Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 727218, 13 pages
http://dx.doi.org/10.1155/2015/727218

Research Article

Research on Solving Systems of Nonlinear Equations Based on Improved PSO

Yugui Li,[1] Yanxu Wei,[2] and Yantao Chu[2]

[1]*College of Materials Science and Engineering, Taiyuan University of Science and Technology, Taiyuan 030024, China*
[2]*College of Mechanical Engineering, Taiyuan University of Science and Technology, Taiyuan 030024, China*

Correspondence should be addressed to Yugui Li; liyugui2008@163.com

Received 5 May 2015; Accepted 15 July 2015

Academic Editor: Valery Sbitnev

Solving systems of nonlinear equations is perhaps one of the most difficult problems in all of numerical computations, especially in a diverse range of engineering applications. The convergence and performance characteristics can be highly sensitive to the initial guess of the solution for most numerical methods such as Newton's method. However, it is very difficult to select reasonable initial guess of the solution for most systems of nonlinear equations. Besides, the computational efficiency is not high enough. Aiming at these problems, an improved particle swarm optimization algorithm (imPSO) is proposed, which can overcome the problem of selecting reasonable initial guess of the solution and improve the computational efficiency. The convergence and performance characteristics of this method are demonstrated through some standard systems. The results show that the improved PSO for solving systems of nonlinear equations has reliable convergence probability, high convergence rate, and solution precision and is a successful approach in solving systems of nonlinear equations.

1. Introduction

Solving systems of nonlinear equations is one of the most important problems in all of numerical computations, especially in a diverse range of engineering applications. Many applied problems can be reduced to solving systems of nonlinear equations, which is one of the most basic problems in mathematics. This task has applications in many scientific fields [1–7]. So great efforts have been made by a lot of people and many constructive theories and algorithms are proposed to solve systems of nonlinear equations [8–11]. However there still exist some problems in solving systems of nonlinear equations. For most traditional numerical methods such as Newton's method, the convergence and performance characteristics can be highly sensitive to the initial guess of solution. However, it is very difficult to select reasonable initial guess of solution for most nonlinear equations. The algorithm would fail or the results may be improper if the initial guess of the solution is unreasonable. Many different combinations of the traditional numerical methods and the intelligent algorithms are applied to solve the systems of nonlinear equations [12, 13], which can overcome the problem of selecting reasonable initial guess of the solution. But the algorithms are too complicated or expensive to calculate when there are a number of systems of nonlinear equations to solve. Many improved intelligent algorithms, such as particle swarm algorithm and genetic algorithm, are proposed to solve systems of nonlinear equations. Though they overcome the problem of selecting reasonable initial guess of the solution, they lack the sophisticated search capabilities in local area, which may lead to convergence stagnation.

Here an improved particle swarm optimization algorithm (imPSO) is put forward, which can overcome the dependence on reasonable initial guess of the solution and improve the computational efficiency.

A system of nonlinear equations can be expressed as

$$F(x) = [f_1(x), f_2(x), f_3(x), \dots, f_n(x)]^T = 0, \quad (1)$$

where $x = (x_1, x_2, x_3, \dots, x_n)^T$ are the n variables.

Set the value of G:

$$G = |f_1(x)| + |f_2(x)| + |f_3(x)| + \cdots + |f_n(x)|. \quad (2)$$

Then the problem of solving nonlinear equations is transformed to a problem of seeking a vector of $\mathbf{x}$ to minimize the value of G and the best value of G is zero, which becomes an optimization problem. Then the imPSO is employed to solve the optimization problem.

Most solutions of the nonlinear equations in engineering have a limitative span, according to which we can initialize the initial guess. The sophisticated search capabilities in local area are improved by changing the parameters in the particle swarm algorithm. And the unnecessary iterations will be cancelled if the value of G meets the standard (such as $G < P$), which can improve computational efficiency.

2. Particle Swarm Algorithm

Particle swarm optimization algorithm (PSO), originating from the study of birds seeking food, is a kind of intelligent optimization algorithm, which is proposed by Eberhart and Kennedy in 1995 [17], and then, in order to promote the explorations in early optimization stages, the inertia weight w is introduced into PSO [18]. Owing to its simple structure, PSO is developing rapidly and has plenty of modified forms.

A modified PSO put forward by Shi can be expressed as

$$v_i^{k+1} = wv_i^k + c_1 r_1 \left(pbest_i^k - x_i^k\right) + c_2 r_2 \left(gbest_i^k - x_i^k\right), \quad (3)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}. \quad (4)$$

Equation (3) is to update the velocity and (4) is to update the new position; v is the velocity and x is the position; w represents the inertia weight; i represents the ith particle and k is the kth generation; c_1 and c_2 denote weighing factors called accelerated coefficients; r_1 and r_2 are random variables uniformly distributed within $[0, 1]$; $pbest_i$ denotes the ith personal best fitness and $gbest_i$ denotes the ith global best fitness; the initial velocity and position of each particle are random variables generated by the standard normal distribution.

3. Improved Particle Swarm Algorithm

3.1. The Selection of Inertia Weight w. Most intelligent optimization algorithms require a large search scope in earlier optimization stages to avoid falling into local optimal value and a fast convergence speed in latter optimization stages to get the optimal value quickly. The inertia weight w is one of the most important factors that affect the search scope and convergence speed of PSO. The search scope will be large when inertia weight w is big and convergence speed will be fast when inertia weight w is small. So it is very important to select the inertia weight w.

In order to meet the demands, the inertia weight w can be expressed by the following function:

$$w = a - c\frac{1}{b^{gbest(i)} + 1} + d\frac{1}{f^{bfc} + 1}, \quad (5)$$

where the inertia weight w varies with the difference between $gbest(i)$ and bfc. a is a parameter, which is a number between 0.8 and 1. b, c, d, and f are parameters selected according to the nonlinear equations. Generally b is a number between 1 and 1.5; c is a number between 0.6 and 1.2; d is a number between 0.05 and 0.2; f is a number between 1 and 2.5. $gbest(i)$ is the ith global best fitness, and bfc is the standard deviation of all the ith generation particles.

During the process of optimization, the inertia weight w becomes smaller and smaller along with $gbest(i)$ becoming smaller and smaller according to the second part of (5). Since solving nonlinear equations is to make (2) equal to zero (or a value close to zero), the lower bound of *gbest* is zero. But if the *gbest* is too big, the inertia weight w is too big to converge, so the *gbest* in (5) has an upper bound *ugbest*. If the *gbest* is bigger than *ugbest*, *gbest* will equal *ugbest* in (5).

If the ith generation particles are scattered, the algorithm will have a large searching scope. However, if the ith generation particles are too centralized, the algorithm will have a small searching scope and may be lost into local optimal value. The standard deviation bfc reflects the distribution of particles. According to the third part of (5), the inertia weight w becomes bigger and bigger along with bfc becoming smaller and smaller. bfc may become zero at last, so the upper bound of inertia weight w is a, and the lower bound of inertia weight w is $a - b/2$, so the inertia weight w can meet the demands.

3.2. Dynamic Conditions of Stopping Iterating. For most intelligent optimization algorithms, there must be enough iterations to guarantee getting the best value. However, the best value can be got through a few iterations for PSO, and then there is many iterations that will be useless, which leads to a low computational efficiency. Besides, if PSO is lost into local optimal value, it may lead to useless iterations and the wrong results. And these problems should be found timely and solved.

Aiming at these problems, a comprehensive plan is proposed.

Assign zero (or a value close to zero) to the standardized fitness value (sfv) and then the iterations will be cancelled if the fitness value is less than or equal to sfv; that is, $gbest(i)$ equals zero. Now the solutions of the nonlinear equations are got and the useless iterations are avoided. That is,

$$gbest(i) \le \text{sfv}. \quad (6)$$

For the problem of being lost into local optimal value, if the difference between $gbest(i)$ and $gbest(i-X)$ and the difference between $positionb(i)$ and $positionb(i-X)$ equal zero, PSO is considered as being lost into local optimal value. That is,

$$\begin{gathered} gbest(i) - gbest(i-X) = 0, \\ positionb(i) - positionb(i-X) = 0, \end{gathered} \quad (7)$$

where $gbest(i)$ is the ith best fitness value and the $gbest(i-X)$ is the best fitness value before X iteration. $positionb(i)$ is the ith best position, and the $positionb(i-X)$ is the best position before X iteration. X is a parameter between 50 and 250 generally.

