Soft Comput (2016) 20:4055–4067
DOI 10.1007/s00500-015-1741-2

METHODOLOGIES AND APPLICATION

Optimal design of fractional-order PID controller for five bar linkage robot using a new particle swarm optimization algorithm

Mohammad Pourmahmood Aghababa[1]

Published online: 17 June 2015

Abstract This paper introduces a new version of the particle swarm optimization (PSO) method. Two basic modifications for the conventional PSO algorithm are proposed to improve the performance of the algorithm. The first modification inserts adaptive accelerator parameters into the original velocity update formula of the PSO which speeds up the convergence rate of the algorithm. The ability of the algorithm in escaping from local optima is improved using the second modification. In this case, some particles of the swarm, which are named the superseding particles, are selected to be mutated with some probability. The proposed modified PSO (MPSO) is simple to be implemented, fast and reliable. To validate the efficiency and applicability of the MPSO, it is applied for designing optimal fractional-order PID (FOPID) controllers for some benchmark transfer functions. Then, the introduced MPSO is applied for tuning the parameters of FOPID controllers for a five bar linkage robot. Sensitivity analysis over the fractional order of the PID controller is also provided. Numerical simulations reveal that the MPSO can optimally tune the parameters of FOPID controllers.

Keywords Particle swarm optimization · Fractional-order PID · Robot · Controller · Sensitivity analysis

1 Introduction

The main goal of an optimization algorithm is to seek values for a set of parameters that maximize/minimize objective functions subject to some constraints. Some of the optimization algorithms are traditional optimization techniques which use exact methods to find the best solution. If an optimization problem is solvable, then the traditional optimization algorithms can discover the global best solution. However, as the search space increases the computation and implementation costs of the exact algorithms are increased. Therefore, when the search space complexity increases, the exact algorithms are slow to find the global optimum. Linear and nonlinear programming, brute force or exhaustive search and divide and conquer methods are some of the exact-based optimization methods.

Calculus provides the tools for finding the optimum of many objective functions. Calculus-based methods can quickly find a single optimum but require a search scheme to find the global optimum. Continuous functions with analytical derivatives are necessary. If there are too many variables, then it is difficult to find all the optimum points. In such algorithms, the gradient of the objective function serves as the compass heading pointing to the steepest downhill path. It works well when the optimum is nearby, but cannot deal with cliffs or boundaries, where the gradient cannot be calculated.

Other optimization algorithms are heuristic algorithms, such as genetic algorithm (GA), particle swarm optimization (PSO), ant colony optimization (ACO), simulated annealing (SA), harmony search (HS), etc. Heuristic algorithms have several advantages compared to the other algorithms as follows (Wang et al. 2005):

1. Heuristic algorithms can be easily implemented.
2. They can be used efficiently in a multiprocessor environment.
3. They are able to deal with both continuous and discrete optimization problems.

Communicated by V. Loia.

✉ Mohammad Pourmahmood Aghababa
m.p.aghababa@ee.uut.ac.ir; m.pour13@gmail.com

[1] Faculty of Electrical and Computer Engineering, Urmia University of Technology, Urmia, Iran

One of the applied and well-known heuristic optimization methods is the particle swarm optimization. The PSO is a population-based searching technique proposed in 1995 (Kennedy and Eberhart 1995) as an alternative to the genetic algorithm. Its development is based on the observations of social behavior of animals such as bird flocking and fish schooling. Compared to GA, PSO has some attractive characteristics. First, PSO has memory which utilizes the knowledge of good solutions retained by all particles, whereas in GA, previous knowledge of the problem is destroyed once the population is updated. Second, PSO has constructive cooperation between particles and particles in the swarm share their information (Hung et al. 2008).

However, trapping in local optima and slow convergence rate are two main weaknesses of the original heuristic PSO. In order to improve the performance of the PSO and to overcome the above-mentioned weaknesses of the PSO, many studies have been performed in the literature. In Eberhart and Shi (2001), a random inertia weight PSO method (RPSO) has been proposed for tracking dynamic systems. Another new version of the PSO which modifies the velocity update formula of the original PSO, is the constriction factor approach PSO (CPSO) (Clerc and Kennedy 2002). In Ratnaweera et al. (2004), a simple PSO with time-varying acceleration coefficients has been proposed. The idea in Ratnaweera et al. (2004) is to have a high diversity for early iterations and a high convergence for late iterations. Meissner et al. (Van den Bergh and Engelbrecht 2002) have introduced a guaranteed convergence PSO which addresses the problem of stagnation and increases local convergence by using the global best particle to randomly search in an adaptively changing radius at every iteration. In Chen et al. (2010), a PSO-based intelligent decision algorithm has been constructed for VLSI floor planning problem. Forecasting of financial time series using a support vector machine optimized by PSO has been carried out in Zhiqiang et al. (2013). Tassopoulos and Beligiannis (2012) have proposed a PSO based parametric optimization method to solve effectively the school timetabling problem. However, since most of the above-mentioned works have been proposed some hybrid models for the improvement of the original PSO, they are either so complex to be implemented in real world situations or they are with high computational complexity.

Proportional–Integral–Derivative (PID) control is one of the earliest control strategies. It has been widely used in the industrial control field. Its widespread acceptability can be recognized by the familiarity with which it is perceived amongst researchers and practitioners within the control community, simple structure and effectiveness of algorithm, relative ease and high speed of adjustment with minimal down-time and wide range of applications where its reliability and robustness produce good control performances (Gaing 2004).

Fractional calculus, with more than 300 years old history, is a classical mathematical idea which allows to arbitrary (non-integer) order differentiation and integration. Although it has a long history, its applications to physics and engineering are only a recent subject of interest. It has been recognized that many systems in interdisciplinary fields can be elegantly described with the help of fractional-order differential equations. On the other hand, fractional order controllers may be employed to achieve feedback control objectives for such systems. Modeling and control topics using the concept of fractional order systems have recently attracted more attentions because of the introduction of useful applications (Podlubny 1999; Aghababa 2014a, b, c, 2015a, b, c, d).

Recently, fractional calculus has been used for designing fractional-order PID (FOPID) controllers. Various studies have demonstrated that the FOPID controllers improve the performance and robustness of the traditional PID controllers (Lee and Chang 2010). However, successful applications of FOPID controllers require the satisfactory tuning of five parameters: the proportional gain, the integrating gain, the derivative gain, the integrating order and the derivative order. Traditionally, these parameters have been determined by a trial and error approach. Manual tuning of FOPID controller is very tedious, time consuming and laborious to implement, especially where the performance of the controller mainly depends on the experiences of design engineers. Of late, some parameter tuning methods have been proposed to reduce the time consumption for determining the five controller parameters. The most well-known tuning method is the Ziegler–Nichols tuning formula (Valerio and Costa 2006), which determines suitable parameters by observing a gain and a frequency on which the plant becomes oscillatory.

More recently, artificial intelligence techniques such as electromagnetism-like algorithm (Lee and Chang 2010), fuzzy logic (Das et al. 2012), PSO and GA approaches (Bingul and Karahan 2012) and artificial bee colony (Rajasekhar et al. 2011) have been applied to improve the performance of the FOPID controllers. A set of tuning rules for standard (integer-order) PID and fractional-order PID controllers has been proposed in Padula and Visioli (2011). In the above-mentioned studies, it has been shown that the intelligent-based approaches provide good solutions in tuning the parameters of the FOPID controllers. However, there are several causes for developing improved techniques to design FOPID controllers. One of them is the important impact it may give because of the general use of the controllers. The other one is the enhancing operation of FOPID controllers that can be result from improved design techniques. Finally, an optimal tuned FOPID controller is more interested in real-world applications.

