mathematics

MDPI

Article

An Improved Arithmetic Optimization Algorithm for Numerical Optimization Problems

Mengnan Chen [1], Yongquan Zhou [1,2,*] and Qifang Luo [1,2]

[1] College of Artificial Intelligence, Guangxi University for Nationalities, Nanning 530006, China; 2020210812000995@stu.gxmzu.edu.cn (M.C.); 20060043@gxun.edu.cn (Q.L.)
[2] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China
* Correspondence: zhouyongquan@gxun.edu.cn; Tel.: +86-136-0788-2594

Abstract: The arithmetic optimization algorithm is a recently proposed metaheuristic algorithm. In this paper, an improved arithmetic optimization algorithm (IAOA) based on the population control strategy is introduced to solve numerical optimization problems. By classifying the population and adaptively controlling the number of individuals in the subpopulation, the information of each individual can be used effectively, which speeds up the algorithm to find the optimal value, avoids falling into local optimum, and improves the accuracy of the solution. The performance of the proposed IAOA algorithm is evaluated on six systems of nonlinear equations, ten integrations, and engineering problems. The results show that the proposed algorithm outperforms other algorithms in terms of convergence speed, convergence accuracy, stability, and robustness.

Keywords: arithmetic optimization algorithm; population control strategy; systems of nonlinear equations; numerical integrals; metaheuristic

MSC: 68T20

Citation: Chen, M.; Zhou, Y.; Luo, Q. An Improved Arithmetic Optimization Algorithm for Numerical Optimization Problems. *Mathematics* **2022**, *10*, 2152. https://doi.org/10.3390/math10122152

Academic Editor: Frank Werner

Received: 28 May 2022
Accepted: 17 June 2022
Published: 20 June 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

1. Introduction

In the practical application calculations of science and engineering, many mathematical problems will be involved, such as nonlinear equation systems (NESs), numerical integration, etc. There are tremendous methods for solving NESs, including traditional techniques and intelligent optimization algorithms. Traditional techniques to solve NESs use gradient information [1], such as Newton's method [2,3], quasi-Newton's method [4], steepest descent method, etc. Due to relying on the selection of initial points and being prone to falling into optimal local one, these methods cannot obtain high-quality solutions for some specific problems. The metaheuristic algorithms, however, have the characteristics of low requirements for the initial point, a wide range of solutions, high efficiency, and robustness. These break through the limitations of traditional methods in solving problems. In recent years, metaheuristic algorithms have made great contributions in solving NESs (Karr et al. [5]; Ouyang et al. [6]; Jaberipour et al. [7]; Pourjafari et al. [8]; Jia et al. [9]; Ren et al. [10]; Cai et al. [11]; Abdollahi et al. [12]; Hirsch et al. [13]; Sacco et al. [14]; Gong et al. [15]; Ariyaratne et al. [16]; Gong et al. [17]; Ibrahim et al. [18]; Liao et al. [19]; Ning et al. [20]; Rizk-Allah et al. [21]; Ji et al. [22]; Turgut et al. [23]).

Numerical integration is a very basic computational problem. It is well-known that, when calculating the definite integral, the integrand is required to be easily given and then solved by the Newton-Leibniz formula. However, this method has many limitations, because in many practical problems, the original function of the integrand cannot be expressed, or the calculation is too complicated, so the definite integral of the integrand is replaced by a suitable finite sum approximation. The traditional numerical integration methods include the trapezoidal method, rectangle method, Romberg method, Gauss method, Simpson's method, Newton's method, etc. The above methods all divide the

integral interval into equal parts, and the calculation efficiency is not high. Therefore, it is of great significance to find a new technique with a fast convergence speed, high precision, and strong robustness for numerical integration. Zhou et al. [24], based on the evolutionary strategy method, worked to solve numerical integration. Wei et al. [25] researched the numerical integration method based on particle swarm optimization. Wei et al. [26], based on functional networks, worked to solve numerical integration. Deng et al. [27] solved the numerical integration problems based on the differential evolution algorithm. Xiao et al. [28] applied the improved bat algorithm in numerical integration. The quality of the solution obtained by the above techniques was higher than the traditional methods.

All along, engineering optimization problems have been a popular area of research. Metaheuristic algorithms have been widely applied to engineering optimization problems due to their great practical significance, such as applied to the automatic adjustment of controller coefficients (Szczepanski et al. [29]; Hu et al. [30]), applied to system identification (Szczepanski et al. [31]; Liu et al. [32]), applied to global path planning (Szczepanski et al. [33]; Brand et al. [34]), and applied to robotic arm scheduling (Szczepanski et al. [35]; Kolakowska et al. [36]).

The Arithmetic Optimization Algorithm (AOA) [37] is a novel metaheuristic algorithm proposed by Abualigah et al. in 2021. AOA is a mathematical model technique that simulates the behaviors of Arithmetic operators (i.e., Multiplication, Division, Subtraction, and Addition) and their influence on the best local solution. Some improvements and practical applications of the algorithm have been made by scholars. Premkumar et al. [38] proposed a multi-objective arithmetic optimization algorithm (MOAOA) for solving real-world multi-objective CEC-2021-constrained optimization problems. Bansal. et al. [39] used a binary arithmetic optimization algorithm for integrated features and feature selection. Agushaka et al. [40] introduced an advanced arithmetic optimization algorithm for solving mechanical engineering design problems. Abualigah et al. [41] presented a novel evolutionary arithmetic optimization algorithm for multilevel thresholding segmentation. Xu et al. [42] hybridized an extreme learning machine and a developed version of the arithmetic optimization algorithm for model identification of the proton exchange membrane fuel cells. Izci et al. [43] introduced an improved arithmetic optimization algorithm for the optimal design of controlled PID. Khatir et al. [44] proposed an improved artificial neural network using the arithmetic optimization algorithm for damage assessments.

The basic AOA still has some drawbacks. For instance, it is easy to fall into a local optimum due to the location update based on the optimal value, premature convergence, and low solution accuracy, which need to be solved. Furthermore, in order to seek a more efficient way to solve numerical problems, in this paper, an improved arithmetic optimization algorithm (IAOA) based on the population control strategy is proposed to solve numerical optimization problems. By classifying the population and adaptively controlling the number of individuals in the subpopulation, the information of each individual can be used effectively while increasing the population diversity. More individuals are needed in the early iterations to perform a large-scale search that avoids falling into the local optimum. The search around the optimal value later in the iterations by more individuals speeds up the algorithm to find the optimal value and improves the accuracy of the solution. The performance of the proposed IAOA algorithm is evaluated on six systems of nonlinear equations, ten integrations, and engineering problems. The results show that the proposed algorithm outperforms the other algorithms in terms of convergence speed, convergence accuracy, stability, and robustness.

The main structure of this paper is as follows. Section 2 reviews the relevant knowledge for the nonlinear equation systems, integration, and basic arithmetic optimization algorithm (AOA). Section 3 introduces the proposed IAOA in detail. Section 4 presents experimental results, comparisons, and analyses. Section 5 concludes the work and proposes future research directions.

2. Preliminaries

2.1. Nonlinear Equation Systems

Generally, a nonlinear equation system can be formulated as follows.

$$NES = \begin{cases} f_1(x_1, x_2, ..., x_D) = 0 \\ \vdots \\ f_i(x_1, x_2, ..., x_D) = 0 \\ \vdots \\ f_n(x_1, x_2, ..., x_D) = 0 \end{cases} \quad (1)$$

where x is a D-dimensional decision variable, and n is the number of equations. Some equations are linear; the others are nonlinear. If x^* satisfies $f_i(x^*) = 0$, then x^* is a root of the system of equations.

Before using the optimization algorithm to solve the NES, first is to convert it into a single-objective optimization problem [17] as follows.

$$\min f(x) = \sum_{i=1}^{n} f_i^2(x), x = (x_1, x_2, ..., x_i, ..., x_D) \quad (2)$$

Finding the minimum of an optimization problem is equivalent to finding the root of the NES.

2.2. Numerical Integration

Definite integrals are very basic mathematical calculation problems as follows.

$$\int_a^b f(x)dx \quad (3)$$

where $f(x)$ represents the integrand function, and a and b represent the upper and lower bounds, respectively.

Usually, firstly, we find the original function $F(x)$ of the integrand when finding a definite integral and then use the Newton-Leibniz formula as follows:

$$\int_a^b f(x)dx = F(b) - F(a), (F'(x) = f(x)) \quad (4)$$

However, in many cases, it is difficult to obtain the original function $F(x)$, so the Newton-Leibniz formula will not be able to be used.

In addition, the rest of the numerical quadrature methods are based on the quadrature formula of equidistant node division and summation or stipulate that the equidistant nodes remain unchanged during the whole process of calculating, as shown in Figure 1a. There need more nodes to obtain a high accuracy. However, the best segmentation is not the predetermined equidistant points, as shown in Figure 1b. Randomly generated subintervals has unequal intervals according to the concave and convex changes of the function curve, so the obtained value has a higher accuracy than the traditional methods. Based on this idea, there is another integral method based on non-equidistant point division [24]. First, generate some points randomly on the integral interval, and then, the algorithm is used to optimize these split points. Finally, a higher accuracy value will be obtained. This not only calculates the definite integral of the function in the usual sense but also calculates the integral of the singular function and the integral of the oscillatory function for this method [27]. The flow of the numerical integration algorithm based on unequal point segmentation is as follows [24].

(1) Randomly initialize the population in the search space S.
(2) Arrange each individual in the integral interval in ascending order. The integral interval has $n(n = D + 2)$ nodes and $n - 1$ segments. Calculate the distance h_i between two adjacent nodes and the function $f(x_k)$ value of each node, then calculate the

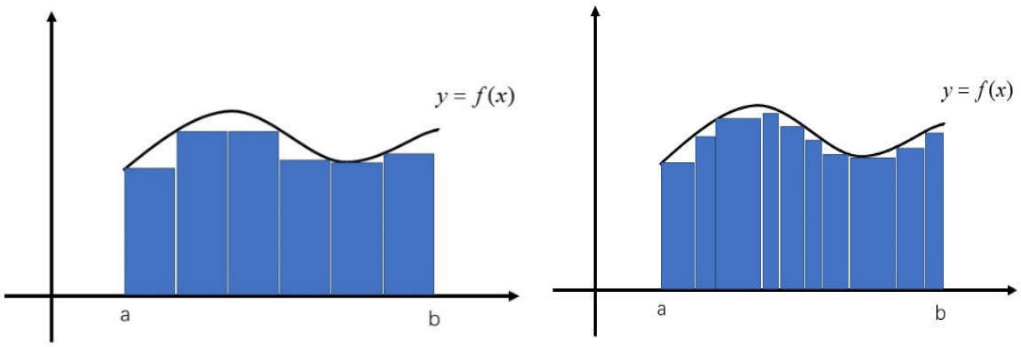
function value corresponding to the $D + 2$ nodes and the function value of the middle node of each subsection. Find the minimum value w_j and the maximum value W_j (j = 1, 2, …, $D + 1$) among the function values of the left endpoint, middle node, and right endpoint of each subsection.

(3) Calculate fitness value. $F(i) = \frac{1}{2}\sum_{j=1}^{D+1} h_j|W_j - w_j|$.

(4) Update individuals through an optimization algorithm.

(5) Repeat step 4 until reaching the stop condition.

(6) Get the accuracy and integral values.



(**a**) (**b**)

Figure 1. Two methods of segmentation when solving numerical integrals: (**a**) equidistant division and (**b**) equidistant division.

The numerical integration method based on Hermite interpolation only needs to provide the value of the integral node functions and has high precision. However, this method is based on equidistant segmentation. In this paper, the adaptability of unequal-spaced partitioning and the numerical integration method based on Hermite interpolation are combined to solve the numerical integration problem, and the formula is as follows:

$$\int_a^b f(x)dx = \sum_{k=1}^{n} \frac{h_i}{2}[f(x_k) + f(x_{k+1})] - \frac{\sum_{i=1}^{n-1} \frac{25}{144} h_i[f(a) + f(b)]}{n-1} + \frac{\sum_{i=1}^{n-1} \frac{h_i}{3}[f(a+h_i) + f(b-h_i)]}{n-1} - \frac{\sum_{i=1}^{n-1} \frac{h_i}{4}[f(a+2h_i) + f(b-2h_i)]}{n-1} + \frac{\sum_{i=1}^{n-1} \frac{h_i}{9}[f(a+3h_i) + f(b-3h_i)]}{n-1} - \frac{\sum_{i=1}^{n-1} \frac{h_i}{48}[f(a+4h_i) + f(b-4h_i)]}{n-1} \quad (5)$$

where n is the number of random split points, h_i is the distance between two adjacent points, and $f(x)$ is the integrand function. The advantage of this method is that it does not need to calculate the derivative value and only needs to provide the node function value. Before using the optimization algorithm to solve the integration, the first step is to convert it into a single-objective optimization problem as follows:

$$\min F(x) = \left| \int_a^b f(x)dx - E \right| \quad (6)$$

where $\int_a^b f(x)dx$ is obtained by Equation (5), and E means the exact value.

