DOI 10.24425/aee.2021.137574

# Comparative study of PID controller designs for AVR using different optimization techniques

HAYA HESHAM, M. EZZAT, RANIA A. SWIEF

*Electrical Power and Machines Department, Faculty of Engineering, Ain Shams University*
*1 Elsarayat St., Abbaseya, 11517 Cairo, Egypt*

*e-mail: {haya.hesham/moh_ezzat}@eng.asu.edu.eg, rania.swief@gmail.com*

(Received: 11.09.2020, revised: 12.02.2021)

**Abstract:** This paper presents the optimal PID tuning study to improve the dynamic performance of an automatic voltage regulation (AVR) system. The system under study consists of a synchronous generator whose reference voltage changes in a step function and tries to overcome the transient behavior of its terminal voltage smoothly. To optimally control the performance, different optimization techniques are applied to tune the controller gains to obtain the minimum steady state error (main objective) and better dynamic characteristics (rise time, settling time, max overshoot, etc.). Then the AVR system responses with a PID controller based on different optimization techniques are compared to find out which is the best technique.

**Key words:** AVR system, comparison, optimization techniques, PID controller

## 1. Introduction

The system that supplies load centers with electrical energy flowing from electricity generation centers is called an interconnected system. This energy flow should continuously meet the demand to maintain the system stability [1].

In power system studies, changes in system load always occur depending on many factors including time, weather conditions, etc. and this affects the main two parameters of the system: output voltage and frequency levels of the generators. When voltage and frequency cannot be kept at the required values, a power outage will take place in the system. This event is called black out and is one of the most catastrophic events occurred in a power system. So, any Power Generation System should be controlled in order to keep the voltage and frequency levels within the required values, regardless of the load changes. The automatic load frequency control (ALFC) is the

controller used to maintain the frequency stability and the Automatic Voltage Regulator (AVR) is the controller used to keep the electric generator voltage at the desired value by controlling the generator exciter dc voltage [2].

The AVR response is very important for voltage stability. AVRs affect the dynamic response; transient events that happen in electrical power systems [4–8]. So Proportional Integral Derivative (PID) type controllers are used because their structure is very simple. The purpose of using the PID controller is to enhance the dynamic stability/transient response and to decrease or eliminate the steady state error. If the value of proportional gain ($Kp$) increased, rise time and the steady state error decreased but oscillations increased, and the system will be unstable. Similarly, by increasing the integral gain ($Ki$) value, rise time also decreased with eliminating the steady state error but overshoot increased, and transient response got worse. And with regard to derivative gain ($Kd$), increasing its value leads to reducing oscillations, overshoot and settling time which means improving stability of the system but no effect on the steady state error [9–11]. So, a good selection of proportional, integral and derivative gains is very important for getting an acceptable transient response. Gains selection is called "tuning" in the literature [12, 13]. Dif ferent types of optimization techniques are used for tuning: Classical methods, Heuristic methods and Artificial Intelligence methods [14, 15].

A PID controller is the most used controller but not the only one. Current researches discuss different types such as a PI controller [1], PD controller, fractional order PID controller (FOPID) that uses the fractional order of $Kd$ and $Ki$ instead of integer order [16], PIDD$^2$ that is a four-term controller composed of proportional, integral, derivative, and second order derivative terms [17].

In this paper, different optimization techniques like FPA [9, 18, 19], TLBO [20, 21], HS [22], and LUS [23] are applied to tune the PID parameters to improve both the steady state error and the dynamic performance of the system, then the results will be compared. The paper is divided into 5 sections: **Introduction** at which the importance of the AVR system and description of a PID controller are shown, **Automatic voltage regulation system** at which the system under study is described, **Optimization techniques** at which different optimization techniques are explained, **Comparison between different optimization techniques** at which PID controller parameters are obtained by different algorithms and the transient response of each one is compared to the other and **Conclusion** at which the best technique is chosen.

## 2. Automatic voltage regulation system

All modern AVRs operate by comparing the actual output voltage to input reference voltage. An amplifier is used to amplify the difference then the amplified difference is used to control the regulation element in such a way as to minify the error of the voltage. This composes a negative feedback control loop [4], and to improve the static accuracy, we can increase the open loop gain (amplifier gain, $Ka^*$ exciter gain, $Ke^*$ generator gain, $Kg$) but this can decrease stability.

Using SIMULINK, the block diagram of an AVR system can be built by getting the time function of each element in the real model then converting the functions from the time domain into the S-domain using LAPLACE [5].

Fig. 1 and Fig. 2 [7] show the schematic and the Simulink block modeling of the AVR.

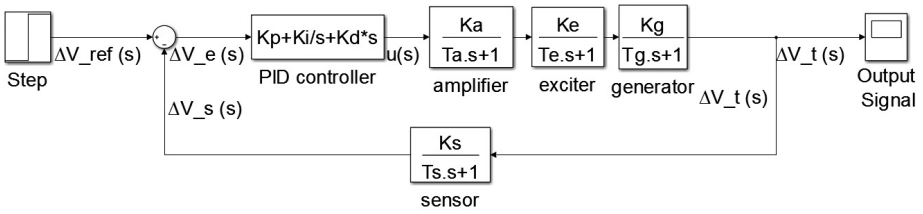


Fig. 1. The AVR system's block diagram

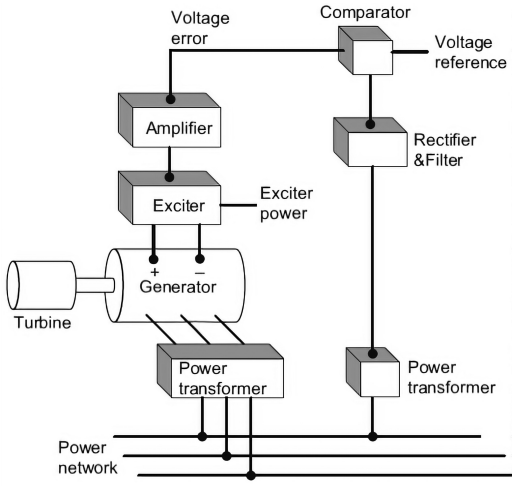


Fig. 2. Real model of AVR system

The PID controller implemented in Simulink model is not the PID controller existed in the Simulink library, it is the subsystem shown in Fig. 3, that is because the PID controller existed in the Simulink library is not ideal, its equation is

$$Kp + Ki \cdot \frac{1}{s} + Kd \cdot \frac{N}{1 + N \cdot \frac{1}{s}},$$

where $N$ is the filter coefficient.