This paper proposes a modified new version of the standard PSO algorithm as an alternative numerical optimization algorithm. Two main modifications are proposed

for improving the performance of the traditional PSO. The first modification inserts adaptive accelerator parameters into the original velocity update formula of the PSO to speed up the convergence rate of the algorithm. The second modification improves the ability of the original PSO algorithm in escaping from local optima trap. The idea is to introduce some mutated particles, which are named superseding particles, to the swarm to enhance the diversification property of the algorithm. The proposed modified PSO (MPSO) is applied for determining the optimal parameters' values of the FOPID controllers. The problem of designing FOPID controller is first formulated as an optimization problem. Then, the objective function is defined as minimization of four performance indexes, namely the maximum overshoot, the settling time, the rise time and the integral absolute error of step response. Subsequently, an optimal FOPID controller is designed for a five bar linkage manipulator robot using the developed MPSO. The advantages of this methodology are its simplicity in theory and practice as well as its less computation burden, high-quality solution and speedy convergence.

The rest of this paper is organized as follows: In Sect. 2, preliminaries of the fractional derivatives are given. Section 3 deals with the introduction of the proposed modified PSO algorithm. The design procedure of the fractional PID controller for some typical benchmark transfer functions is presented in Sect. 4. In Sect. 5, the modified PSO method is applied for designing a fractional PID controller for a five-bar-linkage manipulator robot. At last, this paper ends with some concluding remarks in Sect. 6.

2 Preliminaries

In what follows, first some basic definitions of fractional calculus are given. Subsequently, the integer-order approximation of the fractional derivatives is presented.

2.1 Fractional calculus

The basic definitions of the fractional calculus are as follows (Podlubny 1999):

Definition 1 The αth-order fractional integration of function $f(t)$ is defined as below.

$$ {}_{t_0}D_t^{-\alpha} f(t) = \frac{1}{\Gamma(\alpha)} \int_{t_0}^{t} \frac{f(\tau)}{(t-\tau)^{1-\alpha}} \mathrm{d}\tau, \tag{1} $$

where $\Gamma(.)$ is the Gamma function.

Definition 2 The Riemann–Liouville fractional derivative of order α of function $f(t)$ is defined as follows:

$$ {}_{t_0}D_t^{\alpha} f(t) = \frac{\mathrm{d}^{\alpha} f(t)}{\mathrm{d}t^{\alpha}} = \frac{1}{\Gamma(m-\alpha)} \frac{\mathrm{d}^m}{\mathrm{d}t^m} \int_{t_0}^{t} \frac{f(\tau)}{(t-\tau)^{\alpha-m+1}} \mathrm{d}\tau, \tag{2} $$

where $m-1 < \alpha \leq m$ and $m \in N$.

Definition 3 The Caputo fractional derivative of order α of a function $f(t)$ is given by

$$ {}_{t_0}D_t^{\alpha} f(t) = \begin{cases} \frac{1}{\Gamma(m-\alpha)} \int_{t_0}^{t} \frac{f^{(m)}(\tau)}{(t-\tau)^{\alpha-m+1}} \mathrm{d}\tau, & m-1 < \alpha < m \\ \frac{\mathrm{d}^m}{\mathrm{d}t^m} f(t), & \alpha = m, \end{cases} \tag{3} $$

where m is the smallest integer number larger than α.

The Laplace transform of the Caputo fractional derivative becomes as follows (Podlubny 1999):

$$ L\{{}_{t_0}D_t^{\alpha} f(t)\} = s^{\alpha} F(p) - \sum_{k=1}^{m-1} s^{\alpha-k-1} f^{(k)}(0), \tag{4} $$

where p is the Laplace operator.

Equation (4) means that, if zero initial conditions are assumed, systems with a dynamic behavior described by differential equations involving fractional derivatives give rise to transfer functions with fractional powers of s.

2.2 Integer-order approximation of fractional derivative

In both simulations and hardware implementations, the most common way of making use of transfer functions involving fractional powers of s is to approximate them with integer-order transfer functions with a similar behavior. So as to completely imitate a fractional transfer function, an integer transfer function would have to contain an infinite number of poles and zeroes. However, it is possible to obtain practical approximations with a finite number of zeroes and poles. One of the well-known approximations has been proposed in Oustaloup et al. (2000) and makes use of a recursive distribution of poles and zeroes. The approximating transfer function is given by

$$ s^{v} \approx k \prod_{n=1}^{N} \frac{1 + (s/\omega_{z,n})}{1 + (s/\omega_{p,n})}, \quad v > 0, \tag{5} $$

where k is a gain and adjusted so that both sides of (5) shall have unit gain at 1 rad/s, and N is the number of poles and zeroes.

Remark 1 Approximation (5) is valid in the frequency range $[\omega_1 \omega_h]$.

Remark 2 The N is chosen beforehand, and the good performance of the approximation strongly depends thereon: low values not only result in simpler approximations, but also cause the appearance of a ripple in both gain and phase behaviors; such a ripple may be practically eliminated increasing N, but the approximation will be computationally heavier.

Frequencies of poles and zeroes in (5) are given by

$$\begin{aligned}
&\omega_{z,1} = \omega_1\sqrt{\eta},\\
&\omega_{p,n} = \omega_{z,n}\upsilon, \quad n = 1,\ldots,N,\\
&\omega_{z,n+1} = \omega_{p,n}\eta, \quad n = 1,\ldots,N-1,\\
&\alpha = (\omega_h/\omega_n)^{\upsilon/N},\\
&\eta = (\omega_h/\omega_1)^{(1-\upsilon)/N}
\end{aligned} \tag{6}$$

The case $\upsilon < 0$ may be dealt with inverting (5). But if $|\upsilon| > 1$ approximations become unsatisfactory; for that reason, it is usual to split fractional powers of s like this:

$$s^{\upsilon} = s^n s^{\delta}, \quad \upsilon = n + \delta, \quad n \in Z, \quad \delta \in [0,1] \tag{7}$$

In this manner, only the latter term has to be approximated.

3 Proposed optimization method

3.1 Standard PSO algorithm

The PSO is a population-based stochastic optimization algorithm modeled based on the simulation of the social behavior of bird flocks. The PSO is a population-based search process where individuals initialized with a swarm of random solutions, referred to as particles. Each particle in the swarm represents a potential solution to the optimization problem, and if the solution is made up of a set of variables, the corresponding particle is a vector of variables. In a PSO system, each particle is flown through the multidimensional search space, adjusting its position in the search space according to its own experience and that of neighboring particles. The particle, therefore, makes use of the best position encountered by itself and that of its neighbors to place itself toward an optimal solution. The performance of each particle is evaluated using a predefined fitness function which encapsulates the characteristics of the optimization problem (Shi and Eberhart 1998).

The core operation of the PSO is the updating formulae of the particles, i.e. the equation of velocity updating and the equation of position updating. The global optimizing model proposed by Shi and Eberhart (1998) is as follows:

$$\begin{aligned}
v_i(t+1) = {} & w \times v_i(t) + \mathrm{RAND} \times C_1 \times (P_{\mathrm{best}}(t) - x_i(t))\\
& + \mathrm{rand} \times C_2 \times (G_{\mathrm{best}}(t) - x_i(t))
\end{aligned} \tag{8}$$

$$x_i(t+1) = x_i(t) + v_i(t+1), \tag{9}$$

where w is the inertia weight factor, $v_i(t)$ is the velocity of particle i at iteration t, $x_i(t)$ is the particle position at iteration t, C_1 and C_2 are two positive constant parameters called acceleration coefficients, RAND and rand are the random functions in the range [0, 1], $P_{\mathrm{best}}(t)$ is the best position of the ith particle and $G_{\mathrm{best}}(t)$ is the best position among all particles in the swarm up to iteration t.

