

## Wykorzystanie algorytmu optymalizacji rojem cząstek do rozwiązywania układów równań nieliniowych

Karol Prus,  
Ewa Figielska\*

Warszawska Wyższa Szkoła Informatyki

---

### Streszczenie

Artykuł dotyczy wykorzystania algorytmu optymalizacji rojem cząstek do rozwiązywania układów równań nieliniowych. Przeprowadzona została eksperymentalna analiza efektywności i skuteczności działania algorytmu w zależności od ustawień jego parametrów.

**Słowa kluczowe** – Algorytm optymalizacji rojem cząstek, układy równań nieliniowych, optymalizacja

---

\* E-mail: [efigielska@ms.wysi.edu.pl](mailto:efigielska@ms.wysi.edu.pl)

**Zgłoszono do druku 5 lipca 2023 r.**

## 1. Wprowadzenie

Potrzeba rozwiązywania układów równań nieliniowych pojawia się w wielu praktycznych problemach inżynierskich, na przykład w analizie kinematycznej mechanizmów [1], [2], analizie nieliniowych obwodów elektrycznych [3], inżynierii wodnej [4]. Niestety, znalezienie rozwiązania nawet pojedynczego równania nieliniowego jest zadaniem trudnym, a zwiększenie liczby równań jeszcze potęguje te trudności. Do rozwiązywania układów równań nieliniowych najczęściej stosowane są podejścia oparte na iteracyjnej metodzie Newtona [5], [6], [7], również w powiązaniu z innymi technikami [8]. Metoda Newtona posiada jednak pewne wady – wymaga ciągłości i różniczkowalności funkcji, a także jest wrażliwa na wybór rozwiązania początkowego – niewłaściwy wybór prowadzi do nieprawidłowych wyników. Dlatego też podejmowane są próby stosowania innego rodzaju metod, w szczególności algorytmów inspirowanych naturą, takich jak algorytm światełka [9], algorytmy ewolucyjne [10], czy też algorytm optymalizacji rojem cząstek [11], [12].

Niniejszy artykuł dotyczy rozwiązywania układów równań nieliniowych za pomocą algorytmu optymalizacji rojem cząstek. Przeprowadzony został eksperyment obliczeniowy, w którym zbadano wpływ różnych ustawień parametrów algorytmu na jego skuteczność i efektywność. Obliczenia przeprowadzono dla czterech układów równań nieliniowych o różnym stopniu trudności.

## 2. Algorytm optymalizacji rojem cząstek

Algorytm optymalizacji rojem cząstek (ang. *Particle Swarm Optimization, PSO*) został opracowany przez Jamesa Kennedy’ego i Russella C. Eberharta w roku 1995 i zastosowany do optymalizacji funkcji nieliniowej oraz trenowania sieci neuronowej [13]. Działanie algorytmu jest inspirowane zachowaniem społecznym stada migrujących ptaków. Każdy ptak podczas lotu komunikuje się z innymi ptakami w stadzie, identyfikuje ptaka znajdującego się w najlepszym położeniu, po czym podąża w jego kierunku uwzględniając swoje

aktualne położenie oraz dotychczasowe doświadczenia. Proces ten jest powtarzany aż stado osiągnie pożądaný cel podróży [14].

Algorytm optymalizacji rojem cząstek jest procedurą iteracyjną działającą z populacją (stadem, rojem) potencjalnych rozwiązań problemu, reprezentowanych przez cząstki (ptaki). Rój składa się z  $n$  cząstek. Pojedyncza cząstka jest opisana przez swoje położenie  $x_i$ , reprezentujące potencjalne rozwiązanie, oraz prędkość  $v_i$ , która określa kierunek i odległość na jaką cząstka może się przemieścić.

