

# Advanced Metaheuristics-Based Approach for Fuzzy Control Systems Tuning

Soufiene Bouallègue, Fatma Toumi, Joseph Haggège  
and Patrick Siarry

**Abstract** In this study, a new advanced metaheuristics-based optimization approach is proposed and successfully applied to design and tuning of a PID-type Fuzzy Logic Controller (FLC). The scaling factors tuning problem of the FLC structure is formulated and systematically resolved, using various constrained metaheuristics such as the Differential Search Algorithm (DSA), Gravitational Search Algorithm (GSA), Artificial Bee Colony (ABC) and Particle Swarm Optimization (PSO). In order to specify more time-domain performance control objectives of the proposed metaheuristics-tuned PID-type FLC, different optimization criteria such as Integral of Square Error (ISE) and Maximum Overshoot (MO) are considered and compared. The classical Genetic Algorithm Optimization (GAO) method is also used as a reference tool to measure the statistical performances of the proposed methods. All these algorithms are implemented and analyzed in order to show the superiority and the effectiveness of the proposed fuzzy control tuning approach. Simulation and real-time experimental results, for an electrical DC drive benchmark, show the advantages of the proposed metaheuristics-tuned PID-type fuzzy control structure in terms of performance and robustness.

---

S. Bouallègue · F. Toumi (✉) · J. Haggège  
Research Laboratory in Automatic Control LA.R.A, National Engineering School  
of Tunis (ENIT), BP 37, Le Belvdre, 1002 Tunis, Tunisia  
e-mail: fatima.toumi@enit.mu.tn

S. Bouallègue  
e-mail: soufiene.bouallegue@issig.mu.tn

J. Haggège  
e-mail: joseph.haggege@enit.mu.tn

F. Toumi · P. Siarry  
Signals, Images and Intelligent Systems Laboratory, LiSSi-EA-3956, University Paris-Est  
Créteil Val de Marne, 61 Avenue du Général de Gaulle, 94010 Créteil, France  
e-mail: siarry@univ-paris12.fr

## 1 Introduction

Fuzzy logic control approach has been widely used in many successful industrial applications. This control strategy, with the Mamdani fuzzy type inference, demonstrated high robustness and effectiveness properties (Azar 2010a, b, 2012; Lee 1998a, b; Passino and Yurkovich 1998). The known PID-type FLC structure, firstly proposed in Qiao and Mizumoto (1996), is especially established and improved within the practical framework (Eker and Torun 2006; Guzelkaya et al. 2003; Woo et al. 2000). This particular fuzzy controller retains the characteristics similar to the conventional PID controller and can be decomposed into the equivalent proportional, integral and derivative control components (Eker and Torun 2006; Qiao and Mizumoto 1996). In this design case, the dynamic behaviour depends on the adequate choice of the fuzzy controller scaling factors. The tuning procedure depends on the control experience and knowledge of the human operator, and it is generally achieved based on a classical trials-errors procedure. There is not up to now a systematic method to guide such a choice. This tuning problem becomes more delicate and hard as the complexity of the control plant increases.

In order to improve further the performance of the transient and steady state responses of the PID-type fuzzy structure, various strategies and methods are proposed to tune their parameters. In Qiao and Mizumoto (1996), proposed a peak observer mechanism-based method to adjust the PID-type FLC parameters. This self-tuning mechanism decreases the equivalent integral control component of the fuzzy controller gradually with the system response process time. On the other hand, Woo et al. (2000) developed a method based on two empirical functions evolved with the system's error information. In Guzelkaya et al. (2003), the authors proposed a technique that adjusts the scaling factors, corresponding to the derivative and integral components, using a fuzzy inference mechanism. However, the major drawback of all these PID-type FLC tuning methods is the difficult choice of their scaling factors and self-tuning mechanisms. The time-domain dynamics of the fuzzy controller depends strongly on this hard choice. The tuning procedure depends on the control experience and knowledge of the human operator, and it is generally achieved based on a classical trials-errors procedure. Hence, having a systematic approach to tune these scaling factors is interesting and the optimization theory may present a promising solution.

In solving this kind of optimization problems, the classical exact optimization algorithms, such as gradient and descent methods, do not provide a suitable solution and are not practical. The relative objective functions are non linear, non analytical and non convex (Bouallègue et al. 2012a, b). Over the last decades, there has been a growing interest in advanced metaheuristic algorithms inspired by the behaviours of natural phenomena (Boussaid et al. 2013; Dréo et al. 2006; Rao and Savsani 2012; Siarry and Michalewicz 2008). It is shown by many researchers that these algorithms are well suited to solve complex computational problems in wide and various ranges of engineering applications summarized around domains of robotics, image and signal processing, electronic circuits design, communication networks,

but more especially the domain of process control design (Bouallègue et al. 2011, 2012a, b; David et al. 2013; Goswami and Chakraborty 2014; Madiouni et al. 2013; Toumi et al. 2014).

Various metaheuristics have been adopted by researchers. The Differential Search Algorithm (DSA) (Civicioglu 2012), Gravitational Search Algorithm (GSA) (Rashedi et al. 2009), Artificial Bee Colony (ABC) (Karaboga 2005) and Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995; Kennedy and Eberhart 1995) algorithms are the most recent proposed techniques in the literature. They will be adapted and improved for the considered fuzzy control design problem. Without any regularity on the cost function to be optimized, the recourse to these stochastic and global optimization techniques is justified by the empirical evidence of their superiority in solving a variety of non-linear, non-convex and non-smooth problems. In comparison with the conventional optimization algorithms, these optimization techniques are a simple concept, easy to implement, and computationally efficient algorithms. Their stochastic behaviour allows overcoming the local minima problem.

In this study, a new approach based on the use of advanced metaheuristics, such as DSA, GSA, ABC and PSO is proposed for systematically tuning the scaling factors of the particular PID-type FLC structure. The well known classical GAO algorithm is used in order to compare the obtained optimization results (Goldberg 1989; MathWorks 2009). This work can be considered as a contribution on the results given in Bouallègue et al. (2012a, b), Toumi et al. (2014). The synthesis and tuning of the fuzzy controller are formulated as a constrained optimization problem which is efficiently solved based on the proposed metaheuristics. In order to specify more robustness and performance control objectives of the proposed metaheuristics-tuned PID-type FLC, different optimization criteria such as ISE and MO are considered and compared.

The remainder of this chapter is organized as follows. In Sect. 2, the studied PID-type FLC structure is presented and formulated as a constrained optimization problem. An external static penalty technique is investigated to handle the problem constraints. The advanced DSA, GSA, ABC and PSO metaheuristic algorithms, used in solving the formulated problem, are described in Sect. 3. Section 4 is dedicated to apply the proposed fuzzy control approach on an electrical DC drive benchmark. All obtained simulation results are compared with each other and analysed. Experimental setup and results are presented within a real-time framework.

## 2 PID-Type FLC Tuning Problem Formulation

In this section, the PID-type fuzzy controller synthesis problem is formulated as a constrained optimization problem which will be resolved through the suggested metaheuristics algorithms.

### 2.1 A Review of PID-Type Fuzzy Control Structure

The particular PID-type fuzzy controller structure, originally proposed by Qiao and Mizumoto within the continuous-time formalism (Qiao and Mizumoto 1996), retains the characteristics similar to the conventional PID controller. This result remains valid while using a particular structure of FLC with triangular uniformly distributed membership functions for the fuzzy inputs and a crisp output, the product-sum inference and the center of gravity defuzzification methods (Bouallègue et al. 2012a, b; Eker and Torun 2006; Guzelkaya et al. 2003; Haggège et al. 2010; Toumi et al. 2014; Woo et al. 2000).

Under these conditions, the equivalent proportional, integral and derivative control components of such a PID-type FLC are given by  $\alpha K_e \Pi + \beta K_d \Delta$ ,  $\beta K_e \Pi$  and  $\alpha K_d \Delta$ , respectively, as shown in Qiao and Mizumoto (1996). In these expressions,  $\Pi$  and  $\Delta$  represent relative coefficients,  $K_e$ ,  $K_d$ ,  $\alpha$  and  $\beta$  denote the scaling factors associated to the inputs and output of the fuzzy controller. When approximating the integral and derivative terms within the discrete-time framework (Bouallègue et al. 2012a, b; Haggège et al. 2010; Toumi et al. 2014), we can consider the closed-loop control structure for a digital PID-type FLC, as shown in Fig. 1. The dynamic behaviour of this PID-type FLC structure is strongly depending on the scaling factors, difficult and delicate to tune.

