Implementasi Algoritma *Particle Swarm* untuk Menyelesaikan Sistem Persamaan Nonlinear

Ardiana Rosita, Yudhi Purwananto dan Rully Soelaiman
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111
E-mail: yudhi@if.its.ac.id

***Abstrak*—Penyelesaian sistem persamaan nonlinear merupakan salah satu permasalahan yang sulit pada komputasi numerik dan berbagai aplikasi teknik. Beberapa metode telah dikembangkan untuk menyelesaikan sistem persamaan ini dan metode Newton merupakan metode yang paling sering digunakan. Namun metode ini memerlukan perkiraan solusi awal dan memilih perkiraan solusi awal yang baik untuk sebagian besar sistem persamaan nonlinear tidaklah mudah. Pada makalah ini, algoritma *Particle Swarm* yang diusulkan oleh Jaberipour dan kawan-kawan[1] diimplementasikan. Algoritma ini merupakan pengembangan dari algoritma *Particle Swarm Optimization* (PSO). Algoritma ini meyelesaikan sistem persamaan nonlinear yang sebelumnya telah diubah menjadi permasalahan optimasi. Uji coba dilakukan terhadap beberapa fungsi dan sistem persamaan nonlinear untuk menguji kinerja dan efisiensi algoritma. Berdasarkan hasil uji coba, beberapa fungsi dan sistem persamaan nonlinear telah konvergen pada iterasi ke 10 sampai 20 dan terdapat fungsi yang konvergen pada iterasi ke 200. Selain itu, solusi yang dihasilkan algoritma *Particle Swarm* mendekati solusi eksak.**

***Kata Kunci*—fungsi, optimasi, *particle swarm*, sistem persamaan nonlinear.**

I. PENDAHULUAN

SISTEM persamaan nonlinear adalah sistem persamaan yang terdiri dari satu atau lebih persamaan nonlinear. Sistem persamaan ini merupakan permasalahan dasar matematika yang banyak dijumpai di berbagai bidang ilmu alam seperti fisika, kimia, komputasi mesin, dan lainnya. Sistem persamaan nonlinear sangat sulit untuk diselesaikan, terlebih pada komputasi numerik dan berbagai aplikasi teknik.

Terdapat banyak metode maupun algoritma yang dikembangkan untuk menyelesaikan sistem persamaan nonlinear. Apabila dibandingkan dengan semua metode yang ada, metode Newton merupakan metode yang paling sering digunakan pada penyelesaian sistem persamaan nonlinear. Namun metode ini memiliki beberapa kekurangan, yaitu tingkat konvergensi dan hasilnya sangat bergantung pada perkiraan awal solusi. Algoritma akan gagal apabila pemilihan nilai perkiraan solusi awal tidak tepat, sedangkan menentukan perkiraan solusi awal yang baik pada sebagian besar sistem persamaan nonlinear tidaklah mudah. Oleh karena itu, dibutuhkan sebuah metode baru yang mampu mengatasi permasalahan tersebut.

Pada makalah ini algoritma optimasi *particle swarm* yang diusulkan oleh Jaberipour et al.[1] diimplementasikan. Algoritma optimasi *Particle Swarm* baru ini merupakan pengembangan dari algoritma optimasi yang terkenal, yaitu algoritma *Particle Swarm Optimization* (PSO) dasar[2]. Algoritma *Particle Swarm* yang diusulkan oleh Jaberipour menyelesaikan sistem persamaan nonlinear yang sebelumnya telah diubah menjadi bentuk permasalahan optimasi, dalam hal ini adalah minimasi. Dengan menggunakan algoritma ini, diharapkan hasil penyelesaian yang didapat efisien dengan rata-rata konvergensi yang tinggi.

