Computers and Mathematics with Applications 62 (2011) 566–576

Contents lists available at ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Particle swarm algorithm for solving systems of nonlinear equations

Majid Jaberipour [a,*], Esmaile Khorram [a], Behrooz Karimi [b]

[a] *Department of Applied Mathematics, Faculty of Mathematics and Computer Science, Amirkabir University of Technology, No. 424, Hafez Ave., 15914 Tehran, Iran*
[b] *Department of Industrial Engineering, Amirkabir University of Technology, No. 424, Hafez Ave., 15914 Tehran, Iran*

A R T I C L E   I N F O

Article history:
Received 14 September 2009
Received in revised form 26 September 2010
Accepted 13 May 2011

Keywords:
Particle swarm
System
Nonlinear equation
Meta-heuristic

A B S T R A C T

Solving systems of nonlinear equations is one of the most difficult problems in all of numerical computation and in a diverse range of engineering applications. Newton's method for solving systems of nonlinear equations can be highly sensitive to the initial guess of the solution. In this study, a new particle swarm optimization algorithm is proposed to solve systems of nonlinear equations. Some standard systems are presented to demonstrate the efficiency of this method.

1. Introduction

Solving systems of nonlinear equations is one of the most difficult problems in all of numerical computation and in a diverse range of engineering applications. Many applied problems are reduced to solving systems of nonlinear equations, which is one of the most basic problems in mathematics. This task has applications in scientific fields such as physics [1–4], chemistry [5], economics [6], computational mechanics, aircraft control etc. Tremendous efforts have been made to solve systems of nonlinear equations and progress along this line now includes a number of constructive theories and algorithms related to systems of nonlinear equations [7–10]. However there still exist some obstacles in solving systems of nonlinear equations. Recently, Luo et al. [11] and Mo et al. [12] solved a system of nonlinear equations using a combination of chaos search and Newton-type methods and a combination of the conjugate direction method (CD) and particle swarm optimization, respectively. Newton-type methods are the most widely used algorithms for solving systems of nonlinear equations, but their convergence and performance characteristics are highly sensitive to the initial guess of the solution supplied to the methods and the algorithm would fail if the initial guess of the solution is improper. However, it is difficult to select a good initial guess for most systems of nonlinear equations. In response to this demand, in this study, an approach to solving systems of nonlinear equations has been presented. Let the form of a system of nonlinear equations be:

$$\begin{aligned} &f_1(x_1, x_2, \ldots, x_n) = 0\\ &f_2(x_1, x_2, \ldots, x_n) = 0\\ &\vdots\\ &f_r(x_1, x_2, \ldots, x_n) = 0. \end{aligned} \tag{1.1}$$

* Corresponding author. Tel.: +98 21 64542549; fax: +98 21 66497930.
E-mail address: Majid.Jaberipour@gmail.com (M. Jaberipour).

0898-1221/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2011.05.031

In order to use global optimization methods, the system of Eq. (1.1) is transformed to an optimization problem. This is achieved by using the auxiliary function:

$$F(x) = \sum_{i=1}^{r} f_i^2(x). \tag{1.2}$$

By definition $F(x) \geq 0$, for the global minimum x^* of $F(x)$ it holds $F(x^*) \geq 0$. If x^*: $F(x^*) = 0$, then it implies that x^* is a global minimum and subsequently $f_1(x^*) = f_2(x^*) = \cdots = f_r(x^*) = 0$ and thus x^* is a root for the corresponding system of equations. In this work, we present a new particle swarm optimization (PSO) algorithm for solving minimization problem (1.2). In the reminder of the paper, it is organized as follows. In Section 2, we describe the basic particle swarm and a proposed particle swarm algorithm. In Section 3, some examples are also presented to demonstrate the effectiveness and robustness of the proposed particle swarm algorithm. Finally, the conclusion is indicated in Section 4.

2. Particle swarm optimization algorithm

The particle swarm optimization (PSO) was proposed by Kennedy and Eberhart [13]. It is a randomized, population-based optimization method that was inspired by the flocking behavior of birds or fish schooling. In PSO, each single solution is a "bird" in the search space. We call it a "particle". A swarm of these particles moves through the search space to find an optimal position. Each particle has a position $x_i = (x_{i1}, x_{i2}, \dots, x_{iN})$ and a velocity $v_i = (v_{i1}, v_{i2}, \dots, v_{iN})$ in the N-dimensional problem space, where i denotes the ith particle and N represents the dimension of the problem or number of unknown variables. PSO is initialized with a group of random particles (solutions) and then searches for optima by updating generations. During every iteration, each particle is updated by following two "best" values. The first one is the position vector of the best solution (fitness) this particle has achieved so far. The fitness value is also stored. This position is called pbest. Another "best" position that is tracked by the particle swarm optimizer is the best position, obtained so far, by any particle in the population. This best position is the current global best and is called gbest. After finding the two best values, the position and velocity of the particles are updated by the following two equations:

$$v_i^k = wv_i^k + c_1r_1(\text{pbest}_i^k - x_i^k) + c_2r_2(\text{gbest}^k - x_i^k)$$
$$x_i^{k+1} = x_i^k + v_i^{k+1}$$

where v_i^k is the velocity of the ith particle at the kth iteration, and x_i^k is the current solution (or position) of the ith particle at the kth iteration. c_1, c_2 are positive constants, and r_1, r_2 are two random variables with uniform distribution between 0 and 1. In this equation, w is the inertia weight which shows the effect of the previous velocity vector on the new vector. An upper bound is placed on the velocity in all dimensions v_{max}. This limitation prevents the particle from moving too rapidly from one region in the search space to another. This value is usually initialized as a function of the range of the problem.