Equation (6) is the standard in which the optimal values are found. Equations (7) are the standards in which PSO is lost into local optimal value. If (7) are tenable, PSO will restart from the starting.

3.3. The Standardized Number of Restarting Times mb. The standardized number of restarting times *mb* is calculated according to the reliability theory.

The probability of succeeding getting the optimal value for a single PSO is p, which can be calculated through thousands of times of computing, being generally between 0.1 and 1. So the probability of succeeding getting the optimal value p_m before $(m+1)$th restarting can be expressed as

$$\begin{aligned} p_m &= p + p(1-p) + p(1-p)^2 + \cdots + p(1-p)^{m-1} \\ &= 1 - (1-p)^m, \end{aligned} \tag{8}$$

$$\begin{gathered} m = \log_{1-p}^{(1-p_m)}, \\ mb > \lceil m \rceil. \end{gathered} \tag{9}$$

If $(1-p)^m$ is sufficiently small, p_m will be large enough that we can believe that the probability of succeeding getting the optimal value p_m equals 1. *mb* can be calculated through (9).

3.4. The Steps of the Improved PSO

Step 1. Set m equal to 1.

Step 2. Judge whether m is less than *mb*. If m is less than *mb*, the algorithm goes to Step 3. If m is not less than *mb*, the algorithm put out "no results."

Step 3. Initialize the v^1 and x^1 randomly, and calculate the *pbest*(1) and *gbest*(1).

Step 4. Judge whether *gbest*(1) is less than sfv. If *gbest*(1) is less than sfv, the algorithm will end. If *gbest*(1) is not less than sfv, the algorithm goes to Step 5.

Step 5. If the *gbest* is bigger than *ugbest*, *gbest* will equal *ugbest* in (5). If the *gbest* is smaller than or equal to *ugbest*, *gbest* will be *gbest* in (5). Update the inertia weight w according to (5), and then calculate the v^i, x^i, *pbest*(i), and *gbest*(i). Judge whether *gbest*(i) is less than sfv. If *gbest*(i) is less than sfv, the algorithm will end. If *gbest*(1) is not less than sfv, the algorithm goes to Step 6.

Step 6. Judge whether i is less than I (the biggest number of iterations in mth time computing). If i is less than I, the algorithm goes to Step 8. If i is not less than I, the algorithm goes to Step 7.

Step 7. Judge whether the algorithm is lost into local optimal value according to (7). If the algorithm is lost into local optimal value, the algorithm goes to Step 8. If the algorithm is not lost into local optimal value, the algorithm goes to Step 5.

Step 8. Set $m = m + 1$. And then the algorithm goes to Step 2.

And then the imPSO is formed, whose steps are shown in Figure 1.

4. Experiments and Results

In this section, benchmark functions are employed to investigate the performance of the imPSO.

Test 1 (Freudenstein-Roth function). Consider

$$\begin{aligned} f_1(x) = {} & \left[-13 + x_1 + \left(5x_2 - x_2^2 - 2\right)x_2\right]^2 \\ & + \left[-29 + x_1 + \left(x_2^2 + x_2 - 14\right)x_2\right]^2, \\ & -5.12 \le x_i \le 5.12. \end{aligned} \tag{10}$$

The minimum value of the Freudenstein-Roth function is zero when x is located at point $x = (5, 4)$.

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.5$, $d = 0.2$, and $f = 2$. The calculated result is $(5, 4)$, and the calculated minimum value of the Freudenstein-Roth function is zero. The number of iterations is 93. m is 1. The convergence history of test 1 is showed in Figure 2, and the variation of w is shown in Figure 3.

The number of iterations of PSO is 474, whose convergence history is showed in Figure 4.

Test 2 (Rosenbrock function). Consider

$$f_2(x) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2. \tag{11}$$

The Rosenbrock function is a nonconvex pathological function, which has a long narrow and curved valley in the function. So it may require a large number of iterations to obtain the best solution. The minimum value of the Rosenbrock function is zero when x is located at point $x = (1, 1)$.

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.5$, $d = 0.2$, and $f = 2$. The calculated result is $(1, 1)$, and the calculated minimum value of the Rosenbrock function is zero. The number of iterations is 75. m is 1. The convergence history of test 2 is showed in Figure 5, and the variation of w is shown in Figure 6.

The number of iterations of PSO is 435, whose convergence history is showed in Figure 7.

Test 3 (Schaffer function). Consider

$$f_3'(x) = 0.5 - \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}. \tag{12}$$

The Schaffer function has the global maximum value 1 when x is located at point $x = (0, 0)$. However, the Schaffer function has many local maximums 0.9903 at the points around the point $x = (0, 0)$. It is very hard to get the global maximum. Here the Schaffer function is transformed to get the minimum value through the following equation:

$$f_3(x) = 1 - f_3'(x). \tag{13}$$

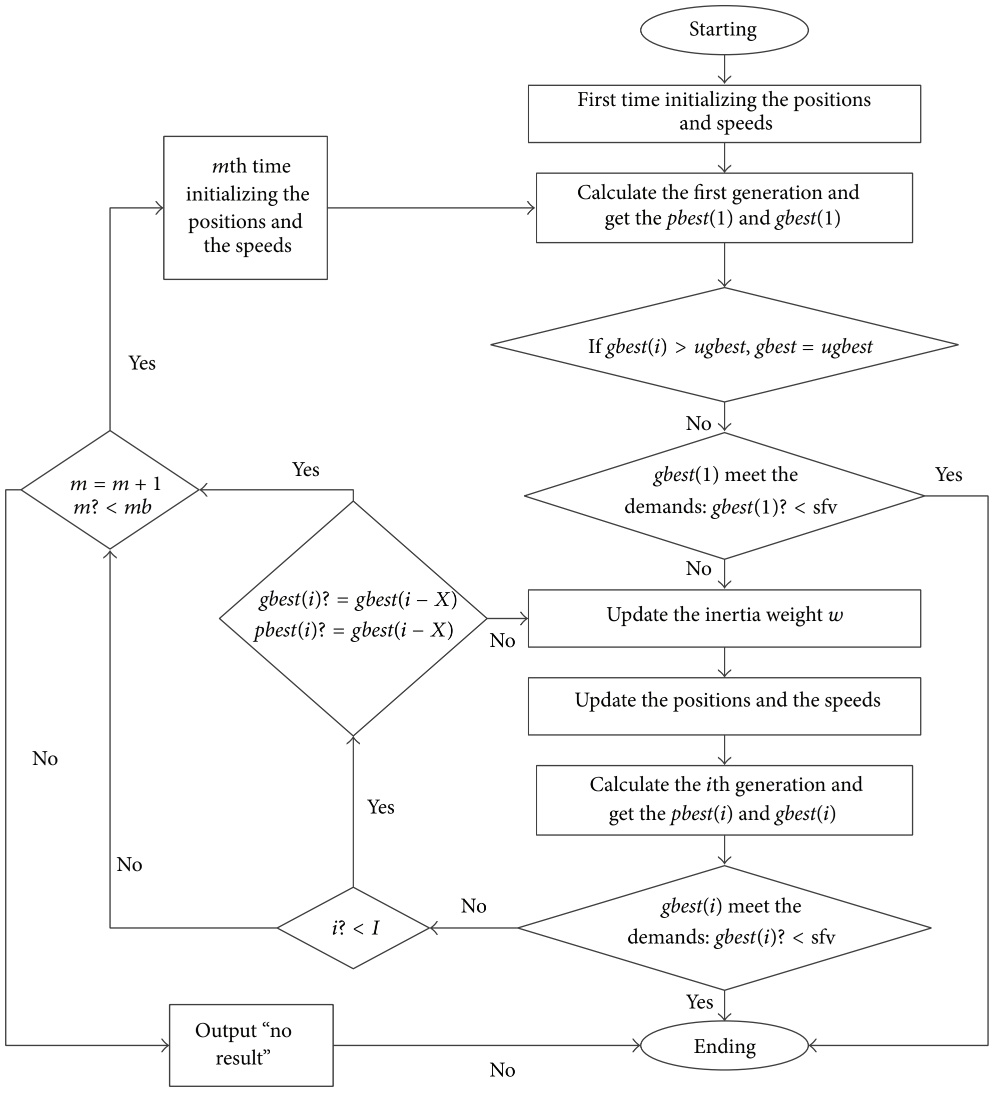
Starting
First time initializing the positions and speeds
Calculate the first generation and get the *pbest*(1) and *gbest*(1)
*m*th time initializing the positions and the speeds
If *gbest*(*i*) > *ugbest*, *gbest* = *ugbest*
Yes
No
gbest(1) meet the demands: *gbest*(1)? < sfv
Yes
m = *m* + 1
m? < *mb*
Yes
No
gbest(*i*)? = *gbest*(*i* − *X*)
pbest(*i*)? = *gbest*(*i* − *X*)
No
Update the inertia weight *w*
Update the positions and the speeds
Calculate the *i*th generation and get the *pbest*(*i*) and *gbest*(*i*)
No
Yes
No
No
i? < *I*
gbest(*i*) meet the demands: *gbest*(*i*)? < sfv
Yes
Output "no result"
Ending
No

Figure 1: Calculation flow chart of imPSO.

