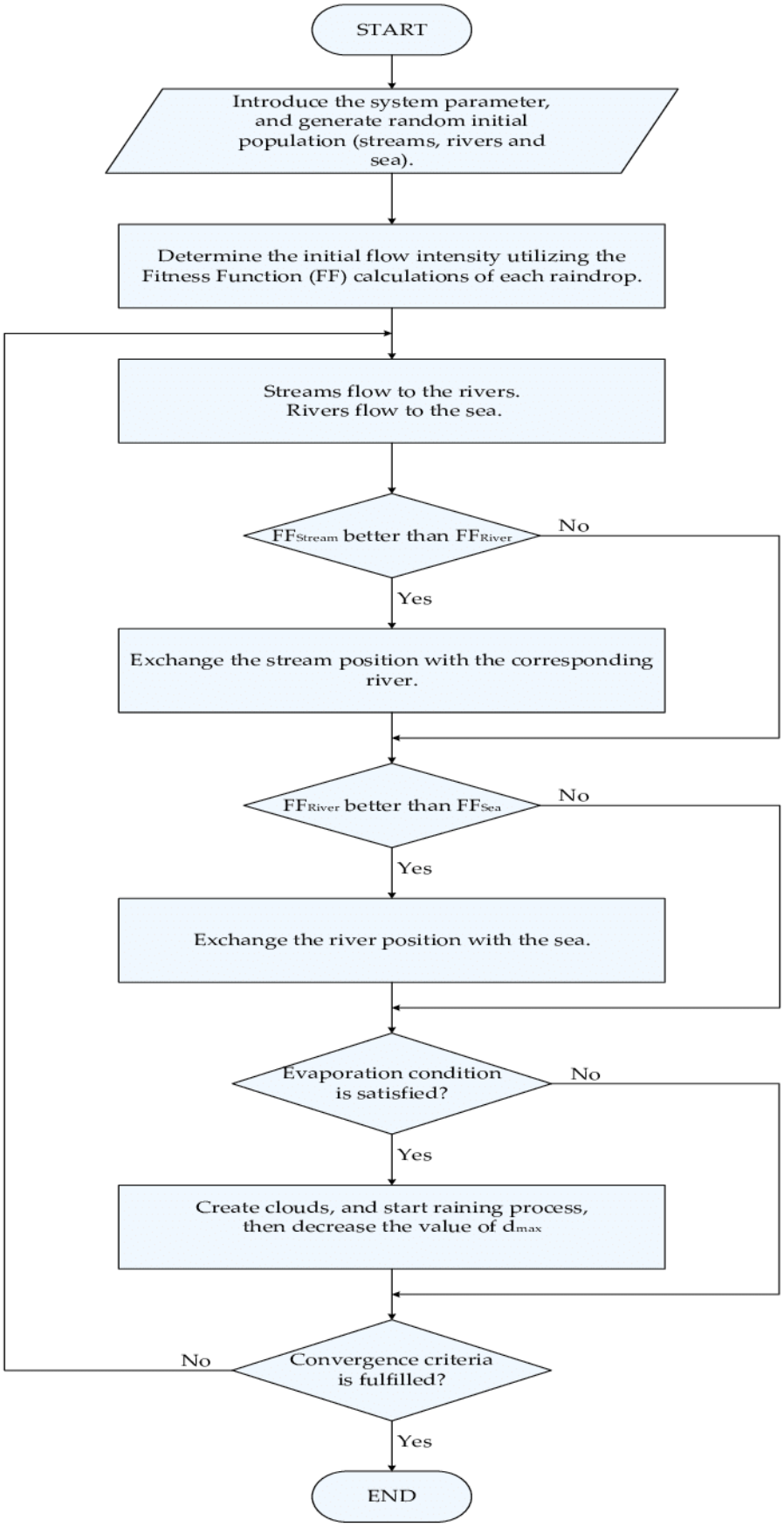


Figure 2.15: Flowchart of WCO Algorithm

Set user parameter of the WCA: N_{pop}, Nsr, d_{max} and Maximum Iteration ;
Determine the number of streams (individuals) which flow to the rivers and sea using equations 2.4.10 and 2.4.11 ;
Create randomly initial population ;
Define the intensity of flow (How many streams flow to their corresponding rivers and sea) using equation 2.4.13;
while *(t < Maximum Iteration) or (any stopping condition)* **do**
  for *i* = 1*:Population Size(N_{pop})* **do**
    Stream flows to its corresponding rivers and sea using equations 2.4.14 and 2.4.15 ;
    % Calculate the objective function of the generated stream ;
    if *F New Stream < F river* **then**
      River = New Stream ;
      if *F New Stream < F Sea* **then**
        Sea = New Stream;
      end
    end
    % River flows to the sea using equation 2.4.16 ;
    if *F New River < F Sea* **then**
      Sea = New River ;
    end
  end
  for *i* = 1*: number of rivers (Nsr)* **do**
    if *distance (Sea and River) < d_{max}) or (rand < 0.1)* **then**
      New streams are created using equation 2.4.18 ;
    end
  end
  % Reduce the d_{max} using equation 2.4.19
end
% Post process results and visualization

Algorithm 4: The pseudo code of WCO algorithm

2.4.5 Teaching Learning –Based Algorithm (TLBA)

2.4.5.1 Introduction

Teaching–learning-based optimization (TLBO) is a metaheuristics algorithm proposed by Rao et al. (2011, 2012a, b) and Rao and Savsani (2012). The TLBO algorithm is a population approach (human-based), which simulates the teaching and learning processes within a classroom to solve the problem at hand. The algorithm describes two basic modes of the learning: (i) through teacher (known as teacher phase) and (ii) through interaction with the other learners (known as learner phase[29].

Due to its attractive characters such as simple concept, without the specific algorithm parameters, easy implementation, and rapid convergence, TLBO has captured great attention and has been extended to handle constrained, multi objective, large-scale, and dynamic optimization problems. Furthermore, TLBO has also been successfully applied to many scientific and engineering fields, such as neural network training, power system dispatch, and production scheduling[30].

2.4.5.2 TLBO algorithm Concept

In this optimization algorithm, a group of learners is considered as population and different subjects offered to the learners are considered as different design variables of the optimization problem and a learner's result is analogous to the 'fitness' value of the optimization problem. The best solution in the entire population is considered as the teacher. The design variables are actually the parameters involved in the objective function of the given optimization problem and the best solution is the best value of the objective function. The working of TLBO is divided into two parts, 'Teacher phase' and 'Learner phase'.

In the teacher phase, the learner with the best grade in the population is selected to be the teacher. The teacher is responsible for training the learners and improving the mean grade of the class. In the learner phase, each learner randomly selects a learner to interact with. The ultimate goal is to improve the mean grade of the class at this phase. The above procedures are repeated iteratively until the stopping condition is met[29]

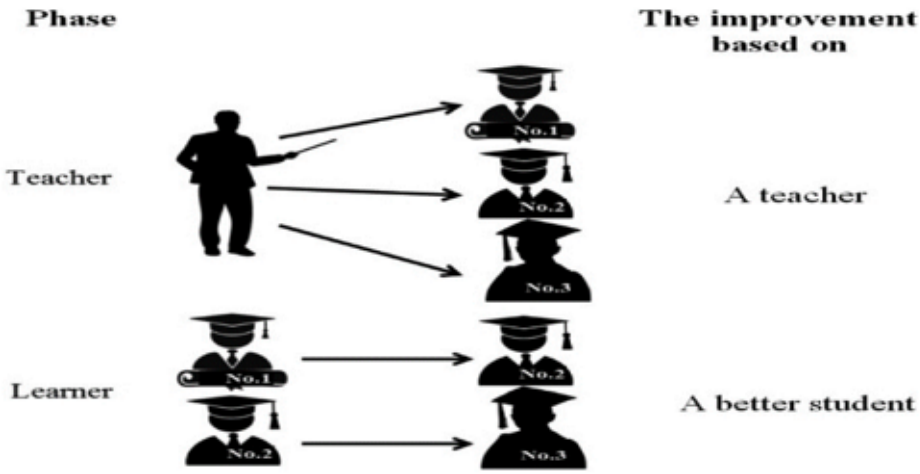


Figure 2.16: Teaching Learning Based Optimization method

2.4.5.3 TLBO algorithm parameters

In order to implement TLBO algorithm, the following steps are involved in the proposed algorithm:

- **Initialize the learner population**

$$X_i = [x_i 1, x_{i2}, \ldots, x_{ik}, \ldots, x_{iM}] \ ; \ i = 1, 2, \ldots, M \tag{2.4.26}$$

Where N is number of problem variables and M is he population size. The amount of facings were generated by the equation 2.4.27 as follow:

$$X_{ik} = LB + rand(0,1) \times (UB - LB) \tag{2.4.27}$$

Where LB and UB respectively denote the upper and lower bounds of Xik, and rand $(0,1)$ denotes a random number in range $[0,1]$.

- **Evaluate the grade of learners** By substituting the learner's $X - ik$(decision variables) value into the objective function ,we can obtain the learner's grade (i.e., the objective value) and then select the learner with the highest grade as teacher and set it to X_{best}.

- **The teacher phase** During this phase, a teacher tries to increase the mean result of the class to his or her level depending on his or her capability. The difference between the teacher and the learners can be as follow:

$$Difference_{Mean} = r\left(X_{best} - T_f M\right) \tag{2.4.28}$$

Where M is the mean grade of the class, X_{best} is the teacher and T_f is the teaching factor, which decides the mean grade to be changed, and r is a random number in the range [0, 1]. The value of T_f can be either 1 or 2, and it is randomly decided with equal probability, as in equation 2.4.29

$$T_f = round\left[1 + rand(0,1)(2-1)\right] \tag{2.4.29}$$

Based on the Difference Mean the existing solution is updated in the teacher phase according to the following expression :

$$X_{ik}^{new} = X_{ik} + Difference_{Mean} \tag{2.4.30}$$

X_{ik}^{new} is accepted if it gives better function value. All the accepted function values at the end of the teacher phase are maintained and these values become the input to the learner phase. The learner phase depends upon the teacher phase[8].

- **The learner phase** Learners are allowed to randomly exchange knowledge with other learners for enhancing his/her knowledge. A learner will learn new information if the other learners have more knowledge than he or she has; a learner does not learn anything new if the other learners do not have more knowledge[28].

Let X_a be the other randomly selected learner. The learning of X from Xa can be mathematically expressed as in equation 2.4.31

$$X_{ik}^{new} = \begin{cases} X_{ik} + r\left(X_{ik} - X_{ik}^{a}\right) & \text{if } f\left(X_{ik}\right) < f\left(X_{ik}^{a}\right) \\ X_{ik} + r\left(X_{ik}^{a} - X_{ik}\right) & \text{if } f\left(X_{ik}\right) \geq f\left(X_{ik}^{a}\right) \end{cases} \tag{2.4.31}$$

Similar to the teacher phase, those with a better grade between the learner and the newly generated learner will be accepted.

- **Termination** Check whether the stopping condition is met, If the condition is met (maximum number of generation is achieved), the algorithm terminates and outputs the current solution, which is the optimal solution for this generation; otherwise, evaluate the grade of all learners for the next teacher phase.

The TLBO algorithm can be summarized as Flow char and pseudo code in Figure 2.17 and Algorithm 6 respectively.

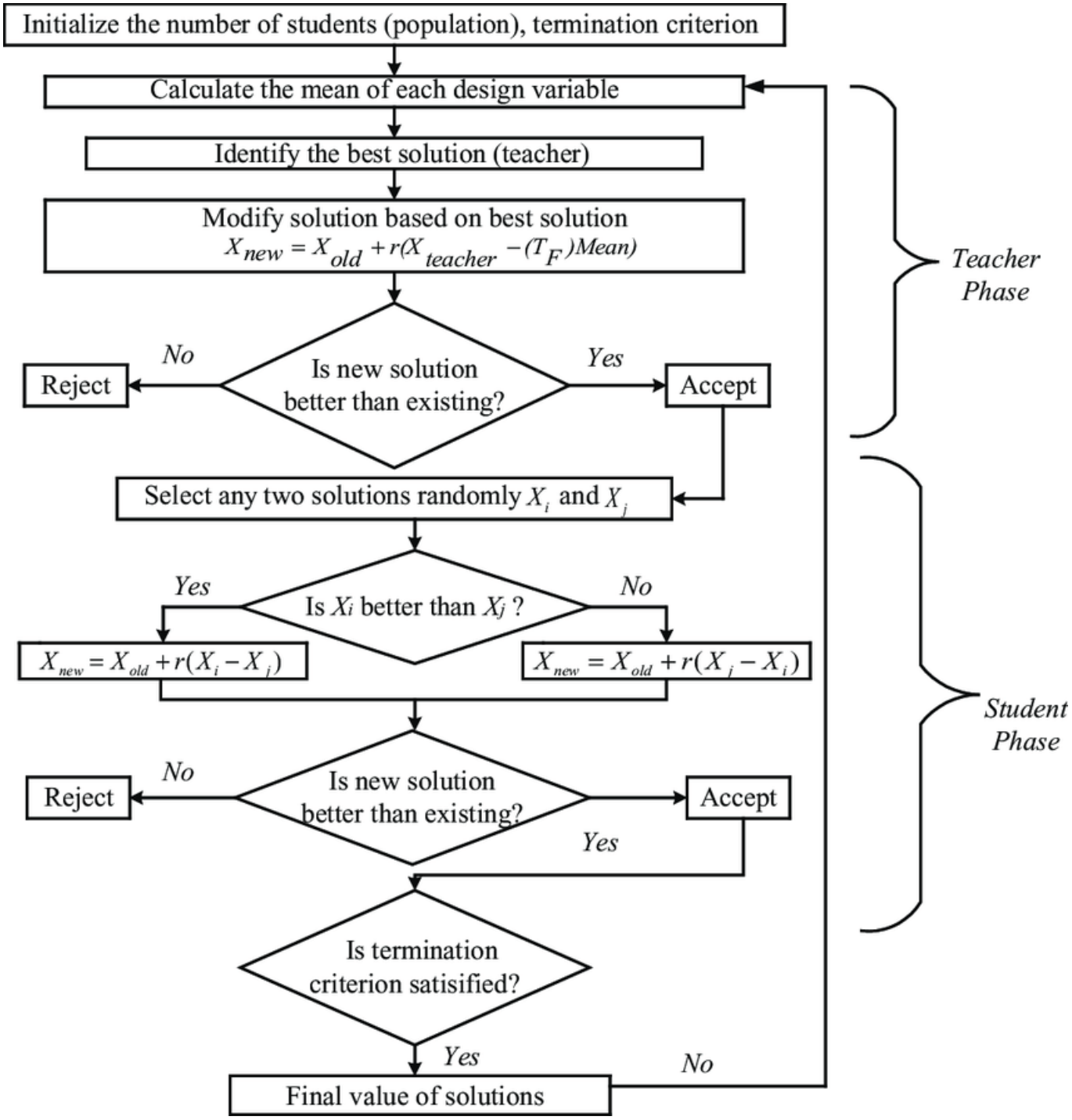


Figure 2.17: Flow Chart of TLBO Algorithm

Initialize population ;
for *individual i =1:M* **do**
  Compute theobjective value $f(x_i)$;
end
while *The stopping condition is not satisfied* **do**
  for $g = 1 : g_{max}$) **do**
    for $i = 1 : M$(*Teaching phase*) **do**
      Genrate the new candidate $X_i^{'}(t)$;
      if $f\left(X_i^{'}(t)\right) \geq f\left(X_i(t)\right)$ **then**
        $f\left(X_i^{'}(t)\right) = f\left(X_i(t)\right)$;
      end
    end
    for $i = 1 : M$(*Learning phase*) **do**
      Genrate the $X_i^{'}(t)$ according to: ;
      if $f\left(X_i^{'}(t)\right) \geq f\left(X_i(t)\right)$ **then**
        $f\left(X_i(t)\right) = f\left(X_i^{*}(t)\right)$;
      end
    end
  end
end

Algorithm 5: The pseudo code of TLBO algorithm

2.4.6 Hill climbing Algorithm (HCA)

2.4.6.1 Introduction

Hill climbing algorithm is a local search algorithm, which continuously moves in the direction of increasing value to find the peak of the mountain or best solution to the problem it terminates when it reaches a peak value where no neighbor has a higher value[31].

2.4.6.2 HC Algorithm Concept

HC algorithm has a node that comprises two parts: state and value. It begins with a non-optimal state (the hill's base or any random state) and upgrades this state until a certain precondition is met (the objective function is used as the basis for this precondition)[31].

When the current state is improved, the algorithm will perform further incremental changes to the improved state until a peak solution is achieved.A hill-climbing algorithm has four main features[31]:

1. It employs a greedy approach: This means that it moves in a direction in which the cost function is optimized (establishing the local maxima or minima).
2. No Backtracking: A hill-climbing algorithm only works on the current state and succeeding states (future), it does not look at the previous states.

3. Feedback mechanism: The algorithm has a feedback mechanism that helps it decide on the direction of movement (whether up or down the hill). The feedback mechanism is enhanced through the generate-and-test technique.
4. Incremental change: The algorithm improves the current solution by incremental changes.

