A new design method for adaptive IIR system identification using hybrid CPSO and DE

Pedro Lagos-Eulogio, Juan Carlos Seck-Tuoh-Mora, Norberto Hernandez-Romero & Joselito Medina-Marin
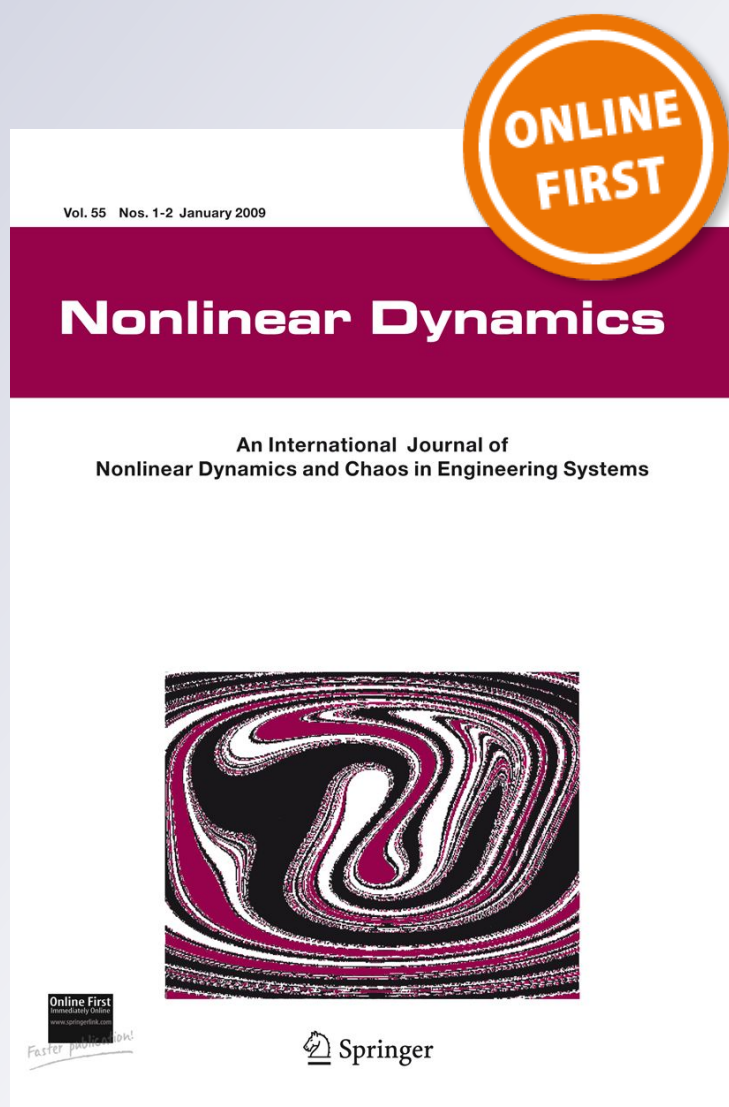
Springer

Springer

CrossMark

ORIGINAL PAPER

A new design method for adaptive IIR system identification using hybrid CPSO and DE

Pedro Lagos-Eulogio ·
Juan Carlos Seck-Tuoh-Mora ·
Norberto Hernandez-Romero ·
Joselito Medina-Marin

Abstract Adaptive infinite impulse response filters have received much attention due to its utilization in a wide range of real-world applications. The design of the IIR filters poses a typically nonlinear, non-differentiable and multimodal problem in the estimation of the coefficient parameters. The aim of the current study is the application of a novel hybrid optimization technique based on the combination of cellular particle swarm optimization and differential evolution called CPSO–DE for the optimal parameter estimation of IIR filters. DE is used as the evolution rule of the cellular part in CPSO to improve the performance of the original CPSO. Benchmark IIR systems commonly used in the specialized literature have been selected for tuning the parameters and demonstrating the effectiveness of the CPSO–DE method. The proposed CPSO–DE method is experimentally compared with two new design methods: the tissue-like membrane system (TMS), the hybrid particle swarm optimization and gravitational search algorithm (HPSO–GSA), the original CPSO-outer and CPSO-inner, and classical implementations of PSO, GSA and DE. Computational results and comparison of CPSO–DE with the other evolutionary and hybrid methods show satisfactory results. The hybridization of CPSO and DE demonstrates powerful estimation ability. In particular, to our knowledge, this hybridization has not yet been investigated for the IIR system identification.

J. C. Seck-Tuoh-Mora (✉) · P. Lagos-Eulogio ·
N. Hernandez-Romero · J. Medina-Marin
Area Academica de Ingenieria, Instituto de Ciencias
Basicas e Ingenieria, Universidad Autonoma del Estado de
Hidalgo, Carr. Pachuca-Tulancingo Km 4.5, 42184 Mineral
de la Reforma, Hgo, Mexico
e-mail: jseck@uaeh.edu.mx

P. Lagos-Eulogio
e-mail: peter_lakes@hotmail.com

N. Hernandez-Romero
e-mail: nhromero@uaeh.edu.mx

J. Medina-Marin
e-mail: jmedina@uaeh.edu.mx

1 Introduction

Adaptive infinite impulse response (IIR) filters have received much attention in recent years due to its current utilization in a wide range of real-world applications such as signal processing, control, communications, parameter identification, image processing and dynamical system modeling [37]. In general, IIR filters are able to model plants more accurately than finite impulse response (FIR) filters [23]. Besides, the IIR filter needs a lesser number of parameters to approximate the dynamical behavior of an unknown plant.

IIR filters are those where the poles and zeros of the transfer function can be adjusted by the coefficient change in the polynomials defining the numerator and

denominator [4]. In contrast to the FIR filter, the IIR filter has the advantage that its output is a function of the previous input and output values.

The design of the adaptive IIR filter poses an interesting challenge involving the estimation of the coefficient parameters by a search algorithm [12,44].

Nevertheless, this design commonly employs a mean square error function (MSE) between the desired response and the output estimated by the filter. MSE is typically nonlinear, non-differentiable and multimodal [24]. Algorithms based on gradient-step methods have been employed in the design of adaptive IIR filters and are able to determine efficiently the optimal solution for unimodal objective functions. However, these can easily fall into local minima and do not converge into the global minimum for multimodal cases [3,32].

One alternative in the design of adaptive IIR filters are evolutionary and metaheuristic algorithms. These methods have been successfully applied in the mathematical optimization of non-differentiable, nonlinear and multimodal functions. Therefore, there is a significant increase of research in the utilization of these algorithms for the design of adaptive IIR filters.

A number of methods have been proposed for the optimization of adaptive IIR filter design using bio-inspired techniques. For instance, genetic algorithms [13,21,26,42,43], cat swarm optimization [27], ant colony optimization [14], modified firefly algorithm [33], particle swarm optimization (PSO) [2,8,11,17, 19] and differential evolution (DE) [20]. From these works, PSO and DE show better performance. In order to avoid a premature convergence into local minima and improve the variety of solutions, some hybrid algorithms have been presented combining different techniques [1,12,28,44].

Cellular particle swarm optimization (CPSO) is a recent proposal combining the features of PSO with the neighborhood behavior of cellular automata (CA) [35]. CPSO has been tested on a variety of optimization problems, for instance in milling process [6], the layout of truss structures [10], and the job shop scheduling problem [7], among others. The obtained results have indicated that compared to the existing evolutionary algorithms, the method shows three advantages: better convergence, stronger robustness and a better balance between exploration and exploitation.

The hybridization study of PSO and CA has shown powerful optimization ability for solving complex problems. In particular, to our knowledge, CPSO has not yet been investigated for the problem of IIR system identification.

Based on the above consideration, the motivation of the current study is the application of a novel hybrid optimization technique based on the combination of PSO, CA and DE (CPSO–DE) in the optimal parameter estimation for adaptive IIR system identification. In particular, DE is used as the evolution rule of the cellular part of CPSO to execute a local search in order to improve each particle of the swarm. This hybridization improves the performance of the original CPSO in the parameter estimation.

The paper is organized as follows: Sect. 2 describes the preliminaries of adaptive IIR filters. Section 3 explains the basics of PSO, DE and the two variants of CPSO. Section 4 presents the hybrid algorithm CPSO–DE proposed in this paper for the optimal design of adaptive IIR filter. Section 5 shows the computational results and comparison of CPSO–DE with other evolutionary and hybrid methods, obtaining satisfactory results. The last section gives the concluding remarks of the paper.

2 Adaptive IIR filter

The transfer function in Z of a IIR filter is defined by:

$$\frac{Y(z)}{U(z)} = \frac{b_0 + b_1 z^{-1} + \cdots + b_N z^{-N}}{1 + a_1 z^{-1} + \cdots + a_M z^{-M}} \tag{1}$$

where $Y(z)$ and $U(z)$ are the output and the input, respectively, of the IIR filter, a_1 a_2 $\ldots$ a_M and b_0 b_1 $\ldots$ b_N are the real coefficients of the polynomials, and N and M express the corresponding order of the numerator and the denominator. The difference equation for Eq. 1 is defined by:

$$y(k) + \cdots + a_M y(k-M) = b_0 u(k) + \cdots + b_N u(k-N) \tag{2}$$

If Eq. 2 is represented using a summation notation:

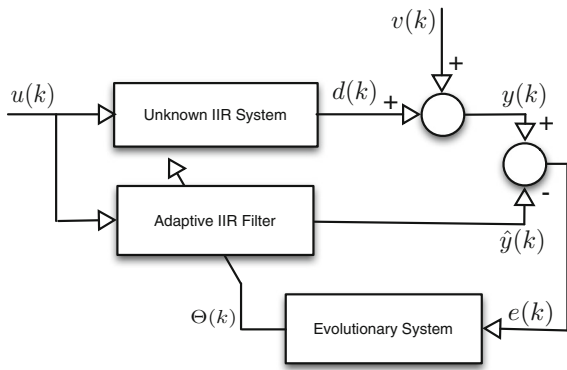$$y(k) = -\sum_{j=1}^{M} a_j y(k-j) + \sum_{i=0}^{N} b_i u(k-i) \tag{3}$$

Fig. 1 Identification of a unknown IIR system by means of an adaptive filter

the difference equation is rewritten as:

$$y(k) = \Theta \phi^T(k) \tag{4}$$

where $\Theta = [-a_1, -a_2, \ldots, -a_M, b_0, b_1, \ldots, b_N]$ and $\phi = [y(k-1), \ldots, y(k-M), u(k), u(k-1), \ldots, u(k-N)]$.

For an adaptive IIR filter, Θ is a vector whose elements must be adjusted in order to obtain a desired output $y(k)$. Thus, the output of the IIR filter is a function of Θ. Figure 1 presents the block diagram for an adaptive IIR filter, which is a classic scheme to identify unknown dynamical systems.