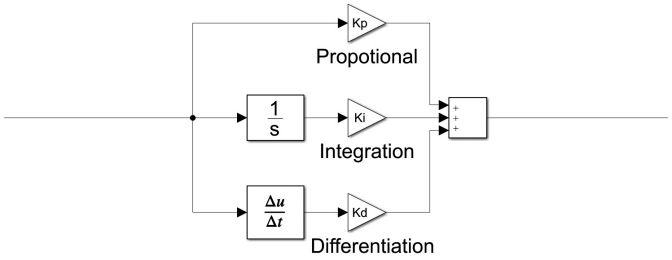


Fig. 3. The PID controller subsystem's block diagram

The governing equations of the proposed system are described as shown in Table 1 [6] and Equations (1), (2).

Table 1. The AVR system's transfer function and parameter limits

| | **Transfer function** | **Parameter limit** | **Used values** |
|---|---|---|---|
| PID controller | $K_p + \left(\frac{K_i}{s}\right) + K_d s$ | $0.2 \leq K_p K_i$, $K_d \leq 2$ | Optimum values |
| Amplifier | $K_a/(1 + sT_a)$ | $10 \leq K_a \leq 40$ $0.02 \leq T_a \leq 0.1$ | $K_a = 10, T_a = 0.1$ |
| Exciter | $K_e/(1 + sT_e)$ | $1 \leq K_e \leq 10$ $0.4 \leq T_e \leq 1$ | $K_e = 1, T_e = 0.4$ |
| Generator | $K_g/(1 + sT_g)$ | $0.7 \leq K_g \leq 1$ $1 \leq T_g \leq 2$ | $K_g = 1, T_g = 1$ |
| Sensor | $K_s/(1 + sT_s)$ | $0.001 \leq T_s \leq 0.06$ | $K_s = 1, T_s = 0.01$ |

$$\frac{\Delta V_t(s)}{\Delta V_{\text{ref}}(s)} = \\ = \frac{\left(s^2 K_d + sK_p + K_i\right)\left(K_a K_e K_g\right)(1 + sT_s)}{s\,(1 + sT_a)\,(1 + sT_e)\left(1 + sT_g\right)(1 + sT_s) + \left(K_a K_e K_g K_s\right)\left(s^2 K_d + sK_p + K_i\right)}. \quad (1)$$

Equation (1) is the transfer function got from the block diagram, each $K$ is the gain of its block and each $T$ is the time constant [8].

$$u(s) = \Delta V_e(s)\left(K_p + \frac{K_i}{s} + K_d s\right), \quad (2)$$

where $u(s)$ is the PID controller output signal and $\Delta Ve(s)$ is the voltage error (PID controller input signal).

## 3. Optimization techniques

### 3.1. Flower pollination algorithm (FPA)

Artificial intelligence optimization techniques are based on the nature inspired and biological system. In industrial and engineering applications, they are now trend in solving difficult problems with complex constraints to find the optimal solution [14, 15].

In 2012, Xin-She Yang observed the behavior of flower plants in the pollination process in nature for reproduction, so he was inspired by this process and mimicked it. As a result, an FPA is introduced as an optimization algorithm [9]. He also programmed the FPA MATLAB code that is used in this paper.

This algorithm has 4 assumptions or rules [18]:

i. Levy flights performed by pollinators which carry pollen with the global pollination process can be considered as cross and biotic pollination.
ii. Local pollination can be also considered as abiotic and self-pollination.
Note: classification of pollination can be shown in Fig. 4.
iii. Constancy of flowers is proportional to the resemblance of the two involved flowers and can be considered as the reproduction probability.
iv. In all activities of pollination, a fraction $q$ can be considered in local pollination due to the physical proximity and other factors such as wind. A switch probability $p \in [0, 1]$ can control global and local pollination.

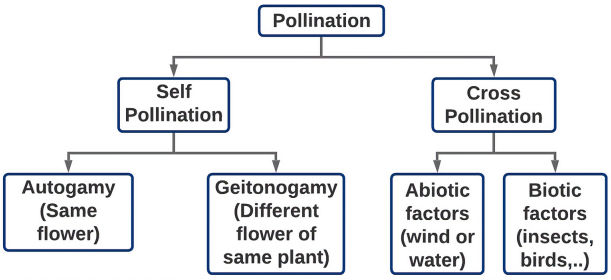


Fig. 4. Classification of pollination

The mathematical model of an FPA can be presented as a Pseudo code which is explained by the flowchart in Fig. 5 [19].