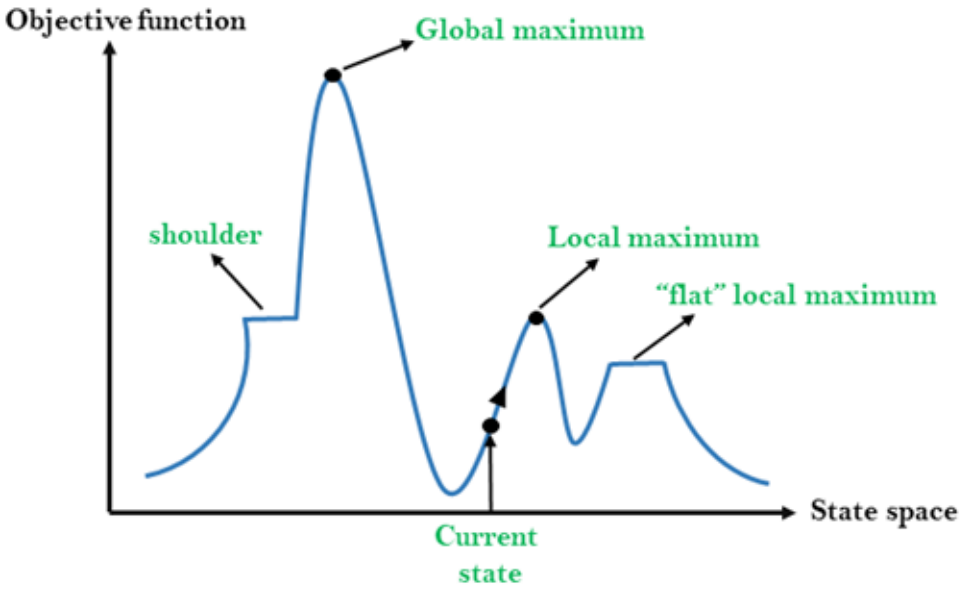


Figure 2.18: state-space diagram of the HC Algorithm

There three Types of Hill Climbing[31]:

- Simple HC
- Steepest-Ascent HC (what we have used)
- Stochastic HC

2.4.6.3 Algorithm for Steepest-Ascent HC

- Step 1: Evaluate the initial state, if it is goal state (desired solution) then return success and stop, else make the current state as your initial state.
- Step 2: Loop (evaluate he objective value of each neighbor and compare to the current state) until a solution is found or the current state does not change.

  Let S be a state such that any successor of the current state will be better than it.

  For each operator that applies to the current state;

  Apply the new operator and generate a new state.

  Evaluate the new state (its objective value).

  If it is goal state, then return it and quit, else compare it to the S.

  If it is better than S, then set new state as S.

  If the S is better than the current state, then set the current state to S.
- Step 5 : Exit.

2.5 Conclusion

This chapter has presented stochastic optimization methods and their classification, which are mainly divided into two categories: heuristic and metaheuristic. After that, six types of metaheuristic optimization methods have been introduced and investigated, refereed to as Particle swarm optimization (PSO), Cuckoo Search Optimization CSO, Moth Flame Optimization, Water cycle WCO, Teaching Learning Based Optimization TLBO and Hill Climibing Optimization.

Chapter 3

PID Controller

3.1 Introduction

Proportional, Integral and Derivative (PID) controller is one of the commonly used control systems in the industry. In process control, the PID control module is a building block which provides the regulation and disturbance rejection for single loop, cascade, multi-loop and multi-input multi-output control schemes (More than 90% of control loops are of PI/PID type) and has remained the most widely used controller until today. In fact, PID controller optimization remains an open challenge. Indeed, specifications, given in terms of frequency or time domain, can hardly be taken into account with a standard deterministic algorithm, in order to overcome these challenges, many academic and industrial efforts are concentrated on Improve PID control performance and robustness, primarily in the areas of tuning rules, identification schemes, and adaptation techniques.

In this chapter, detailed discussion on the fundamental theory that underpins this type of three-term process control. The dynamics associated with each control mode will also be discussed and the advantages and shortcomings associated with each type of control will be given. The tuning of a PID controller is investigated as well, and the Meta-heuristic algorithms defined in the previous chapter is used to tackle the optimization problem.

3.2 The Proportional-Integral-Derivative (PID) Control Theory

The controller is the unit designed to create a stable closed-loop system and achieve some pre-specified dynamic and static process performance requirements. The input to the controller unit is usually an error signal which is the difference between a desired set-point and the actual measured output[32].

The PID controller is a generalization of the lead-leg compensation in the frequency domain[33]. The PID can be classified into three main types according to the effects of setting coefficients on the controller's behavior, the three types are[34] [35]:

(a) **Parallel PID:**

P, I and D are operating independently and then their outputs gathered together as one output. Parallel PID transfer function is generally written in the "parallel form" given by equations 3.2.1 and 3.2.2.

For the time domain:

$$G(t) = K_p + K_I \int_0^t d\tau + K_D \frac{d}{dt} \tag{3.2.1}$$

For the frequency domain:

$$G(s) = K_p + K_I \frac{1}{s} + K_D s \tag{3.2.2}$$

also:

$$G(s) = K \frac{\left(\frac{s}{\omega_c}\right)^2 + 2\delta_c \frac{s}{\omega_c} + 1}{s} \tag{3.2.3}$$

where:

$$\omega_c = \sqrt{\frac{K_I}{K_D}} \tag{3.2.4}$$

$$\delta_c = \frac{K_P}{2\sqrt{K_I K_P}} = K_I \tag{3.2.5}$$

Where K_P is the proportional gain, K_I the integral gain, K_D the derivative gain.

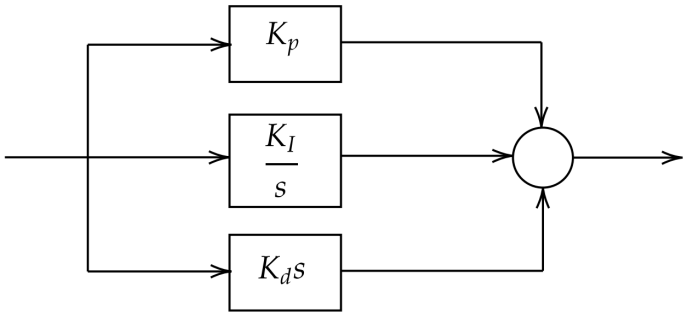


Figure 3.1: Parallel PID Controller

(b) **Series PID:**

PID controller may also be realized in the "series form" if the elements or controllers of D and I are based entirely on the controller P. In other words, they cannot work without the controller P. In this case, the series PID controller can be implemented as a cascade of a PD and a PI controller and its transfer function is in the form of:

For the time domain:

$$G(t) = K_p \left(\alpha + T_D \int_0^t d\tau\right) . \left(1 + \frac{1}{\alpha T_I} \frac{d}{dt}\right) \tag{3.2.6}$$

For the frequency domain:

$$G(s) = K_p (\alpha + T_D s) \left(1 + \frac{1}{\alpha T_I s}\right) \tag{3.2.7}$$

where:

$$\alpha = \frac{1 \pm \sqrt{1 - \frac{4T_D}{T_I}}}{2} > 0 \tag{3.2.8}$$

Where T_I the integral time constant and T_D the derivative time constant.

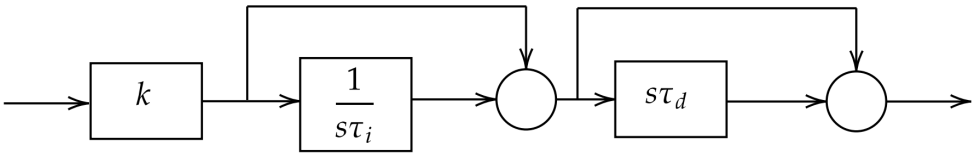


Figure 3.2: Series PID Controller

(c) **Mixed PID (Standard PID):**

It is combination of the two types (Serial and Parallel) where any modification of the Kp coefficient will affect the proportional, integral and derivative actions. Standard PID transfer function is generally written as:

For the time domain:

$$G(t) = K_p \left(1 + \frac{1}{T_I}\int_0^t d\tau + T_D \frac{d}{dt}\right) \tag{3.2.9}$$

For the frequency domain:

$$G(s) = K_p \left(1 + \frac{1}{T_I s} + T_D s\right) \tag{3.2.10}$$

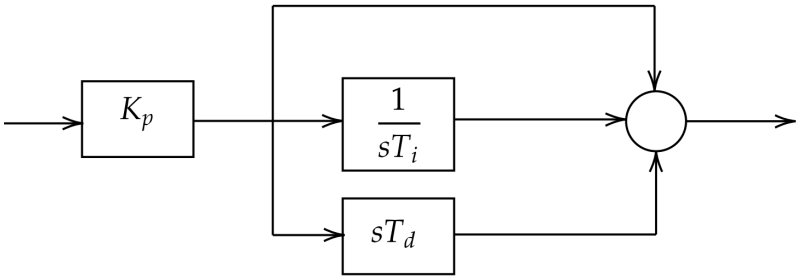


Figure 3.3: Standard PID Controller

The actuator command developed by a PID controller is a weighted sum of the error signal and its integral and derivative. PID stands for the Proportional, Integral, and Derivative terms that sum to create the controller output signal.
For the time domain:

$$U_c(t) = K_p e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{d}{dt}e(t) \tag{3.2.11}$$

For the frequency domain:

$$U_c(s) = \left[K_p + \frac{K_I}{s} + K_D\right] E(s) \tag{3.2.12}$$

3.3 Why using the PID Controller

As the name suggests, PID controller consists of three basic coefficients : proportional, integral and derivative which are varied to get optimal response closed loop systems, the theory of PID and the effects of tuning a closed loop control system are:

- The proportional control used to increase the speed of the response, and to decrease the steady-state error and relative stability.
- The integral control used to eliminate the steady-state error and to decrease the relative stability.
- The derivative control used to increase the relative stability and the sensitivity to noise.

3.4 Control Effects of Proportional, Integral and Derivative Action

The "three-term" functionalities are highlighted by the following:

(a) **The proportional term:**

Proportional control is denoted by the P-term in the PID controller .The output of a proportional controller varies proportionally to the system error according to equation 3.4.1. The proportional response can be adjusted by multiplying the present value of error by a fixed adjustable value K_P which called proportional gain constant (Proportional control action responds to only the present error). The proportional term is given by:

Time domain:

$$U_c(t) = K_P e(t) \tag{3.4.1}$$

Frequency domain:

$$U_c(s) = K_P E(s) \tag{3.4.2}$$

Where $U_c(t)$ is the output of the proportional control, K_P is the proportional gain constant (adjustable), $e(t)$ is the error value which equal to $e(t) = r(t) - y(t)$.

If K_P the (proportional gain) is very high, the system can become unstable, however if K_P is very low, the control action may be too small when responding to system disturbances. Consequently, a proportional controller will have the effect of reducing the rise time, but never eliminate the steady-state error (unless the process has naturally integrating properties).

In some control systems, different description used instead of proportional gain which is proportional band which denoted by $PB\%$. The equation 3.4.3 shows the relationship between the proportional gain and proportional band.

$$K_P = \frac{100}{PB\%} \tag{3.4.3}$$

(b) **The integral term:**

Integral control is denoted by the I-term in the PID controller, it can be called also reset because of its ability to remove the remaining error which could not be removed by the proportional control. The integral in a PID controller is the sum of the instantaneous error over time and gives the accumulated offset that should have been corrected previously (overcomes the shortcoming of proportional control)[36]. The action of the integral controller is based on the principle that the control action should exist as long as the error is different from zero because of the error is multiplied by the integral gain and added to the controller output. Integral control can be seen as continuously looking at the total past history of the error by continuously integrating the area

under the error curve and reducing any offset as result, The greater the error signal the larger the correcting action from the integral controller will be. The integral term is given by:

Time domain:

$$U_c(t) = K_I \int_0^t e(\tau) d\tau \tag{3.4.4}$$

Frequency domain:

$$U_c(s) = \frac{K_I}{s} E(s) \tag{3.4.5}$$

where:

$$K_I = \frac{1}{T_I} \tag{3.4.6}$$

Where T_I is the integral time constant, K_I is the gain of the integral controller, $e(t)$ is the instantaneous error signal.

In the time domain, the effects over the transient response consist of the decrease of the rise time (T_r) and the increase of the settling time (T_s) and the overshoot (O), also it will have the effect of eliminating the steady-state error, but it may make the transient response worse. In the complex plane, the effects of the integral action consist of a displacement of the root locus of the system towards the right half-plane[33].

In fact, the proportional and integration controllers (PI Controller) are widely used in industrial applications to accelerate the process and remove the error during stability. However, the integration controller may cause an overshoot above the desired value because of collecting the data of accumulated errors.

(c) **The derivative term:**

Derivative control is denoted by the D-term in the PID controller .The derivative term is also called the rate of change because of the derivative of the process error is calculated by determining the slope of the error over time and multiplying this rate of change by the derivative gain K_D[36] this introduces an element of prediction into the control action. The time and Laplace domain representations for derivative control are given as:

Time domain:

$$U_c(t) = K_D \frac{d}{dt} e(t) \tag{3.4.7}$$

Frequency domain:

$$U_c(s) = K_D s E(s) \tag{3.4.8}$$

where: Where K_D is the derivative control gain.

In the time domain, a decrease in the overshoot and the settling time, in the complex plane the derivative action produces a displacement of the root locus of the system towards the left half-plane[33].

In short, K_P can be increased to speed up the response but to a value that keeps the system stable, K_I for speeding up the error elimination process and KD for reducing the overshoot with considering the noise, (Derivative control is affected by noise that may cause a lack in stability of the system). However, some of these actions have undesired behaviors. The following table summarizes the most important positive and negative effects of PID actions.

The Effects of each controller K_P, K_D, and K_I on a closed-loop system performances are summarized in the table shown below in table 3.2.

Table 3.1: Positive and Negative effects of PID Actions

	Positive Effects	Negative Effects
Proportional Action	Speed up the response	Increased overshoot
Integral Action	Elimination of the steady state errors (the infinite gain at zero frequency)	Decreased relative stability (the $\frac{\pi}{2}$ phase lag introduced)
Derivative Action	Increased relative stability(the $\frac{\pi}{2}$ phase lead introduced)	Increased sensitivity to high-frequency noise (by the increasing gain with slope of 20 dB/dec)

Table 3.2: The Effect of PID controller parameters in a closed-loop system Performances

Parameter	Rise Time	Overshoot	Settling Time	Steady-state error
K_P	Decrease	Increase	Small Change	Decrease
K_I	Decrease	Increase	Increase	Decrease Significantly
K_D	Minor Decrease	Minor Decrease	Minor Decrease	No effect (Theoretically)

In fact, changing one of these variables can change the effect of the other two. For this reason, table 3.2 should be used only as a reference when we are determining the values for K_P, K_D, and K_I.

3.5 PID Controller tuning methods

The dynamic characteristics of the process control loop will cause the operating conditions in the loop to change, and hence change in loop performances. Changes in system performances may because of the existence of process nonlinearities in the control channel, changes in production strategies and changes in equipment maintenance cycles In order to ensure the continued satisfactory performances of the control loop, many researches have conducted to developing tuning methods for PID controllers However, there is still no well-established general tuning algorithm[33].

The goal of PID controller tuning is to determine parameters that meet closed loop system performance specifications, and the robust performance of the control loop over a wide range of operating conditions should be ensured[37]. Over the years, several heuristic methods that take into consideration the nature of the dynamics present within a process control loop have been proposed, all these methods are based upon the dynamical behavior of the system under either open-loop or closed-loop conditions for the tuning of PID controllers. The first method used the classical tuning rules proposed by Ziegler and Nichols. In other hand, in order to increase the capabilities of PID controllers by adding new features, many artificial intelligence (AI) techniques have been used to improve the controller performances for a wide range of plants while retaining their basic characteristics. All

techniques such as neural network, fuzzy system, and neural-fuzzy logic have been widely applied to proper tuning of PID controller parameters [38].

3.5.0.1 Performance Criteria For Closed-Loop Systems

The function of a feedback control system is to ensure that the closed loop system has desirable dynamic and steady state response characteristics. Ideally, we would like the closed-loop system to satisfy the following performance criteria:

- The closed-loop system must be stable.
- The effects of disturbances are minimized, providing good disturbance rejection.
- Rapid, smooth responses to set-point changes are obtained, that is, good set-point tracking.
- Steady-state error (offset) is eliminated.
- Excessive control action is avoided.
- The control system is robust, that is, insensitive to changes in process conditions and to inaccuracies in the process model.
- Smallest steady state error and percentage overshoot.
- Fastest rising and settling time.

3.5.0.2 The proposed Tuning Methods

In our project, we have used the two types of stochastic optimization methods: heuristic method, which represented by Zeigler-Nichols rule and Meta-heuristic, which represented by six optimization algorithms. Each algorithm refers to specific optimization class. The optimization algorithms are PSO,CSO, MFO,WCO, TLBO and HCO.

1. **The Ziegler-Nichols PID tuning rule**

   A very useful empirical tuning formula was proposed by Ziegler and Nichols in early 1942 where they described two methods for tuning the parameters of P-, PI- and PID controllers. These two methods are the Ziegler Nichols' closed loop method, and the Ziegler-Nichols' open loop method , these two methods use an on-line process experiment followed by the use of rules to calculate the numerical values of the PID coefficients[39]. In our project closed-loop method is presented.

   Ziegler and Nichols' closed loop method can be applied only to processes having a time delay or having dynamics of order higher than 3. It used the following definition of acceptable stability as a basis for their controller tuning rules: "The ratio of the amplitudes of subsequent peaks in the same direction (due to a step change of the disturbance or a step change of the set-point in the control loop) is approximately $\frac{1}{4}$ "[40].

$$\frac{A_2}{A_1} = \frac{1}{4} \tag{3.5.1}$$

However, there is no guaranty that the actual amplitude ratio of a given control system becomes $\frac{1}{4}$ after tuning with one of the Ziegler and Nichols methods, but it should not be very different from $\frac{1}{4}$[40].

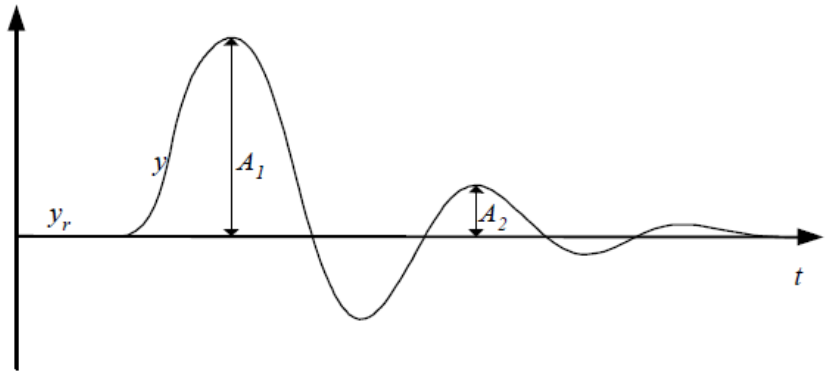


Figure 3.4: Ziegler-Nichols stability study. (According to Ziegler-Nichols, the ratio $\frac{A_2}{A_1} = \frac{1}{4}$ so the system is stable)

2. **PID controller parameters Design Using ZN Rule**

The idea was to tune the controller based on doing a simple experiment, extract some features of process dynamics from the experimental data and determine controller parameters from the features.