As shown in Fig. 1, this particular structure of Mamdani fuzzy controller uses two inputs: the error  $e_k$  and the variation of error  $\Delta e_k$ , to provide the output  $u_k$  that describes the discrete fuzzy control law.

### 2.2 Optimization-Based Problem Formulation

The choice of the adequate values for the scaling factors of the described PID-type FLC structure is often done by a trials-errors hard procedure. This tuning problem becomes difficult and delicate without a systematic design method. To deal with these difficulties, the optimization of these control parameters is proposed like a

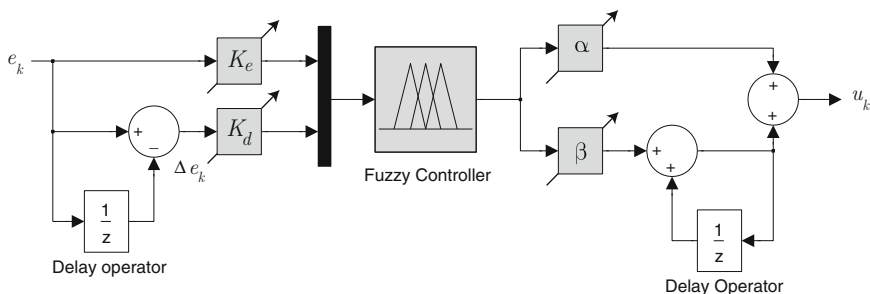


Fig. 1 Proposed discrete-time PID-type FLC structure

promising procedure. This tuning can be formulated as the following constrained optimization problem:

$$\begin{cases} \text{minimize} & f(x) \\ x=(K_e, K_d, \alpha, \beta)^T \in \mathcal{S} \subset \mathbb{R}_+^4 \\ \text{subject to :} & \\ g_1(x) = \delta - \delta^{\max} \leq 0 & \\ g_2(x) = t_s - t_s^{\max} \leq 0 & \\ g_3(x) = E_{ss} - E_{ss}^{\max} \leq 0 & \end{cases} \quad (1)$$

where  $f: \mathbb{R}^4 \rightarrow \mathbb{R}$  the cost function,  $\mathcal{S} = \{x \in \mathbb{R}_+^4, x_{low} \leq x \leq x_{up}\}$  the initial search space, which is supposed containing the desired design parameters, and  $g_l: \mathbb{R}^4 \rightarrow \mathbb{R}$  the nonlinear problem's constraints.

The optimization-based tuning problem (1) consists in finding the optimal decision variables, representing the scaling factors of a given PID-type FLC structure, which minimizes the defined cost function, chosen as the Maximum Overshoot (MO) and the Integral of Square Error (ISE) performance criteria. These cost functions are minimized, using the proposed particular constrained metaheuristics, under various time-domain control constraints such as overshoot  $\delta$ , steady state error  $E_{ss}$ , rise time  $t_r$  and settling time  $t_s$  of the system's step response, as shown in Eq. (1). Their specified maximum values constrain the step response of the tuned PID-type fuzzy controlled system, and can define some time-domain templates.

### 2.3 Proposed Constraints Handling Method

The considered metaheuristics in this study are originally formulated as an unconstrained optimizer. Several techniques have been proposed to deal with constraints. One useful approach is by augmenting the cost function of problem (1) with penalties proportional to the degree of constraint infeasibility. This approach leads to convert the constrained optimization problem into the unconstrained optimization problem. In this paper, the following external static penalty technique is used:

$$\varphi(x) = f(x) + \sum_{l=1}^{n_{con}} \lambda_l \max [0, g_l(x)^2] \quad (2)$$

where  $\lambda_l$  is a prescribed scaling penalty parameter, and  $n_{con}$  is the number of problem constraints  $g_l(x)$ .

### 3 Solving Optimization Problem Using Advanced Algorithms

In this section, the basic concepts as well as the algorithm steps of each proposed advanced metaheuristic are described for solving the formulated PID-type FLC tuning problem.

#### 3.1 Differential Search Algorithm

The Differential Search Algorithm (DSA) is a recent population-based metaheuristic optimization algorithm invented in 2012 by Civicioglu (2012). This global and stochastic algorithm simulates the Brownian-like random-walk movement used by an organism to migrate (Civicioglu 2012; Goswami and Chakraborty 2014; Waghole and Tiwari 2014).

Migration behavior allows the living beings to move from a habitat where capacity and diversity of natural sources are limited to a more efficient habitat. In the migration movement, the migrating species of living beings constitute a superorganism containing large number of individuals. Then it starts to change its position by moving toward more fruitful areas using a Brownian-like random-walk model. The population made up of random solutions of the respective problem corresponds to an artificial-superorganism migrating. The artificial superorganism migrates to global minimum value of the problem. During this migration, the artificial-superorganism tests whether some randomly selected positions are suitable temporarily. If such a position tested is suitable to stop over for a temporary time during the migration, the members of the artificial-superorganism that made such discovery immediately settle at the discovered position and continue their migration from this position (Civicioglu 2012).

In DSA metaheuristic, a superorganism  $X_k^i$  of  $N$  artificial-organisms making up, at every generation  $k = 1, 2, \dots, k_{\max}$ , an artificial-organism with members as much as the size of the problem, defined as follows:

$$X_k^i = \left( x_k^{i,1}, x_k^{i,2}, \dots, x_k^{i,d}, \dots, x_k^{i,D} \right) \quad (3)$$

A member of an artificial-organism, in initial position, is randomly defined by using Eq. (4):

$$x_0^{i,d} = x_{low}^{i,d} + rand(0, 1) \left( x_{up}^{i,d} - x_{low}^{i,d} \right) \quad (4)$$

In DSA, the mechanism of finding a so called Stopover Site, which presents the solution of optimization problem, at the areas remaining between the artificial-organisms, is described by a Brownian-like random walk model. The principle is

based on the move of randomly selected individuals toward the targets of a Donor artificial-organism, denoted as (Civicioglu 2012):

$$X_k^{Donor} = X_k^{random\_shuffling(i)} \quad (5)$$

where the index  $i$  of artificial-organisms is produced by the Shuffling-random function.

The size of the change occurred in the positions of the members of the artificial-organisms is controlled by the Scale factor given as follows:

$$s_k^i = randG\{2rand_1\}(rand_2 - rand_3) \quad (6)$$

where  $rand_1$ ,  $rand_2$  and  $rand_3$  are uniformly distributed random numbers in the interval  $[0, 1]$ ,  $randG$  is a Gamma-random number.

The Stopover Site positions, which are very important for a successful migration, are produced by using Eq. (7):

$$Y_k^i = X_k^i + s_k^i(X_k^{Donor} - X_k^i) \quad (7)$$

So, the individuals of the artificial-organisms of the superorganism to participate in the search process of Stopover Site are determined by a random process based on the manipulation of two control parameters  $p_1$  and  $p_2$ . The algorithm is not much sensitive to these control parameters and the values in the interval  $[0, 0.3]$  usually provide best solutions for a given problem (Civicioglu 2012).

Finally, the steps of the original version of DSA, as described by the pseudo code in Civicioglu (2012), can be summarized as follows:

1. Search space characterization: size of superorganism, dimension of problem, random numbers  $p_1$  and  $p_2$ , ...
2. Randomized generation of the initial population.
3. Fitness evaluation of artificial-organisms.
4. Calculation of the Stopover Site positions in different directions.
5. Randomly select individuals to participate in the search process of Stopover Site.
6. Update the Stopover Site positions and evaluate the new population.
7. Update the superorganism by the new Stopover site positions.
8. Repeat steps 3–7 until the stop criteria are reached.

### 3.2 Gravitational Search Algorithm

The Gravitational Search Algorithm (GSA) is population-based metaheuristic optimization algorithm introduced in 2009 by Rashedi et al. (2009). This algorithm is based on the law of gravity and mass interactions as described in Nobahari et al. (2011),

Precup et al. (2011), Rashedi et al. (2009). The search agents are a set of masses which interact with each other based on the Newtonian gravity and the law of motion.

Several applications of this algorithm in various areas of engineering are investigated (Nobahari et al. 2011; Precup et al. 2011; Rao and Savsani 2012). In GSA, the particles, called also agents, are considered as bodies and their performance is measured by their masses. All these bodies attract each other by the gravity force that causes a global movement of all objects towards the objects with heavier masses. These agents correspond to the optimum solutions in the search space (Rashedi et al. 2009). Indeed, each agent presents a solution of optimization problem and is characterised by its position, inertial mass, active and passive gravitational masses. The GSA is navigated by properly adjusting the gravitational and inertia masses leading masses to be attracted by the heaviest object.