II. ALGORTIMA *PARTICLE SWARM* UNTUK MENYELESAIKAN SISTEM PERSAMAAN NONLINEAR

Pada makalah ini algoritma optimasi *particle swarm* digunakan untuk menyelesaikan sistem persamaan nonlinear. Sebelum membahas lebih lanjut tentang algoritma, langkah awal yang dilakukan adalah sistem persamaan nonlinear diubah menjadi permasalahan optimasi. Bentuk sistem persamaannya sebagai berikut :

$$\begin{aligned} &f_1(x_1, x_2, ..., x_n) = 0 \\ &f_2(x_1, x_2, ..., x_n) = 0 \\ &: \\ &f_m(x_1, x_2, ..., x_n) = 0. \end{aligned} \tag{1}$$

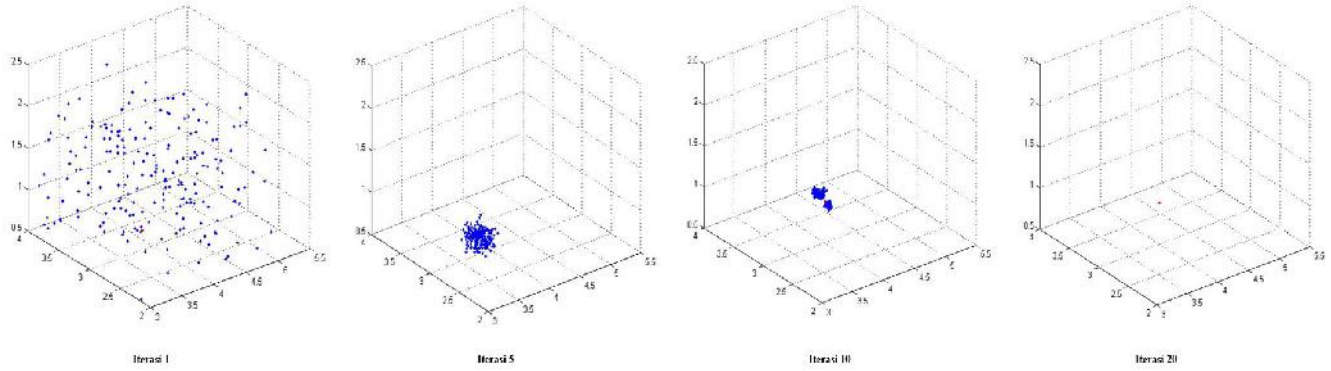
Agar dapat menggunakan metode optimasi global, sistem persamaan (1) diubah menjadi fungsi kuadratik sebagai berikut:

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

Untuk menyelesaikan sistem persamaan nonlinear sama dengan meminimalkan fungsi (2).

A. Algoritma Particle Swarm Optimization (PSO)

Algoritma *Particle Swarm Optimization* (PSO) adalah teknik optimasi berdasarkan populasi yang terinspirasi oleh perilaku sosial dari pergerakan burung atau ikan (*bird flocking* atau *fish schooling*). PSO sebagai alat optimasi menyediakan prosedur pencarian berbasis populasi dimana masing-masing individu yang disebut partikel mengubah posisi mereka terhadap waktu. Pada sistem PSO, masing-masing partikel terbang mengitari ruang pencarian multi dimensional



Gambar. 1. Visualisasi perubahan posisi populasi saat mencari posisi terbaik

(*multidimensional search space*) dan menyesuaikan posisinya berdasarkan pengalaman pribadinya dan pengalaman partikel di sebelahnya. Visualisasi pergerakan populasi dalam mencari posisi terbaik ditampilkan pada Gambar 1. Dari penjelasan di atas dapat dikatakan bahwa algoritma PSO menggabungkan metode pencarian lokal (*local search*) dengan metode pencarian global (*global search*)[3].

