Vol. 37, No. 5 ACTA AUTOMATICA SINICA May, 2011

# PSO with Adaptive Mutation and Inertia Weight and Its Application in Parameter Estimation of Dynamic Systems

ALFI Alireza[1]

**Abstract** An important problem in engineering is the unknown parameters estimation in nonlinear systems. In this paper, a novel adaptive particle swarm optimization (APSO) method is proposed to solve this problem. This work considers two new aspects, namely an adaptive mutation mechanism and a dynamic inertia weight into the conventional particle swarm optimization (PSO) method. These mechanisms are employed to enhance global search ability and to increase accuracy. First, three well-known benchmark functions namely Griewank, Rosenbrock and Rastrigrin are utilized to test the ability of a search algorithm for identifying the global optimum. The performance of the proposed APSO is compared with advanced algorithms such as a nonlinearly decreasing weight PSO (NDWPSO) and a real-coded genetic algorithm (GA), in terms of parameter accuracy and convergence speed. It is confirmed that the proposed APSO is more successful than other aforementioned algorithms. Finally, the feasibility of this algorithm is demonstrated through estimating the parameters of two kinds of highly nonlinear systems as the case studies.

**Key words** Particle swarm optimization (PSO), parameter estimation, nonlinear dynamics, inertia weight, adaptive mutation

**DOI** 10.3724/SP.J.1004.2011.00541

In recent years, nonlinear systems have drawn much attention to describe spatiotemporal phenomena in mechanical, electrical and chemical systems. To better understand these phenomena and to bring together experiment and theory, much has been achieved to model these systems with partial differential equations. Although information about the physical properties for many of these systems is available, normally not all dynamical parameters are known and therefore, have to be estimated.

The least-squares method is a basic technique often used for parameters estimation. It has been successfully used to estimate the parameters in static and dynamical systems, respectively[1]. But, the least-squares method is only suitable for the model structure of system having the property of being linear in the parameters. Once the form of model structure is not linear in the parameters, this approach may be invalid. Heuristic algorithms especially with stochastic search techniques seem to be a more hopeful approach and provide a powerful means to solve this problem. They seem to be a promising alternative to traditional techniques, since 1) the objective function′s gradient is not required, 2) they are not sensitive to starting point, and 3) they usually do not get stuck into so called local optima. Because of these, genetic algorithm (GA) was used for identification of nonlinear systems[2−5].

Recently, particle swarm optimization (PSO) algorithm has been becomes available and promising techniques for real world optimization problems[6−7]. Compared to GA, PSO takes less time for each function evaluation as it does not use many of GA operators like mutation, crossover and selection operator[8]. Due to the simple concept, easy implementation and quick convergence, nowadays PSO has gained much attention and wide applications in different fields[9]. But only a couple of applications opted for PSO in nonlinear parameter estimation[10−13]. Authors of these papers showed that PSO is a feasible approach to parameter estimation of nonlinear systems. In [10, 12], a conventional PSO algorithm was used. In [11] a simple mutation without adaptation mechanism is incorporated into PSO. Although PSO has shown some important advances by providing high speed of convergence in specific problems, it exhibits some critical shortages. First, it sometimes is easy to be trapped in local optimum. Second, the convergence rate decreases considerably in the later period of evolution; when reaching a near optimal solution, the algorithm stops optimizing, and thus the achieved accuracy of algorithm is limited[14].

Motivated by the aforementioned researches, the goal of this paper is to present an adaptive particle swarm optimization (APSO) algorithm for unknown parameter estimation of nonlinear systems. Two novel modifications are incorporated into the conventional PSO scheme. First, an adaptive mutation mechanism is introduced to enhance the global search ability and convergence speed of PSO. In this mutation mechanism, when a particle is chosen to mutate, a Gaussian random disturbance is added to its current position. This disturbance has a variable step size which dynamically decreases according to current best solution fitness. Second, a dynamic inertia weight is introduced to improve the accuracy of PSO. The inertia weigh is set as a function of current best solution fitness. If the fitness of current best solution does not improve significantly, the inertia weight decreases slowly since it still needs to globally explore the search space. On the other hand if the fitness of current best solution improve significantly, it decreases fast to facilitate finer local explorations because the algorithm reaches a near optimum solution.

The results are compared to those obtained by real-coded GA and nonlinearly decreasing weight PSO (NDWPSO). It has been demonstrated that APSO has better performance than GA and NDWPSO in solving the parameter estimation problem of nonlinear systems. In the rest of this paper whenever it is referred as "GA", it means "real-coded GA".

## 1 Nonlinear system estimation

If we do not have a priori knowledge about the real system, then structure identification becomes a difficult problem and we have to select the structure by trial and error. Fortunately, we know a great deal about the structures of most engineering systems and industrial processes; usually it is possible to derive a specific class of models that can best describe the real system. As a result, the system identification problem is usually reduced to that of parameter

Manuscript received August 18, 2010; accepted January 22, 2011

1. Faculty of Electrical and Robotic Engineering, Shahrood 36199-95161, Iran

estimation.

In order to explore the problem of parameter estimation in this paper, the following $n$-dimensional nonlinear system is considered:

$$\dot{X} = F(X, X_{\circ}, \theta) \tag{1}$$

where $X = [x_1, x_2, \cdots, x_n]^{\rm T} \in \mathbf{R}^n$ is the state vectors, $X_{\circ}$ denotes the initial states, $\theta = [\theta_1, \theta_2, \cdots, \theta_n]^{\rm T} \in \mathbf{R}^m$ is the unknown parameters vector and $F : \mathbf{R}^n \times \mathbf{R}^m \rightarrow \mathbf{R}^n$ is a given nonlinear vector function. In order to estimate the unknown parameters in (1), an estimated model is defined below:

$$\dot{\hat{X}} = F(\hat{X}, X_{\circ}, \hat{\theta}) \tag{2}$$

where $\hat{X} = [\hat{x}_1, \hat{x}_2, \cdots, \hat{x}_n] \in \mathbf{R}^n$ and $\hat{\theta} = [\hat{\theta}_1, \hat{\theta}_2, \cdots, \hat{\theta}_n]^{\rm T} \in \mathbf{R}^m$ are the estimated state vector and the estimated parameter vector, respectively.

