Chapter 3
Calibration of Fractional Fuzzy Controllers by Using the Social-Spider Method

Fuzzy controllers (FCs) based on integer concepts have proved their interesting capacities in several engineering domains. The fact that dynamic processes can be more precisely modeled by using fractional systems has generated a great interest in considering the design of FCs under fractional principles. In the design of fractional FCs, the parameter adjustment operation is converted into a multidimensional optimization task where fractional orders, and controller parameters, are assumed as decision elements. In the design of fractional FCs, the complexity of the optimization problem produces multi-modal error surfaces which are significantly hard to solve. Several metaheuristic algorithms have been successfully used to identify the optimal elements of fractional FCs. But, most of them present a big weakness since they usually get sub-optimal solutions as a result of their improper balance between exploitation and exploration in their search process. This chapter analyses the optimal parameter calibration of fractional FCs. To determine the best elements, the approach employs the Social Spider Optimization (SSO) algorithm, which is based on the simulation of the cooperative operation of social-spiders. In SSO, candidate solutions represent a group of spiders, which interact with each other by considering the biological concepts of the spider colony. Different to most of the metaheuristic algorithms, the approach explicitly avoids the concentration of solutions in the promising positions, eliminating critical defects such as the premature convergence and the deficient balance of exploration–exploitation.

3.1 Introduction

A fractional order model is a system that is characterized by a fractional differential equation containing derivatives of non-integer order. In fractional calculus, the integration and the differentiation operators are generalized into a non-integer order element, where is a fractional number and a and t symbolize the operator limits [1, 2]. Several dynamic systems can be more accurately described and controlled by

E. Cuevas et al., *Advances in Metaheuristics Algorithms: Methods and Applications*, Studies in Computational Intelligence 775,
https://doi.org/10.1007/978-3-319-89309-9_3

fractional models in comparison to integer order schemes. For this reason, in the last decade, the fractional order controllers [3–5] have attracted the interests of several research communities.

A fractional-order controller incorporates an integrator of order and a differentiator of order. The superior performance of such controllers with regard to conventional PIDs has been widely demonstrated in the literature [6].

On the other hand, fuzzy logic [7] provides an alternative method to design controllers through the use of heuristic information. Remarkably, such heuristic information may come from a human-operator who directly manipulates the process. In the fuzzy logic methodology, a human operator defines a set of rules on how to control a process, then incorporated it into a fuzzy controller that emulates the decision-making process of the operator [8].

Fractional fuzzy controllers (FFCs) are the results of the combination of conventional fuzzy controllers and fractional operators. Under such combination, FFCs exhibit better results than conventional FCs for an extensive variety of dynamical systems. This capacity is attributed to the additional flexibility offered by the inclusion of the fractional parameters.

Parameter calibration is an important step to implement applications with FFCs. This procedure is long and time consuming, since it is commonly conducted through trial and error. Therefore, the problem of parameter estimation in FFCs can be handled by evolutionary optimization methods. In general, they have demonstrated to deliver interesting results in terms of accuracy and robustness [5]. In these methods, an individual is represented by a candidate parameter set, which symbolizes the configuration of a determined fractional fuzzy controller. Just as the evolution process unfolds, a set of evolutionary operators is applied in order to produce better individuals. The quality of each candidate solution is evaluated through an objective function whose final result represents the performance of the parameter set in terms of the produced error. Some examples of such approaches being applied to the identification of fractional order systems have involved methods such as Genetic Algorithms (GA) [5], Particle Swarm Optimization (PSO) [9], Harmony Search (HS) [10], Gravitational Search Algorithm (GSA) [11] and Cuckoo Search (CS) [12]. Although such algorithms present interesting results, they have exhibited an important limitation: they frequently obtained sub-optimal solutions as a consequence of the limited balance between exploration and exploitation in their search strategies. Such limitation is associated to the evolutionary operators that have been employed to modify individual positions. In such algorithms, during their operation, the position of each individual for the next iteration is updated, producing an attraction towards the position of the best particle seen so far or towards other promising individuals. Therefore, as the algorithm evolves, such behaviors cause that the entire population concentrate rapidly around the best particles, favoring the premature convergence and damaging the appropriate exploration of the search space [13, 14].

The Social Spider Optimization (SSO) algorithm [15] is a recent evolutionary computation method that is inspired on the emulation of the collaborative behavior of social-spiders. In SSO, solutions imitate a set of spiders, which cooperate to each

other based on the natural laws of the cooperative colony. Unlike the most popular evolutionary algorithms such as GA [16], PSO [17], HS [18], GSA [19] and CS [20], it explicitly evades the concentration of individuals in the best positions, avoiding critical flaws such as the premature convergence to sub-optimal solutions and the limited balance of exploration–exploitation. Such characteristics have motivated the use of SSO to solve an extensive variety of engineering applications such as energy theft detection [21], machine learning [22], electromagnetics [23], image processing [24] and integer programming problems [25].

This chapter presents a method for the optimal parameter calibration of fractional FCs based on the SSO algorithm. Under this approach, the calibration process is transformed into a multidimensional optimization problem where fractional orders, as well as controller parameters of the fuzzy system, are considered as a candidate solution to the calibration task. In the method, the SSO algorithm searches the entire parameter space while an objective function evaluates the performance of each parameter set. Conducted by the values of such objective function, the group of candidate solutions are evolved through the SSO algorithm so that the optimal solution can be found. Experimental evidence shows the effectiveness and robustness of the method for calibrating fractional FCs. A comparison with similar methods such as the Genetic Algorithms (GA), the Particle Swarm Optimization (PSO), the Harmony Search (HS), the Gravitational Search Algorithm (GSA) and the Cuckoo Search (CS) on different dynamical systems has been incorporated to demonstrate the performance of this approach. Conclusions of the experimental comparison are validated through statistical tests that properly support the discussion.

The Chapter is organized as follows: Sect. 3.2 introduces the concepts of fractional order systems; Sect. 3.3 describes the fractional fuzzy controller used in the calibration; Sect. 3.4 presents the characteristics of SSO; Sect. 3.5 formulates the parameter calibration problem; Sect. 3.6 shows the experimental results while some final conclusions are discussed in Sect. 3.7.

3.2 Fractional-Order Models

Dynamical fractional-order systems are modeled by using differential equations, which involve non-integer integral and/or derivative operators [26, 27]. Since these operators produce irrational continuous transfer functions, or infinite dimensional discrete transfer functions, fractional models are normally studied through simulation tools instead of analytical methods [5, 28–33]. The remainder of this section provides a background of the fundamental aspects of the fractional calculus, and the discrete integer-order approximations of fractional order operators that are used in this paper.