$u(k)$ is the input signal for the unknown system and the adaptive IIR filter, and $d(k)$ is the output signal of the unknown system. $v(k)$ is a noise signal added to $d(k)$; then, $y(k)$ is an output signal of the unknown system with noise. $\hat{y}(k)$ is the estimate output signal from the adaptive IIR filter; $e(k)$ is the deviation between the outputs of both systems; and $\Delta\Theta(k)$ is a rate change in the coefficients of the adaptive IIR filter introduced by the evolutionary algorithm to minimize $e(k)$.

Therefore, the identification of the coefficients in Θ can be handled as an optimization problem:

$$\min E(\Theta) = \min \frac{1}{L} \sum_{k=1}^{N} (d(k) - y(k))^2 \tag{5}$$

where $E(\Theta)$ is the MSE produced by the proposed coefficients for the IIR filter, and L is the number of input samples.

3 Particle swarm optimization and differential evolution

Particle swarm optimization (PSO) is a population-based random search method originally developed from studies of social behavior of birds flocks [16]. Since its inception in 1995 by Eberhart and Kennedy [15,34], it has become one of the most important swarm intelligence-based algorithms. PSO is easy to implement, requires little computational resources and few control parameters [5,9].

PSO starts with an initial population of randomly generated particles. Each particle is represented as a candidate solution to a problem in a D-dimensional space. We denote the ith particle as $X_i = (x_{i1}, x_{i2}, \ldots, x_{iD})$, and its velocity as $V_i = (v_{i1}, v_{i2}, \ldots, v_{iD})$. During the search process, each particle is attracted by its own previous personal best position (P_i) and the global best position discovered by the swarm (P_g). At each time step t, the velocity V_i and the position X_i of particle i are updated as follows:

$$V_i^{t+1} = w^t V_i^t + c_1 r_1 (P_i^t - X_i^t) + c_2 r_2 (P_g^t - X_i^t) \tag{6}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{7}$$

where $i = 1, 2, \ldots, Q$ is the particle index and Q is the population size, c_1 and c_2 are the cognitive and social acceleration parameters, respectively, and r_1 and r_2 are two uniform distributed random numbers within$[0, 1]$. Each velocity V_i is bounded between $V_{\min}$ and $V_{\max}$. Parameter w is the inertial weight used to balance the global and local search [34]. The inertial weight decreases linearly as:

$$w^t = w_{\max} - t \left(\frac{w_{\max} - w_{\min}}{T} \right) \tag{8}$$

where $w_{\max}$ and $w_{\min}$ define the range of the inertial weight, $t = 1, 2, \ldots, T$ is the iteration number and T is a predefined maximum number of iterations.

3.1 Differential evolution

Differential evolution (DE) is a population-based stochastic method for global optimization introduced by Ken Price and Rainer Storn [29,38–41].

Similar to PSO, DE starts with an initial population X randomly generated with Q members searching in a

D-dimensional space. The basic operators of DE are: mutation, crossover and selection.

The most common scheme "DE/rand/1" creates a new solution $O_i = (o_{i1}, o_{i2}, \ldots, o_{iD})$ from original solutions in the population as below :

$$O_i = X_{r_1} + c_3(X_{r_2} - X_{r_3}) \tag{9}$$

where $r_1, r_2, r_3 \in \{1, 2, \ldots, Q\}$ are randomly chosen integers, distinct from each other and also different from i. Factor c_3 is a real value between $[0, 2]$ used to scale the differential variation $(X_{r_2} - X_{r_3})$.

The crossover is introduced to increment the population's diversity by creating a trial vector $H_i = (h_{i1}, h_{i2}, \ldots, h_{iD})$:

$$h_{ij} = \begin{cases} o_{ij}, & \text{if } r_{ij} \leq C_r \text{ or } j = j_{\text{rand}}, \\ x_{ij}, & \text{otherwise}, \end{cases} \tag{10}$$

where r_{ij} is an uniformly distributed random number within $[0, 1]$, $C_r \in [0, 1]$ is the crossover probability factor and $j_{rand} \in \{1, 2, \ldots, D\}$ is a randomly chosen index, which ensures that H_i copies at least one component from O_i. The selection operation is employed to decide which element (o_{ij} or x_{ij}) should be a member of the next generation by

$$X'_i = \begin{cases} O_i, & if \ f(O_i) \leq f(X_i), \\ X_i, & \text{otherwise}, \end{cases} \tag{11}$$

where X'_i is the ith new population individual and $f()$ is the objective function used to compute the fitness values.

3.2 Cellular particle swarm optimization

Cellular particle swarm optimization (CPSO) is a variant of PSO proposed by Shi, Liu, Gao and Zhang [35]. CPSO explores how a particle swarm works similar to cellular automata (CA) [22]. Cellular automata ideas lead to two versions of CPSO, namely, CPSO-inner and CPSO-outer. In the CPSO-inner, each particle is updated by using the information inside the swarm. Meanwhile CPSO-outer exploits the information outside the swarm during the updating process. Particles

in the swarm (smart-cells) communicate with cells outside the swarm in order to improve their fitness value. The cellular automata elements used in the PSO algorithm are:

(a) configuration: (Q particles or smart-cells);
(b) cell space: the set of all cells;
(c) cell state: the particle's information at time t, $S_i^t = [P_i^t, P_g^t, V_i^t, X_i^t]$;
(d) neighborhood: $\Phi(i) = \{i + \delta_j\}, 1 \leq j \leq l$ (l is the neighborhood size).
(e) transition rule: $S_i^{t+1} = \varphi(S_i^t \cup S_{\Phi(i)}^t)$

3.3 CPSO-inner

In CPSO-inner, all information is derived from the cells inside the swarm. The communication between cells is limited locally to neighborhoods according to three typical lattice structures (cubic, trigonal and hexagonal). The lattice contains the same number of grids that the swarm size.

In CPSO-inner, the transition rule defines a new state $S_i(P_\Phi)$ for every cell i as:

$$S_i^{t+1}(P_\Phi) = \varphi \left(f \left(S_i^t (P_i) \right), f \left(S_{i+\delta_1}^t \left(P_{i+\delta_1} \right) \right), \ldots, \right.$$
$$\left. f \left(S_{i+\delta_l}^t \left(P_{i+\delta_l} \right) \right) \right) \tag{12}$$

where in this case φ returns the neighbor with best fitness value.

In order to integrate CA with PSO, the velocity V_i, position X_i and personal best state $S_i(P_i)$ of the ith particle are combined as below:

$$V_i^{t+1} = w^t V_i^t + c_1 r_1 (S_i^t(P_i) - X_i^t)$$
$$+ c_2 r_2 (S_i^{t+1}(P_\Phi) - X_i^t) \tag{13}$$
$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{14}$$

3.4 CPSO-outer

In CPSO-outer, the generalized CA strategy is extended by two types of particles: "smart-cell" and "cell." A smart-cell represents a particle of PSO; on the other hand, a cell represents a candidate solution not sampled yet in the search space. The ith particle's position X_i^t defines the cell state $S_i^t = X_i^t$. Every smart-cell constructs its neighborhood by the next function:

$$\Phi(j) = \begin{cases} X_i^t + \frac{f(P_g^t)}{f(X_i^t)} R_j \circ V_i^t & f(X_i^t) \neq 0, \quad f(P_g^t) \geq 0 \\ X_i^t + \left| \frac{f(X_i^t)}{f(P_g^t)} \right| R_j \circ V_i^t & f(X_i^t) \neq 0, \quad f(P_g^t) < 0 \\ X_i^t + \left(\frac{e^{f(P_g^t)}}{e^{f(X_i^t)}} \right)^2 R_j \circ V_i^t & f(X_i^t) = 0, \quad f(P_g^t) \geq 0 \\ X_i^t + \left(\frac{e^{f(X_i^t)}}{e^{f(P_g^t)}} \right)^2 R_j \circ V_i^t & f(X_i^t) = 0, \quad f(P_g^t) < 0 \end{cases}$$

$$(15)$$

where R_j is the $1 \times D$ vector of direction coefficients composed of D uniform random numbers in $[-1, 1]$ for $1 \leq j \leq l$ and "$\circ$" is the Hadamard product.

At the early iterations, Eq. 15 produces small changes, when the difference between the fitness value of particles with that of the P_g is relatively large. Then, when the particles converge at a point, Eq. 15 provokes larger changes to improve the search. The transition rule in CPSO-outer is similar that the defined in Eq. 12. In this case, the ith particle is replaced by its neighbor with the best fitness values. CPSO-outer gives to particles the ability to make a wise jump, to improve the exploring of the search space in a local competition and enhance the diversity of the swarm.

4 Hybrid cellular particle swarm optimization and differential evolution

In this work, a type of hybridization is presented between CPSO and DE. To achieve this, we are using the key concepts of CPSO-inner and CPSO-outer as base structure, and the idea is to improve the generation of new local information applying DE over the swarm.

CPSO-inner is a kind of local search model, and take the information inside the swarm according to a type of CA lattice structure. For this reason, its optimization capability varies sharply. On the other hand, CPSO-outer has better performance because every particle is able to generate new information from outside the swarm in order to jump from local optima to better positions.

Our proposal (called CPSO–DE) consists of taking the best of both versions, where every particle generates locally new and better information from inside the swarm by an improved transition rule based on DE.

In CPSO–DE, the ith particle's position X_i^t is the cell state S_i^t. We define "smart-cells" and "cells" as well. First, only smart-cells participate in the updating process using the PSO algorithm as follows:

$$V_i^{t+1} = w^t V_i^t + c_1 r_1 (P_i^t - S_i^t) + c_2 r_2 (Pg^t - S_i^t)$$

$$(16)$$

$$S_i^{t+1} = S_i^t + V_i^{t+1} \tag{17}$$

In CPSO-outer, the transition rule generates random cells within an arbitrary radius from a smart-cell. In CPSO–DE, however, the inception of new cells is more complex because each smart-cell uses the stochastic optimization method DE in order to produce its neighborhood.

The operators used for determining the neighborhood of each smart-cell are mutation and crossover. Mutation is achieved by the basic schema according to Eq. 9 that calculates a mutated vector O_i for each particle's neighbor as follows:

$$O_{i,k}^t = S_{r_1}^t + c_3 (S_{r_2}^t - S_{r_3}^t) \tag{18}$$

where $k = 1, 2, \ldots, l$ enumerates every neighbor and l is the neighborhood size. $S_{r_1}^t$, $S_{r_3}^t$ and $S_{r_3}^t$ are updated for each neighbor. The crossover follows Eq. 10 in order to create l trial vectors $H_{i,k}$ combining the information of the current smart-cell with each one of the l mutated vector. The advantage of applying differential evolution is to improve the diversity of neighbors instead of a simple l random variations of the smart-cell, obtaining a better jumping ability provided by the swarm. Finally, Eq. 12 is applied over the trial vectors to update the state of the current smart-cell:

$$S_i^{t+1}(P_\Phi) = \varphi(f(S_i^{t+1}), f(H_{i,1}),$$
$$f(H_{i,2}), \ldots, f(H_{i,l})) \tag{19}$$

The transition rule in Eq. 19 means that the cell in the neighborhood (include the same smart-cell) with best fitness value is chosen for updating the smart-cell state.