Tiap partikel memiliki posisi $x_i = (x_{i1}, x_{i2}, ..., x_{iN})$ dan kecepatan $v_i = (v_{i1}, v_{i2}, ..., v_{iN})$ pada ruang pencarian berdimensi *N*, dimana *i* menyatakan partikel ke-*i* dan *N* menyatakan dimensi ruang pencarian atau jumlah variabel yang belum diketahui pada sistem persamaan nonlinear. Inisialisasi algoritma PSO dimulai dengan menetapkan posisi awal partikel secara acak (solusi) dan kemudian mencari nilai optimal dengan memperbarui posisinya. Seperti yang telah dijelaskan di atas, setiap iterasi masing-masing partikel memperbarui posisinya mengikuti dua nilai terbaik, yaitu solusi terbaik yang telah didapat oleh masing-masing partikel (pbest) dan solusi terbaik pada populasi (gbest). Setelah mendapatkan dua nilai terbaik, posisi dan kecepatan partikel diperbarui dengan menggunakan persamaan berikut :

$$v_k^i = wv_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest^k - x_i^k)$$
$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (3)$$

dimana v_i^k adalah kecepatan partikel ke *i* pada iterasi ke *k*, dan x_i^k adalah solusi (posisi) partikel ke *i* pada iterasi ke *k*. c_1, c_2 adalah konstanta positif, dan r_1, r_2 adalah dua variabel acak terdistribusi *uniform* antara 0 sampai 1. Pada persamaan di atas, *w* adalah bobot inersi yang menunjukkan pengaruh perubahan kecepatan dari vektor lama ke vektor yang baru.

B. Algoritma Particle Swarm yang diusulkan oleh Jaberipour

Algoritma PSO biasanya konvergen dengan cepat saat pencarian awal dan semakin lama semakin melambat. Selain itu, variabel *w*, c_1, dan c_2 merupakan faktor kritis yang mempengaruhi konvergensi algoritma PSO[4],[5]. Untuk mengatasi permasalahan tersebut maka penulis mengimplementasikan algoritma baru yaitu algoritma *Particle Swarm* yang diusulkan oleh Jaberipour dan kawan-kawan. Perbedaan utama antara algoritma ini dengan algoritma *Particle Swarm* dasar yaitu pada saat memperbarui masing-masing partikel. Pada algoritma ini, posisi dan kecepatan partikel diperbarui dengan menggunakan persamaan sebagai berikut :

$$v_i^{k+1} = (2r_1 - 0.5)v_i^k + (2r_2 - 0.5)(pbest_i^k - x_i^k) + (2r_3 - 0.5)(gbest^k - x_i^k) \quad (4)$$

$$w^{k+1} = (2r_4 - 0.5)(gbest^k - pbest_i^k) + (2r_5 - 0.5)(gbest^k - x_i^k)$$

$$x_i^{k+1} = pbest_i^k + (2r_6 - 0.5)v_i^{k+1} + (2r_7 - 0.5)w^{k+1} \quad (5)$$

dimana r_1, r_2, r_3, r_4, r_5, r_6, dan r_7 adalah variabel nilai acak antara 0 dan 1. Langkah-langkah algoritma *Particle Swarm* yang diusulkan oleh Jaberipour ditunjukkan sebagai berikut :

1. Inisialisasi populasi dan kecepatan secara acak
2. Menghitung nilai *fitness* masing-masing partikel dan menentukan pbest dan gbest. Nilai *fitness* awal dari masing-masing $pbest_i$ sama dengan nilai *fitness* posisi awal partikel
3. Mengubah kecepatan partikel dengan menggunakan persamaan (4)
4. Memindahkan masing-masing partikel ke posisi yang baru menggunakan persamaan (5)
5. Membandingkan nilai *fitness* masing-masing partikel dengan pbest. Apabilai nilai *fitness* partikel ke *i* lebih kecil daripada nilai *fitness* $pbest_i$, maka nilai $pbest_i$ digantikan oleh partikel ke *i*. Nilai $pbest_i$ kemudian disimpan pada matriks M_{mxn}, dimana *m* adalah jumlah partikel dan *n* menunjukkan jumlah variabel yang belum diketahui
6. Nilai $pbest_i$ terburuk diantara partikel terbaik pada matriks M dipilih sebagai $pworst_i$
7. Memilih secara acak salah satu komponen $pworst_i$, yaitu *l*, dan memperbarui nilainya dengan persamaan berikut :