The position of the mass corresponds to a solution of the problem, and its gravitational and inertial masses are determined using a cost function. The exploitation capability of this algorithm is guaranteed by the movement of the heavy masses, more slowly than the lighter ones.

Let us consider a population with  $N$  agents. The position of the  $i$ th agent at iteration time  $k$  is defined as:

$$X_k^i = \left( x_k^{i,1}, x_k^{i,2}, \dots, x_k^{i,d}, \dots, x_k^{i,D} \right) \quad (8)$$

where  $x_k^{i,d}$  presents the position of the  $i$ th particle in the  $d$ th dimension of search space of size  $D$ .

At a specific time “ $t$ ”, denoted by the actual iteration “ $k$ ”, the force acting on mass “ $i$ ” from mass “ $j$ ” is given as follows:

$$F_k^{ij,d} = G_k \frac{M_k^{pi} \times M_k^{aj}}{R_k^{ij} + \varepsilon} \left( x_k^{j,d} - x_k^{i,d} \right) \quad (9)$$

where  $M_k^{aj}$  is the active gravitational mass related to agent  $j$ ,  $M_k^{pi}$  is the passive gravitational mass related to agent  $i$ ,  $G_k$  is the gravitational constant at time  $k$ ,  $\varepsilon$  is a small constant, and  $R_k^{ij}$  is the Euclidian distance between two agents  $i$  and  $j$ , defined as:

$$R_k^{ij} = \left\| X_k^i, X_k^j \right\|_2 \quad (10)$$

To give a stochastic characteristic to this algorithm, authors of GSA suppose that the total force that acts on agent  $i$  is a randomly weighted sum of  $j$ th components of the forces exerted from other bodies, given as follows (Rashedi et al. 2009):

$$F_k^{i,d} = \sum_{j=1, j \neq i}^N \text{rand}^j F_k^{ij,d} \quad (11)$$

where  $\text{rand}^j$  is a random number in the interval  $[0, 1]$ .



By the law of motion, the acceleration of the agent  $i$  at time  $k$ , and in  $d$ th direction, is given as follows:

$$a_k^{i,d} = \frac{F_k^{i,d}}{M_k^{ii}} \tag{12}$$

where  $M_k^{ii}$  is the inertial mass of  $i$ th agent in the search space with dimension  $d$ .

Hence, the position and the velocity of an agent are updated respectively by the mean of equations of movement given as follows:

$$x_{k+1}^{i,d} = x_k^{i,d} + v_{k+1}^{i,d} \tag{13}$$

$$v_{k+1}^{i,d} = rand^i v_k^{i,d} + a_k^{i,d} \tag{14}$$

where  $rand^i$  is a uniform random number in the interval  $[0, 1]$ , used to give a randomized characteristic to the search.

To control the search accuracy, the gravitational constant  $G_k$ , is initialized at the beginning and will be reduced with time. In this study, we use an exponentially decreasing of this algorithm parameter, as follows:

$$G_k = G_0 e^{-\eta \frac{k}{k_{max}}} \tag{15}$$

where  $G_0$  is the initial value of  $G_k$ ,  $\eta$  is a control parameter to set, and  $k_{max}$  is the total number of iterations.

In GSA, gravitational and inertia masses are calculated by the fitness evaluation. A heavier mass means a more efficient agent. Better agents have higher attractions and walk more slowly.

As given in Rashedi et al. (2009), the values of masses are calculated using the fitness function and gravitational and inertial masses are updated by the following equations:

$$M_k^{ai} = M_k^{pi} = M_k^{ii} = M_k^i \tag{16}$$

$$M_k^i = \frac{m_k^i}{\sum_{j=1}^N m_k^j} \tag{17}$$

$$m_k^i = \frac{fit_k^i - worst_k}{best_k - worst_k} \tag{18}$$

where  $fit_k^i$  represents the fitness value of the agent  $i$  at iteration  $k$ , and,  $worst_k$  and  $best_k$  are defined, for a minimization problem, as follows:

$$best_k = \min_{1 \leq j \leq N} fit_k^j \quad (19)$$

$$worst_k = \max_{1 \leq j \leq N} fit_k^j \quad (20)$$

To perform a good compromise between exploration and exploitation, authors of GSA choose to reduce the number of agents with lapse of iterations in Eq. (11), which will be modified as:

$$F_k^{i,d} = \sum_{j \in Kbest, j \neq i} rand^j F_k^{ij,d} \quad (21)$$

where  $Kbest$  is the set of the first  $K$  agents with best fitness and biggest mass that will attract the others.

The algorithm parameter  $Kbest$  is a function of iterations with the initial value  $K_0$ , usually set to the total size of population  $N$  at the beginning, and linearly decreasing with time. At the end of search, there will be just one agent applying force to the others.

Finally, the steps of the original version of GSA, as described in Rashedi et al. (2009), can be summarized as follows:

1. Search space characterization: number of agents, dimension of problem, control parameters  $G_0, K_0, \dots$
2. Randomized generation of the initial population.
3. Fitness evaluation of agents.
4. Update the algorithm parameters  $G_k, best_k, worst_k$  and  $M_k^i$  for each agent and at each iteration.
5. Calculation of the total force in different directions.
6. Calculation of acceleration and velocity.
7. Updating agents' position.
8. Repeat steps 3–7 until the stop criteria are reached.

### 3.3 Artificial Bee Colony

The Artificial Bee Colony (ABC) is a population-based metaheuristic optimization algorithm introduced in 2005 by Karaboga (2005). The principle of such an algorithm is based on the intelligent foraging behavior of honey bee swarm (Basturk and Karaboga 2006; Karaboga 2005; Karaboga and Akay 2009; Karaboga and Basturk 2007, 2008). The ABC algorithm has been enormously successful in various industrial domains and a wide range of engineering applications as summarized in Karaboga et al. (2012).

In this formalism, the population of the artificial bees' colony is constituted of three groups: employed bees, onlookers and scouts. Employed bees search the destination where food is available, translated by the amount of their nectar. They collect the food and return back to their origin, where they perform waggle dance depending on the amount of nectar's food available at the destination. The onlooker bee watches the dance and follows the employed bee depending on the probability of the available food.

In ABC algorithm, the population of bees is divided into two parts consisting of employed bees and onlooker bees. The sizes of each part are usually taken equal to. Employed bee, representing a potential solution in the search space with dimension, updates its new position by using the movement Eq. (22) and follows greedy selection to find the best solution. The objective function associated with the solution is measured by the amount of food.

Let us consider a population with  $N/2$  individuals in the search space. The position of the  $i$ th employer at iteration time  $k$  is defined as:

$$X_k^i = \left( x_k^{i,1}, x_k^{i,2}, \dots, x_k^{i,d}, \dots, x_k^{i,D} \right) \tag{22}$$

where  $D$  is the number of decision variables,  $i$  is the index on  $N/2$  employers.

In the  $d$ th dimension of search space, the new position of the  $i$ th employer, as well as of the  $i$ th onlooker, is updated by means of the movement equation given as follows:

$$x_{k+1}^{i,d} = x_k^{i,d} + r_k^i \left( x_k^{i,d} - x_k^{m,d} \right) \tag{23}$$

where  $r_k^i$  is a uniformly distributed random number in the interval  $[-1, 1]$ . It can be also chosen as a normally distributed random number with mean equal to zero and variance equal to one as given in Karaboga (2005). The employer's index  $m \neq i$  is a randomly number in the interval  $[1, N/2]$ .

Besides, an onlooker bee chooses a food source depending on the probability value of each solution associated with that food source, calculated as follows:

$$P_k^{i,d} = \frac{f_k^{i,d}}{\sum_n^{N/2} f_k^{n,d}} \tag{24}$$

where  $f_k^{i,d}$  is the fitness value of the  $i$ th solution at iteration  $k$ .

When the food source of an employed bee cannot be improved for some pre-determined number of cycles, called "Limit for abandonment" and denoted by  $L$ , the source food becomes abandoned and the employer behaves as a scout bee and it searches for the new food source using the following equation:

$$x_k^{i,d} = x_{low}^{i,d} + rand(0, 1) \left( x_{up}^{i,d} - x_{low}^{i,d} \right) \quad (25)$$

where  $x_{low}^{i,d}$  and  $x_{up}^{i,d}$  are the lower and upper ranges, respectively, for decision variables in the dimension.