In the Pseudo code [9]:

– Equation (3) can be used to perform local pollination and Equation (4) can be used to perform global pollination.

$$y_i^{t+1} = y_i^t + P(y_i^t - f^*), \tag{3}$$

$$y_i^{t+1} = y_i^t + \epsilon(y_j^t - y_k^t). \tag{4}$$

– To get a good response for different applications, put $P = 0.8$.
– A population size $n$ is typically from 10 to 25, put $n = 20$.
– MaxIteration = 2000.
– Initialize the population/solutions randomly by using the `rand` function in MATLAB.

| Pseudo code for flower pollination algorithm |
|---|
| Initialize objective as minimization. |
| Define the population for n flowers. |
| Find current best solution $f^*$ in the initial population. |
| Describe the switch probability $P \in [0, 1]$. |
| **While** ($t <$ MaxIteration) |
| **For** $i = 1$: $n$ |
| **If** rand $< P$ |
| Define step size $P$ which follow Lèvy distribution. |

Use Equation (3) to perform global pollination.
**else**
Define $\in$ for uniform distribution [0, 1].
Randomly select $j$ and $k$ among all the solution.
Perform local pollination by Equation (4).
**end if**
Calculate new solution.
If calculation solution is better, then update it in population.
**end for**
Get the optimal solution $f^*$.
**end while**

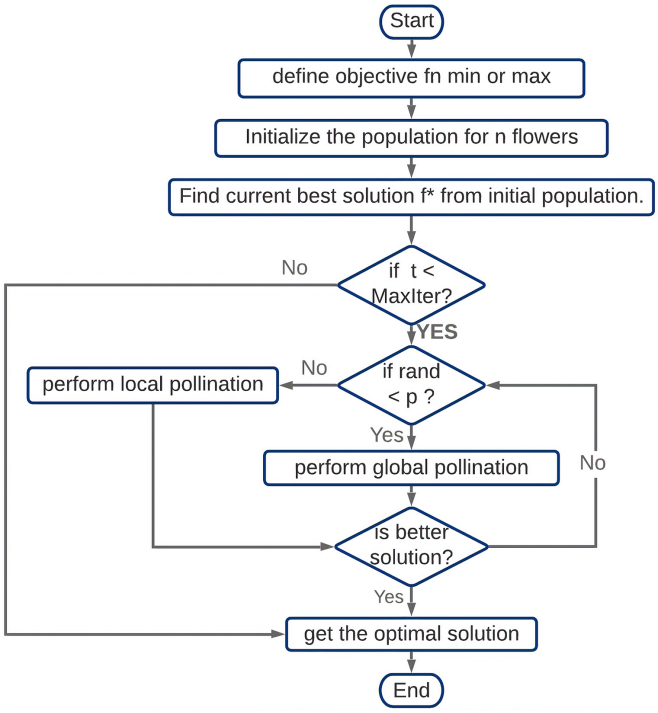


Fig. 5. Flowchart of FPA

### 3.2. Teaching–learning based optimization (TLBO) algorithm

A TLBO algorithm is an unparalleled optimization technique, whose required control parameters are not inscrutable. TLBO's control parameters are population size and number of generations. There are two main technicalities of learning: the first is by a teacher and the second is by interaction with other learners. A class of the educator is depicted aside from TLBO. The

main result is the educator, the example variable is the parameter winded alongside the objective function. TLBO mainly consists of two parts: a teacher phase and learner phase. The teacher phase is the one in which the educator is the main learning source, and learner phase is the one in which the communication between students is the main learning method.

The confessed function values at the end of the teacher phase are developed and these values are transformed into the input to the learner phase.

In 2015, the TLBO MATLAB code has been developed by S. Mostapha Kalami Heris (Member of Yarpiz Team which is the publisher)

Teacher phase:

Based on his efficacy, a teacher widens the mean impact of a class in a subject educated by him. In every iteration of "$i$" we assume that there are "$m$" subjects, "$n$" students, $N_{i,j}$ is the mean outcome of students in a different subject of "$j$". The leading global outcome $X_{tot}\_K_{bst}$, "$i$" is the product of the influential learner $K_{bst}$ and comprises all the coordinated subjects in the blended population of learners.

The teacher is nonetheless regarded as a remarkably educated character and teaches students to obtain outstanding results. The algorithm uses the leading result as the teacher. The difference between the docent outcome of the teacher and the actual mean outcomes of each subject is calculated by:

$$\mathit{Diff}\_\mathrm{Mean}_{j,k,i} = r_i \left( X_{j,k_{\mathrm{best}},i} - T_f N_{j,i} \right), \tag{5}$$

where $X_{j,k_{\mathrm{best}},i}$ is the result of the best learners (i.e. teacher) in the subject $j$, $T_f$ is the teaching factor that decides the mean value to be turned on and its dimension is 1 or 2, $r_i$ is a random value between 0 and 1, $T_f$ is not considered as a planned parameter but clarified by:

$$T_f = \mathrm{round}\ [1 + \mathit{rnd}.\ (0, 1)\ \{1\}]\,. \tag{6}$$

In the teacher's phase the actual solution is refreshed according to $\mathit{Diff}\_\mathrm{Mean}_{j,k,i}$ as follows:

$$X'_{j,k,i} = X_{j,k,i} + \mathit{Diff}\_\mathrm{Mean}_{j,k,i}\,. \tag{7}$$

Learner phase:

Learners increase their abilities by mixing. If a learner has more experience than others, he/she inscribes advanced items. Taking into account a population size "$n$," the learning parameters are the following:

Selecting two learners would randomly presume $P$ and $Q$, so,

$$X\ tot.'_{P,i} \neq X\ tot.'_{Q,i},$$

where $X\ tot.'_{P,i}$ is the updated value of $X\ tot._{P,i}$ and $X\ tot.'_{Q,i}$ is the updated value of $X\ tot._{Q,i}$.

$$X\ tot.''_{j,P,i} = X'_{j,P,i} + r_i \left( X'_{j,P,i} - X'_{j,Q,i} \right). \tag{8}$$

TLBO is explained by the flowchart in Fig. 6 [20] and [21].