Figure 2 illustrates the CPSO–DE mechanism. A two-dimensional space is considered and divided by infinite virtual grids. Every grid contains only one solution. Among these grids, smart-cells are marked by gray circles and available neighbors by circles with gray dots. The smart-cell S_i^t is updated to S_i^{t+1} by the PSO algorithm according to Eqs. 16 and 17. Then, three random smart-cells ($S_{r_1}^t$, $S_{r_2}^t$, $S_{r_3}^t$) are selected to generate the mutated vector $O_{i,k}^t$ with Ec. 18. Finally, the possible neighbor is determined by crossover process; thus, the neighbor can be: $H_{i,k}(\alpha_2, \beta_1)$, $H_{i,k}(\alpha_1, \beta_2)$, $O_{i,k}^t$ and even S_i^{t+1}. The process is repeated l times for each smart-cell.
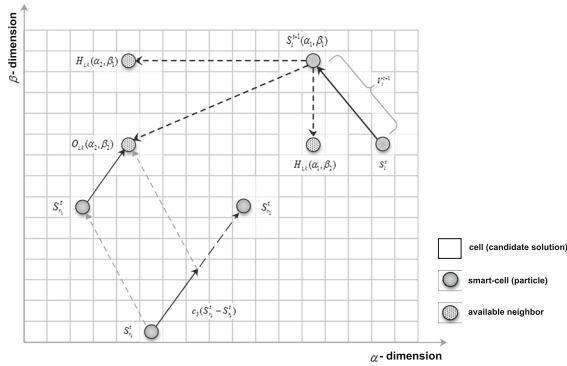
Fig. 2 Schematic drawing for CPSO–DE

The proposed CPSO–DE method is described in Algorithm 1. First, the algorithm sets the control parameters; x^{min}, x^{max}, v_{max}, Q, l, D, w_{min}, w_{max}, c_1, c_2, c_3, C_r and T. Next, the velocity, best local position and state (S) are randomly initialized for each cell as in the PSO method. Then, each particle is evaluated and the best global position is identified. In line 11, the process halts according to the stopping criteria of iteration and convergence. Otherwise, the velocity and state of ith cell is updated in lines 13 and 14, respectively. Later, following the DE method, the neighborhood of size l is generated for each cell. Each neighbor is defined by a mutation vector calculated from 3 random cell states (S_{r_1},S_{r_2} and S_{r_3}) in line 18. The crossover process in line 19 determines the final neighbor. The transition rule inspired in CA behavior is applied in line 27 to determined the new cell state. Finally, the best local and global position are updated in lines 31 and 34, respectively. The process is repeated by each cell and neighbor. This method will be tuned and tested with the identification of IIR filters in the next section.

4.1 Computational complexity analysis

We briefly analyze the time complexity of the proposed method. The identification method consists of four main steps: initialization, local search with differential evolution, swarm evolution with PSO and halting judgment. Note that the CPSO–DE has Q "smart-cells" and each one generates l new particles by differential evolution. Let T be the maximum iteration number. Initialization step contains a single loop (Q times), so its time complexity is $O(Q)$. For local search, there are triple loop (Q, l and T times); therefore, its time complexity is $O(QlT)$. The swarm evolution step con-

---

Algorithm 1 proposed CPSO–DE algorithm

1: //** Initialization
2: Set the control parameters: x^{min}, x^{max}, v_{max}, Q, l, D, w_{min}, w_{max}, c_1, c_2, c_3, C_r, T;
3: **for** $i = 1$ to Q **do**
4: initialize $S_i \in (x_{min}, x_{max})$ randomly;
5: initialize $V_i \in (-v_{max}, v_{max})$ randomly;
6: $P_i = S_i$;
7: **end for**
8: Evaluate each particle;
9: Identify the best global position P_g;
10: //**Loop
11: **while** stopping criterion $t < T$ is not satisfied **do**
12: **for** $i = 1$ to Q **do**
13:   $V_i^{t+1} = w^t V_i^t + c_1 r_1 (P_i^t - S_i^t) + c_2 r_2 (Pg^t - S_i^t)$;
14:   $S_i^{t+1} = S_i^t + V_i^{t+1}$;
15:   //Generate l neighbors using DE method
16:   **for** $k = 1$ to l **do**
17:    // Mutation
18:    $O_{i,k}^t = S_{r_1}^t + c_3(S_{r_2}^t - S_{r_3}^t)$;
19:    **for** $d = 1$ to D **do**
20:     //Crossover
21:     **if** $rand \leq C_r$ or $d = jrand$ **then**
22:      $H_{k,d}^t = O_{k,d}^t$;
23:     **else**
24:      $H_{k,d}^t = S_{i,d}^{t+1}$;
25:     **end if**
26:    **end for**
27:    **if** $fitness(H_k^t) < fitness(S_i^{t+1})$ **then**
28:     $S_i^{t+1} = H_k^t$;
29:    **end if**
30:   **end for**
31:   **if** $fitness(S_i^{t+1}) < fitness(P_i^{t+1})$ **then**
32:    Update P_i^{t+1};
33:   **end if**
34:   **if** $fitness(P_i^{t+1}) < fitness(P_g^{t+1})$ **then**
35:    Update P_g^{t+1};
36:   **end if**
37: **end for**
38: **end while**

tains double loop (Q and T times), so its time complexity is $O(QT)$. For halting step, its time complexity is $O(1)$. Therefore, the time complexity of the proposed method is $O(QlT)$.

5 Simulation results and comparison

5.1 Sensitivity analysis of population and neighborhood size

The performance of CPSO–DE depends mainly on the population and neighborhood sizes. Thus, five different benchmark examples with the actual order and reduced order of the plants have been used for

tuning these parameters. For the sake of simplicity, the possibles values of the population size Q and the number of neighbors l are selected from the set {20, 30, 40, 50, 60, 70, 80, 90, 100} and {5, 10, 15, 20, 25}, respectively. Results are obtained taking the average over 50 independent runs. To overcome the problem caused by the differences of the MSE values for different plants, a normalized function is implemented as follows:

$$\Psi_\kappa(\tau) = \frac{fit_\kappa(\tau) - fit_{\min}(\tau)}{fit_{\max}(\tau) - fit_{\min}(\tau)}, \quad \in [0, 1] \quad (20)$$

where τ denotes different benchmark plants ($\tau =$ Example 1: Case 1,...,Example 5: Case 1, Example 1: Case 2,..., Example 5: Case 2).

κ represents the combination between population and neighborhood; in this way, $\kappa = 2ij + m(1-i) - j$, where i and j represents the group index of different parameters Q and l (seemingly, $i = 1, 2, \ldots, 9$ and $j = 1, 2, \ldots, 5$), respectively. m is the neighborhood group size ($m = 5$). $fit_\kappa(\tau)$ is average MSE fitness under the κth combinations (ith, jth). $fit_{\min}(\tau)$ and $fit_{\max}(\tau)$ denote the minimum and maximum MSE fitness under all κth combination for plant τ, respectively. It is evident from the results that the performance of the CPSO–DE is severely affected in two ways. First, all the populations smaller than 30 individuals ($\kappa \leq 10$) have bad performance, as shown in Figs. 3 and 4 (examples with full order and reduced order, respectively). Second, neighborhood sizes greater than 5 individuals have good performance, as shown in Fig. 4.

In conclusion, the CPSO–DE has better performance with population greater than 30 individuals and more than 5 neighbors. It is clear in Fig. 4 that the best results are obtained when population and neighborhood size are fixed at 100 and 25, respectively. Run time, however, is an important issue; therefore, the first point of convergence in the full-order cases ($\kappa = 12$) is chosen for our computational simulations. Hence, a population size of $Q = 40$ particles and a neighborhood size $l = 10$ are a reasonable choice for the proposed algorithm in the following comparative analysis.

5.2 Parameters settings

In the next experiments, five benchmark IIR systems reported in [18,25,27,36] and [31] have been selected
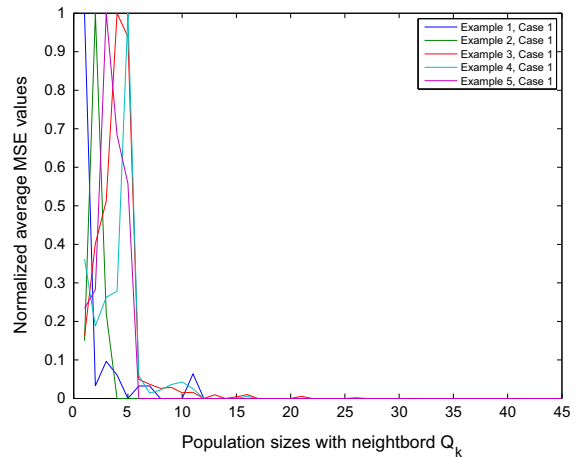
Fig. 3 Normalized average MSE values with different sizes of population and neighborhood for full-order IIR filters
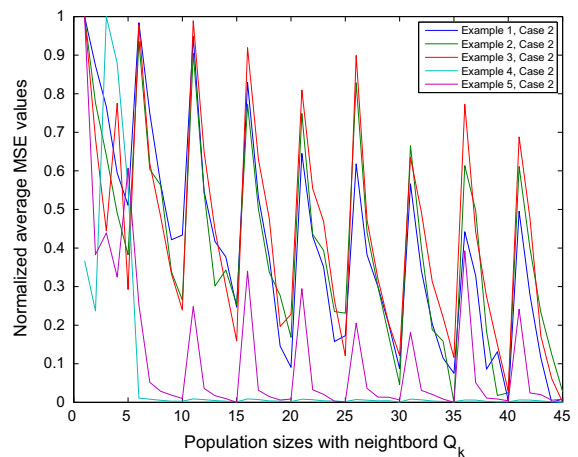
Fig. 4 Normalized average MSE values with different sizes of population and neighborhood for reduced-order IIR filters

to demonstrate the applicability and effectiveness of the CPSO–DE method. For all the cases, the input signal $x(k)$ is a white noise with zero mean, unit variance and uniform distribution, the noise $v(k)$ is absent and the data samples length $L = 200$. For maintaining stability, the search space of adaptive IIR filter coefficient used for each case is restricted in the range $(-2, 2)$ and 50 independent runs are carried out for all algorithms.