The tuning procedure is as follows:

(a) Turn the PID controller into a P controller by setting: $K_I = 0$ and $K_D = 0$.

(b) Increase K_P until there are sustained oscillations in the signals in the control system (that means the system being on the stability limit).The resulted K_p value is called the ultimate (or critical) gain, K_U.

(c) Measure the ultimate (or critical) period P_U of the sustained oscillations as it is shown in Figure 3.5.

(d) Calculate the controller parameter values (K_P, K_I and K_D) according to Table 3.3.

(e) Use these parameter values in the controller.

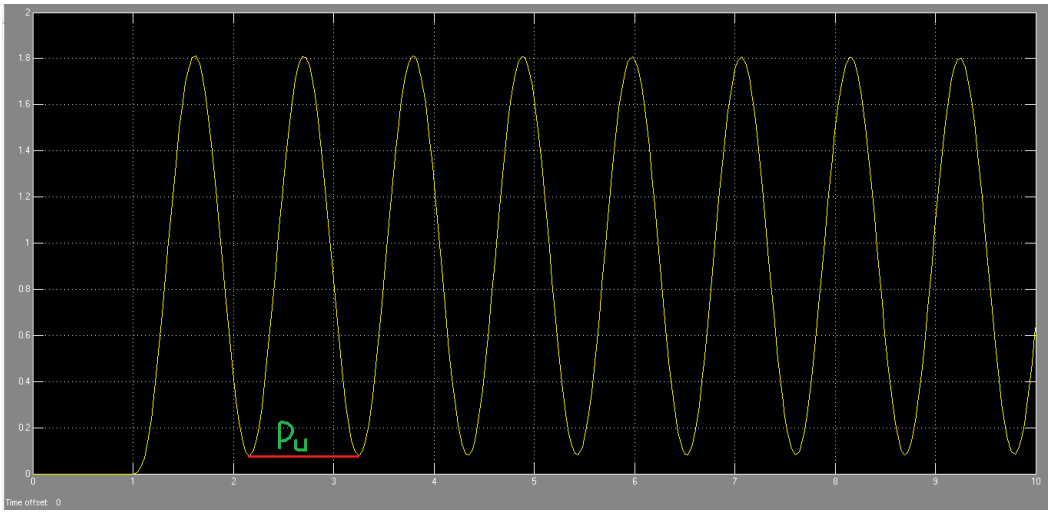


Figure 3.5: The ultimate period measurement of ZN closed loop tuning method

Using the parameters K_{pu} and P_u, we can set the values of K_P, K_I and K_D according to the formula shown in Table 3.3.

Table 3.3: Controller parameters for closed loop Ziegler-Nichols method

Controller type	K_p	Ti	T_D	Ki	K_D
P	$0.5K_U$	-	-	-	-
PI	$0.45\,K_U$	0.8 Pu	-	$0.54\,K_U$ / Pu	-
PD	$0.8\,K_U$	-	0.125 Pu	-	0.1.Ku.Pu
PID	$0.6\,K_U$	0.5 Pu	0.125 Pu	$1.2\,K_U$ / Pu	0.075.Ku.Pu

3.6 The design process of the proposed algorithms

As we discussed in chapter 2, the general structure of the proposed algorithms, how it is being established, programmed and created. In this section, we discuss them as an alternative methods for tuning the PID controller. The design process of each of PSO-PID, MFO-PID, WCO-OPID, TLBO-PID, -PID and HCO-PID controller for AVR system is summarized as follows:

- Step1: Determine the parameters of AVR system.
- Step2: Set the parameters of the proposed algorithms.
- Step3: Specify the lower and upper bound of the three controller parameters (The lower and upper bound set depending on the algorithm).
- Step4: Set $t = 0$ and randomly initialize the position of K particle(s) within the range of each controller parameter (HCO is an individual-based algorithm so it uses a single particle).
- Step5: Test the stability of the closed-loop system.
- Step6: Calculate overshoot M_P, steady state error E_{ss}, raise time T_r and settling time T_s from the step response of closed-loop system.
- Step7: Evaluate the fitness value of each particle using the equation of fitness function (we have used more than one)
- Step8: Compare each individual's evaluation value with its personal best; the best evaluation among them is denoted as G_{best}.
- Step9: Modify the particle position of each individual K according to a specified algorithm.
- Step10: If the number of iteration reaches the maximum, then go to step 11, otherwise go to step 5.

- Step11: The latest G_{best} is the optimal controller parameter.

Figure 3.6 illustrates the basic block diagram of the proposed Optimization algorithm (O.A) based PID controller tuning.

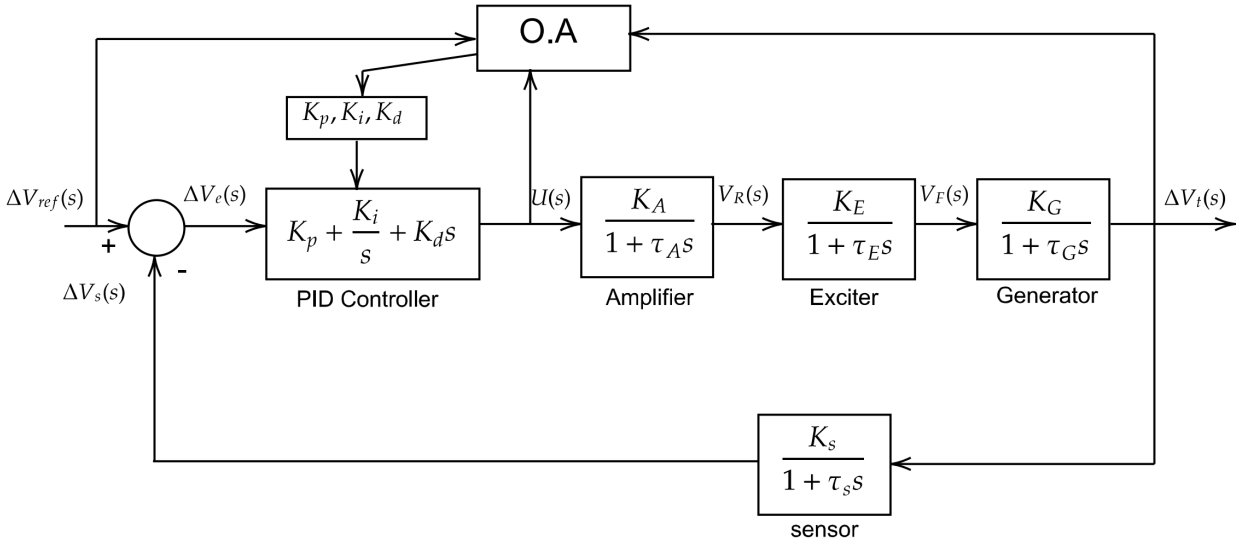


Figure 3.6: Proposed optimization algorithms based PID tuning in AVR system

3.7 Conclusion

In this chapter, it has been covered the characteristics of PID control. The types and roles of PID controllers have been described. Next, the tuning process of PID controllers has been explained. Finally, tuning techniques using Ziegler Nichols and the six metaheurstique optimization methods have been presented, with the steps taken by those techniques to obtain suitable PID parameters.

Chapter 4

Simulation and results

4.1 Introduction

The main objective of this work is to design and implement an efficient PID controller for AVR system in a synchronous generator. In the proposed approach, the design problem is formulated as an optimization problem control and the proposed optimization algorithms are employed to search for optimal controller parameters.

The effects of these objective functions on the system's performance in terms of maximum overshoot, settling time, rising time, and peak time are investigated. The suggested controller's performance was also evaluated using a variety of analytical methodologies, including transient response analysis, stability verification, mean running time analysis, convergence, and robustness analysis. The results of the research are also compared to other recently published optimization algorithms to demonstrate the efficacy of the suggested methodologies and objective functions. The performance of the proposed controller has also been done employing various analysis methods such as transient response analysis, stability verification, mean running time analysis ,convergence .... In addition, robustness analysis of the proposed AVR system is performed by varying the time constants and the gains of amplifier, exciter, generator and sensor.

4.2 Uncontrolled AVR Performance

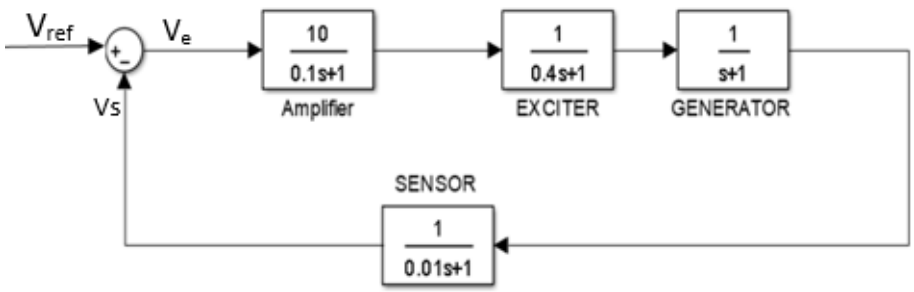


Figure 4.1: Block diagram of an uncontrolled AVR system

The uncontrolled system transfer function is:

$$G(s) = \frac{0.1s + 10}{0.0004s^4 + 0.0454s^3 + 0.555s^2 + 1.51s + 11} \tag{4.2.1}$$

4.2.1 Simulation results without PID Controller

By applying a step input signal to the system, its response is depicted in the Figure 4.2

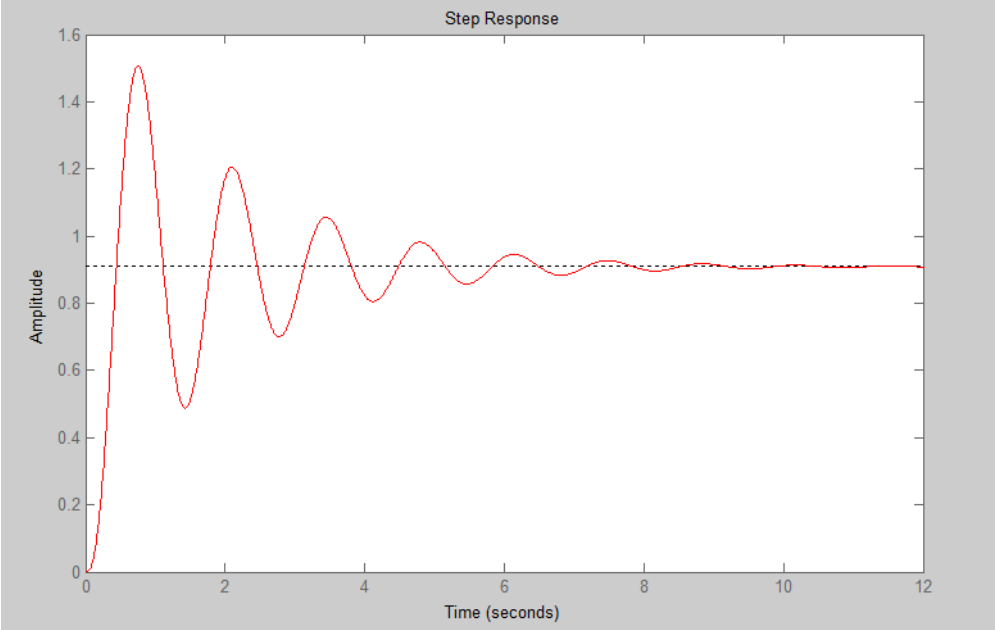


Figure 4.2: Output voltage step response of AVR system in the absence of controller

The uncontrolled system performances are summarized in the following table.

Table 4.1: Terminal voltage step response of an AVR system without PID controller

Rise Time (s) T_r	Settling Time 2% (s) T_s	Overshoot $M_p(\%)$	Steady State Error E_{ss}
0.261	6.99	65.7	0.091

From Figure 4.2, the step voltage response is highly oscillatory and has a steady state error , with long settling time with 6.99 sec and large overshoot, for example, in providing a 220V operating apparatus with 364.54V can cause irreversible damage.

It is obvious that the dynamic response of the AVR system needs to be improved and the steady state error should be eliminated by using a PID controller.

4.3 Controlled AVR Analysis

4.3.1 Simulation Results of ZN Optimization Method

AVR system with a PID controller tuned using ZN method. By following the steps defined in section 3.6, we have measured the ultimate gain K_U and ultimate period P_U and by using the formulas given in Table 3.3 we calculated the PID controller parameter, the results are in the following Table:

Table 4.2: The calculated PID controller parameters using closed loop ZN method

K_U	P_U	K_P	K_I	K_D
1.7	1.0963	1.02	1.861	0.1163

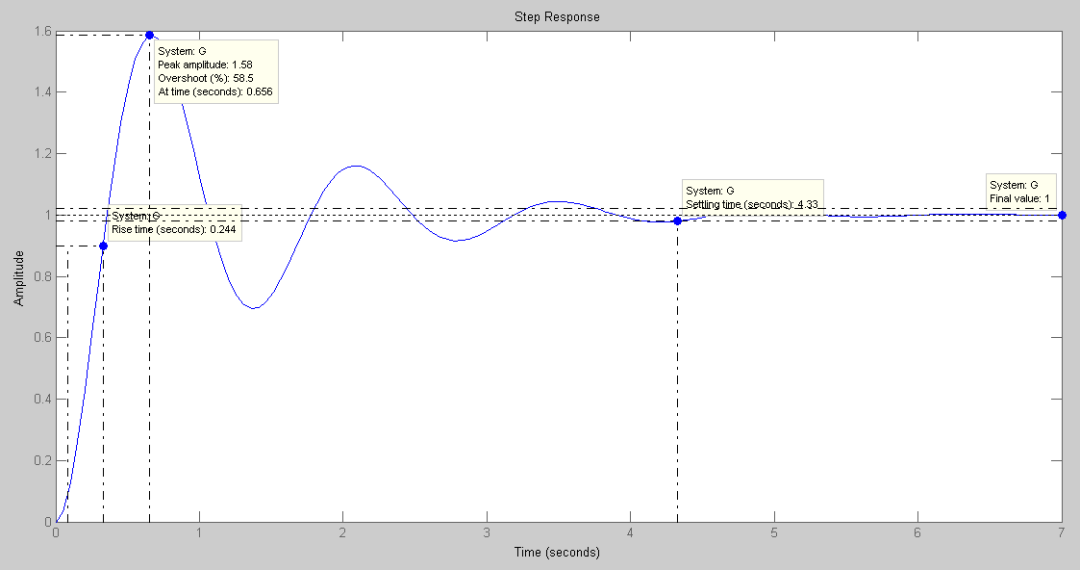


Figure 4.3: Step response of the controlled AVR system with ZN tuning method

After performing the simulation of this case, we find that the performances criteria values are:

Table 4.3: Controlled AVR system Performances

Rise Time (s) T_r	Peak Time (s) T_P	Settling Time (s) T_s	Overshoot $M_p(\%)$	Steady State Error E_{ss}
0.244	0.656	4.33	58.5	0

After tuning PID controller of AVR system, we deduce the following points:

- This method did not give the desired results (very bad system performances).
- It forced the process into a condition of marginal stability in the transient state (High Overshoot) that may lead to unstable operation due to the external disturbances.
- This method is not applicable for the AVR system due to the high Over-Voltage.

4.3.2 Performance Estimation of PID Controller

In control system design and analysis or for optimal control purposes, performance indices are calculated to be used as quantitative measures to evaluate a system's performance, where a control system is judged as an optimum system when the system parameters are adjusted so that the index used in the design reaches its minimum value, while constraints of the controlled system are respected[41]. The commonly used indices are the integrated absolute error (IAE), the integral of squared-error (ISE), the integrated of time-weighted-squared-error (ITSE) and the integrated of time-weighted-absolute-error (ITAE). They are often employed in control system design because it can be evaluated analytically in the frequency domain However; the four integral performances criteria in the frequency domain have their own advantages and disadvantages. For example, a disadvantage of the IAE and ISE criteria is that its minimization can result in a response with relatively small overshoot but a long settling time because the ISE performance criterion weights all errors equally independent of time. Although the ITSE performance criterion can overcome the disadvantage of the ISE criterion[33].

The fitness function is special type of objective function that minimize or maximize some preference goals[42]. In order to find an optimum PID controller parameters K_P, K_I and K_D that guarantee our project objectives, many fitness functions have been used in our simulations, Table 4.2 provides some of them. Several performance criteria have been used to form the objective functions to be minimized:

$$\text{Integral of Absolute-error } IAE = \int_0^T |e(t)|\,dt \tag{4.3.1}$$

$$\text{Integral of squared-error } ISE = \int_0^T e^2(t)dt \tag{4.3.2}$$

$$\text{integral time squared-error } ITSE = \int_0^T te^2(t)dt \tag{4.3.3}$$

$$\text{Integral time absolute-error } ITAE = \int_0^T t\,|e(t)|\,dt \tag{4.3.4}$$

Where the error $e(t)$ is the difference between the set-point $r(t)$ and the output $y(t)$; $e(t) = r(t) - y(t)$.

Some of the cost functions have scalar weighting factors, for example in the cost function $J5$[14], β is the weighting factor, we can set $\beta$to be larger than 0.7 to reduce the overshoot and steady-state error and we can set β to be smaller than 0.7 to reduce the rise time and settling time, in our project β is set in the range of The $J6$, $J7$, $J8$, $J9$, $J10$ and $J11$ cost functions have other scalar weighting factors $w_i(i = 1,2\ldots 11)$ which have the effect on increasing or reducing the performances criteria. For example, in the cost function $J7$ if reducing settling time is the most significant issue, we should use w2 big to significantly weight it in the cost function $J7$. Similarly, if reducing the Overshoot is more important, we should choose a large value for w_1.

Therefore, a control design with a trial performance index J is usually required ,the computation of the optimal controller parameters is followed by the simulation run to see how the system responds to this controller. If the response is not satisfactory, the process is repeated with a new J and other parameters and control weightings. After several after several repetitions of optimization repetitions, the optimum parameters of the controller is applied to the actual system to obtain a satisfactory performance criteria.