3.2 Modified PSO algorithm

In general, trapping in local optima and the slow convergence rate are two main weaknesses of the original PSO. In this paper, two modifications are proposed to overcome the aforementioned weaknesses of the original PSO.

The slow convergence rate of the standard PSO, which occurs before providing a precise solution, is closely related to the lack of any adaptive time varying accelerators in the velocity updating formula. In Eq. (8), the constants C_1 and C_2 determine the step size of the particle movements through the P_{best} and G_{best}, respectively. Thus, in the original PSO, the algorithm step sizes are about constant and same for all the particles in the swarm. However, when optimizing complex multimodal functions, these constants are not able to meet the algorithm convergence requirements. Therefore, a mechanism which provides more precise acceleration to the algorithm movements is necessary. One mechanism is the use of time varying step sizes which introduces more sensitive and faster movements. To make this, the values of the cost functions in the iterations can be adopted, where in each iteration the value of the cost function can be interpreted as a criterion that gives the relative improvement of the present movement with respect to the previous movement. Moreover, when a new personal best position is found, it means that a better path (solution) in the search space is explored by a particle. Thus, an accelerated movement in the new found direction can quickly get the particle in the good path. Also, the introduction of a new global best position means that the swarm discoveries a better path (solution) in the whole search space. Therefore, more accelerated movements should be done in the new direction. So, the difference between the values of the cost function in the different iterations can be chosen as accelerators of the PSO. Accordingly, the insertion of two time varying multipliers to the original step sizes in Eq. (8) is proposed here. In this case, the time varying accelerators can provide more sensitive and adaptive movements. In this paper, the original velocity updating formulae is modified as follows:

$$\begin{aligned}
v_i(t+1) = {} & w \times v_i(t) + \mathrm{RAND} \times C_1\\
& \times \frac{(f(P_{\mathrm{best}}(t)) - f(x_i(t)))}{f(P_{\mathrm{best}}(t))} \times (P_{\mathrm{best}}(t) - x_i(t)) + \mathrm{rand} \times C_2
\end{aligned}$$

$$\times \frac{(f(G_{\text{best}}(t)) - f(x_i(t)))}{f(G_{\text{best}}(t))} \times (G_{\text{best}}(t) - x_i(t)), \quad (10)$$

where $f(P_{\text{best}}(t))$ is the best fitness function found by ith particle at iteration t and $f(G_{\text{best}}(t))$ is the best objective function found by the swarm up to iteration t. It is assumed that if $f(P_{\text{best}}(t))$ and/or $f(G_{\text{best}}(t))$ is equal to zero, then it is replaced by a small positive constant ε.

Remark 3 The terms $\frac{(f(P_{\text{best}}(t)) - f(x_i(t)))}{f(P_{\text{best}}(t))}$ and $\frac{(f(G_{\text{best}}(t)) - f(x_i(t)))}{f(G_{\text{best}}(t))}$ are named normalized local and global adaptive coefficients, respectively. In each iteration, the former term defines a normalized movement step size in the direction of best position which is found by ith particle and the latter term defines a normalized movement step size in the direction of the best optimum point which have been found by the swarm, yet. In other words, the time varying accelerators decrease or increase the movement step size relative to being close or far from the optimum point, respectively. By means of this method, velocity can be updated adaptively instead of being fixed or changed linearly.

The other weakness of the standard PSO occurs when strongly multi-modal problems are being optimized. In such cases, the PSO algorithm usually suffers from the premature suboptimal convergence (simply premature convergence or stagnation). Sticking in local optima happens when some poor particles attract the swarm prevent further exploration of the search space. According to Angeline (1998), although PSO finds good solutions faster than other evolutionary algorithms, it usually cannot improve the quality of the solutions as the number of iterations is increased. The rationale behind this problem is that the particles converge to a single point, which is on the line between the global best and personal best positions. This point is not guaranteed to be even a local optimum. Proofs can be found in Bergh and Engelbrecht (2002). Another reason for this problem is the fast rate of information flow between particles, resulting in the creation of similar particles (with a loss in diversity) which increases the possibility of being trapped in local minima (Riget and Vesterstrom 2002). This feature prevents the standard PSO from being really of practical interest for a lot of applications. Therefore, an alternative should increase the diversity property of the algorithm to prevent premature convergence and trapping local optimums. To do this, in this paper, the use of a new mutation procedure is proposed as follows: When new positions are determined for the particles of the swarm using Eqs. (9) and (10), γ % of the particles are randomly selected and are replaced by some other particles namely superseding particles with a probability. The superseding particles are generated using a random change in the current particle. If the new particle (i.e. the superseding particle) has a better cost function compared to the current particle's cost function, it is accepted. Otherwise, the superseding particle is accepted with a probability of $e^{-\frac{\Delta f(t)}{f(G_{\text{best}}(t))}}$, where $\Delta f(t)$ is the difference of the cost functions of the superseding and current particles at iteration t. If the superseding particle is rejected, no replacement is done. This process is repeated in each iteration, for all particles.

Remark 4 To save the best solutions, the best particle in the swarm is exempted from the mutation.

Remark 5 In this paper, two main modifications are suggested for the original PSO. The first modification is to add some new accelerator coefficients to the terms into the available velocity formula. From a computational cost point of view and based on the proposed new velocity updating formula (10), it is clear that no extra computational cost is added to the algorithm via this modification (where the values of the accelerator coefficients are already computed in the original PSO, too). The second modification is to introduce the so-called superseding particles into the swarm. In this case, since the number of the superseding particles is small compared to the number of all the particles in the swarm (γ % of all particles maybe selected for mutation where γ has a small value), this modification does not impose a great deal computational cost of to the MPSO algorithm compared to the original PSO.

The pseudo-code of the proposed MPSO algorithm is given in "Appendix".

4 FOPID controller design

4.1 Objective function definition

The FOPID controller is used to improve the dynamic response and to reduce the steady-state error. The transfer function of a FOPID controller is described as

$$G(s) = K_{\text{P}} + K_{\text{I}}/s^{\lambda} + K_{\text{D}}s^{\mu}, \quad (11)$$

where K_{P}, K_{I} and K_{D} are the proportional gain, integral and derivative time constants, respectively, and λ and μ are fractional powers. For designing an optimal FOPID controller, a suitable objective function that represents system requirements should be defined based on some desired specifications. Some typical output specifications in the time domain are the overshoot (M_{p}), rise time (T_{r}), settling time (T_{ss}) and steady-state error (E_{ss}). In general, three kinds of the performance criteria, including the integrated absolute error (IAE), the integral of squared-error (ISE) and the integrated of time-weighted-squared-error (ITSE), are usually considered in the control design under step testing input. It is worth noticing that using different performance indices makes different solutions for the optimal FOPID controllers.

The above-mentioned three integral performance criteria have some advantages and disadvantages. For example, a disadvantage of the IAE and ISE criteria is that their minimization can result in a response with relatively small overshoot but a long settling time. Although the ITSE performance criterion can overcome the disadvantage of the ISE criterion, the derivation processes of the analytical formula are complex and time-consuming (Gaing 2004). The IAE, ISE and ITSE performance criteria formulas are defined as follows:

$$\mathrm{IAE} = \int_0^\infty |r(t) - y(t)|\mathrm{d}t = \int_0^\infty |e(t)|\mathrm{d}t \tag{12}$$

$$\mathrm{ISE} = \int_0^\infty e^2(t)\mathrm{d}t \tag{13}$$

$$\mathrm{ITSE} = \int_0^\infty te^2(t)\mathrm{d}t \tag{14}$$

In this paper, another time domain performance criterion is defined by

$$J_\alpha(K) = \frac{1}{1+\mathrm{e}^{-\alpha}}(T_\mathrm{r} + T_\mathrm{s}) + \frac{\mathrm{e}^{-\alpha}}{1+\mathrm{e}^{-\alpha}}(M_\mathrm{p} + E_\mathrm{ss}), \tag{15}$$

where K is $[K_\mathrm{P}, K_\mathrm{I}, K_\mathrm{D}, \lambda, \mu]$ is the parameter vector and $\alpha \in [-5, 5]$ is the weighting factor. The optimum selection of α depends on the designer's requirements and the characteristics of the plant under control. One can set α to be smaller than 0 to reduce the overshoot and steady-state error. On the other hand, if α is to be set larger than 0 the rise time and settling time are reduced. On the other hand, if α is set to 0, then all the performance criteria (i.e. the overshoot, rise time, settling time and steady-state error) will have the same worth.