3.2.1 Fractional Calculus

Fractional calculus is a generalization of integration and differentiation to the non-integer order fundamental operator. The differential-integral operator, denoted by ${}_aD_t^\alpha$, takes both the fractional derivative and the fractional integral into a single expression defined as:

$$ {}_aD_t^\alpha = \begin{cases} \frac{d^\alpha}{dt^\alpha}, & \alpha > 0, \\ 1, & \alpha = 0, \\ \int_a^t (d\tau)^\alpha, & \alpha < 0. \end{cases} \tag{3.1} $$

where a and t represent the operation bounds, whereas $\alpha \in \Re$. The commonly used definitions for fractional derivatives are the Grünwald-Letnikov, Riemann-Liouville [34] and Caputo [35]. According to the Grünwald-Letnikov approximation, the fractional-order derivative of order α is defined as follows:

$$ D_t^\alpha f(t) = \lim_{h \to 0} \frac{1}{h^\alpha} \sum_{j=0}^{\infty} (-1)^j \binom{\alpha}{j} f(t - jh) \tag{3.2} $$

In the numerical calculation of fractional-order derivatives, the explicit numerical approximation of the α-th derivative at the points $kh, (k = 1, 2, \ldots)$ maintains the following form [36]:

$$ {}_{(k-M_m/h)}D_{t_k}^\alpha f(t) \approx h^{-\alpha} \sum_{j=0}^{k} (-1)^j \binom{\alpha}{j} f(t_k - j) \tag{3.3} $$

where M_m is the memory length $t_k = kh$, h is the time step and $(-1)^j \binom{\alpha}{j}$ are the binomial coefficients. For their calculation, we can use the following expression:

$$ c_0^{(\alpha)} = 1, \quad c_j^{(\alpha)} = \left(1 - \frac{1+\alpha}{j}\right) c_{j-1}^{(\alpha)} \tag{3.4} $$

Then, the general numerical solution of the fractional differential equation is defined as follows:

$$ y(t_k) = f(y(t_k), t_k) h^\alpha - \sum_{j=1}^{k} c_j^{(\alpha)} y(t_{k-j}) \tag{3.5} $$

3.2.2 Approximation of Fractional Operators

Assuming zero initial conditions, the fractional operator is defined in the Laplace domain as $L({}_aD_t^{\alpha}f(t)) = s^{\alpha}F(s)$. Several approaches [37–39] have been proposed for producing discrete versions of continuous operators of type s^{α}. In this chapter, the Grünwald-Letnikov approximation has been used due to its interesting properties for generating discrete equivalences [40–43]. Under this method, the discretization considers the following model:

$$D^{\alpha}(z^{-1}) = \left(\frac{1-z^{-1}}{T_c}\right) = \sum_{k=0}^{\infty}\left(\frac{1}{T_c}\right)^{\alpha}(-1)\binom{\alpha}{k}z^{-k} = \sum_{k=0}^{\infty}h^{\alpha}(k)z^{-k}, \tag{3.6}$$

where $h^{\alpha}(k)$ is the impulse response sequence, whereas T_c represents the sampling frequency. It has been already demonstrated in the literature [36] that rational models converge faster than polynomial methods. Consequently, the Padé approximation approach has been employed to obtain a fractional model from the impulse response by using the definition provided in Eq. 3.7.

$$H(z^{-1}) = \frac{b_0 + b_1z^{-1} + \cdots + b_mz^{-m}}{1 + a_1z^{-1} + \cdots + a_nz^{-n}} = \sum_{k=0}^{\infty}h(k)z^{-k}, \quad m \leq n \tag{3.7}$$

where m, n and the parameters a_i and b_i are calculated by adjusting the first $m + n + 1$ coefficients of $h^{\alpha}(k)$.

3.3 Fuzzy Controller

A fuzzy controller (FC) is a nonlinear system produced from empirical rules. Such empirical information may come from a human operator who directly manipulates the process. Each rule, just as the natural language, presents an IF-THEN format. The collection of all rules constitutes the rule base that emulates the decision-making process of the operator. An important characteristic of one FC is the partitioning of the control scheme into regions [44]. At each region, the control strategy can be simply modeled by using a rule that associates the region under which certain actions are performed. Despite proposing several configurations of FCs in the literature, the fuzzy fractional $\mathrm{PD}^{\alpha} + \mathrm{I}$ has been selected since it presents interesting characteristics of robustness and stability [5]. In this structure, the integral error is incorporated to the output of the fuzzy fractional PD^{α} controller. Under this configuration, the integral action supports the elimination of the final steady state error.

The controller configuration is shown in Fig. 3.1. In the Figure, E, DE and IE represents the error, the fractional derivative error and the integral error, respectively. It has four gains K_p, K_d, K_i and K_u to be calibrated, the first three gains

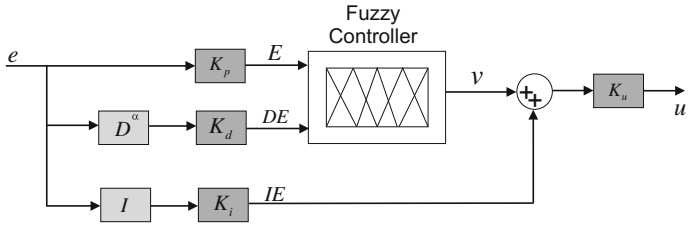

Fig. 3.1 Fuzzy $PD^{\alpha} + I$ controller

Table 3.1 Rule base of the controller to be calibrated

E/DE	NL	NM	NS	ZR	PS	PM	PL
NL	NL	NL	NL	NL	NM	NS	ZR
NM	NL	NL	NL	NM	NS	ZR	PS
NS	NL	NL	NM	NS	ZR	PS	PM
ZR	NL	NM	NS	ZR	PS	PM	PL
PS	NM	NS	ZR	PS	PM	PL	PL
PM	NS	ZR	PS	PM	PL	PL	PL
PL	ZR	PS	PM	PL	PL	PL	PL

correspond to the input and the last one to the output. The control action u is a nonlinear mapping function of E, DE and IE with the following model:

$$\begin{aligned} u(k) &= (f(E, CE) + IE)K_u \\ u(k) &= \left[f\left(K_p e, K_d D^{\alpha} e\right) + K_i Ie\right] \cdot K_u, \end{aligned} \tag{3.8}$$

A fuzzy controller consists of three conceptual components: a rule base, which contains a selection of fuzzy rules; a database, which defines the membership functions used by the fuzzy rules; and a reasoning mechanism, which performs the inference procedure. There are two different fuzzy systems: the Mamdani [45] and the Takagi-Sugeno (TS) [46]. In order to maintain compatibility to similar works reported in the literature, the rule base and the membership functions are selected the same as [5, 9]. Under such conditions, Table 3.1 shows the rule base used by the fuzzy controller to be calibrated. In Table 3.1, **NL**, **NM**, **NS**, **ZR**, **PS**, **PM** and **PL** represent the linguistic variables "Negative Large", "Negative Medium", "Negative Small", "Zero", "Positive Small", "Positive Medium" and "Positive Large", respectively. Figure 3.2 shows the membership functions that model the premises and the consequences of each rule. Consequently, a determined rule from Table 3.1 can be constructed in the following form:

If ***E*** *is* **NL** and ***DE*** is **ZR** then ***v*** is **NL**

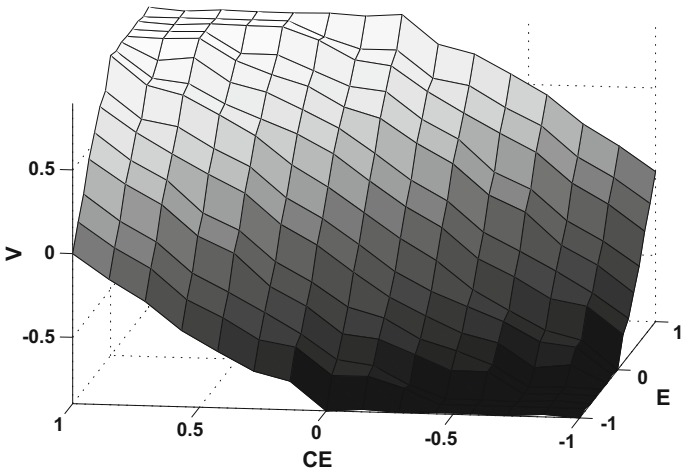

Fig. 3.2 Control surface

In this rule, the control strategy can be simply modeled as follows: if the error is "Negative Large" and the error derivate "Zero" then the output is "Negative Large". The acting of all rules produces the control strategy which is shown by the non-linear surface in Fig. 3.2.