Combine the optimization algorithm with Equation (5), and the whole solution process is as follows.

(1) Randomly initialize the population in the search space S.

(2) Arrange each individual in the integral interval in ascending order. The integral interval has $n(n = D + 2)$ nodes and $n - 1$ segments. Calculate the distance h_i between two adjacent nodes and the function $f(x_k)$ value of each node and then bring them into Equation (5).
(3) Calculate the fitness value by Equation (6).
(4) Update individuals through an optimization algorithm.
(5) Repeat step 4 until reaching the stop condition.
(6) Get the accuracy and integral values.

2.3. The Arithmetic Optimization Algorithm (AOA)

The AOA algorithm is a population-based metaheuristic algorithm to solve optimization problems by utilizing mathematical operators (Multiplication ("×"), Division ("÷"), Subtraction ("−"), and Addition ("+")). The specific description is as follows.

2.3.1. Initialization Phase

Generate a candidate solution matrix randomly.

$$X = \begin{pmatrix} x_{1,1} & \cdots & \cdots & x_{1,j} & x_{1,n-1} & x_{1,n} \\ x_{2,1} & \cdots & \cdots & x_{2,j} & x_{2,n-1} & x_{2,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{N-1,1} & \cdots & \cdots & x_{N-1,j} & x_{N-1,n-1} & x_{N-1,n} \\ x_{N,1} & \cdots & \cdots & x_{N,j} & x_{N,n-1} & x_{N,n} \end{pmatrix} \quad (7)$$

After the initialization step, calculate the Math Optimizer Accelerated (MOA) function and use it to choose between exploration and exploitation. The function is as follows:

$$MOA(t) = Min + t \times \left(\frac{Max - Min}{T} \right) \quad (8)$$

where $Max = 0.9$ denotes the maximum and $Min = 0.2$ denotes the minimum of the function value, MOA (t) represents the function value of the current iteration, and T and t represent the maximum number of iterations and current iteration, respectively.

2.3.2. Exploration Phase

During the exploration phase, the operators (Multiplication ("×") and Division ("÷")) are used to explore the space randomly when the $MOA > 0.5$. The mathematical model is as follows:

$$x_{i,j}(t+1) = \begin{cases} best(x_j) \div (MOP + \varepsilon) \times ((UB_j - LB_j) \times \mu + LB_j), r_2 < 0.5 \\ best(x_j) \times MOP \times ((UB_j - LB_j) \times \mu + LB_j), otherwise \end{cases} \quad (9)$$

where r_2 is a random number, $x_{i,j}(t + 1)$ represents the jth position of ith solution in the $(t + 1)$th iteration, $best(x_j)$ denotes the jth position in the global optimal solution, ε is a small integer number that avoids the case where the denominator is zero in division, UB_j and LB_j represents the upper and lower bounds of each dimension, respectively, and μ is equal to 0.5. The Math Optimizer probability (MOP) is as follows:

$$MOP(t) = 1 - \frac{t^{\frac{1}{\alpha}}}{T^{\frac{1}{\alpha}}} \quad (10)$$

where $MOP(t)$ represents the function value for the current iteration, and α is a sensitive parameter and equal to 5.

2.3.3. Exploitation Phase

During the exploration phase, the operators (Subtraction ("−") and Addition ("+")) are used to execute the exploitation. When $MOA < 0.5$, the mathematical model as follows:

$$x_{i,j}(t+1) = \begin{cases} best(x_j) - MOP \times ((UB_j - LB_j) \times \mu + LB_j), r_3 < 0.5 \\ best(x_j) + MOP \times ((UB_j - LB_j) \times \mu + LB_j), otherwise \end{cases} \quad (11)$$

where r_3 is a random number. The pseudo-code of the AOA is as follows (Algorithm 1) [37].

Algorithm 1 AOA

1. Set up the initial parameters α, μ.
2. Initialize the population randomly.
3. for t = 1: T
4. Calculate the fitness function and select the best solution.
5. Update the *MOA* (using Equation (8)) and *MOP* (using Equation (10)).
6. for i = 1: N
7. for j = 1: Dim
8. Generate the random values between [0, 1] (r_1, r_2, r_3)
9. if $r_1 > MOA$
10. if $r_2 > 0.5$
11. Update the position of the individual by Equation (9).
12. else
13. Update the position of the individual by Equation (9).
14. end
15. else
16. if $r_3 > 0.5$
17. Update the position of the individual by Equation (11).
18. else
19. Update the position of the individual by Equation (11).
20. end
21. end
22. end
23. end
24. $t = t + 1$
25. end
26. Return the best solution (x).

3. Our Proposed IAOA

3.1. Motivation for Improving the AOA

In AOA, the population is updated based on the optimal global solution. Once it falls into the optimal local one, the entire population will stagnate. There is premature coverage, in some cases [33]. In addition, this algorithm does not fully utilize the information of the individuals in the population. Therefore, to make full use of the information of the individuals and address the weakness of AOA, the improved arithmetic optimization algorithm (IAOA) is proposed in this paper.

3.2. Population Control Mechanism

In the basic arithmetic optimization algorithm (AOA), the operators (Multiplication ("×"), Division ("÷"), Subtraction ("−"), and Addition ("+")) are used to wrap around an optimal solution to search randomly in space, and it will lead to a loss of population diversity. Therefore, it is necessary to classify for the population.

3.2.1. The First Subpopulation

Sort the population according to the fitness value and select the first *num_best* individuals as the first subpopulation:

$$num\_best = round(0.1N + 0.5N(1 - t/T)) \tag{12}$$

where N is the number of individuals, and t and T represent the current iteration and maximum iterations, respectively. Then, these individuals update their position by getting information about each other. The mathematical model is as follows:

$$x_{best\_i}(t+1) = x_{best\_i}(t) + rand \times \left(best(x) - \frac{x_{best\_i}(t) + x_{best\_j}(t)}{2} \times \omega \right) \tag{13}$$

$$x_{best\_j}(t+1) = x_{best\_j}(t) + rand \times \left(best(x) - \frac{x_{best\_i}(t) + x_{best\_j}(t)}{2} \times \omega \right) \tag{14}$$

where $x_{best\_i}(t + 1)$ denotes the position of ith individual in the next iteration, the same as $x_{best\_j}(t + 1)$, $best(x)$ represents the global optimum that has been found through individuals after t iterations, x_{best_j} is selected from the first class randomly, and ω means the information acquisition rate and takes the value 1 or 2.

3.2.2. The Second Subpopulation

Select *num_middle* individuals from the population as the second subpopulation.

$$num\_middle = round(0.3 \times N) \tag{15}$$

These individuals fall between *num_best* and *num_worst* in the population. Then, these individuals update their position, and the updated model is as follows:

$$x_{mid\_i}(t+1) = x_{mid\_i}(t) + Levy \times (best(x) - x_{mid\_j}) \tag{16}$$

where $x_{mid\_i}(t + 1)$ denotes the position of ith individual in the next iteration, *Levy* is the Levy distribution function [45,46], and x_{mid_j} is selected from the second class randomly.

3.2.3. The Third Subpopulation

Select *num_worst* individuals from the population as the final subpopulation.

$$num\_worst = N - (num\_best + num\_middle) \tag{17}$$

In the final class, the individuals update their position by the following equation:

$$x_{worst\_i}(t+1) = x_{worst\_i} + \left(\frac{t}{T} \times best(x) - x_{worst\_j} \right) \tag{18}$$

where $x_{worst\_i}(t + 1)$ denotes the position of ith individual in the next iteration, and $best(x)$ represents the global optimum that has been found through individuals after t iterations.

At the early iteration of IAOA, there are more individuals in the first subpopulation for speeding up the update of the global optimum. At the later iterations of the algorithm, the number of individuals in the first subpopulation decreases, which solves the operator crowding problem near the optimum. In addition, the number of individuals in the third subpopulation increases, which effectively prevents the population from falling into the local optimum. The second subpopulation utilizes the Levy flight for small-step updates to find more promising areas. The above strategy can effectively overcome the weaknesses of traditional AOA and improve its performance. The pseudo-code of the IAOA in algorithm 2 is as follows (Algorithm 2). Figure 2 is the flowchart of the IAOA.

Algorithm 2 IAOA

1. Set up the initial parameters α, μ.
2. Initialize the population randomly.
3. for t = 1: T
4. Calculate the fitness function and select the best solution.
5. Calculate the number of the first subpopulation by Equation (12).
6. Update the first subpopulation by Equations (13) and (14).
7. Calculate the number of the second subpopulation by Equation (15).
8. Update the second subpopulation by Equation (16).
9. Calculate the number of the third subpopulation by Equation (17).
10. Update the third subpopulation by Equation (18).
11. Update the *MOA* (using Equation (8)) and *MOP* (using Equation (10)).
12. for i =1: N
13. for j = 1: Dim
14. Generate the random values between [0, 1] (r_1, r_2, r_3)
15. if $r_1 > MOA$
16. if $r_2 > 0.5$
17. Update the position of the individual by Equation (9).
18. else
19. Update the position of the individual by Equation (9).
20. end
21. else
22. if $r_3 > 0.5$
23. Update the position of the individual by Equation (11).
24. else
25. Update the position of the individual by Equation (11).
26. end
27. end
28. end
29. end
30. $t = t + 1$
31. end
32. Return the best solution (x).

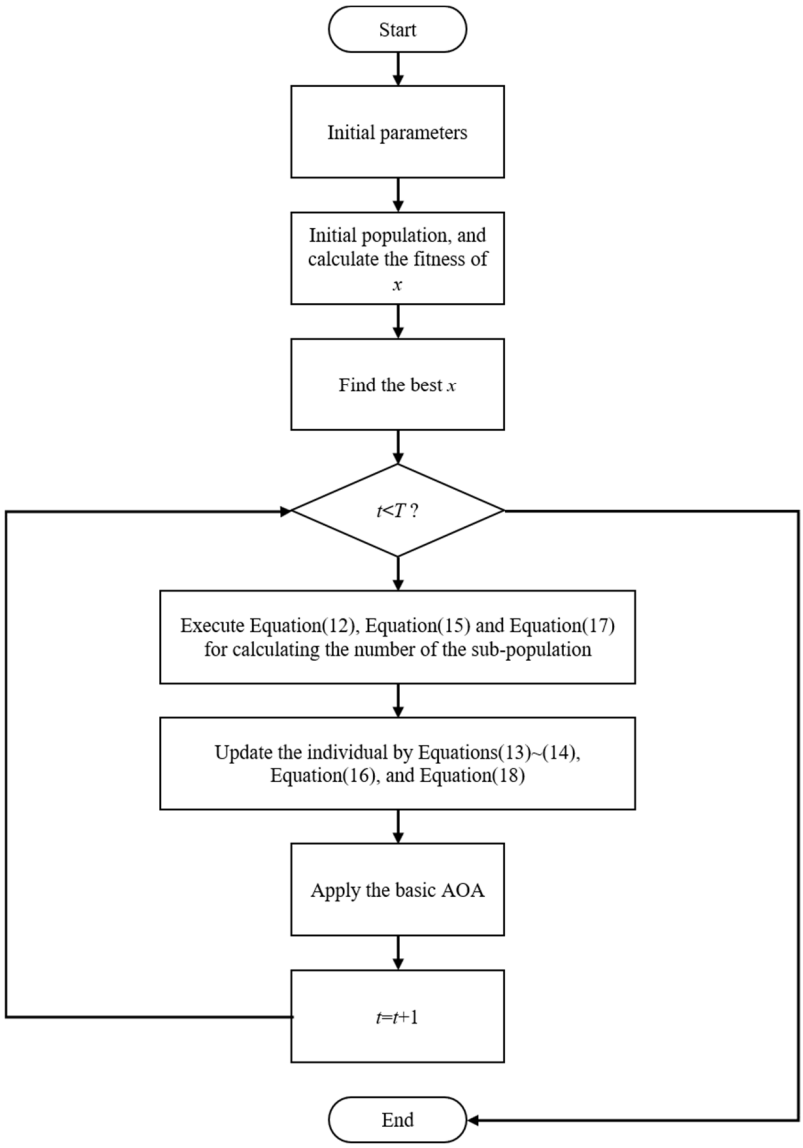


Figure 2. Flowchart of the IAOA.

4. Numerical Experiments and Analysis

4.1. Parameter Settings

Here, six groups of NESs and ten groups of integration have been used to demonstrate the efficiency of the IAOA. The IAOA compares several popular algorithms and two improved arithmetic optimization algorithms (The Arithmetic Optimization Algorithm (AOA) [37], Sine Cosine Algorithm (SCA) [47], Whale Optimization Algorithm (WOA) [48], Grey Wolf Optimizer (GWO) [49], Harris hawks optimization (HHO) [50], Slime mould algorithm (SMA) [51], Differential evolution(DE) [52], Cuckoo search algorithm (CSA) [53], Advanced arithmetic optimization algorithm (nAOA) [40], and a developed version of Arithmetic Optimization Algorithm (dAOA) [42]) for tackling NES. Among them, the parameters of these algorithms are all from the original version. These algorithms are evaluated from four aspects: the average value, the optimal value, the worst value, and the standard deviation. All algorithms are executed on MATLAB 2021a, running on a computer with a Windows 10 operating system, Intel(R) Core (TM) i7-9700 CPU @ 3.00 GHz, 16 GB of Random Access Memory (RAM), and run 30 times independently for all test problems. The flowchart for handling issues by the IAOA is shown in Figure 3.