2.1. Proposed particle swarm optimization algorithm (PPSO)

The basic PSO approach typically converges rapidly during the initial search period and then slows. It has the tendency of being trapped in local minima and slow convergence. Furthermore, inertia weight w, c_1 and c_2 are critical factors that affect the convergence of PSO [14–17]. In order to overcome these problems, we introduced the proposed particle swarm optimization algorithm. The key difference between PPSO and the basic PSO method is in the way of updating each particle. In this algorithm, the position and velocity of the particles are updated by the following equations:

$$v_i^{k+1} = (2r_1 - 0.5)v_i^k + (2r_2 - 0.5)(\text{pbest}_i^k - x_i^k) + (2r_3 - 0.5)(\text{gbest}^k - x_i^k) \tag{2.1}$$
$$w^{k+1} = (2r_4 - 0.5)(\text{gbest}^k - \text{pbest}_i^k) + (2r_5 - 0.5)(\text{gbest}^k - x_i^k)$$
$$x_i^{k+1} = \text{pbest}_i^k + (2r_6 - 0.5)v_i^{k+1} + (2r_7 - 0.5)w^{k+1} \tag{2.2}$$

where $r_1, r_2, r_3, r_4, r_5, r_6$ and r_7 are random numbers between 0 and 1. The procedure of the proposed algorithm is summarized as follows:

Step 1. Random generation of an initial population and velocities.

Step 2. Computing the fitness of each particle and determining pbest of each particle and gbest. The initial fitness value of each pbest_i is equal to the fitness value of the current position of each particle.

Step 3. Change the velocity of the particle according to (2.1).

Step 4. Move each particle to a new position using Eq. (2.2).

Step 5. The fitness value of each particle is compared with that of its corresponding pbest. If the fitness value of the ith particle is smaller than that of pbest_i, pbest_i is replaced with the ith particle. The new pbest_i is then stored in the $M_{m\times n}$ matrix, where m is number of particles and n represents number of unknown variables.

Step 6. The worst pbest_i among all the best particles of the M matrix is chosen as $pworst_i$.

Table 1
Results of PPSO.

Variable and function value	Initial iteration	After 100 iterations	After 200 iterations	After 300 iterations	After 420 iterations
x_1	2.0000	0.341	−0.00019	0.000000	0.000000
x_2	2.0000	0.251	−0.00026	−0.000002	0.000000
x_3	−2.0000	−0.0226	0.00013	0.000000	0.000000
x_4	2.0000	0.0012	0.00002	0.000000	0.000000
x_5	−2.0000	−0.0197	0.000015	−0.000001	0.000000
x_6	2.000	0.0247	0.00019	0.000001	0.000000
x_7	2.0000	0.0262	−0.00003	−0.000002	0.000000
x_8	−1.9675	−0.0223	0.00019	−0.000003	0.000000
x_9	1.9871	−0.0135	−0.00010	−0.000004	0.000000
x_{10}	1.0285	−0.0038	−0.00012	−0.000002	0.000000
$f(X)$	37.278233	0.926675	0.00004	0.000000	0.000000

Step 7. Selecting randomly a component l of $pworst_i$ and updating this component as follows:

$$pworst_{i,l}^{new} = pworst_{i,l} + (2r_8 - 0.5)\frac{\frac{\partial f}{\partial pworst_{i,l}}(pworst_i)}{x_U(l) - x_L(l)}, \tag{2.3}$$

where x_U is the upper bound and x_L is the lower bound of each variable. r_8 is a random number between 0 and 1. It is obvious, in most cases, that the functions are too complicated or expensive to calculate. To overcoming this problem, finite differencing is an approach to the calculation of approximate derivatives whose motivation comes from Taylor's theorem. A more accurate approximation to the derivative can be obtained by using the central difference formula, defined as:

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x+\epsilon e_i) - f(x-\epsilon e_i)}{2\epsilon}$$

where f is the objective function, e_i is the ith unit vector and $\epsilon = 10^{-8}$. So Eq. (2.3) is updated as follows:

$$pworst_{i,l}^{new} = pworst_{i,l} + (2r_8 - 0.5)\frac{f(pworst_i + \epsilon e_l) - f(pworst_i - \epsilon e_l)}{2\epsilon(x_U(l) - x_L(l))}. \tag{2.4}$$

Step 8. The fitness value of $pworst_i^{new}$ is compared with that of its corresponding $pworst_i$. If the fitness value $pworst_i^{new}$ is smaller than that of $pworst_i$, $pworst_i$ is replaced with the $pworst_i^{new}$. The new pbest$_i$ is then stored in the M matrix.

Step 9. The best pbest$_i$ among all the particles is chosen as the gbest.

Step 10. Go to *step* 3, and repeat until convergence.

3. Experiment and results

Five benchmark functions are given to investigate the performance of PPSO in this section.