$$pworst_{i,l}^{new} = pworst_{i,l} + (2r_8 - 0.5)\frac{\frac{\partial f}{\partial pworst_{i,l}}(pworst_i)}{x_U(l) - x_L(l)} \quad (6)$$

dimana r_8 adalah nilai acak antara 0 dan 1, x_U adalah batas atas dan x_L adalah batas bawah dari tiap variabel. Pada beberapa kasus persamaan (6) terlalu rumit untuk diselesaikan. Untuk mengatasi permasalahan tersebut, *finite differencing* adalah sebuah algoritma yang digunakan untuk menghitung perkiraan turunan yang terinspirasi oleh

teorema Taylor. Perkiraan turunan dengan nilai akurasi tinggi dapat ditentukan dengan menggunakan persamaan *central difference*, yaitu sebagai berikut :

$$\frac{\partial f}{\partial x_i}(x) \approx \frac{f(x + Ee_i) - f(x - Ee_i)}{2E} \quad (7)$$

dimana f adalah fungsi objektif, e_i adalah unit vektor dari $pworst_i$, dan E bernilai 10^{-8}. Oleh karena itu persamaan (6) dapat diperbaiki sebagai berikut :

$$cd = \frac{f(pworst_i + Ee_i) - f(pworst_i - Ee_i)}{2E(x_U(l) - x_L(l))}$$

$$pworst_{i,l}^{new} = pworst_{i,l} + (2r_8 - 0.5)cd \quad (8)$$

8. Membandingkan nilai *fitness* $pworst_i$ baru dengan nilai *fitness* pworst. Apabila nilai *fitness* $pworst_i$ baru lebih kecil daripada pworst, maka nilai $pworst_i$ digantikan oleh $pworst_i$ baru. Nilai $pbest_i$ baru kemudian disimpan pada matriks M
9. Nilai $pbest_i$ terbaik di antara semua partikel dipilih sebagai gbest. Kembali ke langkah 3 dan ulangi sampai konvergen.

III. UJI COBA

A. Data Uji Coba

Data masukan utama yang dibutuhkan untuk proses uji coba adalah sistem persamaan nonlinear dan beberapa fungsi umum yang sering digunakan untuk menguji kinerja algoritma optimasi. Data masukan lebih detail ditunjukkan pada Tabel 1.

Tabel 1.
Data Uji Coba

Uji Coba	Skenario	Fungsi/Sistem Persamaan Nonlinear
1	A	Fungsi Rastrigin
	B	Fungsi Objektif Sinus
	C	Fungsi Powell Quartic
	D	Fungsi Six-Hump Camelback
	E	Fungsi Rosenbrock
2	A	$f_1(x)=(3-5x_1)x_1+1-2x_2=0$ $f_2(x)=(3-5x_i)x_i+1-x_{i-1}-2x_{i+1}=0 \;\; i=2,....,9$ $f_3(x)=(3-5x_{10})x_{10}+1-x_9=0$
	B	$x_1+\frac{x_2^4 x_4 x_6}{4}+0.75=0$ $x_2+0.405e^{1+x_1x_2}-1.405=0$ $x_3-\frac{x_4x_6}{2}+1.5=0$ $x_4-0.605e^{(1-x_3^2)}-0.395=0$ $x_5-\frac{x_2x_6}{2}+1.5=0$ $x_6-x_1x_5=0$
	C	$x_1^{x_2}+x_2^{x_1}-5x_1x_2x_3=85$ $x_1^3-x_2^{x_3}-x_3^{x_2}=60$ $x_1^{x_3}+x_3^{x_1}-x_2=0$ $3\le x_1\le 5, 2\le x_2\le 4, 0.5\le x_3\le 2.$
	D	$e^{x_1^2}-8x_1\sin(x_2)=0$ $x_1+x_2-1=0$ $(x_3-1)^3=0$ $3x_1-\cos(x_2x_3)-0.5=0$
	E	$x_1^2-625x_2^2-0.25=0$ $e^{-x_1x_2}+2-x_3+\frac{(10\pi-3)}{3}=0$
	F	$x_1^3-3x_1x_2^2-1=0$ $3x_1^2x_2-x_2^3+1=0$