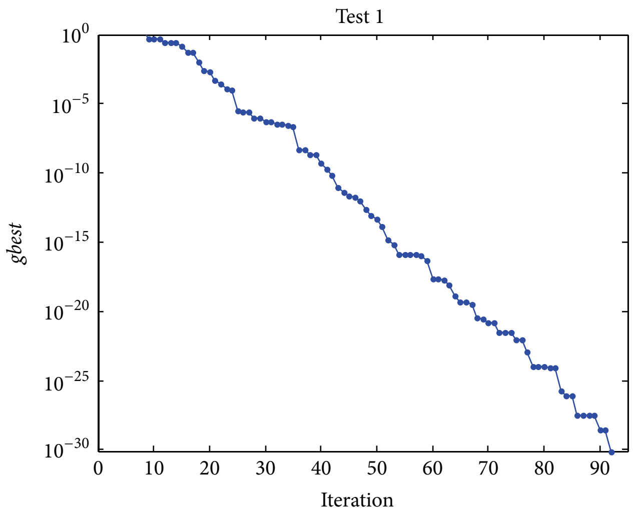

Figure 2: The imPSO convergence history of test 1.

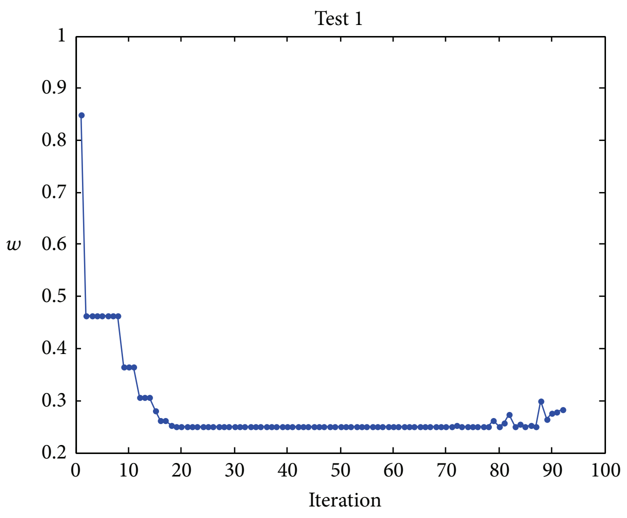

Figure 3: The imPSO variation of w.

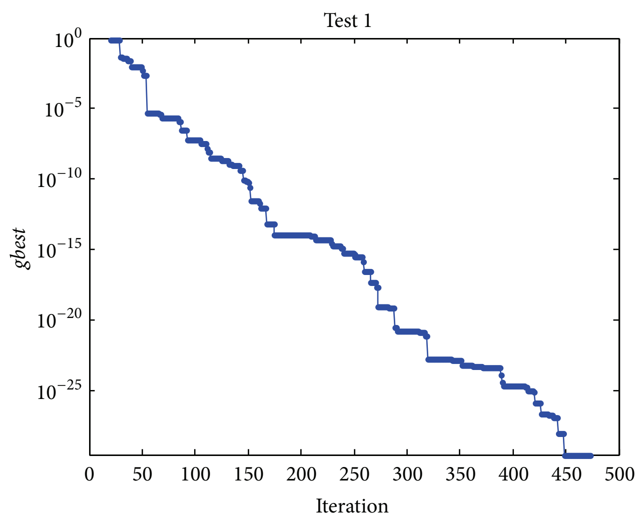

FIGURE 4: The PSO convergence history of test 1.

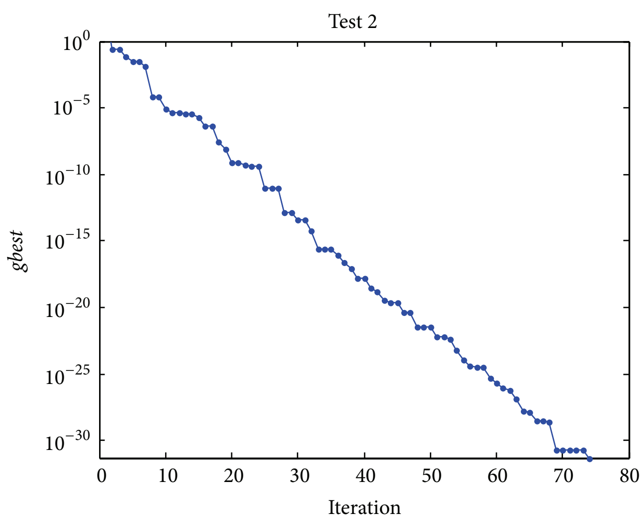

FIGURE 5: The imPSO convergence history of test 2.

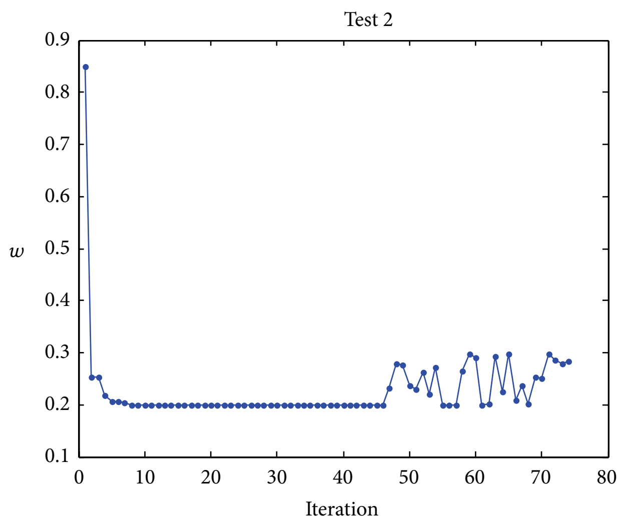

FIGURE 6: The imPSO variation of w.

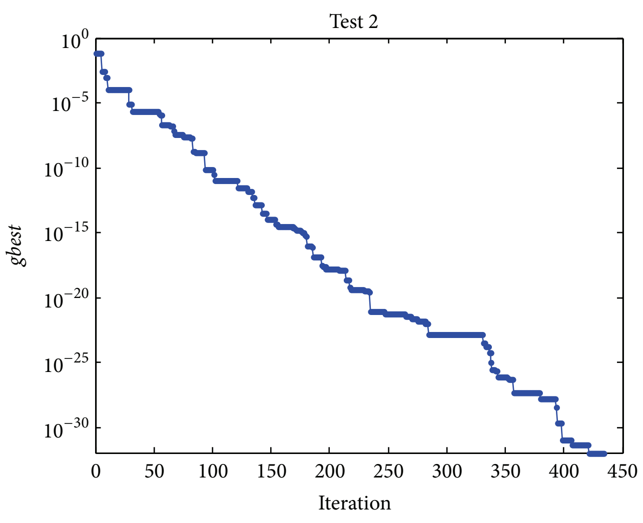

FIGURE 7: The PSO convergence history of test 2.

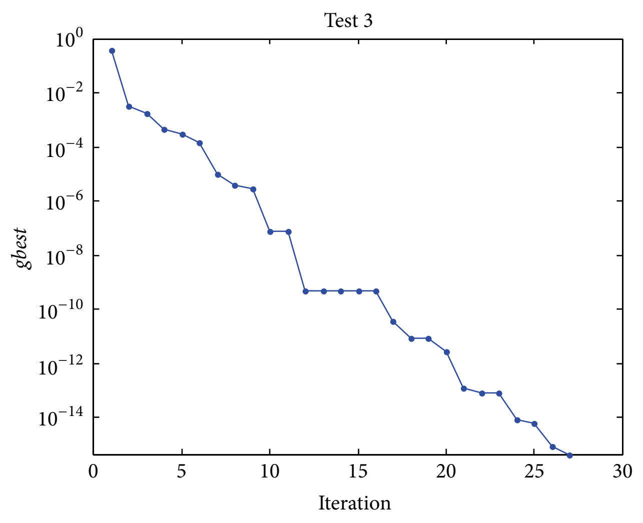

FIGURE 8: The imPSO convergence history of test 3.