The flowchart shows the optimization criterium and the stop conditions. It also shows the used function and parameters which helped to create the simulation code.

– Population size $n$ equals 50.
– Max number of iterations equals 1000.
– Initialize the population/solutions randomly by using the rand function in MATLAB.

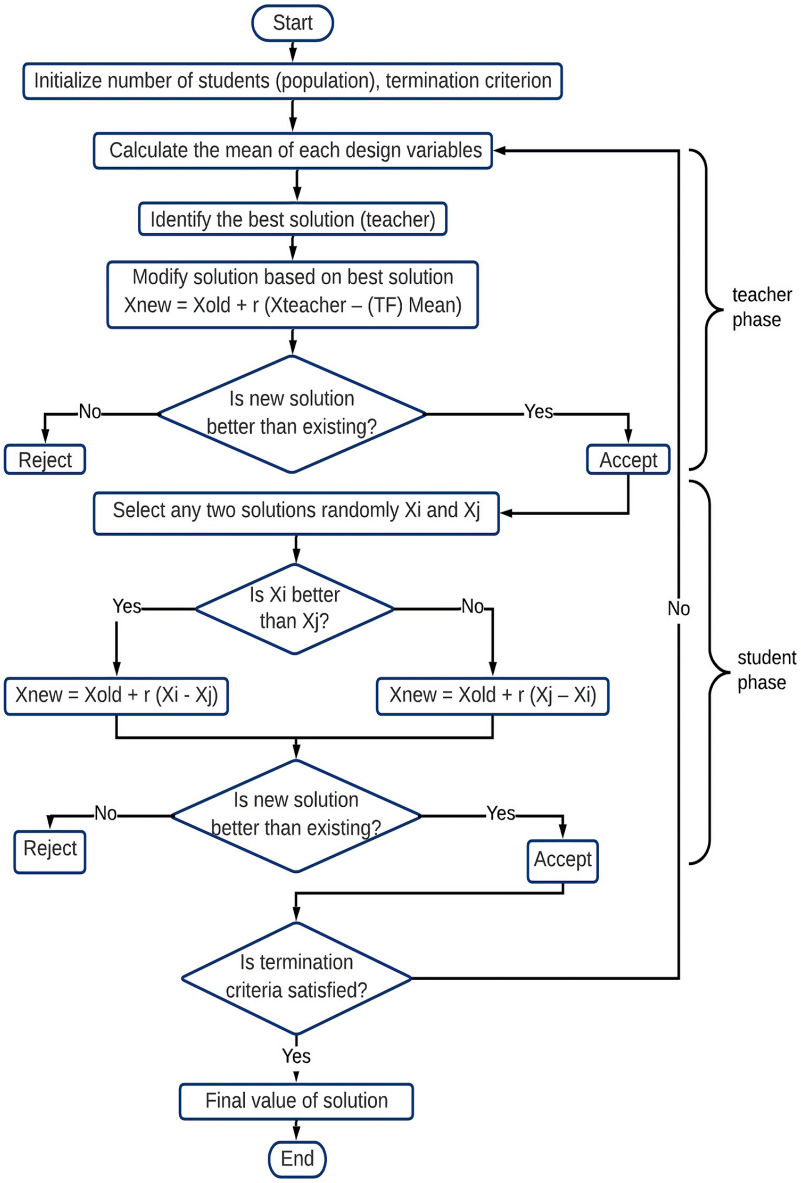


Fig. 6. Flowchart of TLBO

### 3.3. Harmony Search (HS) optimization algorithm

HS is a new meta-heuristic technique. It was introduced by Geem *et al*. in 2001, the Harmony Search (HS) algorithm is insufflated by the basic standards which the musicians use in harmony impromptu. The HS is distinctively characterized by straightforwardness and effectiveness. Recently, it has been effectively utilized in different fields, for example, function optimization,

mechanical structure design, pipe network optimization, and optimization of data classification systems.

As it is known, when artists create harmony, they attempt different potential blends of the musical tones that have been put away in their memory. This quest for the ideal harmony is actually similar to the process of searching for the optimal solution to an engineering problem.

The used Harmony Search algorithm MATLAB code in this research has been developed by Sajjad Yazdani.

Fig. 7 shows the flowchart of the HS algorithm, in which there are four principal steps involved [22].

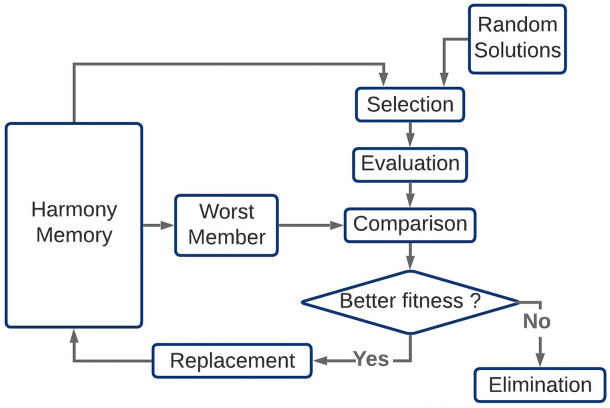


Fig. 7. Flowchart of HS

Step 1: HS Memory (HM) initialization. The initial HM is a set of randomized solutions to the problems of optimization that are taken into account. The HM with the size of $N$ can be seen as following for an n-dimension problem:

$$\mathrm{HM} = \begin{bmatrix} x_1^1, & x_2^1, & \ldots, & x_n^1 \\ x_1^2, & x_2^2, & \ldots, & x_n^2 \\ & \vdots & & \\ x_1^{\mathrm{HMS}}, & x_2^{\mathrm{HMS}}, & \ldots, & x_n^{\mathrm{HMS}} \end{bmatrix},$$

where $[x_1^i, x_2^i, \ldots, x_n^i]$ $(i = 1, 2, \ldots, \mathrm{HMS})$ is the candidate for a solution. The HMS normally varies from 50 to 100.