B. Skenario Uji Coba

Uji coba penyelesaian sistem persamaan nonlinear dengan menggunakan algoritma *Particle Swarm* yang diusulkan oleh Jaberipour terbagi menjadi 3 skenario. Masing-masing skenario mempunyai tujuan yang berbeda. Berikut ditunjukkan penjelesan masing-masing uji coba :

1. Data masukan yang digunakan pada skenario uji coba pertama adalah fungsi umum yang sering digunakan untuk menguji kinerja algoritma optimasi. Tujuan pengujian ini tentu saja untuk mengetahui kinerja dari algoritma *Particle Swarm*.
2. Data masukan yang digunakan pada skenario uji coba kedua adalah sistem persamaan nonlinear. Tujuan pengujian ini adalah mengetahui efisiensi algoritma dalam menyelesaikan sistem persamaan nonlinear
3. Masing-masing skenario uji coba menggunakan jumlah partikel yang berbeda-beda, yaitu 100, 200, 500, dan 1000 partikel. Hal ini bertujuan untuk mengetahui hubungan antara jumlah partikel yang digunakan dengan solusi yang diperoleh.

C. Hasil dan Analisis Uji Coba

Pada uji coba pertama, pengujian dilakukan terhadap lima fungsi objektif. Hal ini dilakukan untuk mengetahui kinerja

Tabel 3.
Perbandingan hasil uji coba pertama

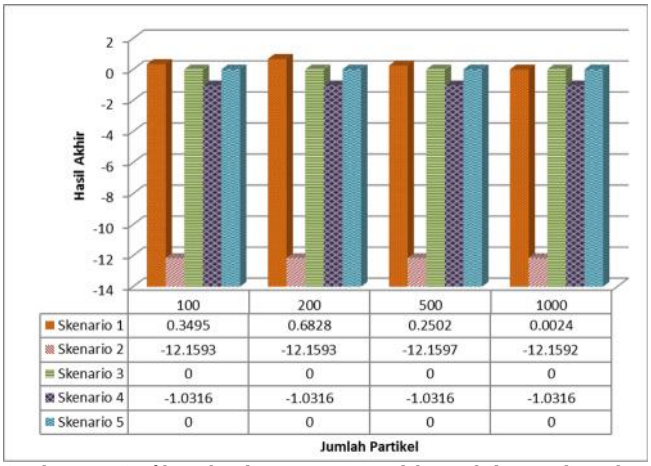
Skenario	Nilai	Hasil Uji Coba	Hasil Eksak	Iterasi Konvergensi
A	x_1	0.0031	0	15
	x_2	-0.0024	0	
	x_3	-0.0005	0	
	x_4	0.0013	0	
	x_5	0.0034	0	
	x_6	-0.0029	0	
	x_7	0.0014	0	
	x_8	0.0012	0	
	x_9	-0.0056	0	
	x_{10}	-0.0000	0	
	$f(x)$	0.0000	0	
B	x_1	5.3621	5.3623	200
	x_2	5.3622	5.3624	
	x_3	5.3630	5.3621	
	x_4	5.3626	5.3633	
	x_5	5.3622	5.3627	
	x_6	5.3623	5.3625	
	x_7	5.3621	5.3624	
	x_8	5.3603	5.3622	
	x_9	5.3678	5.3627	
	x_{10}	5.3494	5.3616	
	$f(x)$	-12.1597	-12.15982	
C	x_1	-0.0003	0	10
	x_2	0.0000	0	
	x_3	0.0000	0	
	x_4	0.0000	0	
	$f(x)$	0.0000	0	
D	x_1	-0.0898	-0.08984	15
	x_2	0.7126	0.71266	
	$f(x)$	-1.0316	-1.0316285	
E	x_1	1.0000	1	10
	x_2	1.0000	1	
	$f(x)$	0.0000	0	