So (13) has the minimum value zero at point $x = (0, 0)$.

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.5$, $d = 0.2$, and $f = 2$. The calculated result is $(0, 0)$, and the calculated minimum value of the Schaffer function is zero. The number of iterations is 28. m is 1. The convergence history of test 3 is showed in Figure 8, and the variation of w is shown in Figure 9.

The number of iterations of PSO is 150, whose convergence history is showed in Figure 10.

Test 4 (Powell quartic function). Consider

$$\begin{aligned}\min f_4(x) = {} & (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 \\ & + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4.\end{aligned} \tag{14}$$

The Powell quartic function is singular at the minimum point. However, it is very hard to get the minimum value [14]. The minimum value zero is obtained at the point $x = (0, 0, 0, 0)$.

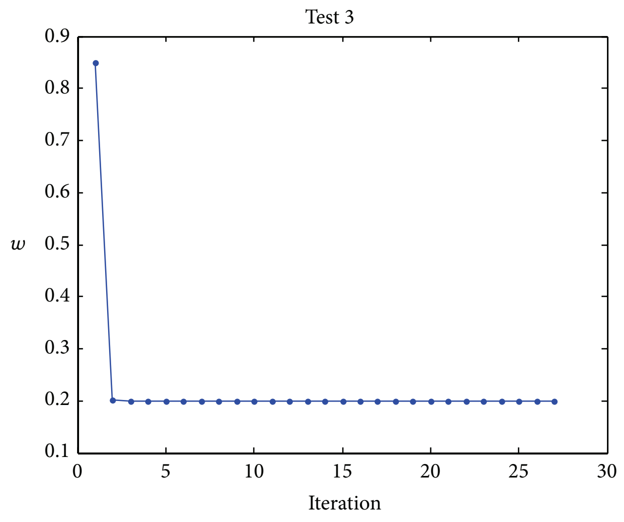

Figure 9: The imPSO variation of w.

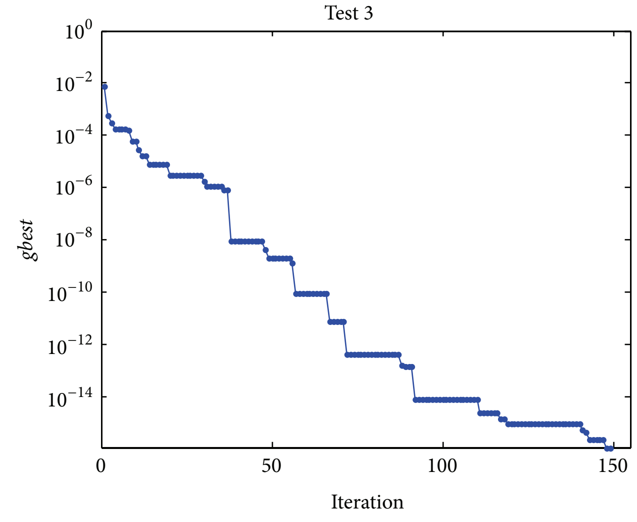

Figure 10: The PSO convergence history of test 3.

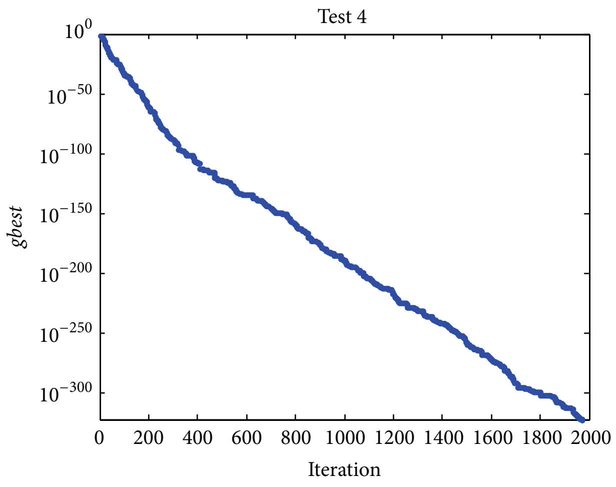

Figure 11: The imPSO convergence history of test 4.

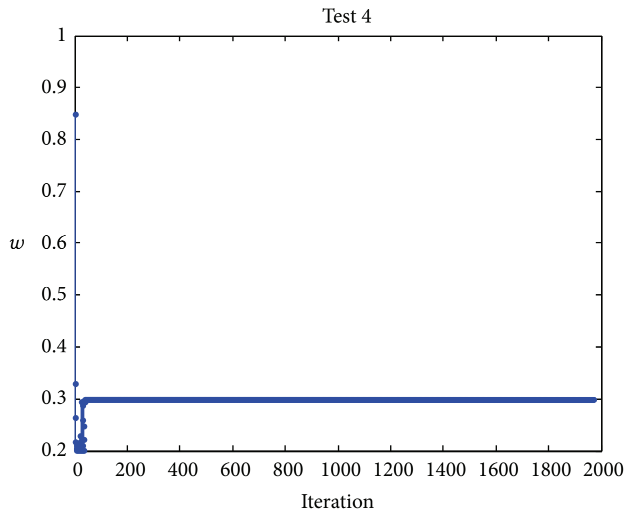

Figure 12: The imPSO variation of w.

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.6$, $d = 0.2$, and $f = 2$. The calculated result is $(0, 0, 0, 0)$, and the calculated minimum value of the Powell quartic function is zero. The number of iterations is 1973. m is 1. The convergence history of test 3 is showed in Figure 11, and the variation of w is shown in Figure 12.

The number of iterations of PSO is 10000 when the minimum value is $1.948645494579354e-315$, whose convergence history is showed in Figure 13.

The number of iterations of PSO in [14] is more than 4000.

Test 5 (Ackley function). Consider

$$f_5(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e, \quad -32 \le x_i \le 32. \tag{15}$$

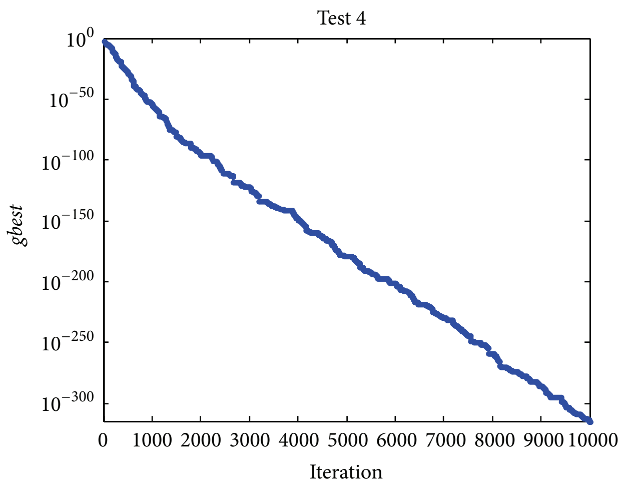

Figure 13: The PSO convergence history of test 4.

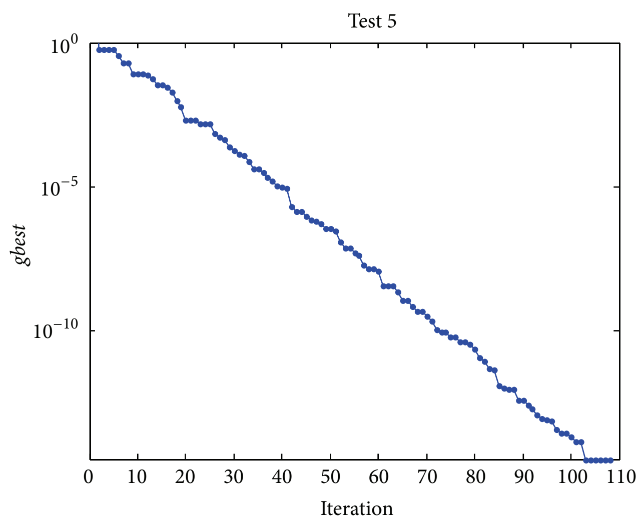

FIGURE 14: The imPSO convergence history of test 5.