4.2 MPSO-based FOPID controller

For designing an optimal FOPID controller, determination of the vector $K = [K_\mathrm{P}, K_\mathrm{I}, K_\mathrm{D}, \lambda, \mu]$ with regard to the minimization of the performance index is the main issue. Here, the minimization process is performed using the proposed MPSO algorithm. For this purpose, step response of the plant is used for computing four performance criteria, including the overshoot (M_p), steady-state error (E_ss), rise time (T_r) and setting time (T_s). At first, the lower and upper bounds of the controller parameters are specified. Then, a swarm of the initial particles is randomly initialized in the specified range. Each particle represents a solution (i.e. controller parameters K) and the corresponding performance index of each particle is evaluated using the step response of the system. Afterward, the main procedure of the proposed heuristic MPSO algorithm starts to work as follows: First, all the parameters of Eq. (10) are determined. Subsequently, new positions of each particle are obtained using Eq. (9). Then, the superseding particles are created. The aforementioned process is repeated until a stopping criterion is satisfied. In this stage, the particle corresponding to the $G_\mathrm{best}(t)$ is designated as the optimal vector K. The flowchart of the design procedure of the MPSO-based FOPID controller is shown in Fig. 1.

4.3 Optimal FOPID controller for typical transfer functions

In order to verify the performance of the proposed MPSO-based FOPID controller, comparative experiments are carried out for the following three typical control plants:

$$G_1(s) = \frac{1}{s^3+6s^2+7s}, \quad G_2(s) = \frac{s^2+2s}{s^4+2s^3+5s^2+s+0.1},$$
$$G_3(s) = \frac{\mathrm{e}^{-0.1s}}{s^2+2s} \tag{16}$$

It is assumed that the suitable ranges of the control parameters are as follows: $K_\mathrm{P} \in [0, 25]$, $K_\mathrm{I} \in [0, 10]$, $K_\mathrm{D} \in [0, 10]$, $\lambda \in [0, 1]$ and $\mu \in [0, 1]$. A population of 20 particles is found to be suitable for providing good solutions. The maximum number of the iterations for all experiments is considered to be 100. Also, α in Eq. (15) is set to 0 (in this case, all the performance criteria have a same merit in the objective function). In order to design the optimal FOPID controller, the MPSO is run 10 times. The other parameters are selected as follows: $\gamma = 15$ and $C_1 = C_2 = 2$.

The performance of the proposed MPSO is compared to the standard PSO (SPSO) (Kennedy and Eberhart 1995), the constriction factor approach PSO (CPSO) (Clerc and Kennedy 2002) and random inertia weight PSO method (RPSO) (Eberhart and Shi 2001). The comparative results for the plants $G_1(s)$, $G_2(s)$ and $G_3(s)$ are summarized in Tables 1, 2 and 3, respectively. It is seen that the proposed MPSO-based FOPID controllers outperform those of the SPSO, RPSO and CPSO obtained controllers. In all tables, one can see that the proposed MPSO gives lesser overshoot compared to the three other methods. On the other hand, although the rise time of the proposed MPSO-based controller is somewhat greater than the rise time of the other methods, both the settling time and steady-state error criteria of the suggested controller are better than those of the other techniques. Nevertheless, one can set the parameter α to a larger value than 0 to result in a short rise time. Moreover, the last column in the three tables shows the computational effort (CE) of the different algorithms consumed to reach the final solution. The CE criterion is defined as the total number of function evaluation done by a method until reaching the

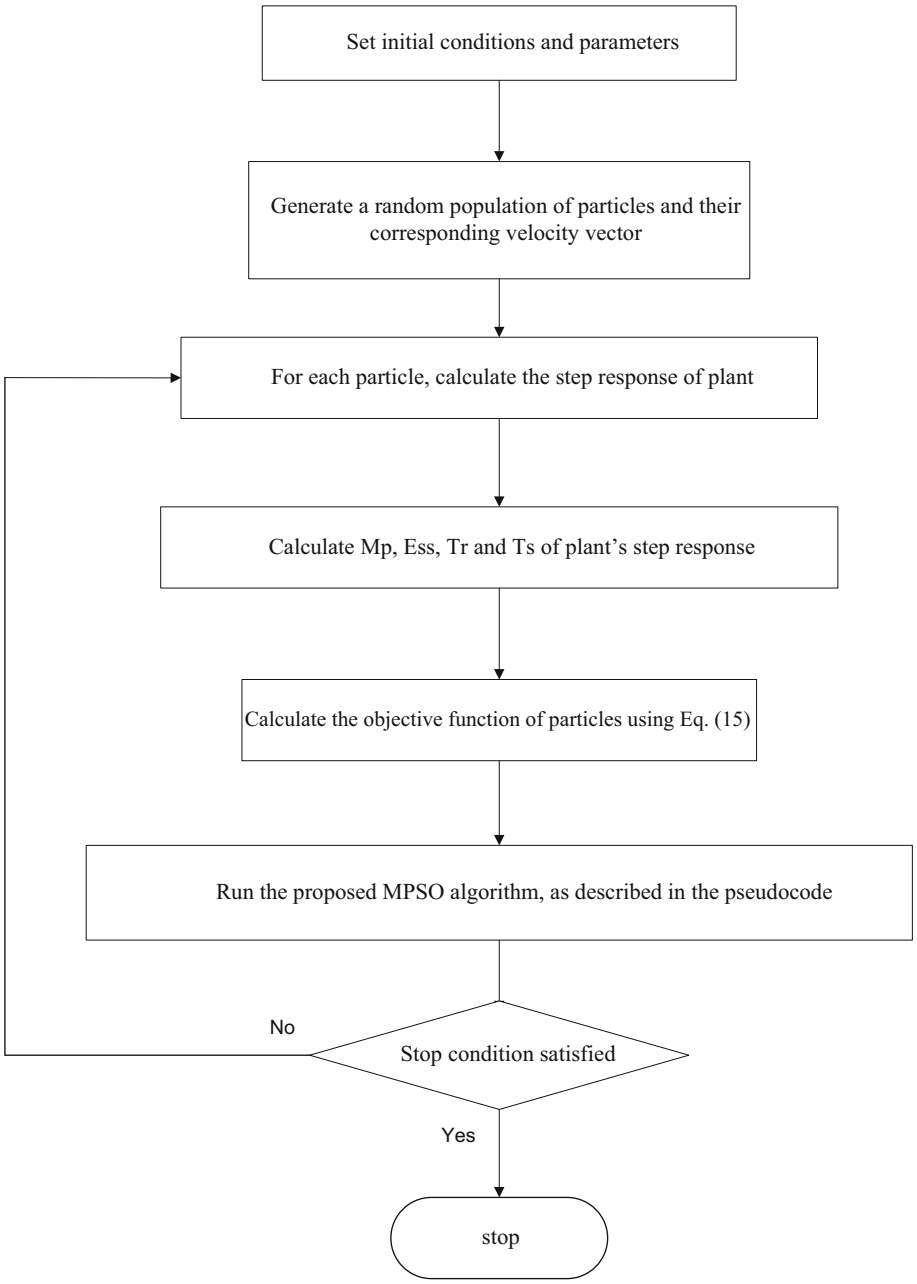

Fig. 1 The design procedure of the MPSO-based FOPID controller

Table 1 Comparative results of the designed different FOPID controllers for the plant $G_1(s)$