3.4 Social Spider Optimization (SSO)

The social spider optimization (SSO) algorithm [15] is an evolutionary computation method that emulates the cooperative behavior of spiders within a communal colony. SSO has been designed to find the global solution of a nonlinear optimization problem with box constraints in the form:

$$\begin{array}{ll} \text{minimize} & f(\mathbf{x}) \quad \mathbf{x} = (x^1, \ldots, x^d) \in \mathbb{R}^d \\ \text{subject to} & \mathbf{x} \in \mathbf{X} \end{array} \tag{3.9}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a nonlinear function whereas $\mathbf{X} = \left\{\mathbf{x} \in \mathbb{R}^d \middle| l_h \leq x^h \leq u_h, h = 1, \ldots, d\right\}$ is a bounded feasible space, constrained by the lower (l_h) and upper (u_h) limits.

SSO utilizes a population $\mathbf{S}$ of N candidate solutions to solve the problem formulated in Eq. 3.1. Each candidate solution represents a spider position whereas the general web symbolizes the search space $\mathbf{X}$. In SSO, the spider population $\mathbf{S}$ is classified into two categories: males ($\mathbf{M}$) and females ($\mathbf{F}$). In order to simulate a real spider colony, in SSO, the number N_f of females $\mathbf{F}$ is randomly selected within a range of 65–90% of the entire population $\mathbf{S}$, whereas the rest N_m is considered as

male individuals ($N_m = N - N_f$). Under such conditions, the Group $\mathbf{F}$ assembles the set of female individuals ($\mathbf{F} = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{N_f}\}$) whereas $\mathbf{M}$ groups the male members ($\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{N_m}\}$), where $\mathbf{S} = \mathbf{F} \cup \mathbf{M}$ ($\mathbf{S} = \{\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_N\}$), such that $\mathbf{S} = \{\mathbf{s}_1 = \mathbf{f}_1, \mathbf{s}_2 = \mathbf{f}_2, \ldots, \mathbf{s}_{N_f} = \mathbf{f}_{N_f}, \mathbf{s}_{N_f+1} = \mathbf{m}_1, \mathbf{s}_{N_f+2} = \mathbf{m}_2, \ldots, \mathbf{s}_N = \mathbf{m}_{N_m}\}$.

In the approach, each spider i maintain a weight w_i according to its solution quality. Therefore, w_i is calculated as follows:

$$w_i = \frac{fitness_i - worst}{best - worst} \tag{3.10}$$

where $fitness_i$ represents the fitness value produced by the evaluation of the i-th spider's position, $i \in 1, \ldots, N$. *best* and *worst* symbolize the best fitness value and worst fitness value of the whole population $\mathbf{S}$, respectively.

In the optimization process, the main mechanism of SSO is the information exchange, which it is simulated trough vibrations produced in the communal web. The vibration that a spider i perceives from a spider j is modeled with the following expression:

$$V_{i,j} = w_j e^{d_{i,j}^2} \tag{3.11}$$

where w_j represents the weight of the spider j and $d_{i,j}^2$ the distance between both spiders. It is considered that each spider i is only able to perceive three types of vibrations $V_{i,c}$, $V_{i,b}$ and $V_{i,f}$.

$V_{i,c}$ is the vibration transmitted by the nearest individual c with a higher weight with regard to $i (w_c > w_i)$. $V_{i,b}$ represents the vibration emitted by best element of the entire population $\mathbf{S}$. Finally, $V_{i,f}$ considers the vibration produced by the nearest female spider. This vibration type is only applicable if i is a male individual.

In the operation of SSO, a population of N spiders is processed from the initial stage (k = 0) to a determined number *gen* of iterations ($k = gen$). Each individual depending on its gender is conducted by a set of different evolutionary operators. Therefore, in case of the female members, a new position $\mathbf{f}_i^{k+1}$ is generated by modifying the current element location $\mathbf{f}_i^k$. The modification is randomly controlled by using a probability factor *PF*. Consequently, the movement is produced in relation to other spiders according their vibrations, which are transmitted trough the communal web:

$$\mathbf{f}_i^{k+1} = \begin{cases} \mathbf{f}_i^k + \alpha \cdot V_{i,c} \cdot (\mathbf{s}_c - \mathbf{f}_i^k) + \beta \cdot V_{i,b} \cdot (\mathbf{s}_b - \mathbf{f}_i^k) + \delta \cdot (\text{rand} - \frac{1}{2}) & \text{with probability } PF \\ \mathbf{f}_i^k - \alpha \cdot V_{i,c} \cdot (\mathbf{s}_c - \mathbf{f}_i^k) - \beta \cdot V_{i,b} \cdot (\mathbf{s}_b - \mathbf{f}_i^k) + \delta \cdot (\text{rand} - \frac{1}{2}) & \text{with probability} 1 - PF \end{cases} \tag{3.12}$$

here α, β, δ and *rand* represent random numbers between [0,1] whereas k is the iteration number. The individuals $\mathbf{s}_c$ and $\mathbf{s}_b$ symbolize the nearest member to i that

maintains a higher weight and the best element of the complete population **S**, respectively.

On the other hand, male spider members are classified into two types: non-dominant (**ND**) and dominant (**D**). The dominant group **D** is composed by the half of the male individuals whose fitness values are better with regard to the complete male set. Consequently, the non-dominant (**ND**) category collects the rest of the male elements. In the optimization process, male members are operated according to the following model:

$$\mathbf{m}_i^{k+1} = \begin{cases} \mathbf{m}_i^k + \alpha \cdot V_{i,f} \cdot (\mathbf{s}_f - \mathbf{m}_i^k) + \delta \cdot (\text{rand} - \frac{1}{2}) & \text{if } \mathbf{m}_i^k \in \mathbf{D} \\ \mathbf{m}_i^k + \alpha \cdot \left(\frac{\sum_{h \in \mathbf{ND}} \mathbf{m}_h^k \cdot w_h}{\sum_{h \in \mathbf{ND}} w_h} - \mathbf{m}_i^k \right) & \text{if } \mathbf{m}_i^k \in \mathbf{ND} \end{cases} \tag{3.13}$$

where $\mathbf{s}_f$ symbolizes the nearest female element to the male individual i.