Test 1: Rastrigin function

$$f(x) = \sum_{i=1}^{10}(x_i^2 - 10\cos(2\pi x_i) + 10) \quad |x_i| \le 5.2.$$

The minimum solution of the Rastrigin function is located at point $x = (0, 0, 0, \ldots, 0, 0, 0)$ with an objective function value equal to $f(x) = 0.0$. When applying the PPSO algorithm to the function, the algorithm found the optimal solution after approximately 300 iterations. Results were compared with Mo et al. [12]. The results of the PPSO algorithm, the result of Mo et al. [12] and convergence history of the PPSO algorithm are shown in Tables 1 and 2 and Fig. 1 respectively.

It is obvious from Tables 1 and 2 that the PPSO results are better than the Mo et al. [12] results.

Test 2: Consider the following function:

$$\max f(\mathbf{x}) = -\sum_{i=1}^{D}\left[\sin(x_i) + \sin\left(\frac{2x_i}{3}\right)\right].$$

The objective function of the above function is 1.21598D. When applying the PPSO algorithm to the function with $D = 10$, the algorithm found the optimal solution after approximately 500 iterations. Results and convergence history are shown in Table 3 and Fig. 2 respectively. In this example, the bound variables were set between 3 and 13.

Also when applying the PPSO algorithm to the function with $D = 100$, the algorithm found the optimal solution after approximately 6000 iterations. Results and convergence history are shown in Table 4 and Fig. 3 respectively.

Test 3: Powell quartic function

$$\min f(\mathbf{x}) = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.$$

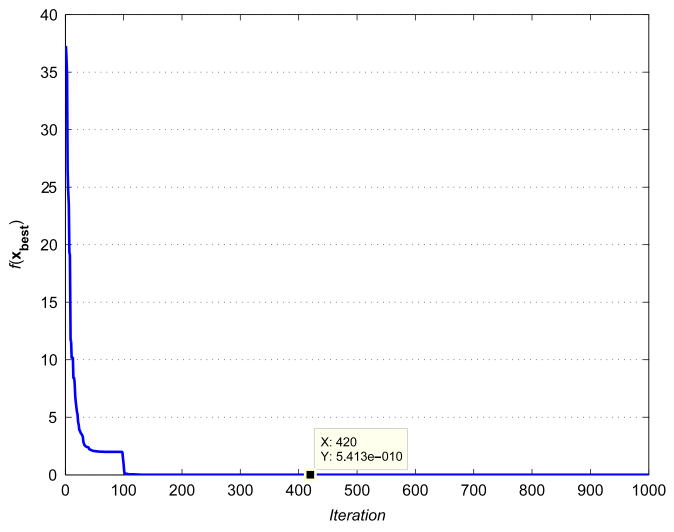

Fig. 1. Convergence history of Rastrigin function.

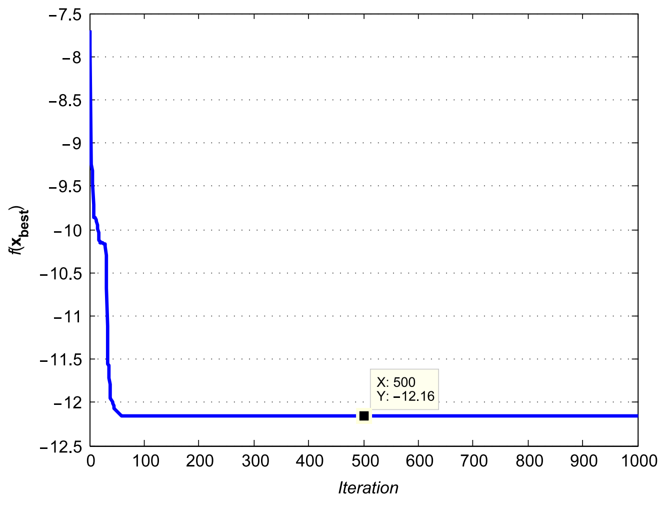

Fig. 2. Convergence history of Test 2 with $D = 10$.

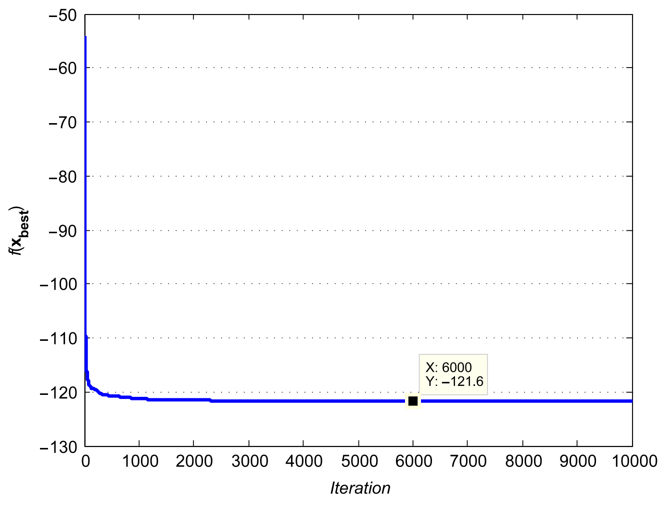

Fig. 3. Convergence history of Test 2 with $D = 100$.

Table 2
Results of Mo et al. [12].