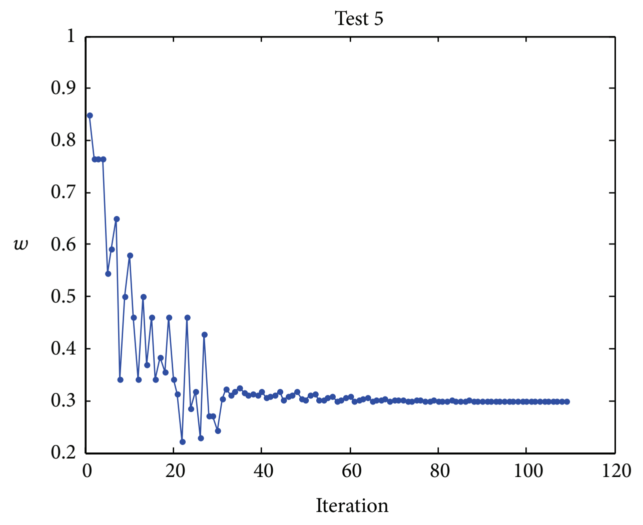

FIGURE 15: The imPSO variation of w.

The dimension here is $n = 4$. Ackley function is a multimodal function, which has a global optimal value and many local optimal values. And the variables have no relevance. It is very easy to be lost into the local optimal values and cannot jump out to find global local optimal values. The minimum value is zero at point $x = (0, 0, 0, 0)$.

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.6$, $d = 0.2$, and $f = 2$. The calculated result is $(-0.312039464691632e-015, 0.386551762889035e-015, 0.037866869728015e-015, 0.089018775346847e-015)$, and the calculated minimum value of the Powell quartic function is zero. The number of iterations is 109. m is 1. The convergence history of test 3 is showed in Figure 14, and the variation of w is shown in Figure 15.

The number of iterations of PSO is 9500 when the minimum value is $3.552713678800501e-015$, whose convergence history is showed in Figure 16.

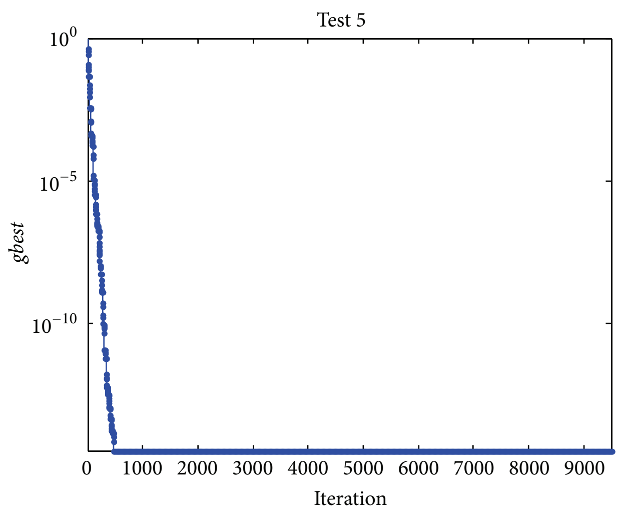

FIGURE 16: The PSO convergence history of test 5.

Test 6 (Griewank function). Consider

$$f_6(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \qquad -600 \leq x_i \leq 600. \tag{16}$$

The dimension here is $n = 6$. Griewank function is a multimodal function, which has a global optimal value and many local optimal values. And the variables are relevant. It is very easy to be lost into the local optimal values and cannot jump out to find global local optimal values. The minimum value is zero at point $x = (0, 0, 0, 0, 0, 0)$.

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.6$, $d = 0.2$, and $f = 2$. The calculated result is $(-0.095852536988518e-007, 0.077579964471561e-007, 0.011250486244803e-007, 0.077788935476922e-007, -0.216911551137599e-007, 0.152876829678368e-007)$, and the calculated minimum value of the Powell quartic function is zero. The number of iterations is 49. m is 1. The convergence history of test 3 is showed in Figure 17, and the variation of w is shown in Figure 18.

The number of iterations of PSO is 9500 when the minimum value is $3.552713678800501e-015$, whose convergence history is showed in Figure 19.

Table 1 shows the results of each test with 10000 times computing. The value of mb is 100.

Table 1 shows the total iteration times of each test with 10000 times computing, from which the mean iteration times can be calculated. Value of $(1-p)^{\text{biggest } m}$ indicated that if the mb is big enough, systems of nonlinear equations can be solved successfully.

5. Case Study

In order to demonstrate the efficiency of the improved particle swarm optimization algorithm for solving systems of nonlinear equations, some standard nonlinear equations systems are employed.

Table 1

Test	Total iteration	Total restarting times m	p	The biggest m	$(1-p)^{\text{biggest } m}$	Fail ($m \geq mb$)
Test 1	1148836	10967	0.9110	6	5.314×10^{-13}	0
Test 2	756972	10000	1	1	0	0
Test 3	379051	11149	0.8964	5	1.193×10^{-05}	0
Test 4	19810789	10000	1	1	0	0
Test 5	1169838	20067	0.4957	18	1.635×10^{-05}	0
Test 6	724026	11627	0.8597	6	7.6269×10^{-06}	0

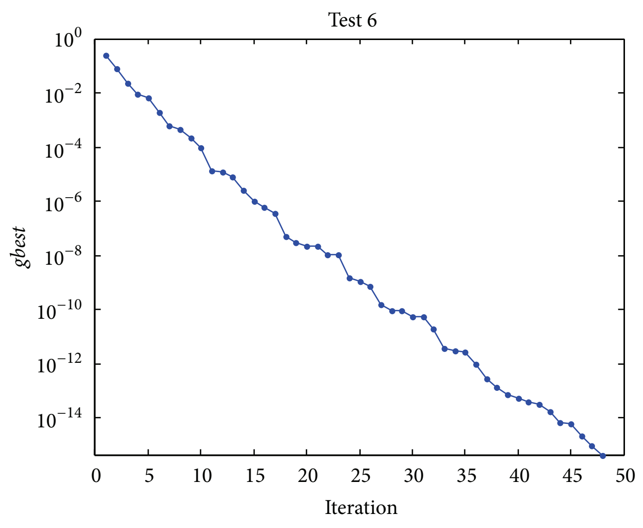

Figure 17: The imPSO convergence history of test 6.

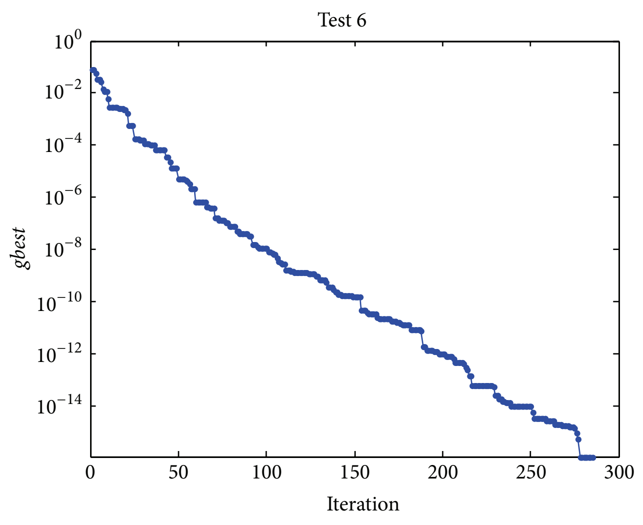

Figure 19: The PSO convergence history of test 6.

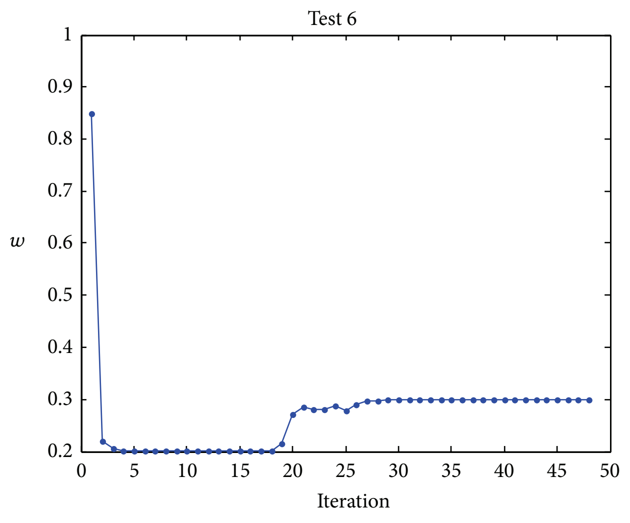

Figure 18: The imPSO variation of w.