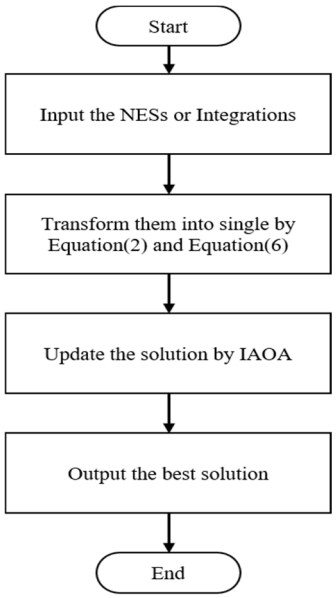


Figure 3. Flowchart for handling issues.

4.2. Application in Solving NESs

Solving nonlinear problems often requires higher-precision solutions in many practical applications. In this section, six nonlinear systems of equations are chosen to evaluate the performance of the IAOA. The characteristics of these equations are different from each other, where problem01 [54] describes the interval arithmetic problem, problem02 [55] describes the multiple steady-states problem, and problem06 [56] describes the molecular conformation. These problems come from real-world applications. For fairness, set the population to 50 and the maximum number of iterations to 200. Tables 1–6 show all the test results of the NES. Best represents the best value, Worst represents the worst value, Mean represents the mean value, Std represents the standard deviation, and *p*-value stands for the Wilcoxon rank–sum test in Table 7. The Wilcoxon *p*-value test is used to verify whether there is an obvious difference between the two sets of data.

Table 1. Comparison of the experimental results for problem01.

Variable	Algorithms			
	AOA	**IAOA**	**SCA**	**WOA**
x_1	0.006361583402960	0.257838650825518	0.186732591196869	0.260832096649832
x_2	0.005731653837062	0.381098185347242	0.399818814038728	0.381680691118263
x_3	0.010586282003880	0.278742562628776	0.008959145137085	0.258353295805450
x_4	0.002593989505334	0.200665586275865	0.227237103605413	0.215307146397956
x_5	0.033520558095432	0.445255928027431	0.003829239926320	0.448797960971748
x_6	0.076424218265631	0.149188813621332	0.185905381801968	0.147397359179682
x_7	0.038862694473151	0.432010769672038	0.368813050526818	0.442390776062597
x_8	−0.000004007877210	0.073406152818720	0.037739989370997	0.137586270569043
x_9	0.029054432130685	0.345966262513093	0.206476235144125	0.342058064566263
x_{10}	0.013690425703394	0.427324518269459	0.363350844915327	0.401475021739693
f	$8.45665838921712 \times 10^{-1}$	$4.73405913551646 \times 10^{-10}$	$1.22078391539763 \times 10^{-1}$	$9.59544885085295 \times 10^{-4}$
Variable	**Algorithms**			
	GWO	**HHO**	**DE**	**CSO**
x_1	0.256851024248810	0.324317023967532	2.000000000000000	0.089951372914250
x_2	0.383565743620699	0.303967192642514	1.948157453190990	0.309487131659014
x_3	0.278312335483674	0.216191961411362	2.000000000000000	0.456410156556233
x_4	0.198737300040942	0.305260974230829	1.815308511546580	0.356392775439902

x_5	0.446311619177502	0.325255783591842	2.000000000000000	0.476086684751138
x_6	0.145894138632280	0.223020351676054	2.000000000000000	0.078921332097133
x_7	0.145894138632280	0.323185143014029	2.000000000000000	0.499580490394335
x_8	−0.007832029555062	0.327973609353822	1.915762141824520	0.197756675883883
x_9	0.343654620394334	0.333430854648433	2.000000000000000	0.228228833675487
x_{10}	0.425902664080806	0.324142888370713	2.000000000000000	0.470195948900759
f	$1.25544451911646 \times 10^{-3}$	$7.79220329211044 \times 10^{-2}$	$7.96261500819178 \times 10^{-2}$	$6.61705221934444 \times 10^{-2}$

Variable	Algorithms		
	SMA	**nAOA**	**dAOA**
x_1	0.249900132290417	0.035430633051580	1.840704485033870
x_2	0.375428314977531	0.053983062784772	1.213421005935260
x_3	0.272448580296318	0.072735305166021	1.203555993641700
x_4	0.199698265955405	0.021399042985613	−0.393935624266822
x_5	0.425934189445810	0.064655913970964	−0.249476549706985
x_6	0.057699959645613	0.012570281350831	0.459915310960444
x_7	0.431865275874618	0.057639809639213	−0.675754718182326
x_8	0.015005640000641	0.005520004765830	−0.895856414267328
x_9	0.347986992756388	0.041229484511092	0.359139808282465
x_{10}	0.415304164782275	0.079595719921909	1.529188120361250
f	$4.47411205566240 \times 10^{-3}$	$6.74563715208325 \times 10^{-1}$	1.91503507134915

Table 2. Comparison of the experimental results for problem02.

Variable	Algorithms			
	AOA	**IAOA**	**SCA**	**WOA**
x_1	0.040781958181860	0.042124781715274	0.000000000000000	0.041561373108785
x_2	0.268625655728691	0.061754610138946	0.266593748985495	0.268697327813652
f	$2.01752031872803 \times 10^{-7}$	$9.24446373305873 \times 10^{-34}$	$8.82826387279195 \times 10^{-5}$	$6.92247231102962 \times 10^{-9}$
Variable	**Algorithms**			
	GWO	**HHO**	**DE**	**CSO**
x_1	0.265622854930434	0.267855297066815	0.266589101862370	0.266620164671422
x_2	0.178718146817611	0.458749279058429	0.327275026016101	0.178514261126008
f	$1.13985864694418 \times 10^{-7}$	$6.55986405733090 \times 10^{-8}$	$1.31654979128584 \times 10^{-18}$	$1.49504500886345 \times 10^{-9}$
Variable	**Algorithms**			
	SMA	**nAOA**	**dAOA**	
x_1	0.021419624272050	0.000000000000000	0.236558250181286	
x_2	0.048075232460874	0.719124811309122	0.508933311549167	
f	$2.89316821274146 \times 10^{-5}$	$3.07109081317222 \times 10^{-5}$	$3.22387407689191 \times 10^{-4}$	

Table 3. Comparison of the experimental results for problem03.

Variable	Algorithms			
	AOA	**IAOA**	**SCA**	**WOA**
x_1	1.990744078311880	−0.947268146986263	−0.225974226141413	−1.424482905343090
x_2	0.220001522814532	−0.785020015568289	1.245763361231140	−0.543544840817441
f	$5.61739095968327 \times 10^{-3}$	$4.02151576372412 \times 10^{-32}$	$7.95691890654021 \times 10^{-4}$	$1.06331568826728 \times 10^{-3}$
Variable	**Algorithms**			
	GWO	**HHO**	**DE**	**CSO**
x_1	−1.794053112053940	−1.495480498807310	−1.791308474954350	−0.212779003619775
x_2	−0.303905803005920	−0.420394691864127	0.301889327351144	−1.257141525856050
f	$2.77808608355359 \times 10^{-5}$	$6.12298193031725 \times 10^{-5}$	$1.84881969881973 \times 10^{-9}$	$6.26348225916795 \times 10^{-7}$
Variable	**Algorithms**			

	SMA	nAOA	dAOA
x_1	−1.791387180972800	−1.475077261850100	−1.580085715978880
x_2	−0.302157020359872	−0.454673564762598	0.4651484d76848022
f	$5.47910691165820 \times 10^{-8}$	$2.17709293383390 \times 10^{-4}$	$5.12705019470938 \times 10^{-2}$

Table 4. Comparison of the experimental results for problem04.

Variable	Algorithms			
	AOA	IAOA	SCA	WOA
x_1	−0.000266868453558	−0.000000091835793	−0.120898772911816	−0.310246574315981
x_2	−0.000267036157051	0.000013971597535	0.491167568359585	0.467564824328878
x_3	−0.000267036274281	0.000030454051416	10.000000000000000	1.071469773086650
x_4	0.000000025430197	0.000010000404353	−0.178108600809833	−0.404219784214681
x_5	−0.000267039311495	0.000011275918099	5.423242568753400	3.552125620609660
x_6	−0.000267036127224	0.000000019800029	−0.049710980654501	−1.834136698070800
x_7	0.000000000091855	−0.000000000138437	0.445662462511328	0.286050311387620
x_8	0.000267036101457	−0.000000454282127	−10.000000000000000	−2.931846497771810
x_9	0.000267033832224	0.000000000736505	−0.144419405019169	−4.812450845354100
x_{10}	0.000267043884482	−0.000002006069864	−0.518105971932846	3.756426716000660
f	$1.08498006397337 \times 10^{-9}$	$7.03339003909689 \times 10^{-16}$	$4.13237426374674 \times 10^{-1}$	$6.47066501369328 \times 10^{-1}$
Variable	**Algorithms**			
	GWO	**HHO**	**DE**	**CSO**
x_1	0.044653752694561	−0.000047703379713	0.160723693838569	−0.009650846541198
x_2	−0.259567674882923	0.000075691075249	0.431923139718368	0.147278561202585
x_3	−1.777013199398760	−0.000029713372367	0.072922517980119	−3.148557575646470
x_4	0.042606334458592	−0.000050184914825	0.447403957744849	−0.512428980703464
x_5	−4.935286036663600	0.000033675529531	−0.197972459731190	−4.175819684412100
x_6	−8.146156623785810	0.000067989452634	1.490110445009050	−7.123183974281880
x_7	−0.108125274969201	0.000031288762826	0.472265426079125	1.268663892956760
x_8	1.747052457418910	0.000048491290536	0.509493705510866	3.198230908839320
x_9	−0.311997778279745	0.000063892452193	1.142101578993260	−4.763105818868310
x_{10}	8.430357427064680	−0.000123055431652	−2.110335475212350	9.463108408596410
f	$7.56734706927375 \times 10^{-3}$	$6.11971561041781 \times 10^{-10}$	$9.87501536049260 \times 10^{-1}$	2.18295386757873
Variable	**Algorithms**			
	SMA	**nAOA**	**dAOA**	
x_1	−0.000000000028677	0.000020144848903	−0.934997016811202	
x_2	0.000014644312649	−0.000060200695401	−1.295640443505010	
x_3	0.000038790339140	−0.000020118018817	−5.634966911723890	
x_4	−0.000000000221797	−0.000060200956330	−4.825343892476190	
x_5	0.000000055701981	−0.000020122803817	0.269511140973028	
x_6	−0.000000030051237	−0.000020134693956	−7.253398121182340	
x_7	0.000000595936232	0.000020123341500	7.557747336452660	
x_8	−0.000000000025333	0.000020925519435	−5.520361069927860	
x_9	0.000000799504725	0.000043615727680	−4.709534880735350	
x_{10}	0.000000000012983	0.000020120622373	8.954470788407880	
f	$1.30095438660555 \times 10^{-10}$	$1.50696700666871 \times 10^{-9}$	$2.07190542503982 \times 10^{2}$	

Table 5. Comparison of the experimental results for problem05.

Variable	Algorithms			
	AOA	**IAOA**	**SCA**	**WOA**
x_1	0.371964486871792	0.500000000000000	0.471178994397267	0.503978268408352
x_2	2.990337880814430	3.141592653589790	3.118271172186020	3.142976305563530
f	$1.89048835343036 \times 10^{-4}$	$1.85873810048745 \times 10^{-28}$	$3.41504906318340 \times 10^{-5}$	$2.00099014478417 \times 10^{-7}$
Variable	**Algorithms**			
	GWO	**HHO**	**DE**	**CSO**
x_1	0.495722089382004	0.503332577729795	0.299448692445072	0.500482294032500
x_2	3.143566564341090	3.142753305279310	2.836927770362990	3.142098043614560
f	$1.12835512797232 \times 10^{-6}$	$1.16071617155615 \times 10^{-7}$	$6.25300383824133 \times 10^{-23}$	$2.13609775136897 \times 10^{-8}$
Variable	**Algorithms**			
	SMA	**nAOA**	**dAOA**	
x_1	0.298949061647857	0.354640044143990	2.956994389007600	
x_2	2.835691250750600	2.956994389007600	1.890717921128260	
f	$1.05189651760469 \times 10^{-8}$	$1.59376404093113 \times 10^{-4}$	$3.65946616757579 \times 10^{-3}$	

Table 6. Comparison of the experimental results for problem06.