Table 4.4: Some used fitness functions

Names	Objective functions
$J1$	ISE
$J2$	IAE
$J3$	$ITSE$
$J4$	$ITAE$
$J5$	$W_K = (1-e^{-\beta}) * (PO + E_{ss}) + e^{-\beta} * (T_s - T_r)$
$J6$	$w_1 * PO + w_2 * log(T_s/0.01) + w_3 * log((T_s - T_r)/0.1)$
$J7$	$ITAE + ITSE + ISE + IAE + w_4 * e^{T_s - T_r} + w_5 * e^{PO} + w_6 * e^{T_s}$
$J8$	$IAE * (1-e^{-w_7}) * (PO + 100 * E_{ss}) + e^{-w_7} * (T_s - T_r))$
$J9$	$w_8 * e^{E_{ss}^2 + PO^2 + PU^2} + w_9 * (T_s - T_r) + e^{ITAE + ITSE}$
$J10$	$e^{\lvert E_{ss}\rvert + w_{10} * \lvert PO\rvert} + (T_s - T_r)$
$J11$	$e^{\lvert E_{ss}\rvert + w_{11} * \lvert PO\rvert} + (T_s - T_r)$
$J12$	$ITAE + 10PO + T_s$

Where K is $[K_P\ K_I\ K_D]$.

4.3.3 Algorithm Settings

The following tables provide some used algorithms with their corresponding settings.

Table 4.5: Some used fitness functions

PSO	CSO	MFO	WCO	TLBO	HCO
Number of iteration Max_Iter = 50	Max_Iter=2000	Max_Iter= 50	Max_Iter=50	Max_Iter=50	-
Number of Population N= 50	Number of nests N=25 ;	N =50	N=50	N=50	-
Number of Trials T =10	T =10	T=10	T=10	T=10	T=10
Acceleration coefficent (c1 =2 , c2= 2)	Constant β = 1.5	shape of the logarithmic spiral Constant b=1	Acceleration coefficent c=2	Teaching Factor TF = randi([1 2])	Initial position = [0.6 0.4 0.2]
Inertia weight(Wmax=0.9 , Wmin=0.4)	Abandon Probability Pa=0.25	-	Number of rivers Nsr= 5	-	stepSize = 0.01
-	-	-	Evaporation condition constans dmax =1e(-16)	-	-

Table 4.6: Range of the three controller parameters

Controller parameters	Lower bound	Upper bound
K_P	0.1	1.9
K_I	0.1	1.9
K_D	0.1	1.9

4.4 Simulation results and discussion

All simulations for the all optimization techniques encodings were made using the MATLAB/Simulink program (R2013b) on an Intel i5-5300 2.30 GHz processor with a 4 GB RAM computer. The important results of this study are shown in the following subsections.

4.4.1 Simulation results using PSO PID controller

After running several trials, setting the PSO parameters and the lower and upper bounds of the three control parameters where K is $[K_P\ K_I\ K_D]$ that was mentioned in Table 4.4 and 4.5 respectively, ultimately we have obtained the optimal solutions. The comparative simulation results obtained for the terminal voltage step response of the AVR system at different objective functions are shown in Figure 4.4 The comparative transient response analysis with performance characteristics of ts , T_r and $M_P\%$ is s given in Table 4.8

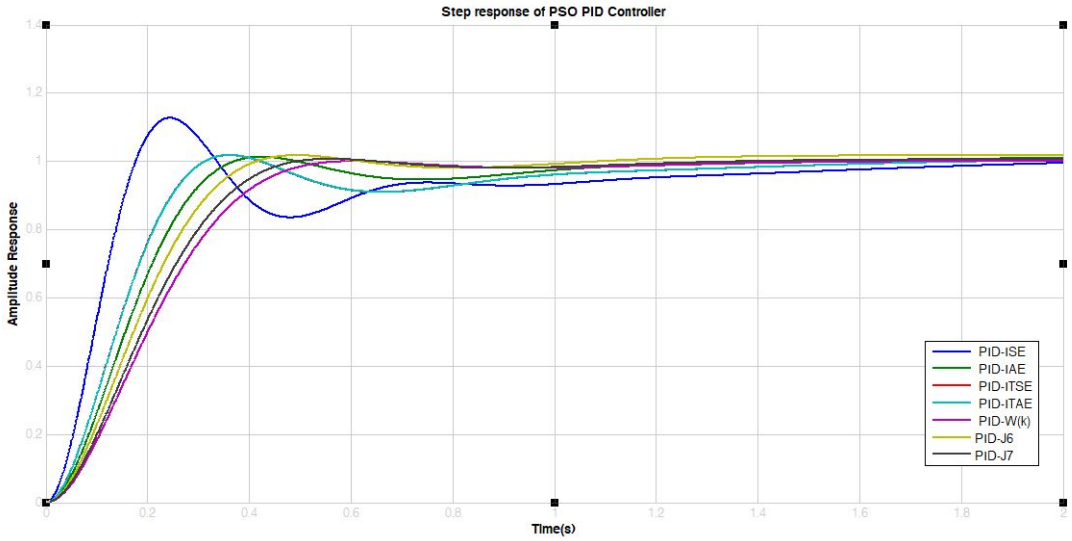


Figure 4.4: Terminal voltage step response of the AVR system with PSO-PID at different fitness functions

Table 4.7: PID gain and time response specifications for various objective functions (PSO based tuning)

PID-PSO/Obj fct	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-ISE	0.7001	0.7010	0.7003	**0.1187**	3.9009	12.67
PID-IAE	0.7006	0.5602	0.3096	0.2287	1.0642	1.2113
PID-ITSE	0.7011	0.54704	0.3791	0.1996	1.349	1.73
PID-ITAE	0.7027	0.5471	0.37852	0.1969	1.3466	1.718
PID-W(k) , β=0.7298 –J (5)	0.6057	0.43203	0.208	0.3154	0.489	**0**
PID-J6	0.6754	0.60005	0.2635	0.2567	0.3841	1.1922
PID-J7	0.6361	0.4824	0.2273	0.2908	0.4258	0.6582

4.4.1.1 Obeservation and analysis results

Based on the comparative performance analysis and performance of various objective functions, the results for the terminal closed loop step response of first set of objectives (ISE, IAE, ITSE, ITAE) are almost the nearly equal, giving us a good rise time ranging from 0,11s to 0,19s and greater settling time varying from 1,06 to 3,90 with small overshoot ranging from 1,71 % to 12,67 % . For the second set of fitness functions (J5, J6, J7) it gave us good results compared to the first set when concerning the settling time and the percentage overshoot , the settling time is shortened with these fitness functions, ranged from 0.38s to 0.48s, and the produced overshoot with J-6 and J-7 is tended to be zero a and no oscillation (MP= 0,00 %) with $w(k)$.

Now by comparing the result obtained by these two sets of fitness functions, it can be seen that the second set have better performance in terms of overshoot and settling time . The first set, on the other hand, provides a faster rising time and a slightly higher overshoot, which is highly undesirable for our system.

From the Table 4.7 it can be clearly seen that the performance of the PSO based PID with objectives functions $w(k)$ and J6 has yielded optimum response with no oscillation and shorter settling time .

4.4.1.2 Comparison with recent and reference work

In order to demonstrate the effectiveness of our proposed PSO-PID controller used in conjunction with the fitness function $J5$, the obtained time domain simulation responses were compared with some recent publications that used to design PSO-PID controller for the same AVR system .

Table 4.8 illustrates the best results produced by our program, as well as some performance results from recent papers on the same subject.

Table 4.8: Comparison of PSO Performance recent/referenced work

PID-PSO	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
$PSO^{J5} - PID$	0.6057	0.43203	0.208	0.3154	0.489	**0**
PID-PSO[43]	0.6770	0.5189	0.2287	0.2762	0.4018	1.12
PID-PSO[14]	0.6254	0.4577	0.2187	0.3070	0.4528	0.4

The optimal step response of the AVR system controlled by PSOJ5-PID controller compared with recent publications is shown in Figure 4.5

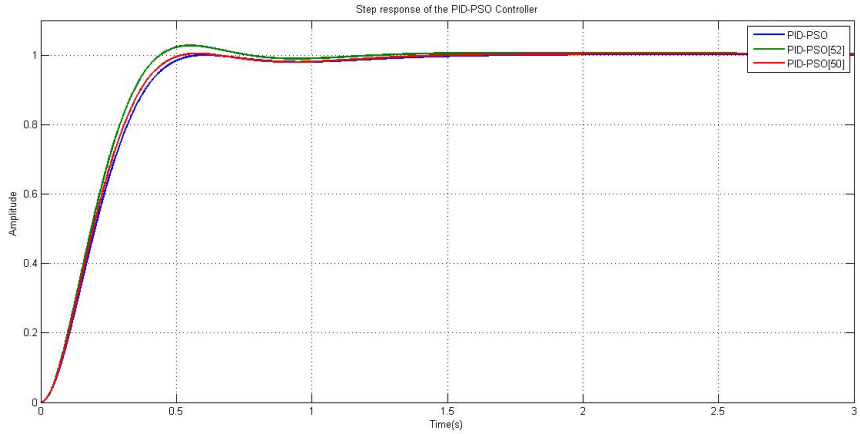


Figure 4.5: Optimal step responses by PSO-PID Comparison to recent/referenced work

The proposed $PSO^{J5} - PID$ controller has good performance when compared to other recent studies, as shown in Table 4.8 and Figure 4.5. Our AVR system has less oscillation structure than the other performances. We can see that our performance results are indeed promising. This indicates that we have achieved our goal of optimizing the AVR based PSO-PID tuning optimization technique.

4.4.2 Simulation Results using CSO PID controller

We finally arrived at the optimal solutions after performing multiple trials and setting the PSO parameters as well as the lower and upper bounds of the three control parameters $[K_P\ K_I\ K_D]$ that were given in Table 4.5 and Table 4.6, respectively.

The comparative simulation results obtained for the terminal voltage step response of the AVR system at different fitness functions are shown in Figure 4.6

The comparative transient response analysis with performance characteristics of T_s, T_r and $M_p\%$ is s given in Table 4.9

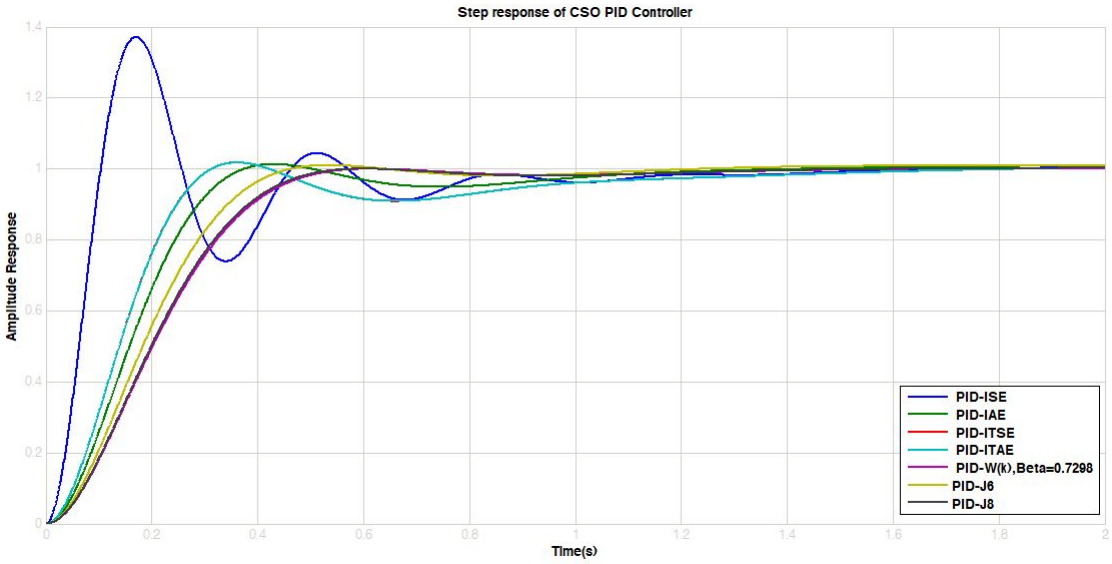


Figure 4.6: Terminal voltage step response of the AVR system with CSO-PID at different fitness functions

Table 4.9: PID gain and time response specifications for various objective functions (CSO based tuning)

PID-CSO/Obj-fct	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-ISE	1.2522	1.5000	1.3939	0.0699	1.1548	36.9822
PID-IAE	0.70000	0.56115	0.30473	0.2313	1.0529	1.2731
PID-ITSE	0.70012	0.54704	0.37941	**0.1965**	1.3503	1.7367
PID-ITAE	0.6999	0.54672	0.37904	0.1967	1.3498	1.7266
PID-W(k) , β=0.7298	0.601758	0.43097	0.2058	0.3183	0.4950	**0**
PID-J6	0.6500	0.51941	0.2394	0.2787	**0.4218**	0.9847
PID-J8	0.6074	0.43788	0.2103	0.3129	0.4858	**0**

4.4.2.1 Obeservation and analysis results

Based on a comparison of performance analysis and performance of various objective functions, the results for the terminal closed loop step response of the first set of objectives (ISE, IAE, ITSE, ITAE) are almost similar, apart from ISE, which produced a greater overshoot M_P=36,98%. They gave us a good rise time ranging from 0,06s to 0,19s and just a little high settling time varying from 1,15s to 1,35s with overshoot ranging from 1,71 % to 36,98 % . For the second set of fitness functions (J5, J6, J8) it gave us good results compared to the first set when concerning the settling time and the percentage overshoot , the settling time is shortened with these fitness functions, ranged from 0.42s to 0.48s, furthermore the produced overshoot with J6 is tended to be zero a and no oscillation (M_P=

0,00 %) with $w(k)$ and J8.

From Table 4.9 it can be clearly seen that the performance of the PSO based PID with objectives functions w(k) , J6 and J8 has yielded optimum response with no oscillation and shorter settling time.

4.4.2.2 Comparison with recent and reference work

Table 4.10 illustrates the best results produced by our program, as well as some Performance results from recent papers on the same subject.

Table 4.10: Comparison of CSO performance to recent/referenced work

PID-CSO	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
$PID-CSO^{J8}$	0.6074	0.43788	0.2103	0.3129	0.4858	**0**
PID-CSO[44]	0.6198	0.4165	0.2126	0.3080	0.4260	0.0001

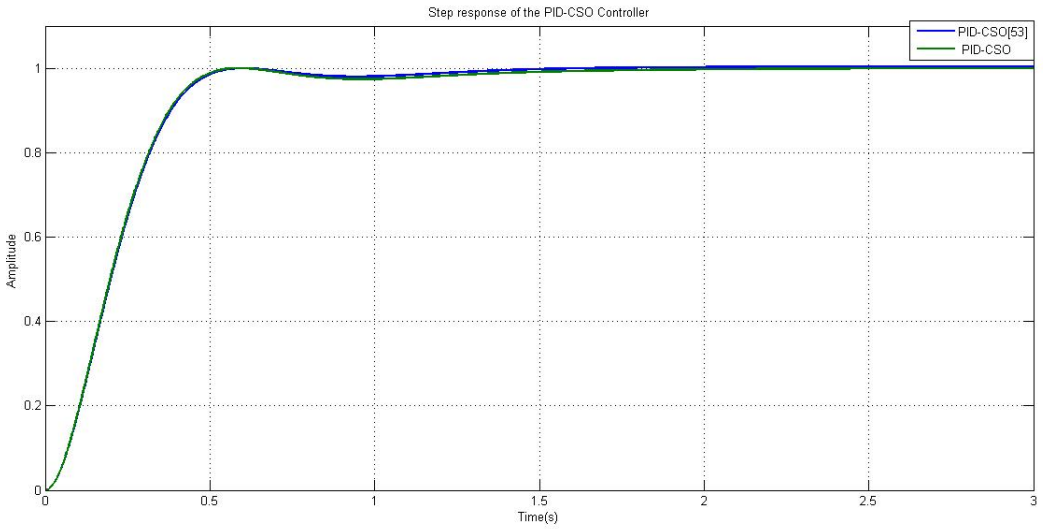


Figure 4.7: Optimal step responses by CSO-PID Comparison to recent/referenced work

The results for the terminal closed loop step response for both works are approximate, as shown in Table 4.10 and Figure 4.7, but our proposed system yielded no oscillation, which is extremely desirable.

Our performance results are definitely encouraging. This indicates that we have achieved our goal of optimizing the AVR based CSO-PID tuning optimization technique.

4.4.3 Simulation Results using MFO PID controller

The comparative simulation results obtained for the terminal voltage step response of the AVR system at different functions are shown in Figure 4.8

The comparative transient response analysis with performance characteristics of T_s, T_r and $M_p\%$ is s given in Table 4.11

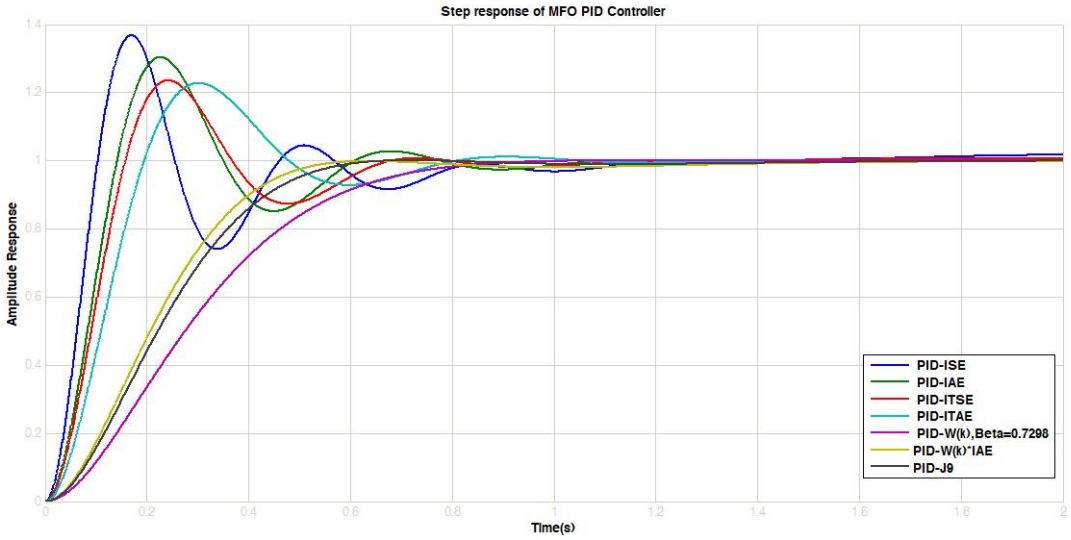


Figure 4.8: Terminal voltage step response of the AVR system with MFO-PID at different fitness functions

Table 4.11: PID gain and time response specifications for various objective functions (MFO based tuning)

PID-MFO/Obj-fct	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-ISE	1.1116	1.9000	1.4066	**0.0697**	3.2288	36.6732
PID-IAE	1.8954	1.3348	0.8345	0.0954	0.9638	30.3157
PID-ITSE	1.4377	1.2230	0.7362	0.1066	0.9549	23.5203
PID-ITAE	1.5643	1.0713	0.5132	0.1319	0.7518	22.7511
PID-W(k) , β=0.7298	0.4406	0.3139	0.1289	0.4853	0.7875	0.0294
PID-W(k)*IAE	0.5920	0.4069	0.1974	0.3289	**0.5141**	**0**
PID-J9	0.5582	0.3899	0.1772	0.3589	0.5610	0.0770

4.4.3.1 Obeservation and analysis results

Based on a comparison of performance analysis and performance of various objective functions, the results for the terminal closed loop step response of the first set of objectives (ISE, IAE, ITSE, ITAE) are so comparable producing high overshoot varying in the range 22,75% to 36, 67% with short rise time ranging from 0,06s to 0,13s and a bit of a long settling time varying from 0,35s to 3,22s. For the second set of fitness functions (J5, $W(k) * IAE$, $J9$) it gave us good results compared to the first set

when concerning the settling time and the percentage overshoot , the settling time is shortened with these fitness functions, ranged from 0.51s to 0.78s, furthermore the produced overshoot with J6 and $J9$ is tended to be zero a and no oscillation (MP= 0,00 %) with $w(k) * IAE$.

