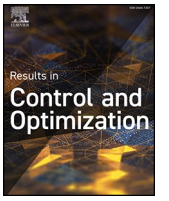
Results in Control and Optimization 13 (2023) 100306

Contents lists available at ScienceDirect

Results in Control and Optimization

journal homepage: www.elsevier.com/locate/rico



Control and performance analyses of a DC motor using optimized PIDs and fuzzy logic controller

Nelson Luis Manuel *, Nihat İnanç, Murat Lüy

Department of Electrical & Electronics Engineering, Kırıkkale University, Kırıkkale 71450, Turkey

ARTICLE INFO

Keywords:
DC motor speed control
Metaheuristic algorithms
Equilibrium optimizer
Particle swarm optimization
Teaching-learning-based optimization
Differential evolution
Genetic algorithm

ABSTRACT

Based on the no-free-lunch theorem, researchers have been proposing optimization algorithms for solving complex engineering problems. This paper analyzes the performance of five metaheuristics: Equilibrium Optimizer (EO), Particle Swarm Optimization (PSO), Teaching-Learning-Based Optimization (TLBO), Differential Evolution (DE), and Genetic Algorithm (GA) in fine-tuning the gains of a Proportional-Integral-Derivative (PID) to control the speed of a DC motor. The selected metaheuristics, in addition to being from distinct classes, are well established in their respective groups. The methods and findings of this study can be summarized in three phases. First, the mathematical model of the DC motor is deduced. Second, detailed descriptions of the aforementioned algorithms are presented. Furthermore, the structures of the applied controllers are discussed. Third, comparisons based on statistical indicators and analyses in the time and frequency domains, in addition to robustness and load disturbance tests, are performed. The results revealed that if a sufficient number of runs is given for each metaheuristic, despite being in different runs, all algorithms are able to propose the same optimal gain values. TLBO presented the highest speed, while GA and DE were the slowest in finding optimal values. Additionally, the results were compared with the Opposition-Based Learning Henry Gas Solubility Optimization (OBL/HBO)-based PID, reported to have better results than some previously published works on this topic, and a Fuzzy Logic Controller (FLC). The five optimized controllers obtained approximately the same results and outperformed the OBL/HGO-based PID, but the FLC was superior compared to the metaheuristic-based PIDs.

1. Introduction

A Direct Current (DC) motor is a machine that converts electrical energy into mechanical energy [1]. There are many types of DC motors, such as brushed DC motors, brushless DC motors, steppers, among others. DC motors can be found in different applications, such as in electric vehicles [2], robotics [3], and home automation [4]. DC motors are widely used due to their good performance, low power and easy-to-control structure [4].

The task for the control engineer is to ensure that the output of the DC motor follows a predefined reference both under normal operating conditions and in the presence of disturbances. The angular speed of the DC motor can be controlled by adjusting the electrical power supplied to it [5]. This may seem like a trivial task; however, it is not, considering all the DC motor's input-output combinations [1].

* Corresponding author.
E-mail addresses: nelsonluismanuel@gmail.com, 198804001@kku.edu.tr (N.L. Manuel).

https://doi.org/10.1016/j.rico.2023.100306
Received 15 May 2023; Received in revised form 25 August 2023; Accepted 21 September 2023
Available online 29 September 2023

Many types of controllers have been applied to control DC motors: Akbar et al. [6] implemented the Model Reference Adaptive Control, Chaouch et al. [7] applied PI controller, Adel et al. [8] and Gabriel Găşpăresc [9] implemented PID controller, Guo and Mohamed [10] applied ANFIS Based Hybrid PID Controller, Ahmad et al. [11] implemented Integral state feedback, Hekimoglu [12] applied Fractional Order PID, for DC motor control. Among other controllers, PID is usually preferred in DC motor applications due to its simplicity and ease of implementation in a real system [13].

With the advances in computing, optimization methods based on artificial intelligence, have been widely used. Approaches such as artificial neural networks and fuzzy logic have been frequently applied in several areas of engineering including control. For instance, Cheon et al. designed a deep learning neural network (DNN) controller based on the Deep Belief Network (DBN) algorithm to control a DC motor [14]. Sardhalia et al. presented a neural network controller (NNC) and a neural network tuner (NNT) for speed regulation of DC motor [15]. Chen et al. applied a fuzzy back propagation (BP) neural network (NN) to regulate the speed of a DC motor in an electric energy vehicle [16]. In artificial neural networks, the training process and convergence time are of paramount importance and significantly long [17]. On the other hand, in the fuzzy logic system, obtaining the proper membership functions for the system to be controlled is generally dependent on data analysis, model tuning and designer skills [18].

To guarantee the desired performance of the system, PID controller depends on the tuning of its three parameters, namely, proportional gain coefficient (K_P), integral gain coefficient (K_I), and derivative gain coefficient (K_D) [19]. Tuning these three parameters properly is not easy. Initially, the process of tuning the PID controller parameters was done either by trial-and-error or by traditional methods. The trial-and-error method, in addition to being time-consuming, is often not able to find the optimal gain values. Traditional methods, viz. Ziegler-Nichols (ZN) and Cohen-Coon (CC) also do not guarantee optimal gain values, resulting in undesirable overshoots and oscillations of the system output [17].

In recent years, metaheuristic algorithms have been proposed to determine the PID controller gains. The use of metaheuristic algorithms in the PID controller optimization process brings improvements over traditional methods such as reduction of tuning time and possibilities to find optimal gains. Metaheuristic algorithms are optimization techniques that mimic ethological, biological, chemical, or even physical phenomena to solve complex problems, which are usually difficult to be addressed by deterministic methods [20]. In general, there is no single criterion for classifying metaheuristic optimization algorithms, although the popularly used classification criterion is based on the different sources of inspiration [21]. According to this criterion, metaheuristic algorithms can be divided into four categories: physics-based algorithms, swarm-intelligence-based algorithms, human-related techniques, and evolutionary or bio-inspired algorithms [22].

Physics-based metaheuristic algorithms are developed on the basis of mathematical modeling of physical laws and phenomena (e.g., electrical charge, gravity, etc.) [21]. Equilibrium Optimizer (EO) [23], Henry Gas Solubility Optimization (HGSO) [24], Gravitational Search Algorithm (GSA) [25], Heat Transfer Search (HTS) [26], and Artificial Chemical Reaction Optimization Algorithm (ACROA) [27], are some of the well-known algorithms under the category of physics-based metaheuristics.

Swarm-intelligence-based algorithms mimic the intelligence of schools, herds, swarms, or flocks of creatures in nature. The main inspiration of these algorithms comes from the social behavior of animal groups, such as locating other individuals, searching for food, and flocking [28]. Particle Swarm Optimization (PSO) [29], Artificial Bee Colony (ABC) [30], Ant Colony Optimization (ACO) [31], and Firefly Algorithm (FA) [32], are some of the renowned swarm-intelligence-based metaheuristics.

Human-related metaheuristic algorithms are established on the basis of mathematical models that mimic different human activities. Teaching-Learning-Based Optimization (TLBO) is regarded as the most famous algorithm in the human-based metaheuristics [33]. TLBO simulates knowledge transfer and interaction between instructor and students in a classroom [34]. Poor and Rich Optimization (PRO) [35], Human Mental Search (HMS) [36], Doctor and Patient Optimization (DPO) [37], among others, also represent some of the algorithms that fall under the class of human-related metaheuristics.

Evolutionary algorithms, mimic the concept of natural or biological evolution [28]. Generally, in evolutionary metaheuristics, an initial population is randomly generated to start the search process. A cost function or fitness function is used to evaluate the fitness of each individual in the population. In subsequent steps the population evolves towards the best global solution. This process continues until the termination condition is satisfied [21]. Genetic Algorithm (GA) [38] and Differential Evolution [39] are commonly accepted as well-established metaheuristics under the category of evolutionary algorithms [22].

Many authors have been proposing new optimization algorithms motivated by the no-free-lunch (NFL) theorem [40]. According to NFL theorem, there is no such metaheuristic algorithm capable of solving all types of optimization problems. The question that this paper attempts to address is the following: In light of the many metaheuristics that have been published in the literature, is there a metaheuristic algorithm that is superior to the others in optimizing the PID controller, considering a DC motor as the plant?

To answer this question, the authors selected five metaheuristic algorithms for the optimal design of PID controllers: EO, PSO, TLBO, DE, and GA. These five optimization algorithms were considered because they belong to distinct categories and are well established in their respective classes. The class of physics-based metaheuristics is represented by EO; the class of swarm-intelligence-based algorithms is represented by PSO; the class of human-related algorithms is represented by TLBO; and the class of evolutionary algorithms is represented by both DE and GA, because the two go hand in hand when it comes to popular metaheuristics under the class of evolutionary algorithms. The availability of sources and codes for faithful replication of these algorithms' performances is another factor considered in their selection. The optimized PID controllers are applied to control a DC motor and subsequently their performances are analysed. For the verification purposes, a Fuzzy Logic Controller (FLC) is designed and comparisons with PID controllers based on the other state-of-art metaheuristic algorithms are carried out. The main contributions of this research paper can be listed as follows:

- Optimization algorithms representing different classes (physics-based, evolution-based, swarm-intelligence-based, and human-based metaheuristics) are applied to fine-tune the PID gains to control a DC motor.
- To make the research self-explanatory, the authors attempted to discuss in full the key aspects of the algorithms and controllers applied in this work.
- Since the initial population is randomly generated, the considered algorithms are submitted to 20 independent runs. In this manner, all algorithms are given the opportunity to display their best performance.
- The acquired results are based on statistical indicators, time and frequency domain analyses, load disturbances, and robustness tests.
- Using the MATLAB/Simulink software, a FLC was designed with two inputs (the error and its derivative) and one output (the control signal), all of which were represented by triangular membership functions. For inference and defuzzification, the Mamdani and centroid methods were considered.
- Furthermore, comparisons with OBL/HBO-based PID, reported as having better results than other previously published works for the same mathematical model of the DC motor, are presented.

The remainder of this paper is structured as follows: In Section 2, the open-loop model of the DC motor is presented. In Section 3, the fitness function used for the optimization of PID controllers is expressed. In Section 4, the aforementioned algorithms are explained in detail. In Section 5, descriptions of PID and FLC controllers are presented in detail. Section 6 is dedicated to the simulation and discussion of the obtained results. Finally, in Section 7, the derived conclusions and some recommendations on the topic are shared.

2. DC motor open-loop modeling

As for the type of excitation, DC motors are divided into two types: self-excited and separately (or externally) excited. In this article, the second type is considered. Shaft speed and armature voltage are considered as motor output and input, respectively. Armature voltage is used to keep the motor at the desired speed. In Fig. 1, an equivalent circuit diagram of separately excited DC motor is shown.

The induced voltage $e_b(t)$, for a constant flux, is directly proportional to the angular velocity $\omega(t)$ of the rotor shaft, multiplied by the constant k_b.

$$e_b(t) = k_b \frac{d\theta(t)}{dt} = k_b \omega(t) \tag{1}$$

By applying KVL to the DC motor armature circuit, the armature voltage $e_a(t)$ can be determined as follows:

$$e_a(t) = L_a \frac{di_a(t)}{dt} + R_a i_a(t) + e_b(t) \tag{2}$$

The torque $T(t)$ produced by the DC motor is directly proportional to the armature current $i_a(t)$ multiplied by the torque coefficient k_t, according to Eq. (3) below:

$$T(t) = k_t i_a(t) \tag{3}$$

Taking into account energy conservation concepts, the torque produced can also be determined by the equivalent sum of torques due to friction and inertia. Therefore:

$$T(t) = J \frac{d\omega(t)}{dt} + B\omega(t) = k_t i_a(t) \tag{4}$$

In Eq. (4), the load torque T_L is not included. This is because the load torque, for analysis purposes, is taken as an external input to the DC motor system. Load torque is considered as a disturbance and its effects are taken into account by applying superposition theorem [41].

Applying Laplace transform to Eqs. (1)—(4), considering zero initial conditions, will result in:

$$E_b(s) = K_b \Omega(s) \tag{5}$$

$$E_a(s) = (L_a s + R_a) I_a(s) + E_b(s) \tag{6}$$

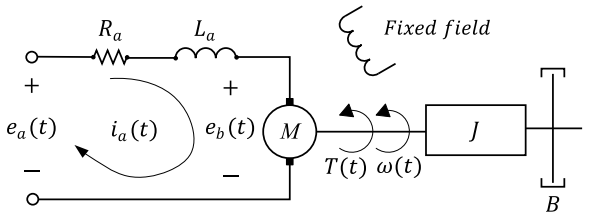


Fig. 1. DC motor equivalent circuit diagram.

$$T(s) = (Js + B)\Omega(s) = K_t I_a(s) \tag{7}$$

From Eqs. (5)—(7), a relationship between the angular speed $\Omega(s)$ and the armature voltage $E_a(s)$ of the DC motor can be established:

$$G_{E_a}(s) = \frac{\Omega(s)}{E_a(s)} = \frac{K_t}{(L_a s + R_a)(Js + B) + K_b K_t} = \frac{K_t}{L_a J s^2 + (R_a J + L_a B)s + (R_a B + K_b K_t)},\ for\ T_L = 0 \tag{8}$$

The relationship expressed in (8) is the open-loop transfer function of the DC motor. Likewise, the transfer function $G_{T_L}(s)$ in Eq. (9), represents the relationship between angular speed $\Omega(s)$ and load torque $T_L(s)$. Fig. 2 shows the DC motor block diagram.

$$G_{T_L}(s) = \frac{\Omega(s)}{T_L(s)} = -\frac{(L_a s + R_a)}{(L_a s + R_a)(Js + B) + K_b K_t} = -\frac{(L_a s + R_a)}{L_a J s^2 + (R_a J + L_a B)s + (R_a B + K_b K_t)},\ for\ E_a = 0 \tag{9}$$

The motor parameters considered in this paper are presented in Table 1. These parameters are chosen in order to allow analogies with other researches [12,24,42-44].