This behaviour of the artificial bee colony reflects a powerful mechanism to escape the problem of trapping in local optima. The value of the “Limit for abandonment” control parameter of ABC algorithm is calculated as follows:

$$L = \frac{N}{2} \times D \quad (26)$$

Finally, the steps of the original version of ABC algorithm, as described in Basturk and Karaboga (2006), Karaboga (2005), Karaboga and Basturk (2007, 2008), can be summarized as follows:

1. Initialize the ABC algorithm parameters: population size  $N$ , limit of abandonment  $L$ , dimension of the search space  $D$ , ...
2. Generate a random population equal to the specified number of employed bees, where each of them contains the value of all the design variables.
3. Obtain the values of the objective function, defined as the amount of nectar for the food source, for all the population members.
4. Update the position of employed bees using Eq. (23), obtain the value of objective function and select the best solutions to replace the existing ones.
5. Run the onlooker bee phase: onlookers proportionally choose the employed bees depending on the amount of nectar found by the employed bees, Eq. (24).
6. Update the value of onlooker bees using Eq. (23) and replace the existing solution with the best new one.
7. Identify the abundant solutions using the limit value. If such solutions exist then these are transformed into the scout bees and the solution is updated using Eq. (25).
8. Repeat the steps 3–7 until the termination criterion is reached, usually chosen as the specified number of generations.

### 3.4 Particle Swarm Optimization

The PSO technique is an evolutionary computation method developed in 1995 by Kennedy and Eberhart (1995), Eberhart and Kennedy (1995). This recent meta-heuristic technique is inspired by the swarming or collaborative behaviour of biological populations. The cooperation and the exchange of information between population individuals allow solving various complex optimization problems. The convergence and parameters selection of the PSO algorithm are proved using several advanced theoretical analysis (Bouallègue et al. 2011, 2012a, b; Madiouni et al. 2013).

PSO has been enormously successful in several and various industrial domains and engineering fields (Bouallègue et al. 2012a; Dréo et al. 2006; Rao and Savsani 2012; Siarry and Michalewicz 2008).

The basic PSO algorithm uses a swarm consisting of  $N$  particles  $N_k^i$ , randomly distributed in the considered initial search space, to find an optimal solution  $x^* = \arg \min f(x) \in \mathbb{R}^D$  of a generic optimization problem. Each particle, that represents a potential solution, is characterised by its position and its velocity  $x_k^{i,d}$  and  $v_k^{i,d}$ , respectively.

At each iteration of the algorithm, and in the  $d$ th direction, the  $i$ th particle position evolves based on the following update rules:

$$x_{k+1}^{i,d} = x_k^{i,d} + v_{k+1}^{i,d} \quad (27)$$

$$v_{k+1}^{i,d} = w_{k+1} v_k^{i,d} + c_1 r_{1,k}^i (p_k^{i,d} - x_k^{i,d}) + c_2 r_{2,k}^i (p_k^{g,d} - x_k^{i,d}) \quad (28)$$

where  $w_{k+1}$  the inertia factor,  $c_1$  and  $c_2$  the cognitive and the social scaling factors respectively,  $r_{1,k}^i$  and  $r_{2,k}^i$  the random numbers uniformly distributed in the interval  $[0,1]$ ,  $p_k^{i,d}$  the best previously obtained position of the  $i$ th particle and  $p_k^{g,d}$  the best obtained position in the entire swarm at the current iteration  $k$ .

In order to improve the exploration and exploitation capacities of the proposed PSO algorithm, we choose for the inertia factor a linear evolution with respect to the algorithm iteration (Bouallègue et al. 2011, 2012a, b; Madiouni et al. 2013):

$$w_{k+1} = w_{\max} - \left( \frac{w_{\max} - w_{\min}}{k_{\max}} \right) k \quad (29)$$

where  $w_{\max} = 0.9$  and  $w_{\min} = 0.4$  represent the maximum and minimum inertia factor values, respectively.

Finally, the steps of the original version of PSO algorithm, as described in Eberhart and Kennedy (1995), Kennedy and Eberhart (1995), can be summarized as follows:

1. Define all PSO algorithm parameters such as swarm size  $N$ , maximum and minimum inertia factor values, cognitive and social coefficients, ...
2. Initialize the particles with random positions and velocities. Evaluate the initial population and determine  $p_0^{i,d}$  and  $p_0^{g,d}$ .
3. For each particle apply the update Eqs. (27)–(29).
4. Evaluate the corresponding fitness values and select the best solutions.
5. Repeat the steps 3–4 until the termination criterion is reached.

## 4 Case Study: PID-Type FLC Tuning for a DC Drive

This section is dedicated to apply the proposed metaheuristics-tuned PID-type FLC for the variable speed control of a DC drive. All the obtained simulations results are presented and discussed.

### 4.1 Plant Model Description

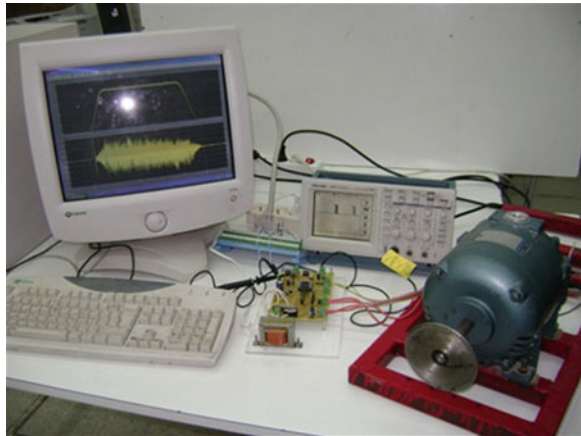
The considered benchmark is a 250 W electrical DC drive shown in Fig. 2. The machine's speed rotation is 3,000 rpm at 180 V DC armature voltage.

The motor is supplied by an AC-DC power converter. The considered electrical DC drive can be described by the following model (Haggège et al. 2009):

$$G(s) = \frac{A}{(1 + \tau_e s)(1 + \tau_m s)} \quad (30)$$

The model's parameters are obtained by an experimental identification procedure and they are summarized in Table 1 with their associated uncertainty bounds. This model is sampled with 10 ms sampling time for simulation and experimental setups.

**Fig. 2** Electrical DC drive benchmark



**Table 1** Identified DC Drive model parameters

Parameters	Nominal values	Uncertainty bounds (%)
$A$	0.05	50
$\tau_m$	300 ms	50
$\tau_e$	14 ms	50

### 4.2 Simulation Results

For this study case, product-sum inference and center of gravity defuzzification methods are adopted. Uniformly distributed and symmetrical membership functions, are assigned for the fuzzy input and output variables, as shown in Fig. 3.

The linguistic levels assigned to the input variables  $e_k$  and  $\Delta e_k$ , and the output variable  $\Delta u_k$  are given as follows: N (Negative), Z (Zero), P (Positive), NB (Negative Big) and PB (Positive Big). The associated fuzzy rule-base is given in Table 2. The view of this rule-base is illustrated in Fig. 4.

For our design, the initial search domain of PID-type FLC parameters is chosen in the limit range of  $x_{low} = (1, 5, 2, 25)$  and  $x_{up} = (5, 10, 10, 50)$ . For all proposed metaheuristics, we use a population size equal to  $N = 30$  and run all used algorithms under  $k_{max} = 100$  iterations. The size of optimization problem is equal to  $D = 4$ . The decision variables are the scaling factors of the studied particular PID-type FLC structure, i.e.,  $\alpha, \beta, K_e$  and  $K_d$ .

In this study, the control problem constraints are defined by the maximum values of the performance criteria: overshoot ( $\delta^{max} = 20\%$ ), settling time ( $t_s^{max} = 0.9$  s) and steady state error ( $E_{ss}^{max} = 0.0001$ ). The scaling penalty parameter is chosen as constant equal to  $\lambda_l = 10^4$ . The algorithm stops when the number of generations reaches the specified value for the maximum number of generations.