The final operation in SSO is mating. It is performed between dominant males and the female individuals. Under this operation, a new individual $\mathbf{s}_{new}$ is produced by the combination of a dominant male $\mathbf{m}_g$ and other female members within a specific range r. The weight of each involved element defines the probability of influence of each spider into $\mathbf{s}_{new}$. The elements with heavier weights are more likely to influence the new individual $\mathbf{s}_{new}$. Once $\mathbf{s}_{new}$ is generated, it is compared with the worst element of the colony. If $\mathbf{s}_{new}$ is better than the worst spider, the worst spider is replaced by $\mathbf{s}_{new}$. Otherwise, $\mathbf{s}_{new}$ is discarded. Figure 3.3 illustrates the operations of the optimization process performed by the SSO algorithm. More details can be found in [15].

3.5 Problem Formulation

In the design stage of fractional FCs, the parameter calibration process is transformed into a multidimensional optimization problem where fractional orders, as well as controller parameters of the fuzzy system, are both considered as decision variables. Under this approach, the complexity of the optimization problem tends to produce multimodal error surfaces whose cost functions are significantly difficult to minimize.

This chapter presents an algorithm for the optimal parameter calibration of fractional FCs. To determine the parameters, the estimation method uses the Social Spider Optimization (SSO) method. Different to the most of existent evolutionary algorithms, the method explicitly evades the concentration of individuals in the best positions, avoiding critical flaws such as the premature convergence to sub-optimal solutions and the limited balance of exploration–exploitation.

Therefore, the calibration process consists of finding the optimal controller parameters that present the best possible performance for the regulation of a dynamical system. Figure 3.4 illustrates the SSO scheme for the parameter calibration process.

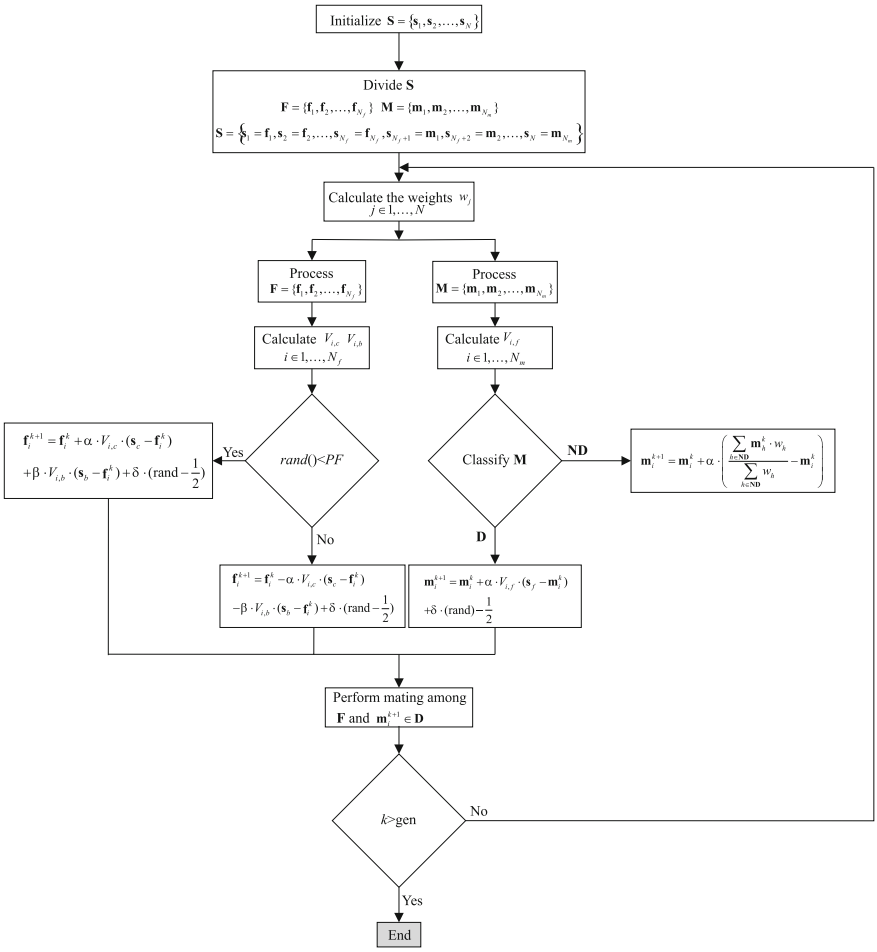

Fig. 3.3 Operations of the optimization process performed by the SSO algorithm

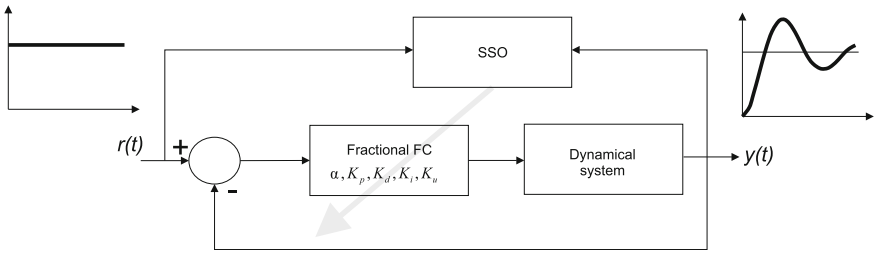

Fig. 3.4 SSO scheme for the parameter calibration process

Under such conditions, the fractional fuzzy controller parameters $(\alpha, K_p, K_d, K_i, K_u)$ represent the dimensions of each candidate solution (spider position) for the calibration problem. To evaluate the performance of the fractional fuzzy controller under each parameter configuration (candidate solution), the Integral Time Absolute Error (ITAE) [47] criterion has been considered. The ITAE index J measures the similarity between the closed-loop step response $y(t)$ produced by a determined parameter configuration $(\alpha, K_p, K_d, K_i, K_u)$ and the step function $r(t)$. Therefore, the quality of each candidate solution is evaluated according to the following model:

$$J(\alpha, K_p, K_d, K_i, K_u) = \int_0^\infty t|r(t) - y(t)| \tag{3.14}$$

Thereby, the problem of parameter calibration can be defined by the following optimization formulation:

$$\begin{array}{lll} \text{minimize} & J(\mathbf{x}) & \mathbf{x} = (\alpha, K_p, K_d, K_i, K_u) \in \mathbb{R}^5 \\ \text{subject to} & & 0 \leq \alpha \leq 3 \\ & & 0 \leq K_p \leq 5 \\ & & 0 \leq K_d \leq 5 \\ & & 0 \leq K_i \leq 5 \\ & & 0 \leq K_u \leq 5 \end{array} \tag{3.15}$$

3.6 Numerical Simulations

This section presents the performance of the SSO scheme for the calibration of fractional FCs considering several dynamical systems. The algorithm is also evaluated in comparison to other similar approaches that are based on evolutionary algorithms. To test the performance of the SSO approach, the technique uses a representative set of three transfer functions that have been previously employed. Equation 3.4–3.6 present the transfer functions that are used in our simulations. Such functions involve three different system categories: High-order plants $(G_1(s))$, non-minimum systems $(G_2(s))$ and dynamical fractional systems $(G_3(s))$.