Substituting the parameters of the DC motor in Eqs. (8) and (9) by the values provided in Table 1, the open-loop transfer function is given as in the expression (10).

$$G_{open-loop}(s) = \begin{cases} \dfrac{\Omega(s)}{E_a(s)} = \dfrac{15}{1.08s^2 + 6.1s + 1.63},\ for\ T_L = 0 \\ \dfrac{\Omega(s)}{T_L(s)} = -\dfrac{(2700s + 400)}{1.08s^2 + 6.1s + 1.63},\ for\ E_a = 0 \end{cases} \tag{10}$$

3. Fitness function and constraints

Generally, when using heuristic algorithms to optimize a problem, it is necessary to define a fitness function. The fitness function is considered as a measure of the optimality of a given solution [45].

For tuning the PID controller parameters, several fitness functions have been used. Some fitness functions are defined in terms of rise time T_r, overshoot M_P, settling time T_s, and steady-state error E_{ss} of the motor output [46,47]. Others are defined in terms of the error integral [12,24,42-44].

In this study, the Integral Time Absolute Error (ITAE) function is adopted as the fitness function. The same function was adopted in other similar works [12,24,42-44]. Using ITAE as objective function, the standard form of the PID controller optimization problem can be described as in (11).

$$\begin{aligned} &Minimize\ (ITAE) = Minimize\left(\int_0^{t_{sim}} t|e(t)|dt\right) \\ &\text{Subject to} \\ &L_b \le K_P, K_I, K_D \le U_b \end{aligned} \tag{11}$$

where t is the simulation time and varies from 0 to t_{sim}; $e(t)$ is the error, which is equal to the difference between set point speed $\omega_{ref}(t)$ and motor speed $\omega(t)$. L_b and U_b are lower and upper bounds, respectively.

4. Applied optimization algorithms

4.1. Equilibrium optimizer

Equilibrium Optimizer (EO), is a heuristic optimization algorithm proposed by Faramaezi et al. (2019), inspired by dynamic mass balance of particles within a control volume [23].

In the EO algorithm, the iterative search process is initially preceded by the random formation of an initial population of candidate solutions according to Eq. (12).

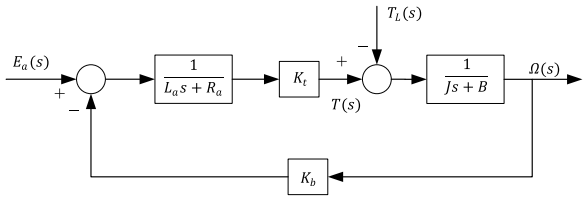


Fig. 2. DC motor system block diagram.

Table 1
DC motor parameters.

Parameters	Values
B	0.0022 N m s rad^{-1}
J	0.0004kg m^2
K_b	0.05 V s rad^{-1}
K_t	0.015 N m A^{-1}
L_a	2.7 H
R_a	0.4 Ω

$$C_i^{initial} = C_{min} + rand_i(C_{max} + C_{min}),\ i = 1,\ 2,\ 3,\ \ldots,\ n \tag{12}$$

$C_i^{initial}$ represents the initial concentration of the i^{th} particle; $rand_i$ is a random vector whose elements are in the interval $[0,\ 1]$; C_{min} and C_{max} are, respectively, the lower and upper bound vectors of the i^{th} particle and n is the number of particles constituting the population.

While the initial population is generated using Eq. (12), the concentrations are updated during the course of the optimization using Eq. (13).

$$\overrightarrow{C}_i^{\,it+1} = \overrightarrow{C}_{eq}^{\,it} + \left(\overrightarrow{C}_i^{\,it} - \overrightarrow{C}_{eq}^{\,it}\right).\overrightarrow{F}_i^{\,it} + \frac{\overrightarrow{G}_i^{\,it}}{\overrightarrow{\lambda}_i^{\,it} V}\left(1 - \overrightarrow{F}_i^{\,it}\right) \tag{13}$$

Where $\overrightarrow{C}_{eq}^{\,it}$ is the concentration of the i^{th} particle before the update (iteration it) and $\overrightarrow{C}_i^{\,it+1}$ is the updated concentration of the i^{th} particle (iteration $it+1$); $\overrightarrow{\lambda}$ refers to the inverse of the residence time and V is the control volume. $\overrightarrow{C}_{eq}^{\,it}$ is a vector chosen randomly among the elements belonging to the equilibrium pool. The equilibrium pool has the following structure:

$$\overrightarrow{C}_{eq,pool}^{\,it} = \left[\overrightarrow{C}_{eq(1)}^{\,it}, \overrightarrow{C}_{eq(2)}^{\,it}, \overrightarrow{C}_{eq(3)}^{\,it}, \overrightarrow{C}_{eq(4)}^{\,it}, \overrightarrow{C}_{eq(avg)}^{\,it}\right] \tag{14}$$

Where the vectors $\overrightarrow{C}_{eq(1)}^{\,it}$, $\overrightarrow{C}_{eq(2)}^{\,it}$, $\overrightarrow{C}_{eq(3)}^{\,it}$ and $\overrightarrow{C}_{eq(4)}^{\,it}$ are the first four best solutions of the generation it; the vector $\overrightarrow{C}_{eq(avg)}^{\,it}$ is the average of the first four best four solutions. $\overrightarrow{F}$ is the exponential term, it is responsible for maintaining a balance between exploitation and exploration during the optimization process. The vector $\overrightarrow{F}$ is determined by Eq. (15).

$$\overrightarrow{F}^{it+1} = e^{-\overrightarrow{\lambda}_i^{it}(t-t_0)} \tag{15}$$

The time t is defined in such a way that it decreases as the number of iterations increases and is given by Eq. (16).

$$t = \left(1 - \frac{it}{it_{max}}\right)^{\left(a_2 \frac{it}{it_{max}}\right)} \tag{16}$$

The terms it and it_{max} represent the current iteration number and the maximum number of iterations, respectively. The term a_2 is a constant that allows management of the algorithm's exploitation ability. The expression of t_0 is defined by Eq. (17).

$$\overrightarrow{t}_0 = \frac{1}{\overrightarrow{\lambda}}\ln\left[-a_1 sign(\overrightarrow{r} - 0.5)\left(1 - e^{-\overrightarrow{\lambda} t}\right)\right] + t \tag{17}$$

The term a_1 is a constant that allows management of the algorithm's exploration ability. High values of a_1 imply high exploration ability and consequently low exploitation performance. Likewise, high values of a_2 result in improvements in exploitation, but low performance in exploration. Parameter a_1 is set to be 2 and parameter a_2 is set to be 1 [23,48].

The component $sign(\overrightarrow{r} - 0.5)$ influences the direction of exploration and exploitation. The elements of the random vector $\overrightarrow{r}$ vary between 0 and 1. If Eq. (17) is substituted into Eq. (15), the vector $\overrightarrow{F}$ can be expressed by Eq. (18).

$$\overrightarrow{F}^{it} = a_1 sign(\overrightarrow{r} - 0.5)\left(e^{-\overrightarrow{\lambda}^{it} t} - 1\right) \tag{18}$$

One of the preponderant factors in the EO algorithm is the generation rate. The generation rate G_i contributes to achieving optimal solutions through the improvement of the exploitation phase. Its expression is given by Eq. (19) [23].

$$\overrightarrow{G}_i^{it} = \overrightarrow{G}_0^{it} e^{-\overrightarrow{\lambda}_i^{it}(t-t_0)} \tag{19}$$

Where

$$\overrightarrow{G}_0^{it} = \overrightarrow{GCP}_i^{it}\left(\overrightarrow{C}_{eq}^{it} - \overrightarrow{\lambda}^{it}\overrightarrow{C}_i^{it}\right) \tag{20}$$

$$\overrightarrow{GCP}_i^{it} = \begin{cases} 0.5r_1 & r_2 \geq GP^{it} \\ 0 & r_2 < GP^{it} \end{cases} \tag{21}$$

The initial value of the generation rate $\overrightarrow{G}_0$, is given by Eq. (20). Eq. (15) is the expression for determining the Generation Rate Control Parameter ($\overrightarrow{GCP}$). The $\overrightarrow{GCP}$ vector dictates the contribution or not of the generation rate in the optimization process of a given particle. This is possible thanks to the existence of another parameter called Generation Probability (GP). The terms r_1 and r_2 are uniformly distributed random values belonging to the interval $[0,\ 1]$. To promote a reasonable balance between exploration and exploitation, GP is chosen to be 0.5 [23].

It is worth noting that in the expression for updating the concentration of a particle, Eq. (11), three terms are present: The first term is the equilibrium concentration, which is randomly chosen among the elements of the equilibrium pool. The second and the third terms are responsible for changes in the concentration (position) of a given particle.

The second term is responsible for the diversification process, thus avoiding premature convergence to local solutions. The third term, which plays the opposite role of the previous one, is in charge of the exploitation process, thus allowing for improvements at a local level of the search space.

The resulting effect of the second and third terms is such that, if both have the same signs, a large variation occurs and, therefore, a greater contribution is given to exploration. On the other hand, if the signs are opposite, small variation occurs and, therefore, a greater contribution is given to exploitation. Fig. 3 represents the flowchart of the Equilibrium Optimizer algorithm.

4.2. Particle swarm optimization

Particle Swarm Optimization (PSO), is a stochastic-type optimization algorithm, inspired by the collective behavior of some animals, such as schools of fish or flocks of birds. PSO, since its proposal by Kennedy and Eberhart (1995), has undergone a multitude of

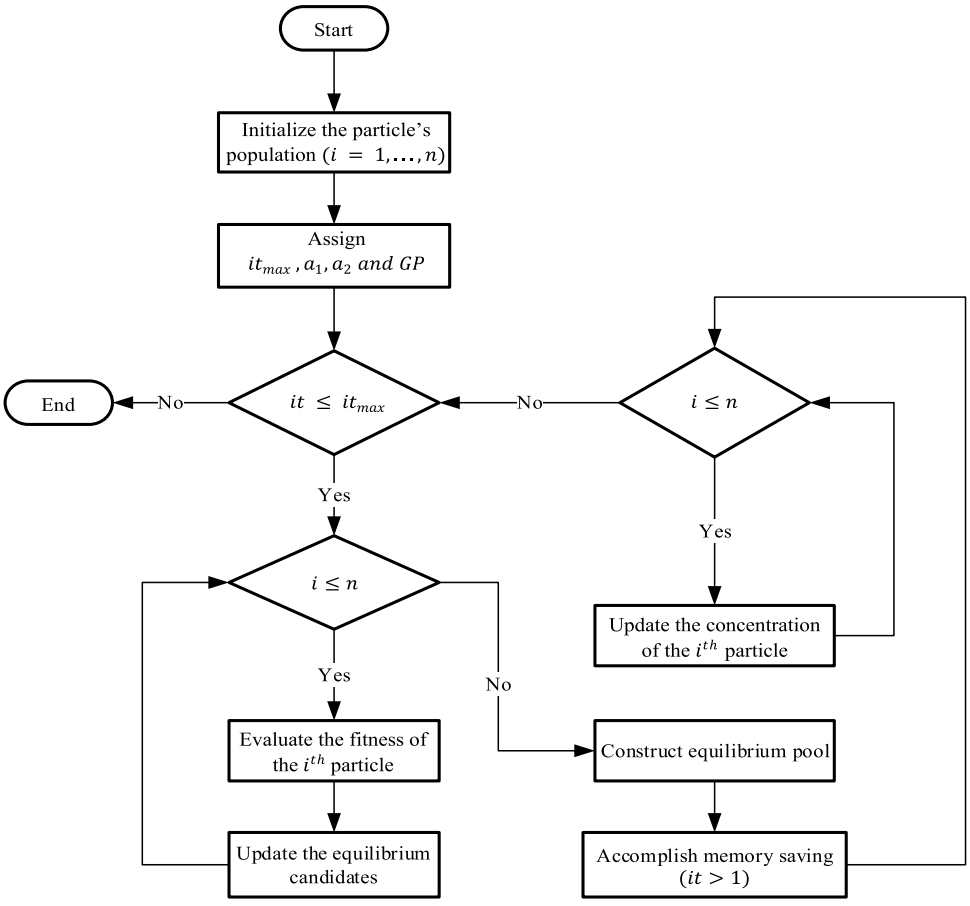


Fig. 3. Flowchart of the EO algorithm.

improvements in order to allow its application in different areas [49].

In the PSO algorithm, each particle is identified by its position and velocity. Particles update their positions in the environment for survival reasons, such as searching for food, searching for safe and predator-free environments, among others. The position of each particle is updated by adjusting the velocity associated with it. There are two versions of PSO algorithm: local version and global version [49]. In the local version, particles update their velocities without taking into account the global information of the swarm. On the other hand, in the global version the information about the best position in the swarm is taken into account when updating the velocity of each particle. In this paper, the global version is considered, i.e., the velocity, and hence the position of each particle is updated taking into account the particle's own experience and the experience of the swarm as a whole.

As with other stochastic optimization methods, the initial population of the swarm in the PSO algorithm is constructed randomly. Having the initial population formed and respecting the restrictions imposed on the search space, the continuation of the optimization process is carried out by Eqs. (22) and (23).

$$\vec{v}_i^{\,it+1} = w^{it+1}\vec{v}_i^{\,it} + c_1\vec{r}_1\left(\vec{p}_{best,i}^{\,it} - \vec{x}_i^{\,it}\right) + c_2\vec{r}_2\left(\vec{g}_{best}^{\,it+1} - \vec{x}_i^{\,it}\right),\ i = 1,\ 2,\ 3,\ \ldots,\ n \tag{22}$$

$$\vec{x}_i^{\,it+1} = \vec{x}_i^{\,it} + \vec{v}_i^{\,it+1},\ i = 1,\ 2,\ 3,\ \ldots,\ n \tag{23}$$

Where $\vec{v}_i^{\,it+1}$ is the velocity of the i^{th} particle in the iteration $it+1$ and $\vec{v}_i^{\,it}$ is the velocity of the i^{th} particle in the iteration it; w is the inertia coefficient; c_1 and c_2 are acceleration coefficients; $\vec{r}_1$ and $\vec{r}_2$ are vectors whose elements are random values belonging to the interval $[0,\ 1]$; $\vec{p}_{best,i}^{\,it}$ is the best position of the i^{th} particle so far; $\vec{g}_{best}^{\,it+1}$ is the best position achieved by the swarm so far; $\vec{x}_i^{\,it}$ and $\vec{x}_i^{\,it+1}$ are, respectively, the position in the previous iteration and the position in the current iteration of the i^{th} particle.