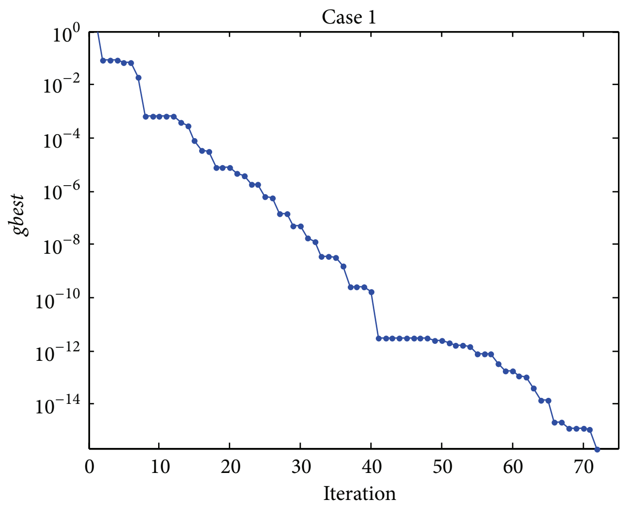

Figure 20: The imPSO convergence history of Case 1.

Case 1 (see [19]). Consider

$$x_1^3 - 3x_1x_2^2 - 1 = 0,$$
$$3x_1^2x_2 - x_2^3 + 1 = 0. \tag{17}$$

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.6$, $d = 0.2$, and $f = 2$. The calculated result is (−0.290514555507251, 1.084215081491351), and the number of iterations is 69, while the number of iterations in [14] is more than 100. m is 1. The convergence history of Case 1 is shown in Figure 20, and the variation of w is shown in Figure 21. The result is more accurate than the result in [19].

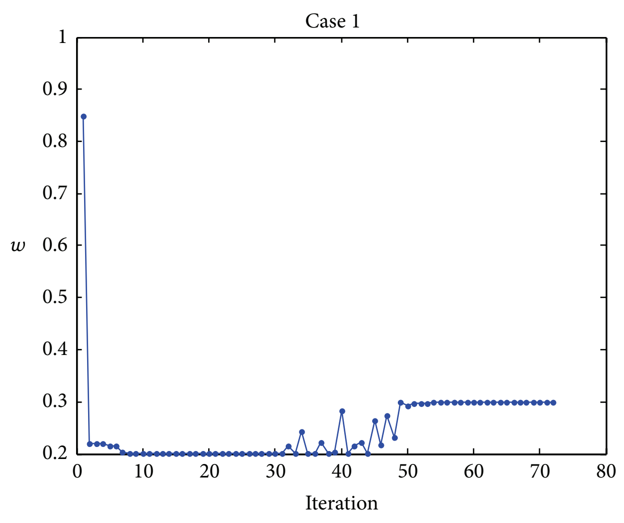

FIGURE 21: The imPSO variation of w.

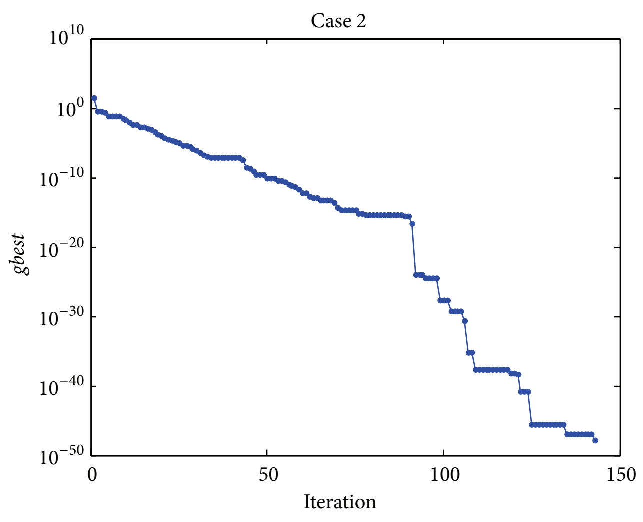

FIGURE 22: The imPSO convergence history of Case 2.

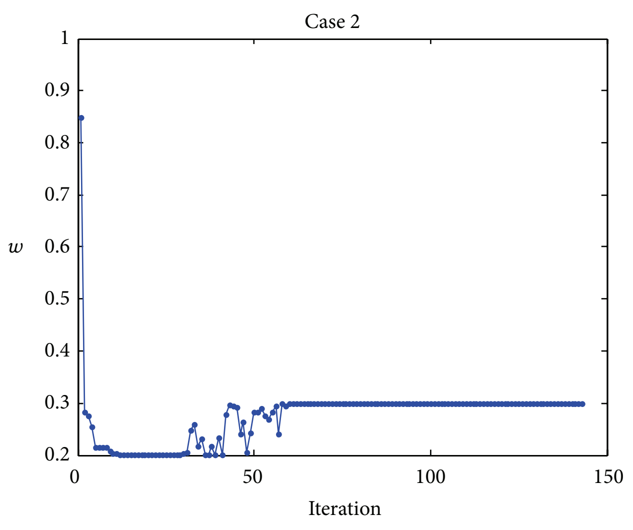

FIGURE 23: The imPSO variation of w.

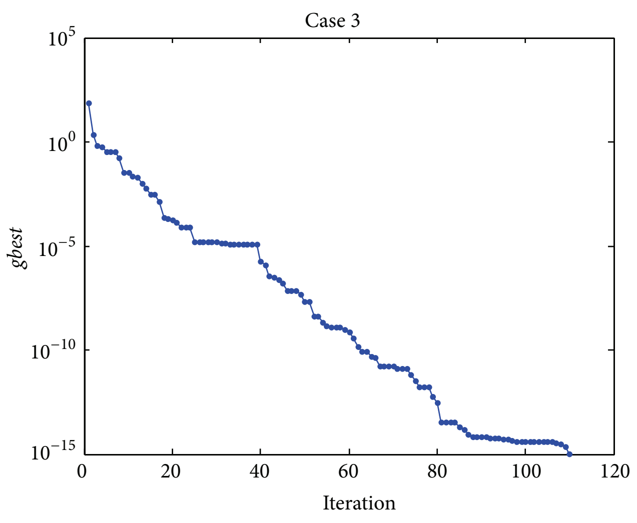

FIGURE 24: The imPSO convergence history of Case 3.

Case 2 (see [20]). Consider

$$\begin{aligned} e^{x_1^2} - 8x_1 \sin(x_2) &= 0, \\ x_1 + x_2 - 1 &= 0, \\ (x_3 - 1)^3 &= 0. \end{aligned} \tag{18}$$

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.6$, $d = 0.2$, and $f = 2$. The calculated result is (0.175598924177659, 0.824401075822341, 1.000000000000000), and the number of iterations is 147. m is 1. The convergence history of Case 2 is shown in Figure 22, and the variation of w is shown in Figure 23. The result is more accurate than the result in [14]. The result in [14] is (0.17559892417766, 0.82440107582234, 1).

Case 3 (see [20]). Consider

$$\begin{aligned} 3x_1 - \cos(x_2 x_3) - 0.5 &= 0, \\ x_1^2 - 625x_2^2 - 0.25 &= 0, \\ e^{-x_1 x_2} + 20x_3 + \frac{(10\pi - 3)}{3} &= 0. \end{aligned} \tag{19}$$

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.6$, $d = 0.2$, and $f = 2$. The calculated result is (0.500000000000000, −0.000000000141655, −0.523598775601840), and the number of iterations is 111, while the number of iterations in [14] is more than 150. m is 1, and the variation of w is shown in Figure 25.

The convergence history of Case 3 is shown in Figure 24. Table 2 shows the results.

Table 2

PPSO [14]		imPSO	
x	$f(x)$	x	$f(x)$
0.5	0	0.500000000000000	0
0	0	−0.000000000141655	0
−0.52359877559662	$3.357669697834353e-11$	−0.523598775601840	$5.329070518200751e-15$

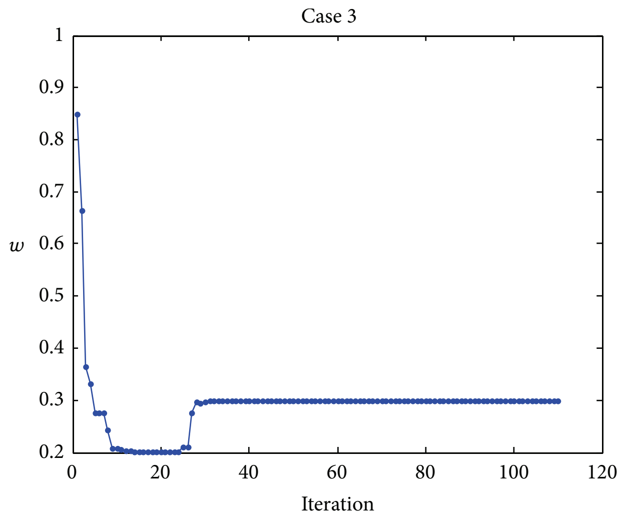

Figure 25: The imPSO variation of w.