Variable	Algorithms			
	AOA	**IAOA**	**SCA**	**WOA**
x_1	0.953663829653960	−0.779548045079158	11.147659127176500	1.516510183032980
x_2	0.663112382731748	−0.779548045079158	0.900762400732728	0.694394649388567
x_3	0.729782844271910	−0.779548045079158	0.919816117314499	10.556407054559600
f	$3.35330112498813 \times 10^{-1}$	$1.00553388370096 \times 10^{-20}$	2.75666643131973	8.65817545834561
Variable	**Algorithms**			
	GWO	**HHO**	**DE**	**CSO**
x_1	0.781303537791760	−0.782460718139219	−0.779277448448367	−0.765447632695953
x_2	0.777872878718449	−0.789339702437282	−0.779700789186745	−0.784775197498564
x_3	0.779780469890485	−0.766810453292313	−0.780020611467694	−0.735052686517780
f	$5.49159538279891 \times 10^{-4}$	$1.00882211687459 \times 10^{-2}$	$6.71295836563811 \times 10^{-6}$	$2.92512803990831 \times 10^{-1}$
Variable	**Algorithms**			
	SMA	**nAOA**	**dAOA**	
x_1	−0.779731780102931	−0.437772635064718	−1.056395480177350	
x_2	−0.779371556451744	−7.659741643877890	6.893981344148980	
x_3	−0.779303513685515	−2.620897335617900	−1.876924860155790	
f	$1.03517116885362 \times 10^{-5}$	1.49720612584788	$2.61017698945353 \times 10^{4}$	

Table 7. Statistical results for the NES.

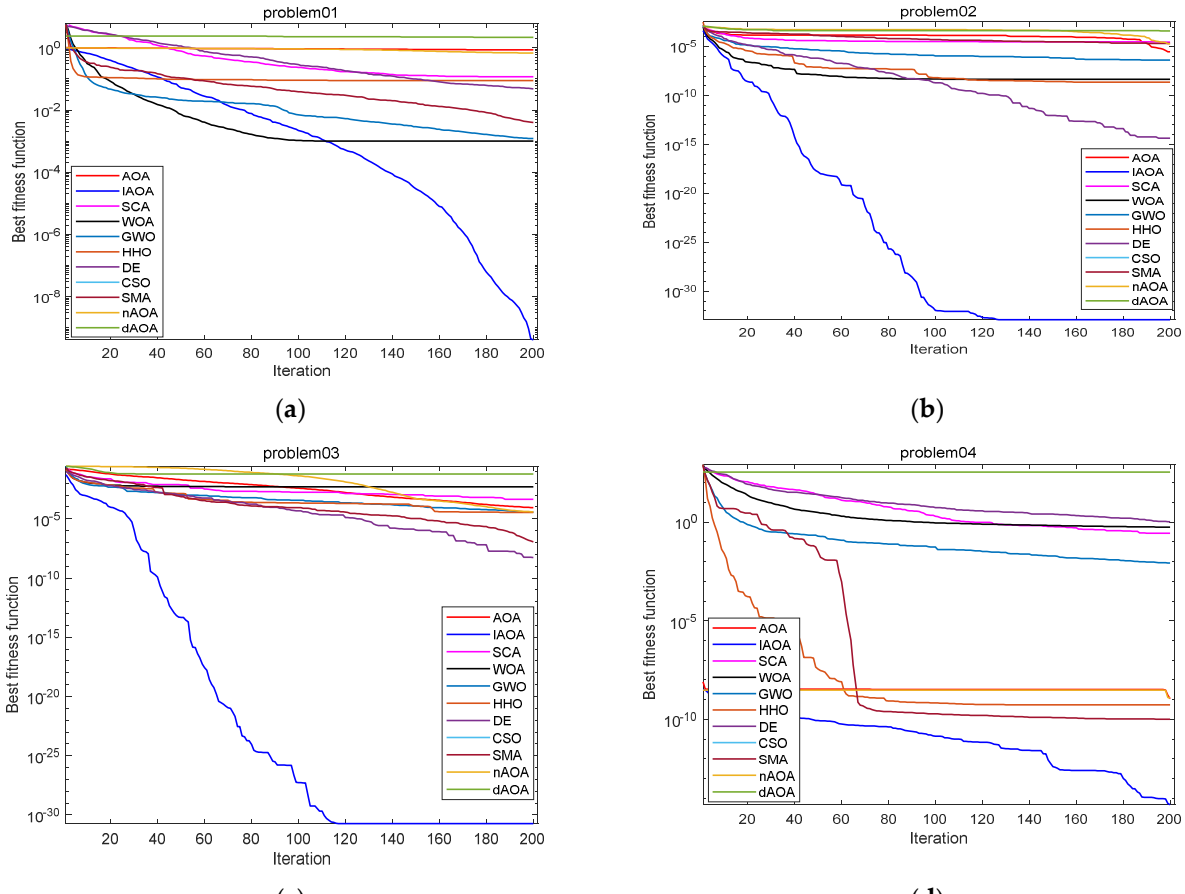
Algorithms		Systems of Nonlinear Equations					
		problem01	**problem02**	**problem03**	**problem04**	**problem05**	**problem06**
AOA	best	7.02711×10^{-1}	1.20198×10^{-8}	8.30574×10^{-12}	2.99534×10^{-10}	5.32587×10^{-6}	1.60969×10^{-8}
	worst	9.05980×10^{-1}	7.47231×10^{-7}	9.55457×10^{-3}	3.58264×10^{-9}	5.96026×10^{-4}	1.00599×10
	mean	8.45666×10^{-1}	2.01752×10^{-7}	3.18486×10^{-4}	1.08498×10^{-9}	1.89049×10^{-4}	3.35330×10^{-1}
	std	4.40686×10^{-2}	1.78065×10^{-7}	1.74442×10^{-3}	8.49280×10^{-10}	1.40374×10^{-4}	1.83668
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}
IAOA	best	1.05462×10^{-10}	0.00000	4.93038×10^{-32}	2.97972×10^{-19}	0.00000	1.81191×10^{-30}
	worst	1.25230×10^{-9}	3.08149×10^{-33}	2.09541×10^{-31}	5.52546×10^{-15}	5.57614×10^{-27}	2.98754×10^{-19}
	mean	4.73406×10^{-10}	9.24446×10^{-34}	7.27231×10^{-32}	7.03339×10^{-16}	1.85874×10^{-28}	1.00553×10^{-20}
	std	2.84371×10^{-10}	1.43626×10^{-33}	4.02152×10^{-32}	1.22291×10^{-15}	1.01806×10^{-27}	5.45273×10^{-20}

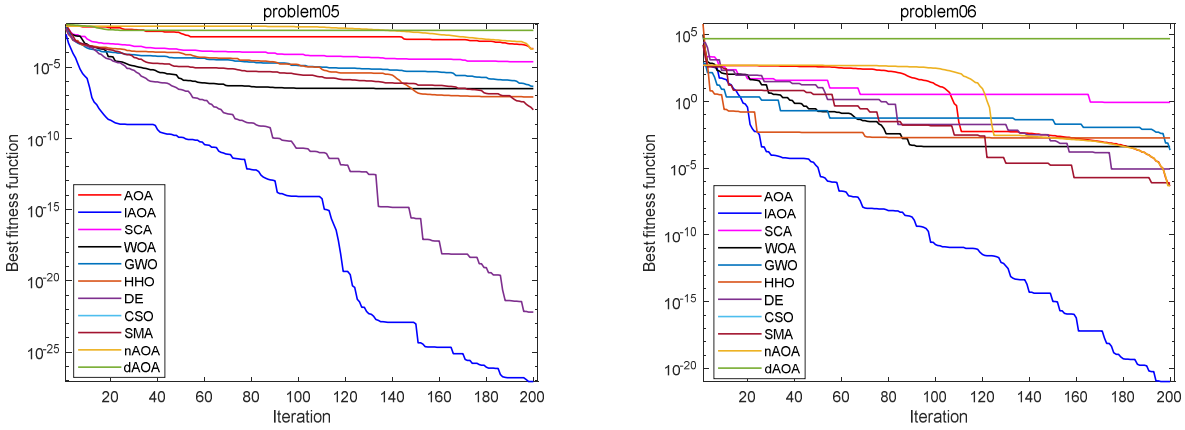
SCA	best	4.64629×10^{-2}	1.20156×10^{-8}	8.29788×10^{-6}	7.08592×10^{-4}	7.53679×10^{-9}	1.19890×10^{-1}
	worst	2.98744×10^{-1}	8.60445×10^{-4}	3.13588×10^{-3}	2.83503	2.00649×10^{-4}	3.29896×10
	mean	1.22078×10^{-1}	8.82826×10^{-5}	5.47683×10^{-4}	4.13237×10^{-1}	3.41505×10^{-5}	2.75667
	std	5.72692×10^{-2}	2.61875×10^{-4}	7.59630×10^{-4}	6.58494×10^{-1}	4.69615×10^{-5}	6.25475
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}
WOA	best	1.87873×10^{-4}	6.72146×10^{-14}	6.18945×10^{-13}	4.04945×10^{-6}	2.16928×10^{-11}	1.76476×10^{-5}
	worst	5.56233×10^{-3}	1.30541×10^{-7}	4.48907×10^{-2}	4.99725	4.78904×10^{-6}	7.91148×10
	mean	9.59545×10^{-4}	6.92247×10^{-9}	4.26773×10^{-3}	6.47067×10^{-1}	2.00099×10^{-7}	8.65818
	std	1.06419×10^{-3}	2.49080×10^{-8}	1.24385×10^{-2}	1.07197	8.71177×10^{-7}	2.24136×10
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}
GWO	best	2.65480×10^{-6}	2.31886×10^{-12}	1.77817×10^{-8}	1.01688×10^{-6}	2.21126×10^{-9}	9.05730×10^{-5}
	worst	6.59898×10^{-3}	1.73256×10^{-6}	9.94266×10^{-2}	5.57604×10^{-2}	1.70979×10^{-5}	1.58625×10^{-3}
	mean	1.25544×10^{-3}	1.13986×10^{-7}	3.33932×10^{-3}	7.56735×10^{-3}	1.12836×10^{-6}	5.49160×10^{-4}
	std	2.25868×10^{-3}	4.16137×10^{-7}	1.81481×10^{-2}	1.36923×10^{-2}	3.33417×10^{-6}	3.69947×10^{-4}
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}
HHO	best	2.03768×10^{-2}	8.99794×10^{-31}	4.93038×10^{-32}	1.21192×10^{-11}	7.70372×10^{-34}	3.83242×10^{-5}
	worst	1.33302×10^{-1}	1.91904×10^{-6}	5.78702×10^{-4}	1.00491×10^{-9}	3.34700×10^{-6}	7.08247×10^{-2}
	mean	7.79220×10^{-2}	6.55986×10^{-8}	4.12782×10^{-5}	6.11972×10^{-10}	1.16072×10^{-7}	1.00882×10^{-2}
	std	2.90524×10^{-2}	3.50117×10^{-7}	1.19896×10^{-4}	2.78236×10^{-10}	6.10656×10^{-7}	1.45023×10^{-2}
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	5.56066×10^{-8}	3.01986×10^{-11}	1.30542×10^{-10}	3.01230×10^{-11}
DE	best	6.05782×10^{-3}	8.15969×10^{-28}	2.49399×10^{-20}	2.59514×10^{-1}	2.59615×10^{-31}	4.23182×10^{-11}
	worst	9.69921×10^{-1}	1.19322×10^{-17}	5.91181×10^{-7}	2.58615	6.37964×10^{-22}	1.17012×10^{-4}
	mean	7.96262×10^{-2}	1.31655×10^{-18}	3.33313×10^{-8}	9.87502×10^{-1}	6.25300×10^{-23}	6.71296×10^{-6}
	std	2.40157×10^{-1}	2.91169×10^{-18}	1.26981×10^{-7}	6.21653×10^{-1}	1.66035×10^{-22}	2.15862×10^{-5}
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	6.22236×10^{-11}	3.01230×10^{-11}
CSO	best	2.82411×10^{-2}	7.30711×10^{-11}	2.92752×10^{-9}	6.03864×10^{-1}	2.67109×10^{-10}	2.27267×10^{-2}
	worst	1.34962×10^{-1}	7.15408×10^{-9}	2.57784×10^{-6}	4.34942	1.32416×10^{-7}	1.31894
	mean	6.61705×10^{-2}	1.49505×10^{-9}	6.53698×10^{-7}	2.18295	2.13610×10^{-8}	2.92513×10^{-1}
	std	2.71383×10^{-2}	1.66707×10^{-9}	5.69101×10^{-7}	1.05318	3.36401×10^{-8}	3.41112×10^{-1}
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}
SMA	best	5.18988×10^{-4}	1.26496×10^{-7}	2.37253×10^{-11}	2.08208×10^{-11}	6.22359×10^{-11}	3.95601×10^{-7}
	worst	1.17331×10^{-2}	2.46549×10^{-4}	5.80093×10^{-7}	2.89907×10^{-10}	5.94920×10^{-8}	4.75099×10^{-5}
	mean	4.47411×10^{-3}	2.89317×10^{-5}	5.98652×10^{-8}	1.30095×10^{-10}	1.05190×10^{-8}	1.03517×10^{-5}
	std	3.00476×10^{-3}	5.64857×10^{-5}	1.28713×10^{-7}	7.25135×10^{-11}	1.30068×10^{-8}	1.04158×10^{-5}
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}
nAOA	best	4.73537×10^{-1}	1.16733×10^{-9}	3.11364×10^{-12}	3.28064×10^{-10}	2.13953×10^{-5}	7.56334×10^{-8}
	worst	7.39125×10^{-1}	9.06936×10^{-4}	8.22290×10^{-1}	2.69391×10^{-9}	4.30978×10^{-4}	4.49162×10
	mean	6.74564×10^{-1}	3.07109×10^{-5}	2.77064×10^{-2}	1.50697×10^{-9}	1.59376×10^{-4}	1.49721
	std	5.68300×10^{-2}	1.65502×10^{-4}	1.50077×10^{-1}	6.31248×10^{-10}	7.06193×10^{-5}	8.20053
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}
dAOA	best	2.01052×10^{-1}	8.99368×10^{-9}	2.54429×10^{-4}	3.09426×10^{-10}	5.69606×10^{-6}	8.50407×10^{-4}
	worst	6.87872	1.28121×10^{-3}	4.68145×10^{-1}	9.87499×10^{2}	1.56431×10^{-2}	3.78263×10^{5}
	mean	1.91504	3.22387×10^{-4}	6.56368×10^{-2}	2.07191×10^{2}	3.65947×10^{-3}	2.61018×10^{4}
	std	2.16147	3.20053×10^{-4}	1.21675×10^{-1}	2.92259×10^{2}	5.26309×10^{-3}	8.07193×10^{4}
	p-value	3.01986×10^{-11}	1.01490×10^{-11}	1.07516×10^{-11}	3.01986×10^{-11}	1.49399×10^{-11}	3.01230×10^{-11}