Since heuristic algorithms depend only on the objective function to guide the search, it must be defined before these algorithms are initialized. In this paper, the mean squared errors (MSEs) between real and estimated responses for a number of given samples are considered as fitness of estimated model parameters. Hence, the objective function is chosen as follows:

$$\text{MSE} = \frac{1}{N}\sum_{k=1}^{N} \text{e}^2 = \frac{1}{N}\sum_{k=1}^{N} [X(k) - \hat{X}(k)]^2 \tag{3}$$

where $N$ is the sampling number and $X(k)$ and $\hat{X}(k)$ are real and estimated values at time $k$, respectively.

The contribution of this paper is to apply the proposed APSO algorithm to minimizing the MSE value such that the actual nonlinear system parameters are accurately estimated. Fig. 1 presents a block diagram of nonlinear system parameter estimation. Considering Fig. 1, the initial state is given to both the real system and the estimated model. Then outputs from the real system and its estimated model are input to the optimization algorithm, where the objective function (MSE) will be calculated.

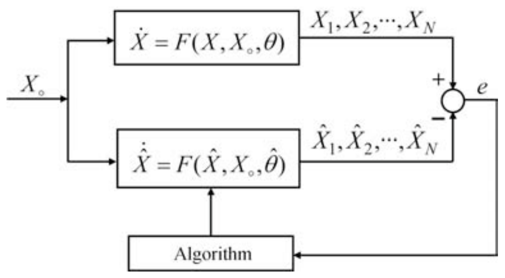
$\dot{X} = F(X, X_{\circ}, \theta)$
$X_1, X_2, \cdots, X_N$
$X_{\circ}$
+
$e$
−
$\dot{\hat{X}} = F(\hat{X}, X_{\circ}, \hat{\theta})$
$\hat{X}_1, \hat{X}_2, \cdots, \hat{X}_N$
Algorithm

Fig. 1 The principle of parameter estimation for nonlinear systems

## 2 PSO-based parameter estimation

### 2.1 Overview of PSO

PSO is an optimization algorithm simulating the social behavior of flocks of birds. PSO is a population-based search process where individuals initialized with a population of random solutions, called as particles, are grouped into a swarm. Each particle in the swarm represents a candidate solution to the optimization problem, and if the solution is made up of a set of variables, the particle can correspondingly be a vector of variables.

In PSO, each particle is flown through the multidimensional search space, adjusting its position in the search space according to its momentum and both individual and global memories. The particle therefore makes use of the best position encountered by itself and that of its neighbors to position itself toward an optimal solution. The fitness of each particle can be evaluated according to the objective function of the optimization problem. At each iteration, the velocity of every particle will be calculated as follows:

$$\begin{aligned} v_i(t+1) = &\, \omega\, v_i(t) + c_1 r_1 (p_{id} - x_i(t)) + \\ &\, c_2 r_2 (p_{gd} - x_i(t)) \end{aligned} \tag{4}$$

where $t$ is the current step number, $\omega$ is the inertia weight, $c_1$ and $c_2$ are the acceleration constants, $r_1$ and $r_2$ are two random numbers in the range [0,1], $x_i(t)$ is the current position of the particle, $p_{id}$ is the best one of the solutions this particle has reached, and $p_{gd}$ is the best one of the solutions all the particles have reached. After calculating the velocity, the new position of every particle can be worked out

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{5}$$

The PSO algorithm performs repeated applications of the update equations above until a stopping criterion is reached.

### 2.2 APSO

Although PSO has shown some important advances by providing high speed of convergence in specific problems, it does exhibit some shortages. It sometimes is easy to be trapped in local optimum and the convergence rate decreases considerably in the later period of evolution[14]. Several approaches have been introduced to cope with these deficiencies[15−25]. Among them, many approaches were attempted to improve the performance of basic PSO by variable inertia weight. The inertia weight controls the local and global exploration capabilities of PSO. A large inertia weight enables the PSO to explore globally, and a small inertia weight enables it to exploit locally. Based on this, a PSO with linearly decreasing weight was introduced to balance the global exploration and the local exploitation in [20]. Also, many approaches have been used the inertia weight as a nonlinear function of iteratin[16, 18, 21].

Although, a good inertia weight adaptation can improve the performance of PSO in terms of accuracy and convergence speed, one can enhance the global search ability of PSO by using an appropriate mutation mechanism as used in other evolutionary algorithms. To gain the advantages of mutation mechanism, some researchers applied the mutation operator to PSO[15, 17, 19]. The PSO algorithm with mutation operator not only has great advantages of convergence property, but also can avoid the premature convergence problem.

In this paper, to achieve the advantages of both inertia weight and mutation mechanism, two modifications including adaptive mutation mechanism and adaptive inertia weight are introduced. Then, these aspects are combined in the proposed adaptive particle swarm optimization (APSO) algorithm is proposed. In the following, these aspects are described sorely.

### 2.2.1 The proposed adaptation of mutation mechanism

Considering a variable step size for mutation operator which decreases as iteration goes on cannot be proper as done previously[11, 15, 19] in the proposed mutation mechanism, when a particle is chosen to mutate, a Gaussian random disturbance is added to it as follows:

$$x_{ij} = x_{ij} + M \times \beta_{ij} \tag{6}$$

where $x_{ij}$ is the $i$-th component of the $j$-th particle, $\beta_{ij}$ is a random variable with Gaussian distribution with zero mean and unit variance, and $M$ is a variable step size which dynamically decreases according to current best solution fitness.