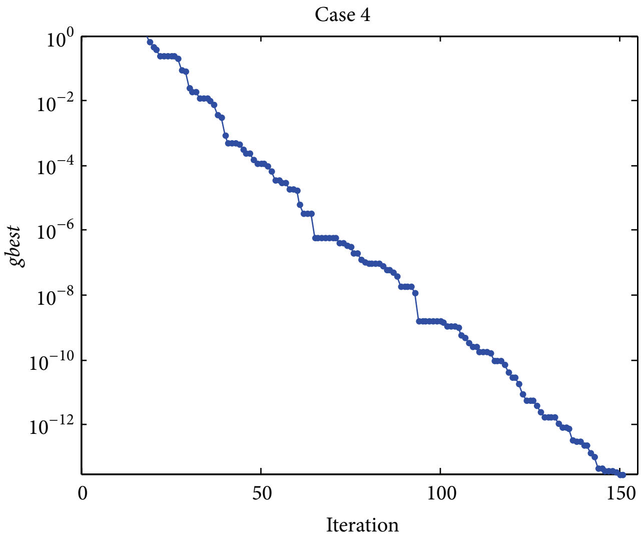

Figure 26: The imPSO convergence history of Case 4.

Case 4 (see [13]). Consider

$$x_1^{x_2}+x_2^{x_1}-5x_1x_2x_3=85,$$
$$x_1^3-x_2^{x_3}-x_3^{x_2}=60, \quad (20)$$
$$x_1^{x_3}+x_3^{x_1}-x_2=2.$$

The values of parameters are $a=1$, $b=1.8$, $c=1.6$, $d=0.2$, and $f=2$. The calculated result is (4.00000000000000, 3.00000000000000, 1.00000000000000), and the number of iterations is 152, while the number of iterations in [14, 16] is more than 200. m is 3. The convergence history of Case 4 is shown in Figure 26, and the variation of w is shown in Figure 27.

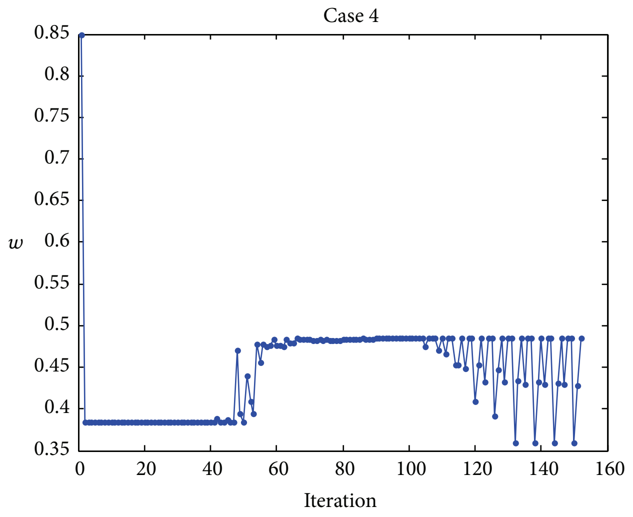

Figure 27: The imPSO variation of w.

Case 5 (see [21]). Consider

$$f_1(x)=bh-(b-2t)(h-2t)=165,$$
$$f_2(x)=\frac{bh^3}{12}-\frac{(b-2t)(h-2t)^3}{12}=9369, \quad (21)$$
$$f_3(x)=\frac{2t(h-t)^2(b-t)^2}{h+b-2t}=6835.$$

The values of parameters are $a=1$, $b=1.8$, $c=1.6$, $d=0.2$, and $f=2$. The calculated result is (12.256519599348696, 22.894938623626285, 2.789817919538154) or (−12.256519599348696, −22.894938623626285, −2.789817919538154), and the number of iterations is 148, while the number of iterations in [14] is more than 200. m is 4. The convergence history of Case 5 is showed in Figure 28, and the variation of w is shown in Figure 29.

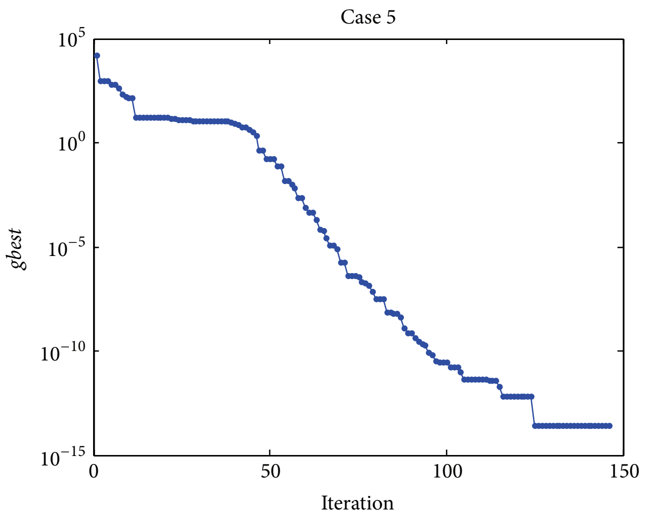

FIGURE 28: The imPSO convergence history of Case 5.

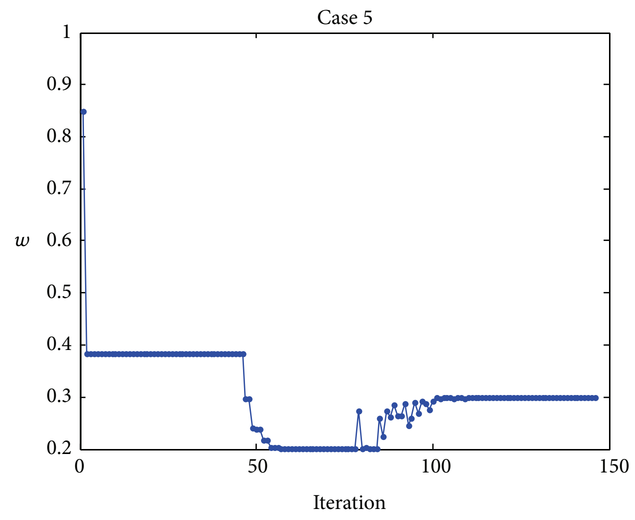

FIGURE 29: The imPSO variation of w.

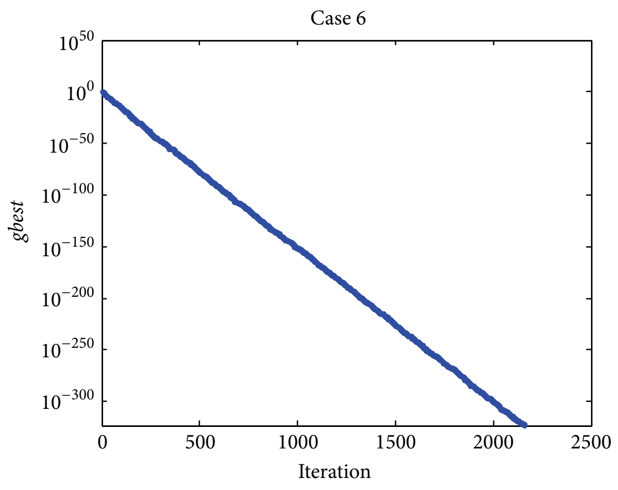

FIGURE 30: The imPSO convergence history of Case 6.

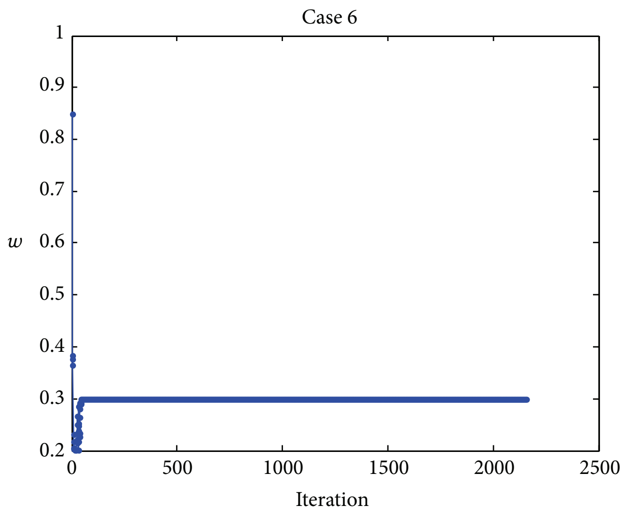

FIGURE 31: The imPSO variation of w.