To evaluate the effectiveness and efficiency of the proposed CPSO–DE method, it is experimentally compared with two new design methods: the TMS [44], and the HPSO–GSA [12]. The CPSO-outer and CPSO-inner presented in [35] are also applied. The original PSO [34], GSA [30] and DE [40] have been used as

well. Population size $Q = 40$ was chosen for the seven algorithms and a maximum number of iterations $T = 500$ was taken for all the experimental simulations. The parameters of CPSO, TMS and HPSO–GSA were obtained from [35,44] and [12], respectively. For PSO, standard velocity model was used, with learning factors $c_1 = c_2 = 1.49491$ and the inertial weight was linearity decreased from 1.4 to 0.6. DE used the original crossover and selection operations reported in [40] and the mutation "DE/rand/1." The parameters for GSA were configured as recommended by [12] and [30]. Input parameters of CPSO–DE were defines as: $l = 10$, $c_1 = c_3 = 0.5$, $c_2 = 2$, $C_r = 0.9$, $w_{max} = 0.75$, $w_{min} = 0.15$.

All optimization programs were executed on a Mac OS environment using Intel Core Xeon, $3.5\,\text{GHz}$, 32G RAM memory, and the codes were developed using MATLAB 7.14.

5.3 Description of IIR system identification problems and parameter settings

In order to prove our proposal, we have taken five classical benchmark identification problems reported in [18,25,27,36] and [31]. Each unknown plant is estimated by an IIR filter in two modes, with the same order and with reduced order.

Example 1 The transfer function of a second-order plant is defined by Eq. 21. This has been considered in the works, [25,36] and [27].

$$G_p(z) = \frac{0.05 - 0.4z^{-1}}{1 - 1.1314z^{-1} + 0.25z^{-2}} \tag{21}$$

Case 1 The transfer function of a second-order IIR filter to model the second-order plant is defined by Eq. 22

$$G_p(z) = \frac{b_0 - b_1 z^{-1}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{22}$$

Case 2 Equation 23 is a reduced-order filter used for identify the dynamics of the second-order plant in Eq. 21.

$$G_p(z) = \frac{b_0}{1 - a_1 z^{-1}} \tag{23}$$

Example 2 The transfer function of a third-order plant is defined by Eq. 24. This has been considered in [27].

$$G_p(z) = \frac{-0.2 - 0.4z^{-1} + 0.5z^{-2}}{1 - 0.6z^{-1} + 0.25z^{-2} - 0.2z^{-3}} \tag{24}$$

Case 1 The transfer function of a third-order IIR filter to model the third-order plant is defined by Eq. 25.

$$G_p(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3}} \tag{25}$$

Case 2 Equation 26 defines a reduced-order filter used to identify the dynamics of the third-order plant in Eq. 24.

$$G_p(z) = \frac{b_0 + b_1 z^{-1}}{1 - a_1 z^{-1} - a_2 z^{-2}} \tag{26}$$

Example 3 The transfer function of a fourth-order plant is defined by Eq. 27. This has been considered in [27].

$$G_p(z) = \frac{1 - 0.9z^{-1} + 0.81z^{-2} - 0.729z^{-3}}{1 + 0.04z^{-1} + 0.2775z^{-2} - 0.2101z^{-3} + 0.14z^{-4}} \tag{27}$$

Case 1 The transfer function of a fourth-order IIR filter to model the fourth-order plant is defined by Eq. 28.

$$G_p(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}} \tag{28}$$

Case 2 Equation 29 describes a third-order IIR filter used to identify the dynamics of the fourth-order plant in Eq. 27.

$$G_p(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3}} \tag{29}$$

Example 4 The transfer function of a fifth-order plant is defined by Eq. 30. This has been considered in [27] and [18].

$$G_p(z) = \frac{0.1084 + 0.5419z^{-1} + 1.0837z^{-2} + 1.0837z^{-3} + 0.5419z^{-4} + 0.1084z^{-5}}{1 + 0.9853z^{-1} + 0.9738z^{-2} - 0.3854z^{-3} + 0.1112z^{-4} + 0.0113z^{-5}} \tag{30}$$

Case 1 The transfer function of a fifth-order IIR filter to model the fifth-order plant is defined in Eq. 31.

$$G_p(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4} - a_5 z^{-5}} \tag{31}$$

Case 2 Equation 32 describes a reduced-order filter used to identify the dynamics of the fifth-order plant in Eq. 30.

$$G_p(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4}} \tag{32}$$

Example 5 The transfer function of a sixth-order plant is defined by Eq. 33. This has been considered in [27] and [18].

$$G_p(z) = \frac{1 - 0.4z^{-2} - 0.65z^{-4} + 0.26z^{-6}}{1 - 0.77z^{-2} - 0.8498z^{-4} + 0.6486z^{-6}} \tag{33}$$

Case 1 The transfer function of a sixth-order IIR filter to model the sixth-order plant is defined by Eq. 34.

$$G_p(z) = \frac{b_0 + b_2 z^{-2} + b_4 z^{-4} + b_6 z^{-6}}{1 - a_2 z^{-2} - a_4 z^{-4} - a_6 z^{-6}} \tag{34}$$

Case 2 Equation 35 describes a reduced-order filter used to identify the dynamics of the sixth-order plant in Eq. 33.

$$G_p(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3} + b_4 z^{-4} + b_5 z^{-5}}{1 - a_1 z^{-1} - a_2 z^{-2} - a_3 z^{-3} - a_4 z^{-4} - a_5 z^{-5}} \tag{35}$$

5.4 Simulation results and comparison

The results obtained in all the simulations are shown in the next tables and figures in terms of the convergence characteristics, mean square error (MSE) and elapsed times for both full and reduced order of the IIR plants. The actual and estimated parameters are also provided for full-order plants. The best results obtained by the algorithms are bold-faced in the respective tables. Each run stops when an error zero is reached or the maximum number of iterations is computed.

Example 1: A second-order plant. Two cases with the same-order and reduced-order IIR filters are imple-

Table 1 Parameter estimation for Example 1, Case 1

Parameter	Actual values	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
b0	0.0500	**0.0500**	0.0500	0.0500	0.0561	**0.0500**	0.0500	0.0500	0.0500
b1	−0.4000	**−0.4000**	−0.4000	−0.4000	−0.4377	**−0.4000**	−0.4000	−0.4000	−0.4000
a1	−1.1314	**−1.1314**	−1.1314	−1.1314	−1.0285	**−1.1314**	−1.1314	−1.1314	−1.1314
a2	0.2500	**0.2500**	0.2500	0.2500	0.1558	**0.2500**	0.2500	0.2500	0.2500

Table 2 Statistical results of MSE values for Example 1, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	6.1231e−21	1.1603e−12	1.2517e−18	1.7082e−03	**0**	5.4758e−19	**0**	**0**
Worst	4.0588e−17	3.4341e−01	7.4226e−15	5.9335e−02	**0**	1.5217e−03	6.1145e−32	**0**
Average	2.2155e−18	2.7972e−02	3.7011e−16	2.3943e−02	**0**	9.3607e−05	2.0335e−32	**0**
Median	3.7351e−19	1.7342e−04	6.6197e−17	2.2454e−02	**0**	1.4113e−05	1.7304e−32	**0**
SD	6.4647e−18	8.0281e−02	1.1364e−15	1.1162e−02	**0**	2.3627e−04	1.7132e−32	**0**

Table 3 Statistical results of elapsed time (sec) for Example 1, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.4985e+01	1.3109e+01	**3.0125e+00**	3.7504e+00	3.8981e+00	9.5980e+00	3.8703e+01	3.4199e+00
Worst	3.6536e+01	1.3876e+01	**3.1821e+00**	3.9211e+00	4.5659e+00	9.9825e+00	4.2480e+01	3.9368e+00
Average	3.5268e+01	1.3287e+01	**3.1023e+00**	3.8361e+00	4.2736e+00	9.6949e+00	4.0694e+01	3.6193e+00
Median	3.5203e+01	1.3265e+01	**3.1058e+00**	3.8343e+00	4.2754e+00	9.6774e+00	4.0798e+01	3.6198e+00
SD	2.6566e−01	1.1314e−01	3.4323e−02	2.9345e−02	1.5228e−01	**6.5540e−02**	5.9698e−01	1.2880e−01

Table 4 Statistical results of MSE values for Example 1, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	1.3758e−01	1.4694e−01	1.4207e−01	1.4513e−01	1.3730e−01	1.3847e−01	1.3713e−01	**1.3260e−01**
Worst	1.5388e−01	3.1550e−01	1.6186e−01	2.4482e−01	2.5462e−01	1.5550e−01	1.4918e−01	**1.4828e−01**
Average	1.4672e−01	1.8042e−01	1.5489e−01	1.8750e−01	1.5555e−01	1.4926e−01	1.4394e−01	**1.4267e−01**
Median	1.4728e−01	1.5976e−01	1.5537e−01	1.9886e−01	1.5148e−01	1.5001e−01	1.4430e−01	**1.4304e−01**
SD	3.1392e−03	4.6353e−02	4.6679e−03	2.8213e−02	2.1957e−02	3.6831e−03	**2.8522e−03**	3.4111e−03

Table 5 Statistical results of elapsed time (sec) for Example 1, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.1068e+01	1.2292e+01	**2.6652e+00**	3.3500e+00	5.1384e+00	7.8470e+00	3.6710e+01	4.1215e+01
Worst	3.1695e+01	1.2615e+01	**2.7813e+00**	3.5428e+00	5.2856e+00	8.3355e+00	3.7279e+01	4.1889e+01
Average	3.1352e+01	1.2465e+01	**2.7329e+00**	3.4114e+00	5.1880e+00	7.9617e+00	3.7046e+01	4.1570e+01
Median	3.1365e+01	1.2462e+01	**2.7317e+00**	3.4142e+00	5.1768e+00	7.9276e+00	3.7068e+01	4.1588e+01
SD	1.3526e−01	7.2901e−02	2.7718e−02	**2.7178e−02**	3.6489e−02	1.0487e−01	1.3075e−01	1.3715e−01

mented to validate the performance of CPSO–DE. The estimated parameters values of different algorithms for Case 1 are listed in Table 1. From the table, CPSO-inner (CPSO-I), CPSO-outer (CPSO-O), PSO, DE, HPSO–GSA, TMS and CPSO–DE are capable to estimate the coefficients better than GSA. Tables 2 and 3 provide a quantitative assessment of the performance of all the algorithms considered in terms of MSE values and elapsed times. It is clear that DE and CPSO–DE provide the best results with respect to MSE values. However, CPSO–DE requires higher elapsed times in comparison with PSO, and less run times than CPSO-O, CPSO-I, GSA, DE, HPSO–GSA and TMS in terms of average times. This is because that the CPSO–DE requires more steps than the PSO to update solutions.

In Case 2, a first-order IIR filter is used to model the second-order plant. The statistical results of MSE values and elapsed times are shown in Tables 4 and 5. Table 4 shows that the CPSO–DE obtains the best

average results in terms of the MSE values, and the best elapsed times are obtained by PSO.

