electronics

MDPI

Article

Research on Active Disturbance Rejection Control with Parameter Tuning for Permanent Magnet Synchronous Motor Based on Improved PSO Algorithm

Ziyang Zhou, Liming Wang, Yang Wang *, Xinlei Zhou and Yipin Tong

School of Electrical Engineering, Naval University of Engineering, Wuhan 430030, China; m22385403@nue.edu.cn (Z.Z.)
* Correspondence: wangyang8828@stu.ouc.edu.cn

Abstract: Addressing the issue of significant speed fluctuations in permanent magnet synchronous motors (PMSM) under load, this paper proposes an active disturbance rejection control strategy based on an improved particle swarm optimization (PSO) algorithm. Initially, the speed of the PMSM is considered as the comprehensive optimization objective, and an active disturbance rejection control (ADRC) model for the PMSM is established by integrating the ADRC with vector control. Subsequently, an adaptive PSO algorithm incorporating genetic algorithms is proposed. This algorithm uses chaotic initialization for uniform particle distribution, introduces adaptive inertia weight and dynamic cognitive factors to enhance search efficiency, and integrates the crossover and mutation modules from genetic algorithms, optimizing mutations using a Gaussian probability function. Simulation results demonstrated that: (1) under identical external load conditions, the proposed ADRC strategy ensured smaller speed fluctuations and a smoother waveform for the PMSM, and (2) compared to the traditional PSO algorithm, the proposed method reduced the speed fluctuation after external load by approximately 26%.

Keywords: active disturbance rejection control; parameter tuning; particle swarm optimization; permanent magnet synchronous motors; cross; mutation; adaptive weights

check for updates

Citation: Zhou, Z.; Wang, L.; Wang, Y.; Zhou, X.; Tong, Y. Research on Active Disturbance Rejection Control with Parameter Tuning for Permanent Magnet Synchronous Motor Based on Improved PSO Algorithm. *Electronics* **2024**, *13*, 3436. https://doi.org/10.3390/electronics13173436

Academic Editor: Ahmed Abu-Siada

Received: 8 July 2024
Revised: 22 August 2024
Accepted: 27 August 2024
Published: 29 August 2024

1. Introduction

Permanent magnet synchronous motors can deliver higher power output for the same weight and volume, making them widely used across various fields. With the increasing application of PMSMs in high-performance AC servo drive systems, such as aerospace, new energy technologies, and electric vehicles, there are heightened control performance requirements for these motors. In recent years, high-performance systems have demanded increasingly stringent speed performance from PMSMs [1], specifically requiring the mitigation of speed fluctuations caused by load torque variations. The authors of [2] demonstrated that in practical systems, the speed performance of PMSMs is affected by disturbances, such as sudden changes in load torque and magnetic flux harmonics. To meet the needs of high-performance systems, it is essential to investigate strategies for suppressing speed loop disturbances in PMSM drive systems under complex conditions to reduce speed fluctuations and achieve smooth, stable speed control.

To address the challenges of motor operation in complex environments, researchers have turned their attention to sensorless control technologies. Currently, sensorless control techniques primarily include the back-EMF method, flux estimation method, and current sampling method [3]. However, these methods are not particularly effective in dealing with external disturbances. The authors of [4] indicated that in PMSM control systems, sudden load torque application is often considered one of the most severe non-periodic disturbances in the speed loop, causing significant speed drops. Among high-performance control methods, traditional PI control and various new intelligent control methods in PMSM vector control systems still have shortcomings [5,6]: The PI control algorithm is simple in structure and easy to implement, but due to the nonlinear characteristics of PMSM, its performance is affected by parameter variations, thus failing to provide ideal disturbance rejection. In contrast, ADRC has strong disturbance rejection capabilities and is a better control method for PMSMs. ADRC uses an extended state observer (ESO) to observe the system's real-time state and internal and external disturbances, compensating for them accordingly. It provides feedback based on the observer's values, offering excellent robustness. The limitations of traditional linear ADRC have been analyzed in the literature [7,8], which proposes an improved ADRC method based on model predictive compensation. Another study [9] introduces an adaptive ESO based on an ADRC, where the gains of the ESO are adjusted in real time to minimize the total disturbance of state estimation errors and measurement noise, and this method has been applied to air–fuel ratio control in gasoline engines. Further research [10] employed interpolation fitting to reconstruct the extended state observer and nonlinear feedback to optimize ADRC. Additionally, in [11], a third-order nonlinear extended state observer was proposed for position and velocity estimation of sensorless internal permanent magnet synchronous motor drives with further improved immunity and accuracy. An ESO with predefined decreasing gain was designed [12] to reduce the impact of measurement noise in steady-state conditions, and experiments were conducted on a magnetic levitation ball system. Another study [13] suggested an offline Q-learning-based adaptive adjustment algorithm for ADRC parameters, applying it to ship heading control. Furthermore, the authors of [14] presented a linear/nonlinear switching extended state observer (L/NLSESO) to address the issue of large disturbance estimation limitations in NLESO, indirectly improving the performance of the corresponding ADRC algorithm. In [15], a new load-adaptive dual-loop drive system based on an improved position–velocity integral self-immunity controller and a parameter fuzzy self-tuning method was proposed, which enhanced the system's accuracy and ability to adapt to load variations.