Method	K_P	K_D	K_I	λ	μ	M_P	T_r	T_s	E_{ss}	J_0	CE
MPSO	11.18	0.56	6.8	0.95	0.98	0.8237	0.2684	0.5913	0.0000	1.6834	158
SPSO	15.45	5.34	5.62	0.79	0.68	20.7754	0.1240	0.8308	0.0005	21.7307	346
CPSO	13.09	4.89	6.61	0.88	0.82	7.272	0.1541	0.8972	0.0004	8.3237	401
RPSO	11.13	6.18	6.26	0.92	0.84	1.1777	0.1725	1.0500	0.0005	2.4007	377

optimal solution. This criterion is an indication of computational time and convergence speed of the algorithm. In this case, it can be seen that the number of function evaluation in the proposed MPSO method is lesser than those in the other methods. This means that the MPSO algorithm outperforms the SPSO, CPSO and RPSO methods from the computational effort point of view. The unit step responses of the case studies $G_1(s)$, $G_2(s)$ and $G_3(s)$ are depicted in Figs. 2, 3 and 4,

Table 2 Comparative results of the designed different FOPID controllers for the plant $G_2(s)$

Method	K_P	K_D	K_I	λ	μ	M_P	T_r	T_s	E_{ss}	J_0	CE
MPSO	8.5	7.11	7.94	0.99	0.93	9.4054	0.0865	0.6880	0.0013	10.1812	169
SPSO	18.23	7.65	8.90	0.97	0.71	17.1797	0.0778	0.6104	0.0016	17.8695	356
CPSO	19.02	9.28	1.30	0.95	0.85	27.2715	0.0813	0.8855	0.0016	28.2399	422
RPSO	17.81	7.39	5.44	0.93	0.88	19.2398	0.0820	0.8347	0.0016	20.1581	399

Table 3 Comparative results of the designed different FOPID controllers for the plant $G_3(s)$

Method	K_P	K_D	K_I	λ	μ	M_P	T_r	T_s	E_{ss}	J_0	CE
MPSO	6.02	1.33	1.87	0.97	0.94	0.3372	0.1228	0.5525	0.0000	1.0125	183
SPSO	5.93	4.51	3.95	0.95	0.82	20.5750	0.0669	1.1430	0.0018	21.7867	395
CPSO	4.24	0.44	5.51	0.64	0.99	15.1305	0.0886	1.1870	0.0010	16.4071	422
RPSO	7.26	0.19	6.97	0.95	0.83	15.8441	0.0641	0.9109	0.0020	16.8211	388

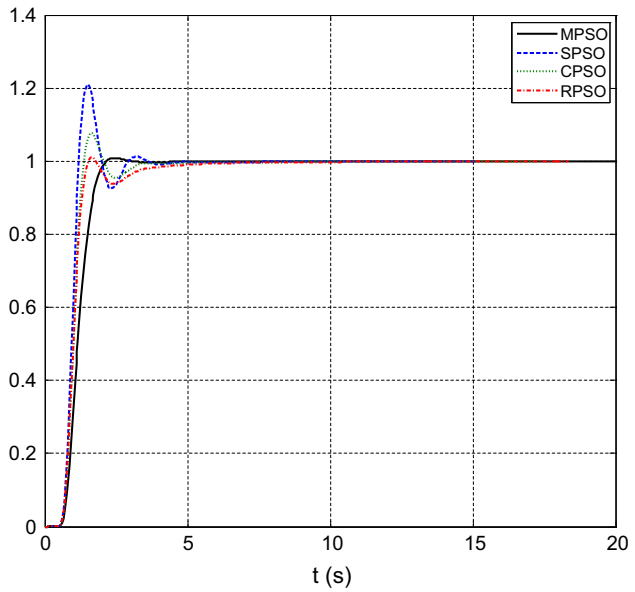

Fig. 2 Step response of $G_1(s)$ with FOPID controller designed by MPSO, SPSO, CPSO and RPSO

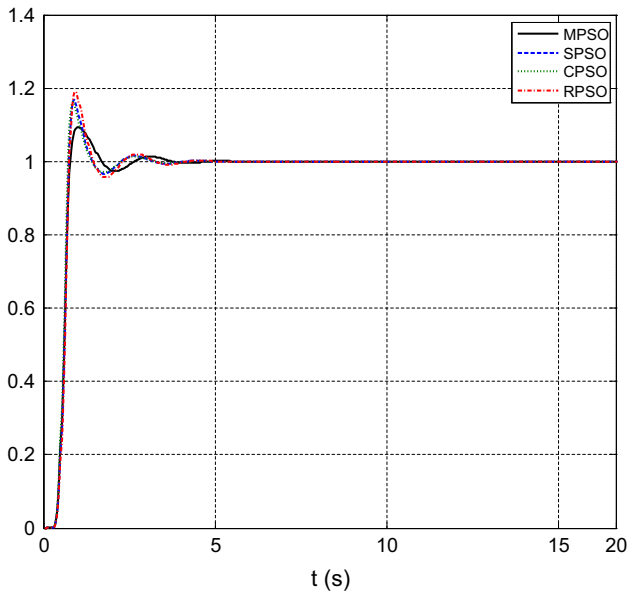

Fig. 3 Step response of $G_2(s)$ with FOPID controller designed by MPSO, SPSO, CPSO and RPSO

respectively. From the simulation results, it can be found that MPSO-based FOPID controllers produce the smooth curve for the output in conjunction with little fluctuation and small overshoot.

5 Design of MPSO-based FOPID controller for five-bar-linkage manipulator robot

5.1 Dynamics of five-bar-linkage manipulator robot

In recent years, there has been a growing interest in the design and control of lightweight robots. Several researchers have studied the modeling and control of a single link flexible beam. Figure 5 shows a typical five-bar-linkage manipulator, where Fig. 6 depicts the five-bar-linkage manipulator schematic where the links form a parallelogram. Let q_i, T_i and I_{h}^i be the joint variable, torque and hub inertia of the ith motor, respectively. Also, let I_i, l_i, l_{c_i} and m_i be the inertia matrix, length, distance to the center of gravity and mass of the ith link, respectively.