It can be seen that expression (22) contains three terms: The first part is called the inertia term. It is from this term that the influence of the velocity of the previous iteration can be included in the next movement. The scalar w dictates how influential prior experience should be on current experience. The second one is the "cognitive" term. This term represents the particle's ability to make its decisions based on its own best experiences. The last component is the social term. This is the term that represents the ability of a particle to communicate with other particles within the swarm.

Briefly, in the PSO algorithm, the next position of a given particle is the result of the inertia of its current position, its own experience and the experience of the swarm as a whole. Fig. 4 below shows a flowchart of the PSO algorithm.

4.3. Real-Coded genetic algorithm

As pointed out by [50], studies on genetic algorithm (GA) were first carried out around the 1960s and 1970s by scientist and engineer John Holland and his collaborators at the University of Michigan. GA is a stochastic-type optimization algorithm inspired by Charles Darwin's theory of evolution [51]. In GA, as in previous algorithms, the optimization process starts with the generation of an initial population. Each member present in the population is called a chromosome. Solution vectors that participate in the generation of other solutions are called parents. Newly generated solutions are called offsprings. The evolution of the initial population over generations is achieved by the GA evolutionary operators, namely: selection, crossover and mutation.

The genetic algorithms (GAs) can be classified as Binary-Coded GA (BCGA) and Real-Coded GA (RCGA). In BCGA, the problem decision variables are represented in binary (zeros and ones). If the optimization problem to be solved has solutions in a continuous search space, using BCGA implies discretizing the search space. The need to convert decision variables to their binary equivalents causes a number of limitations: obtained solutions have fixed precision, hamming cliff problem associated with binary numbers, among others [52].

Unlike BCGA, RCGA does not need to convert the decision variables of the problem to be solved. Therefore, RCGA is a better option when it comes to optimization problems that have continuous search spaces. Due to the advantages presented by RCGA in relation to BCGA and considering that the PID controller gains optimization problem is defined in a continuous search space, in this paper the RCGA is employed. In the Fig. 5, the flowchart of GA is shown and its main stages are explained in the following subsections.

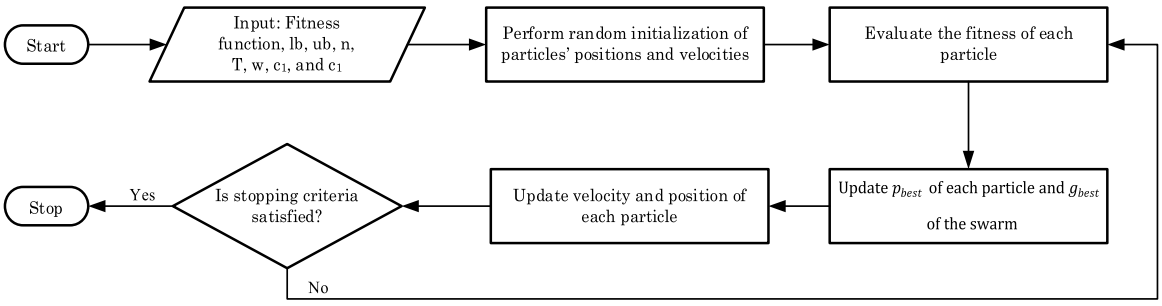


Fig. 4. Flowchart of the PSO algorithm.

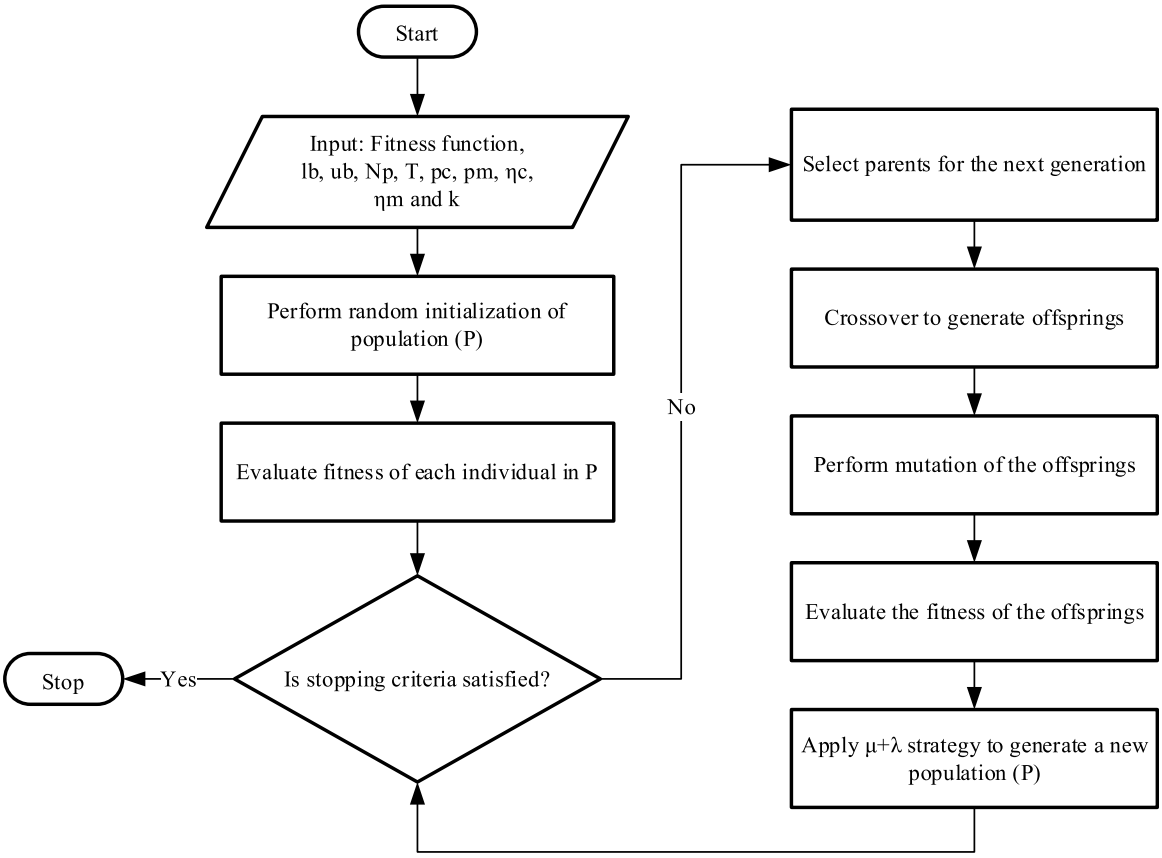


Fig. 5. Flowchart of the GA algorithm.

4.3.1. Population initialization

In GA, the initial population is randomly constructed according to the input data specified for the problem. GA entries include population number (N_P), number of iterations (T), crossover probability (P_c), mutation probability (P_m), crossover distribution index (η_c), mutation distribution index (η_m), tournament size (k), lower bound (l_b) and upper bound (u_b) of decision variables, among other parameters. The decision variables of the initial population chromosomes (X_i) can be stated as in Eq. (24):

$$x_{i,j}^{0} \sim U\left(l_b^j, u_b^j\right),\ j\ \epsilon\ n,\ i\ \epsilon\ N_P \tag{24}$$

where $x_{i,j}^{0}$ represents the j^{th} decision variable of the i^{th} chromosome of the initial population; U is a function that returns uniformly distributed random variables defined in the interval $[l_b^j,\ u_b^j]$; l_b^j and u_b^j are the lower and upper bounds of the j^{th} decision variable, respectively, and n is the number of decision variables of the problem.

4.3.2. Parent selection

In GA, the selection process is responsible for determining which members of the current population should participate in the reproduction of next-generation offsprings. Parents are chosen in the hope that the new generation will be even better than the current one. There are different selection strategy schemes: tournament selection, ranking selection, roulette wheel selection, among others [53]. Tournament selection is one of the most used strategies in GA [53,54]. In tournament selection, a group of individuals is randomly chosen from the current population. The chosen individuals compete against each other and the candidate with the best fitness wins. The number of participants in the competition is called the tournament size [54]. If the tournament size is 2, it is called binary tournament. Although fitter individuals are more likely to be selected for competition, tournament selection gives chance to all individuals in the population, thus promoting population diversity [53]. In this paper, binary tournament selection is applied.

4.3.3. Crossover operator

After the tournaments follows the crossover. The chromosomes that participate in the crossover are the ones that won the tournaments. Crossover is the combination of two candidate solutions (or parents) to generate other solutions (or offsprings) [55]. There are different crossover techniques: blend crossover operator, simulated binary crossover, normal distribution crossover, simplex crossover, parent centric crossover, direction-based crossover, among others [56]. In this article, simulated binary crossover (SBX) is

used. The SBX crossover simulates the operation of single-point binary crossover applied in BCGA [52]. In the SBX crossover, two offsprings ($x_j^{(1,\ t+1)}$ and $x_j^{(2,\ t+1)}$) are obtained from two randomly chosen parents ($x_j^{(1,\ t)}$ and $x_j^{(2,\ t)}$). The mathematical expressions for determining the two offsprings ($x_j^{(1,\ t+1)}$ and $x_j^{(2,\ t+1)}$) are given by Eqs. (25) and (26), respectively.

$$x_j^{(1,\ t+1)} = 0.5\left[\left(1+\beta_j\right)x_j^{(1,\ t)} + \left(1-\beta_j\right)x_j^{(2,\ t)}\right] \tag{25}$$

$$x_j^{(2,\ t+1)} = 0.5\left[\left(1-\beta_j\right)x_j^{(1,\ t)} + \left(1+\beta_j\right)x_j^{(2,\ t)}\right] \tag{26}$$

The parameter β_j is computed as follows:

$$\beta_j = \begin{cases} \left(2u_j\right)^{1/(\eta_c+1)}, & u_j \le 0.5 \\ \left[\dfrac{1}{2\left(1-u_j\right)}\right]^{1/(\eta_c+1)}, & otherwise \end{cases} \tag{27}$$

where u_j is a number randomly chosen for the j^{th} decision variable ($u_j\ \epsilon\ [0,1]$) and η_c is the distribution index for crossover.

4.3.4. Mutation operator

Mutation is the change in the genetic material of an individual of the population [55]. There are different types of mutation strategies: greedy sub tour mutation, exchange mutation, displacement mutation, inversion mutation, scramble mutation, polynomial mutation, among others [57]. In this paper, polynomial mutation is applied. Assuming that the two offsprings reproduced during the crossover ($x_j^{(1,\ t+1)}$ and $x_j^{(2,\ t+1)}$) are subjected to polynomial mutation, their corresponding mutated offsprings ($y_j^{(1,\ t+1)}$ and $y_j^{(2,\ t+1)}$) are determined using Eqs. (28) and (29), respectively.

$$y_j^{(1,\ t+1)} = x_j^{(1,\ t+1)} + \left(u_b^j - l_b^j\right)\delta_j \tag{28}$$

$$y_j^{(2,\ t+1)} = x_j^{(2,\ t+1)} + \left(u_b^j - l_b^j\right)\delta_j \tag{29}$$

The parameter δ_j is obtained as follows:

$$\delta_j = \begin{cases} \left(2r_j\right)^{1/(\eta_m+1)} - 1, & r_j < 0.5 \\ 1 - \left[2\left(1-r_j\right)\right]^{1/(\eta_m+1)}, & r_j \ge 0.5 \end{cases} \tag{30}$$

where r_j is a number randomly chosen for the j^{th} decision variable ($r_j\ \epsilon\ [0,1]$) and η_m is the distribution index for mutation.

4.3.5. Survival strategy

There are different strategies for choosing survivors. In this paper, a strategy called $\mu+\lambda$ is used [58]. Where μ represents the population members that were selected as parents for the generation of offsprings. The λ refers to the offspring population obtained from the crossover and mutation operations. In the $\mu+\lambda$ strategy, population of parents (μ) competes with population of children (λ) for a place in the next generation. From this competition only N_P individuals (either parents or children depending on their fitness) pass to the next generation.

Having the survivors of the generation, the process goes back to the selection operation and this is repeated until the completion criterion is met.

4.4. Differential evolution

Differential Evolution (DE) is a stochastic-type population-based optimization algorithm. DE was proposed by Storn and Price (1996). Inspired by natural selection mechanisms, it is considered to be a simple and powerful algorithm [59]. Each member of the population (also called a chromosome) undergoes two main operations: mutation and recombination (or crossover) [39]. DE works with three types of vectors: target vector, mutant vector and trial vector. Target vector is the vector that is being perturbed, mutant vector is the vector obtained after the mutation operation and trial vector is the vector obtained after the crossover operation. Unlike GA, in DE the mutation operator comes before the crossover. In addition to the main operators (mutation and crossover), a last but not least aspect of DE is the selection strategy. The selection strategy is basically the criterion used to decide whether or not a given chromosome should become a member of the next generation [39].

Let a population composed of N_P D-dimensional vectors be represented as in expression (31):

$$x_i^{(t)} = \left(x_{i1}^{(t)},\ x_{i2}^{(t)}, \ldots, x_{i2}^{(t)}\right),\ i = 1,\ 2,\ \ldots,\ N_P \tag{31}$$

where $x_i^{(t)}$ is the i^{th} chromosome of the t^{th} iteration (or generation); ($x_{i1}^{(t)},\ x_{i2}^{(t)}, \ldots, x_{i2}^{(t)}$) are the decision variables of the i^{th} chromosome in the t^{th} generation and N_P represents the size of the population. As with other population-based algorithms, in DE the initial population

is randomly generated using a uniform distribution. Explanations about the mutation and crossover operators, as well as the selection strategy are given in the following subsections according to [39].

4.4.1. Mutation operator

A mutant vector $v_i^{(t+1)}$ corresponding to a target vector $x_i^{(t)}$ is determined as in expression (32):

$$v_i^{(t+1)} = x_{r1}^{(t)} + F\left(x_{r2}^{(t)} - x_{r3}^{(t)}\right),\ i = 1,\ 2,\ \ldots,\ N_P \tag{32}$$

where $r_1 \neq r_2 \neq r_3$ are haphazardly selected indexes from the current population (i.e., $r_1,\ r_2,\ r_3\ \in \{1,\ 2, \ldots,\ N_P\}$). The target vector $i \notin \{r_1,\ r_2, r_3\}$, hence the population size must be chosen so that the condition $N_P \geq 4$ is satisfied. F is a scaling factor that belongs to the interval $[0,\ 2]$ and plays the role of controlling the amplification of the differential variation.