The aforementioned literature on the improvement of ADRC primarily focuses on optimizing the ESO model. While these optimization methods have enhanced the control performance of ADRC to some extent, there has been limited research on the parameter settings for the ESO and nonlinear feedback, often relying on empirical methods. In ADRC, the mathematical models for differential trackers, state observers, and nonlinear feedback contain multiple unknown parameters that need to be defined. Relying solely on experience to set these parameters makes it challenging to obtain the optimal solution. Therefore, solving for the optimal parameter configuration is imperative.

To address the problem of optimal parameter configuration, many scholars have utilized heuristic optimization algorithms [16–18] to tune ADRC parameters. In recent years, the application of PSO algorithms in the intelligent tuning of ADRC parameters has garnered increasing attention from researchers in related fields. A study [19] employed the PSO algorithm to refine the initial ADRC parameter configuration for achieving precise motion control in antenna servo systems. Simulation results indicated that the enhanced ADRC exhibited advantages, such as small overshoot, fast response speed, strong anti-interference capability, high reliability, and robust performance. However, this algorithm tended to fall into local optima. The authors of [20] proposed an adaptive PSO algorithm, which determines the current evolutionary state based on the evolution factor calculated by the particle swarm and switches between multiple velocity update functions accordingly, thus improving the convergence speed. The authors of [21] used a genetic algorithm to adjust motor ADRC parameters based on a multi-objective optimization function, ultimately determining the ADRC control parameters after multiple iterations. Additionally, an improved PSO algorithm was proposed in [22], which introduced a constraint factor to control the inertia coefficient of particle velocity, addressing the issue of slow search speed in the PSO algorithm, thereby enhancing the search speed and convergence accuracy. Nevertheless, this algorithm still encounters the problem of local optima.

Existing literature on the improvement of PSO algorithms primarily focuses on the inertia weight function and acceleration constants. While these improvements enhance the algorithm's optimization accuracy and convergence speed, two issues remain unaddressed. First, the use of random initialization methods results in an overly dispersed initial search range for the particles, with multiple particles distributed near the interval extrema, thus affecting convergence precision. Second, during the particle swarm update process, the differences in fitness among particles are not considered, leading to a tendency to fall into local optima during the solution process. To address these issues, this paper proposes a globally optimized IPSO algorithm. This algorithm uses logistic chaotic initialization to determine the initial positions of the particles. It adaptively adjusts the inertia weight and velocity factors based on fitness and the number of iterations. Additionally, it integrates genetic algorithms, selecting particles for crossover and mutation based on their fitness to optimize particle adaptability.

The contributions of this work are as follows:

1. Logistic chaotic mapping initialization. Replacing random initialization with logistic chaotic mapping optimizes the distribution of population particles within the value range, ensuring a more uniform distribution and improving the convergence speed.
2. Linearly decreasing inertia weight function. Introducing a linearly decreasing inertia weight function to adaptively adjust the inertia weight improves the acceleration constants by incorporating a dynamic learning function, which adjusts the acceleration constants corresponding to the number of population iterations, thus enhancing adaptability.
3. Genetic algorithm integration. Combining genetic algorithms to select particles for crossover and mutation based on fitness and introducing a Gaussian probability function to help escape local optima, thereby improving optimization accuracy.

The remainder of this article is organized as follows:

Section 2 outlines a PMSM ADRC system, including the mathematical model of the PMSM and the ADRC, and describes the principle of the components of the ADRC. Section 3 presents the basic PSO algorithm and describes in detail the various improved parts of the IPSO algorithm. Section 4 describes the operation of the IPSO algorithm in ADRC and verifies the superiority of the IPSO algorithm by comparing it with other algorithms. Section 5 compares the simulation of the ADRC system of the PMSM and verifies the superiority and effectiveness of the IPSO algorithm. Section 6 summarizes the contents of this paper.

2. Mathematical Model of ADRC and PMSM

In this paper, the output speed of the PMSM was selected as the optimization target. The speed loop uses an ADRC instead of a PI controller to achieve the desired output curve according to actual needs. This approach aims to ensure a smooth output while reducing overshoot, improving response speed, and enhancing disturbance rejection capability.