The dynamic equations of the manipulator are as follows: (Spong and Vidyasagar 2006).

$$T_1 = \left(M_{11} + I_h^1\right)\ddot{q}_1 + M_{12}\ddot{q}_2 + \frac{\partial M_{12}}{\partial q_2}\dot{q}_2^2 + g\left(m_1 d_{c_1} + m_3 l_{c_3} + m_4 l_1\right)\cos q_1$$

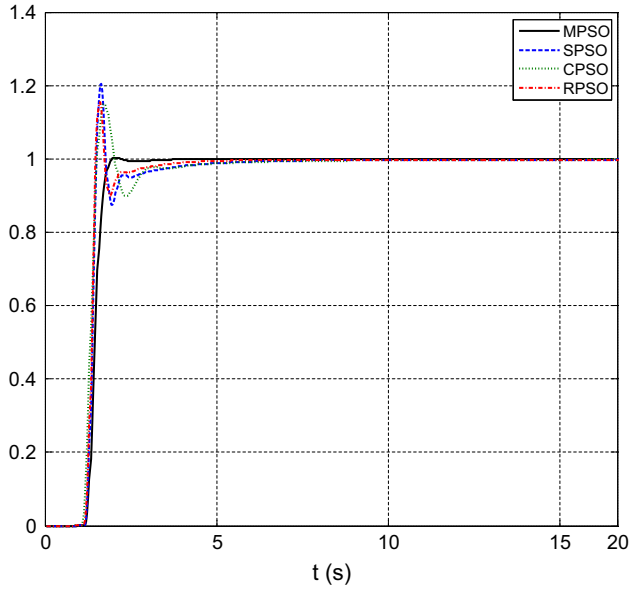

Fig. 4 Step response of $G_3(s)$ with FOPID controller designed by MPSO, SPSO, CPSO and RPSO

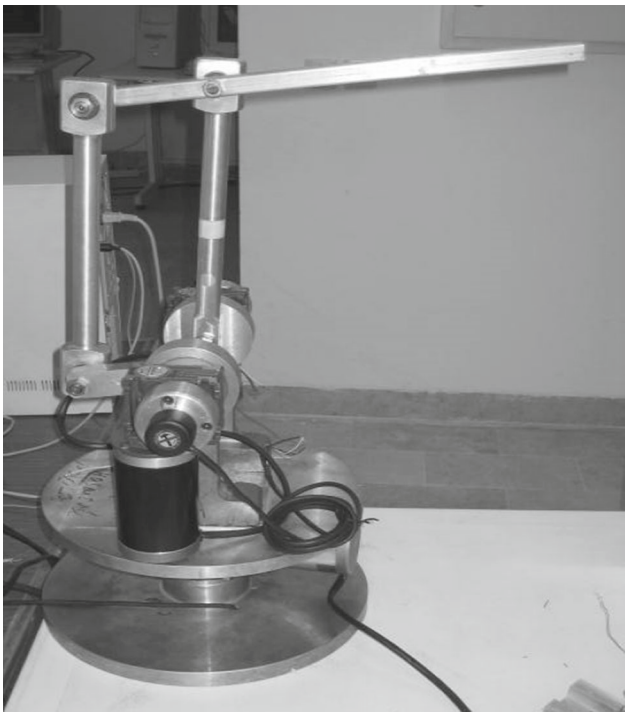

Fig. 5 A typical five-bar-linkage manipulator

$$T_2 = \left(M_{22} + I_h^2\right)\ddot{q}_2 + M_{21}\ddot{q}_1 + \frac{\partial M_{21}}{\partial q_1}\dot{q}_1^2 + g\left(m_1 l_{c_2} + m_3 l_2 + m_4 l_{c_4}\right)\cos q_2, \quad (17)$$

where g is the gravitational constant and $M_{11} = I_{11}^1 + I_{11}^3 + m_1 d_{c_1}^2 + m_3 l_{c_3}^2 + m_4 l_1^2$, $M_{22} = I_{11}^2 + I_{11}^4 + m_2 l_{c_2}^2 + m_3 l_2^2 + m_4 l_{c_4}^2$ and $M_{12} = M_{21} = (m_3 l_{c_3} - m_4 l_{c_4} l_1)\cos(q_1 - q_2)$.

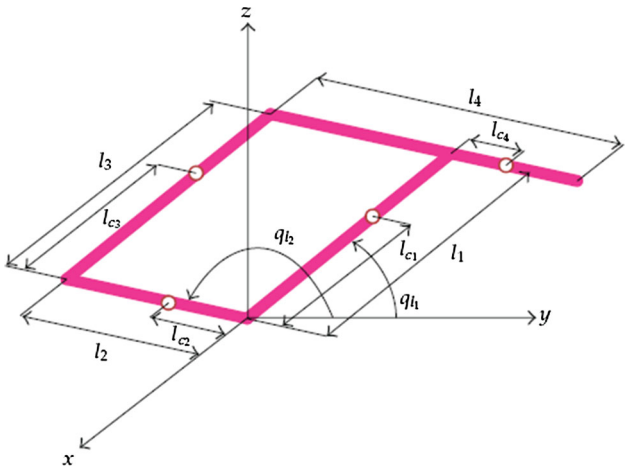

Fig. 6 Planar presentation of the manipulator robot (Badamchizadeh et al. 2010)

Table 4 Five bar linkage robot parameters

Link	Mass (kg)	Length (m)
1	0.288	0.33
2	0.0324	0.12
3	0.3702	0.33
4	0.2981	0.45

From the above-mentioned equations it is revealed that (Badamchizadeh et al. 2010)

$$m_3 l_{c_3} l_2 = m_4 l_{c_4} l_1 \quad (18)$$

Thus, one has $M_{12} = M_{21} = 0$, that is, the matrix of inertia is diagonal and constant. Hence the dynamic equations of this manipulator become

$$T_1 = (M_{11} + I_h^1)\ddot{q}_1 + g\left(m_1 l_{c_1} + m_3 l_{c_3} + m_4 l_1\right)\cos q_1$$
$$T_2 = (M_{22} + I_h^2)\ddot{q}_2 + g\left(m_1 l_{c_2} + m_3 l_2 + m_4 l_{c_4}\right)\cos q_2 \quad (19)$$

Notice that T_1 depends only on q_1 but not on q_2. On the other hand T_2 depends only on q_2, but not on q_1. This discussion helps to explain the popularity of the parallelogram configuration in industrial robots. If the condition (18) is satisfied, then we can adjust the two rotations independently, without worrying about interactions between them.

5.2 MPSO-based FOPID controller for robot

Here, the main goal is to design an optimal MPSO-based FOPID controller for each of the motors of the robot to control their rotations with good performance. In what follows the results are given for one motor, where the same results are also valid for the second motor. Using Eq. (19), the five-bar-linkage manipulator robot is implemented in Matlab and

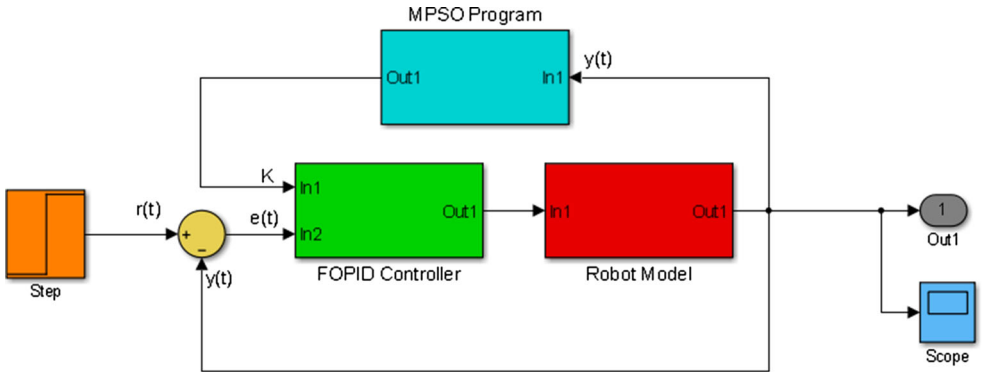

Fig. 7 Simulink diagram of the MPSO-based FOPID controller for robot

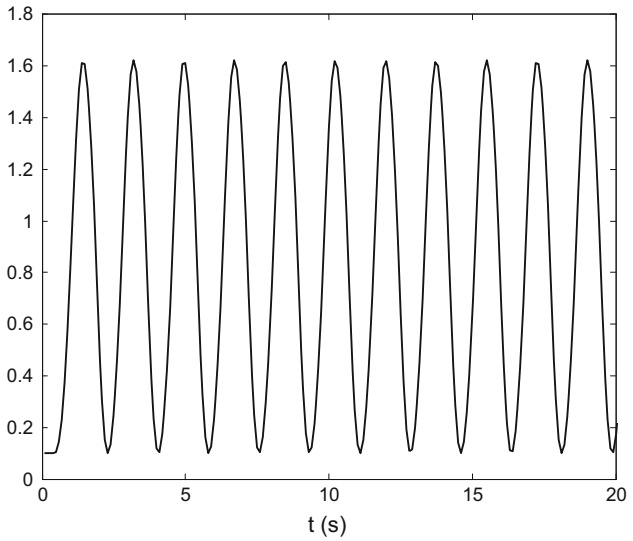

Fig. 8 Response of the robot motor without FOPID controller

Simulink environment. The manipulator specifications are given in Table 4.