From Table 4.11, it can be clearly seen that the performance of the PSO based PID with objectives functions $w(k)$, $w(k) * IAE$ and J9 has yielded optimum response with no oscillation and shorter settling time.

4.4.3.2 Comparison with recent and reference work

Table 4.12 illustrates the best results produced by our program, as well as some Performance results from recent papers on the same subject.

Table 4.12: Comparison of CSO performance to recent/referenced work

$PID-MFO^{W(K)*IAE}$	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-MFO	0.5920	0.4069	0.1974	0.3289	**0.5141**	**0**
PID-MFO[58]	3.000	1.2200	1.2100	0.155	0.8360	16.700

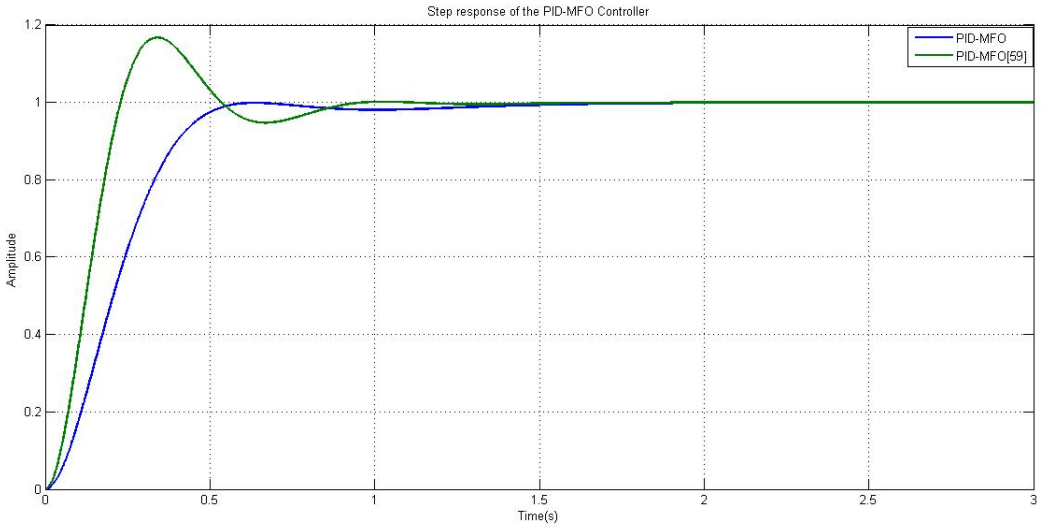


Figure 4.9: Optimal step responses by MFO-PID Comparison to recent/referenced work

The proposed $MFO^{W(K)*IAE} - PID$ controller has good performance when compared to other recent studies, as shown in Table 4.12 and Figure 4.9 Our AVR system has no oscillation structure than the other performances.

Our performance results are definitely encouraging, as we can see. This means that we have succeeded in improving the AVR-based MFO-PID tuning optimization technique.

4.4.4 Simulation Results using WCO-PID controller

The comparative simulation results obtained for the terminal voltage step response of the AVR system at different functions are shown in Figure 4.10

The comparative transient response analysis with performance characteristics of T_s, T_r and $M_p\%$ is s given in Table 4.13

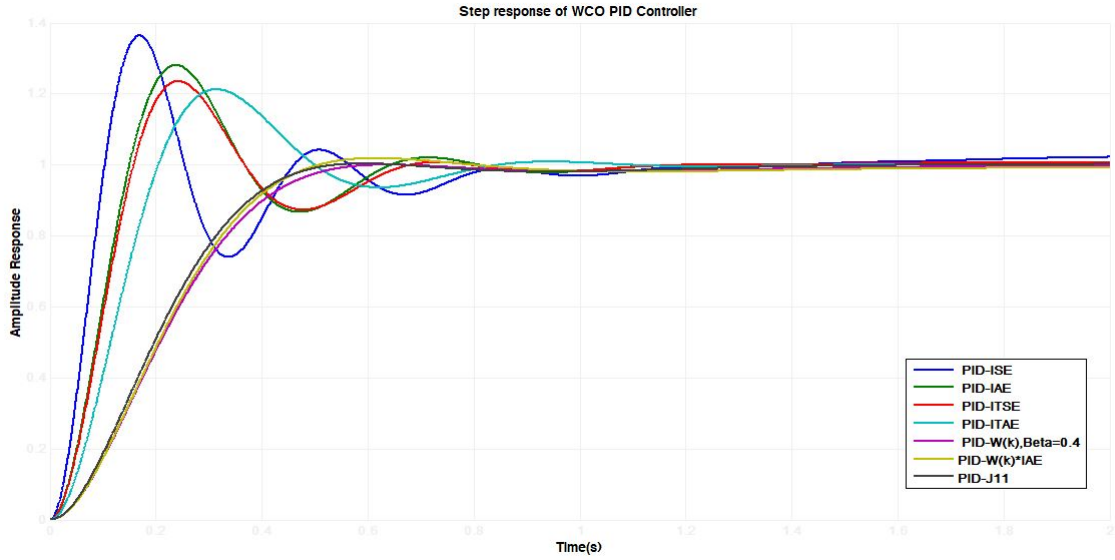


Figure 4.10: Terminal voltage step response of the AVR system with WCO-PID at different fitness functions

Table 4.13: PID gain and time response specifications for various objective functions (WCO based tuning)

PID-WCO/Obj-fct	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-ISE	1.0139	2	1.4120	0.0696	3.1948	36.7628
PID-IAE	1.7890	1.2432	0.7655	0.1015	0.9828	28.0906
PID-ITSE	1.4378	1.2218	0.7364	0.1066	0.9569	23.5245
PID-ITAE	1.4802	1.0153	0.4809	0.1386	0.7769	21.2312
PID-W(k), β=0.4	0.5946	0.3996	0.1955	0.3295	1.0206	4.6438e-05
PID-W(k)*ISE	0.6365	0.4020	0.1963	0.3285	0.5109	**0**
PID-J10	0.6180	0.4440	0.2130	0.3179	**0.4919**	0.1177

4.4.4.1 Obeservation and analysis results

Based on a comparison of performance analysis and performance of various objective functions, the results for the terminal closed loop step response of the first set of objectives (ISE, IAE, ITSE, ITAE

) are so comparable producing high overshoot varying in the range 21,23 % to 36, 67 % with short rise time ranging from 0,06s to 0,13s and a bit of a long settling time varying from 0,77s to 3,19s.

For the second set of fitness functions (J5, W(k)*ISE, J10) it gave us good results compared to the first set when concerning the settling time and the percentage overshoot , the settling time is shortened with these fitness functions, ranged from 0.49s to 1.02s, furthermore the produced overshoot with J5 and j10 is tended to be zero a and no oscillation (M_P= 0,00 %) with $w(k) * ISE$.

From Table 4.13, it can be clearly seen that the performance of the WCO based PID with objectives functions $w(k)$, $w(k) * IAE$ and J10 has yielded optimum response with no oscillation and shorter settling time.

4.4.4.2 Comparison with recent and reference work

Table 4.14 illustrates the best results produced by our program, as well as some Performance results from recent papers on the same subject.

The optimal step response of the AVR system controlled by $WCO^{W(k)*ISE} - PID$ controller compared with recent publications is shown in Figure 4.11

Table 4.14: Optimal step responses by WCO-PID Comparison to recent/referenced work

$PID-WCO$	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-WCO	0.6365	0.4020	0.1963	0.3285	0.5109	**0**
PID-WCO[45]	0.6158	0.4410	0.2158	0.3	0.462	0.476

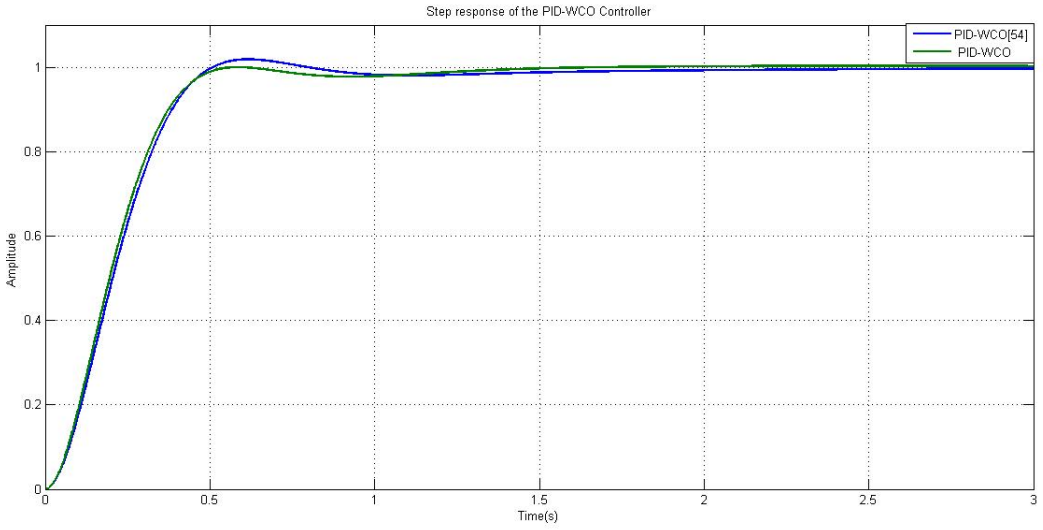


Figure 4.11: Optimal step responses by WCO-PID Comparison to recent/referenced work

As demonstrated in Table 4.14 and Figure 4.11 the results for the terminal closed loop step response of both works are remarkably similar.

Our performance results are definitely encouraging. This indicates that we have achieved our goal of optimizing the AVR based WCO-PID tuning optimization technique.

4.4.5 Simulation Results using TLBO-PID controller

The comparative simulation results obtained for the terminal voltage step response of the AVR system at different functions are shown in Figure 4.12

The comparative transient response analysis with performance characteristics of T_s, T_r and $M_p\%$ is s given in Table 4.15

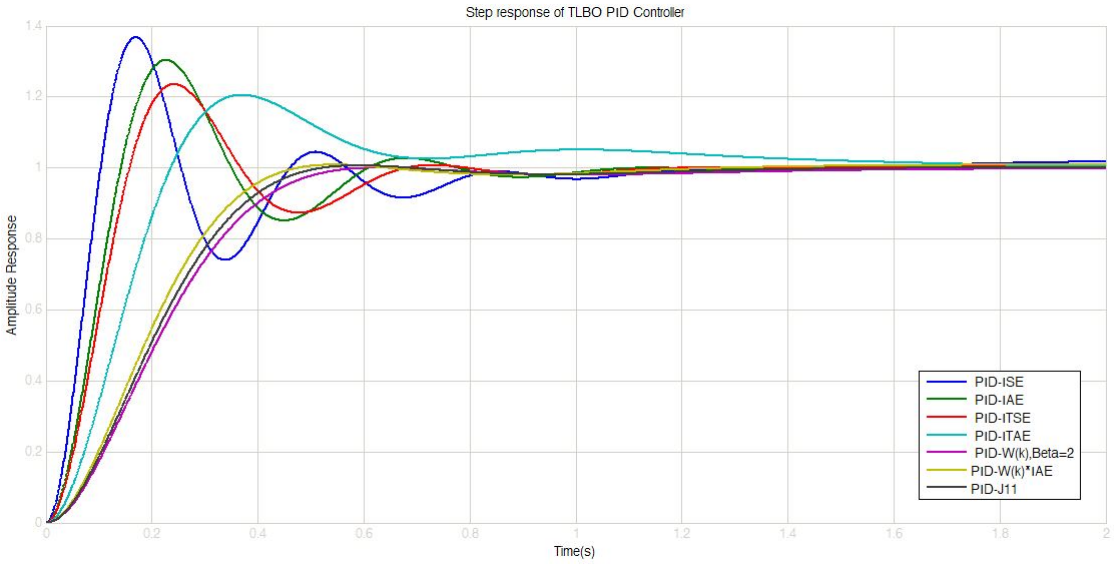


Figure 4.12: Terminal voltage step response of the AVR system with TLBO-PID at different fitness functions

Table 4.15: PID gain and time response specifications for various objective functions (TLBO based tuning)

PID-TLBO/Obj-fct	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-ISE	1.1115	1.8999	1.4065	**0.0697**	3.2288	36.6732
PID-IAE	1.8976	1.3362	0.8346	0.0954	0.9636	30.3437
PID-ITSE	1.4378	1.2218	0.7364	0.1066	0.9569	23.5245
PID-ITAE	1.2298	1.8472	0.3944	0.1623	0.8533	16.3603
PID-W(k), β=2	0.5967	0.4080	0.1982	0.1982	0.5073	**0.0027**
PID-W(k)*IAE	0.6439	0.5044	0.2343	0.2838	**0.4310**	0.8440
PID-J11	0.6236	0.4451	0.2124	0.3058	0.4670	0.6385

4.4.5.1 Obeservation and analysis results

Based on a comparison of performance analysis and performance of various objective functions, the results for the terminal closed loop step response of the first set of objectives (ISE, IAE, ITSE, ITAE) are almost the same producing high overshoot varying in the range 16,36% to 36, 67% with short rise time ranging from 0,06s to 0,16s and a bit of a long settling time varying from 0,85s to 3,22s.

The second set of fitness functions ($J5$, $W(k) * IAE$, J11) produced better results in terms of settling time and percentage overshoot than the first set. The settling time is shorter with these fitness functions, ranging from 0.43s to 0.50s, and the produced overshoot is tended to be zero because there is no oscillation.

The performance of the TLBO-based PID with objectives functions $w(k)$, $w(k) * IAE$, and $J11$ can be shown in Table 4.13, which reveals that the PSO-based PID with objectives functions $w(k)$, $w(k) * IAE$, and $J11$ has given an optimal response with no oscillation and a reduced settling time.

4.4.5.2 Comparison with recent and reference work

Table 4.16 illustrates the best results produced by our program, as well as some Performance results from recent papers on the same subject.

The optimal step response of the AVR system controlled by $TLBO^{W(k)*ISE} - PID$ controller compared with recent publications is shown in Figure 4.13

Table 4.16: Optimal step responses by TLBO-PID Comparison to recent/referenced work

$PID-TLBO$	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-TLBO	0.5967	0.4080	0.1982	0.1982	0.5073	**0.0027**
PID-TLBO[46]	0.5302	0.4001	0.1787	0.3537	0.5603	0.0100

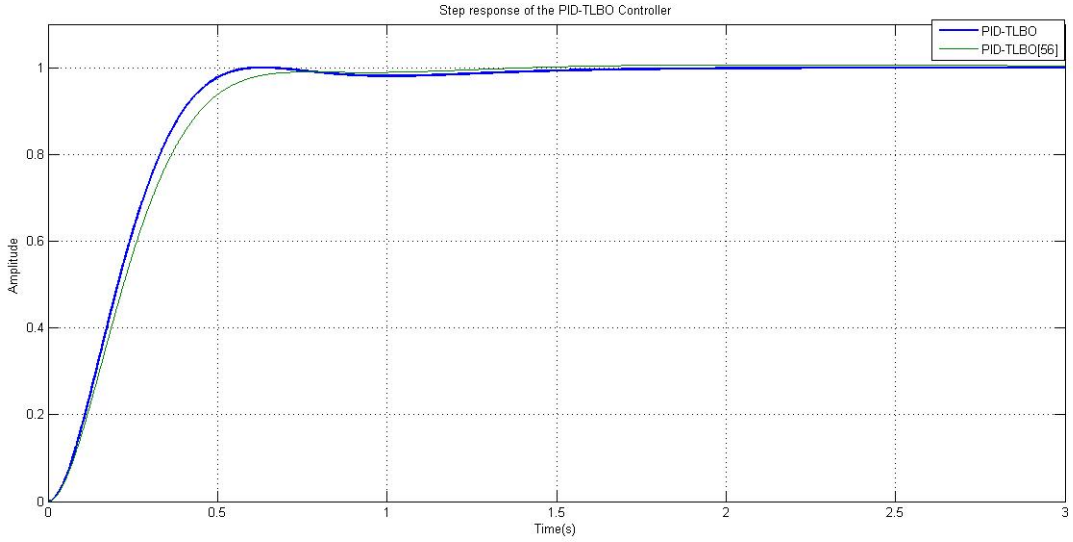


Figure 4.13: Optimal step responses by TLBO-PID Comparison to recent/referenced work

The proposed $TLBO^{W(K)} - PID$controller has good performance when compared to other recent studies, as shown in Table 4.16 and Figure 4.13. Our AVR system has faster rising time and shorter settling time then the other performances.

We can see that our performance results are indeed promising. This indicates that we have achieved our goal of optimizing the AVR based TLBO-PID tuning optimization technique.