The convergence behaviors of the best MSE values of two cases using different algorithms are shown in Fig. 5. For Case 1, it is observed that CPSO–DE, DE and TMS obtain the best MSE fitness zero with different number of iterations without any abrupt oscillations. HPSO–GSA shows similar convergence properties; however, they present different solution quality. CPSO-O shows a final convergence value similar to HPSO–GSA but with a slower convergence. The other algorithms are trapped in local minima. Moreover, CPSO–DE rapidly converges to the minimum fitness compared with other algorithms. For Case 2, it can be seen that algorithms fall into local optimum, but CPSO–DE, TMS, DE and HPSO–GSA are able to improve their optima with very similar convergence curves. CPSO-O reaches as well a similar final convergence value but with a slower convergence. Generally,
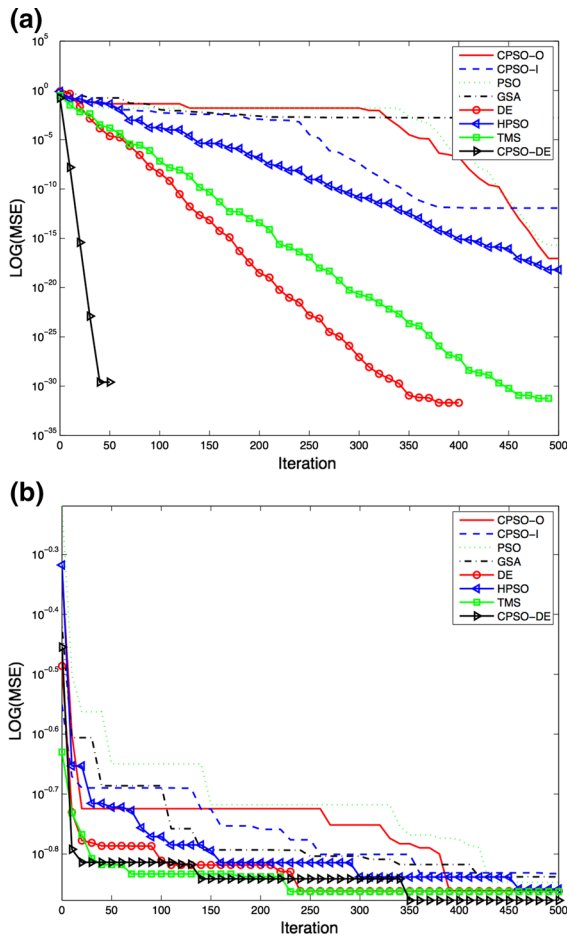
Fig. 5 Convergence behaviors for Example 1: **a** Case 1, **b** Case 2
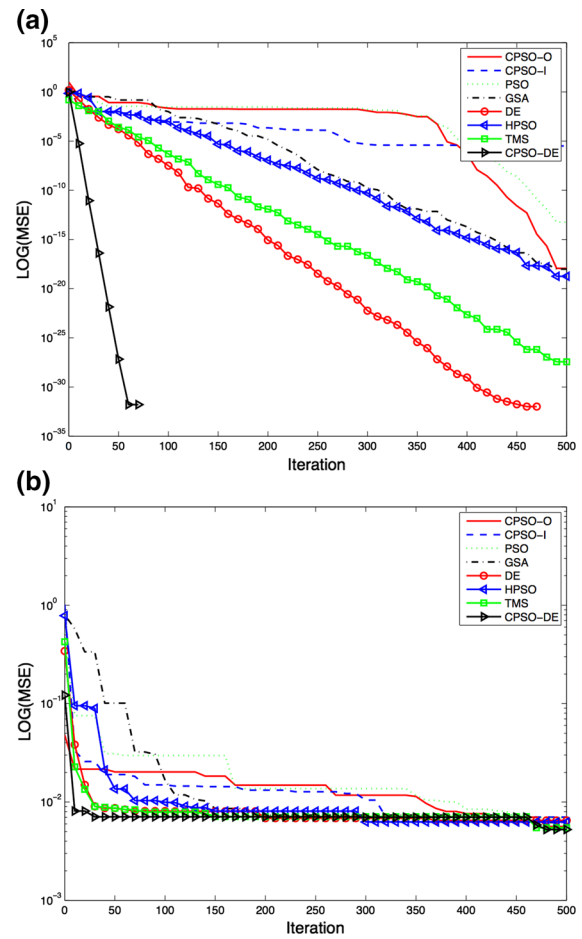
Fig. 6 Convergence behaviors for Example 2: **a** Case 1, **b** Case 2

CPSO–DE is successful in finding the minimum MSE solution among the reported methods and can obtain higher-quality estimated coefficients with better convergence property.

Example 2: A third-order plant. In the first case, the full-order IIR filters is considered. The convergence behaviors in Fig. 6a indicate that CPSO–DE and DE obtain the best MSE fitness zero with different number of iterations without any abrupt oscillations. TMS shows similar convergence properties; nevertheless, with minor quality. Moreover, CPSO–DE rapidly converges to the minimum fitness compared with other algorithms. Table 6 shows that CPSO-O, PSO, GSA, DE, HPSO–GSA, TMS and CPSO–DE are capable to estimate the coefficients better than CPSO-I. Tables 7 and 8 indicate that DE and CPSO–DE give the best results with respect to MSE values. CPSO–DE

demands higher elapsed times in comparison with PSO and GSA and less run times than CPSO-O, CPSO-I, DE, HPSO–GSA and TMS, in terms of average times.

In Case 2, a second-order IIR filter is used to model the plant. The convergence behaviors in Fig. 6b show that TMS and CPSO–DE have very similar convergence curves. The other algorithms obtain almost identical values but with a slower convergence. Table 9 shows that the TMS obtains the best average results in terms of the MSE values, but CPSO–DE is very close to it; besides of having the best MSE value. The best elapsed time in average is obtained by PSO from Table 10. In conclusion, CPSO–DE is successful in finding the minimum or close to minimum MSE values with better convergence property.

Example 3: A fourth-order plant. The first case evaluates the full-order IIR filter. The convergence behav-

Table 6 Parameter estimation for Example 2, Case 1

Parameter	Actual values	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
b0	−0.2000	**−0.2000**	−0.2002	**−0.2000**	**−0.2000**	**−0.2000**	**−0.2000**	**−0.2000**	**−0.2000**
b1	−0.4000	**−0.4000**	−0.4004	**−0.4000**	**−0.4000**	**−0.4000**	**−0.4000**	**−0.4000**	**−0.4000**
b2	0.5000	**0.5000**	0.4967	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**	**0.5000**
a1	−0.6000	**−0.6000**	−0.5951	**−0.6000**	**−0.6000**	**−0.6000**	**−0.6000**	**−0.6000**	**−0.6000**
a2	0.2500	**0.2500**	0.2530	**0.2500**	**0.2500**	**0.2500**	**0.2500**	**0.2500**	**0.2500**
a3	−0.2000	**−0.2000**	−0.1973	**−0.2000**	**−0.2000**	**−0.2000**	**−0.2000**	**−0.2000**	**−0.2000**

Table 7 Statistical results of MSE values for Example 2, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	1.3286e−19	3.2187e−06	3.9259e−16	2.4275e−19	**0**	6.6785e−20	7.8059e−29	**0**
Worst	6.4198e−17	8.5203e−03	7.4410e−03	4.2993e−03	**0**	5.1980e−03	3.3664e−27	**0**
Average	1.0050e−17	2.8657e−03	2.5290e−03	1.2504e−03	**0**	1.2485e−03	6.8046e−28	**0**
Median	5.2314e−18	2.2941e−03	2.7871e−03	9.5153e−04	**0**	8.0069e−04	5.7170e−28	**0**
SD	1.3888e−17	2.4653e−03	1.9392e−03	1.1652e−03	**0**	1.3216e−03	6.7944e−28	**0**

Table 8 Statistical results of elapsed time (sec) for Example 2, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.5150e+01	1.3200e+01	**3.0243e+00**	3.9006e+00	4.8372e+00	1.0992e+01	4.0936e+01	4.7529e+00
Worst	3.5800e+01	1.3461e+01	**3.1933e+00**	4.1964e+00	5.3646e+00	1.1135e+01	4.1566e+01	5.3095e+00
Average	3.5443e+01	1.3329e+01	**3.1129e+00**	4.0098e+00	5.0913e+00	1.1054e+01	4.1295e+01	5.0645e+00
Median	3.5446e+01	1.3337e+01	**3.1136e+00**	3.9837e+00	5.0817e+00	1.1049e+01	4.1303e+01	5.0791e+00
SD	1.5490e−01	6.1022e−02	3.9756e−02	**7.1057e−02**	1.3290e−01	3.2553e−02	1.2514e−01	1.3350e−01

Table 9 Statistical results of MSE values for Example 2, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	5.6373e−03	7.0325e−03	6.9571e−03	6.1915e−03	6.4920e−03	6.2770e−03	5.4848e−03	**5.2547e−03**
Worst	8.1895e−03	1.0810e−02	9.0487e−03	8.4107e−03	8.6226e−03	8.6013e−03	7.5852e−03	**7.3868e−03**
Average	6.9910e−03	8.5101e−03	7.9862e−03	7.5741e−03	7.4975e−03	7.3323e−03	**6.8151e−03**	6.8208e−03
Median	6.9463e−03	8.4443e−03	7.9752e−03	7.6058e−03	7.4198e−03	7.3146e−03	**6.8690e−03**	6.8892e−03
SD	4.4897e−04	7.5835e−04	5.1656e−04	3.9766e−04	4.4614e−04	4.2733e−04	3.8320e−04	**3.5659e−04**