Step 2: Develop a new HM $[x_1', x_2', \ldots, x_n']$ solution. The Harmony Memory Considering Rate (HMCR) is used to generate each part of this solution, $x_j'$. The probability of choosing an element from the HM members is the definition of the HMCR. Thus 1-HMCR is the possibility of its random generation. If $x_j'$ comes from the HM, the j-th of a random HM component is selected and further mutated according to the pitch change rate (PAR). The PAR determines the possibility of an HM nominee being mutated. As can be seen, the $[x_1', x_2', \ldots, x_n']$ improvement is rather similar to the production of offspring in the Genetic Algorithms (GAs) with the mutation and crossover operations. The GA nevertheless produces new chromosomes using one (mutation) or

two (crossover) existing chromosomes, whereas new solutions in the HS system are completely utilized by all HM members.

Step 3: Update the HM. The new solution from Step 2 is evaluated. If it fits more than the worst member of the HM, it will replace it. Otherwise, it is removed.

Step 4: Repeat step 2 to step 3 until the desired number of iterations is fulfilled.

Harmony search parameters:

– HMS = 100, which is harmony memory size (population number),
– $Bw = 0.2$,
– HMCR = 0.95, harmony memory considering rate,
– PAR = 0.3, pitch adjustment rate,
– MaxItr = 1000, which is maximum number of iteration.

Pseudo code for Harmony Search algorithm

```
/* HM initialization */
for (i = 1; i ≤ HMS; i++)
    for (j = 1; j ≤ n; j++)
        Randomly initialize x_j^i in HM.
    endfor
endfor
/* End of HM initialization */
Repeat
    /* Construction and evaluation of new solution candidate x*/
    for (j = 1; j ≤ n; j++)
        if (rand(0, 1) < HMCR)
            Let x_j in x be the jth dimension of a randomly selected HM member.
                if (rand(0, 1) < PAR)
                    Apply pitch adjustment distance bw to mutate x_j: x_j = x_j ±rand (0, 1)*bw
                endif
        else
            Let x_j in x be a random value.
        endif
    endfor
    Evaluate the fitness of x: f(x).
    /* End of construction and evaluation of new solution candidate x */
    /* HM update */
    if (f(x) is better than the fitness of the worst HM member)
        Replace the worst HM member with x.
    else
        Disregard x.
    endif
    /* End of HM update */
Until a preset termination criterion is met.
```

### 3.4. Local unimodal sampling (LUS)

One of the modern heuristic optimization techniques is the optimization depending on an LUS algorithm. Because numerous optimization algorithms utilize local sampling with a fixed sampling range; there is a danger of stalling out in the local optima, thus no global solution can be reached. This issue can be solved by utilizing the LUS optimization technique which reduces the sampling range as optimization steps forward [23].

Various steps involved in an LUS algorithm are

Step 1: initialization of a random current position $\vec{x}$ in the search space.

Step 2: Set the initial search range $\vec{d}$ for the whole search space:

$$\vec{d} \leftarrow \vec{B}_{up} - \vec{B}_{down}.$$

Step 3: Repeat the process below until you reach the fitness threshold.

1. Choose a random vector $\vec{a} \sim U - \vec{d}, \vec{d}$.
2. The new position $\vec{y} = \vec{x} + \vec{a}$.
3. If the position $\vec{x}$ is more fit than the position $\vec{y}$, update the position $\vec{x} \leftarrow \vec{y}$.
4. Reduce the range of the search $\vec{d}$ as $\vec{d}_{\text{new}} = Q \cdot \vec{d}$.

The used Local Unimodal Sampling MATLAB code in this research has been programmed by Magnus Erik Hvass Pedersen.

A Local Unimodal Sampling (LUS) optimizer by Pedersen performs localized sampling of the search-space with a sampling range that initially covers the entire search-space and is decreased exponentially as optimization progresses. LUS works especially well for optimization problems where only short runs can be performed.

Local unimodal sampling parameters:

– acceptable fitness = 0.000001,

– max evaluations = 1000,

– initial solution is selected randomly within the limits by MATLAB functions.

## 4. Comparison between different optimization techniques

The different optimization techniques mentioned in the previous section will be used to determine the values of the PID controller parameters to optimally improve the transient response of an AVR system, and the results will be discussed and compared.

These techniques will be tested under different values of gain of the generator ($Kg$) to get a robust controller. Depending on load, $Kg$ shall range from 0.7 to 1. Four different cases (different values of $Kg$: 1, 0.9, 0.8, and 0.7) will be tested.

The comparison will depend on the values of the steady state error (the main objective of system under study) as that single objective optimization is used. Max overshoot, settling time and rise time will be included in the comparison as Secondary effect. All results will be shown in the following curves and tables.

### 4.1. Results

Notice that:

– sampling time = 10 s,

– steady state error = reference value which equals 1 p.u – actual value of voltage,

– error is measured at time = 10 s, and it is the same point time for all cases.

#### 4.1.1. Without controller

Figure 8 shows the transient response of the AVR system while using no controller at different *Kg*, values of the steady state error, max overshoot, rise time, settling time are concluded from the curves and shown in the following Table 2.