4.4.6 Simulation Results using HCO-PID controller

The comparative simulation results obtained for the terminal voltage step response of the AVR system at different functions are shown in Figure 4.14

The comparative transient response analysis with performance characteristics of T_s, T_r and $M_p\%$ is s given in Table 4.17

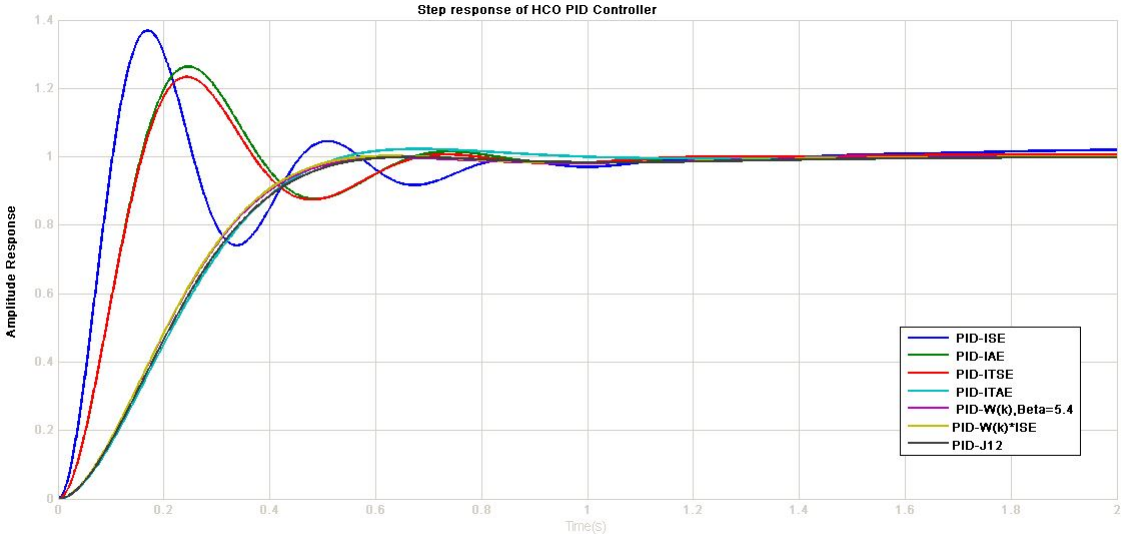


Figure 4.14: Terminal voltage step response of the AVR system with HCO-PID at different fitness functions

Table 4.17: PID gain and time response specifications for various objective functions (HCO based tuning)

PID-HCO/Obj-fct	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-ISE	1.1000	2.0000	1.4100	**0.0696**	3.1988	36.7131
PID-IAE	1.7000	1.1800	0.7200	0.1061	0.6380	26.3689
PID-ITSE	1.4300	1.2100	0.7300	0.1073	0.9434	23.2796
PID-ITAE	0.6000	0.4200	0.1800	0.3355	0.7474	2.2685
PID-W(k), β=5 .4	0.5900	0.4200	0.2000	0.3272	0.5131	**0**
PID-W(k)*ISE	0.600	0.420	0.200	0.3233	**0.5001**	0.2121
PID-J12	0.5800	0.4000	0.1900	0.3394	0.5310	**0**

4.4.6.1 Obeservation and analysis results

Based on a comparison of performance analysis and performance of various objective functions, the results for the terminal closed loop step response of the first set of objectives (ISE, IAE, ITSE, ITAE Based on a comparison of performance analysis and performance of various objective functions, the results for the terminal closed loop step response of the first set of objectives (ISE, IAE, ITSE, ITAE) are remarkably similar, with little high overshoot ranging from 2.26% to 36.71%, short rise time ranging from 0.06s to 0.33s, and a bit of a long settling time .The second set of fitness functions (J5, W(k)*ISE, J12) produced better results in terms of settling time and percentage overshoot than the first set. With these fitness functions, the settling period is reduced, ranging from 0.32s to 0.33s, and the resulting overshoot tends to be zero where no oscillation exists.

The performance of the HCO-based PID with objectives functions $w(k)$, $w(k) * ISE$, and $J12$ can be shown in Table 4.17, which reveals that the HCO-based PID with objectives functions $w(k)$, $w(k) * IAE$, and $J11$ has given an optimal response with no oscillation and a reduced settling time .

It's worth noting that we couldn't find any research papers for the AVR system that used the similar optimization method, on the other hand, our response performances provided good results, indicating that the goal of optimizing the AVR-based HCO-PID tuning optimization approach has been met.

4.5 Stability verification analysis

It has been checked the stability our system with all the proposed algorithms using `isstable()` and `rlocus()` functions.Figure 4.15 is an illustration of the stability analysis. The zoomed map shows that there are no poles in the right half plane, indicating that the system is stable.

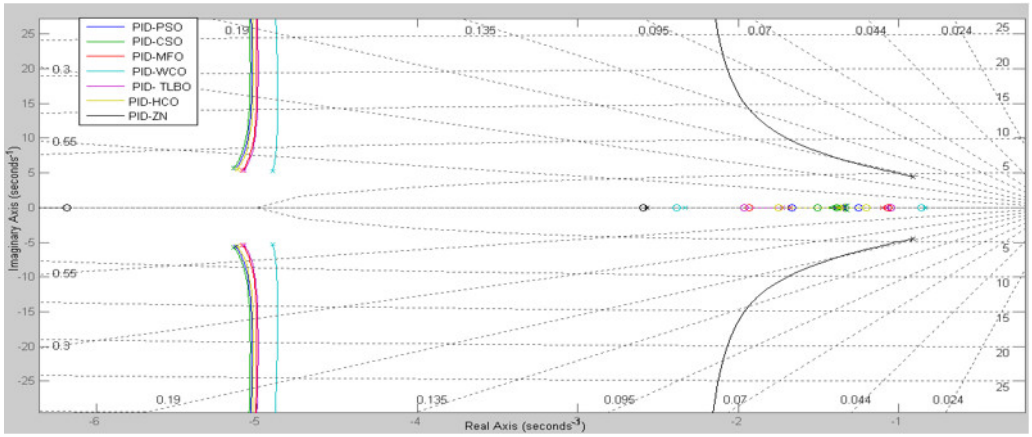


Figure 4.15: Root locus plot for the system

To investigate the stability of the proposed AVR through another point of view, frequency response or bode analysis of the control system is conducted. Bode plot analysis is a useful method for analyzing the frequency response of a control system. Frequency behavior of AVR system based on all the proposed algorithms is obtained using `bode()` function.

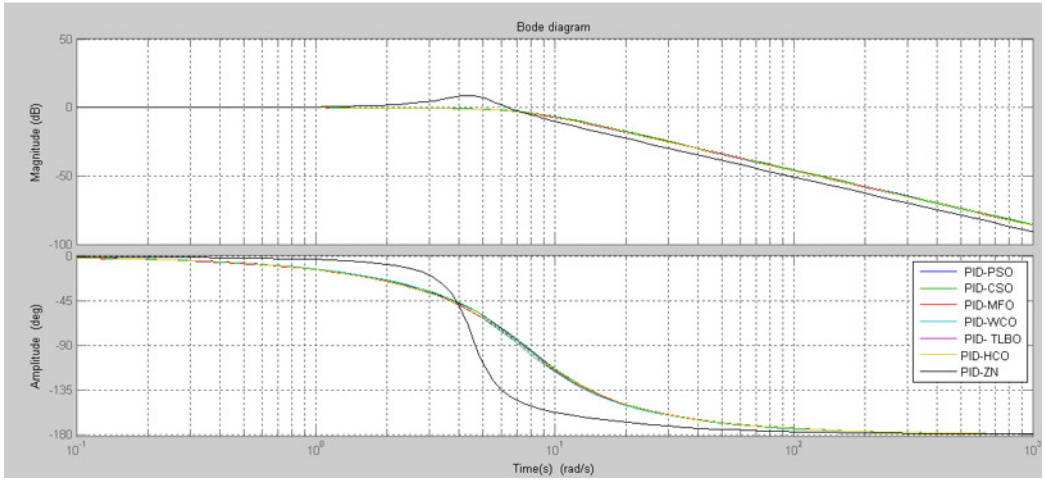


Figure 4.16: Bode diagram of the system

The closed-loop poles and their respective are also computed and gathered in Table 4.18. It is clear that the conjugate poles of the presented AVR system are farther away from the imaginary axis, which makes the system be controlled with the biggest damping ratio ranging from 0.63 to 1. In case of PID-PSO, PID-CSO, PID-MFO, PID-WCO, PID-TLBO and PID-HCO systems but PID-ZN system has two poles with 0.2 damping ratio which explains the existence of a large amount of overshoot percentage using PID-ZN controller.

Table 4.18: Poles and Damping ratio for the proposed optimization methods

	Poles	Damping Ratio
PID-PSO	-100.57	0.9822
	-5.13 + 5.66i	0.9822
	-5.13 – 5.66i	0.6718
	-1.33 + 0.25i	0.6718
	-1.33 - 0.25i	1.0000
PID-CSO	-100.66	0.8827
	-5.20 + 6.39i	0.8827
	-5.20 - 6.39i	0.6314
	-1.22 + 0.65i	0.6314
	-1.22 - 0.65i	1.0000
PID-MFO	-100.54	1.0000
	-5.09 + 5.38i	1.0000
	-5.09 - 5.38i	0.6873
	-1.68	0.687
	-1.10	1.000
PID-WCO	-100.54	1.0000
	-4.90 + 5.26i	1.0000
	-4.90 - 5.26i	0.6818
	-2.33	0.681
	-0.83	1.000
PID-TLBO	-100.55	1.0000
	-5.08 + 5.39i	1.0000
	-5.08 - 5.39i	0.6856
	-1.73	0.685
	-1.07	1.000
PID-HCO	-100.55	0.9899
	-5.12 + 5.46i	0.9899
	-5.12 - 5.46i	0.6844
	-1.35 + 0.19i	0.6844
	-1.35 - 0.19i	1.0000
PID-ZN	-100.30	1.0000
	-8.80	0.2016
	-0.91 + 4.44i	0.2016
	-0.91 - 4.44i	1.0000
	-2.57	1.0000

4.6 Comparison between the Proposed Optimization Methods

The proposed optimization techniques will be compared in this section. There are numerous approaches to compare algorithms, including comparing their step responses, computing time, and convergence characteristics.

4.6.1 Based on Step Responses

4.6.1.1 Using the same objective function

The comparative transient response analysis with performance characteristics of $M_p\%$, T_r and T_s is given in Table 4.19. In the table, the values obtained are with the same objective function ITAE.

Table 4.19: PID gain and time response specifications for various objective functions (PSO based tuning)

Optimization Technqiue	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PSO-ITAE	0.7027	0.5471	0.37852	0.1969	1.3466	1.718
CSO-ITAE	0.6999	0.54672	0.37904	0.1967	1.3498	1.7266
MFO-ITAE	1.5643	1.0713	0.5132	0.1319	0.7518	22.7511
WCO-ITAE	1.4802	1.0153	0.4809	0.1386	0.7769	21.2312
TLBO-ITAE	1.2298	1.8472	0.3944	0.1623	0.8533	16.3603
HCO-ITAE	0.5900	0.4200	0.2000	0.3272	0.5131	0

The comparative simulation results obtained for the terminal voltage step response of the AVR system with the different optimization techniques are shown in Figure 4.17

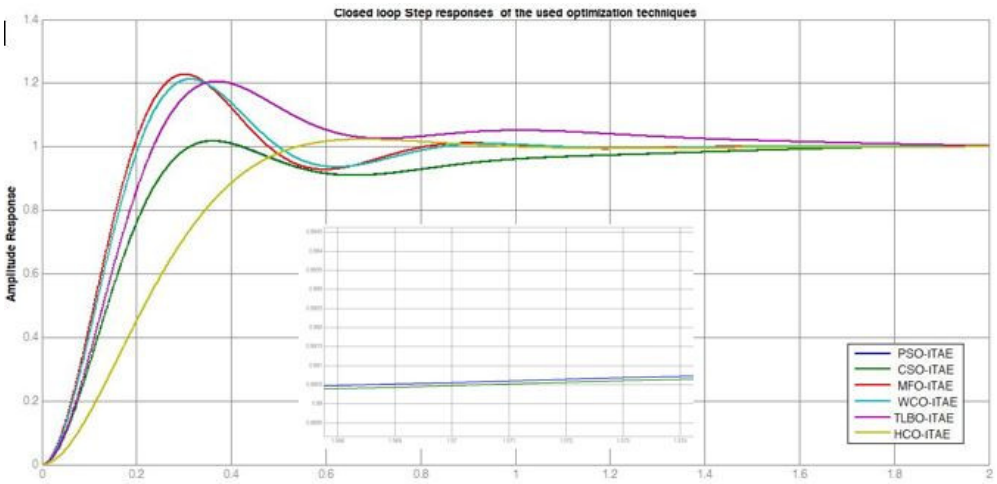


Figure 4.17: Terminal voltage response curves of the AVR system with various optimization techniques

It is evident from Figure 4.17 and Table 4.18 , the MFO, WCO and TLBO techniques delivered a response having a little high overshoot and quicker rise time than the PSO, HCO and CSO method. On the other hand, the HCO technique yielded a system with no oscillation and rapid settling time compared to the other optimization techniques. Furthermore the PSO and CSO performances are

almost the same producing response having little high overshoot and long settling time. The HCO tuned PID controller with the ITAE generates a superior control performance in terms of improved overshoot and settling time

4.6.1.2 Using different objective functions

The comparative transient response analysis with performance characteristics of $M_p\%$, T_r and T_s is given in Table 4.20. In the table, the values obtained are with the different objective functions , each transient performance corresponds to the objective function that results in the best response of the optimization approach applied. The step responses of the AVR system controlled by PID-PSO controller, PID-MFO controller, PID-WCO controller, PID-CSO controller, PID-HCO controller, PID-TLBO controller and PID-ZN are shown in Figure 4.18

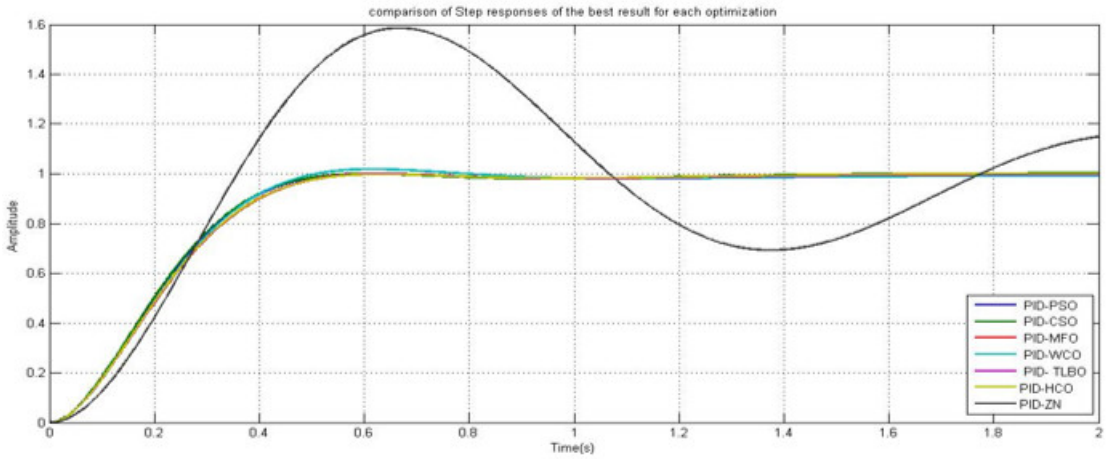


Figure 4.18: Comparison of the proposed optimization methods optimum step responses

Table 4.20: Comparison between the proposed optimization methods results

PID-OA	Controller Parameters			T_r: 0.1→0.9	$T_s \pm 2\%$	$M_P(\%)$
	K_P	K_I	K_D			
PID-PSO	0.6057	0.43203	0.208	0.3154	0.489	**0**
PID-CSO	0.6500	0.51941	0.2394	0.2787	**0.4218**	0.9847
PID-MFO	0.5920	0.4069	0.1974	0.3289	0.5141	**0**
PID-WCO	0.6365	0.4020	0.1963	0.3285	0.5109	**0**
PID- TLBO	0.5967	0.4080	0.1982	**0.1982**	0.5073	0.0027
PID-HCO	0.5900	0.4200	0.2000	0.3272	0.5131	**0**
PID-ZN	1.02	1.861	0.1163	0.244	4.33	58.500

Based on the comparison of performance analysis of the optimization techniques used, it's clearly that the CSO and TLBO's optimum step responses are faster than the other responses In addition , the utilized global optimization approaches clearly resulted in a good step response with no oscillation ($M_P = 0,00$),which is extremely desirable. The rising time of the ZN's optimum step responses is excellent, but the overshoot and settling time are poor and undesired. The PID coefficients are close together, implying that the global solution is somewhere around.

4.6.2 Based on the mean run times

In Table 4.21 the average run time of the PSO, CSO, MFO, TLBO, WCO and HCO algorithms for $J2$, $J5$ and $J10$ cost functions for 50 iterations. Average execution time of each algorithm and the average time to iteration (T/I) for each algorithm are also presented .

Table 4.21: Comparison of mean run times

Cost functions	PSO	CSO	MFO	TLBO	WCO	HCO
$J55$	154.305	146.959	224.003	352.974	158.128	**10.770**
$J2$	194.172	150.533	193.436	510.044	152.336	**12.531**
$J10$	218.974	159.772	206.399	306.174	140.334	**12.096**
Average	189.150	152.4	207.946	389.731	150.267	**11.7**
per iteration	3.783	4.572	4.159	**7.795**	3.005	**0.737**

As it can be seen, HCO algorithm with run time of 0.737 s for each iteration has the lowest program executive time in comparison with the other algorithms. The average executive time for TLBO is more than HCO by 90%, MFO by 46%, CSO by 42% and the double of PSO and WCO.

4.6.3 Based on the convergence curve characteristics

Fortunately, we have recorded some of the convergence graphs, which are PSO, MFO, WCO, TLBO and HCO in minimizing the cost function $J5$.