$$G_1(s) = \frac{1}{(s+1)(1+0.5s)(1+0.25s)(1+0.125s)} \tag{3.16}$$

$$G_2(s) = \frac{1-5s}{(s+1)^3} \tag{3.17}$$

$$G_3(s) = \frac{1}{(s^{1.5}+1)} \tag{3.18}$$

In the experiments, we have applied the SSO algorithm to calibrate the fractional parameters for each dynamical systems, and the results are compared to those produced by the Genetic Algorithms (GA) [5], Particle Swarm Optimization (PSO) [9], Harmony Search (HS) [10], Gravitational Search Algorithm (GSA) [11] and Cuckoo Search (CS) [12]. In the comparison, all methods have been set according to their own reported guidelines. Such configurations are described as follows:

1. PSO, parameters $c_1 = 2$, $c_2 = 2$ and weights factors have been set to $w_{\max} = 0.9$, and $w_{\min} = 0.4$ [17].
2. GA, the crossover probability is 0.55, the mutation probability is 0.10 and number of elite individuals is 2. Furthermore, the roulette wheel selection and the 1-point crossover are both applied.
3. GSA, From the model $G_t = G_O e^{-\alpha \frac{t}{T}}$, it is considered $\alpha = 10$, $G_O = 100$ and $T = 100$ or $T = 500$.
4. HS, its parameters are set as follows: the harmony memory consideration rate $HMCR = 0.7$, the pitch adjustment rate $PAR = 0.3$ and the Bandwidth rate $BW = 0.1$.
5. CS, its elements are configured such as the discovery rate $p_a = 0.25$ and the stability index $\beta = 3/2$.
6. SSO, the parameter PF has been set to 0.7 following an experimental definition.

The experimental results are divided into three sub-sections. In the first Sect. (3.6.1), the performance of the SSO algorithm is evaluated with regard to high-order plants $(G_1(s))$. In the second Sect. (3.6.2), the results for non-minimum systems $(G_2(s))$ are provided and finally, in the third Sect. (3.6.3), the performance of the calibration scheme over fractional dynamical systems $(G_3(s))$ is discussed.

3.6.1 Results Over High-Order Plants $(G_1(s))$

In this experiment, the performance of the SSO calibration scheme is compared to GA, PSO, HS, GSA and CS, considering the regulation of high-order dynamical systems $(G_1(s))$. In the simulations, a temporal response from 0 to 10 s has been considered. In the comparison, all algorithms are operated with a population of 50 individuals ($\mathbf{N} = 50$). To appropriately evaluate the convergence properties of all calibration methods, the maximum number of generations has been set to (A) 100 iterations and (B) 500 iterations. This stop criterion has been selected to maintain compatibility to similar works reported in the literature [5, 9, 28]. By selecting such number of iterations, the experiment aims to test the quality of the produced

Table 3.2 Calibrated parameters for $G_1(s)$ produced by each algorithm after 100 iterations

$G_1(s)$	K_p	K_d	K_i	K_u	α	ITAE
PSO	0.3034	0	0.4475	2.9988	0	5281.2115
GA	0.7581	0.3510	0.3038	4.3276	0.8000	926.1352
GSA	1.3387	2.7209	0.5482	1.0545	0.2311	4164.1935
HS	0.7867	0.8128	0.8271	3.6129	0.9319	3562.1834
CS	0.9700	0.3497	0.4054	3.0002	0.9516	916.5816
SSO	0.8100	0.3493	0.2392	4.8235	0.9897	492.2912

Table 3.3 Calibrated parameters for $G_1(s)$ produced by each algorithm after 500 iterations

$G_1(s)$	K_p	K_d	K_i	K_u	α	ITAE
PSO	0	0.5061	0.4681	4.0578	0.4629	2900.7502
GA	1.1860	0.3826	0.5204	1.5850	0.9497	974.0881
GSA	1.2000	0.6531	0.9607	2.5442	0.8745	1975.3254
HS	1.3112	0.9450	0.9262	1.2702	0.6075	2776.2160
CS	1.0093	0.4506	0.2611	4.6964	1.0002	464.5376
SSO	1.0386	0.4621	0.2751	4.4147	0.9998	473.7492

solutions when the operation of each calibration method is limited to a reduced number of iterations.

All the experimental results in this section consider the analysis of 35 independent executions of each algorithm. Table 3.2 presents the calibrated parameters obtained through each method. Such results consider the best controller parameters in terms of the produced ITAE values after 100 iterations. On the other hand, Table 3.3 shows the calibrated parameters considering 500 iterations.

According to Tables 3.2 and 3.3, the SSO scheme provides better performance than GA, PSO, HS, GSA and CS for both cases. Such differences are directly related to a better trade-off between exploration and exploitation of the SSO method. It is also evident that the SSO method produces similar results with 100 or 500 iterations. Therefore, it can be established that SSO maintains better convergence properties than GA, PSO, HS, GSA and CS in the process of parameter calibration.

Figure 3.5 exhibits the step responses produced by each parameter set, considering 100 and 500 iterations. The remarkable convergence rate of the SSO algorithm can be observed at Fig. 3.5. According to the graphs, the step responses produced by the controller parameters that have been defined through SSO, are practically the same irrespective of the number of iterations. This fact means that the SSO scheme is able to find an acceptable solution in less than 100 iterations.

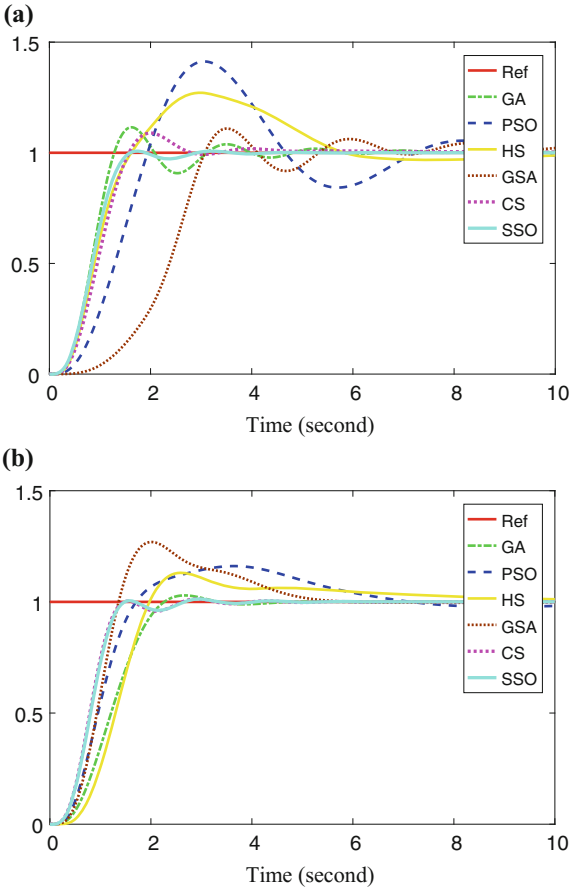

Fig. 3.5 Step responses after applying the calibrated parameters to the high order plant $G_1(s)$ with **a** 100 iterations, and with **b** 500 iterations

3.6.2 *Results Over Non-minimum Systems* $(G_2(s))$

This section presents the comparison of the SSO calibration scheme with GA, PSO, HS, GSA and CS, considering the regulation of non-minimum systems $(G_2(s))$. Non-minimum systems are defined by transfer functions with one or more poles or zeros in the right half of the s-plane. As a consequence, the response of a non-minimum system to a step input exhibits an "undershoot", which indicates that the output of the dynamical system becomes negative first before changing direction to positive values.