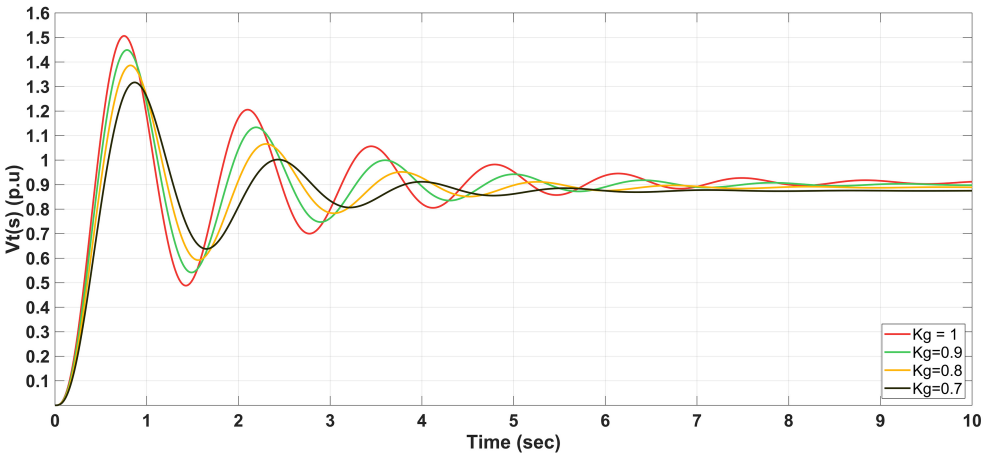


Fig. 8. Vt(s) vs. time curves without controller

Table 2. Transient response parameters without controller

| *Kp* | *Ki* | *Kd* | *Kg* | Error (p.u) | Max overshoot (p.u) | Rise time (s) | Settling time (s) |
|---|---|---|---|---|---|---|---|
| – | – | – | 1.0 | 0.0881 | 1.5066 | 0.4631 | ∞ |
| | | | 0.9 | 0.102 | 1.4515 | 0.4971 | ∞ |
| | | | 0.8 | 0.1108 | 1.3868 | 0.5293 | ∞ |
| | | | 0.7 | 0.1249 | 1.3183 | 0.5766 | ∞ |

#### 4.1.2. FPA

In Figure 9, the PID controller optimized by the FPA is used, so it is obvious that the response has been improved, and this is proved by the values shown in the following Table 3.

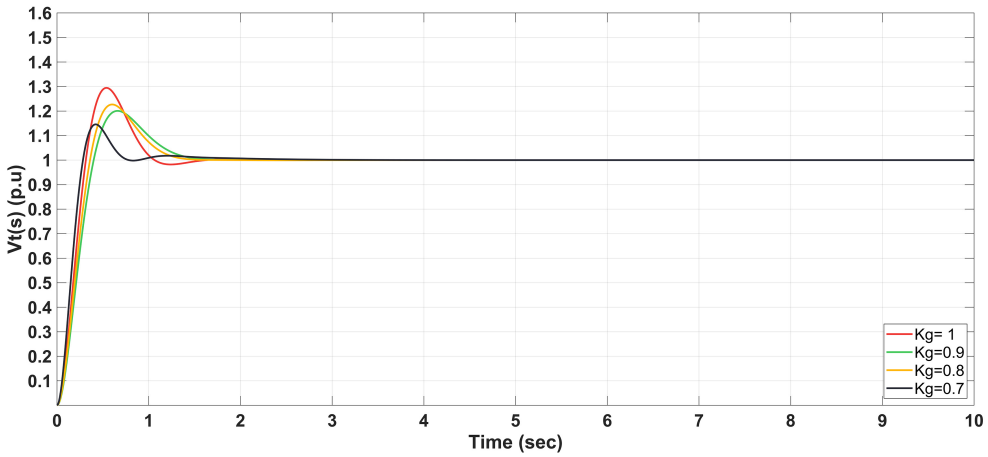


Fig. 9. Vt(s) vs. time curves using FPA

Table 3. Transient response parameters in the case of using FPA

| ***Kp*** | ***Ki*** | ***Kd*** | ***Kg*** | **Error (p.u)** | **Max overshoot (p.u)** | **Rise time (s)** | **Settling time (s)** |
|---|---|---|---|---|---|---|---|
| 1.0882 | 1.4486 | 0.217 | 1.0 | 1.55E–15 | 1.2972 | 0.3025 | 1.000 |
| | | | 0.9 | 3.43E–15 | 1.2033 | 0.399 | 1.351 |
| | | | 0.8 | 6.28E–13 | 1.2326 | 0.3595 | 1.231 |
| | | | 0.7 | 4.75E–13 | 1.1479 | 0.2750 | 0.687 |

#### 4.1.3. TLBO

In Figure 10, the PID controller optimized by TLBO is used and the transient response parameters concluded from the curves are shown in the Table 4.

Table 4. Transient response parameters in the case of using TLBO

| ***Kp*** | ***Ki*** | ***Kd*** | ***Kg*** | **Error (p.u)** | **Max overshoot (p.u)** | **Rise time (s)** | **Settling time (s)** |
|---|---|---|---|---|---|---|---|
| 1.0157 | 1.4691 | 0.2 | 1.0 | 2.22E-16 | 1.3077 | 0.3214 | 1.515 |
| | | | 0.9 | 6.71E-16 | 1.2742 | 0.3623 | 1.192 |
| | | | 0.8 | 2.30E-16 | 1.3104 | 0.3025 | 1.371 |
| | | | 0.7 | 3.53E-16 | 1.2998 | 0.3403 | 1.651 |