Variable	Initial iteration	After 200 iterations	After 400 iterations	After 600 iterations	After 1000 iterations
x_1	0.1431	−0.0001	−0.0007	0.0001	−0.0000
x_2	2.1983	−0.0001	0.0000	0.0001	0.0001
x_3	1.9401	0.0000	0.0000	0.0001	0.0001
x_4	−1.7080	−0.0002	−0.0001	0.0000	−0.0000
x_5	0.2261	−0.9950	−0.9962	−0.9948	0.0001
x_6	0.9392	0.9950	0.9941	0.9949	0.9949
x_7	−0.1129	0.9949	0.9949	0.0001	0.0000
x_8	−0.1516	0.9950	0.9949	0.9949	−0.0000
x_9	−2.1893	−0.0001	0.0000	0.0001	−0.0000
x_{10}	4.9798	0.9950	0.0000	0.0001	−0.0000

Table 3
Results of Test 2 with $D = 10$.

Variable and function value	Initial iteration	After 100 iterations	After 200 iterations	After 300 iterations	After 400 iterations	After 500
x_1	4.9203	5.3737	5.3667	5.3656	5.3626	5.3623
x_2	4.6815	5.3564	5.3601	5.3618	5.3628	5.3624
x_3	4.9207	5.3522	5.3651	5.3658	5.3627	5.3621
x_4	5.5048	5.3846	5.3656	5.3648	5.3636	5.3633
x_5	6.3685	5.3597	5.3628	5.3630	5.3607	5.3627
x_6	6.7112	5.3520	5.3669	5.3640	5.3631	5.3625
x_7	5.6790	5.3369	5.3621	5.3626	5.3613	5.3624
x_8	6.3557	5.3420	5.3626	5.3629	5.3623	5.3622
x_9	11.7889	5.3705	5.3574	5.3647	5.3627	5.3627
x_{10}	10.3531	5.3515	5.3580	5.3592	5.3622	5.3616
$f(\mathbf{x})$	−7.690599	−12.158781	−12.159769	−12.159797	−12.15981	−12.15982

Table 4
Results of Test 2 with $D = 100$.

Variable and function value	Initial iteration	After 1000 iterations	After 2000 iterations	After 3000 iterations	After 4000 iterations	After 5000 iterations	After 6000 iterations
$f(\mathbf{x})$	54.103342	121.208321	121.554754	121.593659	121.596941	121.598050	121.598204

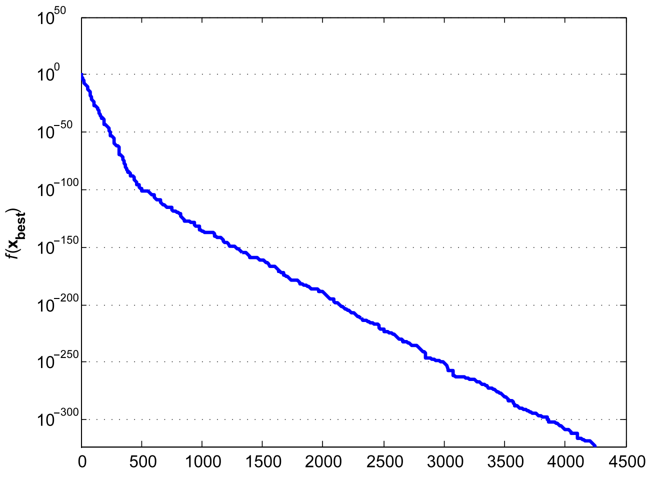

Fig. 4. Convergence history of Powell function.

As the second derivative of the Powell quartic function is singular at the minimum point, it is difficult to obtain the minimum solution using gradient-based algorithms [18]. The minimum solution of the Powell quartic function is located at point $x = (0, 0, 0, 0)$ with an objective function value equal to $f(x) = 0.0$. The convergence history of the PPSO algorithm is shown in Fig. 4.

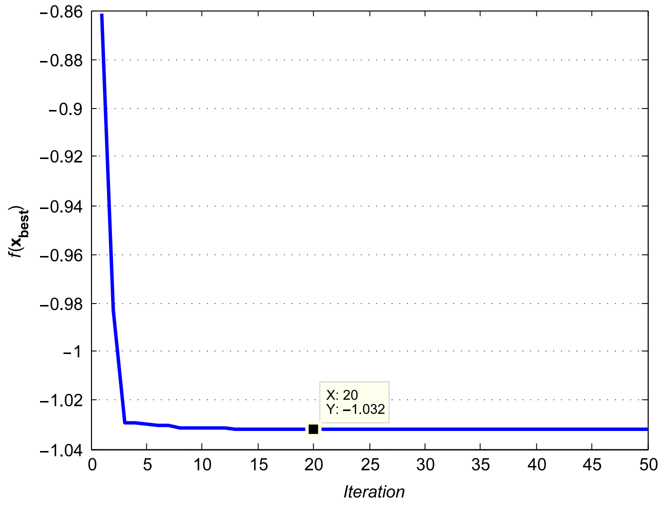

Fig. 5. Convergence history of Six-Hump Camelback function.