4.4.2. Crossover operator

In DE, after mutation, the vector goes through the crossover operation. This operation is carried out in order to increase exploration [39]. The vector resulting from the crossover operation is called trial vector. The expression to compute a trial vector $u_i^{(t+1)}$ is given as follows:

$$u_{ij}^{(t+1)} = \begin{cases} v_{ij}^{(t+1)}, & randb(j) \leq P_C \text{ or } j = rnbr(i) \\ x_{ij}^{(t)}, & randb(j) > P_C \text{ and } j \neq rnbr(i) \end{cases},\ i = 1,\ 2, \ldots, N_P;\ j = 1,\ 2, \ldots, n \tag{33}$$

In (35), $randb(j)$ represents the value contained in the j^{th} position of a randomly generated vector, whose values have a uniform distribution and belong to the interval $[0,\ 1]$; P_C is a user-defined constant and represents the crossover probability ($P_C\ \in [0,\ 1]$); $rnbr(i)$ is randomly chosen index $\in \{1,\ 2,\ \ldots,\ n\}$, which guarantees that at least one of the trial vector decision variables $u_{ij}^{(t+1)}$ comes from the mutant vector $v_{ij}^{(t+1)}$.

4.4.3. Selection strategy

In the canonical DE algorithm, greedy selection is adopted [39]. In greedy selection, a one-to-one competition between the target vector $x_i^{(t)}$ and its corresponding trial vector $u_i^{(t+1)}$ is done. The solution with the best fitness wins and goes to the next generation. Greedy selection can be expressed as in (34):

$$x_i^{(t+1)} = \begin{cases} u_i^{(t+1)}, & \text{if } u_i^{(t+1)} \text{ has better fitness value than } x_i^{(t)} \\ x_i^{(t)}, & \text{otherwise} \end{cases},\ i = 1,\ 2, \ldots, N_P \tag{34}$$

In Fig. 6, a flowchart of the DE algorithm is shown. There are different variants of DE algorithm. The variant presented above is called canonical DE and using the notation presented in [39], it can be written as DE/rand/1/bin.

4.5. Teaching-learning-based optimization

Teaching-Learning-Based Optimization (TLBO) is a population-based stochastic optimization technique proposed by Rao et al. (2011). TLBO is inspired by the knowledge transfer process in a classroom environment. One of the merits of TLBO is the reduced number of input parameters required from the user. Making an analogy with other population-based techniques, in TLBO, the

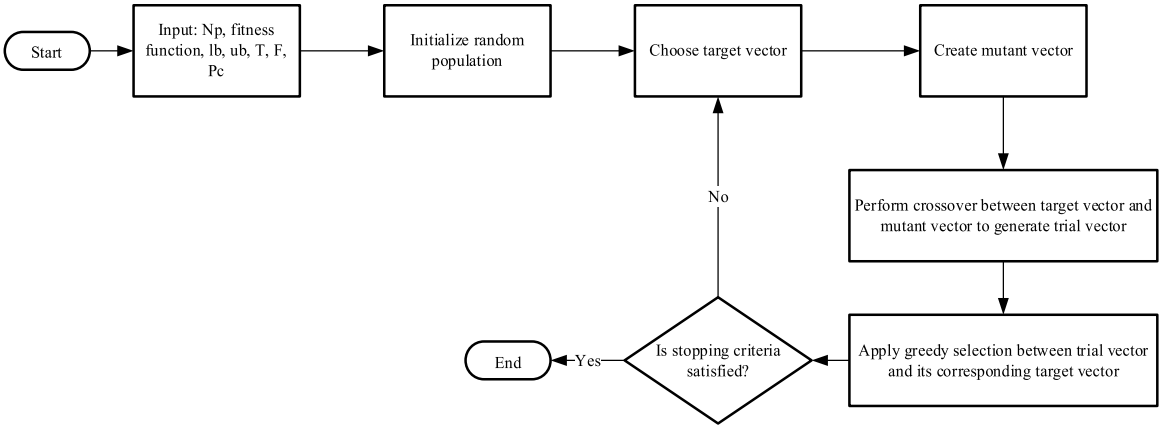


Fig. 6. Flowchart of the DE algorithm.

population is the group of learners, the design variables (or decision variables) are referred to as the different subjects given to learners and the results obtained by the learners are equivalent to their respective fitness. The teacher is the solution with the best fitness so far. The optimization process is separated into two phases: teacher phase and learner phase. In the teacher phase, the teacher (best solution so far) uses his/her knowledge to train the learners (other solutions). It is expected that a good teacher is able to train his/her students in order to obtain better results. The learner phase imitates the process by which students in a class interact with each other in order to clarify their own doubts without the intervention of the teacher. As with the previously presented algorithms, in TLBO an initial population is randomly generated and using the two phases of TLBO the solutions are iteratively modified in order to converge to better solutions. In the next subsections, the mathematical expressions of the two phases are presented.

4.5.1. Teacher phase

A new solution (X_{ij}^{new}) is obtained with the help of the teacher (X_{ij}^{best}) and the mean of the class (X_{ij}^{mean}), as shown in the Eq. (35):

$$X_{ij}^{new} = X_{ij}^{old} + r_{ij}\left(X_{ij}^{best} - T_{F,i}X_{ij}^{mean}\right) \tag{35}$$

where X_{ij}^{new} is the new solution for the i^{th} learner obtained after teacher phase; X_{ij}^{old} is the solution of the i^{th} learner before the class; r_{ij} is a randomly generated number ($r_{ij} \in [0,\ 1]$) and T_F is called teaching factor. The value of teaching factor can either be 1 or 2.

4.5.2. Learner phase

Students increase their knowledge through lectures given by the teacher and through interaction among themselves. A learner interacts with his/her classmates and learns something new in the case that the other learners have more knowledge than he/she does. The expression of the learner phase is given as follows:

$$X_{ij}^{new} = \begin{cases} X_{ij}^{old} + r_{ij}\left(X_{ij}^{old} - X_{ij}^{partner}\right), & \text{if } X_{ij}^{partner} \text{ has better fitness value than } X_{ij}^{old} \\ X_{ij}^{old} - r_{ij}\left(X_{ij}^{old} - X_{ij}^{partner}\right), & \text{otherwise} \end{cases} \tag{36}$$

where X_{ij}^{new} is the new solution for the i^{th} learner obtained after learner phase; X_{ij}^{old} is the solution of the i^{th} learner before interacting with his/her classmate; r_{ij} is a randomly generated number ($r_{ij} \in [0,\ 1]$) and $X_{ij}^{partner}$ is a randomly chosen solution (partner). $X_{ij}^{partner}$ mimics a classmate who tries to help the i^{th} learner, so $X_{ij}^{partner} \neq X_{ij}^{old}$.

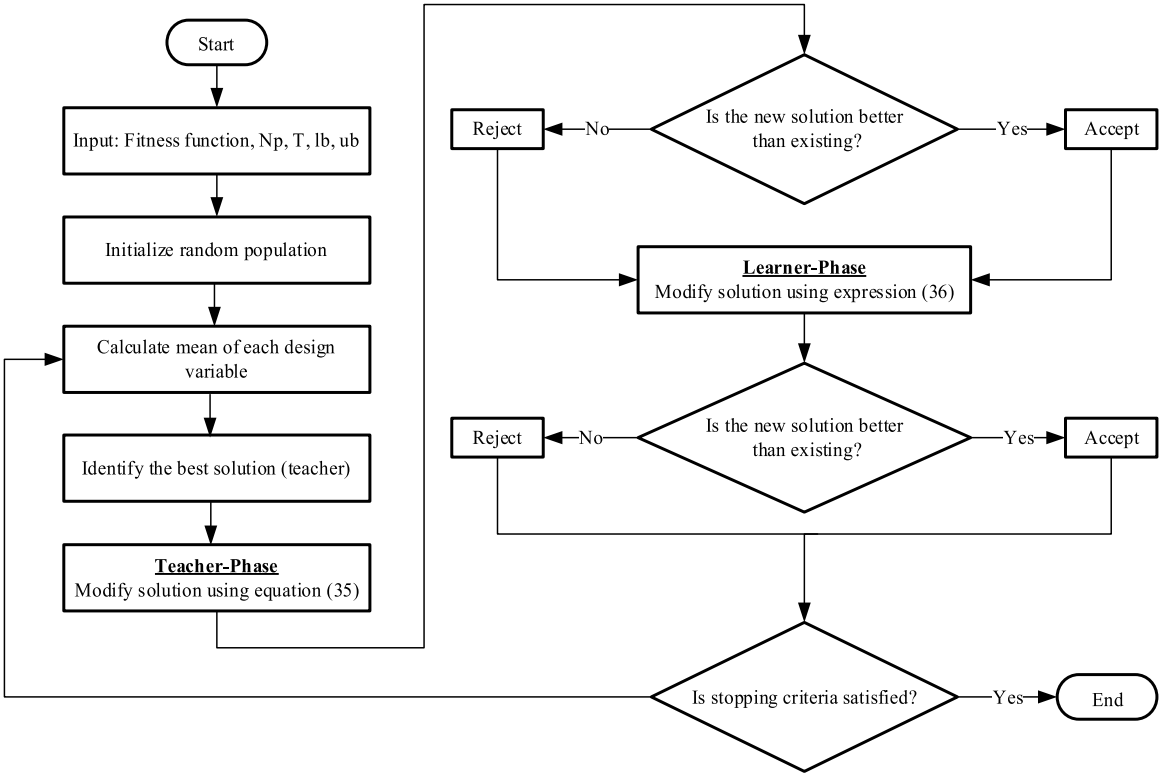


Fig. 7. Flowchart of the TLBO algorithm.

4.5.3. Selection strategy

In TLBO, greedy selection is performed after each phase, i.e., the fitness of the solutions after going through each phase are compared to their values before modification. If the solution obtained after modification is better than the previous one then the current solution is considered, otherwise the previous solution is kept. A flowchart of the TLBO algorithm is shown in Fig. 7.

5. Controllers

5.1. Proportional–integral–derivative controller

The Proportional-Integral-Derivative (PID) controller is widely applied in different industrial control problems and has advantages such as simplicity and ease of implementation [60]. Most of closed-loop systems are controlled using PID or small variations of it [61]. The mathematical expression of a PID controller in the frequency domain is given as follows [62]:

$$G_{PID}(s) = K_P + \frac{K_I}{s} + K_D s \tag{37}$$

where K_P, K_I and K_D are proportional gain, integral gain and derivative gain, respectively. The control objective is such that the equality

$$\lim_{t\to\infty} e(t) = \lim_{t\to\infty}\left(\omega_{ref} - \omega(t)\right) = 0 \tag{38}$$

is satisfied. Where $e(t)$ represents the tracking error signal; ω_{ref} is the reference speed and $\omega(t)$ is the output speed of the DC motor. The control law in the frequency domain $U(s)$ and in the time domain $u(t)$ is given by Eqs. (39) and (40), respectively [62].

$$U(s) = E(s)\left(K_P + \frac{K_I}{s} + K_D s\right) \tag{39}$$

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau)d\tau + K_D \frac{de(t)}{dt} \tag{40}$$

PID is a combination of three terms: The first term represents the proportional (P) control, the second is the integral (I) term, it is responsible for ensuring that the steady-state error becomes zero and the third is the derivative (D) term, its purpose is to improve the stability of the feedback system [61]. Taking into account the parameters of the DC motor considered in this paper, the closed-loop transfer function ($G_{closed-loop}(s)$) with the PID controller is given by expression (41). Fig. 8 shows the closed-loop block diagram of a PID-controlled DC motor.

$$G_{closed-loop}(s) = \begin{cases} \dfrac{\Omega(s)}{\Omega_{ref}(s)} = \dfrac{15\left(K_D s^2 + K_P s + K_I\right)}{1.08s^3 + 6.1s^2 + 1.63s + 15\left(K_D s^2 + K_P s + K_I\right)}, \; for\; T_L = 0 \\ \dfrac{\Omega(s)}{T_L(s)} = -\dfrac{(2700s + 400)s}{1.08s^3 + 6.1s^2 + 1.63s + 15\left(K_D s^2 + K_P + K_I\right)}, \; for\; E_a = 0 \end{cases} \tag{41}$$

Given the plant to be controlled, the design of the PID controller consists of tuning its gains so that the DC motor follows the designed speed. In this paper, DC motor control using PID based on different optimization algorithms is performed. Fig. 9 shows the scheme used to implement five different controllers, viz. EO-based PID, PSO-based PID, RCGA-based PID, DE-based PID, and TLBO-based PID.

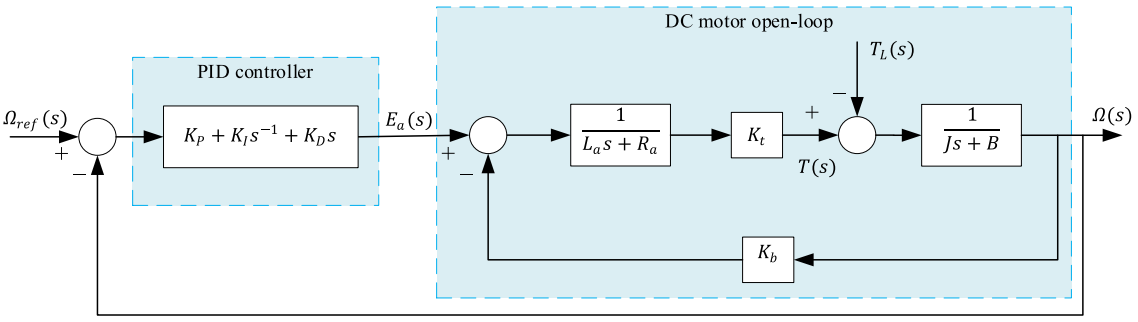


Fig. 8. Block diagram of a PID-controlled DC motor.