The experimental simulation runs from 0 to 50 s. All algorithms are operated with a population of 50 individuals (**N** = 50), matching with the experiments in Sect. 3.6.1. Table 3.4 presents the calibrated parameter of each method after 100 iterations, while Table 3.5 exhibits the results for 500 iterations. Both tables show that the SSO scheme delivers better results than GA, PSO, HS, GSA and CS in

Table 3.4 Calibrated parameters for $G_2(s)$ produced by each algorithm after 100 iterations

$G_2(s)$	K_p	K_d	K_i	K_u	α	ITAE
PSO	0.4645	0	0.2378	0.4147	0.0643	50289.0994
GA	0.6061	0.0326	0.3175	0.2909	0.2000	55043.6316
GSA	0.9377	1.8339	0.5020	0.1427	1.0266	101160.6241
HS	2.2838	3.7685	0.1328	0.5324	2.3214	126996.7047
CS	0.9305	1.1329	1.1045	0.0674	0.0222	90962.6199
SSO	0.4668	0.1165	0.2139	0.4642	0.5470	44368.6620

Table 3.5 Calibrated parameters for $G_2(s)$ produced by each algorithm after 500 iterations

$G_2(s)$	K_p	K_d	K_i	K_u	α	ITAE
PSO	0.4606	0	0.2027	0.4866	0.0688	46912.4985
GA	1.0449	1.1921	0.8839	0.0903	0.0822	81550.0790
GSA	0.8862	1.3919	0.3746	0.2386	1.9259	63186.5783
HS	1.0362	1.1105	0.5360	0.1389	0.9007	91536.3894
CS	0.0386	0.0059	0.0243	4.0625	0.5516	43565.1588
SSO	0.4537	0.1597	0.2004	0.5124	0.6422	41772.3344

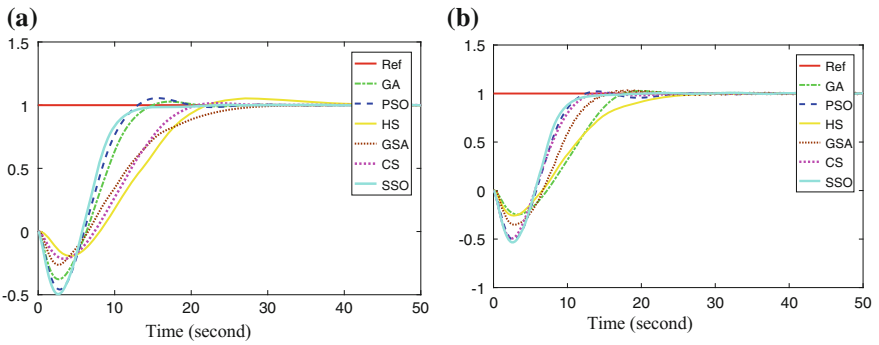

Fig. 3.6 Step responses after applying the calibrated parameters to the high order plant $G_2(s)$ with **a** 100 iterations, and with **b** 500 iterations

terms of the ITAE index. Figure 3.6 presents the step responses produced by each parameter set, considering 100 and 500 iterations. By analyzing the plot in Fig. 3.6, it is observed that the step response of the SSO scheme is less sensitive to the number of iterations than other techniques.

Table 3.6 Calibrated parameters for $G_3(s)$ produced by each algorithm after 100 iterations

$G_3(s)$	K_p	K_d	K_i	K_u	α	ITAE
PSO	1.331	0	0.6937	5	5	311.4558
GA	1.3329	0.6341	0.6130	5	0.4932	97.7016
GSA	1.0823	0.6463	0.2924	4.0152	0.5802	346.6765
HS	0.7867	0.8128	0.8271	3.6129	0.9319	3562.1834
CS	1.2220	0.6590	0.6647	5	0.4232	105.0266
SSO	1.3173	0.6560	0.5932	4.9797	0.5091	98.5974

Table 3.7 Calibrated parameters for $G_3(s)$ produced by each algorithm after 500 iterations

$G_3(s)$	K_p	K_d	K_i	K_u	α	ITAE
PSO	1.0187	1.2010	0.7553	5	0	181.5380
GA	1.3320	0.5599	0.5991	5	0.5631	97.1981
GSA	1.3319	0.7502	0.6689	3.4968	0.5185	152.2198
HS	1.3112	0.9450	0.9262	1.2702	0.6075	2776.2160
CS	1.3325	0.5857	0.5987	4.9999	0.5450	97.0307
SSO	1.3087	0.5883	0.5808	4.9991	0.5642	97.1085

3.6.3 *Results Over Fractional Systems* $(G_3(s))$

Unlike high-order plants and non-minimum systems, fractional dynamical systems produce multimodal error surfaces with different local optima. As a consequence, fractional fuzzy controllers that regulate their behavior are, in general, more difficult to calibrate [9]. Under such conditions, the experiment reflects the capacity of each calibration algorithm to locate the global optimum in presence of several local optima.

In this experiment, the performance of the SSO calibration scheme is compared to GA, PSO, HS, GSA and CS, considering the regulation of fractional dynamical systems $(G_3(s))$. In the simulations, a temporal response from 0 to 3 s is considered. In the test, all algorithms are operated with a population of 50 individuals ($\mathbf{N} = 50$).

The calibrated parameters are averaged over 30 executions obtaining the values reported in Tables 3.6 and 3.7. The results exhibit the configuration for each method with 100 and 500 iterations, respectively. It is evident that the SSO scheme presents better performance than PSO, HS and GSA independently on the number of iterations. However, the difference between GA and the SSO approach in terms of the ITAE index is relatively small for the case of 100 iterations. On the other hand, in the case of 500 iterations, the performance among the SSO approach, GA and CS are practically the same. Figure 3.7 presents the step response that is produced by each parameter set, considering 100 and 500 iterations. Similar to Sects. 6.1 and 6.2, it is demonstrated (from Fig. 3.7) that the SSO-calibrator obtains better solutions than GA, HS and PSO yet demanding a lower number of iterations.