W iteracji  $t$  algorytmu, nowe położenie cząstki  $i$  jest obliczane ze wzoru:

$$x_i^t = x_i^{t-1} + v_i^t \quad \text{dla } i = 1, \dots, n, \quad (1)$$

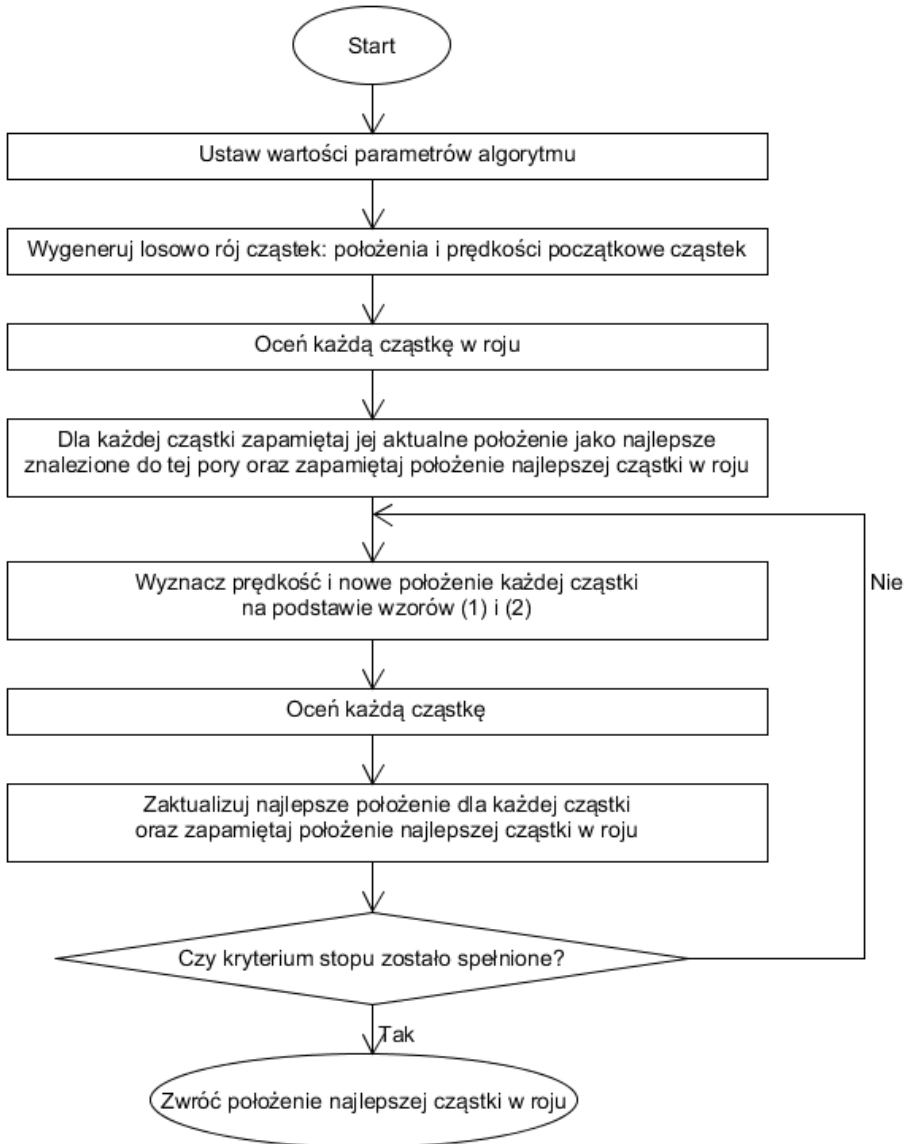
gdzie  $x_i^{t-1}$  jest położeniem cząstki w poprzedniej iteracji,  $v_i^t$  jest prędkością cząstki wyznaczoną na podstawie jej poprzedniej prędkości  $v_i^{t-1}$ , najlepszego dotychczasowego położenia  $p_i$  oraz położenia najlepszej cząstki w roju,  $g$ , według następującego wzoru:

$$v_i^t = wv_i^{t-1} + c_1r_1(p_i - x_i^{t-1}) + c_2r_2(g - x_i^{t-1}) \quad \text{dla } i = 1, \dots, n \quad (2)$$

gdzie  $w$ ,  $c_1$  i  $c_2$  są parametrami algorytmu opisanymi poniżej, a  $r_1$  i  $r_2$  są liczbami losowymi z przedziału  $[0, 1]$ .