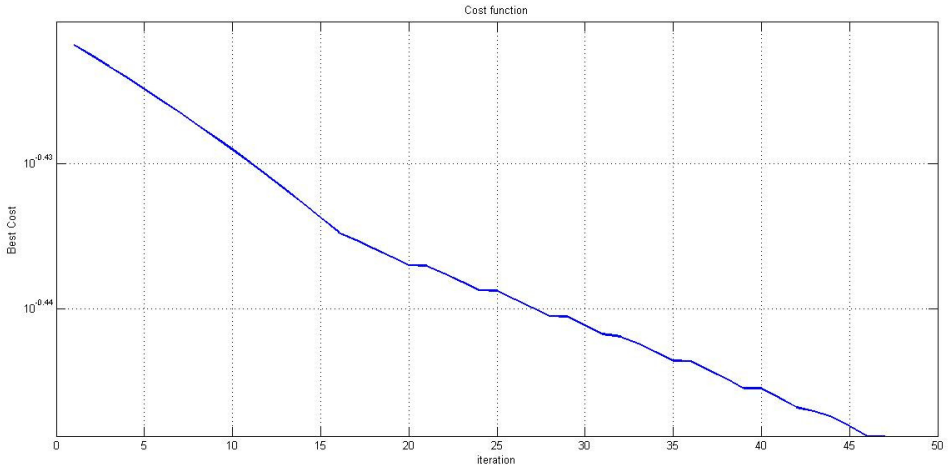


Figure 4.19: Convergence characteristic of HCO Algorithm

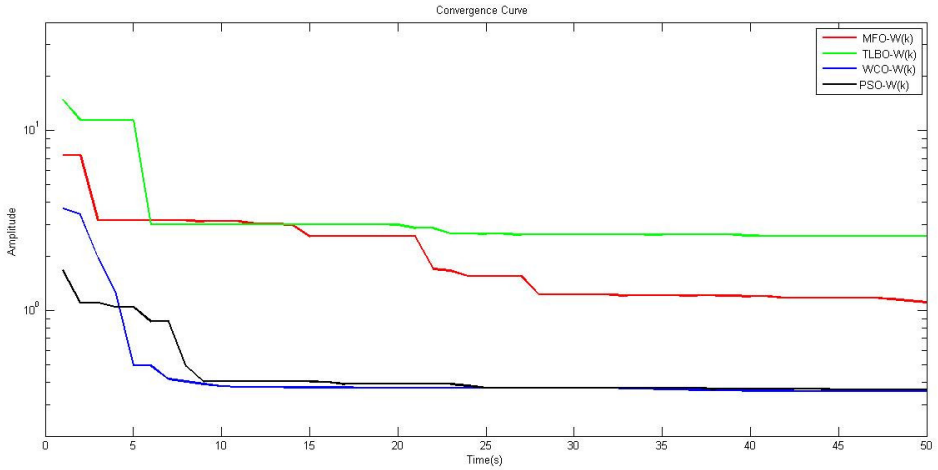


Figure 4.20: Comparison of convergence characteristic of PSO, MFO, WCO and TLBO

As it is shown in Figure 4.20, the convergence curves of all mentioned algorithms demonstrate that the PSO algorithm converges in a minimum number of iterations (approximately 25 iterations) but it has trapped in local optimum points the same as TLBO and WCO algorithms . However, MFO and HCO are the slowest algorithm which converge in maximum number of iterations (more than 45 iterations) but they continued to explore the solution space and are not trapped at local minimum.

4.7 Robustness Analysis

The previous sections of the analysis in this chapter were carried out under nominal system conditions, however, it is possible that the AVR system's components will change their parameters suddenly. Robustness analysis examines the capability of a system to endure the uncertainties in its parameters. In the current section, the uncertainties in AVR amplifier, exciter, generator and sensor gains and time constants are examined.

The following experiments, which incorporate parameter uncertainties in AVR systems due to changes

in load conditions are implemented to demonstrate the robustness of our optimum PID controller, namely HCO-PID controller.

4.7.1 Amplifier uncertainty

Figure 4.21 presents the terminal voltage step response for AVR system with HCO-PID controller when KA varied from 10 to 40. Clearly, as the value of parameter K_A increases, the overshoot M_p and the settling time Ts increases but rising time T_r and becomes faster and the steady state error Ess is smaller. Whereas in Figure 4.22 , as the value of parameter T_A increases, T_r and T_s becoming longer. All the results that we have found is presented in the Table 4.22

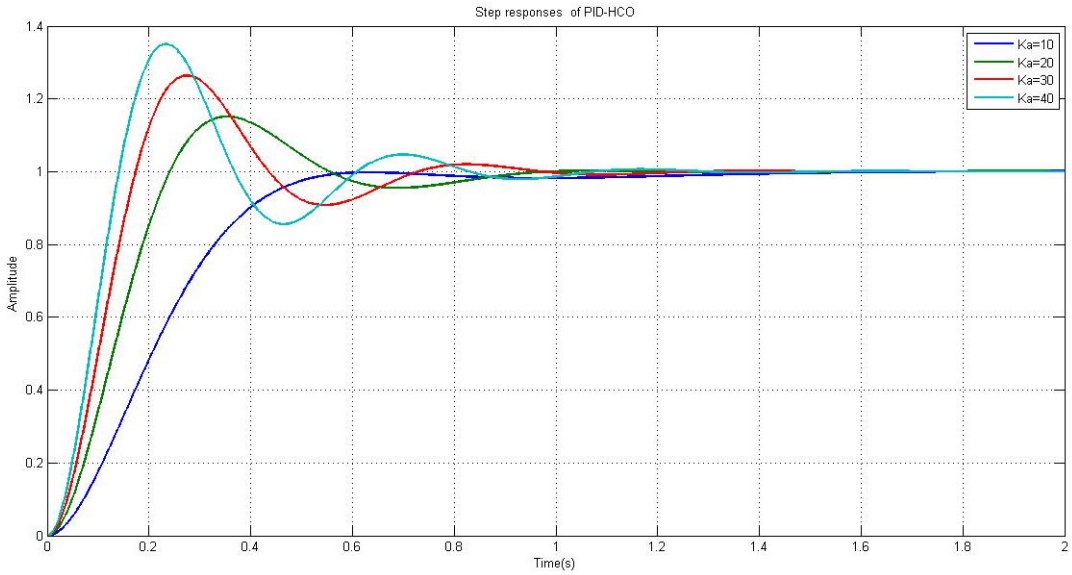


Figure 4.21: Step response under the variation of K_A

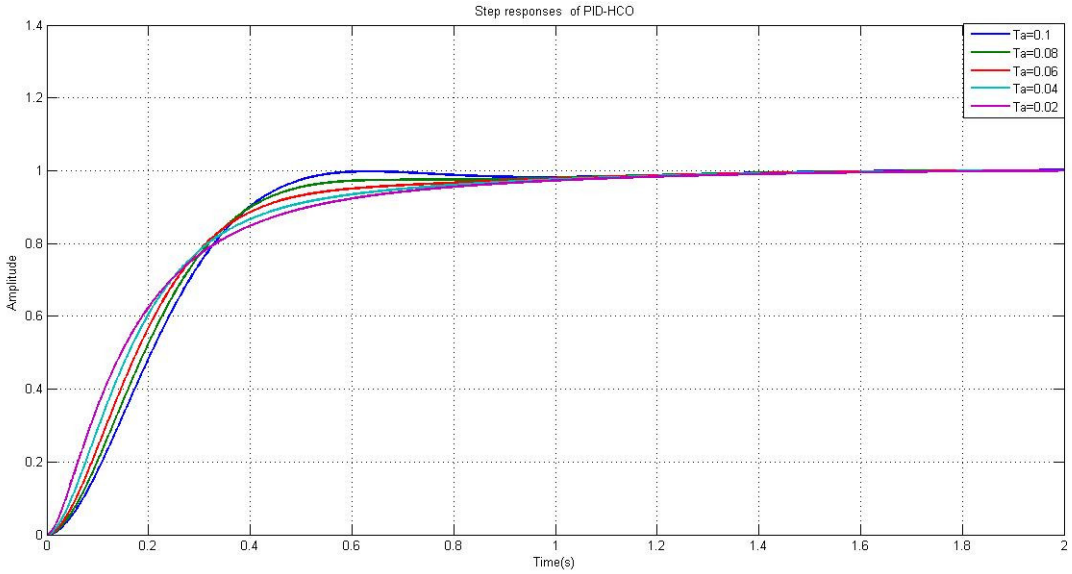


Figure 4.22: Step response under the variation of T_A

Table 4.22: Step response performances under the variation of K_A and T_A

	K_A =10	K_A =20	K_A =30	K_A =40	T_A=0.08	T_A=0.06	T_A=0.04	T_A=0.02
$T_r(10^2 s)$	18.1447	8.5	11.4228	8.7707	23.1616	32.3392	63.7482	201.2756
$T_s(10^2 s)$	29.3615	45.6	87.8752	91.2089	73.1193	90.0156	155.6080	462.0197
$P_O(\%)$	0	14.7814	27.1323	34.4863	0	0	0	0

4.7.2 Exciter uncertainty

Figure 4.23 shows the terminal voltage step response for an AVR system with a HCO-PID controller when K_E varies from 1.0 to 1.2, 1.5, 1.8, 2.0 and T_E varies from 0.4 to 0.6, 0.8, 1.0. Clearly, When K_E varied from 1.0 to 2.0 the overshoot M_P and the settling time T_s becoming, larger but the rising time T_r getting more slower, whereas when TE varied from 0.4 to 1.0 the three mentioned performances increasing .

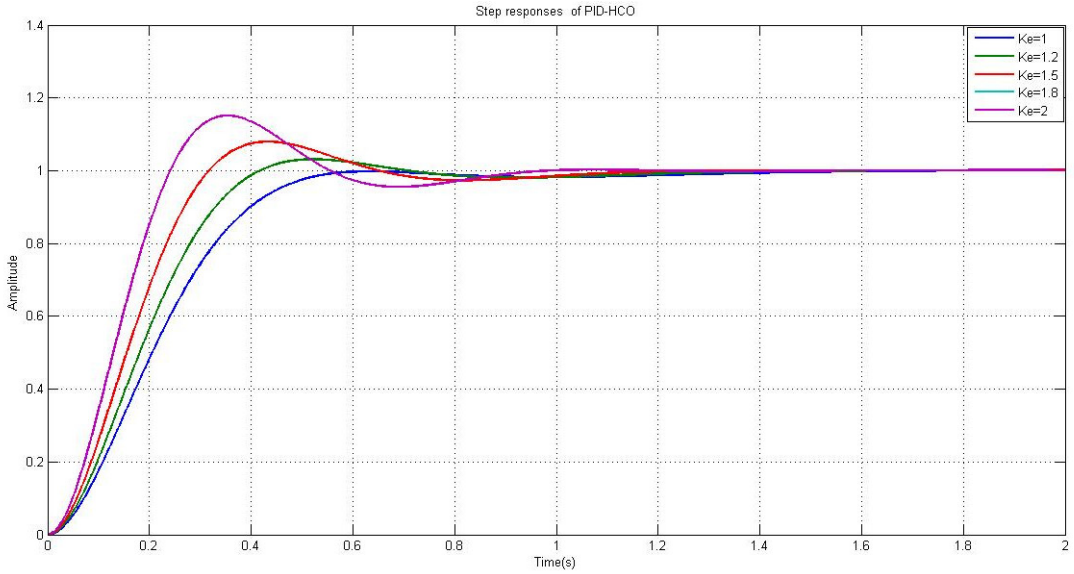


Figure 4.23: Step response under the variation of K_E

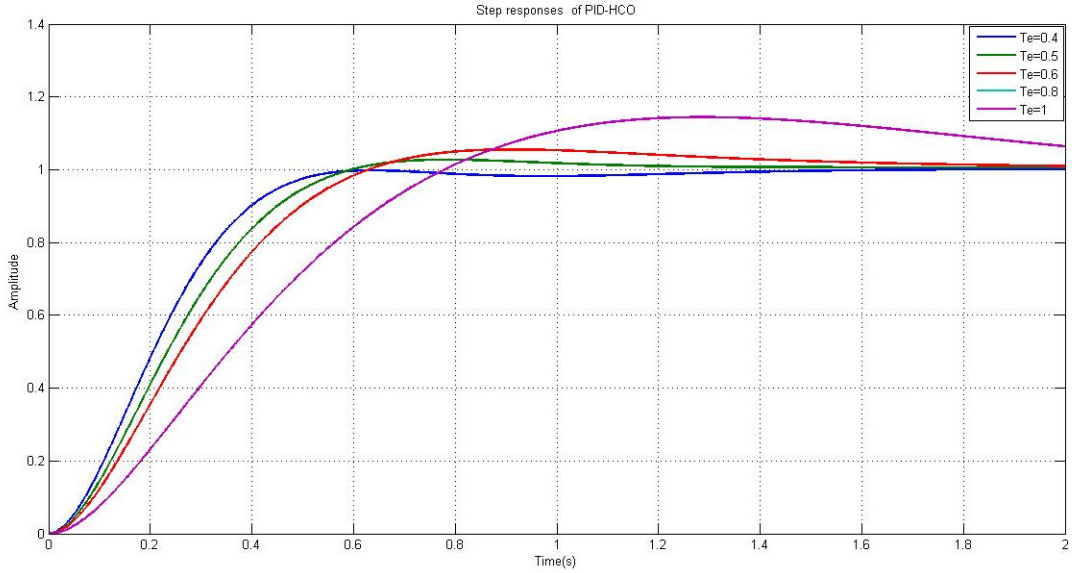


Figure 4.24: Step response under the variation of T_E

Table 4.23: Step response performances under the variation of K_A and T_A

	$K_E = 1$	$K_E = 1.2$	$K_E = 1.5$	$K_E = 1.8$	$K_E = 2$	$T_E = 0.5$	$T_E = 0.6$	$T_E = 0.8$	$T_E = 1$
$T_r(10^2 s)$	18.1447	14.5070	11.3395	9.4160	8.5174	19.4909	40.8471	28.1441	32.9083
$T_s(10^2 s)$	29.3615	34.8852	50.8293	47.3522	45.5578	50.5761	150.0291	120.1968	147.2828
$P_O(\%)$	0	3.2424	8.0411	12.4495	14.7814	2.6295	5.1892	7.9295	10.3812

4.7.3 Generator uncertainty

Figure 4.25 shows the terminal voltage step response for an AVR system with a HCO-PID controller when K_G was changed from 1.0 to 0.9, 0.8, and 0.7, and T_G was changed from 1.0 to 1.3, 1.7, and 1.8. 1.9 and 2.0 points, respectively. The HCO-PID controller is clearly robust against the Variation of the parameters K_G and T_G. The variation of the step response performances is shown in the Table 4.24

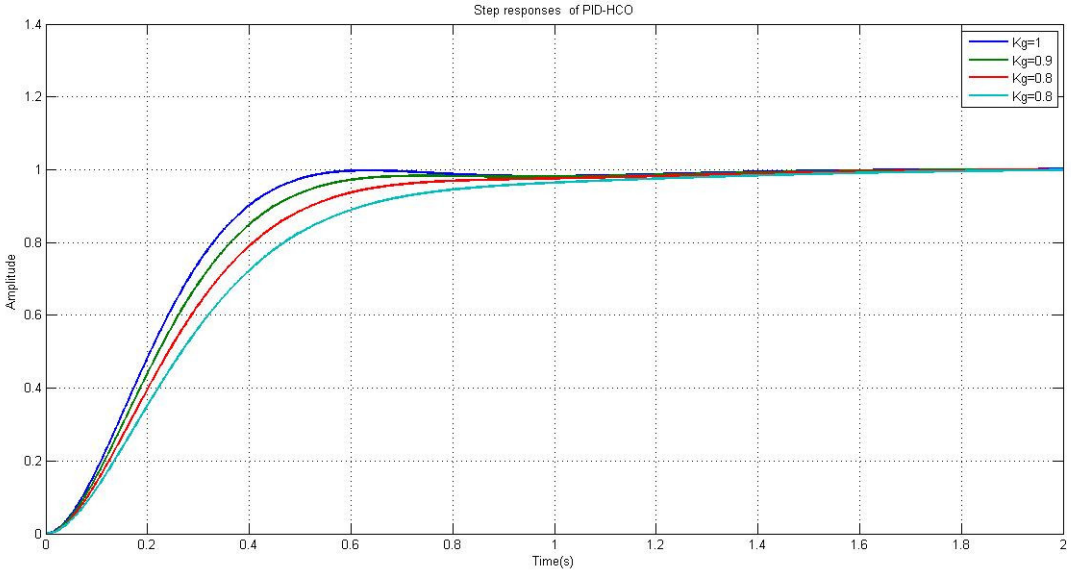


Figure 4.25: Step response under the variation of K_G

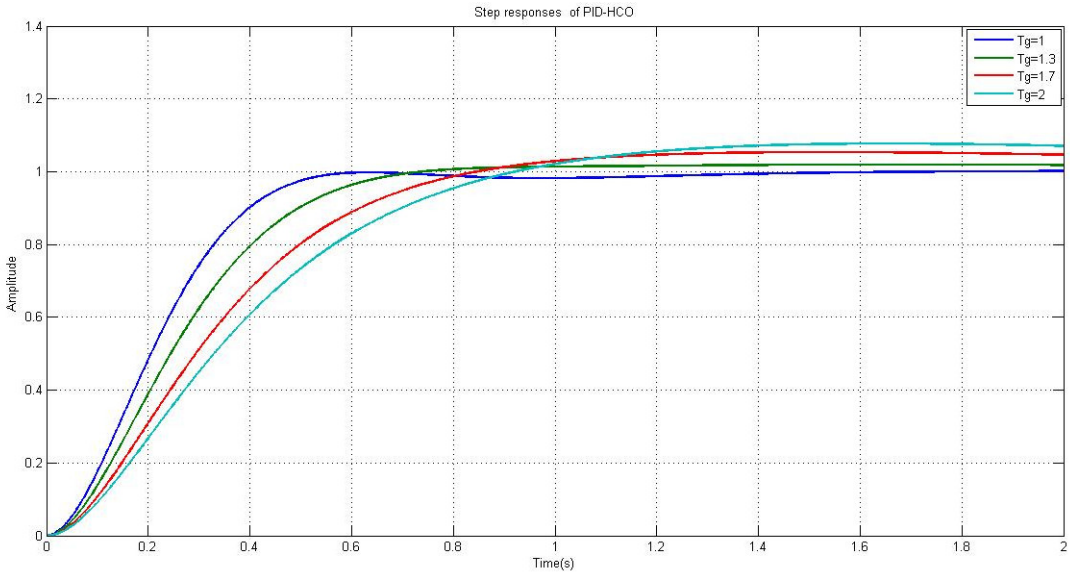


Figure 4.26: Step response under the variation of T_G

Table 4.24: Step response performances under the variation of K_A and T_A

	$K_G = 1$	$K_G = 0.9$	$KG =_0 .8$	$K_G = 0.7$	$T_G = 1.3$	$T_G = 1.7$	$T_G = 2$
$T_r(10^2 s)$	18.1447	21.1168	24.7583	30.5207	13.2731	9.9778	20.4604
$T_s(10^2 s)$	29.3615	62.4535	62.8200	74.0968	51.8944	48.7872	77.5990
$P_O(\%)$	0	5.6235e-04	0	0	4.8064	10.9762	15.2141

4.7.4 Sensor uncertainty

Figure 4.27 shows the terminal voltage step response for an AVR system with an HCO-PID controller when T_s was changed.