Problem 01. The description of the system is as follows [54]:

$$\begin{cases} x_1 - 0.25428722 - 0.18324757x_4x_3x_9 = 0 \\ x_2 - 0.37842197 - 0.16275449x_1x_{10}x_6 = 0 \\ x_3 - 0.27162577 - 0.16955071x_1x_2x_{10} = 0 \\ x_4 - 0.19807914 - 0.15585316x_7x_1x_6 = 0 \\ x_5 - 0.44166728 - 0.19950920x_7x_6x_3 = 0 \\ x_6 - 0.14654113 - 0.18922793x_8x_5x_{10} = 0 \\ x_7 - 0.42937161 - 0.21180486x_2x_5x_8 = 0 \\ x_8 - 0.07056438 - 0.17081208x_1x_7x_6 = 0 \\ x_9 - 0.34504906 - 0.19612740x_{10}x_6x_8 = 0 \\ x_{10} - 0.42651102 - 0.21466544x_4x_8x_1 = 0 \end{cases} \quad (19)$$

There are ten equations in the system, where $x_i \in [-2,2]$, $i = 1, \ldots, n$, and $n = 10$. The aim was to obtain a higher precision solution x ($x_1, \ldots, x_n$) through the proposed optimization method, and the results are recorded in Table 1. The IAOA is better than others compared with several algorithms. The WOA ranks second, and the rest obtain competitive results. The convergence curve for this problem shows in Figure 4a.





(**e**) (**f**)

Figure 4. Convergence curve for tackling the NES (problem01–06 (**a**–**f**)).

Problem 02. The description of the system is as follows [55]:

$$\begin{cases} (1-R)\left[\left(\dfrac{D}{10(1+\beta_1)}-x_1\right)\cdot\exp\left(\dfrac{10x_1}{1+\dfrac{10x_1}{\gamma}}\right)\right]-x_1=0 \\ (1-R)\left[\left(\dfrac{D}{10}-\beta_1 x_1-(1+\beta_2)x_2\right)\cdot\exp\left(\dfrac{10x_2}{1+\dfrac{10x_2}{\gamma}}\right)\right]+x_1-(1+\beta_2)x_2=0 \end{cases} \quad (20)$$

There are two equations in system, where $x_i \in [0,1]$, $i = 1, \ldots, n$, and $n = 2$. In Table 2, the experimental results for this problem proved that the proposed IAOA outperforms the other methods. The DE ranks second, and the rest obtain competitive results. The AOA, WOA, GWO, HHO, and CSO are in the third echelon. Furthermore, the rest are in the fourth echelon. The convergence curve for this problem is shown in Figure 4b.

Problem 03. The description of the system is as follows [13]:

$$\begin{cases} \sin\left(x_1^3\right)-3x_1x_2^2-1=0 \\ \cos\left(3x_1^2x_2\right)-\left|x_2^3\right|+1=0 \end{cases} \quad (21)$$

There are two equations in the system, where $x_i \in [-2,2]$, $i = 1, \ldots, n$, and $n = 2$. The simulation results for this problem are shown in Table 3. It revealed that the IAOA is better than the other algorithms. The DE, CSO, and SMA are in the second echelon. The rest are in the third echelon. The convergence curve for this problem is shown in Figure 4c.

Problem 04. The description of the system is as follows [54]:

$$\begin{cases} x_2+2x_6+x_9+2x_{10}-10^{-5}=0 \\ x_3+x_8-3\cdot 10^{-5}=0 \\ x_1+x_3+2x_5+2x_8+x_9+x_{10}-5\cdot 10^{-5}=0 \\ x_4+2x_7-10^{-5}=0 \\ 0.5140437\cdot 10^{-7}x_5-x_1^2=0 \\ 0.1006932\cdot 10^{-6}x_6-2x_2^2=0 \\ 0.7816278\cdot 10^{-15}x_7-x_4^2=0 \\ 0.1496236\cdot 10^{-6}x_8-x_1x_3=0 \\ 0.6194411\cdot 10^{-7}x_9-x_1x_2=0 \\ 0.2089296\cdot 10^{-14}x_{10}-x_1x_2^2=0 \end{cases} \tag{22}$$

There are ten equations in the system: $x_i \in [-10,10]$, i = 1, …, n, and n = 10. Table 4 shows that the IAOA outperforms the others, and AOA, HHO, SMA, and nAOA obtain the competitive results. The convergence curve for this problem is shown in Figure 4d.

Problem 05. The description of the system is as follows [17]:

$$\begin{cases} 0.5\sin(x_1x_2)-\dfrac{0.25}{\pi}x_2-0.5x_1=0 \\ \left(1-\dfrac{0.25}{\pi}\right)\left[\exp(2x_1)-e\right]+\dfrac{e}{\pi}x_2-2ex_1=0 \end{cases} \tag{23}$$

There are two equations in the system, where $x_1 \in [0.25,1]$ and $x_2 \in [1.5,2\pi]$. In Table 5, the IAOA obtained the optimal solution, DE obtained the suboptimal solution, and the rest of the algorithms obtained competitive results. The convergence curve for this problem is shown in Figure 4e.

Problem 06. The description of the system is as follows [56]:

$$\begin{cases} \beta_{11}+\beta_{12}x_2^2+\beta_{13}x_3^2+\beta_{14}x_2x_3+\beta_{15}x_2^2x_3^2=0 \\ \beta_{21}+\beta_{22}x_3^2+\beta_{23}x_1^2+\beta_{24}x_3x_1+\beta_{25}x_3^2x_1^2=0 \\ \beta_{31}+\beta_{32}x_1^2+\beta_{33}x_2^2+\beta_{34}x_1x_2+\beta_{35}x_1^2x_2^2=0 \end{cases} \tag{24}$$

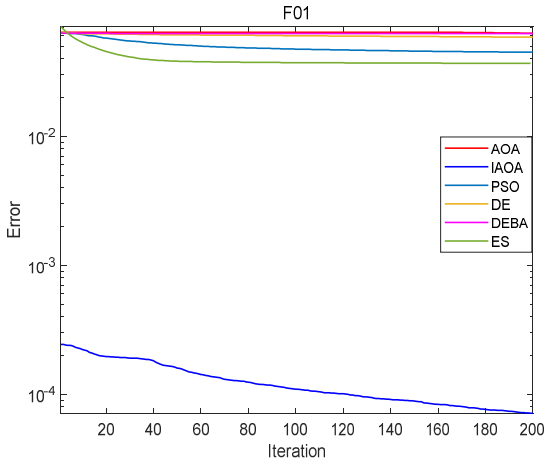
There are three equations in the system, where the details about β_{ij} can be found in the literature [56]: $x_i \in [-20,20]$, i = 1, …, n, and n = 3. In Table 6, the proposed IAOA outperforms the other algorithms; the GWO, SMA, and DE get competitive results. The convergence curve for this problem is shown in Figure 4f.

The statistical results show that the IAOA outperforms all algorithms on the remaining problems in Table 7. These demonstrate that the IAOA has stronger ability and higher stability than the other methods when solving a nonlinear system of equations. In Figure 4, IAOA's convergence speed is slower than the others before the 110th iteration, but after that, the IAOA still maintains a high convergence speed and achieves the optimum at the 200th iteration for problem01; for problem02 and problem03, the IAOA has the fastest speed throughout the whole process and reaches the optimum at the 120th iteration and before 120 iterations, respectively; for problem04, the IAOA is slower than the other algorithms before 70 iterations; however it continues to converge after that and obtains the optimal value after 200 iterations; for problem05, there is a close convergence rate for the IAOA and DE, but a better value is obtained by the IAOA; for problem06, it has a slower convergence speed than the others before 20 iterations, but after that, the fastest

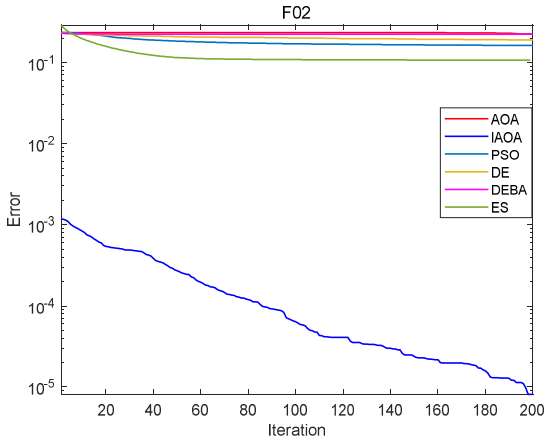
convergence rate is obtained by the IAOA. All the experimental results prove that the algorithm proposed in this paper has the characteristics that include a fast convergence speed, high convergence accuracy, high solution quality, good stability, and strong robustness when dealing with nonlinear systems of equations. The *p*-values of almost all test functions in the table are less than 0.05, indicating that the IAOA is significantly different from the other algorithms.

4.3. Numerical Integration

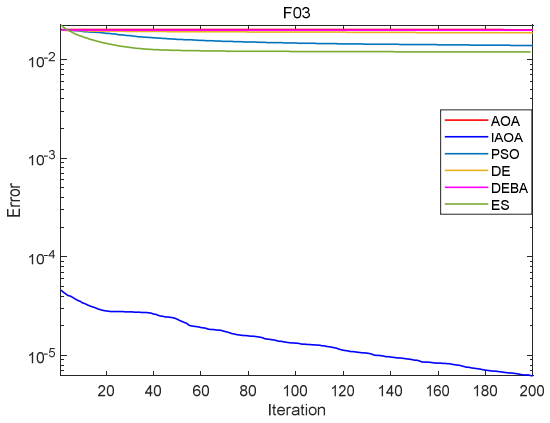
The performance of the proposed new method is evaluated in this section using the ten numerical integration problems in Table 8, where F08 is a singular integral and F10 is an oscillatory integral. The IAOA compared with the traditional methods and population-based algorithms in tackling these cases. Tables 9–12 show the best integral values obtained by solving ten problems in 30 independent runs, where the R-method, T-method, S-method, H-method, G32, and 2n × L5 represent the traditional methods (rectangle method, trapezoid method, Simpson method, Hermite interpolation method, the 32-point Gaussian formula, and the 5-point Gauss-Roberto-Legendre formula). The rest are swarm intelligence algorithms applied to solve numerical integration problems (evolutionary strategy method [24], particle swarm optimization [25], differential evolution algorithm [27], and improved bat algorithm [28]). The population size and the maximum number of iterations are set to 30 and 200 during the process, respectively. In Table 9, for F01, the solution accuracy of the IAOA is higher than the other methods, and then, the S-method, FN, ES, DEBA, PSO, and DE obtain close results; for F02, the IAOA achieves the best result, and the FN, ES, DEBA, PSO, and DE are in the second echelon; for F03, the IAOA achieves the better result compared to the FN, ES, and PSO. The MBFES, DEBA, and DE rank third. In Table 10, for F04, the IAOA gets a perfect result, and the FN, ES, DEBA, PSO, and DE obtain similar values; for F05, the IAOA ranks first, and the FN, ES, DEBA, PSO, and DE rank second; for F06, the IAOA, FN, and DE achieve competitive results. For F07–F09, the IAOA obtains the best value, and the FN, ES, and DEBA rank second in Table 11. The traditional methods (R-method, T-method, and S-method) fail to solve F10; therefore, G32 and 2n × L5 are utilized to tackle this problem. In Table 12, the IAOA and DEBA obtain similar values and ranks first. Tables 13 and 14 are statistical results for the numerical integration (F01–F10) are obtained by swarm intelligence algorithms. For F01–F09, the IAOA is better than the other algorithms across all the assessment criteria (the best value, the worst value, mean value, and standard deviation). However, for F10, the IAOA achieves the only optimal result in the best value, and the rest rank second, in which the DEBA obtains the best results. From Figure 5, the method proposed in this paper has the fastest convergence speed and convergence accuracy for all the problems except F10. The above experimental results prove that the IAOA has fast convergence speed, high solution accuracy, and strong robustness. These enable the IAOA to handle numerical integration problems; therefore, it is a worthwhile direction to apply the IAOA to solve the integration solution problems in practical engineering applications.