Since the initial population in an optimization problem may locate far away from the real optimal solution, PSO is very hard to succeed. So the proposed algorithm starts with a big mutation step size to increase the chance of searching new areas, in order to enhance the global search ability and convergence speed of basic PSO. In contrast when current best solution reaches a near optimum solution, a big step size may make the population get away from the local convergent solution and it can decrease the possibility of the convergent accuracy. Hence, in this case a small step size is used for finer local exploration around the current best solution. As a result, the value of $M$ is based on current best solution fitness and defined in $t$-th iteration as follows:

$$M_t = x_{\max} \times \tanh\left[\frac{1}{\alpha} \times F(P_{gd}^t)\right] \tag{7}$$

where $\tanh(\cdot)$ is an abbreviation for hyperbolic tangent and $F(P_{gd}^t)$ is the fitness of current best solution in $t$-th iteration. The parameter $\alpha$ needs to be pre-defined. The value of $\alpha$ can set equal to the fitness of the best particle in the initial population ($\alpha = F(P_{gd}^1)$). In this case, the variable step size $M$ changes according to the rate of fitness improvement of the best particle. According to (7), since $F(P_{gd}^t)$ is positive, $0 \leq \tanh(\cdot) \leq 1$ and it can be concluded that $0 \leq M \leq x_{\max}$. Notice that since $\tanh(\cdot)$ is a monotonic increasing function, it is clearly obvious that the bigger the fitness of current best solution is, the bigger the mutation step size is, and vice versa.

### 2.2.2 The Proposed Adaptation of Inertia Weight

Since the search process of PSO is nonlinear and highly complicated, linearly and nonlinearly decreasing inertia weight with no feedback taken from the current best solution fitness cannot truly reflect the actual search process. So in this paper, the inertia weigh is set as a function of current best solution fitness in $t$-th iteration as follows:

$$\omega_t = 0.5\left\{1 + \tanh\left[\frac{1}{\alpha} \times F(P_{gd}^t)\right]\right\} \tag{8}$$

It can be concluded that $0.5 \leq \omega \leq 1$. This causes that if the fitness of current best solution does not improve significantly, the inertia weight decreases slowly since it still needs to globally explore the search space. On the other hand, if the fitness of current best solution improves significantly, it decreases fast to facilitate finer local exploration because the algorithm reaches a near optimum solution. The main objective is to achieve better solution accuracy.

APSO algorithm is used to find the best system parameter, which simulates the behavior of the dynamic system. Each particle represents all parameters of the estimated model. The flow chart of the above APSO algorithm for system parameter estimation is shown in Fig. 2.

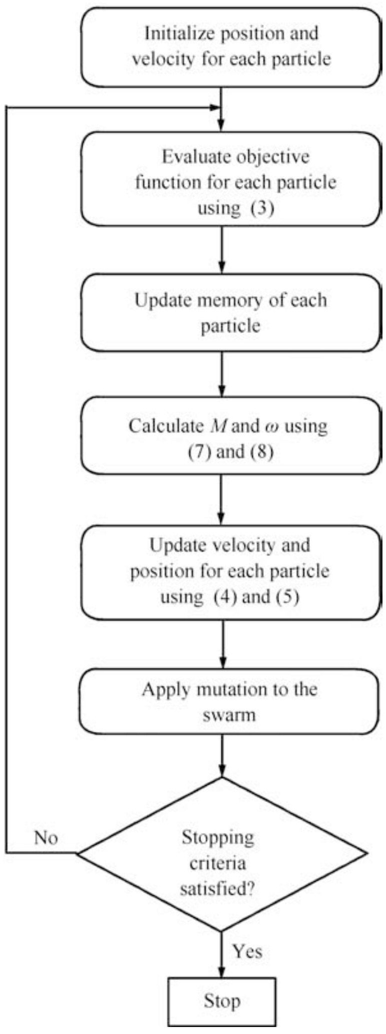
Initialize position and velocity for each particle
Evaluate objective function for each particle using (3)
Update memory of each particle
Calculate $M$ and $\omega$ using (7) and (8)
Update velocity and position for each particle using (4) and (5)
Apply mutation to the swarm
Stopping criteria satisfied?
No
Yes
Stop

Fig. 2 The flow chart of APSO algorithm for parameter estimation

## 3 Testing with benchmark problems

### 3.1 Benchmark functions

From the standard set of benchmark problems available in the literature, three significant functions: one is unimodal (containing only one optimum) and the other two are multimodal (containing many local optima, but only one global optimum) are considered to verify and demonstrate the effectiveness of the proposed APSO algorithm. Three well-known benchmark functions with asymmetric initial range settings namely Griewank, Rosenbrock and Rastrigrin are adopted as the testing functions. The information of the chosen benchmark functions including the dimensions, admissible ranges of variables and optima are summarized in Table 1. In the following, the characteristics of these functions are briefly described.

The Griewank function has many widespread local minima regularly distributed[26]. It is a continuous, multimodal, scalable, convex, and quadratic test function. It is represented by

$$f_1(x)=\sum_{i=1}^{n}\left(\frac{x_i^2}{4\,000}\right)-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1 \tag{9}$$

The Rosenbrock function has a significant interaction between some of the variables. The global minimum is inside a long, narrow, parabolic-shaped flat valley[27]. Finding a valley is trivial, however convergence to the global optimum is difficult and hence this problem has been repeatedly used in assessing the performance of optimization algorithms. It is represented by

$$f_2(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2] \tag{10}$$

The Rastrigrin function is a fairly difficult problem due to the large search space and large number of local optima. The function is highly multimodal and nonlinear such that the locations of the minima are regularly distributed[27]. It is given by

$$f_3(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10] \tag{11}$$

In all benchmark functions, the global minimum is $f_i(x^*)=0$, where $x^*=0$ and $i=1,2,3$.