Tabel 2.
Perbandingan hasil uji coba kedua

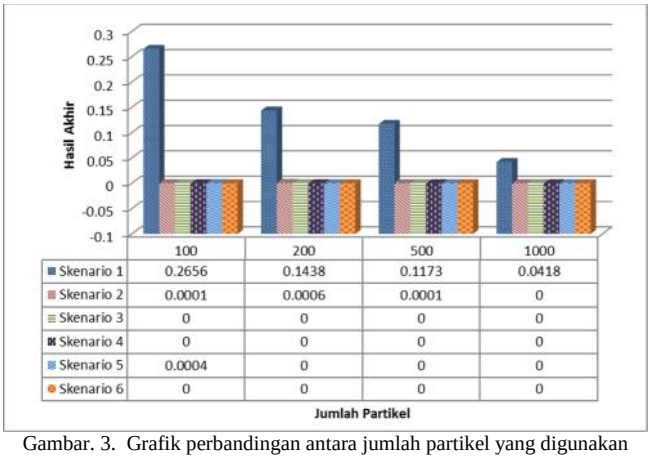
Skenario	Nilai	Hasil Uji Coba	Hasil Jaberipur dkk	Iterasi Konvergensi
A	x_1	-0.1060	-0.3820	50
	x_2	0.3171	-0.4380	
	x_3	0.8061	-0.4459	
	x_4	-0.0875	-0.4469	
	x_5	-0.0582	-0.4469	
	x_6	0.4479	-0.4463	
	x_7	0.7058	-0.4441	
	x_8	0.0794	-0.4361	
	x_9	0.2780	-0.4078	
	x_{10}	0.7688	-0.3095	
B	x_1	-0.9496	-1	20
	x_2	0.9676	1	
	x_3	-1.0488	-1	
	x_4	0.9378	1	
	x_5	-1.0265	-1	
	x_6	0.9728	1	
C	x_1	4	4	10
	x_2	3	3	
	x_3	1	1	
D	x_1	0.1755	0.1756 atau 0.7042	15
	x_2	0.8244	0.8244 atau 0.2957	
	x_3	1.0005	1	
E	x_1	0.5000	0.5	10
	x_2	-0.0000	0	
	x_3	-0.5236	-0.5236	
F	x_1	-0.2905	-0.2905 atau -0.7937	15
	x_2	1.0842	1.0842 atau -0.7937	

algoritma *Particle Swarm* dalam mencari solusi yang optimal. Hasil pengujian yang terbaik dibandingkan dengan hasil eksak dari masing-masing fungsi. Pengujian dianggap berhasil apabila hasil uji coba yang didapat sama atau mendekati hasil eksak fungsi.

Perbandingan hasil uji coba pertama dengan hasil eksak fungsi ditunjukkan pada Tabel 2. Berdasarkan data tabel tersebut hasil pengujian skenario pertama, kedua, dan ketiga mendekati atau hampir sama dengan solusi eksak fungsinya, sedangkan hasil skenario keempat dan kelima memiliki nilai yang sama persis dengan solusi eksak fungsinya. Oleh karena



Gambar. 2. Grafik perbandingan antara jumlah partikel yang digunakan dengan hasil akhir yang didapat pada uji coba pertama



Gambar. 3. Grafik perbandingan antara jumlah partikel yang digunakan dengan hasil akhir yang didapat pada uji coba kedua

itu dapat disimpulkan bahwa algoritma *Particle Swarm* yang diusulkan oleh Jaberipour memiliki kinerja yang baik dalam mencari solusi yang optimal.