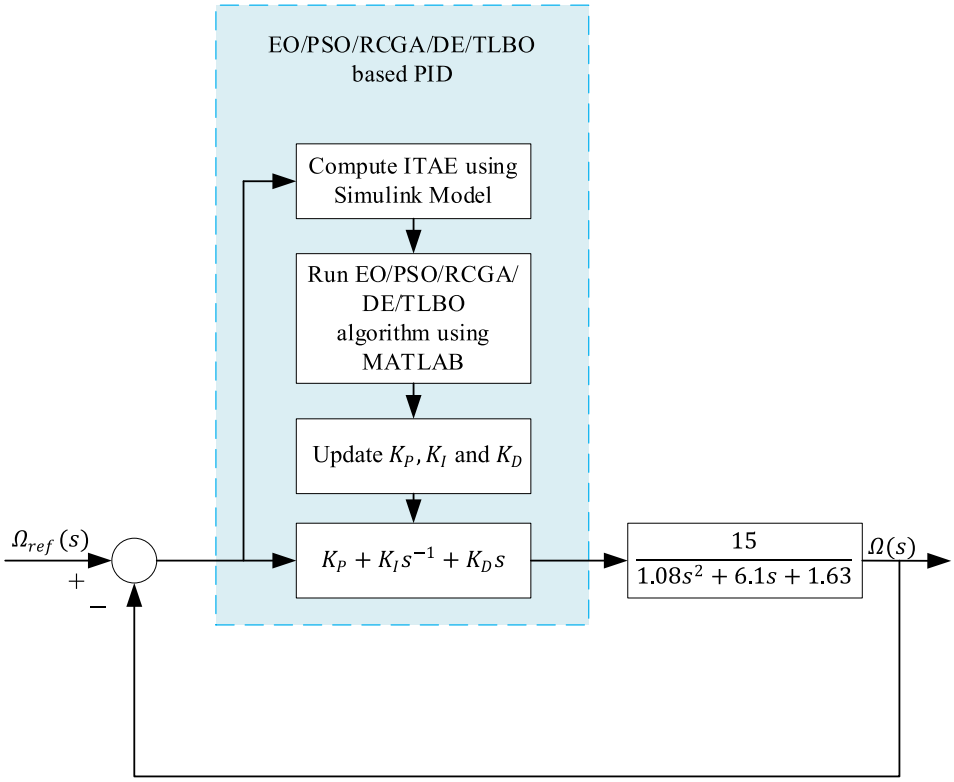


Fig. 9. Scheme used for DC motor control using different PID's.

5.2. *Fuzzy logic controller (FLC)*

The first mathematical foundation of fuzzy logic or fuzzy set theory was given by Professor Zadeh of the University of California in 1965 [63]. Fuzzy logic, on which fuzzy control is based, unlike Boolean logic, deals with situations of uncertainty and imprecision that are similar to the feeling and inference process of human beings [64].

In fuzzy logic, a variable is composed of a set of values that are represented by linguistic expressions. Again, the numerical value of a particular set is defined by a membership function, which varies between zero and one, unlike classical logic, which only accepts either one or zero (true or false, respectively) [63,65]. Although fuzzy theory deals with uncertainties, it is based on a solid mathematical foundation [65]. The Fuzzy Logic Controller (FLC) provides a means from which it is possible to convert an expert-based linguistic control strategy into an automatic control strategy [66]. Since it does not rely on the mathematical model of the plant, FLC can be suitable in situations where the model of the system to be controlled is unknown or ill-defined. Furthermore, FLC can also handle complex multi-dimensional nonlinear systems or cases where sensor signals are not precise [65].

The FLC structure can be divided into four main components: fuzzifier, knowledge base, inference mechanism and defuzzifier [67]. The fuzzification process consists of deriving the membership functions of the inputs and representing them with linguistic variables. This process is analogous to a mapping between the classical set and the fuzzy set [64]. Membership functions can be of different types, such as triangular waveform, trapezoidal waveform, Gaussian waveform, among others. Selection of the type of membership function to be used depends on the application [64].

The knowledge base is a collection of IF-THEN rules, resulting in an algorithm capable of describing what action or output should be taken to achieve the control goal [68].

The inference mechanism, infers the fuzzy value to be sent to the defuzzification unit, according to the fuzzified inputs and the information present in the knowledge base. There are three inference methods commonly used in fuzzy systems: Mamdani, Sugeno and Tsukamoto. The Mamdani method is the first proposed and is the most popular in practice as well as in the literature [69]. In this paper, the Mamdani inference method using the Max-Min composition is applied. To illustrate the idea behind this method, a simple system with two rules where each rule comprises two antecedents and one consequent is considered. This is equivalent to a fuzzy system with two inputs and one output. However, the idea used here can be extended to fuzzy systems with any number of antecedents and any number of consequents. Let x_1 and x_2 be the values of the first and second input, respectively, and y the output, described by a collection of r linguistic rules of type IF-THEN, as shown in expression (42):

$$IF\ x_1\ is\ A_1^k\ and\ x_2\ is\ A_2^k\ THEN\ y^k\ is\ B^k\ \ k = 1,\ 2, \ldots,\ r \tag{42}$$

where A_1^k and A_2^k are the fuzzy sets of the k^{th} antecedent pair and B^k is the fuzzy set of the k^{th} consequent. Without loss of generality, singleton inputs are considered. Based on Mamdani method of inference, and for a set of disjunctive rules, the aggregated output for the r rules is given by:

$$\mu_{B^k}(y) = \max_k \left[min \left[\mu_{A_1^k}(input(i)), \mu_{A_2^k}(input(j)) \right] \right] \quad k = 1,\ 2, \ldots,\ r \tag{43}$$

The output of the Mamdani inference method is a fuzzy set. In order for the FLC output to be interpreted by the controlled system, the fuzzy set has to be converted to a crisp value. This process of converting a fuzzy value to a crisp value is known as defuzzification. There are different defuzzification methods [69]. The three commonly used defuzzification techniques are: Mean of Maximum (MOM) method, Center of Gravity (COG) method and Height method (HM) [64].

The COG method (also called centroid method or center of area) is the most widely applied defuzzification technique. It is equivalent to the formula for calculating the center of gravity in physics [69]. In this paper COG method is employed. Let the set shown in Fig. 10 be a fuzzy value representing the output of the Mamdani inference method. The crisp value (z^*) obtained using the COG method, is computed by expression (44).

$$z^* = \frac{\int \mu_C(z) \cdot z \ dz}{\int \mu_C(z) \ dz} \tag{44}$$

where $\int$refers to an algebraic integration. Fig. 11 shows the structure of the FLC described above and in Fig. 12, the model used to control the DC motor by the FLC is illustrated.

6. Simulation results and discussion

The results presented in this section were obtained using MATLAB/Simulink R2019a software, installed in a personal computer equipped with an Intel® Core™ i3-2328M 2.20GHz processor and 4.00 GB of RAM. Fuzzy Logic Designer, which is one of the applications contained in MATLAB, was used for fuzzy logic controller implementation. The data and strategies used in each algorithm can be found in Tables 2, 3, 4, 5, 6. Table 2 contains the data that are common to all algorithms. A table containing specific parameters for the TLBO algorithm is not presented since it does not require other parameters besides the ones that are common to all the algorithms.

The implemented FLC consists of two inputs (the error and error rate of change) and one defuzzified output. The linguistic values of the inputs and output are partitioned into five (5) fuzzy subsets, which are represented by five (5) membership functions, viz. Negative Large (NL), Small Large (SN), Zero (ZE), Positive Small (PS) and Positive Large (PL). The membership functions employed for the error (e), the error rate of change (de) and for the output (u) of the FLC are depicted in Figs. 13, 14, and 15, respectively.

Table 7 represents the relationship between the two fuzzy antecedents (the error and its derivative) and one fuzzy consequent (the control signal) used to construct the rule base of the controller.

6.1. Analysis of statistical results

As noticed, the initial population of the algorithms is created randomly and depending on the positions of its members within the search space, it can happen that the initial population is very far or very close to the globally optimal solution. That is, while an initial population positioned close to the best solution facilitates the exploration and exploitation processes, an initial population positioned far from the best solution can negatively impact the performance of the algorithm. Therefore, to allow for a fair comparison between algorithms, it is customary to give more than one chance of independent runs for each of them. The applied algorithms (EO, PSO, TLBO, DE and GA) were independently run 20 times. Table 8 below shows the statistical indicators of ITAE obtained for each algorithm. In Fig. 16, the best ITAE values per run are graphed and in Fig. 17, the box plot is given.

From the results shown in Table 8, it can be concluded that all the algorithms applied here propose approximately the same global minimum. It can also be concluded that DE and GA have more dispersed data than EO, PSO and TLBO. From Fig. 16, it can be seen that GA and DE are more sensitive to runs. While EO, PSO, and TLBO are practically insensitive, managing to often converge to the global minimum, regardless of the run number. Fig. 17 summarizes the results shown in Table 8; the height and the presence of outliers in the GA box, reveal that its data are more dispersed in relation to those presented by the other algorithms.

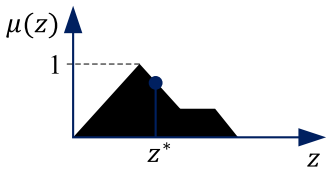


Fig. 10. COG defuzzification method.

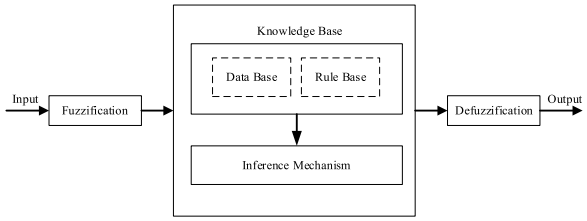


Fig. 11. Structure of a Fuzzy Logic Controller.

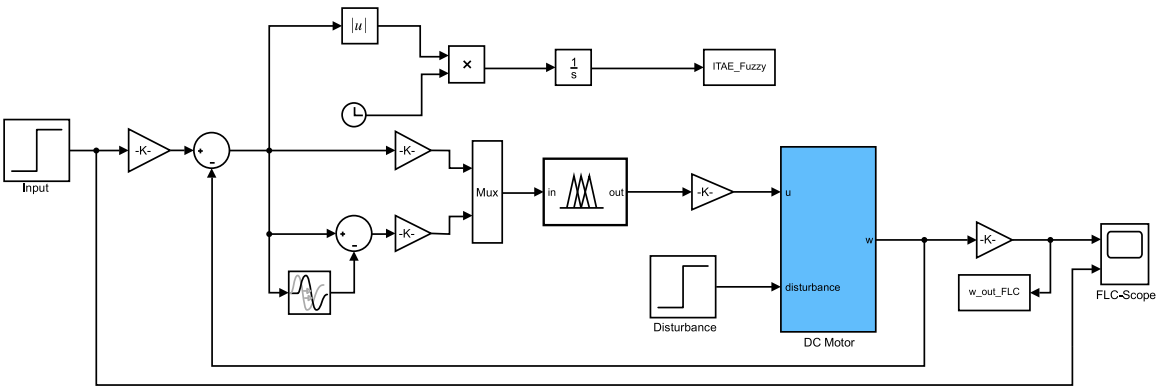


Fig. 12. FLC-controlled DC motor Simulink model.

Table 2
Common parameters.

Parameter	Value
Simulation time	1 s
Dimension of the problem	3
Population size	40
Independent runs	20
Number of iterations	50
Lower bound $[K_P; K_I; K_D]$	$[0.001; 0.001; 0.001]$
Upper bound $[K_P; K_I; K_D]$	$[20; 20; 20]$

Table 3
Parameters of EO algorithm.

Parameter	Value
Exploration constant (a_1)	2
Exploitation constant (a_2)	1
Generation probability (GP)	0.5

Table 4
Parameters of PSO algorithm.

Parameter	Value
Personal acceleration coefficient (c_1)	1.4962
Social acceleration coefficient (c_2)	1.4962
Inertia coefficient (w)	0.7298

6.2. Convergence curve of applied algorithms

To plot the convergence curves of the applied algorithms, the best runs were considered (refer to Fig. 16). From Fig. 18, it can be inferred that the lowest convergence speeds belong to GA and DE. This is because GA and DE need 50 iterations to converge to their best results. On the other hand, TLBO presents the highest convergence speed. TLBO manages to converge to its final result from

Table 5
Parameters of DE algorithm.

Parameter	Value
Lower limit of scaling factor (F_{min})	0.2
Upper limit of scaling factor (F_{max})	0.8
Crossover probability (P_c)	0.2
DE variant type	DE/rand/1/bin
Selection strategy	Greedy

Table 6
Parameters of RCGA algorithm.

Parameter	Value
Crossover probability (P_c)	0.8
Mutation probability (P_m)	0.2
Distribution index for crossover (η_C)	100
Distribution index for mutation (η_m)	100
Parent selection	Binary tournament
Crossover strategy	SBX
Mutation strategy	Polynomial
Survival strategy	$\mu + \lambda$

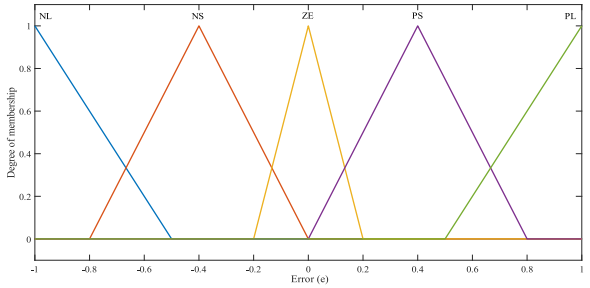


Fig. 13. Membership functions defined for error (e).

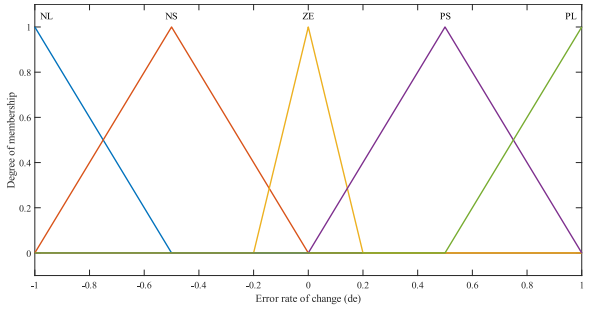


Fig. 14. Membership functions defined for derivative of error (de).

iteration 23 onwards. In Table 9, the controller gains obtained by each algorithm at the end of 50 iterations are presented.

6.3. Time-domain response analyses for a unit step input

Using the results presented in Table 9 and Eq. (41) presented in Section 5.1, time-domain analyses of controllers for a step input were carried out. Fig. 19 presents the motor speed responses for a step input. Figs. 20-22 show the rise time (10 % to 90 % of the steady-state), settling time (±2 % tolerance) and maximum percentage overshoot, respectively, of the responses shown in Fig. 19. Analyzing Figs. 19—22, it can be stated that since the applied algorithms present approximate gain values, the responses in the time-domain are also equivalent.