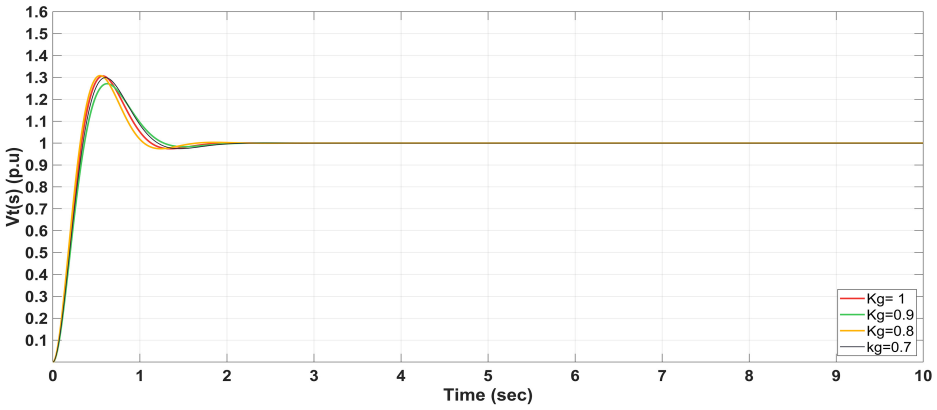


Fig. 10. Vt(s) vs. time curves using TLBO

#### 4.1.4. HS

In Figure 11, the PID controller optimized by HS is used (Table 5).

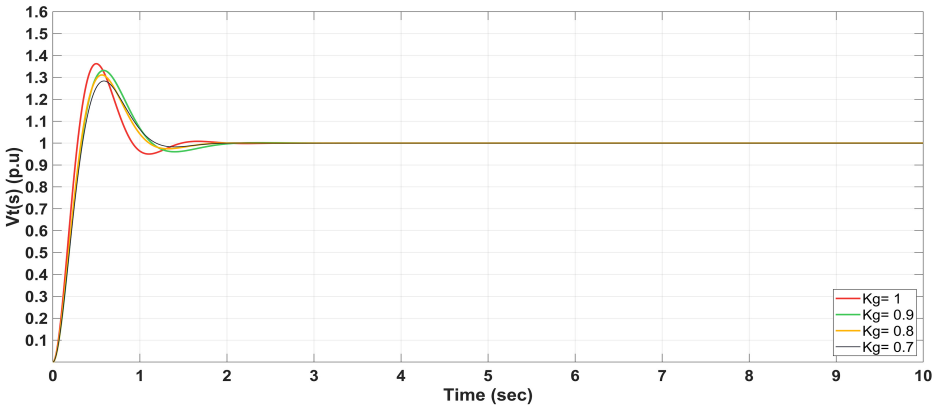


Fig. 11. Vt(s) vs. time curves using HS

Table 5. Transient response parameters in the case of using HS

| ***Kp*** | ***Ki*** | ***Kd*** | ***Kg*** | **Error (p.u)** | **Max overshoot (p.u)** | **Rise time (s)** | **Settling time (s)** |
|---|---|---|---|---|---|---|---|
| 1.2937 | 1.8989 | 0.2372 | 1.0 | 4.00E-15 | 1.3631 | 0.2741 | 1.345 |
| | | | 0.9 | 2.48E-15 | 1.3339 | 0.3219 | 1.69 |
| | | | 0.8 | 1.92E-15 | 1.320 | 0.3122 | 1.51 |
| | | | 0.7 | 2.48E-15 | 1.279 | 0.3311 | 1.12 |

#### 4.1.5. LUS

In Figure 12, the PID controller optimized by LUS is used (Table 6).

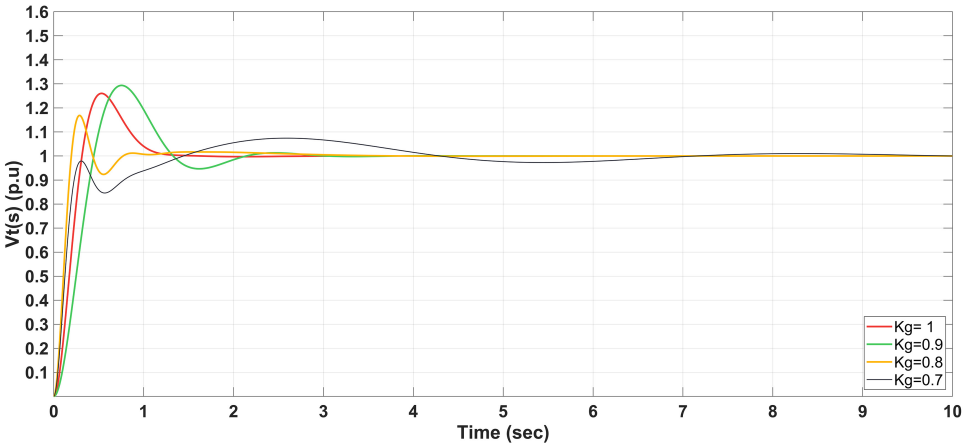


Fig. 12. Vt(s) vs. time curves using LUS

Table 6. Transient response parameters in the case of using LUS

| ***Kp*** | ***Ki*** | ***Kd*** | ***Kg*** | **Error (p.u)** | **Max overshoot (p.u)** | **Rise time (s)** | **Settling time (s)** |
|---|---|---|---|---|---|---|---|
| 1.0382 | 1.4694 | 0.2323 | 1.0 | 3.82E-11 | 1.2629 | 0.3025 | 1.095 |
| | | | 0.9 | 8.52E-09 | 1.2945 | 0.4351 | 1.972 |
| | | | 0.8 | 9.18E-07 | 1.1708 | 0.1942 | 0.718 |
| | | | 0.7 | 9.09E-07 | 1.0761 | 1.4269 | 6.22 |

By comparing the values of the steady state error shown in the previous tables, it is concluded that TLBO is the best optimization technique that gives the smallest error at all tested values of $Kg$.