### 3.2 Performance Measurement

To evaluate the algorithm in term of the search capability, the performance of APSO is compared with PSO, NDWPSO, and GA using the test functions described in Table 1. In NDWPSO, the inertia weight is adapted nonlinearly as follows[16]:

$$\omega^t=\omega_{\min}+\left(\frac{iter_{\max}-iter}{iter_{\max}}\right)^n\times(\omega_{\max}-\omega_{\min}) \tag{12}$$

where $iter_{\max}$ is the maximal number of iterations, and $t$ is the current number of iterations. So as iterations goes, $\omega$ decreases nonlinearly from $\omega_{\max}$ to $\omega_{\min}$ and $n$ is the nonlinear modulation index. We illustrated that $n=1.2$ showed encouraging results for several benchmark problems. First of all, in order to observe the impact of $\alpha$ on the performance of APSO, different values of $\alpha$, 20 particles and 1 000 maximum iterations for three benchmark functions with 10 dimensions were conducted. For each experimental setting, 20 runs of the algorithm were performed. Table 2 listed the mean best fitness values averaged over 20 runs. It is clear that the values in range [0.4, 0.7] for $\alpha$ can all lead to acceptable performance. Based on this, in present paper, $\alpha$ is set to 0.6.

To compare the proposed APSO with other algorithms, for each benchmark function, three dimensions were tested: 10, 20 and 30; correspondingly, the maximum numbers of generations were set as 1 000, 1 500 and 2 000. In addition, for investigation of the scalability of the algorithms, three population sizes 20, 40 and 80 were used for each function with different dimensions. For each experimental setting, 30 runs of the algorithm were performed. Moreover, to perform fair comparison, the same computational effort was used in all of PSO, NDWPSO and APSO. Thereby, in both PSO and APSO, we set $c_1=c_2=2$ and $V_{\max}$ and

Table 1 Properties of the benchmark functions

| Function | Search space $[x_{\min},x_{\max}]$ | Optimum point | Modality |
|---|---|---|---|
| Griewank | $[-600,600]^n$ | 0 | multimodal |
| Rastrigrin | $[-5.12,5.12]^n$ | 0 | multimodal |
| Rosenbrock | $[-30,30]^n$ | 0 | unimodal |

Table 2 The mean best fitness values with different values of $\alpha$

| | $\alpha$ | 1 | 0.9 | 0.8 | 0.7 | 0.6 | 0.5 | 0.4 | 0.3 | 0.2 | 0.1 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Griewank | 0.1058 | 0.1040 | 0.0989 | 0.0952 | 0.0967 | 0.0955 | 0.0989 | 0.1067 | 0.1008 | 0.1055 |
| MSE | Rastrigrin | 5.3145 | 5.1756 | 5.2457 | 5.1587 | 5.1667 | 5.1467 | 5.1708 | 5.1945 | 5.3063 | 5.2625 |
| | Rosenbrock | 5.9832 | 5.9378 | 5.8856 | 5.8943 | 5.8526 | 5.9106 | 5.8427 | 5.8479 | 5.9822 | 6.0368 |

Table 3 Mean fitness values for Griewank function

| Population size | Dimension | Generation | PSO | NDWPSO | GA | APSO |
|---|---|---|---|---|---|---|
| | 10 | 1 000 | 0.1543 | 0.1294 | 0.2253 | 0.0983 |
| 20 | 20 | 1 500 | 0.0636 | 0.0492 | 0.1432 | 0.0237 |
| | 30 | 2 000 | 0.0597 | 0.0421 | 0.1053 | 0.0117 |
| | 10 | 1 000 | 0.1483 | 0.1187 | 0.2161 | 0.0952 |
| 40 | 20 | 1 500 | 0.0608 | 0.0451 | 0.1296 | 0.0201 |
| | 30 | 2 000 | 0.0496 | 0.0293 | 0.0994 | 0.0105 |
| | 10 | 1 000 | 0.0972 | 0.0852 | 0.1745 | 0.0689 |
| 80 | 20 | 1 500 | 0.0594 | 0.0418 | 0.1247 | 0.0199 |
| | 30 | 2 000 | 0.0361 | 0.0285 | 0.0863 | 0.0102 |

Table 4 Mean fitness values for Rastrigrin function

| Population size | Dimension | Generation | PSO | NDWPSO | GA | APSO |
|---|---|---|---|---|---|---|
| | 10 | 1 000 | 6.1057 | 5.7735 | 7.8546 | 5.1565 |
| 20 | 20 | 1 500 | 23.4572 | 20.6359 | 27.9632 | 16.0456 |
| | 30 | 2 000 | 50.4518 | 48.9464 | 55.6473 | 42.2325 |
| | 10 | 1 000 | 4.1565 | 3.8095 | 6.0784 | 2.9468 |
| 40 | 20 | 1 500 | 20.1457 | 19.0883 | 24.2374 | 15.3678 |
| | 30 | 2 000 | 41.1568 | 39.8539 | 45.9746 | 33.7538 |
| | 10 | 1 000 | 2.7325 | 2.5856 | 3.0744 | 2.0457 |
| 80 | 20 | 1 500 | 15.0343 | 14.9647 | 18.8538 | 10.0563 |
| | 30 | 2 000 | 32.4578 | 30.9773 | 36.9556 | 25.3473 |

Table 5 Mean fitness values for Rosenbrock function

| Population size | Dimension | Generation | PSO | NDWPSO | GA | APSO |
|---|---|---|---|---|---|---|
| | 10 | 1 000 | 25.3463 | 20.4258 | 32.4326 | 5.8467 |
| 20 | 20 | 1 500 | 74.1453 | 69.1346 | 88.3457 | 47.9842 |
| | 30 | 2 000 | 142.3465 | 126.1235 | 176.4572 | 100.4528 |
| | 10 | 1 000 | 22.916 | 16.1433 | 29.1276 | 4.5431 |
| 40 | 20 | 1 500 | 59.2419 | 52.2478 | 76.2085 | 38.3464 |
| | 30 | 2 000 | 87.3837 | 90.7609 | 122.6792 | 72.5473 |
| | 10 | 1 000 | 17.4744 | 14.3463 | 23.3467 | 4.1680 |
| 80 | 20 | 1 500 | 51.6443 | 43.6479 | 68.6724 | 27.9547 |
| | 30 | 2 000 | 99.4563 | 74.6156 | 97.7245 | 69.0609 |