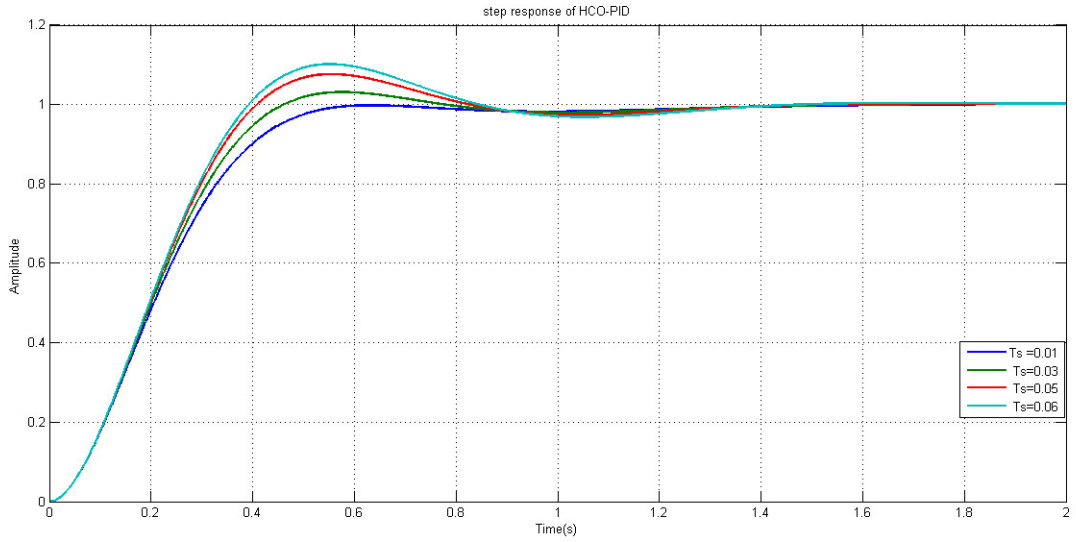


Figure 4.27: Step response under the variation of T_s

Table 4.25: Step response performances under the variation of T_s

	$T_s = 0.01$	$T_s = 0.03$	$T_s = 0.05$	$T_s = 0.06$
$T_r(10^2 s)$	18.1447	14.2221	11.1004	9.9245
$T_s(10^2 s)$	29.3615	51.8013	50.0635	47.6513
$P_O(\%)$	0	3.0844	7.3516	9.7854

The average variance of the peak overshoot, settling time, and rising time, when all time constants and gains are taken into account are about 10%, 30%, and 22%, respectively. These findings demonstrate that the HCO-PID controller is robust, preserving the desired transient response regardless of variations in any of the gains or time constants over the change interval in question. It is worth mention that we have tested the other PID controllers and we have gotten almost the same robustness results.

4.8 Rejection of the Disturbances

By introducing three types of disturbances into the AVR system: control signal disturbance, load disturbance, and measurement noise, the ability of the PID controllers to cope with different disturbances is examined. the AVR block diagram with the considered disturbances is shown in Figure 4.28

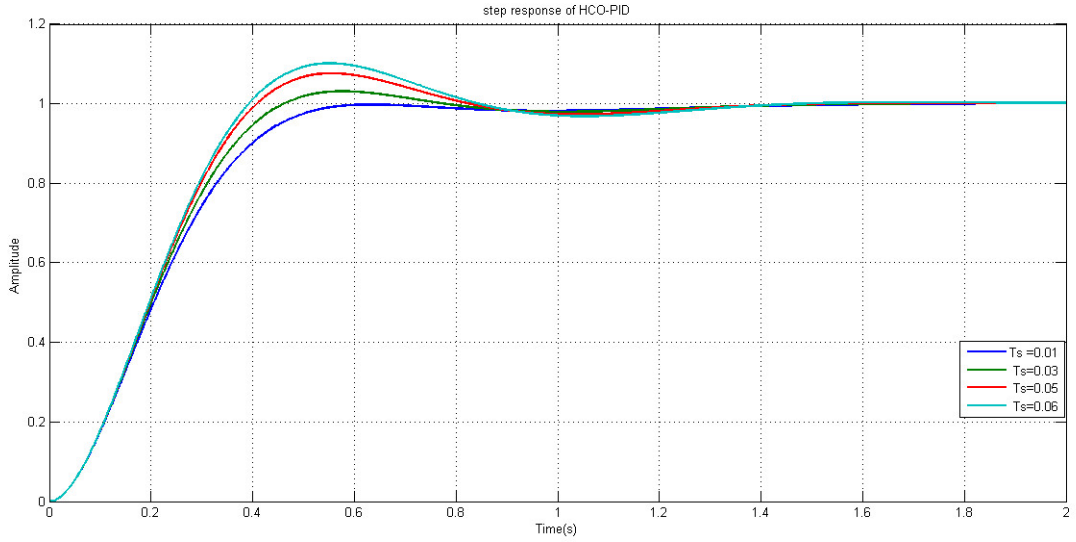


Figure 4.28: Block diagram of the AVR system considering different kinds of disturbances

4.8.1 Control disturbances

4.8.2 Load Disturbances

4.8.3 Measurement (Sensor) Disturbances

4.9 Conclusion

In this chapter, The proposed metahuristic optimization techniques and the classical Z-N method based tuning PID for AVR systems were analysed in depth. It was demonstrated that the six proposed metaheuristique optimization approaches produced interestingly promising results in comparison to the current state of research in this area. It has been demonstrated that the cost function chosen is significantly relevant from one algorithm to the next. Furthermore, a comparison of the various optimization approaches has been made in terms of step response, computing time, and convergence characteristics. Our system's stability and robustness have also been tested against amplifier, exciter, generator, and sensor uncertainties. Our optimized AVR system was shown to be purely stable and robust to gain variations, time delay and different types of disturbances .

General Conclusion

In this report, For the purpose of tuning the PID controller of an automatic voltage regulator, six distinct nature-inspired optimization approaches have been implemented, namely PSO, CSO, MFO, TLBO,HCO, and WCO. It has proven to be a success because the provided methods resulted in performances that were considered to be significantly encouraging compared to recent published literature. This demonstrates that the project's main objective of enhancing AVR performance through the use of the six recommended methods has been accomplished. The presented techniques solve an optimization problem to determine the PID controller parameters by minimizing the objective function to achieve the desired system responses. Then, a comparative analysis of PID controller design methods for AVR is performed .For AVR system model using PID controller ,PSO,CSO,MFO,WCO,TLB and HCO showed better response then the AVR system system with Z-N classical method.

By comparing step responses, it was revealed that the CSO and TLBO algorithms perform better than the PSO, MFO, WCO, and HCO algorithms. However, the HCO algorithm has a faster computing time. On the other hand, in terms of convergence curve characteristics, the PSO outperforms the other methods. Moreover, root locus and bode plot were performed to evaluate the stability of the optimum AVR system , it was found that the proposed system has good stability structure . The superiority of the work presented is studying the proposed HCO-PID controller robustness to system parameter changes and disturbance rejection which are not covered in many literatures.

In this context, there are a variety of approaches to implement an AVR system.For example a hybrid Algorithm by Combination of two from the proposed optimization algorithms. Finally, fractional order PID can be used to achieve better results. As a result, a more in-depth analysis of the latter to determine its practical applicability would revolutionize the automation field as a whole. .

References

[1] M. Micev, M. Ćalasan, and D. Oliva, "Fractional order pid controller design for an avr system using chaotic yellow saddle goatfish algorithm", *Mathematics*, vol. 8, no. 7, p. 1182, 2020.

[2] R. Lahcene, S. Abdeldjalil, and K. Aissa, "Optimal tuning of fractional order pid controller for avr system using simulated annealing optimization algorithm", in *2017 5th International Conference on Electrical Engineering-Boumerdes (ICEE-B)*, IEEE, 2017, pp. 1–6.

[3] M. O. Singh, S. Agarwal, S. Singh, and Z. Khan, "Automatic voltage control for power system stability using pid and fuzzy logic controller", *International Journal of Engineering Research & Technology*, vol. 2, pp. 193–198, 2013.

[4] M. Attia, A. Abdelaziz, and A. Mosaad, "Whale optimization algorithm to tune pid and pida controllers on avr system", *Ain Shams Engineering Journal*, vol. 10, Jul. 2019. DOI: 10.1016/j.asej.2019.07.004.

[5] P. P. G. Dhawale, P. Patil, N. Kumbhar, R. Mandlik, P. Nikam, and S. Kamble, "Automatic voltage regulator", *International Journal of Scientific Engineering and Science*, vol. 3, no. 4, May 2019. DOI: 10.5281/zenodo.2840181. [Online]. Available: https://doi.org/10.5281/zenodo.2840181.

[6] T. A. Jumani, M. W. Mustafa, Z. Hussain, M. Md. Rasid, M. S. Saeed, M. M. Memon, I. Khan, and K. S. Nisar, "Jaya optimization algorithm for transient response and stability enhancement of a fractional-order pid based automatic voltage regulator system", *Alexandria Engineering Journal*, vol. 59, no. 4, pp. 2429–2440, 2020, ISSN: 1110-0168. DOI: https://doi.org/10.1016/j.aej.2020.03.005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1110016820301046.

[7] Y. Meraihi, A. B. Gabis, S. Mirjalili, and A. Ramdane-Cherif, "Grasshopper optimization algorithm: Theory, variants, and applications", *IEEE Access*, vol. 9, pp. 50 001–50 024, 2021.

[8] X. Shan, K. Liu, and P.-L. Sun, "Modified bat algorithm based on lévy flight and opposition based learning", *Scientific Programming*, vol. 2016, 2016.

[9] L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A. H. Gandomi, "The arithmetic optimization algorithm", *Computer methods in applied mechanics and engineering*, vol. 376, p. 113 609, 2021.

[10] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm", *Knowledge-based systems*, vol. 89, pp. 228–249, 2015.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization", in *Proceedings of ICNN'95-international conference on neural networks*, IEEE, vol. 4, 1995, pp. 1942–1948.

[12] A. P. Piotrowski, J. J. Napiorkowski, and A. E. Piotrowska, "Population size in particle swarm optimization", *Swarm and Evolutionary Computation*, vol. 58, p. 100 718, 2020.

[13] D. Wang, D. Tan, and L. Liu, "Particle swarm optimization algorithm: An overview", *Soft Comput.*, vol. 22, no. 2, pp. 387–408, Jan. 2018, ISSN: 1432-7643. DOI: 10.1007/s00500-016-2474-6. [Online]. Available: https://doi.org/10.1007/s00500-016-2474-6.

[14] Z.-L. Gaing, "A particle swarm optimization approach for optimum design of pid controller in avr system", *IEEE transactions on energy conversion*, vol. 19, no. 2, pp. 384–391, 2004.

[15] A. P. Engelbrecht, *Computational intelligence: an introduction*. John Wiley & Sons, 2007.

[16] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights", in *2009 World congress on nature & biologically inspired computing (NaBIC)*, Ieee, 2009, pp. 210–214.

[17] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems", *Engineering with computers*, vol. 29, no. 1, pp. 17–35, 2013.

[18] M. Črepinšek, S.-H. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: A survey", *ACM computing surveys (CSUR)*, vol. 45, no. 3, pp. 1–33, 2013.

[19] V. V. G. Pentapalli and R. K. Varma, "Cuckoo search optimization and its applications: A review", *Int. J. Adv. Res. Comput. Commun. Eng*, vol. 5, no. 11, pp. 2319–5940, 2016.

[20] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search", *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[21] ——, "Cuckoo search: Recent advances and applications", *Neural Computing and Applications*, vol. 24, no. 1, pp. 169–174, 2014.

[22] E. Valian, S. Mohanna, and S. Tavakoli, "Improved cuckoo search algorithm for feed forward neural network training", *International Journal of Artificial Intelligence & Applications*, vol. 2, Jul. 2011. DOI: 10.5121/ijaia.2011.2304.

[23] E. Valian, S. Tavakoli, S. Mohanna, and A. Haghi, "Improved cuckoo search for reliability optimization problems", *Computers & Industrial Engineering*, vol. 64, no. 1, pp. 459–468, 2013.

[24] X.-S. Yang and S. Deb, "Multiobjective cuckoo search for design optimization", *Computers & Operations Research*, vol. 40, no. 6, pp. 1616–1624, 2013.

[25] A. E. Hassanien, T. Gaber, U. Mokhtar, and H. Hefny, "An improved moth flame optimization algorithm based on rough sets for tomato diseases detection", *Computers and Electronics in Agriculture*, vol. 136, pp. 86–96, 2017.

[26] nbspAjay Dixit, nbspMahesh Lokhande, and N. Joshi, "Optimization of automatic voltage regulator using moth flame optimization algorithm", *International Journal of Engineering Development and Research*, vol. 4, pp. 367–371, 2016.

[27] M. Abd El-Hameed and A. El-Fergany, "Water cycle algorithm-based load frequency controller for interconnected power systems comprising non-linearity", *IET Generation, Transmission & Distribution*, vol. 10, Jul. 2016. DOI: 10.1049/iet-gtd.2016.0699.

[28] A. A. Heidari, R. A. Abbaspour, and A. R. Jordehi, "An efficient chaotic water cycle algorithm for optimization tasks", *Neural Computing and Applications*, vol. 28, no. 1, pp. 57–85, 2017.

[29] Y.-K. Chen, S.-X. Weng, and T.-P. Liu, "Teaching–learning based optimization (tlbo) with variable neighborhood search to retail shelf-space allocation", *Mathematics*, vol. 8, no. 8, p. 1296, 2020.

[30] X. Chen, B. Xu, K. Yu, and W. Du, "Teaching-learning-based optimization with learning enthusiasm mechanism and its application in chemical engineering", *Journal of Applied Mathematics*, vol. 2018, 2018.

[31] javatpoint. (). Hill climbing algorithm in artificial intelligence, [Online]. Available: https://www.javatpoint.com/hill-climbing-algorithm-in-ai (visited on 06/10/2021).

[32] J. Crowe, G. Chen, R. Ferdous, D. Greenwood, M. Grimble, H. Huang, J. Jeng, M. A. Johnson, M. Katebi, S. Kwong, *et al.*, *PID control: new identification and design methods*. Springer, 2005.

[33] C. A. Monje, Y. Chen, B. M. Vinagre, D. Xue, and V. Feliu-Batlle, *Fractional-order systems and controls: fundamentals and applications*. Springer Science & Business Media, 2010.

[34] K. J. Åström and T. Hägglund, *PID controllers: theory, design, and tuning*. Instrument society of America Research Triangle Park, NC, 1995, vol. 2.

[35] P. de Larminat, *Automatique appliquée*. Lavoisier, 2009, ISBN: 9782746223813.

[36] R. C. Panda, *Introduction to PID controllers: theory, tuning and application to frontier areas*. BoD–Books on Demand, 2012. [Online]. Available: https://www.intechopen.com/books/introduction-to-pid-controllers-theory-tuning-and-application-to-frontier-areas/.

[37] B. Kristiansson and B. Lennartson, "Robust and optimal tuning of pi and pid controllers", *IEE Proceedings-Control Theory and Applications*, vol. 149, no. 1, pp. 17–25, 2002.

[38] A. J. R. Medina, G. T. Pulido, and J. G. Ramirez-Torres, "A comparative study of neighborhood topologies for particle swarm optimizers", in *IJCCI*, 2009, pp. 152–159.

[39] M. Johnson and M. Moradi, *PID Control: New Identification and Design Methods*, ser. Probability and its applications. Springer London, 2006, ISBN: 9781846281488.

[40] J. G. Ziegler, N. B. Nichols, *et al.*, "Optimum settings for automatic controllers", *trans. ASME*, vol. 64, no. 11, 1942.

[41] F. A. S. babu and S. B. Chiranjeevi, "Implementation of fractional order pid controller for an avr system using ga and aco optimization techniques", *IFAC-PapersOnLine*, vol. 49, no. 1, pp. 456–461, 2016, ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2016.03.096.

[42] A. Maghawry, R. Hodhod, Y. Omar, and M. Kholief, "An approach for optimizing multi-objective problems using hybrid genetic algorithms", *Soft Computing*, vol. 25, no. 1, pp. 389–405, 2021.

[43] M. Rahimian and K. Raahemifar, "Optimal pid controller design for avr system using particle swarm optimization algorithm", in *2011 24th Canadian conference on electrical and computer engineering (CCECE)*, IEEE, 2011, pp. 000 337–000 340.

[44] Z. Bingul and O. Karahan, "A novel performance criterion approach to optimum design of pid controller using cuckoo search algorithm for avr system", *Journal of the Franklin Institute*, vol. 355, no. 13, pp. 5534–5559, 2018.

[45] N. Pachauri, "Water cycle algorithm-based pid controller for avr", *COMPEL: Int J for Computation and Maths. in Electrical and Electronic Eng.*, vol. 39, no. 3, pp. 551–567, Jun. 18, 2020, ISSN: 0332-1649. DOI: doi:10.1108/COMPEL-01-2020-0057.

[46] S. Chatterjee and V. Mukherjee, "Pid controller for automatic voltage regulator using teaching–learning based optimization technique", *International Journal of Electrical Power & Energy Systems*, vol. 77, pp. 418–429, 2016.