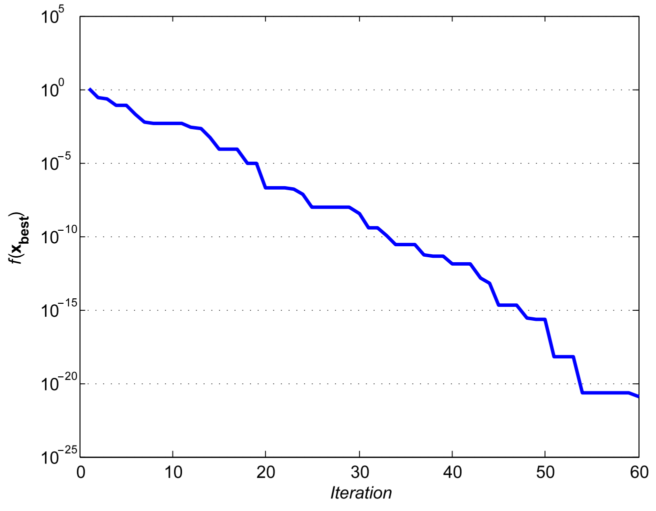

Fig. 6. Convergence history of Rosenbrock function.

Test 4: Six-Hump Camelback

$$\min f(\mathbf{x}) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4.$$

The Six-Hump Camelback has six local optima two of which are global. So the result from gradient-based algorithms may depend on the selection of an initial point. The global optima are located at either $x = (-0.08984, 0.71266)$ or $x = (0.08984, -0.71266)$, each with a corresponding function value equal to $f(x) = -1.0316285$. Convergence history of PPSO algorithm is shown in Fig. 5.

Test 5: Rosenbrock function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2.$$

Because of a long narrow and curved valley present in the function, gradient-based algorithms may require a large number of iterations to obtain the optimal solution. The minimum solution of the Rosenbrock function is at point $x = (1, 1)$ with an objective function value equal to $f(x) = 0.0$. Convergence history of PPSO algorithm is shown in Fig. 6. In this example, the bound variables were set between -10.0 and 10.0.

4. Examples

Now, some standard systems are borrowed from the literature to demonstrate the efficiency of the proposed particle swarm optimization algorithm for solving systems of nonlinear equations. Result of seven case studies are shown in Table 7 and convergence history of each case study is demonstrated in Figs. 7–13.

Table 5
Comparison results of PPSO with Mo et al. [12] and Luo et al. [11].

Methods	b	h	t	$f_1(x)$	$f_2(x)$	$f_3(x)$
PPSO (present study)	43.155566052654329	10.12895020227820	12.944048457756352	165	9369	6835
PPSO (present study)	−7.602995198463455	−24.541982377674739	−11.576715672202731	165	9369	6835
Mo et al. [12]	8.943089	23.271482	12.912774	251.2378	9369	6835
Luo et al. [11]	12.5655	22.8949	2.7898	408.6488	9544.3	7213.1
Luo et al. [11]	−12.5655	−22.8949	−2.7898	408.6488	9544.3	7213.1
Luo et al. [11]	8.943089	23.271482	12.912774	251.2378	9369	6835
Luo et al. [11]	−8.943089	−23.271482	−12.912774	251.2378	9369	6835
Luo et al. [11]	−2.3637	35.7564	3.0151	−334.0376	9369	6835
Luo et al. [11]	2.3637	−35.7564	−3.0151	−334.0376	9369	6835

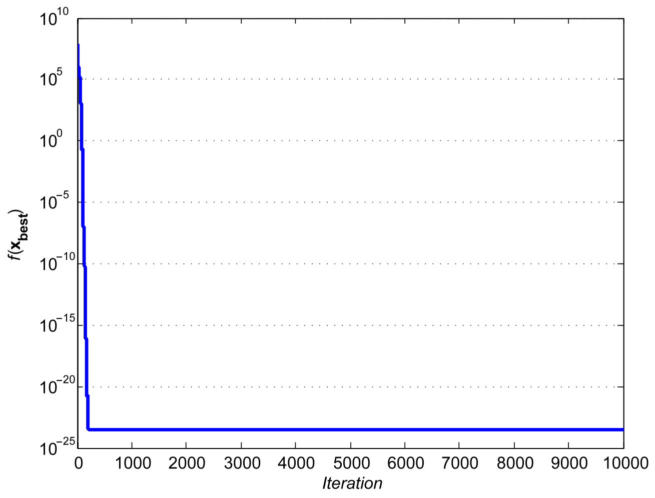

Fig. 7. Convergence history of case study 1.

Numerical test results

Case study 1. Geometry size of thin wall rectangle girder section:

$$\begin{aligned} f_1(x) &= bh - (b-2h)(h-2t) = 165 \\ f_2(x) &= \frac{bh^3}{12} - \frac{(b-2t)(h-2t)^3}{12} = 9369 \\ f_3(x) &= \frac{2t(h-t)^2(b-t)^2}{h+b-2t} = 6835, \end{aligned} \tag{4.1}$$

where b is the width of the section, h is the height of the section and t is the thickness of the section. Recently, Mo et al. [12] solved the above system using a combination of the conjugate direction method (CD) and particle swarm optimization. Also Luo et al. [11] solved the above system using a combination of chaos search and Newton-type methods. Luo et al. [11] found six different solutions. When applying the PPSO algorithm, two solutions were found. Table 5 lists the best solutions obtained by the PPSO method, and compares them with earlier results reported by Mo et al. [12] and Luo et al. [11].

It is obvious from Table 5 that the PPSO results are the exact solution and outperform other two results.