Table 6 Analysis results for the benchmark functions for dimension of 10

| | Griewank | | Rastrigrin | | Rosenbrock | |
|---|---|---|---|---|---|---|
| Method | Population size | St.D. | Population size | St.D. | Population size | St.D. |
| | 20 | 0.0271 | 20 | 0.1973 | 20 | 4.1323 |
| PSO | 40 | 0.0168 | 40 | 0.1634 | 40 | 3.2472 |
| | 80 | 0.0148 | 80 | 0.1632 | 80 | 3.1545 |
| | 20 | 0.0248 | 20 | 0.1823 | 20 | 3.6832 |
| NDWPSO | 40 | 0.0174 | 40 | 0.1467 | 40 | 2.8573 |
| | 80 | 0.0118 | 80 | 0.1568 | 80 | 2.7435 |
| | 20 | 0.3473 | 20 | 0.2267 | 20 | 9.3247 |
| GN | 40 | 0.2659 | 40 | 0.2045 | 40 | 6.2466 |
| | 80 | 0.1945 | 80 | 0.1846 | 80 | 6.9367 |
| | 20 | 0.0054 | 20 | 0.1358 | 20 | 1.3471 |
| APSO | 40 | 0.0036 | 40 | 0.1064 | 40 | 1.2376 |
| | 80 | 0.0029 | 80 | 0.1084 | 80 | 1.1450 |

$V_{\min}$ were equal to the length of the search space[6, 22]. Moreover, in NDWPSO the inertia weight $\omega$ decreased nonlinearly from 0.9 to 0.4[16] and in APSO, $\omega$ was determined using (8) as described in Section 3.2. Also, in GA, the crossover probability $P_c$ and the mutation probability $P_m$ were set to 0.8 and 0.1, respectively[28−29].

The performance measurement for the three functions is listed in Tables $3 \sim 6$. The performance results are exhibited in terms of the mean fitness values of the best particle and the standard deviation (denoted by St.D.). From Tables $3 \sim 6$, it can be seen that the fitness value of the proposed APSO algorithm is smaller than others across three test functions. Comparative results shown in Table 6 indicate that the proposed APSO has good global search ability. During almost every run, the APSO could find the optimum of the three complex test functions. Ultimately, from these tables, it is clearly obvious that the proposed APSO algorithm outperforms other algorithms.

## 4 Case study

In this section, the proposed APSO algorithm is used to identify parameters of two nonlinear systems used for parameter estimation in [2, 10−11]. Those systems are difficult to describe because even though we could able to obtain a feasible functional structure modeling the system, they typically depend on some physical parameters which are hard to be determined.

The aim is to avoid confusion between the results of APSO and other algorithms, so only the results of NDWPSO which is the better than PSO in terms of convergence speed are compared, excluding the PSO results. The parameters were chosen based on values represented in Section 4.2.

**Example 1.** An unstable nonlinear system is described by[2]

$$\begin{aligned} x_1(k+1) &= a_1 x_1(k) x_2(k), && x_1(0)=1 \\ x_2(k+1) &= a_2 x_1^2(k) + u(k), && x_2(0)=1 \\ y(k) &= a_3 x_2(k) - a_4 x_1^2(k) && \end{aligned} \tag{13}$$

In this example, the real system parameters of (13) are assumed to be $[a_1, a_2, a_3, a_4] = [0.5,\ 0.3,\ 1.8,\ 0.9]$. In addition, the searching ranges are set as follows: $0 \le a_1 \le 2$, $0 \le a_2 \le 2$, $0 \le a_3 \le 2$ and $0 \le a_4 \le 2$

To do a fair comparison, the maximum generation and population sizes are set to 150 and 50, respectively for GA, NDWPSO and APSO. The optimization process is repeated 20 times independently. The averages of these results are provided in Table 7.

Figs. 3 $\sim$ 7 confirm the success of optimization process by using APSO algorithm in comparison with the other algorithms for the estimated parameters $a_1$, $a_2$, $a_3$, $a_4$ and objective function, respectively. These figures are considered for a middle run of each algorithm. Moreover, to compare the computational time of these algorithms, a threshold of $10^{-5}$ is considered as the stopping condition, in contrast to a predefined number of generation. Then each algorithm runs 20 times and the average of the elapsed time is considered as a criterion for computational time.

Table 8 illustrates the results obtained by APSO, NDWPSO and GA algorithms. The results indicate in how many iterations and necessary time, the convergence of the solution or success is met.