Parametr  $w$ , nazywany wagą inercji, wpływa na zdolność cząstek do zachowywania poprzedniej prędkości. Wraz ze wzrostem wartości  $w$ , zwiększa się zdolność cząstek do przeszukiwania nowych obszarów przestrzeni rozwiązań, jednak zbyt duża wartość tego parametru może nadmiernie spowolnić zbieżność algorytmu. Parametr  $c_1$ , nazywany współczynnikiem uczenia kognitywnego, wyraża, ile zaufania cząstka ma do samej siebie. Za pomocą tego współczynnika można kontrolować intensywność przeszukiwania lokalnych obszarów znajdujących się w otoczeniu najlepszych położení poszczególnych cząstek. Parametr  $c_2$ , nazywany współczynnikiem uczenia społecznego, wyraża, ile zaufania cząstka ma do najlepszej cząstki w roju. Za pomocą współczynnika  $c_2$  można kontrolować szybkość dążenia do aktualnie najlepszego globalnego rozwiązania [15].

Schemat działania algorytmu optymalizacji rojem cząstek jest przedstawiony rysunku 1.



Rysunek 1. Schemat działania algorytmu optymalizacji rojem cząstek

### 3. Rozwiązywanie układów równań nieliniowych jako problemu optymalizacyjnego

Układ  $m$  równań nieliniowych ma postać:

$$\begin{cases} f_1(x_1, x_2, \dots, x_n) = 0 \\ f_2(x_1, x_2, \dots, x_n) = 0 \\ \vdots \\ f_m(x_1, x_2, \dots, x_n) = 0 \end{cases} \quad (3)$$

gdzie  $f_i$  jest funkcją  $n$  zmiennych, przy czym przynajmniej jedna funkcja  $f_i$  jest nieliniowa.

W celu rozwiązania układu równań (3) za pomocą algorytmu optymalizacji rojem cząstek, układ ten jest przekształcany do problemu optymalizacyjnego, w którym poszukiwane jest globalne minimum pomocniczej funkcji  $F(x)$  o następującej postaci [12]:

$$F(x) = \sum_{i=1}^m f_i^2(x) \quad (4)$$

Z definicji najmniejsza wartość  $F(x)$  wynosi 0. Jeżeli  $F(x^*) = 0$ , to  $x^*$  jest globalnym minimum. Stąd  $f_1(x^*) = f_2(x^*) = \dots = f_m(x^*) = 0$ , więc  $x^*$  jest rozwiązaniem układu równań (3).

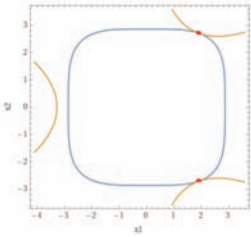
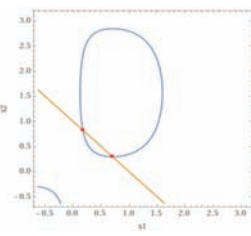
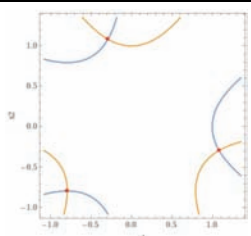
### 4. Eksperyment obliczeniowy

Eksperyment obliczeniowy polegał na rozwiązaniu za pomocą algorytmu optymalizacji rojem cząstek 4 układów równań o różnym stopniu trudności. Obliczenia przeprowadzono dla 21 różnych ustawień wartości parametrów algorytmu. Dla każdego z ustawień algorytm był uruchamiany 10 razy.

Jako kryterium oceny algorytmu przyjęto jego skuteczność wyrażoną jako stosunek liczby przebiegów, w których zostało znalezione rozwiązanie do liczby wszystkich przebiegów algorytmu. Efektywność algorytmu określono przez zmierzenie czasu wykonywania obliczeń do chwili znalezienia najlepszego rozwiązania.

Obliczenia przeprowadzono na komputerze z procesorem Intel Core i7-4710HQ i systemem operacyjnym Windows 10.