Table 10 Statistical results of elapsed time (sec) for Example 2, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.1407e+01	1.2351e+01	**2.6795e+00**	3.4547e+00	5.1545e+00	9.2299e+00	3.7207e+01	4.2563e+01
Worst	3.3001e+01	1.3744e+01	**2.8642e+00**	3.5860e+00	5.3361e+00	9.4689e+00	3.9060e+01	4.2989e+01
Average	3.1722e+01	1.2563e+01	**2.7656e+00**	3.5196e+00	5.2416e+00	9.3024e+00	3.7508e+01	4.2764e+01
Median	3.1670e+01	1.2529e+01	**2.7674e+00**	3.5194e+00	5.2404e+00	9.2954e+00	3.7473e+01	4.2762e+01
SD	2.5942e−01	2.3855e−01	3.4053e−02	**2.8486e−02**	4.1664e−02	4.0600e−02	2.6007e−01	9.6326e−02
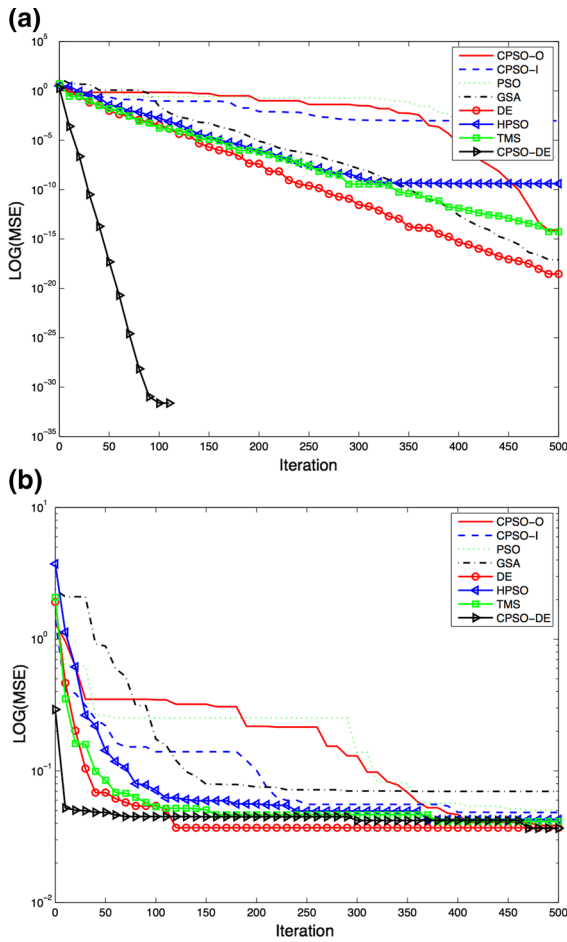
Fig. 7 Convergence behaviors for Example 3: **a** Case 1, **b** Case 2

O improves its convergence in the last iterations. The other algorithms are trapped in local minima. Table 11 shows that CPSO-O, GSA, DE, TMS and CPSO–DE are capable to estimate the coefficients better than the other algorithms. Tables 12 and 13 show that CPSO–DE gives the best results with respect to MSE values requiring higher elapsed times in comparison with PSO, GSA and DE; and less run times than CPSO-O, CPSO-I, HPSO–GSA and TMS in terms of average times.

In Case 2, a third-order IIR filter is used to model the fourth-order plant. The convergence behaviors in Fig. 7b show that algorithms fall into local optimum, but DE, CPSO–DE, TMS and HPSO are able to improve their optima with similar convergence curves. CPSO-O reaches as well a similar final convergence value but with slower convergence. Tables 14 and 15 show that the CPSO–DE obtains the best average MSE value, and the best elapsed time is obtained by PSO.

Example 4: A fifth-order plant. The first case evaluates the full-order IIR filter. The convergence behaviors in Fig. 8a show that CPSO–DE falls into a local minimum, but rapidly improves and obtains the best MSE fitness zero without abrupt oscillations. The other algorithms are trapped in local minima. Table 16 shows that CPSO–DE is able to estimate the coefficients better than all the other algorithms. Tables 17 and 18 show that CPSO–DE provides the best results with respect to MSE values requiring higher elapsed times in comparison with all the other algorithms, except for TMS, because of its complexity.

In Case 2, a fourth-order IIR filter is used to model the fifth-order plant. The convergence behaviors in Fig. 8b show that CPSO–DE rapidly converges to the best MSE fitness value without abrupt oscillations. The other algorithms fall into local optimum, but DE, TMS

iors in Fig. 7a show that CPSO–DE rapidly converges to the minimum MSE fitness zero with the least number of iterations without any abrupt oscillations. DE, GSA and TMS have similar convergence properties; however, they present minor solution quality. CPSO-

Table 11 Parameter estimation for Example 3, Case 1

Parameter	Actual values	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
b0	1.0000	**1.0000**	0.9945	0.9998	**1.0000**	**1.0000**	1.0000	**1.0000**	**1.0000**
b1	−0.9000	**−0.9000**	−0.9366	−0.8997	**−0.9000**	**−0.9000**	−0.9001	**−0.9000**	**−0.9000**
b2	0.8100	**0.8100**	0.7623	0.8099	**0.8100**	**0.8100**	0.8100	**0.8100**	**0.8100**
b3	−0.7290	**−0.7290**	−0.7243	−0.7294	**−0.7290**	**−0.7290**	−0.7290	**−0.7290**	**−0.7290**
a1	0.0400	**0.0400**	0.0163	0.0402	**0.0400**	**0.0400**	0.0400	**0.0400**	**0.0400**
a2	0.2775	**0.2775**	0.2105	0.2778	**0.2775**	**0.2775**	0.2775	**0.2775**	**0.2775**
a3	−0.2101	**−0.2101**	−0.2619	−0.2102	**−0.2101**	**−0.2101**	−0.2101	**−0.2101**	**−0.2101**
a4	0.1400	**0.1400**	0.1208	0.1398	**0.1400**	**0.1400**	0.1400	**0.1400**	**0.1400**

Table 12 Statistical results of MSE values for Example 3, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	2.2965e−17	9.1661e−04	1.7118e−07	8.6246e−19	1.7031e−19	3.9989e−10	1.4150e−15	**0**
Worst	2.4391e−12	1.5951e+00	1.7571e+00	3.6981e−02	6.6484e−14	1.6632e−02	2.3263e−14	**0**
Average	5.6765e−14	1.1222e−01	2.0139e−01	2.2445e−03	1.3475e−15	1.4243e−03	8.9766e−15	**0**
Median	3.5690e−15	3.7672e−02	1.0389e−02	4.0746e−06	6.2446e−18	1.8963e−04	7.8148e−15	**0**
SD	3.4398e−13	2.8361e−01	3.8855e−01	7.2302e−03	9.3997e−15	3.2158e−03	5.7721e−15	**0**

Table 13 Statistical results of elapsed time (sec) for Example 3, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.5366e+01	1.3297e+01	**3.0559e+00**	4.0114e+00	5.5819e+00	1.2373e+01	4.1200e+01	7.6131e+00
Worst	3.6927e+01	1.3601e+01	**3.2097e+00**	4.3891e+00	5.8553e+00	1.3039e+01	4.1805e+01	8.8820e+00
Average	3.5707e+01	1.3400e+01	**3.1384e+00**	4.0803e+00	5.6705e+00	1.2490e+01	4.1475e+01	8.2924e+00
Median	3.5666e+01	1.3397e+01	**3.1416e+00**	4.0727e+00	5.6688e+00	1.2444e+01	4.1474e+01	8.3065e+00
SD	2.3738e−01	5.8139e−02	**3.3445e−02**	5.5445e−02	4.9860e−02	1.4103e−01	1.4367e−01	3.0243e−01

Table 14 Statistical results of MSE values for Example 3, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.8562e−02	4.7852e−02	4.6200e−02	6.9870e−02	3.7049e−02	4.2816e−02	4.1275e−02	**3.6715e−02**
Worst	1.2836e−01	4.1386e−01	8.1652e−01	1.3269e−01	1.3920e−01	1.3240e−01	4.7850e−02	**4.5797e−02**
Average	4.5589e−02	1.2095e−01	1.3822e−01	1.1285e−01	5.2809e−02	9.5739e−02	4.4325e−02	**4.2406e−02**
Median	4.3882e−02	1.1341e−01	1.2809e−01	1.1897e−01	4.8195e−02	9.8409e−02	4.4662e−02	**4.2487e−02**
SD	1.2271e−02	6.8288e−02	1.2975e−01	1.6484e−02	2.0860e−02	2.3702e−02	1.6707e−03	**1.8811e−03**

and CPSO-O are able to improve their optima with a convergence slower than CPSO–DE. Tables 19 and 20 show that the CPSO–DE obtains the best average MSE value, and the best elapsed time is obtained by PSO.

Example 5: A sixth-order plant. The first case evaluates the full-order IIR filter. The convergence behaviors in Fig. 9a show that CPSO–DE rapidly converges to best MSE fitness value, with least number of iterations, without any abrupt oscillations. The other algorithms are trapped in local minima. Table 21 shows that CPSO–DE is able to estimate the coefficients better than all the other algorithms. Tables 22 and 23 show that CPSO–DE provides the best results with respect to MSE values requiring higher elapsed times, except for TMS.

Table 15 Statistical results of elapsed time (sec) for Example 3, Case 2

|	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.1630e+01	1.2489e+01	**2.7075e+00**	3.5661e+00	5.2035e+00	1.0635e+01	3.7574e+01	4.1413e+01
Worst	3.3000e+01	1.2739e+01	**2.9262e+00**	3.7049e+00	5.4134e+00	1.0912e+01	3.9211e+01	4.2141e+01
Average	3.1917e+01	1.2625e+01	**2.7896e+00**	3.6251e+00	5.2829e+00	1.0713e+01	3.7967e+01	4.1818e+01
Median	3.1880e+01	1.2635e+01	**2.7902e+00**	3.6264e+00	5.2724e+00	1.0707e+01	3.7953e+01	4.1821e+01
SD	2.3928e−01	5.8722e−02	3.8261e−02	**3.0179e−02**	4.4839e−02	4.4271e−02	2.4437e−01	1.7772e−01

(a)

(b)
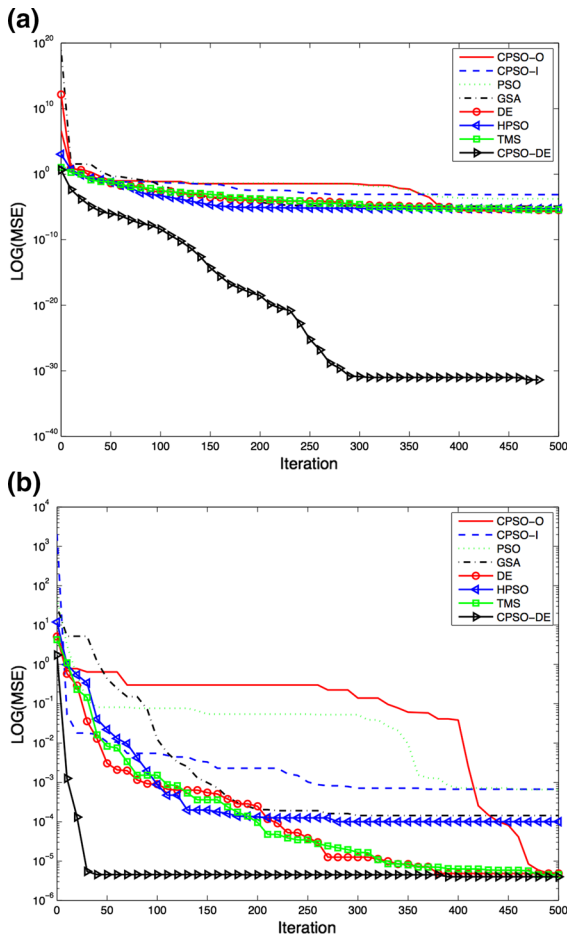
Fig. 8 Convergence behaviors for Example 4: **a** Case 1, **b** Case 2

In Case 2, a fifth-order IIR filter is used to model the sixth-order plant. The convergence behaviors in Fig. 9b show that CPSO–DE falls into local optimum, but is able to improve and converge to the best MSE fitness value. The other algorithms reach final convergence values with different quality. Tables 24 and 25 show that the CPSO–DE obtains the best average results in terms of the MSE values, and the best elapsed times are obtained by PSO.