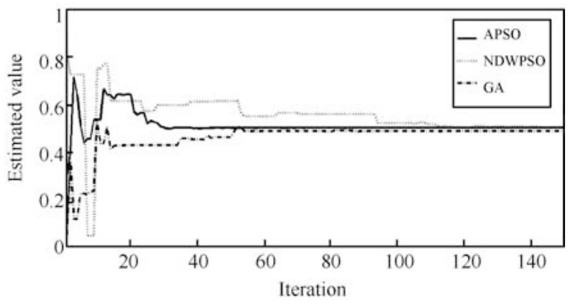

Fig. 3 Comparison of trajectories of parameter $a_1$ for a middle run

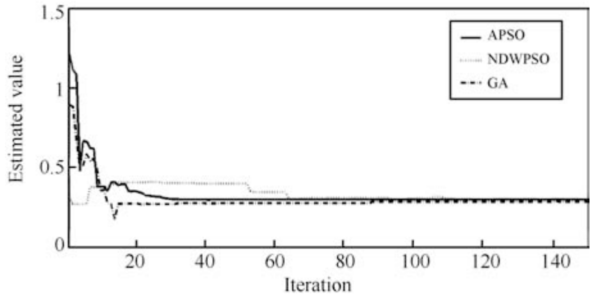

Fig. 4 Comparison of trajectories of parameter $a_2$ for a middle run

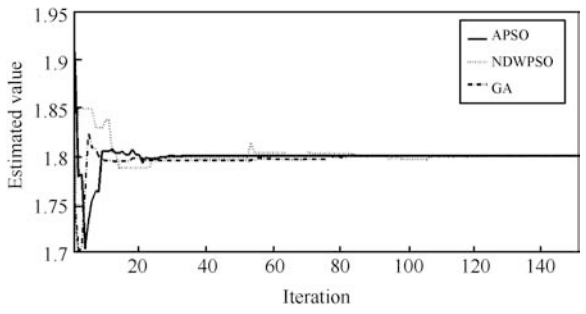

Fig. 5 Comparison of trajectories of parameter $a_3$ for a middle run

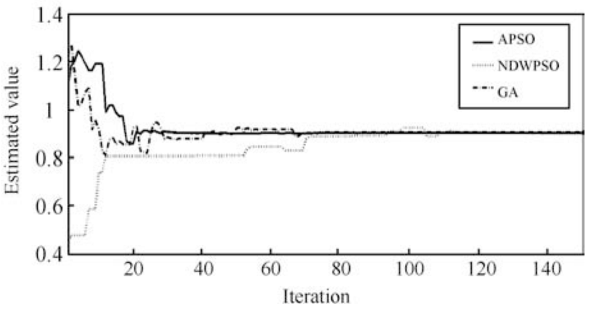

Fig. 6 Comparison of trajectories of parameter $a_4$ for a middle run

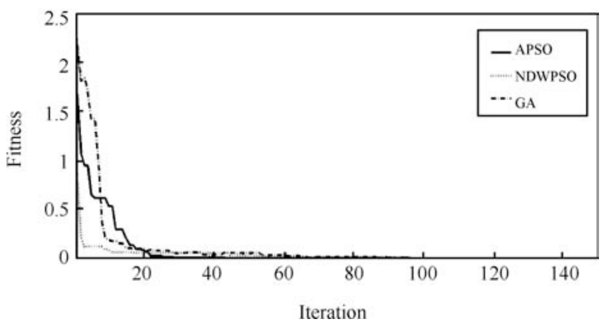

Fig. 7 Comparison of convergence of objective function for a middle run in Example 1

**Example 2. Lorenz chaotic system**

A chaotic system is a nonlinear deterministic system and its prominent characteristic is the sensitive dependence on initial conditions. Due to the complexity and unpredictable behavior of chaotic systems it is difficult to determine parameters of these systems. So a Lorenz system which is a known chaotic system is considered to show the performance of the proposed algorithm in parameter estimation of chaotic nonlinear systems. The Lorenz system is described by[14]

$$\begin{aligned} \dot{x}_1 &= \theta_1 (x_2 - x_1) \\ \dot{x}_2 &= \theta_2 x_1 - x_2 - x_1 x_3 \\ \dot{x}_3 &= x_1 x_2 + \theta_3 x_3 \end{aligned} \tag{14}$$

where vector parameter $\theta = [\theta_1, \theta_2, \theta_3]^{\mathrm{T}} = [10,\ 28,\ 2.6667]^{\mathrm{T}}$ must be estimated. The searching ranges are set as follows: $0 \le \theta_1 \le 20$, $0 \le \theta_2 \le 50$, and $0 \le \theta_3 \le 5$.

Table 7 A comparison between using GA, NDWPSO and APSO for Example 1

| | $a_1$ | $a_2$ | $a_3$ | $a_4$ | MSE | St.D. | Elapsed time (s) |
|---|---|---|---|---|---|---|---|
| Real parameters | 0.5000 | 0.3000 | 1.8000 | 0.9000 | - | - | - |
| GA | 0.4945 | 0.2991 | 1.7994 | 0.8885 | $7.05\times10^{-3}$ | $6.85\times10^{-1}$ | 11.3619 |
| NDWPSO | 0.5001 | 0.3002 | 1.8000 | 0.9002 | $5.60\times10^{-11}$ | $5.45\times10^{-8}$ | 3.7612 |
| APSO | 0.5000 | 0.3000 | 1.8000 | 0.9000 | $3.86\times10^{-22}$ | $2.83\times10^{-20}$ | 3.8367 |

Table 8 Iterations and time required by GA, NDWPSO and APSO algorithms for Example 1

| | Best result | | | Average result | | | Worst result | | |
|---|---|---|---|---|---|---|---|---|---|
| | GA | NDWPSO | APSO | GA | NDWPSO | APSO | GA | NDWPSO | APSO |
| Iterations | 178 | 95 | 41 | 211 | 119 | 48 | 302 | 133 | 56 |
| Elapsed time (s) | 8.89 | 2.45 | 1.08 | 10.62 | 3.12 | 1.26 | 5.05 | 3.46 | 1.47 |

Table 9 A comparison between using GA, NDWPSO and APSO for Example 2

| | $\theta_1$ | $\theta_2$ | $\theta_3$ | MSE | St.D. | Elapsed time (s) |
|---|---|---|---|---|---|---|
| Real parameters | 10.0000 | 28.0000 | 2.6667 | - | - | - |
| GA | 10.0377 | 27.9961 | 2.6603 | $8.02\times10^{-4}$ | $4.61\times10^{-2}$ | 14.9254 |
| NDWPSO | 10.0022 | 28.0035 | 2.6666 | $1.72\times10^{-9}$ | $3.73\times10^{-6}$ | 4.9136 |
| APSO | 10.0000 | 28.0000 | 2.6667 | $2.66\times10^{-20}$ | $8.74\times10^{-19}$ | 5.1135 |

Table 10 Iterations and time required by GA, NDWPSO and APSO algorithms for Example 2

| | Best result | | | Average result | | | Worst result | | |
|---|---|---|---|---|---|---|---|---|---|
| | GA | NDWPSO | APSO | GA | NDWPSO | APSO | GA | NDWPSO | APSO |
| Iterations | 191 | 114 | 57 | 232 | 135 | 66 | 345 | 157 | 79 |
| Elapsed time (s) | 9.59 | 3.02 | 1.51 | 11.58 | 3.55 | 1.77 | 17.15 | 4.11 | 3.96 |

According to (3), in this case, the objective function is chosen as

$$\mathrm{MSE} = \frac{1}{N}\sum_{k=1}^{N}([x_1(k)-\hat{x}_1(k)]^2 + [x_2(k)-\hat{x}_2(k)]^2 + [x_3(k)-\hat{x}_3(k)]^2) \tag{15}$$

where $N$ is the sampling number and $x(k)$ and $\hat{x}(k)$ are real and estimated values at time $k$, respectively.