Tabela 1. Układy równań

U#	Układ równań	Rozwiązanie	
U1	$x_1^4 + x_2^4 - 67 = 0$ $x_1^3 - 3x_1x_2^2 + 35 = 0$	$(x_1; x_2) \approx$ $(1,94242; -2,69517)$ $(x_1; x_2) \approx$ $(1,94242; 2,69517)$	 <p>— <math>x_1^4 + x_2^4 - 67 = 0</math>  — <math>x_1^3 - 3x_1x_2^2 + 35 = 0</math></p>
U2	$e^{x_1^2} - 8x_1 \sin x_2 = 0$ $x_1 + x_2 - 1 = 0$ $(x_3 - 1)^3 = 0$	$(x_1; x_2; x_3) \approx$ $(0,17559; 0,82440; 1)$ $(x_1; x_2; x_3) \approx$ $(0,70424; 0,29575; 1)$ Uwaga: wykres uwzględnia wyłącznie dwa pierwsze równania	 <p>— <math>e^{x_1^2} - 8x_1 \sin(x_2) = 0</math>  — <math>x_1 + x_2 - 1 = 0</math></p>
U3	$x_1^3 - 3x_1x_2^2 - 1 = 0$ $3x_1^2x_2 - x_2^3 + 1 = 0$	$(x_1; x_2) \approx$ $(-0,29052; 1,08422)$ $(x_1; x_2) \approx$ $(1,08422; -0,29052)$ $(x_1; x_2) \approx$ $(-0,79370; -0,79370)$	 <p>— <math>x_1^3 - 3x_1x_2^2 - 1 = 0</math>  — <math>3x_1^2x_2 - x_2^3 + 1 = 0</math></p>
U4	$x_1 + \frac{x_2^2x_4x_6}{4} + 0,75 = 0$ $x_2 + 0,405e^{1+x_1x_2} - 1,405 = 0$ $x_3 - \frac{x_4x_6}{2} + 1,5 = 0$ $x_4 - 0,605e^{(1-x_3^2)} - 0,395 = 0$ $x_5 - \frac{x_2x_6}{2} + 1,5 = 0$ $x_6 - x_1x_5 = 0$	$(x_1; x_2; x_3; x_4; x_5; x_6) = (-1; 1; -1; 1; -1; 1)$	

#### 4.1. Układy równań

Układy równań użyte w eksperymencie zostały zaczerpnięte z [9], [6], [16] oraz [11] i oznaczone odpowiednio jako U1, U2, U3 i U4. Ich rozwiązania przedstawiono w tabeli 1. Wykresy dla równań dwóch zmiennych zostały wykonane za pomocą kalkulatora on-line [17].

#### 4.2. Parametry algorytmu

Działanie algorytmu optymalizacji rojem cząstek zależy od odpowiedniego doboru wartości jego parametrów: liczby cząstek w roju  $n$ , wagi inercji,  $w$  i współczynników uczenia  $c_1$ ,  $c_2$ . W eksperymencie przebadanych zostało 21 ustawień wartości parametrów, jak pokazano w tabeli 2. Wartości użyte w ustawieniach P7, P14 i P21 zostały wskazane w [18] jako najlepsze przy znajdowaniu optimum funkcji nieliniowej. Jako warunek stopu przyjęto wykonanie 10000 iteracji.

**Tabela 2.** Ustawienia wartości parametrów algorytmu

P#	$c_1, c_2$	$w$	$n$	P#	$c_1, c_2$	$w$	$n$	P#	$c_1, c_2$	$w$	$n$
P1	2	1	25	P8	2	1	50	P15	2	1	100
P2	2	0,5	25	P9	2	0,5	50	P16	2	0,5	100
P3	1,5	1	25	P10	1,5	1	50	P17	1,5	1	100
P4	1,5	0,5	25	P11	1,5	0,5	50	P18	1,5	0,5	100
P5	1	1	25	P12	1	1	50	P19	1	1	100
P6	1	0,5	25	P13	1	0,5	50	P20	1	0,5	100
P7	1,49445	0,729	25	P14	1,49445	0,729	50	P21	1,49445	0,729	100