F01
Error
Iteration
AOA
IAOA
PSO
DE
DEBA
ES

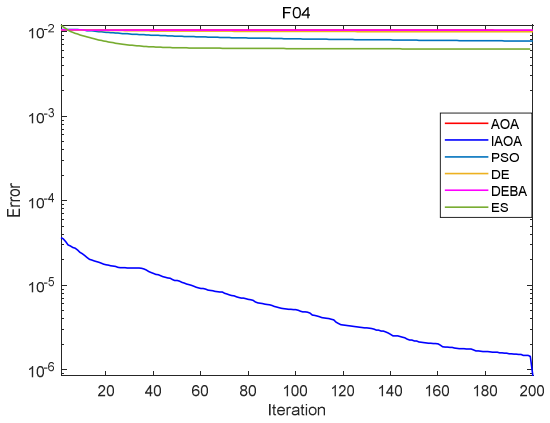
(**a**)


F02
Error
Iteration
AOA
IAOA
PSO
DE
DEBA
ES

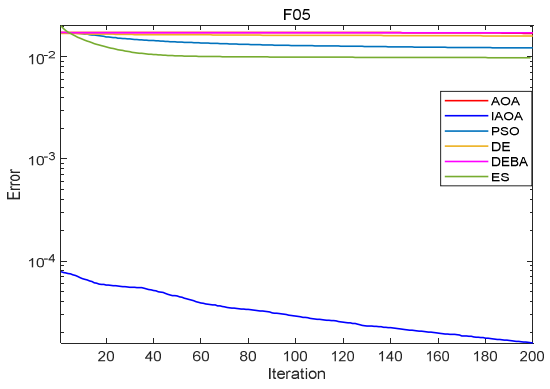
(**b**)


F03
Error
Iteration
AOA
IAOA
PSO
DE
DEBA
ES

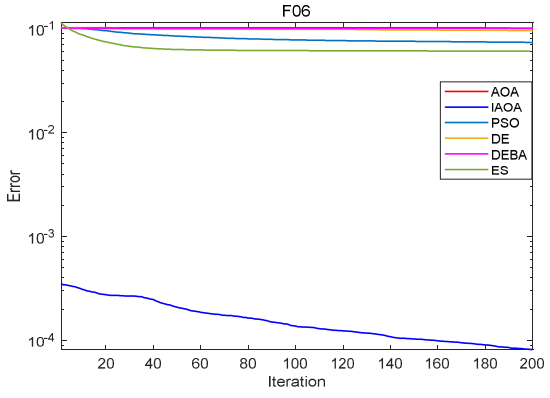
(**c**)


F04
Error
Iteration
AOA
IAOA
PSO
DE
DEBA
ES

(**d**)


F05
Error
Iteration
AOA
IAOA
PSO
DE
DEBA
ES

(**e**)


F06
Error
Iteration
AOA
IAOA
PSO
DE
DEBA
ES

(**f**)

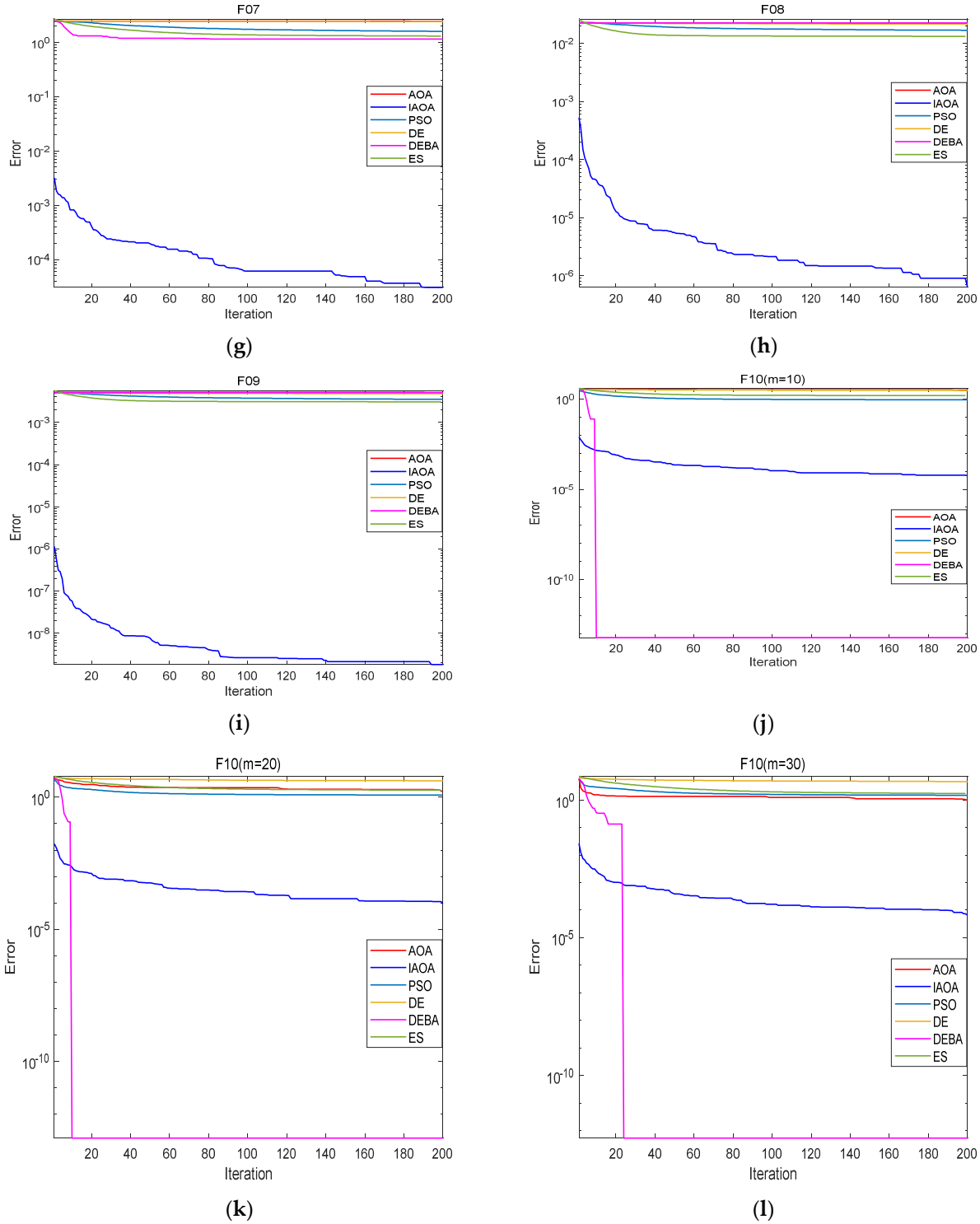


Figure 5. Convergence curve for the numerical integrations (F01–F10 (**a**–**l**)).

Table 8. Details of the integrations F01–F10.

Integrations	Details	Range
F01	$f(x) = x^2$	[0, 2]
F02	$f(x) = x^4$	[0, 2]
F03	$f(x) = \sqrt{1 + x^2}$	[0, 2]

F04	$f(x)=\frac{1}{1+x}$	[0, 2]
F05	$f(x)=\sin x$	[0, 2]
F06	$f(x)=e^{x}$	[0, 2]
F07	$f(x)=\sqrt{1+(\cos x)^{2}}$	[0, 48]
F08	$f(x)=\begin{cases} e^{-x}, 0\le x<1 \\ e^{-x/2}, 1\le x<2 \\ e^{-x/3}, 2\le x\le 3 \end{cases}$	[0, 3]
F09	$f(x)=e^{-x^{2}}$	[0, 1]
F10	$f(x)=x\cos x\sin xmx, (m=10,20,30)$	[0, 2π]

Table 9. Comparison of the experimental results for F01–F03.

Methods	**Integrations**		
	F01	**F02**	**F03**
R-method	2.000	2.000	2.828
T-method	4.000	16.000	3.236
S-method	2.667	6.667	2.964
H-method	2.830	7.066	3.048
FN [26]	2.667	6.3995	2.95789
MBFES [24]	2.659	6.338	2.956
ES [24]	2.666	6.398	2.9577
DEBA [28]	2.66698573	6.401201	2.958169
PSO [25]	2.666	6.398	2.9578
DE [27]	2.667	6.3995	2.958
AOA	2.61006134	6.20147125	2.94004382
IAOA	2.66661710	6.40000000	2.95788286
Exact	2.66666667	6.40000000	2.95788572

Table 10. Comparison of the experimental results for F04–F06.

Methods	**Integrations**		
	F04	**F05**	**F06**
R-method	1.000	1.683	5.437
T-method	1.333	0.909	8.389
S-method	1.111	1.425	6.421
H-method	1.112	1.452	6.691
FN [26]	1.0986	1.416	6.389
MBFES [24]	1.090	1.419	6.390
ES [24]	1.098	1.416	6.388
DEBA [28]	1.098754	1.416082	6.388921
PSO [25]	1.0985	1.416	6.3887
DE [27]	1.099	1.416	6.389
AOA	1.08923818	1.40101546	6.29531692
IAOA	1.09861229	1.41613957	6.38901606
Exact	1.09861229	1.41614684	6.38905610

Table 11. Comparison of the experimental results for F07–F09.

Methods	Integrations		
	F07	**F08**	**F09**
R-method	52.13975183	1.51349542	0.77782078
T-method	62.43737140	1.61179305	0.74621972
S-method	117.61490334	2.48720505	0.74683657
H-method	58.99776108	1.56164258	0.75403569
FN [26]	58.4705	1.54604	0.746823
MBFES [24]	58.48828	1.5455	0.74652
ES [24]	58.47065	1.5459805	0.74683
DEBA [28]	58.470505372351	1.5460388345767	0.7468269544604
PSO	56.80139775	1.52897330	0.74328459
DE	56.04598085	1.52425900	0.74202909
AOA	56.17497970	1.52641514	0.74223182
IAOA	58.47046915	1.54603603	0.74682413
Exact	58.47046915	1.54603603	0.74682413

Table 12. Comparison of the experimental results for F10.

Methods	Integrations		
	F10 (m = 10)	**F10 (m = 20)**	**F10 (m = 30)**
G32	−0.6340207	−1.2092524	−1.5822272
2n × L5	−0.55875940	−0.27789620	−0.18508448
H-method	−0.21043575	0.17309499	−0.02945756
MBFES [24]	−0.68134052	−0.37280425	−0.17305621
ES [24]	−0.65034080	−0.30583435	−0.23556815
DEBA	−0.63466518	−0.31494663	−0.20967248
PSO	−1.50150183	−1.33949737	−1.10170197
DE [27]	−0.63982173	−0.31035906	−0.21438251
AOA	−3.07253909	−0.56489050	−0.42642997
IAOA	−0.63466518	−0.31494663	−0.20967248
Exact	−0.63466518	−0.31494663	−0.20967248

Table 13. Statistical results for the numerical integrations (F01–F06).