Generally, CPSO–DE is successful in finding the minimum MSE solution among the reported methods and can obtain higher-quality estimated coefficients with better convergence property.

6 Conclusion

This paper has introduced the use of CPSO–DE hybrid algorithm to develop a novel method for identifying the optimal set of system coefficients with both full-order and reduced-order adaptive IIR filter design.

The proposed method has been compared with seven state-of-the-art evolutionary algorithms. Simulation results show that the proposed method has advantages over PSO, GSA, DE, HPSO–GSA, TMS and both versions of CPSO in terms of the convergence speed and the MSE levels.

Incorporation of CPSO optimization with DE local search brings a remarkable improvement in finding the optimal set of coefficients. The two mechanism can enhance the diversity of solutions and balance exploitation and exploration during the process design.

Table 16 Parameter estimation for Example 4, Case 1

Parameter	Actual values	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
b0	0.1084	0.1083	0.1024	0.1098	0.1082	0.1086	**0.1084**	0.1082	**0.1084**
b1	0.5419	0.4537	0.4196	0.4472	0.4350	0.4798	0.4354	0.4442	**0.5419**
b2	1.0837	0.6768	0.5400	0.5931	0.5857	0.8000	0.5843	0.6341	**1.0837**
b3	1.0837	0.3638	0.1388	0.1066	0.1890	0.5845	0.1810	0.2880	**1.0837**
b4	0.5419	−0.0439	−0.2587	−0.3777	−0.1979	0.1410	−0.2138	−0.1052	**0.5419**
b5	0.1084	−0.0798	−0.1784	−0.2527	−0.1315	−0.0173	−0.1459	−0.0993	**0.1084**
a1	0.9853	0.1730	−0.0649	0.0877	0.0021	0.4155	0.0026	0.0860	**0.9853**
a2	0.9738	0.4848	0.3028	0.0303	0.3266	0.6446	0.3119	0.4349	**0.9738**
a3	0.3864	−0.1761	−0.4071	−0.3036	−0.2880	−0.0033	−0.3054	−0.2367	**0.3864**
a4	0.1112	0.0190	0.0152	−0.1857	−0.0303	0.0547	−0.0350	0.0109	**0.1112**
a5	0.0113	−0.0230	−0.0572	−0.0068	−0.0195	−0.0127	−0.0247	−0.0262	**0.0113**

Table 17 Statistical results of MSE values for Example 4, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	2.1999e−06	7.4836e−04	1.6786e−04	1.2450e−05	3.4353e−06	5.3348e−06	3.8633e−06	**0**
Worst	4.0191e+00	1.1497e+01	4.8704e+00	2.7055e−03	5.1897e−05	6.9642e−03	1.6057e−05	**3.9675e−12**
Average	8.0534e−02	4.3247e−01	6.5404e−01	3.8463e−04	1.2373e−05	1.1671e−03	7.9513e−06	**8.8606e−14**
Median	4.3789e−05	2.3032e−02	2.9080e−03	1.9818e−04	8.5143e−06	4.9839e−04	7.6694e−06	**7.5459e−22**
SD	5.6836e−01	1.7558e+00	1.3100e+00	5.3004e−04	1.0899e−05	1.4853e−03	2.4645e−06	**5.6089e−13**

Table 18 Statistical results of elapsed time (sec) for Example 4, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.5677e+01	1.3325e+01	**3.0990e+00**	4.1641e+00	5.6170e+00	1.4506e+01	4.1450e+01	3.8593e+01
Worst	3.6957e+01	1.3625e+01	**3.4090e+00**	4.3172e+00	5.8445e+00	1.4688e+01	4.3189e+01	4.1342e+01
Average	3.5947e+01	1.3446e+01	**3.1797e+00**	4.2493e+00	5.7003e+00	1.4571e+01	4.1837e+01	4.0532e+01
Median	3.5924e+01	1.3443e+01	**3.1781e+00**	4.2420e+00	5.6917e+00	1.4576e+01	4.1830e+01	4.0543e+01
SD	2.3941e−01	7.6531e−02	4.5911e−02	**3.2870e−02**	3.8589e−02	3.5747e−02	2.6191e−01	3.4734e−01

Table 19 Statistical results of MSE values for Example 4, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	4.9195e−06	6.6119e−04	6.5606e−04	1.4333e−04	4.8512e−06	1.0010e−04	4.5063e−06	**4.0005e−06**
Worst	9.0648e−04	2.7088e+00	3.4265e+00	6.9707e−03	8.8817e−04	4.6017e−03	6.0816e−06	**4.7179e−06**
Average	4.0197e−04	9.2550e−02	2.2836e−01	1.4301e−03	8.8873e−05	1.4014e−03	5.4523e−06	**4.4581e−06**
Median	4.1133e−04	1.4699e−02	6.7667e−03	1.2339e−03	5.7813e−06	1.1563e−03	5.4750e−06	**4.4905e−06**
SD	2.0441e−04	3.9656e−01	6.8656e−01	1.1492e−03	2.3553e−04	1.0041e−03	2.9333e−07	**1.6183e−07**

Table 20 Statistical results of elapsed time (sec) for Example 4, Case 2

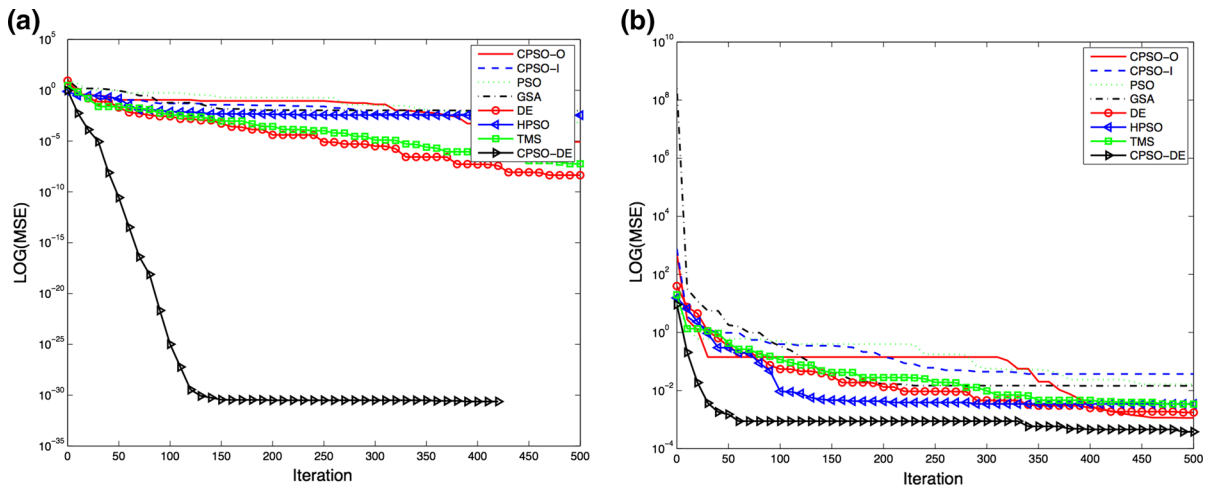|	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.1814e+01	1.2551e+01	**2.7425e+00**	3.7244e+00	5.2250e+00	1.2761e+01	3.8107e+01	4.3212e+01
Worst	3.2758e+01	1.2911e+01	**2.9249e+00**	3.8515e+00	5.4272e+00	1.3329e+01	3.8868e+01	4.3837e+01
Average	3.2173e+01	1.2733e+01	**2.8218e+00**	3.7921e+00	5.3447e+00	1.2861e+01	3.8480e+01	4.3460e+01
Median	3.2166e+01	1.2739e+01	**2.8234e+00**	3.7923e+00	5.3442e+00	1.2823e+01	3.8469e+01	4.3447e+01
SD	1.7153e−01	6.7695e−02	3.9925e−02	**2.3680e−02**	4.4699e−02	1.3830e−01	1.4096e−01	1.3791e−01

(a)

(b)

Fig. 9 Convergence behaviors for Example 5: **a** Case 1, **b** Case 2

Table 21 Parameter estimation for Example 5, Case 1

Parameter	Actual values	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
b0	1.0000	0.9997	0.9819	0.9883	0.9637	**1.0000**	1.0110	**1.0000**	**1.0000**
b2	−0.4000	−0.3605	0.7585	0.3384	0.1156	−0.4002	0.1437	−0.4008	**−0.4000**
b4	−0.6500	−0.6388	−0.1822	−0.3740	0.1040	−0.6501	−0.2681	−0.6502	**−0.6500**
b6	0.2600	0.2441	−0.2180	−0.0264	−0.1936	0.2602	−0.0805	0.2603	**0.2600**
a2	−0.7700	−0.7330	0.4159	−0.0290	−0.2801	−0.7702	−0.2273	−0.7707	**−0.7700**
a4	−0.8498	−0.8512	−0.8254	−0.8462	−0.2639	**−0.8498**	−0.6666	−0.8497	**−0.8498**
a6	0.6486	0.6179	−0.3980	0.0118	−0.3154	0.6488	0.0076	0.6492	**0.6486**

Table 22 Statistical results of MSE values for Example 5, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	8.4183e−06	3.4101e−03	6.6511e−04	9.7450e−03	3.8979e−09	3.4783e−03	2.0003e−08	**0**
Worst	1.2697e−02	4.6958e−01	8.9340e−01	2.3120e−02	1.7511e−02	1.4845e−02	7.2892e−06	**2.9666e−31**
Average	4.6542e−03	4.0849e−02	8.1987e−02	1.3663e−02	1.6406e−03	8.9550e−03	1.0293e−06	**1.2868e−31**
Median	4.6889e−03	1.8678e−02	1.0554e−02	1.3293e−02	8.3921e−07	8.9943e−03	5.7962e−07	**1.7960e−31**
SD	4.1387e−03	8.6818e−02	2.2959e−01	2.7166e−03	4.3299e−03	2.6572e−03	1.2722e−06	**1.2353e−31**

However, the complexity of the CPSO–DE is higher than that of PSO, GSA, DE and near to that of CPSO-inner, CPSO-outer, HPSO–GSA and TMS. Nevertheless, in all full-order cases, the proposed approach shows convergence in a fewer number of iterations than the other algorithms.

The IIR filters considered in this paper are general, further work should imply the modification of the proposed method to design optimal special filters to solve special image/signal processing problems. Other alternative is to extend the application of the CPSO–DE and explore other optimization problems.