The following process is done to determine the optimal values of the parameters of the proposed MPSO-based FOPID controller (i.e. the vector K). First, the MPSO algorithm is initialized, randomly. Each solution $K = [K_P, K_I, K_D, \lambda, \mu]$ is sent to the Simulink block and the values of four performance criteria in the time domain, i.e. M_p, E_{ss}, T_r and T_s, are calculated. Afterwards, the objective function (15) is evaluated for each solution according to the obtained performance criteria. Then, the main procedure of the proposed MPSO algorithm is performed and new solutions are computed. At the end of any iteration, the program checks the stop criterion. When one termination condition is satisfied, the program stops and the latest global best solution is returned as the best solution of K. The Simulink diagram of the proposed procedure is illustrated in Fig. 7.

The parameter setting for the MPSO is as follows: The maximum iteration number is considered equal to 200. The parameter α is set to 0. The swarm is initialized with 30 particles. The other parameters are also selected as follows: $K_P \in [0, 30]$, $K_I \in [0, 5]$, $K_D \in [0, 5]$, $\lambda \in [0, 1]$, $\mu \in [0, 1]$, $\gamma = 10$ and $C_1 = C_2 = 2$.

Figure 8 illustrates the state of the system with no control which is not a stable response. In other words, the system output is oscillatory and requires a suitable control. For comparison, the results of the PSO-based FOPID (Bingul and Karahan 2012), GA-based FOPID (Meng and Xue 2009) and ABC-based FOPID (Rajasekhar et al. 2011) controllers for the robot control are also obtained. The comparative results are summarized in Table 5. It is seen that the proposed MPSO-based FOPID controller outperforms those of the PSO-based FOPID, GA-based FOPID and ABC-based FOPID controllers. It is noted that the last column of Table 5 compares the controllers based on the control signal energy (CSE) where it is computed using $\int_0^\infty |u(t)|dt$. One can see that the MPSO-based FOPID controller needs less control energy compared to the other controllers, except the ABC method. However, the results of the ABC-based FOPID controller for the other criteria, such as the overshoot, the settling

Table 5 Summary of the simulation results of five-bar-linkage robot motor using different methods

Method	K_P	K_I	K_D	λ	μ	M_P	T_r	T_s	E_{ss}	J_0	CE	CSE
MPSO	29.19	3.01	4.59	0.98	0.08	2.4161	0.0819	0.2399	0.0039	2.7418	512	238.668
PSO	25.92	4.65	3.70	0.69	0.56	19.0576	0.0610	0.3392	0.0048	19.4626	844	292.31
GA	26.81	2.65	4.11	0.85	0.92	4.3768	0.0767	0.5650	0.0048	5.0233	798	352.117
ABC	29.09	1.91	2.99	0.91	0.31	19.2400	0.0734	0.4092	0.0043	19.7269	926	225.020

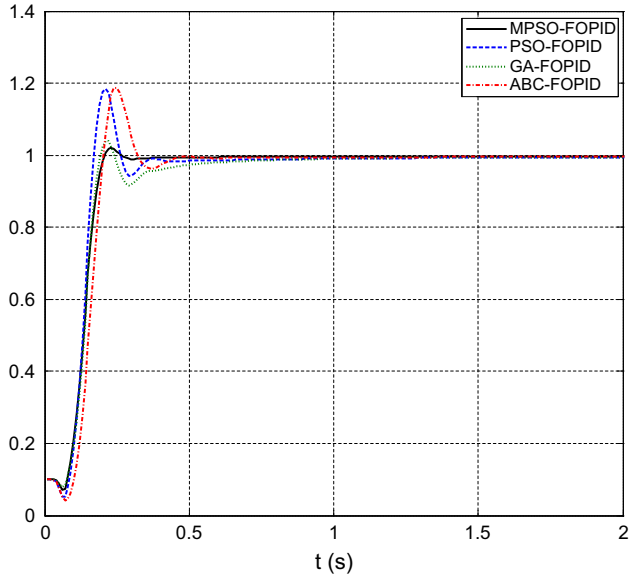

Fig. 9 Step response of the robot motor rotation using MPSO method

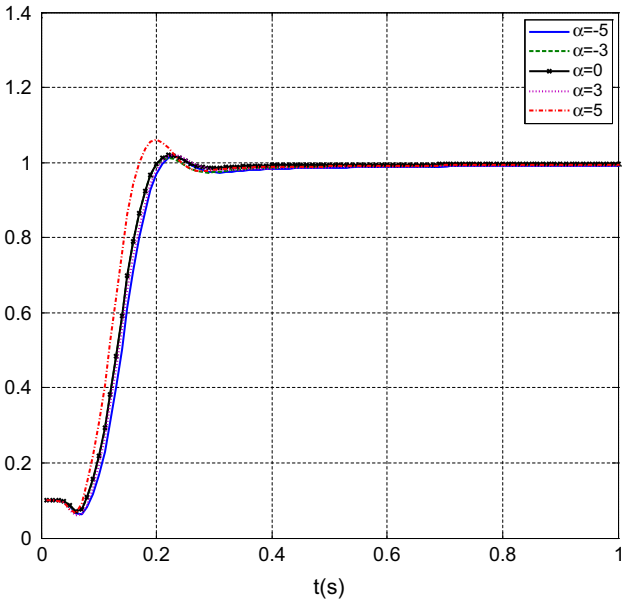

Fig. 10 The step responses of the robot with five typical values of α

time, the steady-state error and the computational effort, are weaker than the corresponding results of the MPSO-based FOPID controller. This means that the controller obtained by the MPSO method is not only with a suitable transient and steady response, but also it is a cheap energy control scheme leading to a more practicable controller in real world applications. Figure 9 reveals the step response of the motor rotation for different FOPID controllers. It is obvious that the proposed MPSO-based FOPID controller produces a smooth and fast output compared to the other methods.

Finally, sensitivity analysis is performed with different values for the parameter α in (15) to effects of the parameter α on the optimal results of the proposed MPSO-based FOPID controller for the robot. To do this, the parameter α is changed from -5 to 5 with a step size of 1. The obtained results are given in Table 6. It can be seen that when $\alpha < 0$, the overshoot and steady-state error are reduced and the rise time and settling time are increased. In contrast, the case $\alpha > 0$ results in small rise time and settling time and enlarges the overshoot and steady-state error. On the other hand, in the case of $\alpha = 0$ all the performance criteria (i.e. the overshoot, rise time, settling time and steady-state error) are of the same worth. Moreover, the step responses of the robot with five typical values of α are shown in Fig. 10. Table 6 and Fig. 10 reveal that the value of the parameter α has little impact on the optimum value of the performance criterion $J_\alpha(K)$ in (15).