Algorithms		Integrations					
		F01	**F02**	**F03**	**F04**	**F05**	**F06**
AOA	best	5.660532×10^{-2}	1.985287×10^{-1}	1.784189×10^{-2}	9.374106×10^{-3}	1.513137×10^{-2}	9.373918×10^{-2}
	worst	6.785842×10^{-2}	2.466178×10^{-1}	2.112411×10^{-2}	1.103594×10^{-2}	1.827849×10^{-2}	1.105054×10^{-1}
	mean	6.196485×10^{-2}	2.238141×10^{-1}	1.970905×10^{-2}	1.041648×10^{-2}	1.679104×10^{-2}	1.013200×10^{-1}
	std	2.473863×10^{-3}	1.277362×10^{-2}	6.790772×10^{-4}	4.381854×10^{-4}	7.886715×10^{-4}	3.985235×10^{-3}
IAOA	best	4.956295×10^{-5}	0.000000	2.855397×10^{-6}	0.000000	7.267277×10^{-6}	4.004088×10^{-5}
	worst	1.070986×10^{-4}	9.632589×10^{-6}	1.471988×10^{-5}	7.241931×10^{-6}	3.035345×10^{-5}	1.136393×10^{-4}
	mean	7.267766×10^{-5}	9.617999×10^{-7}	6.357033×10^{-6}	1.274560×10^{-6}	1.595556×10^{-5}	7.989662×10^{-5}
	std	1.561025×10^{-5}	2.672207×10^{-6}	2.828416×10^{-6}	1.942626×10^{-6}	5.989208×10^{-6}	2.032255×10^{-5}
PSO [25]	best	3.966996×10^{-2}	1.282142×10^{-1}	1.263049×10^{-2}	6.772669×10^{-3}	1.115352×10^{-2}	6.495427×10^{-2}
	worst	5.467546×10^{-2}	1.880821×10^{-1}	1.614274×10^{-2}	9.112184×10^{-3}	1.385859×10^{-2}	9.718717×10^{-2}
	mean	4.406724×10^{-2}	1.593799×10^{-1}	1.405265×10^{-2}	7.745239×10^{-3}	1.208230×10^{-2}	7.327404×10^{-2}
	std	3.262431×10^{-3}	1.528260×10^{-2}	9.707823×10^{-4}	6.532329×10^{-4}	7.146743×10^{-4}	6.698801×10^{-3}
DE [27]	best	5.444535×10^{-2}	1.776272×10^{-1}	1.740389×10^{-2}	9.410606×10^{-3}	1.537737×10^{-2}	9.229490×10^{-2}
	worst	6.223208×10^{-2}	1.992612×10^{-1}	1.943564×10^{-2}	1.043440×10^{-2}	1.668422×10^{-2}	1.003285×10^{-1}
	mean	5.887766×10^{-2}	1.887098×10^{-1}	1.881844×10^{-2}	1.003350×10^{-2}	1.606658×10^{-2}	9.665791×10^{-2}

	std	1.717478×10^{-3}	5.056921×10^{-3}	4.230737×10^{-4}	2.412656×10^{-4}	3.636407×10^{-4}	1.886442×10^{-3}
DEBA [28]	best	5.858312×10^{-2}	1.958779×10^{-1}	1.797733×10^{-2}	9.632554×10^{-3}	1.541447×10^{-2}	9.078063×10^{-2}
	worst	6.805128×10^{-2}	2.566962×10^{-1}	2.194973×10^{-2}	1.144459×10^{-2}	1.824156×10^{-2}	1.096576×10^{-1}
	mean	6.306158×10^{-2}	2.287206×10^{-1}	2.005007×10^{-2}	1.048558×10^{-2}	1.700868×10^{-2}	1.008133×10^{-1}
	std	2.059708×10^{-3}	1.384008×10^{-2}	8.428458×10^{-4}	4.319549×10^{-4}	7.193521×10^{-4}	4.457879×10^{-3}
ES [24]	best	3.634854×10^{-2}	1.053634×10^{-1}	1.178783×10^{-2}	6.152581×10^{-3}	9.742411×10^{-3}	6.028495×10^{-2}
	worst	3.704455×10^{-2}	1.076016×10^{-1}	1.197536×10^{-2}	6.272540×10^{-3}	9.921388×10^{-3}	6.120127×10^{-2}
	mean	3.662145×10^{-2}	1.064150×10^{-1}	1.189432×10^{-2}	6.206519×10^{-3}	9.813727×10^{-3}	6.070549×10^{-2}
	std	1.618502×10^{-4}	4.726931×10^{-4}	4.687831×10^{-5}	2.718416×10^{-5}	4.560503×10^{-5}	2.303572×10^{-4}

Table 14. Statistical results for numerical integrations (F07–F10).

Algorithms		**Integrations**					
		F07	**F08**	**F09**	**F10 (m = 10)**	**F10 (m = 20)**	**F10 (m = 30)**
AOA	best	2.295489	1.962088×10^{-2}	4.592313×10^{-3}	2.437873	2.499438×10^{-1}	2.167574×10^{-1}
	worst	2.524012	2.400262×10^{-2}	5.421672×10^{-3}	3.611012	3.429053	3.115022
	mean	2.424997	2.226327×10^{-2}	5.031127×10^{-3}	3.225836	1.617425	9.721188×10^{-1}
	std	5.634089×10^{-2}	1.017542×10^{-3}	2.167135×10^{-4}	2.620454×10^{-1}	9.081448×10^{-1}	7.417795×10^{-1}
IAOA	best	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	worst	4.285648×10^{-4}	9.665730×10^{-6}	7.650313×10^{-9}	4.941453×10^{-4}	8.932970×10^{-4}	4.121824×10^{-4}
	mean	5.817808×10^{-5}	1.079836×10^{-6}	1.094646×10^{-9}	6.843408×10^{-5}	9.159354×10^{-5}	6.487479×10^{-5}
	std	9.331558×10^{-5}	2.377176×10^{-6}	2.051844×10^{-9}	1.219906×10^{-4}	1.972260×10^{-4}	9.370544×10^{-5}
PSO [25]	best	1.093717	1.499542×10^{-2}	3.212480×10^{-3}	5.688245×10^{-1}	1.024550	8.920294×10^{-1}
	worst	2.077297	2.010782×10^{-2}	4.674802×10^{-3}	1.599995	1.485451	1.953066
	mean	1.669071	1.706272×10^{-2}	3.539538×10^{-3}	8.668366×10^{-1}	1.219538	1.489201
	std	2.419795×10^{-1}	1.205259×10^{-3}	3.409595×10^{-4}	2.759571×10^{-1}	1.216184×10^{-1}	2.065585×10^{-1}
DE [27]	best	2.255785	2.091958×10^{-2}	4.575317×10^{-3}	2.543013	3.461794	3.889322
	worst	2.522405	2.254710×10^{-2}	5.009106×10^{-3}	3.236645	4.684467	5.201887
	mean	2.424488	2.177702×10^{-2}	4.795040×10^{-3}	3.015091	4.242609	4.687029
	std	5.766110×10^{-2}	4.602533×10^{-4}	1.146454×10^{-4}	1.967397×10^{-1}	2.313007×10^{-1}	2.923496×10^{-1}
DEBA [28]	best	2.361570×10^{-1}	2.057410×10^{-2}	4.776881×10^{-3}	6.043389×10^{-14}	1.208677×10^{-13}	5.319404×10^{-13}
	worst	2.468831	2.474051×10^{-2}	5.441200×10^{-3}	6.043389×10^{-14}	1.208677×10^{-13}	5.319404×10^{-13}
	mean	1.163514	2.294436×10^{-2}	5.157892×10^{-3}	6.043389×10^{-14}	1.208677×10^{-13}	5.319404×10^{-13}
	std	6.919695×10^{-1}	9.765442×10^{-4}	1.475304×10^{-4}	3.851264×10^{-29}	7.702528×10^{-29}	3.081011×10^{-28}
ES [24]	best	1.298269	1.319474×10^{-2}	3.051746×10^{-3}	1.460773	1.634373	1.152204
	worst	1.321623	1.341748×10^{-2}	3.121709×10^{-3}	1.665912	2.355153	2.380726
	mean	1.308546	1.331615×10^{-2}	3.081151×10^{-3}	1.568781	1.869004	1.719830
	std	5.523404×10^{-3}	5.640941×10^{-5}	1.521690×10^{-5}	4.627499×10^{-2}	1.831224×10^{-1}	2.898513×10^{-1}

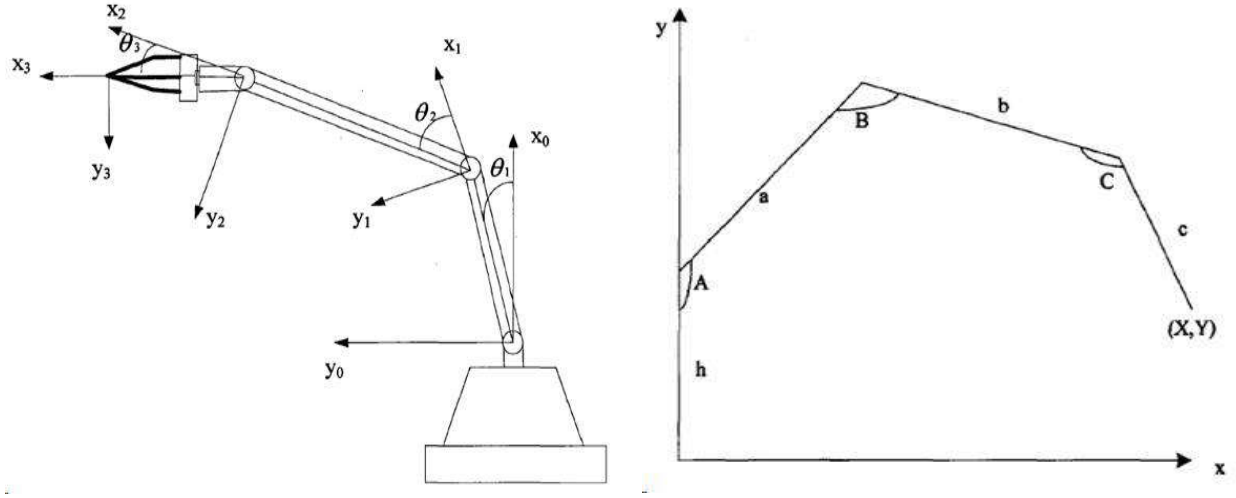
4.4. Sovling Engineering Problem

Compared with three-dimensional motion, planar motion restricts the robot to a single plane and is simpler to calculate. However, most robot mechanisms can simplify plane mechanisms or planes for tackling. Now, the robotic arm plays an increasingly important role, which has also attracted the extensive attention of researchers. Improving the working efficiency of the robotic arm under the premise of low energy consumption is a challenging problem facing the industrial field [57]. The kinematics of the robotic arm mainly include forward kinematics and inverse kinematics. One is the pose of the end effector determined according to the rotation angle of each joint based on the base coordinates; the other is taking the end joint as the starting point and, finally, back-to-base coordinates. The inverse kinematics problem is essentially a nonlinear equation problem. The tasks performed by the robotic arm are usually described by its base coordinate system in practical applications. Therefore, the inverse kinematics solution is particularly important in

the field of the control. The robotic arm model [58] is shown in Figure 6a, and the mathematical model in coordinates is shown in Figure 6b. The nonlinear equation system for this model is as follows.

$$\begin{cases} 10,000 \times ((a \times \sin(A_2) - b \times \sin(A_2 + B_2) + c \times \sin(A_2 + B_2 + C_2) - X)^2) = 0 \\ 10,000 \times ((h - a \times \cos(A_2) - b \times \cos(A_2 + B_2) + c \times \cos(A_2 + B_2 + C_2) - Y)^2) = 0 \\ |A_2 - A_1| + |B_2 - B_1| + |C_2 - C_1| = 0 \end{cases} \quad (25)$$

where a = 16.5 cm; b = 7.9 cm; c = 5.3 cm; and h = 7.4 cm (A_1 = 150°, B_1 = 132.7026°, and C_1 = 127.0177°) are the initial angles of the three joints; (X = 10 cm, Y = 10 cm) is the coordinate of the end effector; and (A_2, B_2, and C_2) are the aims required to obtain three joint angles in the final stage. The first two equations in the nonlinear equation system find the three joint angles when the end effector reaches the target position (X, Y), and the third equation ensures that the change of the joint angle is the smallest to meet the requirements for saving energy.



(**a**) (**b**)

Figure 6. (**a**) The model of a robotic arm, and (**b**) a mathematical model for a robotic arm.

Tables 15–18 demonstrate that the IAOA obtains the closest results to the initial angle compared with the PSO, GA and PSSA in solving the inverse kinematics problem of the robotic arm. This shows that the method proposed in this paper allows the robotic arm to consume less energy during movement. In Table 19, f represents the fitness value obtain by Equation (25) and is the difference between the final angle and initial angle of the joint. Obviously, the IAOA achieves the best results for both evaluations. Therefore, it is a great significance to the stability, operation efficiency, operation accuracy, and energy consumption of the robotic arm trajectory control. A new method is provided for the inverse motion solution, which makes up for the deficiency of the traditional method.

Table 15. The results obtained by the IAOA for the engineering problem.

Algorithm		**Joint Angles**		
		A_2	B_2	C_2
IAOA	initial angle	150	132.7026	127.0177
	Result	145.7291	139.0180	123.9864

Table 16. The results obtained by the PSO for the engineering problem.

Algorithm		Joint Angles		
		A_2	B_2	C_2
PSO	initial angle	150	132.7026	127.0177
	result	139.6534	68.2235	96.4886

Table 17. The results obtained by the GA for the engineering problem.

Algorithm		Joint Angles		
		A_2	B_2	C_2
GA	initial angle	150	132.7026	127.0177
	result	129.8653	118.9625	52.6691

Table 18. The results obtained by the PSSA for the engineering problem.

Algorithm		Joint Angles		
		A_2	B_2	C_2
PSSA [58]	initial angle	150	132.7026	127.0177
	result	147.1015	92.5371	89.5116

Table 19. Comparison of the experimental results for the IAOA, PSO, GA, and PSSA.