The maximum generation and population sizes are set to 200 and 50, respectively for GA, NDWPSO and APSO. Table 9 lists the average results obtained by GA, NDWPSO and APSO, where each algorithm is implemented 20 times independently.

Figs. 8 $\sim$ 11 confirm the success of optimization process by using APSO algorithm in comparison with GA and NDWPSO for the estimated parameters $\theta_1$, $\theta_2$, $\theta_3$ and objective function, respectively. These figures are considered for a middle run of each algorithm. Notice that in order to better show the difference of convergence speed for these algorithms, Figs. 8$\sim$11 are depicted from iteration 1 to iteration 80. To compare the computational time of the algorithms, a threshold of $10^{-6}$ is considered as the stopping condition, in contrast to a predefined number of generations. Then each algorithm runs 20 times and the average of elapsed time is considered as a criterion for computational time. Table 10 illustrates the results obtained by APSO, NDWPSO and GA algorithms.

From the above two examples, the results presented demonstrate that a good optimal performance can be achieved by the proposed APSO algorithm. From Tables 7 and 9, it can be seen that there is obviously difference between the different parameter estimation algorithms in the MSE and standard deviation denoted by St.D. Then, it is clearly apparent that more accurate and robust parameter estimation can be implemented using APSO in comparison to GA and NDWPSO for both nonlinear dynamical systems. In addition, as Figs. 3$\sim$11 shown, the trajectories of the estimated parameters asymptotically converge to their actual values. Again, it is clearly obvious that APSO converges much faster than GA and NDWPSO. Also, from Tables 8 and 10, it is confirmed that the proposed algorithm spends extremely fewer iterations and less computational time to reach a predefined threshold as compared with other algorithms.

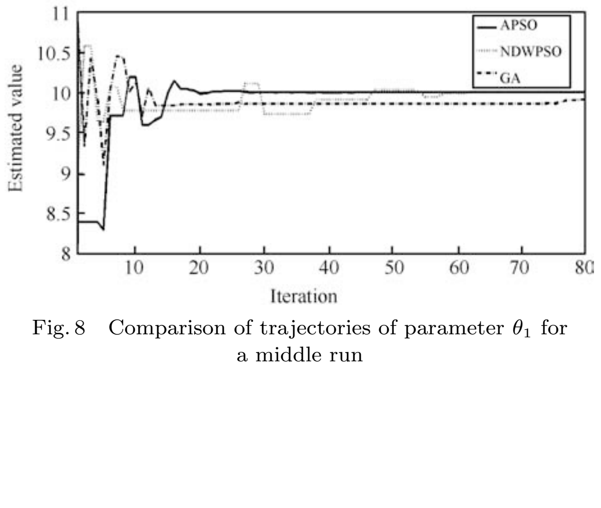

Fig. 8 Comparison of trajectories of parameter $\theta_1$ for a middle run

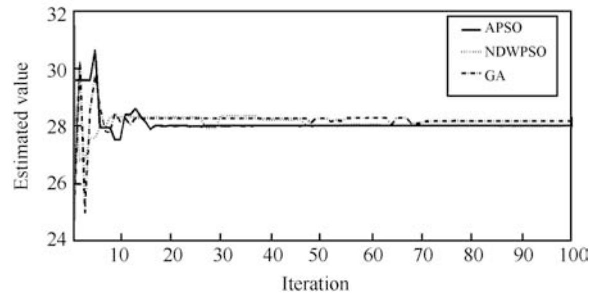

Fig. 9 Comparison of trajectories of parameter $\theta_2$ for a middle run

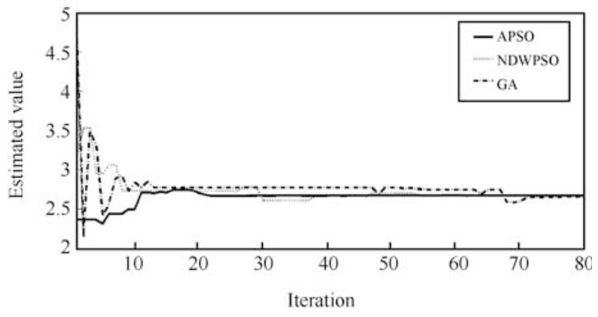

Fig. 10 Comparison of trajectories of parameter $\theta_3$ for a middle run

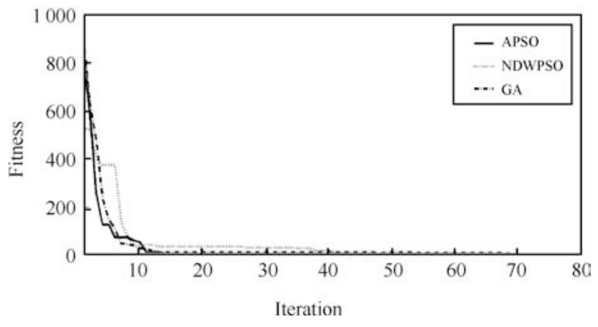

Fig. 11 Comparison of convergences of objective functions for a middle run in Example 2

## 5 Conclusions

An adaptive particle swarm optimization (APSO) was proposed by incorporating two novel modifications into the conventional PSO. An adaptive mutation mechanism was introduced to enhance the global search ability and convergence speed of PSO. Idem, a dynamic inertia weight was applied to improve the accuracy of PSO. The proposed APSO was introduced to solve the parameter estimation problem for nonlinear systems. The simulation results depicted that the proposed APSO has much better potential in terms of solution accuracy and better convergence speed in comparison with real coded GA and NDWPSO when applied to system parameter estimation.