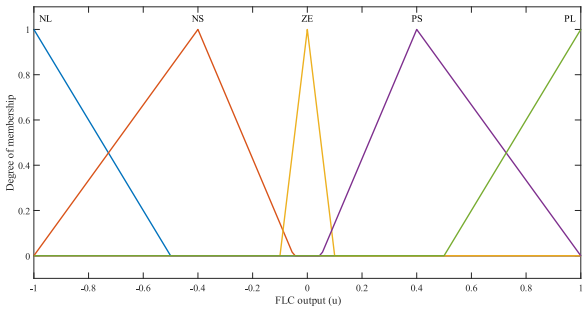


Fig. 15. Membership functions defined for FLC output (u).

Table 7
Matrix of rules used for FLC.

e *de*	NL	NS	ZE	PS	PL
NL	NL	NL	NL	NS	ZE
NS	NL	NL	NS	ZE	PS
ZE	NL	NS	ZE	PS	PL
PS	NS	ZE	PS	PL	PL
PL	ZE	PS	PL	PL	PL

Table 8
Values of statistical indicators of ITAE objective function.

Statistical indicator	EO	PSO	TLBO	DE	GA
Maximum	0.000415	0.000414	0.000413	0.000452	0.002107
Minimum	0.000413	0.000413	0.000413	0.000414	0.000419
Mean	0.000414	0.000414	0.000413	0.000428	0.000844
Median	0.000413	0.000413	0.000413	0.000423	0.000610
Standard deviation	5.659e-07	2.256e-07	2.188e-10	1.296e-05	0.000524

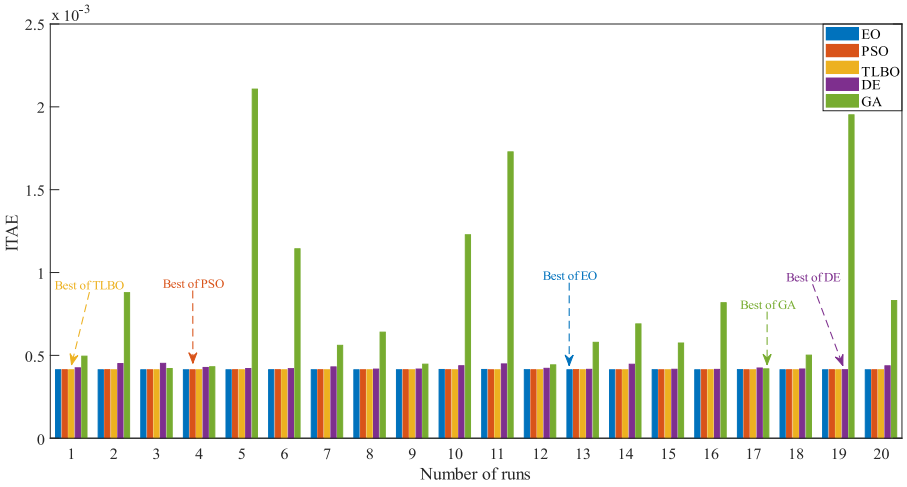


Fig. 16. Best ITAE values obtained in each run.

6.4. Frequency-domain analyses for a unit step input

In this subsection, analyzes in the frequency-domain are presented. In Fig. 23, the Bode plots of the controllers are illustrated. In Table 10, bandwidth, gain and phase margins are given. Again, for the same reason previously invoked, the frequency-domain responses of the controllers proposed by the applied algorithms are approximately equivalent.

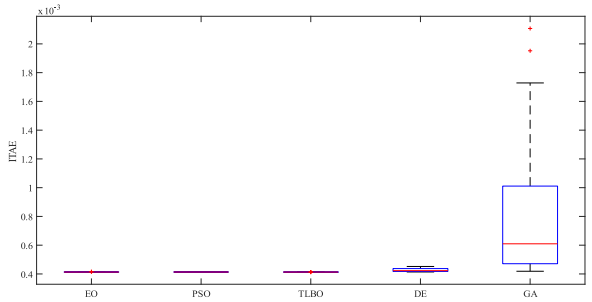


Fig. 17. Boxplot of the ITAE values for each algorithm.

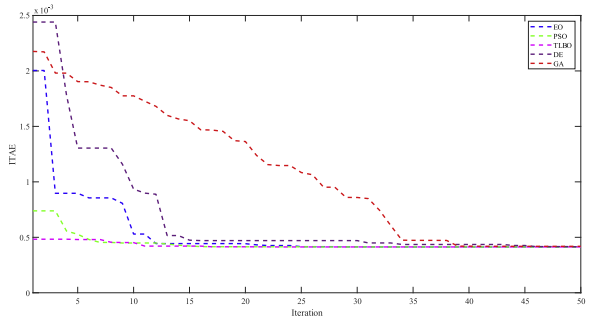


Fig. 18. Convergence curves.

Table 9
Gains obtained by each algorithm.

Controller	K_P	K_I	K_D
EO-PID	20	5.344	3.541
PSO-PID	20	5.344	3.541
TLBO-PID	20	5.344	3.541
DE-PID	20	5.342	3.540
GA-PID	20	5.353	3.547

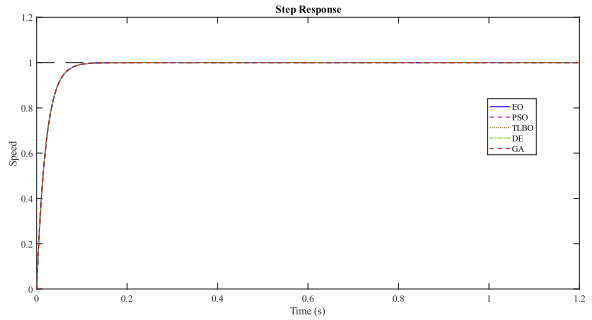


Fig. 19. DC motor speed comparison.

6.5. Comparison of the controllers output signal

To measure the strength of the control signal, a first-order filter was applied to the derivative component of the PID. Therefore, the non-ideal PID expression is given as:

$$G_{PID}^{filter}(s) = K_P + \frac{K_I}{s} + \frac{K_D s}{T_f s + 1} \tag{45}$$

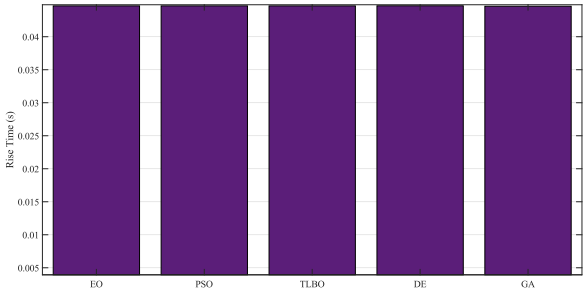


Fig. 20. Rise times for each controller.

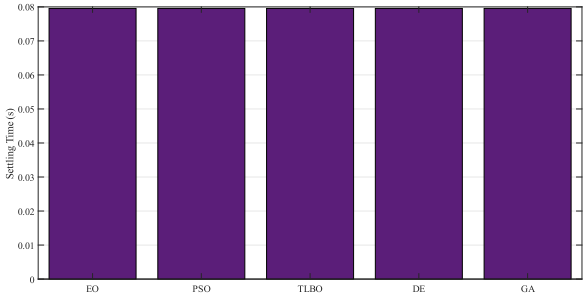


Fig. 21. Settling times for each controller.

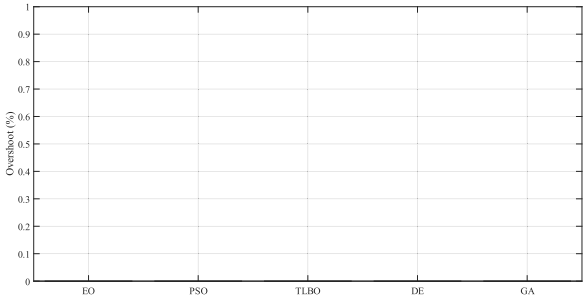


Fig. 22. Overshoot for each controller.

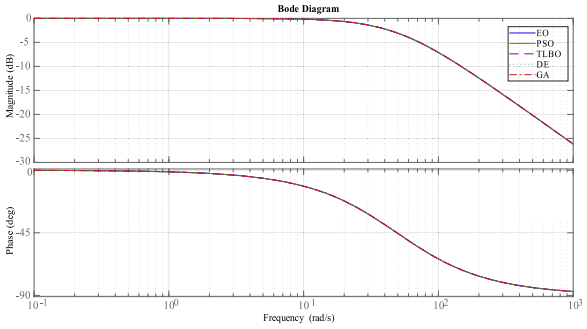


Fig. 23. Bode plots for each controller.

Table 10
Frequency-domain performance results.

Controller	Bandwidth (Hz)	Gain margin (dB)	Phase margin (deg.)
EO-PID	49.064	∞	180
PSO-PID	49.063	∞	180
TLBO-PID	49.064	∞	180
DE-PID	49.047	∞	180
GA-PID	49.137	∞	180

Table 11
Control signal strength measurements.

Controller	U_{max}	Energy
EO-PID	3561	6343.875
PSO-PID	3560.9	6343.520
TLBO-PID	3561	6343.875
DE-PID	3559.6	6338.905
GA-PID	**3567**	**6365.198**

where T_f is the time constant of the filter. T_f is set to be 0.001. Measurements of the control signals of the applied controllers are presented in the Table 11 below. The results show that GA-PID approach presents relatively high control effort.

6.6. Performance comparison with fuzzy logic controller

In this subsection, the controllers previously presented are compared with the FLC. Fig. 24 shows controller speed responses to a step input. Table 12 contains the rise time, settling time and percentage overshoot values for each controller. As clearly seen, FLC presents the best rise and settling times with a negligible overshoot percentage when compared to other controllers.

6.7. Performances under disturbance

To assess the performance of controllers when submitted to disturbances, a load disturbance corresponding to 0.01 Nm is applied. Figs. 25a and 25b show the responses obtained when the disturbance is applied at $t = 0$ s and $t = 0.6$ s, respectively. As noticed, FLC has a higher load disturbance suppression capacity, lower undershoot and good transient characteristics when compared to other controllers.

6.8. Robustness analysis of controllers

Controller robustness is an important aspect and needs to be evaluated in order to determine the ability of the controller to maintain the system within acceptable operational ranges when exposed to uncertainties. In this paper, to assess the robustness of the presented controllers, the electrical resistance (R_a) and torque constant (K_t) of the DC motor were separately varied ±25 % and ±20 %, respectively, thus creating four different cases or operating points as presented in Table 13.

The comparative results obtained for the transient response analysis of the DC motor speed corresponding to the 4 cases are presented in Tables 14, 15, 16, 17. Likewise, the comparative speed step response curves are illustrated in Figs. 26, 27, 28, and 29. Despite variations in DC motor parameters, the FLC manages to maintain approximately the same performance as if it were in a normal operation. In all 4 cases, FLC has the lowest rise and settling times. In case 1, FLC has the lowest percentage of overshoot and presents some negligible overshoot percentages in the remaining cases.

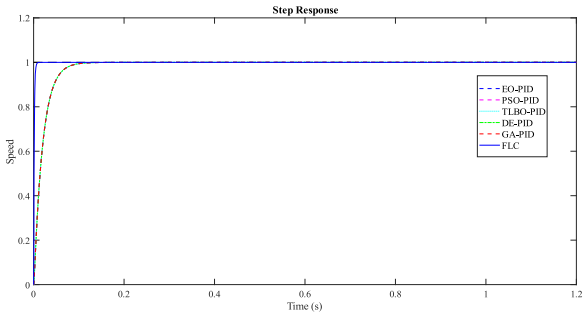


Fig. 24. Performance comparison with FLC.

Table 12
Comparison of transient responses.

Controller	$T_r(s)$	$T_s(s)$	$M_P(\%)$
EO-PID	0.0447	0.0795	**0.0000**
PSO-PID	0.0447	0.0795	**0.0000**
TLBO-PID	0.0447	0.0795	**0.0000**
DE-PID	0.0447	0.0795	**0.0000**
GA-PID	0.0446	0.0796	**0.0000**
FLC	**0.0026**	**0.0053**	0.0040

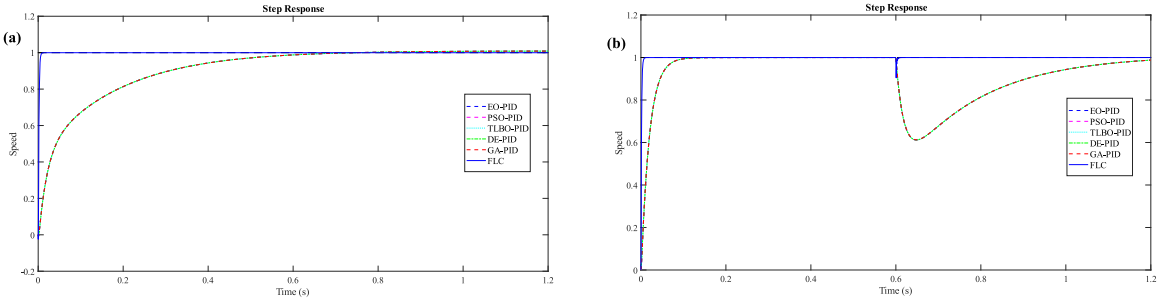


Fig. 25. **(a)** Load disturbance at $t = 0\ s$; **(b)** Load disturbance at $t = 0.6\ s$.

Table 13
Parameters of EO algorithm.

Motor parameter	Case 1	Case 2	Case 3	Case 4
K_t	0.012	0.012	0.018	0.018
R_a	0.30	0.50	0.30	0.50

Table 14
Transient characteristics for case 1.

Controller	$T_r(s)$	$T_s(s)$	$M_P(\%)$
EO-PID	0.0535	0.0967	0.0144
PSO-PID	0.0535	0.0967	0.0144
TLBO-PID	0.0535	0.0967	0.0144
DE-PID	0.0535	0.0967	0.0147
GA-PID	0.0534	0.0968	0.0134
FLC	**0.0026**	**0.0051**	**0.0054**

Table 15
Transient characteristics for case 2.