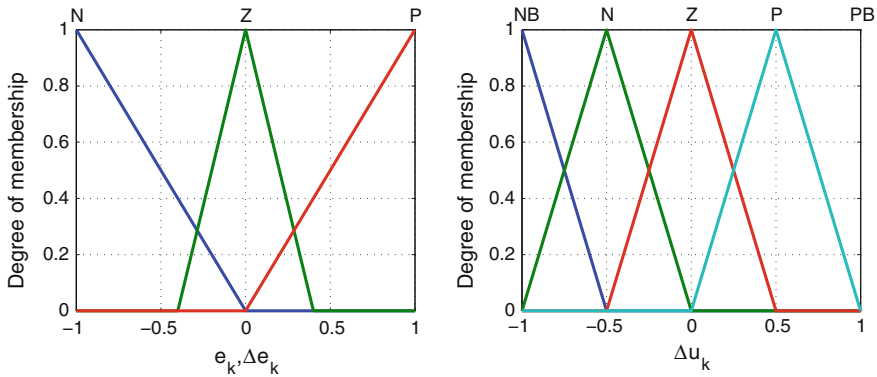
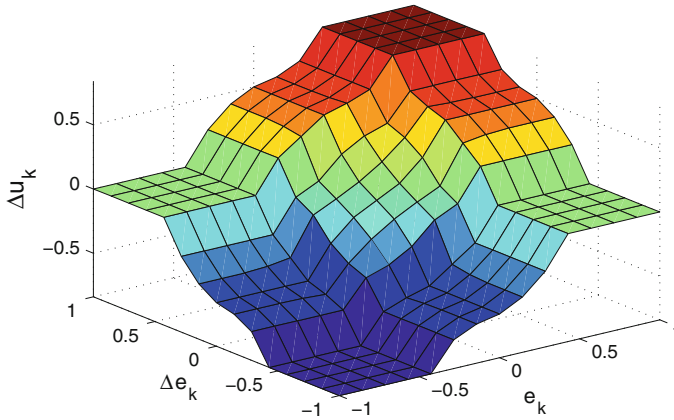


Fig. 3 Membership functions for fuzzy inputs and output variables

Table 2 Fuzzy rule-base for the standard FLC

$e_k, \Delta e_k$	N	Z	P
N	NB	N	Z
Z	N	Z	P
P	Z	P	PB



**Fig. 4** View of the fuzzy rule-base for the standard FLC

For the software implementation of the proposed metaheuristics, the control parameters of each algorithm are set as follows:

- DSA: random numbers Stopover site research  $p_1 = p_2 = 0.3rand(0, 1)$ ;
- GSA: initial value of gravitational constant  $G_0 = 75$ , parameter  $\eta = 20$ , initial value of the *Kbest* agents  $K_0 = N = 30$  which is decreased linearly to 1;
- ABC: Limit of abandonment  $L = 60$ ;
- PSO: cognitive and social coefficients equal to  $c_1 = c_2 = 2$ , inertia factor decreasing linearly from 0.9 to 0.4;
- GAO: Stochastic Uniform selection and Gaussian mutation methods, Elite Count equal to 2 and Crossover Fraction equal to 0.8.

In order to get statistical data on the quality of results and so to validate the proposed approaches, we run all implemented algorithms 20 times. Feasible solutions are usually found within an acceptable CPU computation time. The obtained optimization results are summarized in Tables 3 and 4.

### 4.3 Results Analysis and Discussion

According to the statistical analysis of Tables 3 and 4, as well as the numerical simulations in Figs. 5, 6, 7 and 8, we observe that the proposed approaches produce near results in comparison with each other and with the standard GAO-based method. Globally, the algorithms convergences always take place in the same region of the design space whatever is the initial population. This result indicates that the algorithms succeed in finding a region of the interesting research space to explore.

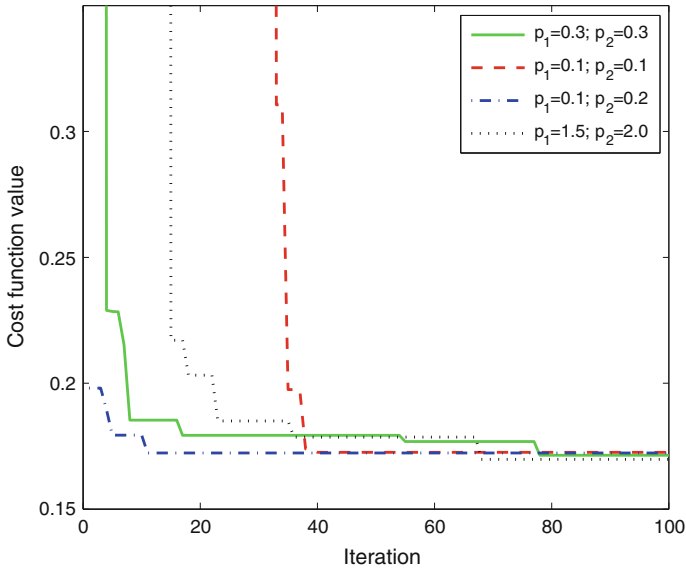


**Table 3** Optimization results from 20 trials of problem (1.1): ISE criterion

Algorithm	Best	Mean	Worst	ST deviation
DSA	0.1621	0.1691	0.1760	0.0045
GSA	0.1556	0.1710	0.1800	0.0193
ABC	0.1928	0.2274	0.2322	0.0134
PSO	0.1600	0.1715	0.1802	0.0140
GAO	0.1643	0.1722	0.1799	0.0086

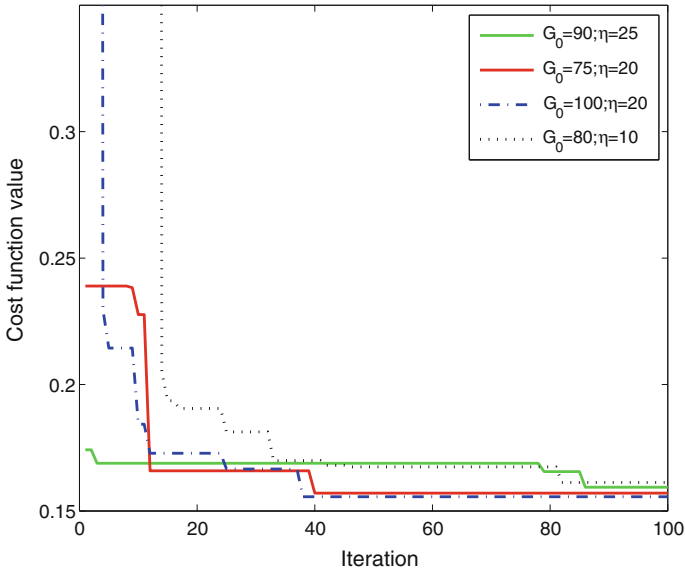
**Table 4** Optimization results from 20 trials of problem (1.1): MO criterion

Algorithm	Best	Mean	Worst	ST deviation
DSA	0.0365	0.0722	0.1307	0.0277
GSA	0.0307	0.0624	0.0096	0.0315
ABC	0.1305	0.1550	0.1972	0.0412
PSO	0.0422	0.0936	0.1420	0.0511
GAO	0.0411	0.0913	0.1300	0.0373

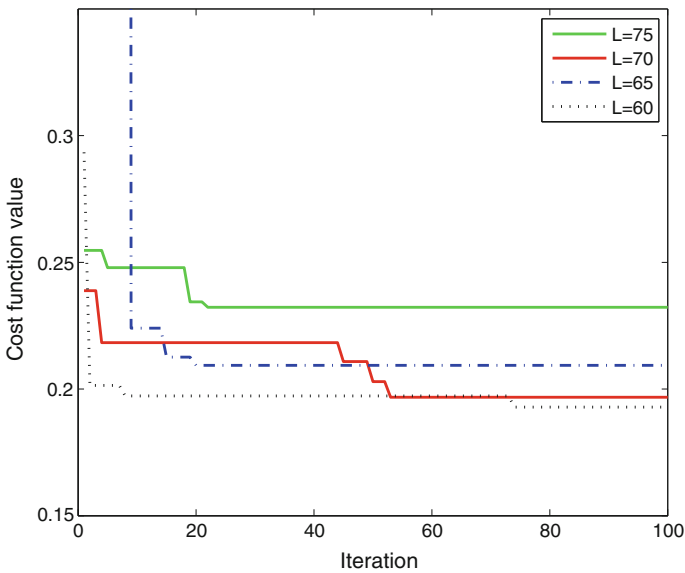


**Fig. 5** Robustness convergence under control parameters variation of the DSA-based approach: ISE criterion case

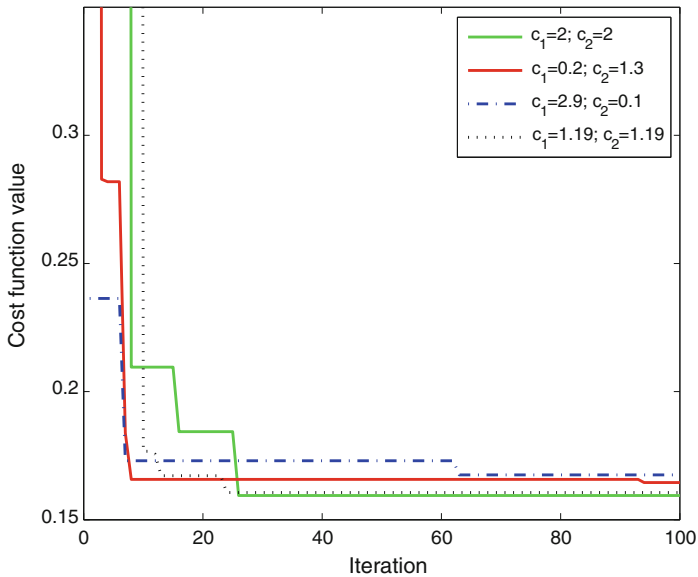
In this case study, we tested the proposed algorithms with different values of the population size in the range of [20, 50]. Globally, all the results found are close to each other. The best values of this control parameter are usually obtained while using a population size equal to 30.