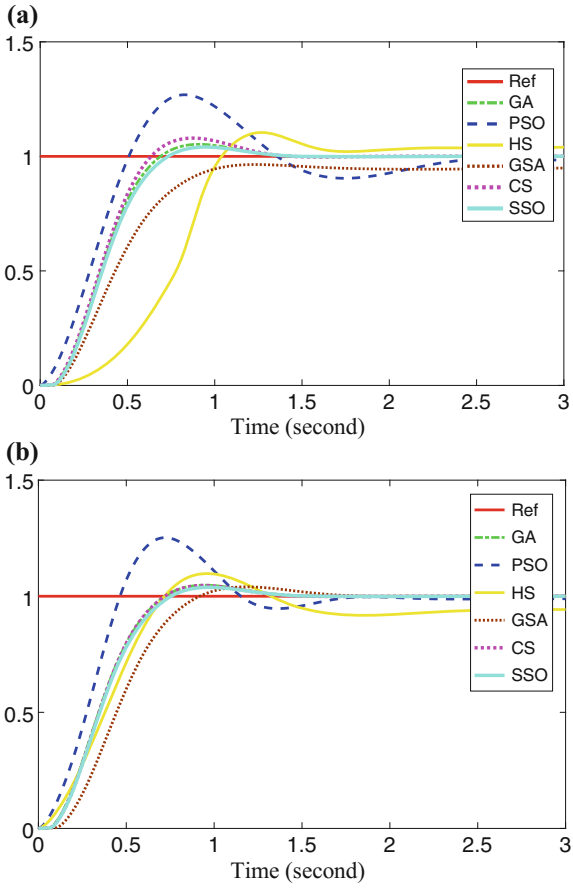

Fig. 3.7 Step responses after applying the calibrated parameters to the high order plant $G_3(s)$ with **a** 100 iterations, and with **b** 500 iterations

Finally, in order to stress the importance of the fractional PD$^\alpha$ + I scheme, an experiment that evaluates the influence of the parameter α in the regulation of $G_3(s)$ is conducted. In the experiment, the fractional PD$^\alpha$ + I controller is operated as an integer fuzzy controller by setting $\alpha = 1$. Under such conditions, the rest of the parameters of PD$^\alpha$ + I (K_p, K_d, K_i, K_u) are calibrated through the SSO approach considering the regulation of the fractional system $G_3(s)$. Then, the values α vary from 0 to 1 while registering the performance of the regulation.

As a result of the optimization method, the following parameter values are obtained: $(K_p, K_d, K_i, K_u) \equiv (1.3087, 0.5883, 0.5808, 4.0012)$ with ITAE = 418.8032. After calibrating the integer fuzzy controller, the values of α are modified from 0 to 1, while the parameter set remains fixed to $(1.3087, 0.5883, 0.5808, 4.0012)$. Table 3.8 presents the results obtained from the experiment. Such values report the regulation quality of $G_3(s)$ in terms of the ITEA values. By analyzing Table 3.8, it is clear that the regulation quality strongly depends on the selection of the order for α. Particularly in

Table 3.8 Regulation quality of $G_3(s)$ in terms of the ITEA values

α	ITAE
0	216.839600
0.1	198.666000
0.2	180.783670
0.3	159.871905
0.4	134.127486
0.5	108.724153
0.6	**105.942730**
0.7	136.404140
0.8	183.213124
0.9	262.521840
1	418.803215

Bold elements represent the best values

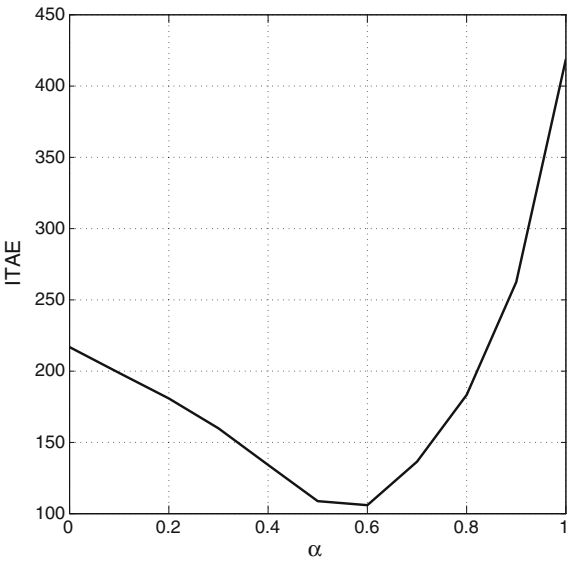

Fig. 3.8 Influence of α in the regulation quality of $G_3(s)$ in terms of the ITEA values

this experiment, the best regulation performance is reached when the order of α is set to 0.6. Figure 3.8 presents the influence of α in terms of the regulation quality.

3.7 Conclusions

Due to its multiple applications, the calibration of fractional fuzzy controllers has attracted the interests of several research communities. In the calibration process, the parameter estimation is transformed into a multidimensional optimization problem whose fractional order and the corresponding controller parameters of the

fuzzy system are considered as decision variables. Under this approach, the complexity of fractional-order chaotic systems tends to produce multimodal error surfaces for which their cost functions are significantly difficult to minimize. Several algorithms that are based on evolutionary computation principles have been successfully applied to calibrate the parameters of fuzzy systems. However, most of them still have an important limitation since they frequently obtain sub-optimal results as a consequence of an inappropriate balance between exploration and exploitation in their search strategies.

This chapter presents a method for the optimal parameter calibration of fractional FCs that is based on the SSO algorithm. The SSO algorithm is a novel evolutionary computation method that is inspired on the emulation of the collaborative behavior of social-spiders. Unlike most of the existing evolutionary algorithms, the method explicitly evades the concentration of individuals in best positions, avoiding critical flaws such as the premature convergence to sub-optimal solutions and the limited balance of exploration–exploitation.

In order to illustrate the proficiency and the robustness of this approach, SSO scheme has been experimentally evaluated considering three different system categories: high-order plants, non-minimum systems and dynamical fractional systems. To assess the performance of the SSO algorithm, it has been compared to other similar evolutionary approaches such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), Harmony Search (HS), Gravitational Search Algorithm (GSA) and Cuckoo Search (CS). The experiments have demonstrated that the SSO method outperforms other techniques in terms of solution quality and convergence.

References

1. Oldham, K.B., Spanier, J.: The Fractional Calculus: Theory and Application of Differentiation and Integration to Arbitrary Order. Academic Press, New York (1974)
2. Podlubny, I.: Fractional Differential Equations. Academic Press, San Diego (1999)
3. Das, S., Pan, I., Das, S., Gupta, A.: A novel fractional order fuzzy PID controller and its optimal time domain tuning based on integral performance indices. J. Eng. Appl. Artif. Intell. **25**, 430–442 (2012)
4. Delavari, H., Ghaderi, R., Ranjbar, A., Momani, S.: Fuzzy fractional order sliding mode controller for nonlinear systems. Commun. Nonlinear Sci. Numer. Simul. **15**(4), 963–978 (2010)
5. Jesus, I.S., Barbosa, R.S.: Genetic optimization of fuzzy fractional PD + I controllers. ISA Trans. **57**, 220–230 (2015)
6. Barbosa, R.S., Jesus, I.S.: A methodology for the design of fuzzy fractional PID controllers. In: ICINCO 2013—Proceedings of the 10th International Conference on Informatics in Control, Automation and Robotics, vol. 1, pp. 276–281 (2013)
7. Zadeh, L.A.: Fuzzy sets. Inf. Control **8**, 338–353 (1965)
8. He, Y., Chen, H., He, Z., Zhou, L.: Multi-attribute decision making based on neutral averaging operators for intuitionistic fuzzy information. Appl. Soft Comput. **27**, 64–76 (2015)
9. Pana, I., Das, S.: Fractional order fuzzy control of hybrid power system with renewable generation using chaotic PSO. ISA Trans. **62**, 19–29 (2016)