Controller	$T_r(s)$	$T_s(s)$	$M_P(\%)$
EO-PID	0.0535	0.0971	**0.0000**
PSO-PID	0.0535	0.0971	**0.0000**
TLBO-PID	0.0535	0.0971	**0.0000**
DE-PID	0.0535	0.0971	**0.0000**
GA-PID	0.0534	0.0972	**0.0000**
FLC	**0.0026**	**0.0051**	0.0040

6.9. Comparison with the results obtained by other similar research papers

In this final subsection, the results provided in this paper are compared with the results presented in [24], which is a recently published paper and demonstrated results that surpassed other previously available works on the same topic. The authors proposed Henry Gas Solubility Optimization with Opposition-Based Learning based PID (OBL/HGO-PID) as a new technique for DC motor control. The results obtained in that paper were compared not only with its original version (HGSO-PID), but also with Atom Search Optimization based PID (ASO-PID), grey Wolf Optimization based PID (GWO-PID), Stochastic Fractal Search based PID (SFS-PID) and

Table 16
Transient characteristics for case 3.

Controller	$T_r(s)$	$T_s(s)$	$M_P(\%)$
EO-PID	0.0348	0.0629	0.0024
PSO-PID	0.0348	0.0629	0.0024
TLBO-PID	0.0348	0.0629	0.0024
DE-PID	0.0348	0.0629	0.0031
GA-PID	0.0347	0.0628	**0.0015**
FLC	**0.0025**	**0.0052**	0.0042

Table 17
Transient characteristics for case 4.

Controller	$T_r(s)$	$T_s(s)$	$M_P(\%)$
EO-PID	0.0348	0.0630	**0.0000**
PSO-PID	0.0348	0.0630	**0.0000**
TLBO-PID	0.0348	0.0630	**0.0000**
DE-PID	0.0348	0.0630	**0.0000**
GA-PID	0.0347	0.0630	**0.0000**
FLC	**0.0025**	**0.0053**	0.0040

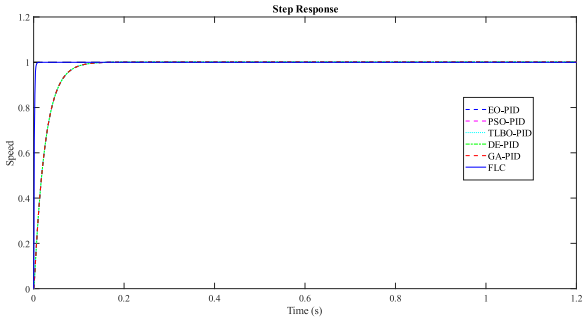


Fig. 26. Speed step responses under case 1 conditions.

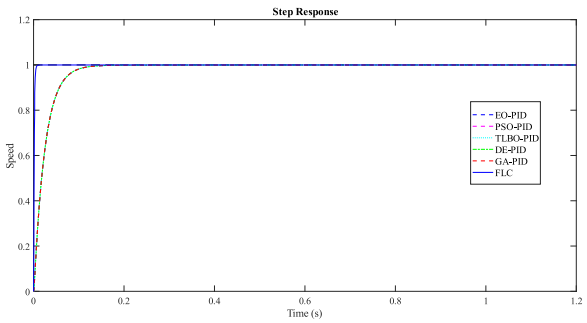


Fig. 27. Speed step responses under case 2 conditions.

Sine-Cosine Algorithm based PID (SCA-PID). From the comparisons with the other approaches, the authors demonstrated that the results obtained using OBL/HGO-PID were better. The results presented by the aforementioned research paper are presented in Tables 18—22 and in Figs. 30,31a and 31b. Where in Table 18, the proposed PID gains are given, Table 19 contains the transient characteristics, Table 20 contains the results of the frequency-domain analysis, in Tables 21—22, the results of the time-domain analysis for the 4 cases considered during the robustness analysis are presented. In Figs. 30,31a and 31b, the comparative speed step response curves under normal conditions, under 0.01 Nm load disturbance at $t = 0$ s and under 0.01 Nm load disturbance at $t = 6$ s, respectively, are plotted.

Comparing the results presented in Table 19 with those presented in Table 12, the results presented in Table 20 with those presented in Table 10, the results presented in Tables 21-22 with those presented in Tables 14-17, it can be clearly derived that both the fuzzy logic controller, as well as the PID controllers based on the algorithms considered in this paper, present relatively better results

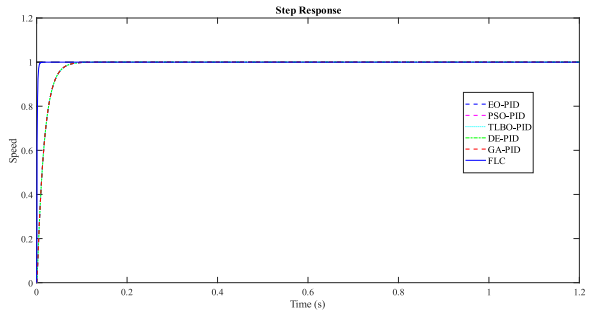


Fig. 28. Speed step responses under case 3 conditions.

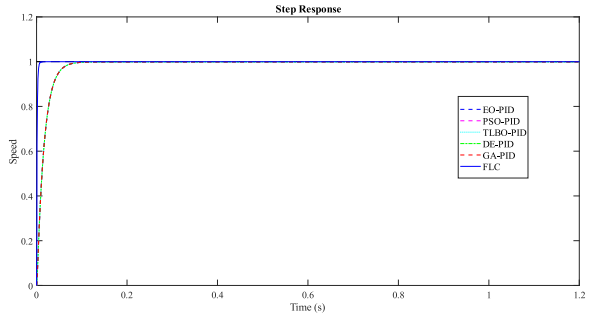


Fig. 29. Speed step responses under case 4 conditions.

Table 18
Gains of the controllers used for comparison.

Controller	K_P	K_I	K_D
HGSO-PID	13.4430	1.2059	2.2707
OBL/HGSO-PID	16.9327	0.9508	2.8512

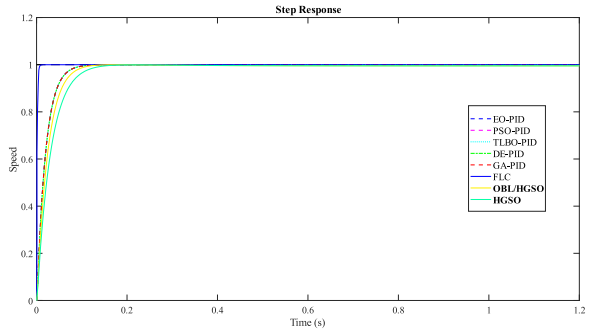


Fig. 30. Comparison with HGSO-PID and OBL/HGSO-PID.

than HGSO-PID and OBL/HGSO-PID [24]. Consequently, it can be deduced that the results presented in this paper are also better than ASO-PID [12], GWO-PID [42], SFS-PID [70] and SCA-PID [71]. As for the robustness analyses, the results found in the present paper are superior in all the other cases, except for cases 2 and 4, where the FLC presents some negligible percentages of overshoot. From Figs. 30,31a and 31b, the numerical results provided and the statements announced here can be graphically observed.

7. Conclusions

In this paper, performance analyses of optimized PIDs for DC motor speed control were conducted. The PIDs were based on five optimization algorithms: EO, PSO, TLBO, DE, and GA. These metaheuristics were selected because they belong to distinct classes in

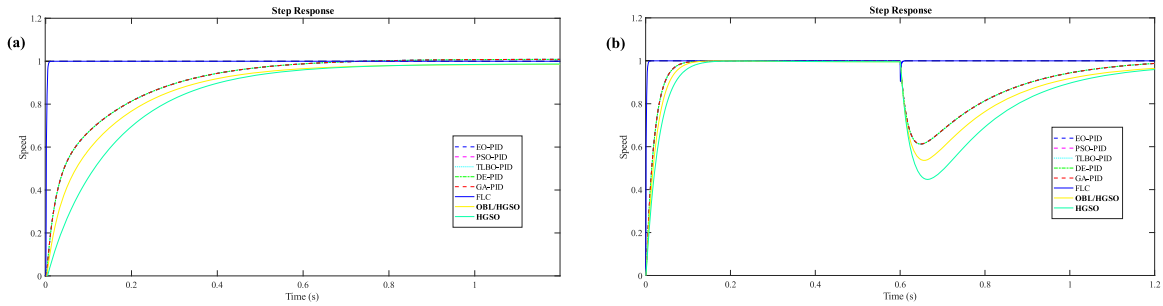


Fig. 31. **(a)** Step responses under load disturbance at $t = 0\ s$; **(b)** Step responses under load disturbance at $t = 0.6\ s$.

Table 19
Transient characteristics of the controllers used for comparison.

Controller	$T_r(s)$	$T_s(s)$	$M_P(\%)$
HGSO-PID	0.0684	0.1186	0.0000
OBL/HGSO-PID	0.0546	0.0946	0.0000

Table 20
Frequency-domain characteristics of the controllers used for comparison.

Controller	Bandwidth (Hz)	Gain margin (dB)	Phase margin (deg.)
HGSO-PID	31.7975	∞	180
OBL/HGSO-PID	39.8561	∞	180

Table 21
Robustness analysis for HGSO-PID.

Cases	$T_r(s)$	$T_s(s)$	$M_P(\%)$
1	0.0849	0.1455	0.0127
2	0.0856	0.1499	0.0000
3	0.0570	0.0982	0.0659
4	0.0573	0.1002	0.0000

Table 22
Robustness analysis for OBL/HGSO-PID.

Cases	$T_r(s)$	$T_s(s)$	$M_P(\%)$
1	0.0678	0.1163	0.0560
2	0.0682	0.1191	0.0000
3	0.0455	0.0785	0.0959
4	0.0457	0.0798	0.0000

addition to being well established in their respective groups. To carry out this research, the mathematical model of the DC motor was deduced and represented through transfer functions. Detailed descriptions and mathematical expressions of the previously mentioned metaheuristics were presented. ITAE was used as the fitness function. Since the first solutions in population-based metaheuristics are randomly generated, to allow a fair comparison, all algorithms underwent 20 independent runs. The results obtained were subjected to statistical analysis using indicators such as mean, median, minimum, maximum, and standard deviation. The time and frequency domain response characteristics (rise time, settling time, maximum percentage overshoot, bandwidth, gain margin, and phase margin) of the DC motor were also computed. The controllers were subjected to different tests, including analyses of control signal strength, performance under load disturbance, and robustness (varying $\pm$ 25 % the electrical resistance and $\pm$ 20 % the torque constant). Additionally, a comparison was also made between the optimized PIDs and FLC. Furthermore, the results were also compared with the OBL/HGO algorithm, which was reported to be superior to some previous studies on the same topic. Analyzing the statistical results, it can be concluded that due to their stochastic nature, it may be that in a given run one algorithm obtains a better performance than the other, but when a sufficient number of runs is considered, the five metaheuristics, despite being in different runs, encountered approximate gains. In terms of convergence, TLBO exhibited the highest convergence speed, whereas GA and DE were slower to achieve optimal results. It was also verified that GA and DE presented more dispersion in the ITAE values than the others. Based on the

analyses in the time and frequency domains, along with load disturbance and robustness tests, it was verified that the five optimized controllers displayed the same results and outperformed the OBL/HGO. However, the designed FLC was superior compared to the metaheuristic-based PIDs. Future work may consider including other emerging AI-based optimization techniques, such as machine learning and deep learning, among others. A class of optimization algorithms called game-based metaheuristics has been reported recently; it would be similarly pertinent to consider this category in forthcoming studies. Please note that all results presented here were based on simulations performed in the MATLAB/Simulink environment.

Funding statement

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

CRediT authorship contribution statement

Nelson Luis Manuel: Conceptualization, Methodology, Software, Formal analysis, Investigation, Writing – original draft, Writing – review & editing, Validation. **Nihat İnanç:** Formal analysis, Writing – review & editing, Supervision, Project administration, Validation. **Murat Lüy:** Methodology, Investigation, Writing – original draft, Supervision, Project administration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All data generated or analysed during this research are explicitly included in the text of this paper. There are no additional external datasets. Further inquiries can be forwarded to the corresponding author.

Acknowledgments

The authors are especially indebted for the kind remarks and error corrections provided by the anonymous reviewers, which contributed to improving the overall quality of this paper.

References

[1] Ma'arif A, Setiawan NR. Control of DC motor using integral state feedback and comparison with PID: simulation and Arduino implementation. J Robot Control 2021;2(5):456–61. https://doi.org/10.18196/jrc.25122.
[2] Huang C, Lei F, Han X, Zhang Z. Determination of modeling parameters for a brushless DC motor that satisfies the power performance of an electric vehicle. Meas Control 2019;52(7–8):765–74. https://doi.org/10.1177/0020294019842607.
[3] Yun J, et al. Self-adjusting force/bit blending control based on quantitative factor-scale factor fuzzy-PID bit control. Alex Eng J 2022;61(6):4389–97. https://doi.org/10.1016/j.aej.2021.09.067. Jun.
[4] Dursun EH, Levent ML, Durdu A, Aydoğdu Ö. Speed control of a variable loaded DC motor by using sliding mode and iterative learning control. Int J Electr Energy 2017;5(1):22–8. https://doi.org/10.18178/ijoee.5.1.22-28.
[5] Prathibanandhi K, Ramesh R. Hybrid control technique for minimizing the torque ripple of brushless direct current motor. Meas Control 2018;51(7–8):321–35. https://doi.org/10.1177/0020294018786753.
[6] Munadi MAA, Naniwa T, Taniai Y. Model reference adaptive control for DC motor based on simulink. In: 2016 6th International Annual Engineering Seminar (InAES); 2016. p. 101–6. https://doi.org/10.1109/INAES.2016.7821915. Aug.
[7] Chaouch S, et al. DC-motor control using Arduino-Uno board for wire-feed system. In: 2018 international conference on electrical sciences and technologies in Maghreb (CISTEM); 2018. p. 1–6. https://doi.org/10.1109/CISTEM.2018.8613492. Oct.
[8] Adel Z, Hamou AA, Abdellatif S. Design of real-time PID tracking controller using Arduino Mega 2560for a permanent magnet DC motor under real disturbances. In: 2018 international conference on electrical sciences and technologies in Maghreb (CISTEM); 2018. p. 1–5. https://doi.org/10.1109/CISTEM.2018.8613560. Oct.
[9] Gasparesc G. PID control of a DC motor using Labview interface for embedded platforms. In: 2016 12th IEEE international symposium on electronics and telecommunications (ISETC); 2016. p. 145–8. https://doi.org/10.1109/ISETC.2016.7781078. Oct.
[10] Guo Y, Mohamed MEA. Speed control of direct current motor using ANFIS based Hybrid P-I-D configuration controller. IEEE Access 2020;8:125638–47. https://doi.org/10.1109/ACCESS.2020.3007615.
[11] Ahmad M, Khan A, Raza MA, Ullah S. A study of state feedback controllers for pole placement. In: 2018 5th international multi-topic ICT conference (IMTIC); 2018. p. 1–6. https://doi.org/10.1109/IMTIC.2018.8467276. Apr.
[12] Hekimoglu B. Optimal tuning of fractional order PID controller for DC motor speed control via chaotic atom search optimization algorithm. IEEE Access 2019;7:38100–14. https://doi.org/10.1109/ACCESS.2019.2905961.
[13] Somwanshi D, Bundele M, Kumar G, Parashar G. Comparison of Fuzzy-PID and PID controller for speed control of DC motor using LabVIEW. Procedia Comput Sci 2019;152:252–60. https://doi.org/10.1016/j.procs.2019.05.019.
[14] Cheon K, Kim J, Hamadache M, Lee D. On replacing PID controller with deep learning controller for DC motor system. J Autom Control Eng 2015;3(6):452–6. https://doi.org/10.12720/joace.3.6.452-456.
[15] Sardhalia M, Baru S, Gupta S, Shukla A. Comparative performance study of different controllers for speed regulation of DC motor. In: 2022 IEEE 10th power india international conference (PIICON); 2022. p. 1–6. https://doi.org/10.1109/PIICON56320.2022.10045232. Nov.
[16] Chen W, Lan W, Li X. Research on a control method of DC speed regulating electric energy vehicle based on neural network. In: 2022 3rd international conference on computer vision, image and deep learning & international conference on computer engineering and applications (CVIDL & ICCEA); 2022. p. 80–5. https://doi.org/10.1109/CVIDLICCEA56201.2022.9824655. May.