Table 23 Statistical results of elapsed time (sec) for Example 5, Case 1

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.6128e+01	1.3395e+01	**3.1739e+00**	3.9982e+00	5.6369e+00	1.1768e+01	4.1717e+01	1.6483e+01
Worst	3.7135e+01	1.4177e+01	**3.3563e+00**	4.4065e+00	5.9044e+00	1.2169e+01	4.2366e+01	4.1233e+01
Average	3.6380e+01	1.3589e+01	**3.2401e+00**	4.0887e+00	5.7374e+00	1.1832e+01	4.2001e+01	3.6894e+01
Median	3.6368e+01	1.3546e+01	**3.2307e+00**	4.0793e+00	5.7278e+00	1.1826e+01	4.2005e+01	4.0731e+01
SD	1.6912e−01	1.5260e−01	**4.0442e−02**	5.5647e−02	6.2841e−02	5.6901e−02	1.4013e−01	5.7977e+00

Table 24 Statistical results of MSE values for Example 5, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	1.1216e−03	3.7221e−02	1.5865e−02	1.4501e−02	1.7703e−03	3.4503e−03	1.8074e−03	**2.9986e−04**
Worst	1.2402e+00	3.6858e+00	7.1201e+00	4.3050e−01	2.3536e−01	3.5547e−01	6.8597e−03	**6.1120e−04**
Average	4.9205e−02	5.7173e−01	9.1083e−01	1.3597e−01	3.6972e−02	5.9204e−02	3.1728e−03	**4.6355e−04**
Median	1.1660e−02	2.8886e−01	1.1191e−01	1.1079e−01	1.1407e−02	3.3001e−02	2.8823e−03	**4.6702e−04**
SD	1.7882e−01	8.7238e−01	1.4734e+00	9.9163e−02	6.3016e−02	6.4647e−02	1.1399e−03	**5.8450e−05**

Table 25 Statistical results of elapsed time (sec) for Example 5, Case 2

	CPSO-O	CPSO-I	PSO	GSA	DE	HPSO–GSA	TMS	CPSO–DE
Best	3.2124e+01	1.2597e+01	**2.7731e+00**	3.8469e+00	5.2689e+00	1.4159e+01	3.8475e+01	4.2211e+01
Worst	3.3608e+01	1.3237e+01	**3.0490e+00**	4.2288e+00	5.6793e+00	1.4694e+01	4.0129e+01	4.2814e+01
Average	3.2369e+01	1.2755e+01	**2.8400e+00**	3.9015e+00	5.3786e+00	1.4224e+01	3.8800e+01	4.2593e+01
Median	3.2341e+01	1.2746e+01	**2.8354e+00**	3.8989e+00	5.3744e+00	1.4213e+01	3.8760e+01	4.2589e+01
SD	2.3730e−01	9.6633e−02	**4.8776e−02**	5.3071e−02	6.0789e−02	7.3758e−02	2.5985e−01	1.2488e−01

References

1. Agrawal, N., Kumar, A., Bajaj, V.: Hybrid method based optimized design of digital iir filter. In: Communications and Signal Processing (ICCSP), International Conference on 2015, pp. 1549–1554. IEEE (2015)
2. Chen, S., Luk, B.L.: Digital iir filter design using particle swarm optimisation. Int. J. Modell. Identif. Control **9**(4), 327–335 (2010)
3. Cuevas, E., Gálvez, J., Hinojosa, S., Avalos, O., Zaldívar, D., Pérez-Cisneros, M.: A comparison of evolutionary computation techniques for iir model identification. J. Appl. Math. **2014** (2014)
4. Diniz, P.S.: Adaptive Filtering: Algorithms ans Practical Implementation. Springer, Berlin (2013)
5. Eberhart, R.C., Shi, Y.: Particle swarm optimization: developments, applications and resources. In: Evolutionary Computation, 2001. Proceedings of the 2001 Congress on, vol. 1, pp. 81–86. IEEE (2001)
6. Gao, L., Huang, J., Li, X.: An effective cellular particle swarm optimization for parameters optimization of a multi-pass milling process. Appl. Soft Comput. **12**(11), 3490–3499 (2012)
7. Gao, L., Li, X., Wen, X., Lu, C., Wen, F.: A hybrid algorithm based on a new neighborhood structure evaluation method for job shop scheduling problem. Comput. Ind. Eng. **88**, 417–429 (2015)
8. Gao, Y., Li, Y., Qian, H.: The design of iir digital filter based on chaos particle swarm optimization algorithm. In: Genetic and Evolutionary Computing, 2008. WGEC'08. Second International Conference on, pp. 303–306. IEEE (2008)
9. Gao, Z., Liao, X.: Rational approximation for fractional-order system by particle swarm optimization. Nonlinear Dyn. **67**(2), 1387–1395 (2012)
10. Gholizadeh, S.: Layout optimization of truss structures by hybridizing cellular automata and particle swarm optimization. Comput. Struct. **125**, 86–99 (2013)
11. Hou, Z., LU, Z.S.: Particle swarm optimization algorithm for iir digital filters design. J. Circuits Syst. **8**(4), 16–20 (2003)

12. Jiang, S., Wang, Y., Ji, Z.: A new design method for adaptive iir system identification using hybrid particle swarm optimization and gravitational search algorithm. Nonlinear Dyn. **79**(4), 2553–2576 (2015)

13. Karaboga, N., Cetinkaya, B.: Design of minimum phase digital iir filters by using genetic algorithm. In: Proceedings of the 6th Nordic signal Processing Symposium-NORSIG, vol. 2004 (2004)

14. Karaboga, N., Kalinli, A., Karaboga, D.: Designing digital iir filters using ant colony optimisation algorithm. Eng. Appl. Artif. Intell. **17**(3), 301–309 (2004)

15. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Neural Networks, 1995. Proceedings IEEE International Conference on, vol. 4, pp. 1942–1948. IEEE (1995)

16. Kennedy, J., Kennedy, J.F., Eberhart, R.C., Shi, Y.: Swarm intelligence. Morgan Kaufmann (2001)

17. Krusienski, D., Jenkins, W.: Adaptive filtering via particle swarm optimization. In: Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on, vol. 1, pp. 571–575. IEEE (2003)

18. Krusienski, D., Jenkins, W.: Design and performance of adaptive systems based on structured stochastic optimization strategies. Circuits Syst. Mag. IEEE **5**(1), 8–20 (2005)

19. Krusienski, D.J., Jenkins, W.K.: Particle swarm optimization for adaptive iir filter structures. In: Evolutionary Computation, 2004. CEC2004. Congress on, vol. 1, pp. 965–970. IEEE (2004)

20. Luitel, B., Venayagamoorthy, G.K.: Differential evolution particle swarm optimization for digital filter design. In: Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on, pp. 3954–3961. IEEE (2008)

21. Ma, Q., Cowan, C.F.: Genetic algorithms applied to the adaptation of iir filters. Signal Process. **48**(2), 155–163 (1996)

22. McIntosh, H.V.: One Dimensional Cellular Automata. Luniver Press, Bristol (2009)

23. Mostajabi, T., Poshtan, J., Mostajabi, Z.: Iir model identification via evolutionary algorithms. Artif. Intell. Rev. **44**(1), 87–101 (2015)

24. Nayeri, M., Jenkins, W.: Alternate realizations to adaptive iir filters and properties of their performance surfaces. IEEE Trans. Circuits Syst. **36**(4), 485–496 (1989)

25. Netto, S.L., Diniz, P.S., Agathoklis, P.: Adaptive iir filtering algorithms for system identification: a general framework. Edu. IEEE Trans. **38**(1), 54–66 (1995)

26. Ng, S., Leung, S., Chung, C., Luk, A., Lau, W.: The genetic search approach. A new learning algorithm for adaptive iir filtering. Signal Process. Mag. IEEE **13**(6), 38–46 (1996)

27. Panda, G., Pradhan, P.M., Majhi, B.: Iir system identification using cat swarm optimization. Expert Syst. Appl. **38**(10), 12671–12683 (2011)

28. Peng, H., Wang, J.: A hybrid approach based on tissue p systems and artificial bee colony for iir system identification. Neural Computing and Applications, pp. 1–11 (2016)

29. Price, K., Storn, R.: Differential evolution: a simple evolution strategy for fast optimization. Dr. Dobbs J. **22**(4), 18–24 (1997)

30. Rashedi, E., Nezamabadi-Pour, H., Saryazdi, S.: Gsa: a gravitational search algorithm. Inf. Sci. **179**(13), 2232–2248 (2009)

31. Saha, S.K., Kar, R., Mandal, D., Ghoshal, S.: Optimal stable iir low pass filter design using modified firefly algorithm. In: Swarm, Evolutionary, and Memetic Computing, pp. 98–109. Springer (2013)

32. Saha, S.K., Kar, R., Mandal, D., Ghoshal, S.P., Mukherjee, V.: A new design method using opposition-based bat algorithm for iir system identification problem. Int. J. Bio Inspir. Comput. **5**(2), 99–132 (2013)

33. Shafaati, M., Mojallali, H.: Modified firefly optimization for iir system identification. J. Control Eng. Appl. Inform. **14**(4), 59–69 (2012)

34. Shi, Y., Eberhart, R.: A modified particle swarm optimizer. In: Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on, pp. 69–73. IEEE (1998)

35. Shi, Y., Liu, H., Gao, L., Zhang, G.: Cellular particle swarm optimization. Inf. Sci. **181**(20), 4460–4493 (2011)

36. Shynk, J.J.: Adaptive iir filtering. ASSP Mag. IEEE **6**(2), 4–21 (1989)

37. Singh, R., Verma, H.: Teaching–learning-based optimization algorithm for parameter identification in the design of iir filters. J. Inst. Eng. India Ser. B **94**(4), 285–294 (2013)

38. Storn, R.: On the usage of differential evolution for function optimization. In: Fuzzy Information Processing Society, 1996. NAFIPS., 1996 Biennial Conference of the North American, pp. 519–523. IEEE (1996)

39. Storn, R., Price, K.: Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, vol. 3. ICSI Berkeley (1995)

40. Storn, R., Price, K.: Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. **11**(4), 341–359 (1997)

41. Storn, R., Price, K.V.: Minimizing the real functions of the icec'96 contest by differential evolution. In: International Conference on Evolutionary Computation, pp. 842–844 (1996)

42. Tang, K.S., Man, K.F., Kwong, S., He, Q.: Genetic algorithms and their applications. Signal Process. Mag. IEEE **13**(6), 22–37 (1996)

43. Tang, K.S., Man, K.F., Kwong, S., Liu, Z.F.: Design and optimization of iir filter structure using hierarchical genetic algorithms. Ind. Electron. IEEE Trans. **45**(3), 481–487 (1998)

44. Wang, J., Shi, P., Peng, H.: Membrane computing model for iir filter design. Inf. Sci. **329**, 164–176 (2016)