Case study 2. [19]

$$\begin{aligned} f_1(x) &= (3-5x_1)x_1 + 1 - 2x_2 = 0 \\ f_2(x) &= (3-5x_i)x_i + 1 - x_{i-1} - 2x_{i+1} = 0 \quad i = 2, \ldots, 9 \\ f_3(x) &= (3-5x_{10})x_{10} + 1 - x_9 = 0. \end{aligned} \tag{4.2}$$

This system has ten variables and ten equations. Mo et al. [12] solved the above system using a combination of the conjugate direction method (CD) and particle swarm optimization. Table 6 lists the best solution obtained by the PPSO method, and compares them with earlier results reported by Mo et al. [12].

It is obvious from Table 6 that the PPSO results are very close to the exact solution and the solution accuracy is so far better than the solution found by Mo et al. [12].

Also, we solved the following test problems. Table 7 lists the best solutions obtained by the PPSO method and other methods.

Table 6
Comparison results of PPSO with Mo et al. [12].

Solutions and equations	Mo et al. [12]	PPSO (present study)
x_1	0.915551	−0.382084303665302
x_2	−0.222256	−0.438097493266449
x_3	−0.414654	−0.445927622082887
x_4	−0.439254	−0.446971296832353
x_5	0.420892	−0.446951484687073
x_6	−0.354588	−0.446355652774381
x_7	−0.135767	−0.444141158727146
x_8	0.427562	−0.436187333892258
x_9	0.752203	−0.407858897094904
x_{10}	−0.440697	−0.309566878544907
$f_1(x)$	−3.1680e−006	0
$f_2(x)$	3.5232e−007	0
$f_3(x)$	−1.6986e−006	8.8e−016
$f_4(x)$	1.7710e−006	1.11e−016
$f_5(x)$	−1.6836	1.11e−016
$f_6(x)$	2.5254	−1.11e−016
$f_7(x)$	−0.8418	−1.11e−016
$f_8(x)$	−3.9144e−007	−1.11e−016
$f_9(x)$	6.8078e−007	1.11e−016
$f_{10}(x)$	2.3396e−007	−1.66e−016

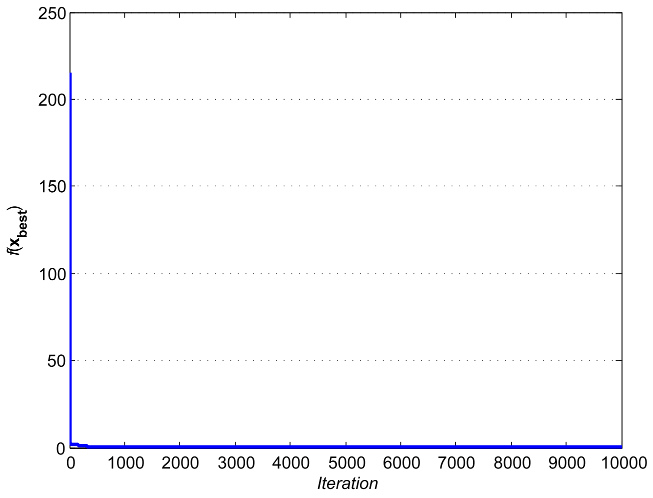

Fig. 8. Convergence history of case study 2.

Case study 3. [8]

$$\begin{aligned}
&x_1 + \frac{x_2^4 x_4 x_6}{4} + 0.75 = 0\\
&x_2 + 0.405e^{1+x_1x_2} - 1.405 = 0\\
&x_3 - \frac{x_4 x_6}{2} + 1.5 = 0\\
&x_4 - 0.605e^{(1-x_3^2)} - 0.395 = 0\\
&x_5 - \frac{x_2 x_6}{2} + 1.5 = 0\\
&x_6 - x_1 x_5 = 0.
\end{aligned}$$

Case study 4. [12]

$$\begin{aligned}
&x_1^{x_2} + x_2^{x_1} - 5x_1x_2x_3 = 85\\
&x_1^3 - x_2^{x_3} - x_3^{x_2} = 60\\
&x_1^{x_3} + x_3^{x_1} - x_2 = 2\\
&3 \le x_1 \le 5, \qquad 2 \le x_2 \le 4, \qquad 0.5 \le x_3 \le 2.
\end{aligned}$$

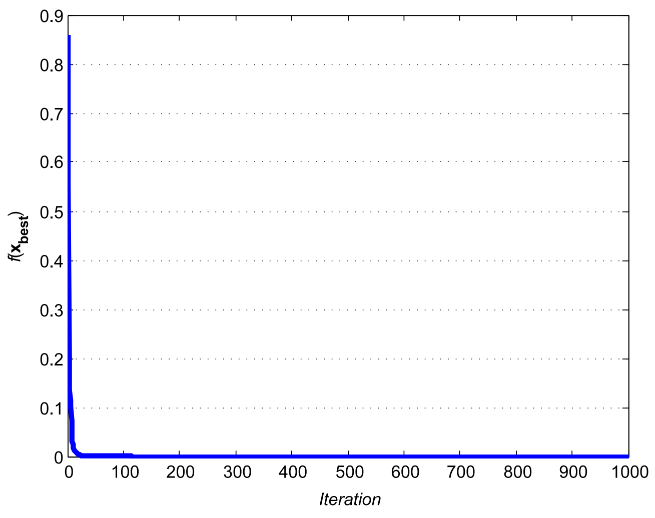

Fig. 9. Convergence history of case study 3.