[17] Ekinci S, Hekimoglu B. Improved kidney-inspired algorithm approach for tuning of PID controller in AVR system. IEEE Access 2019;7:39935–47. https://doi.org/10.1109/ACCESS.2019.2906980.
[18] Peng F, Wang Y, Xuan H, Nguyen TVT. Efficient road traffic anti-collision warning system based on fuzzy nonlinear programming. Int J Syst Assur Eng Manag 2022;13(S1):456–61. https://doi.org/10.1007/s13198-021-01468-2. Mar.
[19] Rahayu ES, Ma'arif A, Çakan A. Particle Swarm Optimization (PSO) tuning of PID control on DC motor. Int J Robot Control Syst 2022;2(2):435–47. https://doi.org/10.31763/ijrcs.v2i2.476. Jul.
[20] Ismaeel AAK, Elshaarawy IA, Houssein EH, Ismail FH, Hassanien AE. Enhanced elephant herding optimization for global optimization. IEEE Access 2019;7(c):34738–52. https://doi.org/10.1109/ACCESS.2019.2904679.
[21] Hashim FA, Houssein EH, Mabrouk MS, Al-Atabany W, Mirjalili S. Henry gas solubility optimization: a novel physics-based algorithm. Future Gener Comput Syst 2019;101:646–67. https://doi.org/10.1016/j.future.2019.07.015. Dec.
[22] Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 2016;96:120–33. https://doi.org/10.1016/j.knosys.2015.12.022. Mar.
[23] Faramarzi A, Heidarinejad M, Stephens B, Mirjalili S. Equilibrium optimizer: a novel optimization algorithm. Knowl Based Syst 2020;191:105190. https://doi.org/10.1016/j.knosys.2019.105190.
[24] Ekinci S, Hekimoğlu B, Izci D. Opposition based Henry gas solubility optimization as a novel algorithm for PID control of DC motor. Eng Sci Technol Int J 2021;24(2):331–42. https://doi.org/10.1016/j.jestch.2020.08.011.
[25] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: a gravitational search algorithm. Inf Sci 2009;179(13):2232–48. https://doi.org/10.1016/j.ins.2009.03.004. Jun.
[26] Patel VK, Savsani VJ. Heat transfer search (HTS): a novel optimization algorithm. Inf Sci 2015;324:217–46. https://doi.org/10.1016/j.ins.2015.06.044. Dec.
[27] Alatas B. ACROA: artificial chemical reaction optimization algorithm for global optimization. Expert Syst Appl 2011;38(10):13170–80. https://doi.org/10.1016/j.eswa.2011.04.126. Sep.
[28] Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM. Salp Swarm Algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 2017;114:163–91. https://doi.org/10.1016/j.advengsoft.2017.07.002. Dec.
[29] El-Gammal AAA, El-Samahy AA. A modified design of PID controller for DC motor drives using particle swarm optimization PSO. In: POWERENG 2009 - 2nd international conference on power engineering, energy and electrical drives proceeding; 2009. p. 419–24. https://doi.org/10.1109/POWERENG.2009.4915157.
[30] Yavuz G, Durmuş B, Aydın D. Artificial Bee Colony Algorithm with distant savants for constrained optimization. Appl Soft Comput 2022;116:108343. https://doi.org/10.1016/j.asoc.2021.108343. Feb.
[31] Dorigo M, Blum C. Ant colony optimization theory: a survey. Theor Comput Sci 2005;344(2–3):243–78. https://doi.org/10.1016/j.tcs.2005.05.020. Nov.
[32] Li J, Wei X, Li B, Zeng Z. A survey on firefly algorithms. Neurocomputing 2022;500:662–78. https://doi.org/10.1016/j.neucom.2022.05.100. Aug.
[33] Dehghani M, Trojovská E, Trojovský P. A new human-based metaheuristic algorithm for solving optimization problems on the base of simulation of driving training process. Sci Rep 2022;12(1):9924. https://doi.org/10.1038/s41598-022-14225-7. Jun.
[34] Rao RV, Savsani VJ, Vakharia DP. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. CAD Comput Aided Des 2011;43(3):303–15. https://doi.org/10.1016/j.cad.2010.12.015.
[35] Samareh Moosavi SH, Bardsiri VK. Poor and rich optimization algorithm: a new human-based and multi populations algorithm. Eng Appl Artif Intell 2019;86:165–81. https://doi.org/10.1016/j.engappai.2019.08.025. November 2018Nov.
[36] Mousavirad SJ, Ebrahimpour-Komleh H. Human mental search: a new population-based metaheuristic optimization algorithm. Appl Intell 2017;47(3):850–87. https://doi.org/10.1007/s10489-017-0903-6. Oct.
[37] Dehghani M, et al. A new "doctor and patient" optimization algorithm: an application to energy commitment problem. Appl Sci 2020;10(17):5791. https://doi.org/10.3390/app10175791. Aug.
[38] Ibrahim MA, Mahmood AK, Sultan NS. Optimal PID controller of a brushless DC motor using genetic algorithm. Int J Power Electron Drive Syst 2019;10(2):822–30. https://doi.org/10.11591/ijpeds.v10.i2.822-830.
[39] Storn R, Price K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 1997;11:341–59. https://doi.org/10.1023/A:1008202821328.
[40] Adam SP, Alexandropoulos SAN, Pardalos PM, Vrahatis MN. No free lunch theorem: A Review. In: Demetriou I, Pardalos P, editors. Approximation and optimization. Optimization and its applications, 145. Cham: Springer; 2019. p. 57–82. https://doi.org/10.1007/978-3-030-12767-1_5.
[41] Xia CL. Permanent magnet brushless DC motor drives and controls. Singapore: John Wiley & Sons Singapore Pte. Ltd.; 2012.
[42] Agarwal J, Parmar G, Gupta R, Sikander A. Analysis of grey wolf optimizer based fractional order PID controller in speed control of DC motor. Microsyst Technol 2018;24(12):4997–5006. https://doi.org/10.1007/s00542-018-3920-4.
[43] Khanam I, Parmar G. Application of SFS algorithm in control of DC motor and comparative analysis. In: 2017 4th IEEE Uttar Pradesh section international conference on electrical, computer and electronics UPCON 2017; 2017. p. 256–61. https://doi.org/10.1109/UPCON.2017.8251057. 2018-Janua.
[44] Ekinci S, Hekimoglu B, Demiroren A, Eker E. Speed control of DC motor using improved sine cosine algorithm based PID controller. In: 3rd international symposium on multidisciplinary studies and innovative technologies ISMSIT 2019 - Proceeding; 2019. https://doi.org/10.1109/ISMSIT.2019.8932907. December.
[45] Abdi H. Profit-based unit commitment problem: a review of models, methods, challenges, and future directions. Renew Sustain Energy Rev 2021;138:110504. https://doi.org/10.1016/j.rser.2020.110504. March 2020.
[46] Gaing ZL. A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE Trans Energy Convers 2004;19(2):384–91. https://doi.org/10.1109/TEC.2003.821821. Jun.
[47] Achanta RK, Pamula VK. DC motor speed control using PID controller tuned by Jaya optimization algorithm. In: IEEE international conference on power, control, signals and instrumentation engineering ICPCSI 2017; 2018. p. 983–7. https://doi.org/10.1109/ICPCSI.2017.8391856.
[48] Gupta S, Deep K, Mirjalili S. An efficient equilibrium optimizer with mutation strategy for numerical optimization. Appl Soft Comput J 2020;96. https://doi.org/10.1016/j.asoc.2020.106542. July.
[49] Wang D, Tan D, Liu L. Particle swarm optimization algorithm: an overview. Soft Comput 2018;22(2):387–408. https://doi.org/10.1007/s00500-016-2474-6.
[50] Ding Y, Chen L, Hao K. Bio-inspired optimization algorithms. In: Bio-inspired collaborative intelligent control and optimization118. Singapore: Springer; 2018. p. 317–91. https://doi.org/10.1007/978-981-10-6689-4_8.
[51] Pattanaik JK, Basu M, Dash DP. Improved real coded genetic algorithm for dynamic economic dispatch. J Electr Syst Inf Technol 2018;5(3):349–62. https://doi.org/10.1016/j.jesit.2018.03.002.
[52] Deb K, Agrawal RB. Simulated binary crossover for continuous search space. Complex Syst 1994;9:1–34.
[53] Razali NM, Geraghty J. Genetic algorithm performance with different selection strategies in solving TSP. In: Proceedings of world congress on engineering, WCE 2011. 2; 2011. p. 1134–9.
[54] Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithms. Foundations of genetic algorithms. 1. Amsterdam, The Netherlands: Elsevier; 1991. p. 69–93. https://doi.org/10.1016/B978-0-08-050684-5.50008-2.
[55] Rivera G, Cisneros L, Sánchez-Solís P, Rangel-Valdez N, Rodas-Osollo J. Genetic algorithm for scheduling optimization considering heterogeneous containers: a real-world case study. Axioms 2020;9(1). https://doi.org/10.3390/axioms9010027.
[56] Chuang YC, Chen CT, Hwang C. A real-coded genetic algorithm with a direction-based crossover operator. Inf Sci 2015;305(1):320–48. https://doi.org/10.1016/j.ins.2015.01.026. Jun.
[57] Albayrak M, Allahverdi N. Development a new mutation operator to solve the traveling salesman problem by aid of genetic algorithms. Expert Syst Appl 2011;38(3):1313–20. https://doi.org/10.1016/j.eswa.2010.07.006.
[58] Deighan DS, Field SE, Capano CD, Khanna G. Genetic-algorithm-optimized neural networks for gravitational wave classification. Neural Comput Appl 2021. https://doi.org/10.1007/s00521-021-06024-4.

[59] Lu Y, Zhou J, Qin H, Wang Y, Zhang Y. Chaotic differential evolution methods for dynamic economic dispatch with valve-point effects. Eng Appl Artif Intell 2011;24(2):378–87. https://doi.org/10.1016/j.engappai.2010.10.014. Mar.
[60] dos Santos Coelho L. Tuning of PID controller for an automatic regulator voltage system using chaotic optimization approach. Chaos Solitons Fractals 2009;39 (4):1504–14. https://doi.org/10.1016/j.chaos.2007.06.018.
[61] Åström KJ, Hägglund T. PID controllers: theory, design, and tuning, 2. Research Triangle Park, North Carolina: Instrument Society of America; 1995.
[62] Yu GR, Hwang RC. Optimal PID speed control of brush less DC motors using LQR approach. In: Conference proceeding of the IEEE international conference on systems, man and cybernetics. 1; 2004. p. 473–8. https://doi.org/10.1109/ICSMC.2004.1398343.
[63] Zadeh LA. Fuzzy sets. Inf Control 1965;8:338–53.
[64] Bai Y, Wang D. Fundamentals of fuzzy logic control —fuzzy sets, fuzzy rules and defuzzifications. Advanced fuzzy logic technologies in industrial applications. London: Springer London; 2006. p. 17–36.
[65] Sousa GCD, Bose BK. A fuzzy set theory based control of a phase-controlled converter DC machine drive. In: Conference record of the 1991 IEEE industry applications society annual meeting. 30; 1994. p. 854–61. https://doi.org/10.1109/IAS.1991.178338.
[66] Lee CC. Fuzzy logic in control systems: fuzzy logic controller. I. IEEE Trans Syst Man Cybern 1990;20(2):419–35. https://doi.org/10.1109/21.52552.
[67] Jiang W. The application of the fuzzy theory in the design of intelligent building control of water tank. J Softw 2011;6(6):1082–8. https://doi.org/10.4304/jsw.6.6.1082-1088. Jun.
[68] Neethu U, Jisha VR. Speed control of Brushless DC motor: a comparative study. In: PEDES 2012 - IEEE international conference on power electronics, drives and energy systems; 2012. https://doi.org/10.1109/PEDES.2012.6484349.
[69] Ross TJ. Fuzzy logic with engineering applications. John Wiley & Sons; 2004.
[70] Bhatt R, Parmar G, Gupta R, Sikander A. Application of stochastic fractal search in approximation and control of LTI systems. Microsyst Technol 2019;25(1): 105–14. https://doi.org/10.1007/s00542-018-3939-6.
[71] Agarwal J, Parmar G, Gupta R. Application of sine cosine algorithm in optimal control of DC motor and robustness analysis. Wulfenia J 2017;24(11):77–95.