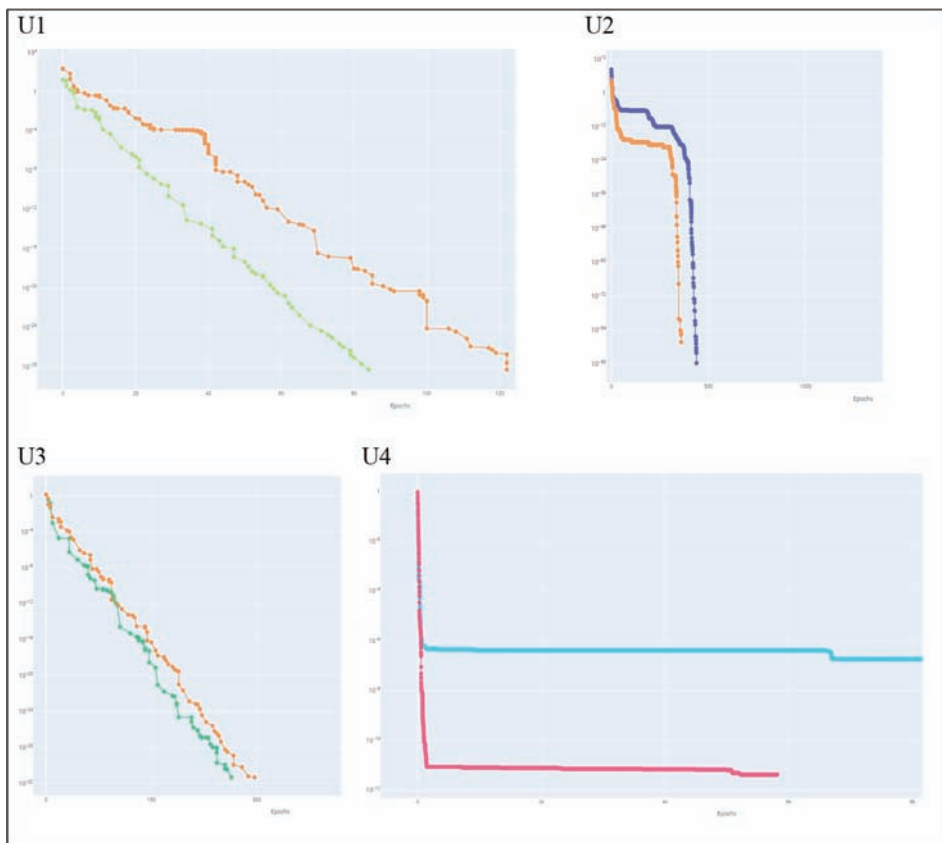
#### 4.3. Wyniki

W tabeli 3 przedstawiono najlepsze wyniki uzyskane dla poszczególnych układów równań, tzn. maksymalne wartości średniej (po 10 przebiegach) skuteczności wraz z ustawieniami parametrów algorytmu, przy których zostały one osiągnięte. Wyniki te pokazują, że algorytm był w stanie znaleźć rozwiązanie

dla każdego z rozważanych układów równań. Co więcej, skuteczność była stosunkowo wysoka – wahała się od 50% do 100%. Najłatwiejszym do rozwiązania układem równań okazał się U3, a najtrudniejszym – U4. Najlepszym i najbardziej uniwersalnym ustawieniem parametrów algorytmu jest zestaw P21.

**Tabela 3.** Maksymalna średnia skuteczność algorytmu dla poszczególnych układów równań

Układ równań	Największa średnia skuteczność	Ustawienia parametrów algorytmu zapewniające największą skuteczność
U1	90%	P14, P20, P21
U2	80%	P14, P21
U3	100%	P20, P21
U4	50%	P21