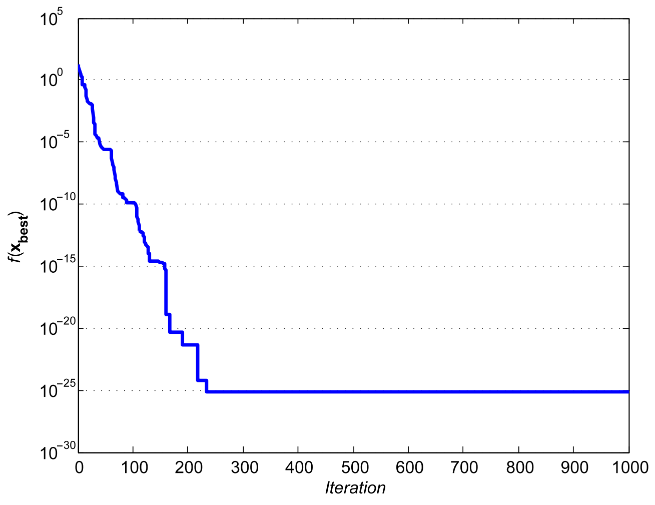

Fig. 10. Convergence history of case study 4.

Case study 5. [20]

$$e^{x_1^2} - 8x_1 \sin(x_2) = 0$$
$$x_1 + x_2 - 1 = 0$$
$$(x_3 - 1)^3 = 0.$$

When applying the PPSO method, two solutions were obtained.

Case study 6. [20]

$$3x_1 - \cos(x_2 x_3) - 0.5 = 0$$
$$x_1^2 - 625x_2^2 - 0.25 = 0$$
$$e^{-x_1 x_2} + 20x_3 + \frac{(10\pi - 3)}{3} = 0.$$

Case study 7. [21]

$$x_1^3 - 3x_1 x_2^2 - 1 = 0$$
$$3x_1^2 x_2 - x_2^3 + 1 = 0.$$

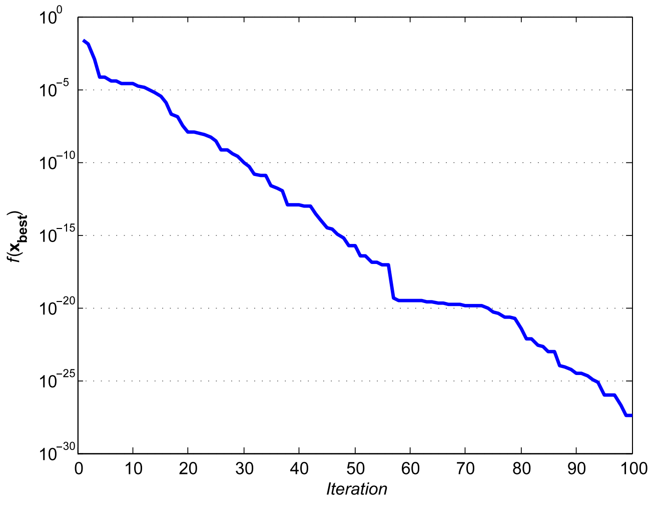

Fig. 11. Convergence history of case study 5.

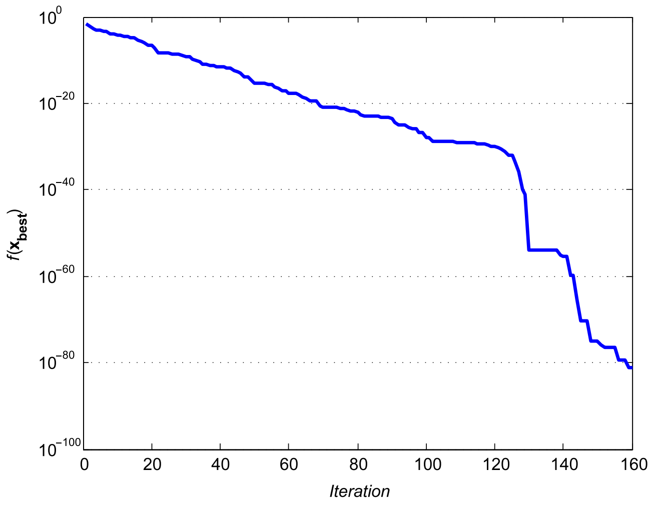

Fig. 12. Convergence history of case study 6.

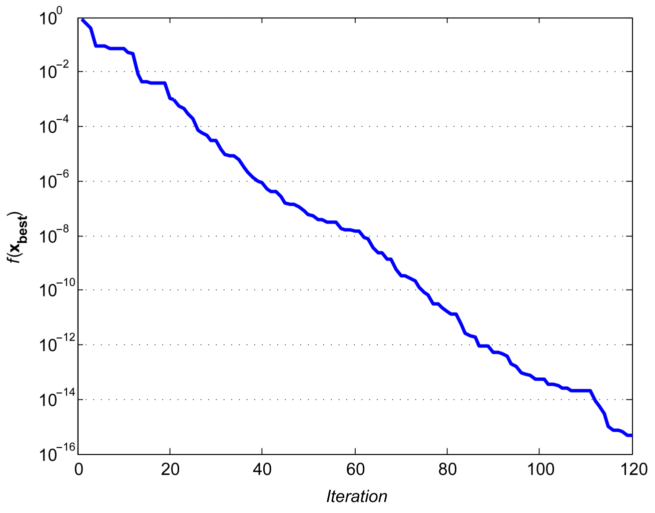

Fig. 13. Convergence history of case study 7.