Uji coba yang kedua dilakukan terhadap enam sistem persamaan nonlinear yang berbeda. Pengujian ini bertujuan untuk mengetahui efisiensi algoritma *Particle Swarm* yang diusulkan oleh Jaberipour dalam menyelesaikan sistem persamaan nonlinear. Hasil pengujian terbaik dibandingkan dengan hasil yang didapat oleh Jaberipur dkk.

Perbandingan hasil uji coba keduan dengan hasil Jaberipour et al. ditampilkan pada Tabel 3. Berdasarkan data tabel tersebut, hasil akhir skenario pertama dan kedua mendekati hasil yang didapatkan oleh Jaberipour et al. sedangkan hasil akhir skenario ketiga, keempat, kelima, dan keenam sama persis dengan hasil Jaberipour. Berdasarkan data hasil terbaik pada Tabel 2 dan Tabel 3, algoritma *Particle Swarm* yang diusulkan oleh Jaberipour pada penyelesaian beberapa fungsi telah konvergen pada iterasi ke 10 sampai 20, namun ada fungsi yang konvergen pada iterasi ke 200. Berdasarkan hasil tersebut dapat disimpulkan bahwa algoritma *Particle Swarm* yang diusulkan oleh Jaberipour efisien dalam menyelesaikan sistem persamaan nonlinear.

Masing-masing skenario pada uji coba pertama dan kedua menggunakan jumlah partikel yang berbeda-beda. Hal ini dilakukan untuk mengetahui hubungan antara jumlah partikel yang digunakan dengan hasil akhir yang didapat. Grafik hubungan antara jumlah partikel dengan nilai akhir fungsi *f(x)* pada uji coba pertama ditampilkan pada Gambar 2, sedangkan grafik hubungan untuk uji coba kedua ditampilkan pada Gambar 3. Berdasarkan gambar tersebut, hasil akhir yang didapat dengan menggunakan jumlah partikel yang berbeda tidak jauh berbeda. Oleh karena itu disimpulkan bahwa jumlah partikel yang digunakan tidak mempengaruhi hasil akhir.

IV. KESIMPULAN

Berdasarkan sistem yang telah dibuat beserta serangkaian uji coba yang telah dilakukan, maka dapat ditarik kesimpulan sebagai berikut :

1. Algoritma *Particle Swarm* yang diusulkan oleh Jaberipour dapat digunakan untuk menyelesaikan sistem persamaan nonlinear.
2. Algoritma *Particle Swarm* yang diusulkan oleh Jaberipour memiliki kinerja yang baik dalam mencari solusi yang optimal. Hal ini dibuktikan dengan hasil uji coba pertama, solusi yang didapat mendekati solusi eksak masing-masing fungsinya.
3. Berdasarkan hasil pengujian terbaik dari masing-masing skenario uji coba, penyelesaian beberapa fungsi dengan algoritma *Particle Swarm* yang diusulkan oleh Jaberipour telah konvergen pada iterasi ke 10 sampai iterasi ke 20. Namun, terdapat penyelesaian fungsi yang konvergen pada iterasi ke 200.
4. Jumlah partikel yang digunakan oleh algoritma *Particle Swarm* dengan nilai minimal 100 partikel tidak mempengaruhi hasil akhir yang didapatkan.

DAFTAR PUSTAKA

[1] Jaberipour Majid, Khorram Esmaile, Karimi Behrooz, "Particle swarm algorithm for solving systems of nonlinear equations," *Elsevier Computers and Mathematics with Applications ScienceDirect* (2011) 566-576.

[2] J. Kennedy, R.C. Eberhart, "Particle swarm optimization," *IEEE International Conference on Neural Networks*, IEEE Service Center, Piscataway, NJ, 1942-1948.

[3] C. K. Mohan. E. Ozcan. Particle Swarm Optimization Homepage. Available: http://www.cis.syr.edu/~mohan/pso/

[4] R. Eberhart, Y. Shi, "Comparing inertia weight and constriction factors in particle swarm optimization," *in: Proceeding of the 2000 Congress on Evol. Comput.* (2000) 84-88.

[5] 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.* (2002) 792-797.