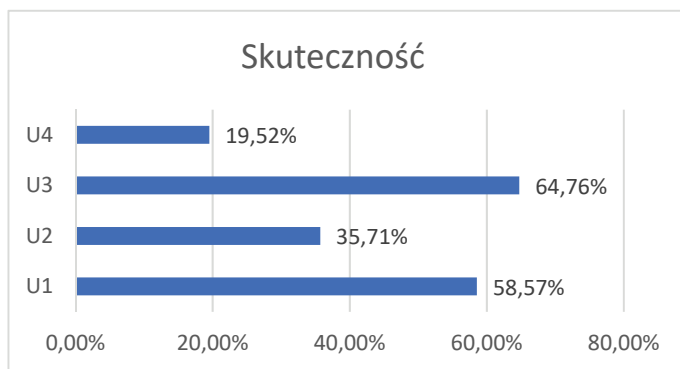


**Rysunek 2.** Przykładowe wykresy zbieżności algorytmu z ustawieniem parametrów P21



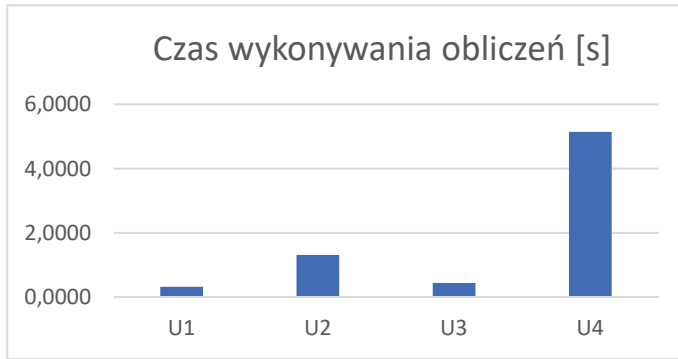
Wykresy zbieżności algorytmu przy ustawieniu parametrów P21 zostały pokazane na rysunku 2. Widzimy tu wartości minimalizowanej funkcji  $F(x)$  (wyrażenie (4)) uzyskane w dwóch przykładowych przebiegach algorytmu dla każdego z rozważanych układów równań. Dla układu U4 wyraźnie widać przedwczesną zbieżność do złej jakości minimum lokalnego (linia niebieska), tu wartości minimalizowanej funkcji  $F(x)$  (wyrażenie (4)) uzyskane w dwóch przykładowych przebiegach algorytmu dla każdego z rozważanych układów równań. Dla układu U4 wyraźnie widać przedwczesną zbieżność do złej jakości minimum lokalnego (linia niebieska).

Na rysunku 3 przedstawiono średnią skuteczność algorytmu dla poszczególnych układów równań, uwzględniając wszystkie badane ustawienia parametrów. Najwyższą średnią skuteczność, około 65%, algorytm osiągnął dla układu U3, a najniższą, około 20%, dla U4. Potwierdziły się tym samym wcześniejsze spostrzeżenia, co do trudności badanych układów równań. Najtrudniejszy z nich, U4, posiada największą liczbę zmiennych i równań oraz zawiera składniki potęgowe ze zmiennymi w wykładnikach. Drugi pod względem trudności, U2, ma mniej równań i zmiennych niż U4, ale również zawiera składnik potęgowy ze zmienną w wykładniku. Takiego składnika nie posiadają natomiast układy U1 i U3, które okazały się najłatwiejsze do rozwiązania.

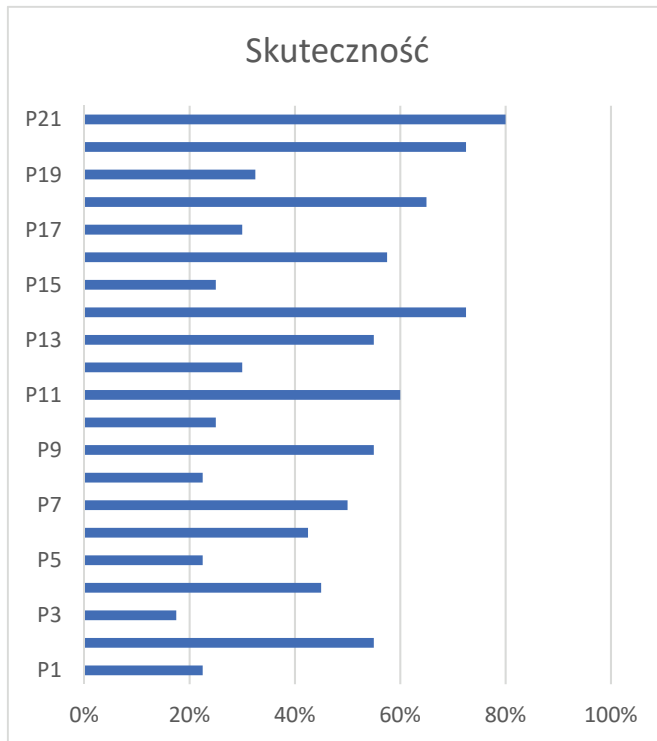


**Rysunek 3.** Średnia skuteczność algorytmu dla poszczególnych układów równań

Średni czas wykonywania obliczeń (rysunek 4) był niewielki i nie przekraczał 5 sekund dla najtrudniejszego układu U4, 1.5 sekundy dla U2 i 0.5 sekundy dla układów U1 i U3.