10. Roy, G.G., Chakraborty, P., Das, S.: Designing fractional-order PIλDμ controller using differential harmony search algorithm. Int. J. Bio-Inspired Comput. **2**(5), 303–309 (2010)
11. Xu, Y., Zhou, J., Xue, X., Fu, W., Zhu, W., Li, C.: An adaptively fast fuzzy fractional order PID control for pumped storage hydro unit using improved gravitational search algorithm. Energy Convers. Manage. **111**, 67–78 (2016)
12. Sharma, R., Rana, K.P.S., Kumar, V.: Performance analysis of fractional order fuzzy PID controllers applied to a robotic manipulator. Expert Syst. Appl. **41**, 4274–4289 (2014)
13. Tan, K.C., Chiam, S.C., Mamun, A.A., Goh, C.K.: Balancing exploration and exploitation with adaptive variation for evolutionary multi-objective optimization. Eur. J. Oper. Res. **197**, 701–713 (2009)
14. Chen, G., Low, C.P., Yang, Z.: Preserving and exploiting genetic diversity in evolutionary programming algorithms. IEEE Trans. Evol. Comput. **13**(3), 661–673 (2009)
15. Cuevas, E., Cienfuegos, M., Zaldívar, D., Pérez-Cisneros, M.: A swarm optimization algorithm inspired in the behavior of the social-spider. Expert Syst. Appl. **40**(16), 6374–6384 (2013)
16. Goldberg, D.E.: Genetic Algorithm in Search Optimization and Machine Learning. Addison-Wesley, Boston (1989)
17. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948, December 1995
18. Geem, Z.W., Kim, J., Loganathan, G.: Music-inspired optimization algorithm harmony search. Simulation **76**, 60–68 (2001)
19. Rashedi, E., Nezamabadi-pour, H., Saryazdi, S.: GSA: a gravitational search algorithm. Inf. Sci. **179**, 2232–2248 (2009)
20. Yang, X.S., Deb, S.: Engineering optimization by cuckoo search. Int. J. Math. Model. Numer. Optim. **1**(4), 339–343 (2010)
21. Pereira, D.R., Pazoti, M.A., Pereira, L.A.M., Rodrigues, D., Ramos, C.O., Souza, A.N., Papa, J.P.: Social-spider optimization-based support vector machines applied for energy theft detection. Comput. Electr. Eng. **49**, 25–38 (2016)
22. Mirjalili, S.Z., Saremi, S., Mirjalili, S.M.: Designing evolutionary feedforward neural networks using social spider optimization algorithm. Neural Comput. Appl. (8), 1919–1928 (2015)
23. Klein, C.E., Segundo, E.H.V., Mariani, V.C., Coelho, L.D.S.: Modified social-spider optimization algorithm applied to electromagnetic optimization. IEEE Trans. Magn. **52**(3), 2–10 (2016)
24. Ouadfel, S., Taleb-Ahmed, A.: Social spiders optimization and flower pollination algorithm for multilevel image thresholding: a performance study. Expert Syst. Appl. **55**, 566–584 (2016)
25. Tawhid, M.A., Ali, A.F.: A simplex social spider algorithm for solving integer programming and minimax problems. Memetic Comput. (In press)
26. Jesus, I., Machado, J.: Application of fractional calculus in the control of heat systems. J. Adv. Comput. Intell. Intell. Inform. **11**(9), 1086–1091 (2007)
27. Machado, J.: Analysis and design of fractional-order digital control systems. SAMS **27**, 107–122 (1997)
28. Liu, L., Pann, F., Xue, D.: Variable-order fuzzy fractional PID controller. ISA Trans. **55**, 227–233 (2015)
29. Shah, P., Agashe, S.: Review of fractional PID controller. Mechatronics **38**, 29–41 (2016)
30. El-Khazali, R.: Fractional-order PIλDμ controller design. Comput. Math. Appl. **66**, 639–646 (2013)
31. Owolabi, K.M.: Robust and adaptive techniques for numerical simulation of nonlinear partial differential equations of fractional order. Commun. Nonlinear SCi Numer. Simul. **44**, 304–317 (2017)
32. Hélie, T.: Simulation of fractional-order low-pass filters. IEEE/ACM Trans. Audio Speech Lang. Process. **22**(11), 1636–1647 (2014)

33. Hwang, C., Leu, J.-F., Tsay, S.-Y.: A note on time-domain simulation of feedback fractional-order systems. IEEE Trans. Autom. Control **47**(4), 625–631 (2002)
34. Podlubny, I.: Fractional Differential Equations. Academic Press, USA (1998)
35. Miller, K.S., Ross, B.: An Introduction to the Fractional Calculus and Fractional Differential Equations. Wiley, USA (1993)
36. Dorcak, L.: Numerical Models for the Simulation of the Fractional-Order Control Systems, UEF-04-94,Kosice, Technical Report, November, (1994)
37. Barbosa, R., Machado, J.A., Silva, M.: Time domain design of fractional differintegrators using least-squares. Signal Process. **86**(10), 2567–2581 (2006)
38. Chen, Y.Q., Vinagre, B., Podlubny, I.: Continued fraction expansion to discretize fractional order derivatives—an expository review. Nonlinear Dyn. **38**(1–4), 155–170 (2004)
39. Vinagre Blas, M., Chen, Y., Petráš, I.: Two direct Tustin discretization methods for fractional-order differentiator/integrator. Frankl. Inst. **340**(5), 349–362 (2003)
40. Jacobs, B.A.: A new Grünwald-Letnikov derivative derived from a second-order scheme. Abstr. Appl. Anal. **2015**, Article ID 952057, 9 pages (2015). https://doi.org/10.1155/2015/952057
41. Scherer, R., Kalla, S.L., Tang, Y., Huang, J.: The Grünwald-Letnikov method for fractional differential equations. Comput. Math. Appl. **62**, 902–917 (2011)
42. Liua, H., Lia, S., Cao, J., Li, G., Alsaedi, A., Alsaadi, F.E.: Adaptive fuzzy prescribed performance controller design for a class of uncertain fractional-order nonlinear systems with external disturbances. Neurocomputing (In Press)
43. Bigdeli, N.: The design of a non-minimal state space fractional-order predictive functional controller for fractional systems of arbitrary order. J. Process Control **29**, 45–56 (2015)
44. Cordón, O., Herrera, F.: A three-stage evolutionary process for learning descriptive and approximate fuzzy-logic-controller knowledge bases from examples. Int. J. Approximate Reasoning **17**(4), 369–407 (1997)
45. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cybern. **SMC-15**, 116–132 (1985)
46. Mamdani, E., Assilian, S.: An experiment in linguistic synthesis with a fuzzy logic controller. Int. J. Man Mach. Stud. **7**, 1–13 (1975)
47. Xu, J.-X., Li, C., Hang, C.C.: Tuning of fuzzy PI controllers based on gain/phase margin specifications and ITAE index. ISA Trans. **35**(1), 79–91 (1996)