**Fig. 6** Robustness convergence under control parameters variation of the GSA-based approach: ISE criterion case



**Fig. 7** Robustness convergence under control parameters variation of the ABC-based approach: ISE criterion case



**Fig. 8** Robustness convergence under control parameters variation of the PSO-based approach: ISE criterion case

Both for the MO and ISE criteria, the robustness on convergence of the proposed algorithms is guaranteed under their main control parameters variation. The qualities of the obtained solution, the fast convergence as well as the simple software implementation are comparable with the standard GAO-based approach. According to the convergence plots of the implemented metaheuristics, i.e., results of Figs. 5, 6, 7 and 8, the exploitation and exploration capabilities of these algorithms are ever guaranteed.

In this study, only simulation results from the ISE criterion case are illustrated. The main difference between performances of the implemented metaheuristics is their relative quickness or slowness in terms of CPU computation time. For this particular optimization problem, the quickness of DSA and PSO is specially marked in comparison with other techniques. Indeed, while using a Pentium IV, 1.73 GHz and MATLAB 7.7.0, the CPU computation times for the PSO algorithm are about 328 and 360 s in the MO and ISE criterion, respectively. For the DSA algorithm, these are about 296 and 310 s, respectively. For example and in the case of GSA metaheuristic, we obtain about 540 and 521 s for the above criterion respectively.

For the ISE criterion case, all optimization results are close to each other in terms of solutions quality, except those obtained by the ABC-based method. The relative numerical simulation shows the sensitivity of this algorithm under the “Limit for abandonment” parameter variation. The best optimization result, with fitness value equal to 0.1928, is obtained with  $L = 60$ .

On the other hand, the scaling parameters  $\lambda_i$ , given in Eq. 2, will be linearly increased at each iteration step so constraints are gradually enforced. In a generic and typical optimization problem, the quality of the solution will directly depend on the value of this algorithm control parameter. In this chapter and in order to make the proposed approach simple, great and constant scaling penalty parameters, equal to  $10^4$ , are used for numerical simulations. Indeed, simulation results show that with great values of  $\lambda_i$ , the control system performances are weakly degraded and the effects on the tuning parameters are less meaningful. The proposed constrained and improved algorithms convergence is faster than the case with linearly variable scaling parameters.

The time-domain performances of the proposed metaheuristics-tuned PID-type FLC structure are illustrated in Figs. 9 and 10. Only simulations from the DSA and PSO techniques implementation are presented. All results, for various obtained decision variables, are acceptable and show the effectiveness of the proposed fuzzy controllers tuning method. The robustness, in terms of external disturbances rejection, and tracking performances are guaranteed with degradations for some considered methods. The considered time-domain constraints for the PID-type FC tuning problems, such as the maximum values of overshoot  $\delta^{\max} = 20\%$ , steady state  $E_{ss}^{\max} = 0.0001$  and settling time  $t_s^{\max} = 0.9$  s, are usually respected.

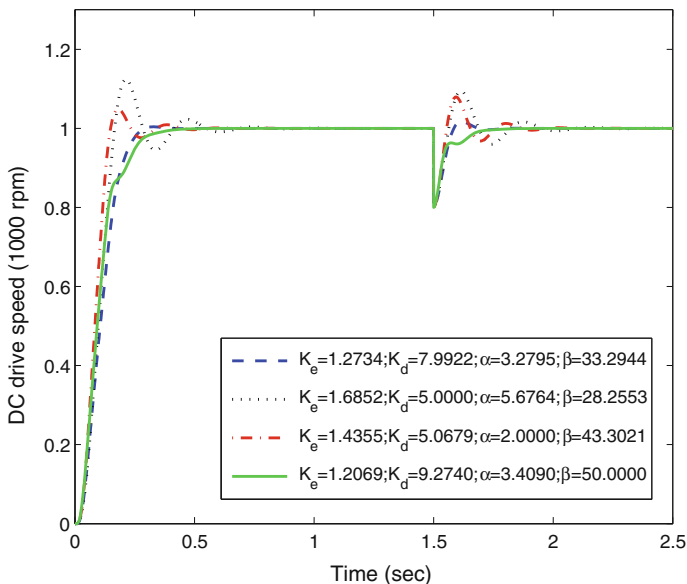
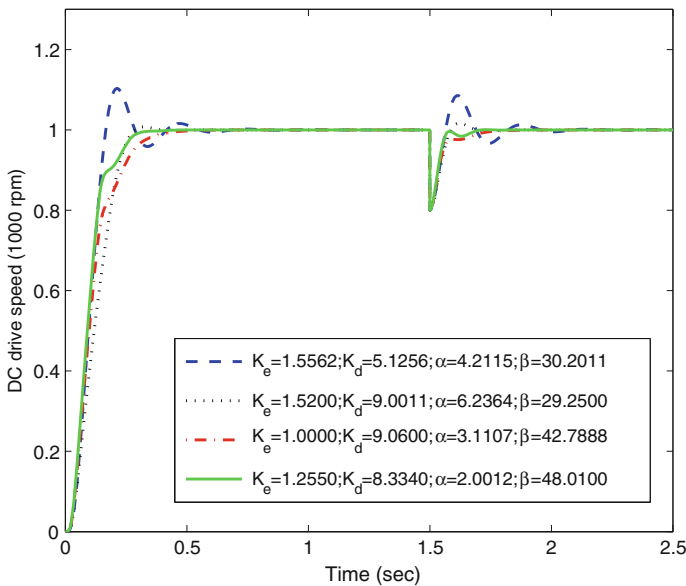


Fig. 9 Step responses of the DSA-tuned PID-type fuzzy controlled system: ISE criterion case



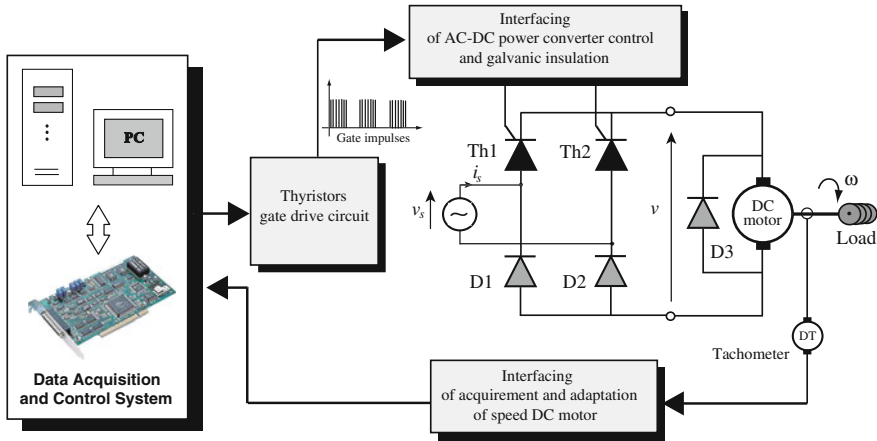
**Fig. 10** Step responses of the PSO-tuned PID-type fuzzy controlled system: ISE criterion case

#### 4.4 Experimental Results

In order to illustrate the efficiency of the proposed metaheuristics-tuned PID-type fuzzy control structure, we try to implement the controller within a real-time framework. The developed real-time application acquires input data, speed of the DC drive, and generates control signal for thyristors of AC-DC power converter as a PWM signal (Haggège et al. 2009). This is achieved using a digital control system based on a PC computer and a PCI-1710 multi-functions data acquisition board which is compatible with MATLAB/Simulink as described in Fig. 11.