Table 7
Optimal results of case study.

Case study	Methods	x_1	x_2	x_3	x_4	x_5	x_6
Case 3	Mo et al. [12]	-1	1	-1	1	-1	1
	Krzyworzcka [8]	-1	1	-1	1	-1	1
	PPSO (present study)	-1	1	-1	1	-1	1
Case 4	Mo et al. [12]	4	3	1			
	PPSO (present study)	4	3	1			
Case 5	PPSO (present study)	0.17559892417766	0.82440107582234	1			
	PPSO (present study)	0.70424696664893	0.29575303335107	1			
Case 6	PPSO (present study)	0.5	0	−0.52359877559662			
Case 7	PPSO (present study)	−0.29051455550725	1.08421508149135				
		−0.793700525984100	−0.793700525984100				

5. Conclusions

In this paper, an approach for solving a system of nonlinear equations was presented. A system of nonlinear equations was converted to the minimization problem. Also we suggested solving the minimization problem using a new particle swarm optimization algorithm. Some standard problems were presented to demonstrate the efficiency of finding the best solution of a system of nonlinear equations using the proposed particle swarm optimization algorithm.

Acknowledgments

The authors would like to express their appreciation to the anonymous referees for reviewing our paper and thank them for their valuable comments that improved the quality of the paper.

References

[1] K. Kowalski, K. Jankowski, Towards complete solutions to systems of nonlinear equations of many-electron theories, Phys. Rev. Lett. 81 (1998) 1195–1198.
[2] B.M. Barbashov, V.V. Nesterenko, A.M. Chervyakov, General solutions of nonlinear equations in the geometric theory of the relativistic string, Commun. Math. Phys. 84 (1982) 471–481.
[3] S.R. Bickham, S.A. Kiselev, A.J. Sievers, Stationary and moving intrinsic localized modes in one-dimensional monatomic lattices with cubic and quartic anharmonicity, Phys. Rev. B 47 (1993) 14206–14211.
[4] G. Yuan, X. Lu, A new backtracking inexact BFGS method for symmetric nonlinear equations, Comput. Math. Appl. 55 (2008) 116–129.
[5] A. Holstad, Numerical solution of nonlinear equations in chemical speciation calculations, Comput. Geosci. 3 (1999) 229–257.
[6] K. Argyros, On the solution of undetermined systems of nonlinear equations in Euclidean spaces, Pure Math. Appl. 4 (1993) 199–209.
[7] J. Ortega, W. Rheiboldt, Iterative Solution of Nonlinear Equation in Several Variable, Academic Press, New York, 1970.
[8] S. Krzyworzcka, Extension of the Lanczos and CGS methods to systems of nonlinear equations, J. Comput. Appl. Math. 69 (1996) 181–190.
[9] J. Nocedal, S.J. Wright, Numerical Optimization, Spring Science+Business Media Inc, 1999.
[10] D.G. Huang, Z.G. Ceng, Y. Ma, Nonlinear Numerical Analysis, Wuhan University Press, Wuhan, 2000.
[11] Y.Z. Luo, G.J. Tang, L.N. Zhou, Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method, Appl. Softw. Comput. 8 (2008) 1068–1073.
[12] Y. Mo, H. Liu, Q. Wang, Conjugate direction particle swarm optimization solving systems of nonlinear equations, Comput. Math. Appl. 57 (2009) 1877–1882.
[13] J. Kennedy, R.C. Eberhart, Particle swarm optimization. in: Proceedings of the 1995 IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, NJ, (1995) 1942-1948.
[14] R. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimisation, in: Proceedings of the 2000 Congress on Evol. Comput. (2000) 84–88.
[15] A. El-Gallad, M. El-Hawary, A. Sallam, A. Kalas, Enhancing the particle swarm optimizer via proper parameters selection, IEEE Canadian Conference on Electr. Comput. Eng. 2 (2002) 792–797.
[16] R. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the Congress on Evol. Comput. 3 (2006) 816-822.
[17] M. Iwamatsu, Locating all the global minima using multi-species particle swarm optimizer: the inertia weight and the constriction factor variants, in: Proceedings of the Congress on Evol. Comput. 3 (2006) 816-822.
[18] A.R. Conn, K. Scheinberg, P.L. Toint, On the convergence of derivative-free methods for unconstrained optimization, in: A. Iserles, M. Buhmann (Eds.), Approximation Theory and Optimization: Tributes to M.J.D. Powell, Cambridge University Press, Cambridge, UK, 1997.
[19] S. Krzyworzcka, Extension of the Lanczos and CGS methods to systems of nonlinear equations, Journal of Computational and Applied Mathematics 69 (1) (1996) 181–190.
[20] José L. Hueso, Eulalia Martínez, Juan R. Torregrosa, Modified Newton's method for systems of nonlinear equations with singular Jacobian, J. Comput. Appl. Math. 224 (2009) 77–83.
[21] G.H. Nedzhibov, A family of multi-point iterative methods for solving systems of nonlinear equations, J. Comput. Appl. Math. 222 (2008) 244–250.