6 Conclusions

In this paper, a novel simple modified particle swarm optimization (MPSO) algorithm is proposed to overcome two main shortages of the traditional PSO method, namely slow convergence rate and trapping in local optima. Inserting accelerator parameters into the velocity updating formula

Table 6 Results of sensitivity analysis for the parameter α in (15)

α	K_P	K_I	K_D	λ	μ	M_P	T_r	T_s	E_{ss}	J_0
−5	21.02	3.18	4.94	0.92	0.18	2.3268	0.08261	0.3342	0.0038	2.7474
−4	26.15	3.11	3.99	0.96	0.21	2.3455	0.0823	0.3128	0.0038	2.7444
−3	25.84	3.31	4.10	0.95	0.16	2.3659	0.0822	0.2958	0.0038	2.7477
−2	18.63	3.31	2.27	0.88	0.28	2.3003	0.0824	0.3589	0.0039	2.7455
−1	25.23	2.69	3.56	0.99	0.56	2.3595	0.0851	0.3000	0.0039	2.7485
0	29.19	3.01	4.59	0.98	0.08	2.4161	0.0819	0.2399	0.0039	2.7418
1	30	3.97	5.00	0.92	0.01	2.4325	0.0719	0.2339	0.0040	2.7423
2	23.8	3.20	4.93	0.93	0.12	2.4240	0.0806	0.2353	0.0041	2.7440
3	20.71	3.03	4.28	0.91	0.15	2.4277	0.0803	0.2355	0.0045	2.7480
4	25.72	3.84	4.04	0.89	0.09	2.4315	0.0728	0.2334	0.0045	2.7422
5	26.66	4.19	4.45	0.87	0.05	2.4426	0.0696	0.2315	0.0042	2.7479

and adding a new mutation process to the original PSO are the main proposed modifications. The introduced MPSO is applied for tuning the parameters of the fractional-order PID controllers for some typical transfer functions. Moreover, the proposed optimization method is implemented to design an optimal FOPID controller for a five-bar-linkage robot manipulator. Comparative simulation results reveal that the proposed MPSO can effectively tune the parameters of the FOPID controllers. From an application point of view, the introduced MPOS technique is simple and fast, has a suitable control energy and it can be easily implemented in real-world applications via a microcontroller chip.

Appendix: Pseudocode for MPSO algorithm

Begin;

1. Set initial values such as swarm size, random particles, random velocity vector, maximum number of iterations, etc;
2. For each particle calculate the objective value;
3. Update the global and local best particles and their corresponding objective values;
4. Find the new positions of each particle using Eqs. (9) and (10);
5. Replace up to γ % of the particles by superseding particles;
6. Check if termination condition is true then stop; otherwise, go to step 2;

End.

References

Aghababa MP (2014a) Fractional modeling and control of a complex nonlinear energy supply demand system. Complexity. doi:10.1002/cplx.21533

Aghababa MP (2014b) Chaotic behavior in fractional-order horizontal platform systems and its suppression using a fractional finite-time control strategy. J Mech Sci Tech 28:1875–1880

Aghababa MP (2014c) A Lyapunov based control scheme for robust stabilization of fractional chaotic systems. Nonlinear Dyn 78:2129–2140

Aghababa MP (2015a) A fractional sliding mode for finite-time control scheme with application to stabilization of electrostatic and electromechanical transducers. Appl Math Model. doi:10.1016/j.apm.2015.01.053

Aghababa MP (2015b) Adaptive control of nonlinear complex Holling II predator–prey system with unknown parameters. Complexity. doi:10.1002/cplx.21685

Aghababa MP (2015c) Control of non-integer-order dynamical systems using sliding mode scheme. Complexity. doi:10.1002/cplx.21682

Aghababa MP (2015d) Design of hierarchical terminal sliding mode control scheme for fractional-order systems. IET Sci Meas Technol 9:122–133

Angeline P (1998) Using selection to improve particle swarm optimization. In: Optimization conference on evolutionary computation, Piscataway, pp 84–89

Badamchizadeh MA, Hassanzadeh I, Fallah MA (2010) Extended and unscented kalman filtering applied to a flexible-joint robot with jerk estimation. Discrete Dyn Nat Soc 2010 (article ID 482972)

Bergh FV, Engelbrecht AP (2002) A new locally convergent particle swarm optimiser. In: Proceedings of the IEEE conference on systems, man, and cybernetics, Hammamet. doi:10.1109/ICSMC.2002.1176018

Bingul Z, Karahan O (2012) Fractional PID controllers tuned by evolutionary algorithms for robot trajectory control. Turk J Electr Eng Comput Sci 20:1123–1136

Chen G, Guo W, Chen Y (2010) A PSO-based intelligent decision algorithm for VLSI floor planning. Soft Comput 12:1329–1337

Clerc M, Kennedy J (2002) The particle swarm: explosion, stability, and convergence in a multidimensional complex space. IEEE Trans Evolut Comput 6:58–73

Das S, Pan I, Das S, Gupta A (2012) A novel fractional order fuzzy PID controller and its optimal time domain tuning based on integral performance indices. Eng Appl Artif Intell 25:430–442

Eberhart RC, Shi Y (2001) Tracking and optimizing dynamic systems with particle swarms. In: Proceedings of IEEE congress on evolutionary computation, Seoul, pp 94–97

Gaing Z-L (2004) A particle swarm optimization approach for optimum design of PID controller in AVR system. IEEE Trans Energy Convers 19:384–391

Hung H-L, Huang Y-F, Yeh C-M, Tan T-H (2008) Performance of particle swarm optimization techniques on PAPR reduction for OFDM systems. In: IEEE international conference on systems, man and cybernetics, Singapore, pp 2390–2395

Kennedy J, Eberhart R (1995) Particle swarm optimization. Proc IEEE Int Conf Neural Netw (Perth) 4:1942–1948

Lee CH, Chang FK (2010) Fractional-order PID controller optimization via improved electromagnetism-like algorithm. Expert Syst Appl 37:8871–8878

Meng L, Xue D (2009) Design of an optimal fractional-order PID controller using multi-objective GA optimization. Chinese control and decision conference (CCDC). Guilin 2009:3849–3853

Oustaloup A, Levron F, Mathieu B, Nanot F (2000) Frequency-band complex noninteger differentiator: characterization and synthesis. IEEE Trans Circuits Syst 47:25–39

Padula F, Visioli A (2011) Tuning rules for optimal PID and fractional-order PID controllers. J Process Control 21:69–81

Podlubny I (1999) Fractional differential equations. Academic Press, San Diego

Rajasekhar A, Abraham A, Pant M (2011) Design of fractional order PID controller using sobol mutated artificial bee colony algorithm. In: 11th international conference on hybrid intelligent systems (HIS), Melacca, pp 151–156

Ratnaweera A, Halgamuge SK, Watson HC (2004) Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients. IEEE Trans Evolut Comput 8:240–255

Riget J, Vesterstrom J (2002) A diversity-guided particle swarm optimizer. In: EVALife technical report no. 2002-2

Shi Y, Eberhart R (1998) A modified particle swarm optimizer. In: Proceedings of the IEEE international conference on evolutionary computation. IEEE Press, Piscataway, pp 69–73

Spong MW, Vidyasagar M (2006) Robot dynamics and control. Wiley, New York

Tassopoulos IX, Beligiannis GN (2012) Using particle swarm optimization to solve effectively the school timetabling problem. Soft Comput 16:1229–1252

Valerio D, Costa JS (2006) Tuning of fractional PID controllers with Ziegler–Nichols-type rules. Signal Process 86:2771–2784

Van den Bergh F, Engelbrecht AP (2002) A new locally convergent particle swarm optimizer. Proc IEEE Int Conf Syst Man Cybern 3:94–99

Wang L, Chen K, Ong YS (eds) (2005) Advances in natural computation. Springer, Berlin

Zhiqiang G, Huaiqing W, Quan L (2013) Financial time series forecasting using LPP and SVM optimized by PSO. Soft Comput 17:805–818