Case 6 (neurophysiology application). Consider

$$\begin{aligned} x_1^2 + x_3^2 &= 1, \\ x_2^2 + x_4^2 &= 1, \\ x_5 x_3^3 + x_6 x_4^3 &= 0, \\ x_5 x_1^3 + x_6 x_2^3 &= 0, \\ x_5 x_1 x_3^2 + x_6 x_2 x_4^2 &= 0, \\ x_5 x_1^2 x_3 + x_6 x_2^2 x_4 &= 0. \end{aligned} \tag{22}$$

The values of parameters are $a = 1$, $b = 1.8$, $c = 1.4$, $d = 0.2$, and $f = 1.05$. The calculated result with imPSO is $(-0.994926998030712, 0.539699980991932, -0.100599545672903, 0.841857428854381, 0, 0)$, and the number of iterations is 2161. m is 1. The convergence history of Case 6 is showed in Figure 30, and the variation of w is shown in Figure 31. The best results in different methods and the results of imPSO are shown in Table 3.

Table 4 shows the results of Case 1 with 10000 times computing under different parameters. The value of mb is 100 and the values of parameters are $b = 1.8$ and $f = 1.05$.

According to (1) and (2), total iterations become fewer when parameter d is bigger, but the probability of convergence becomes bigger. According to (3) and (4), total iterations become fewer when parameter c is smaller, but the probability of convergence becomes bigger. According to (5) and (6), total iterations become fewer when parameter a is smaller, but the probability of convergence becomes smaller. Table 4 shows that total iterations become fewer when parameter $a - c/2$ is less, but the probability of convergence becomes smaller.

Table 3: Different results with different methods of Case 6.

The results in [15]		The results of ICA [16]		The best results of imPSO	
x	$f(x)$	x	$f(x)$	x	$f(x)$
−0.8078668904	0.0050092197	−0.041096050919063	0	−0.994926998030712	0
−0.9560562726	0.0366973076	0.041096050919063	0	0.539699980991932	0
0.5850998782	0.0124852708	0.999155200456294	0	−0.100599545672903	0
−0.2219439027	0.0276342907	−0.999155200456294	0	0.841857428854381	0
0.0620152964	0.0168784849	0.098733550533454	0	0	0
−0.0057942792	0.0248569233	0.098733550533454	0	0	0

Table 4

	Parameters	Total iteration	Total restarting times m	p	Biggest m	$(1-p)^{\text{biggest } m}$	Fail ($m \geq mb$)
(1)	$a = 1.0, c = 1.6, d = 0.4$	815371	14955	0.6697	11	5.104×10^{-06}	0
(2)	$a = 1.0, c = 1.6, d = 0.2$	756348	15011	0.6664	9	5.117×10^{-05}	0
(3)	$a = 1.0, c = 1.6, d = 0.2$	755928	15009	0.6673	8	1.501×10^{-04}	0
(4)	$a = 1.0, c = 1.4, d = 0.2$	1006006	14946	0.6711	10	1.481×10^{-05}	0
(5)	$a = 1.0, c = 1.4, d = 0.2$	1005359	14942	0.6726	8	1.320×10^{-04}	0
(6)	$a = 0.8, c = 1.4, d = 0.2$	550127	15074	0.6656	8	1.481×10^{-05}	0

6. Conclusions

In this paper, a method for solving a system of nonlinear equations is proposed, which is converted to an optimization problem, and an improved particle swarm algorithm is employed to solve the optimization problem. This method overcomes the dependence on reasonable initial guess of the solution. And then some standard systems of nonlinear equations are presented to demonstrate the convergence probability, convergence rate, and solution precision in finding the best solution of nonlinear equations with the improved particle swarm algorithm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The project is subsidized by The Research Project Supported by Shanxi Scholarship Council of China 2014-065 and Shanxi Province Natural Science Fund Project 2014011024-5.

References

[1] B. M. Barbashov, V. V. Nesterenko, and A. M. Chervyakov, "General solutions of nonlinear equations in the geometric theory of the relativistic string," *Communications in Mathematical Physics*, vol. 84, no. 4, pp. 471–481, 1982.

[2] S. R. Bickham, S. A. Kiselev, and A. J. Sievers, "Stationary and moving intrinsic localized modes in one-dimensional monatomic lattices with cubic and quartic anharmonicity," *Physical Review B*, vol. 47, no. 21, pp. 14206–14211, 1993.

[3] M. Khalilzadeh, F. Kianfar, A. S. Chaleshtari, S. Shadrokh, and M. Ranjbar, "A modified PSO algorithm for minimizing the total costs of resources in MRCPSP," *Mathematical Problems in Engineering*, vol. 2012, Article ID 365697, 18 pages, 2012.

[4] R. K. Sahu, S. Panda, and G. T. C. Sekhar, "A novel hybrid PSO-PS optimized fuzzy PI controller for AGC in multi area interconnected power systems," *International Journal of Electrical Power & Energy Systems*, vol. 64, pp. 880–893, 2015.

[5] Y.-Z. Hsieh, M.-C. Su, and P.-C. Wang, "A PSO-based rule extractor for medical diagnosis," *Journal of Biomedical Informatics*, vol. 49, pp. 53–60, 2014.

[6] A. Holstad, "Numerical solution of nonlinear equations in chemical speciation calculations," *Computational Geosciences*, vol. 3, no. 3-4, pp. 229–257, 1999.

[7] I. K. Argyros, "On the solution of undetermined systems of nonlinear equations in Euclidean spaces," *Pure Mathematics and Applications*, vol. 4, no. 2, pp. 199–209, 1993.

[8] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, vol. 30, SIAM, 2000.

[9] S. Krzyworzcka, "Extension of the Lanczos and CGS methods to systems of nonlinear equations," *Journal of Computational and Applied Mathematics*, vol. 69, no. 1, pp. 181–190, 1996.

[10] S. J. Wright and J. Nocedal, *Numerical optimization*, vol. 2, Springer, New York, NY, USA, 1999.

[11] J. Kou, Y. Li, and X. Wang, "A composite fourth-order iterative method for solving non-linear equations," *Applied Mathematics and Computation*, vol. 184, no. 2, pp. 471–475, 2007.

[12] Y.-Z. Luo, G.-J. Tang, and L.-N. Zhou, "Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-Newton method," *Applied Soft Computing Journal*, vol. 8, no. 2, pp. 1068–1073, 2008.

[13] Y. Mo, H. Liu, and Q. Wang, "Conjugate direction particle swarm optimization solving systems of nonlinear equations," *Computers & Mathematics with Applications*, vol. 57, no. 11-12, pp. 1877–1882, 2009.

[14] M. Jaberipour, E. Khorram, and B. Karimi, "Particle swarm algorithm for solving systems of nonlinear equations," *Computers & Mathematics with Applications*, vol. 62, no. 2, pp. 566–576, 2011.

[15] C. Grosan and A. Abraham, “A new approach for solving nonlinear equations systems,” *IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans*, vol. 38, no. 3, pp. 698–714, 2008.

[16] M. Abdollahi, A. Isazadeh, and D. Abdollahi, “Imperialist competitive algorithm for solving systems of nonlinear equations,” *Computers & Mathematics with Applications*, vol. 65, no. 12, pp. 1894–1908, 2013.

[17] R. C. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Proceedings of the 6th International Symposium on Micro Machine and Human Science*, vol. 1, pp. 39–43, October 1995.

[18] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC ’98)*, pp. 69–73, IEEE, Anchorage, Alaska, USA, May 1998.

[19] G. H. Nedzhibov, “A family of multi-point iterative methods for solving systems of nonlinear equations,” *Journal of Computational and Applied Mathematics*, vol. 222, no. 2, pp. 244–250, 2008.

[20] J. L. Hueso, E. Martínez, and J. R. Torregrosa, “Modified Newton’s method for systems of nonlinear equations with singular Jacobian,” *Journal of Computational and Applied Mathematics*, vol. 224, no. 1, pp. 77–83, 2009.

[21] C. Wang, R. Luo, K. Wu, and B. Han, “A new filled function method for an unconstrained nonlinear equation,” *Journal of Computational and Applied Mathematics*, vol. 235, no. 6, pp. 1689–1699, 2011.