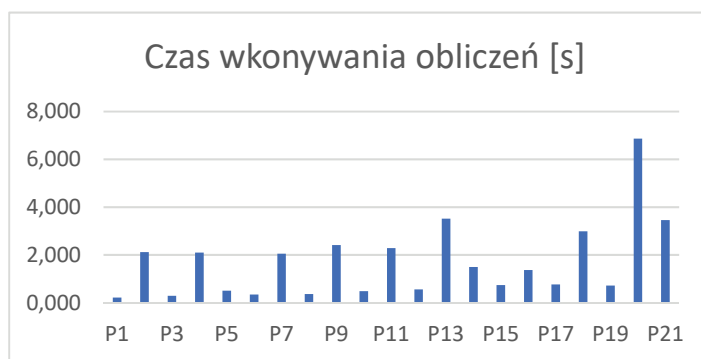
**Rysunek 4.** Średni czas wykonywania obliczeń dla różnych układów równań



**Rysunek 5.** Średnia skuteczność algorytmu dla różnych ustawień wartości jego parametrów

Na rysunku 5 pokazana została średnia skuteczność osiągnięta przy różnych ustawieniach parametrów algorytmu, uwzględniając wszystkie układy równań.

Największą skuteczność, średnio 80%, zapewniło ustawienie P21. Średnia skuteczność o wartości co najmniej 50% została osiągnięta także dla ustawień P20, P14, P18, P11, P16, P13, P9, P2, P7, co wskazuje, że istotny wpływ na działanie algorytmu ma wartość wagi inercji,  $w$ , która we wszystkich wymienionych ustawieniach jest mniejsza od 1. Dodatkowo można zaobserwować, że w przypadku, gdy wartości pozostałych parametrów ( $c_1$ ,  $c_2$  i  $n$ ) są ustalone, algorytm z  $w = 0,5$  lub  $w = 0,729$  daje zawsze lepsze wyniki niż w przypadku, gdy  $w = 1$ . Na skuteczność algorytmu ma wpływ również wielkość roju – najlepsze wyniki uzyskano dla roju składającego się ze 100 cząstek, przy czym na podstawie dodatkowych obliczeń stwierdzono, że dalsze zwiększenie jego rozmiaru nie poprawia skuteczności. Zbyt mały rój (25 cząstek) ogranicza możliwości eksploracyjne algorytmu. Jeżeli chodzi o współczynniki uczenia, to w zakresie badanych wartości nie zaobserwowano, żeby miały one istotny wpływ na działanie algorytmu.



Rysunek 6. Średni czas wykonywania obliczeń algorytmu dla różnych ustawień parametrów

Średni czas obliczeń (rysunek 6) był niewielki – nie przekraczał 7 sekund. Dla najlepszego ustawienia parametrów algorytmu, P21, wynosił około 3,5 sekundy.

Na podstawie rysunków 5 i 6 widać, że lepsza skuteczność algorytmu zawsze związana jest z dłuższym czasem wykonywania obliczeń, co sugeruje, że w takich przypadkach algorytm ucieka z lokalnych optimum i przeprowadza dalszą eksplorację przestrzeni rozwiązań.

## 5. Podsumowanie

W artykule opisano zastosowanie algorytmu optymalizacji rojem cząstek do rozwiązywania układów równań nieliniowych. Przeprowadzony został eksperyment obliczeniowy, którego wyniki pozwoliły zidentyfikować najlepsze i najgorsze ustawienia parametrów algorytmu. Wskazały również na potrzebę opracowania dodatkowych mechanizmów, które podniosłyby skuteczność algorytmu przy rozwiązywaniu bardziej złożonych układów równań, na przykład zawierających wyrażenia potęgowe ze zmiennymi w wykładniku.