The power part of the controlled process is constituted of the single-phase bridge rectifier converter. Figure 11 shows the considered half-controlled bridge rectifier, constituted by two thyristors and two diodes. The presence of thyristors makes the average output voltage controllable. A thyristor can be triggered by the application of a positive gate voltage and hence a gate current supplied from a gate drive circuit. The control voltage is generated with the help of a gate drive circuit, which is called a firing or triggering circuit. The used bridge thyristors are switched ON by a train of high-frequency impulses.

In order to obtain an impulse train, beginning with a fixed delay after the AC supply source zero-crossing, it is necessary to generate a sawtooth signal, synchronized with this zero-crossing. This is achieved using a capacitor charged with a constant current during the 10 ms half period of the AC source, and abruptly discharged at every zero-crossing instant. The constant current is obtained using a BC547 bipolar transistor whose base voltage is maintained constant due to a

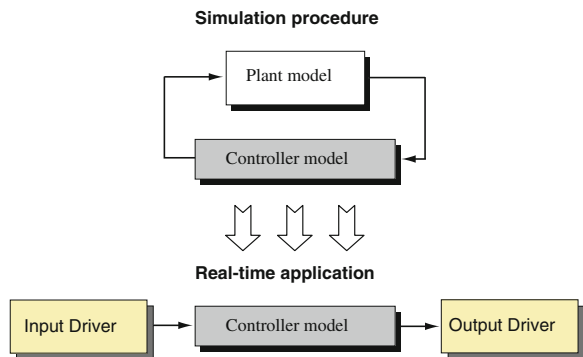


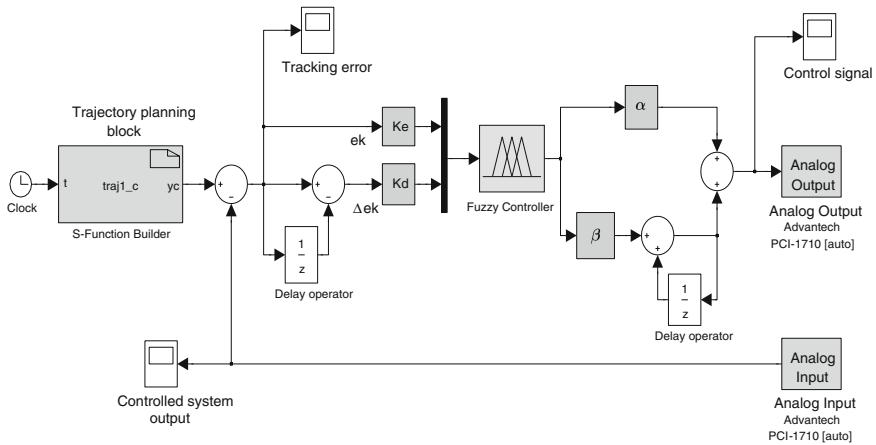
**Fig. 11** The proposed experimental setup schematic for DC drive control

polarization bridge constituted by a resistor and two 1N4148 diodes in series. This transistor acts as a current sink, whose value can be determined by an adjustable emitter resistor, to make the capacitor be fully charged after exactly 10 ms. The obtained synchronous saw-tooth signal is compared with a variable DC voltage, using an LM393 comparator, in order to generate a PWM signal which drives a NE555 timer, used as an astable multi-vibrator, producing the impulse train needed to control the thyristors firing. This impulse train is applied to the base of a 2N1711 bipolar transistor which drives an impulse transformer that ensuring the galvanic isolation between the control circuit and the power circuit.

The nominal model of the studied plant and the controller model obtained in synthesis development phase were used to implement the real-time controller. The model of the plant was removed from the simulation model, and instead of it, input device drivers (sensor) and output device driver (actuator) were introduced as shown in Fig. 12. These device drivers close the feedback loop when moving from simulations to experiments. According to this concept, the Fig. 13 illustrates the

**Fig. 12** Synoptic of the PCI-1710 based real-time controller implementation





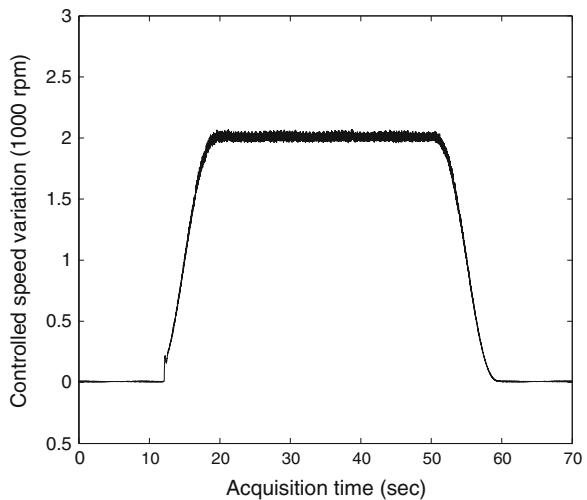
**Fig. 13** PCI-1710 board based implementation of the proposed FLC structure

principle of the implementation based on the Real-Time Windows Target tool of MATLAB/Simulink.

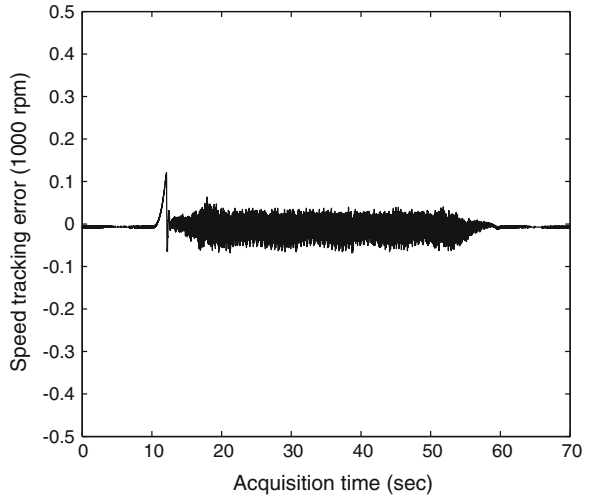
The real-time fuzzy controller is developed through the compilation and linking stage, in a form of a Dynamic Link Library (DLL), which is then loaded in memory and started-up. The used environment of real-time controller has some capabilities such as automatic code generation in C language, automatic compilation, start-up of a real-time program and external mode start-up of the simulation phase model allowing for real-time set monitoring and on-line adjustment of its parameters.

The real-time implementation of the proposed metaheuristics-tuned PID-type FLC leads to the experimental results of Figs. 14, 15, 16 and 17.

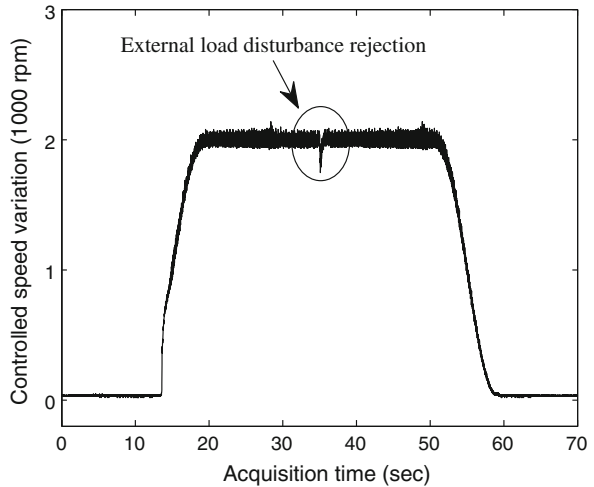
**Fig. 14** Experimental results of the PID-type FLC implementation: controlled speed variation



**Fig. 15** Experimental results of the PID-type FLC implementation: speed tracking error



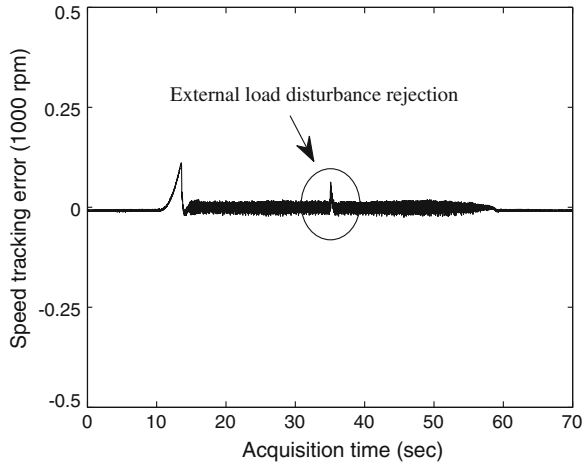
**Fig. 16** Robustness of the external disturbance rejection: controlled speed variation



In comparison with the results in Haggège et al. (2009) for a such plant, obtained by the use of a full order  $\mathcal{H}_\infty$  controller, as well as those obtained by PID-type FLC with trials-errors tuning method in Haggège et al. (2010), the experimental results of this study are satisfactory for a simple, non conventional and systematic meta-heuristics-based control approach. They point out the controller's viability and performance. As shown in Figs. 14 and 15, the measured speed tracking error is small (less than 10 % of set point) showing the high performances of the proposed control, especially in terms of tracking. The robustness, in terms of external load disturbances of the proposed PID-type FLC approach, is shown in Figs. 16 and 17. The proposed fuzzy controller leads to reject the additive disturbances on the controlled system output with a fast and more damped dynamic.