Objective Funtions	Algorithms			
	IAOA	PSO	GA	PSSA
f	1.3618 × 10	3.0608×10^6	3.2329×10^6	2.0199×10^5
$\lvert A_2 - A_1\rvert + \lvert B_2 - B_1\rvert + \lvert C_2 - C_1\rvert$	13.6176	105.3548	118.2234	80.5701

5. Conclusions and Future Works

In this paper, the shortcomings are analyzed of the traditional AOA so that an improved AOA based on a population control strategy is proposed to overcome the weakness. The algorithm can find the best global value faster by classifying the population and adaptively controlling the number of individuals in each subpopulation. This method effectively enhances the information sharing strength between individuals, can better search the space, avoids falling into the local optimum, accelerates the convergence process, and improves the optimization accuracy. The AOA, IAOA, and some other algorithms are compared based on solving 6 nonlinear systems of equations, 10 numerical integrations, and an engineering problem. The experimental results show that the IAOA can solve these problems well and outperform the other algorithms. In the future, the IAOA can be used to solve more nonlinear problems in practical engineering applications. Secondly, it can try to solve multiple integrals. Finally, the algorithm can be further improved and enhanced in its performance.

Author Contributions: Conceptualization and methodology, M.C. and Y.Z.; software, M.C.; writing—original draft preparation, M.C.; writing—review and editing, Y.Z. and Q.L.; and funding acquisition, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, Grant No. U21A20464 and 62066005.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Broyden, C.G. A class of methods for solving nonlinear simultaneous equations. *Math. Comput.* **1965**, *19*, 577–593.
2. Ramos, H.; Monteiro, M.T.T. A new approach based on the newton's method to solve systems of nonlinear equations. *J. Comput. Appl. Math.* **2017**, *318*, 3–13.
3. Hueso, J.L.; Martínez, E.; Torregrosa, J.R. Modified newton's method for systems of nonlinear equations with singular Jacobian. *J. Comput. Appl. Math.* **2009**, *224*, 77–83.
4. Luo, Y.Z.; Tang, G.J.; Zhou, L.N. Hybrid approach for solving systems of nonlinear equations using chaos optimization and quasi-newton method. *Appl. Soft Comput.* **2008**, *8*, 1068–1073.
5. Karr, C.L.; Weck, B.; Freeman, L.M. Solutions to systems of nonlinear equations via a genetic algorithm. *Eng. Appl. Artif. Intell.* **1998**, *11*, 369–375.
6. Ouyang, A.J.; Zhou, Y.Q.; Luo, Q.F. Hybrid particle swarm optimization algorithm for solving systems of nonlinear equations. In Proceedings of the 2009 IEEE International Conference on Granular Computing, Nanchang, China, 17–19 August 2009; pp. 460–465.
7. Jaberipour, M.; Khorram, E.; Karimi, B. Particle swarm algorithm for solving systems of nonlinear equations. *Comput. Math. Appl.* **2011**, *62*, 566–576.
8. Pourjafari, E.; Mojallali, H. Solving nonlinear equations systems with a new approach based on invasive weed optimization algorithm and clustering. *Swarm Evol. Comput.* **2012**, *4*, 33–43.
9. Jia, R.M.; He, D.X. Hybrid artificial bee colony algorithm for solving nonlinear system of equations. In Proceedings of the 2012 Eighth International Conference on Computational Intelligence and Security, Guangzhou, China, 17–18 November 2012; pp. 56–60.
10. Ren, H.M.; Wu, L.; Bi, W.H.; Argyros, I.K. Solving nonlinear equations system via an efficient genetic algorithm with symmetric and harmonious individuals. *Appl. Math. Comput.* **2013**, *219*, 10967–10973.
11. Cai, R.Z.; Yue, G.L. A novel firefly algorithm of solving nonlinear equation group. *Appl. Mech. Mater.* **2013**, *389*, 918–923.
12. Abdollahi, M.; Isazadeh, A.; Abdollahi, D. Imperialist competitive algorithm for solving systems of nonlinear equations. *Comput. Math. Appl.* **2013**, *65*, 1894–1908.
13. Hirsch, M.J.; Pardalos, P.M.; Resende, M.G.C. Solving systems of nonlinear equations with continuous GRASP. *Nonlinear Anal. Real World Appl.* **2009**, *10*, 2000–2006.
14. Sacco, W.F.; Henderson, N. Finding all solutions of nonlinear systems using a hybrid metaheuristic with fuzzy clustering means. *Appl. Soft Comput.* **2011**, *11*, 5424–5432.
15. Gong, W.Y.; Wang, Y.; Cai, Z.H.; Yang, S. A weighted bi-objective transformation technique for locating multiple optimal solutions of nonlinear equation systems. *IEEE Trans. Evol. Comput.* **2017**, *21*, 697–713.
16. Ariyaratne, M.K.A.; Fernando, T.G.I.; Weerakoon, S. Solving systems of nonlinear equations using a modified firefly algorithm (MODFA). *Swarm Evol. Comput.* **2019**, *48*, 72–92.
17. Gong, W.Y.; Wang, Y.; Cai, Z.H.; Wang, L. Finding multiple roots of nonlinear equation systems via a repulsion-based adaptive differential evolution. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 1499–1513.
18. Ibrahim, A.M.; Tawhid, M.A. A hybridization of differential evolution and monarch butterfly optimization for solving systems of nonlinear equations. *J. Comput. Des. Eng.* **2019**, *6*, 354–367.
19. Liao, Z.W.; Gong, W.Y.; Wang, L. Memetic niching-based evolutionary algorithms for solving nonlinear equation system. *Expert Syst. Appl.* **2020**, *149*, 113–261.
20. Ning, G.Y.; Zhou, Y.Q. Application of improved differential evolution algorithm in solving equations. *Int. J. Comput. Intell. Syst.* **2021**, *14*, 199.
21. Rizk-Allah, R.M. A quantum-based sine cosine algorithm for solving general systems of nonlinear equations. *Artif. Intell. Rev.* **2021**, *54*, 3939–3990.
22. Ji, J.Y.; Man, L.W. An improved dynamic multi-objective optimization approach for nonlinear equation systems. *Inf. Sci.* **2021**, *576*, 204–227.
23. Turgut, O.E.; Turgut, M.S.; Coban, M.T. Chaotic quantum behaved particle swarm optimization algorithm for solving nonlinear system of equations. *Comput. Math. Appl.* **2014**, *68*, 508–530.
24. Zhou, Y.Q.; Zhang, M.; Zhao, B. Numerical integration of arbitrary functions based on evolutionary strategy method. *Chin. J. Comput.* **2008**, *21*, 196–206.
25. Wei, X.Q.; Zhou, Y.Q. Research on numerical integration method based on particle swarm optimization. *Microelectron. Comput.* **2009**, *26*, 117–119.
26. Wei, X.X.; Zhou, Y.Q.; Lan, X.L. Research on a numerical integration method based on functional networks. *Comput. Sci.* **2009**, *36*, 224–226.
27. Deng, Z.X.; Huang, F.D.; Liu, X.J. A differential evolution algorithm for solving numerical integration problems. *Comput. Eng.* **2011**, *37*, 206–207.
28. Xiao, H.H.; Duan, Y.M. Application of improved bat algorithm in numerical integration. *J. Intell. Syst.* **2014**, *9*, 364–371.
29. Szczepanski, R.; Kaminski, M.; Tarczewski, T. Auto-tuning process of state feedback speed controller applied for two-mass system. *Energies* **2020**, *13*, 3067.
30. Hu, H.B.; Hu, Q.B.; Lu, Z.Y.; Xu, D. Optimal PID controller design in PMSM servo system via particle swarm optimization. In Proceedings of the 31st Annual Conference of IEEE Industrial Electronics Society, IECON 2005, Raleigh, NC, USA, 6–10 November 2005; p. 5.

31. Szczepanski, R.; Tarczewski, T.; Niewiara, L.J.; Stojic, D. Isdentification of mechanical parameters in servo-drive system. In Proceedings of the 2021 IEEE 19th International Power Electronics and Motion Control Conference (PEMC), Gliwice, Poland, 25–29 April 2021; pp. 566–573.
32. Liu, L.; Cartes, D.A.; Liu, W. Particle Swarm Optimization Based Parameter Identification Applied to PMSM. In Proceedings of the 2007 American Control Conference, New York, NY, USA, 9–13 July 2007; pp. 2955–2960.
33. Szczepanski, R.; Tarczewski, T. Global path planning for mobile robot based on artificial bee colony and Dijkstra's algorithms. In Proceedings of the 2021 IEEE 19th International Power Electronics and Motion Control Conference (PEMC), Gliwice, Poland, 25–29 April 2021; pp. 724–730.
34. Brand, M.; Masuda, M.; Wehner, N.; Yu, X.H. Ant colony optimization algorithm for robot path planning. In Proceedings of the 2010 International Conference on Computer Design and Applications, Qinhuangdao, China, 25–27 June 2010; pp. 436–440.
35. Szczepanski, R.; Erwinski, K.; Tejer, M.; Bereit, A.; Tarczewski, T. Optimal scheduling for palletizing task using robotic arm and artificial bee colony algorithm. *Eng. Appl. Artif. Intell.* **2022**, *113*, 104976.
36. Kolakowska, E.; Smith, S.F.; Kristiansen, M. Constraint optimization model of a scheduling problem for a robotic arm in automatic systems. *Robot. Auton. Syst.* **2014**, *62*, 267–280.
37. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The arithmetic optimization algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609.
38. Premkumar, M.; Jangir, P.; Kumar, D.S.; Sowmya, R.; Alhelou, H.H.; Abualigah, L.; Yildiz, A.R.; Mirjalili, S. A new arithmetic optimization algorithm for solving real-world multi-objective CEC-2021 constrained optimization problems: Diversity analysis and validations. *IEEE Access* **2021**, *9*, 84263–84295.
39. Bansal, P.; Gehlot, K.; Singhal, A.; Gupta, A. Automatic detection of osteosarcoma based on integrated features and feature selection using binary arithmetic optimization algorithm. *Multimed. Tools Appl.* **2022**, *81*, 8807–8834.
40. Agushaka, J.O.; Ezugwu, A.E. Advanced arithmetic optimization algorithm for solving mechanical engineering design problems. *PLoS ONE* **2021**, *16*, e0255703.
41. Abualigah, L.; Diabat, A.; Sumari, P.; Gandomi, A. A novel evolutionary arithmetic optimization algorithm for multilevel thresholding segmentation of COVID-19 CT images. *Processes* **2021**, *9*, 1155.
42. Xu, Y.P.; Tan, J.W.; Zhu, D.J.; Ouyang, P.; Taheri, B. Model identification of the proton exchange membrane fuel cells by extreme learning machine and a developed version of arithmetic optimization algorithm. *Energy Rep.* **2021**, *7*, 2332–2342.
43. Izci, D.; Ekinci, S.; Kayri, M.; Eker, E. A novel improved arithmetic optimization algorithm for optimal design of PID controlled and Bode's ideal transfer function-based automobile cruise control system. *Evol. Syst.* **2021**, *13*, 453–468.
44. Khatir, S.; Tiachacht, S.; Thanh, C.L.; Ghandourah, E.; Mirjalili, S.; Wahab, M.A. An improved artificial neural network using arithmetic optimization algorithm for damage assessment in FGM composite plates. *Compos. Struct.* **2021**, *273*, 114–287.
45. Viswanathan, G.M.; Afanasyev, V.; Buldyrev, S.; Murphy, E.J.; Prince, P.A.; Stanley, H.E. Lévy flight search patterns of wandering albatrosses. *Nature* **1996**, *381*, 413–415.
46. Humphries, N.; Queiroz, N.; Dyer, J. et al. Environmental context explains Lévy and Brownian movement patterns of marine predators. *Nature* **2010**, *465*, 1066–1069.
47. Mirjalili, S. A sine cosine Algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133.
48. Mirjalili, S.; Lewis, A. The whale optimization algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67.
49. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey wolf optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61.
50. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris hawks optimization: Algorithm and applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872.
51. Li, S.M.; Chen, H.L.; Wang, M.J.; Heidari, A.A.; Mirjalili, S. Slime mould algorithm: A new method for stochastic optimization. *Future Gener. Comput. Syst.* **2020**, *111*, 300–323.
52. Price, K.V. Differential evolution: A fast and simple numerical optimizer. In Proceeding of the North American Fuzzy Information Processing, Berkeley, CA, USA, 19–22 June 1996; pp. 524–527.
53. Gandomi, A.H.; Yang, X.S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35.
54. Grosan, C.; Abraham, A. A new approach for solving nonlinear equations systems. *IEEE Trans. Syst. Man Cybern. Part A Syst. Hum.* **2008**, *38*, 698–714.
55. Floudas, C.A. Recent advances in global optimization for process synthesis, design and control: Enclosure of all solutions. *Comput. Chem. Eng.* **1999**, *23*, S963–S973.
56. Nikkhah-Bahrami, M.; Oftadeh, R. An effective iterative method for computing real and complex roots of systems of nonlinear equations. *Appl. Math. Comput.* **2009**, *215*, 1813–1820.
57. Ding, X. *Robot Control Research*; Zhejiang University Press: Hangzhou, China, 2006; pp. 37–38.
58. Xiang, Z.H.; Zhou, Y.Q.; Luo, Q.F.; Wen, C. PSSA: Polar coordinate salp swarm algorithm for curve design problems. *Neural Process Lett.* **2020**, *52*, 615–645.