## Literatura

- [1] J. Frączek, „Modelowanie mechanizmów przestrzennych metodą układów wieloczłonowych”, *Prace Naukowe Politechniki Warszawskiej. Mechanika* nr 196, pp. 3–138, 2002.
- [2] M. Jeż, A. Świder, „Analiza drgań nieliniowych jednocylindrowego silnika tłokowego”, *Journal of KONES* Vol. 8, No. 3-4, pp. 98–105, 2001.
- [3] M. Gajer, „Zastosowanie algorytmu ewolucyjnego do analizy nieliniowych obwodów elektrycznych”, *Przegląd Elektrotechniczny* Vol. 86, No. 7, pp. 342–345, 2010.
- [4] R. Szymkiewicz, *Metody numeryczne w inżynierii wodnej*. Politechnika Gdańska, 2007.
- [5] B. Saheya, G. Chen, Y. Sui, and C. Wu, “A new Newton-like method for solving nonlinear equations” *SpringerPlus* Vol. 5, pp. 1–13, 2016. [Online]. <https://doi.org/10.1186/s40064-016-2909-7>
- [6] J.L. Hueso, E. Martínez, 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. [Online]. <https://doi.org/10.1016/j.cam.2008.04.013>
- [7] J.R. Sharma and H. Arora, “On efficient weighted-Newton methods for solving systems of nonlinear equations”, *Applied Mathematics and Computation* Vol. 222, pp. 497–506, 2013. [Online]. <https://doi.org/10.1016/j.amc.2013.07.066>

- [8] 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* Vol. 8, No. 2, pp. 1068-1073, 2008. [Online]. <https://doi.org/10.1016/j.asoc.2007.05.013>
- [9] M.K.A. Ariyaratne, T.G.I. Fernando, S. Weerakoon, “Solving systems of nonlinear equations using a modified firefly algorithm (MODFA)”, *Swarm and Evolutionary Computation* Vol. 48, pp. 72-92, 2019. [Online]. <https://doi.org/10.1016/j.swevo.2019.03.010>
- [10] C. Grosan, A. Abraham, A. Gelbukh, “Evolutionary Method for Nonlinear Systems of Equations”, *MICAI 2006: Advances in Artificial Intelligence*, Berlin, Heidelberg, 2006, pp. 283–293. [Online]. <http://isda05.softcomputing.net/micai06.pdf>
- [11] Y. Mo, H. Liu, 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. [Online]. <https://doi.org/10.1016/j.camwa.2008.10.005>
- [12] M. Jaberipour, E. Khorram, B. Karimi, “Particle swarm algorithm for solving systems of nonlinear equations”, *Computers & Mathematics with Applications* Vol. 62, No. 2, pp. 566–576, 2011. [Online]. <https://doi.org/10.1016/j.camwa.2011.05.031>
- [13] J. Kennedy, R. Eberhart, “Particle swarm optimization”, *Proceedings of IEEE International Conference on Neural Networks IV* Vol. 4, 1995, pp. 1942–1948. [Online]. [https://www.academia.edu/download/30280202/\\_reading6\\_1995\\_particle\\_swarming.pdf](https://www.academia.edu/download/30280202/_reading6_1995_particle_swarming.pdf)
- [14] E. Elbeltagi, T. Hegazy, D. Grierson, “Comparison among five evolutionary-based optimization algorithms”, *Advanced engineering informatics* Vol. 19, No. 1, pp. 43–53, 2005. [Online]. <https://doi.org/10.1016/j.aei.2005.01.004>
- [15] Y. He, W.J. Ma, J. P. Zhang, “The parameters selection of PSO algorithm influencing on performance of fault diagnosis”, *MATEC Web of conferences* Vol. 63, 2016, p. 02019. [Online]. [https://www.matec-conferences.org/articles/mateconf/pdf/2016/26/mateconf\\_mmme2016\\_02019.pdf](https://www.matec-conferences.org/articles/mateconf/pdf/2016/26/mateconf_mmme2016_02019.pdf)

- [16] 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. [Online]. <https://doi.org/10.1016/j.cam.2007.10.054>
- [17] Kalkulator Wolfram Alpha. [Online]. <https://www.wolframalpha.com>
- [18] J. McCaffrey, “Neural Network Lab: Particle Swarm Optimization Using C#”, *Visual Studio Magazine* 2013. [Online]. <https://visualstudiomagazine.com/Articles/2013/11/01/Particle-Swarm-Optimization.aspx>
- 

## **Particle swarm optimization algorithm for solving systems of nonlinear equations**

### **Abstract**

The article concerns the use of a particle swarm optimization algorithm for solving nonlinear equation systems. An experimental analysis of the effectiveness and efficiency of the algorithm has been conducted, considering various settings of its parameters.

**Keywords** – *particle swarm optimization algorithm, systems of nonlinear equations, optimization*