## 5. Conclusion

This paper investigates the stability analysis of an AVR system. The transient response of the AVR without a PID controller and with a PID controller based on different optimization techniques has been studied and it is found that the dynamic response has been improved significantly in the case when a PID controller exists. The results of different techniques used to optimize the steady state error (main objective) are compared at different $Kg$ and it is obvious that TLBO is better than other techniques. The steady state error is the main purpose but not the only one. Other dynamic characteristics which affect the stability of the system such as max overshoot, settling time and

rise time are included in the study which means that the transient stability has been taken into consideration.

**References**

[1] Mahmut Temel Özdemir, Vedat Çelik, *Stability analysis of the automatic voltage regulation system with PI controller*, Journal of Sakarya University Institute of Science, vol. 21, no. 4, pp. 698–705 (2017).

[2] Challapuram Yaswanth Reddy *et al.*, *Laboratory implementation of Automatic Voltage Regulator*, Biennial International Conference on Power and Energy Systems: Towards Sustainable Energy (PESTSE), Bangalore, pp. 1–6 (2016).

[3] Saidy M., Huges F.M., *A predictive integrated voltage regulator and power system stabilizer*, Elsevier proceedings on Electrical Power and Energy Systems, vol. 7, no. 2, pp. 101–111 (1995).

[4] Rakesh Singh Lodhi, Abhishek Saraf, *Survey on PID Controller Based Automatic Voltage Regulator*, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, vol. 5, no. 9, pp. 7424–7429 (2016).

[5] Haluk Gozde, Cengiz Taplamacioglu M., *Comparative performance analysis of artificial bee colony algorithm for automatic voltage regulator (AVR) system*, Journal of the Franklin Institute, vol. 348, no. 8, pp. 1927–1946 (2011).

[6] Gaing Z.-L., *A particle swarm optimization approach for optimum design of PID controller in AVR system*, IEEE Trans. Energy Convers., vol. 19, no. 2, pp. 384–391 (2004).

[7] Elgard O.I., *Electric Energy Systems Theory*, New York, Mc Graw-Hill (1982).

[8] Mukherjee V., Ghoshal S.P., *Intelligent particle swarm optimized fuzzy PID controller for AVR system*, Electron. Power Syst. Res., vol. 77, no. 12, pp. 1689–1698 (2007).

[9] Sambariya D.K., Tripti Gupta, *Optimal Design of PID Controller for an AVR System Using Flower Pollination Algorithm*, Journal of Automation and Control, vol. 6, iss. 1, pp. 1–14 (2018).

[10] dos Santos Coelho L., *Tuning of PID controller for an automatic regulator voltage system using chaotic optimization approach*, Chaos, Solitons and Fractals, vol. 39, no. 4, pp. 1504–1514 (2009).

[11] Qader M.R., *Identifying the optimal controller strategy for DC motors*, Archives of Electrical Engineering, vol. 68, no. 1, pp. 101–114 (2019).

[12] Eswaramma K., Surya Kalyan G., *An Automatic Voltage Regulator AVR System Control using a P-I-DD Controller*, Journal of Advance Engineering and Research Development, vol. 4, no. 6, pp. 499–506 (2017).

[13] Aström K.J., Hägglund T., *PID Controllers: Theory, Design, and Tuning*, Instrument Society of America, USA (1995).

[14] Yang X.-S., *Flower pollination algorithm for global optimization*, Lecture Notes in Computer Science, vol. 7445, pp. 240–249 (2012).

[15] Chiroma H., Shuib N.L.M., Muaz S.A., Abubakar A.I., Ila L.B., Maitama J.Z., *A review of the applications of bio-inspired flower pollination algorithm*, Procedia Computer Science, vol. 62, pp. 435–441 (2015).

[16] Mihailo Micev, Martin Calasan, Diego Oliva, *Fractional Order PID Controller Design for an AVR System Using Chaotic Yellow Saddle Goatfish Algorithm*, Mathematics, vol. 8, no. 1182 (2020), DOI: 10.3390/math8071182.

[17] Sahib M.A., *A novel optimal PID plus second order derivative controller for AVR system*, Engineering Science and Technology, an International Journal, vol. 18, iss. 2, pp. 194–206 (2015).

[18] Abdel-Raouf Osama, Abdel-Baset M., el-Henawy I., *A new hybrid flower pollination algorithm for solving constrained global optimization problems*, International Journal of Applied Operational Research-An Open Access Journal, vol. 4, no. 2, pp. 1–13 (2014).

[19] Sambariya D.K., Gupta T., *Optimal design of PID controller for an AVR system using monarch butterfly optimization*, International Conference on Information, Communication, Instrumentation and Control (ICICIC), Indore, India, pp. 1−6 (2017).

[20] Priyambada S., Mohanty P.K., Sahu B.K., *Automatic voltage regulator using TLBO algorithm optimized PID controller*, 2014 9th International Conference on Industrial and Information Systems (ICIIS), Gwalior, India, pp. 1–6 (2014).

[21] Niknam Taher, Rasoul Azizipanah-Abarghooee, Narimani Mohammad Rasoul, *A new multi objective optimization approach based on TLBO for location of automatic voltage regulators in distribution systems*, Engineering Applications of Artificial Intelligence, vol. 25, pp. 1577–1588 (2012).

[22] Askarzadeh Alireza, Rashedi Esmat, *Harmony Search Algorithm. Recent Developments in intelligent Nature-Inspired Computin*g (2017), DOI: 10.4018/978-1-5225-2322-2.ch001.

[23] Mohanty Pradeep, Sahu Binod, Panda Sidhartha, Kar Sanjeeb, Mishra Nandan, *Performance Analysis and Design of Proportional Integral Derivative Controlled Automatic Voltage Regulator System Using Local Unimodal Sampling Optimization Technique*, pp. 566–576 (2012), DOI: 10.1007/ 978-3-642-35380-2_66.