**Fig. 17** Robustness of the external disturbance rejection: speed tracking error



Globally, the obtained simulation and experimental results for the considered ISE and MO criteria are satisfactory. Others performance criteria, such as gain and margin specifications (Azar and Serrano 2014), can be used in order to improve the robustness and efficiency of the proposed fuzzy control approach.

## 5 Conclusion

A new method for tuning the scaling factors of Mamdani fuzzy controllers, based on advanced metaheuristics, is proposed and successfully applied to an electrical DC drive speed control. This efficient metaheuristics-based tool leads to a robust and systematic PID-type fuzzy control design approach. The comparative study shows the efficiency of the proposed techniques in terms of convergence speed and quality of the obtained solutions. This hybrid PID-type fuzzy design methodology is systematic, practical and simple without need to exact analytical plant model description. The obtained simulation and experimental results show the efficiency in terms of performance and robustness. All used DSA, GSA, ABC and PSO techniques produce near results in comparison with each others. Small degradations are always marked by going from one technique to another. The application of the proposed control approach, for more complex and non linear systems, constitutes our future works. The tuning of other fuzzy control structures, such as those described by Takagi-Sugeno inference mechanism, will be investigated.

## References

- Azar, A. T. (Ed.) (2010a). *Fuzzy systems*. Vienna, Austria: INTECH. ISBN 978-953-7619-92-3.
- Azar, A. T. (2010b). Adaptive neuro-fuzzy systems. In *Fuzzy systems*. Vienna, Austria: INTECH. ISBN 978-953-7619-92-3.
- Azar, A. T. (2012). Overview of type-2 fuzzy logic systems. *International Journal of Fuzzy System Applications*, 2(4), 1–28.
- Azar, A. T., & Serrano, F. E. (2014). Robust IMC-PID tuning for cascade control systems with gain and phase margin specifications. *Neural Computing and Applications*,. doi:10.1007/s00521-014-1560-x.
- Basturk, B. & Karaboga, D. (2006). An artificial bee colony (ABC) algorithm for numeric function optimization. In *Proceedings of IEEE Swarm Intelligence Symposium*, May 12–14, Indianapolis, USA.
- Bouallègue, S., Haggège, J., Ayadi, M., & Benrejeb, M. (2012a). PID-type fuzzy logic controller tuning based on particle swarm optimization. *Engineering Applications of Artificial Intelligence*, 25(3), 484–493.
- Bouallègue, S., Haggège, J., & Benrejeb, M. (2011). Particle swarm optimization-based fixed-structure  $\mathcal{H}_\infty$  control design. *International Journal of Control, Automation and Systems*, 9(2), 258–266.
- Bouallègue, S., Haggège, J., & Benrejeb, M. (2012b). A new method for tuning PID-type fuzzy controllers using particle swarm optimization. In *Fuzzy Controllers: Recent Advances in Theory and Applications* (pp. 139–162). Vienna, Austria: INTECH. ISBN 978-953-51-0759-0.
- Boussaid, I., Lepagnot, J., & Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences*, 237(1), 82–117.
- Civicioglu, P. (2012). Transforming geocentric Cartesian coordinates to geodetic coordinates by using differential search algorithm. *Computers and Geosciences*, 46(1), 229–247.
- David, R. C., Precup, R. E., Petriu, E. M., Radac, M. B., & Preitl, S. (2013). Gravitational search algorithm-based design of fuzzy control systems with a reduced parametric sensitivity. *Information Sciences*, 247(1), 154–173.
- Dréo, J., Siarry, P., Pétrowski, A., & Taillard, E. (2006). *Metaheuristics for Hard Optimization Methods and Case Studies*. Heidelberg: Springer.
- Eberhart, R. & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the 6th International Symposium on Micro Machine and Human Science* (pp. 39–43), October 4–6, Nagoya, Japan.
- Eker, I., & Torun, Y. (2006). Fuzzy logic control to be conventional method. *Energy Conversion and Management*, 47(4), 377–394.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Boston: Addison-Wesley Publishing Company.
- Goswami, D., & Chakraborty, S. (2014). Differential search algorithm-based parametric optimization of electrochemical micromachining processes. *International Journal of Industrial Engineering Computations*, 5(1), 41–54.
- Guzelkaya, M., Eksin, I., & Yesil, E. (2003). Self-tuning of PID type fuzzy logic controller coefficients via relative rate observer. *Engineering Applications of Artificial Intelligence*, 16(3), 227–236.
- Haggège, J., Ayadi, M., Bouallègue, S., & Benrejeb, M. (2010). Design of Fuzzy Flatness-based Controller for a DC Drive. *Control and Intelligent Systems*, 38(3), 164–172.
- Haggège, J., Bouallègue, S., & Benrejeb, M. (2009). Robust  $\mathcal{H}_\infty$  Design for a DC Drive. *International Review of Automatic Control*, 2(4), 415–422.
- Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Technical report TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, Turkey.
- Karaboga, D., & Akay, B. (2009). A comparative study of Artificial Bee Colony algorithm. *Applied Mathematics and Computation*, 214(1), 108–132.

- Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of Global Optimization*, 39(3), 459–471.
- Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing*, 8(1), 687–697.
- Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2012). A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artificial Intelligent Review*, 42, 21–57. doi:10.1007/s10462-012-9328-0.
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings. of IEEE International Joint Conference on Neural Networks* (pp. 1942–1948), November 27–December 01, Perth, Australia.
- Lee, C. C. (1998a). Fuzzy logic in control systems: Fuzzy logic controller-part I. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2), 404–418.
- Lee, C. C. (1998b). Fuzzy logic in control systems: Fuzzy logic controller-part II. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(2), 419–435.
- Madiouni, R., Bouallègue, S., Haggège, J., & Siarry, P. (2013). Particle swarm optimization-based design of polynomial RST controllers. In *Proceedings. of the 10th IEEE International Multi-Conference on Systems, Signals and Devices* (pp. 1–7), Hammamet, Tunisia.
- MathWorks. (2009). Genetic algorithm and direct search toolbox user's guide.
- Nobahari, H., Nikusokhan, M., & Siarry, P. (2011). Non-dominated sorting gravitational search algorithm. In *Proceedings. of the International conference on swarm intelligence* (pp. 1–10), June 14–15, Cergy, France.
- Passino, K. M. & Yurkovich, S. (1998). *Fuzzy control*. Boston, Addison Wesley Longman.
- Precup, R. E., David, R. C., Petriu, E. M., Preitl, S., & Radac, M. B. (2011). Gravitational search algorithms in fuzzy control systems tuning. In *Proceedings. of the 18th IFAC World Congress* (pp. 13624–13629), August 28–September 02, Milano, Italy.
- Qiao, W. Z., & Mizumoto, M. (1996). PID type fuzzy controller and parameters adaptive method. *Fuzzy Sets and Systems*, 78(1), 23–35.
- Rao, R. V. and Savsani, V. J. (2012). *Mechanical design optimization using advanced optimization techniques*. Heidelberg: Springer.
- Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA: A gravitational search algorithm. *Information Sciences*, 179(13), 2232–2248.
- Siarry, P., & Michalewicz, Z. (2008). *Advances in metaheuristics for hard optimization*. New York: Springer.
- Toumi, F., Bouallègue, S., Haggège, J., & Siarry, P. (2014). Differential search algorithm-based approach for PID-type fuzzy controller tuning. In *Proceedings. of the International Conference on Control, Engineering & Information Technology*, March 22–25, Sousse, Tunisia.
- Waghole, V., & Tiwari, R. (2014). Optimization of needle roller bearing design using novel hybrid methods. *Mechanism and Machine Theory*, 72(1), 71–85.
- Woo, Z. W., Chung, H. Y., & Lin, J. J. (2000). A PID type fuzzy controller with self-tuning scaling factors. *Fuzzy Sets and Systems*, 115(2), 321–326.