### References

1 Astrom K J, Wittenmark B. *Adaptive Control*. Massachusetts Addison-Wesley: 1995

2 Chang W D. Nonlinear system identification and control using a real-coded genetic algorithm. *Applied Mathematical Modeling*, 2007, **31**(3): 541−550

3 Dai Dong, Ma Xi-Kui, Li Fu-Cai, You Yong. An approach of parameter estimation for a chaotic system based on genetic algorithm. *Acta Physica Sinica*, 2002, **51**(11): 2459−2462

4 Kömürcü M I, Tutkun N, Özölçer I H. Akpinar A. Estimation of the beach bar parameters using the genetic algorithms. *Applied Mathematics and Computation*, 2008, **195**(1): 49−60

5 Billings S A, Mao K Z. Structure for detection for nonlinear rational models using genetic algorithms. *International Journal of Systems Science*, 1998, **29**(3): 223−231

6 Kennedy J, Eberhart R C. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks. Perth, Australia: IEEE, 1995. 1942−1948

7 Bergh F V, Engelbrecht A P. A study of particle swarm optimization particle trajectories. *Information Sciences*, 2006, **176**(8): 937−971

8 Kennedy J, Eberhart R C. The particle swarm: social adaptation in informal-processing systems. *New Ideas in Optimization*. Maidenhead: McGraw-Hill, UK, 1999. 379−387

9 Eberhart R C, Shi Y H. Particle swarm optimization: developments, applications and resources. In: Proceedings of the Congress on Evolutionary Computation. Seoul, South Korea: IEEE, 2001. 81−86

10 He Q, Wang L, Liu, B. Parameter estimation for chaotic systems by particle swarm optimization. *Chaos, Solitons and Fractals*, 2007, **34**(2): 654−661

11 Ye M Y. *Parameter identification of dynamical systems based on improved particle swarm optimization. Lecture Nates in Control and Information Sciences*, Springer-Verlag Berlin Heidelberg, 2006. 351−360

12 Tang Y G, Guan X P. Parameter estimation for time-delay chaotic system by particle swarm optimization, *Chaos, Solitons and Fractals*, 2009, **40**(3): 1391−1398

13 Lin C J, Lee C Y. Non-linear system control using a recurrent fuzzy neural network based on improved particle swarm optimisation, *International Journal of Systems Science*, 2010, **41**(4): 381−395

14 Gaing Z L. A particle swarm optimization approach for optimum design of PID controller in AVR system. *IEEE Transactions on Energy Conversion*, 2004, **19**(2): 384−391

15 Andrews P S. An investigation into mutation operators for particle swarm optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. Vancouver, Canade: IEEE, 2006. 1044−1051

16 Chatterjee A, Siarry P. Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, *Computers and Operations Research*, 2006, **33**(3): 859−871

17 Higasshi N, Iba H. Particle swarm optimization with Gaussian mutation, In: Proceedings Swarm Intelligence Symposium. Washington D. C., USA: IEEE, 2003. 72−79

18 Jiao B, Lian Z G, Gu X S. A dynamic inertia weight particle swarm optimization algorithm. *Chaos, Solitons and Fractals*, 2008, **37**(3): 698−705

19 Stacey A, Jancic M, Grundy I. Particle swarm optimization with mutation, In: Proceedings of the IEEE Congress on Evolutionary Computation, Washington D. C., USA: IEEE, 2003. 1425−1430

20 Shi Y H, Eberhart R C. Parameter selection in particle swarm optimization. In: Proceedings of the 7th International Conference on Evolutionary Programming. London, UK: Springer-Verlag, 1998. 591−600

21 Yang X M, Yuan J S, Yuan J Y, Mao H N. A modified particle swarm optimizer with dynamic adaptation. *Applied Mathematics and Computation*, 2007, **189**(2): 1205−1213

22 Chen J Y, Qin Z, Liu Y, Lu J. Particle Swarm Optimization with Local Search. In: Proceedings of the IEEE International Conference Neural Networks and Brains. Beijing, China: IEEE, 2005. 481−484

23 Modares H, Alfi A, Fateh M M. Parameter identification of chaotic dynamic systems through an improved particle swarm optimization. *Expert Systems with Applications*, 2010, **37**(5): 3714−3720

24 Alfi A, Modares H. System identification and control using adaptive particle swarm optimization. *Applied Mathematical Modelling*, 2011, **35**(3): 1210−1221

25 Modares H, Alfi A, Sistani M B N. Parameter estimation of bilinear systems based on an adaptive particle swarm optimization. *Engineering Applications of Artificial Intelligence*, 2010, **23**(7): 1105−1111

26 Alatas B, Akin E, Ozer A B. Chaos embedded particle swarm optimization algorithms. *Chaos, Solitons and Fractals*, 2009, **40**(4): 1715−1734

27 Griewank A O. Generalized descent of global optimization. *Journal of Optimization Theory and Applications*, 1981, **34**(1): 11−39

28 Grefenstette J J. Optimization of control parameters for genetic algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 1986, **16**(1): 122−128

29 Schaffer J D, Caruana R. A Eshelman L J, Das R A. Study of control parameters affecting online performance of genetic algorithms for function optimization, In: Proceedings of the 3rd International Conference on Genetic Algorithms. San Francisco, USA: Morgan Kaufmann Publishers, 1989. 51−60

**ALFI Alireza** Received his B. Sc. degree from Ferdowsi University of Mashhad, Mashhad, Iran, in 2000, and his M.Sc. and Ph. D. degrees from Iran University of Technology, Tehran, Iran, in 2002 and 2007, all in electrical engineering. He jointed Shahrood University of Technology in 2008, where he is currently an assistant professor of electrical engineering. His research interest covers time delay systems, teleoperation systems, heuristic optimization, control theory, and chaotic systems.