2.1. SPMSM Mathematical Model

This paper focused on the surface-mounted permanent magnet synchronous motor (SPMSM), ignoring the nonlinear characteristics of the motor, airgap permeance, internal permeance of the permanent magnets, and rotor winding damping. Additionally, the three-phase stator windings of this motor are symmetrically distributed in space, and the airgap magnetic field is sinusoidally distributed. The mathematical model of the SPMSM can be obtained based on motor control theory. To design a better controller model, the Park transformation is typically used to convert the PMSM mathematical model from the α-β coordinate system to the d-q coordinate system. The mathematical model of PMSM in the two-phase synchronous rotating d-q coordinate system is as follows:

$$\begin{cases} u_d = Ri_d + L_d\frac{di_d}{dt} - \omega_e L_q i_q \\ u_q = Ri_q + L_q\frac{di_q}{dt} + \omega_e L_d i_d + \omega_e \psi_f \end{cases} \tag{1}$$

Magnetic flux equation:

$$\begin{cases} \psi_d = L_d i_d + \psi_f \\ \psi_q = L_q i_q \end{cases} \tag{2}$$

Electromagnetic torque equation:

$$T_e = \frac{3}{2} p_n \left[\psi_f i_q + \left(L_d - L_q \right) i_d i_q \right] \tag{3}$$

Mechanical motion equation:

$$J \frac{d\omega_m}{dt} = T_e - T_L - B\omega_m \tag{4}$$

Here, u_d and u_q are the stator voltage components in the d-axis and q-axis, respectively; ψ_d and ψ_q are the stator flux components in the d-axis and q-axis, respectively; L_d and L_q are the inductance components in the d-axis and q-axis, respectively; ω_e is the electrical angular speed; ψ_f is the permanent magnet flux linkage; J is the moment of inertia; B is the damping coefficient; T_L is the load torque, and ω_m is the mechanical angular speed of the motor.

2.2. ADRC Mathematical Model

ADRC is a classical model-free control method with the capability to control nonlinear systems. The ADRC consists of three main components: the tracking differentiator (TD), the nonlinear state error feedback (NLSEF), and the ESO. The TD tracks the desired setpoint signal used for arranging the transition process, reducing overshoot during system regulation. The ESO, the core component of the ADRC controller, is primarily used for observing the system's state feedback and compensating for disturbances through the extended state. The NLSEF generates control outputs to compensate for the total disturbances in the system. This structure is illustrated in Figure 1.

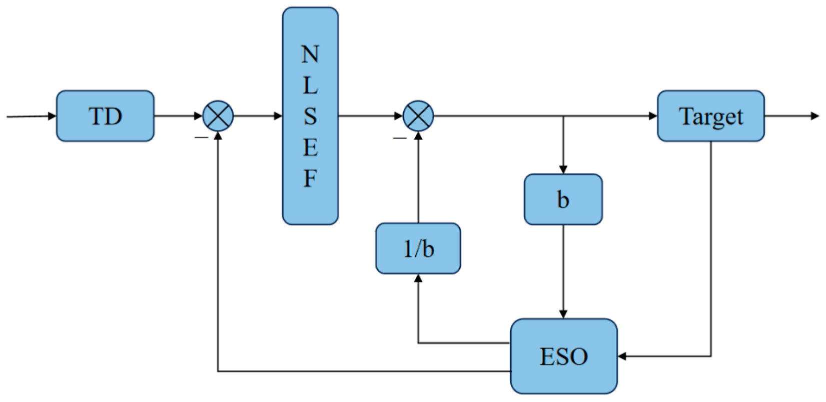

Figure 1. ADRC structure.

In this paper, we constructed an ADRC mathematical model for speed control of permanent magnet synchronous motors, with the following Equation (5) motor speed equation derived from Equation (4):

$$\frac{dw_m}{dt} = \frac{1}{J} T_e - \frac{1}{J} T_L - \frac{B}{J} w_m \tag{5}$$

For SPMSM, the stator inductance meets the following Equation (6):

$$L_d = L_q \tag{6}$$

The motor speed equation can be obtained from Equations (5) and (6), as shown in Equation (7):

$$\frac{dw_m}{dt} = \frac{3P_n\psi_f}{2J}i_q - \frac{1}{J}T_L - \frac{B}{J}w_m \tag{7}$$

The TD mathematical model is shown in Equation (8):

$$\begin{cases} e_0 = x_1 - v_0 \\ \dot{x}_1 = x_2 \\ \dot{x}_1 = -r \cdot \text{fal}(e_0, \alpha_0, \delta) \end{cases} \tag{8}$$

The nonlinear smooth function, fal, is defined in Equation (9):

$$\text{fal}(e, \alpha, \delta) = \begin{cases} |e|^{\alpha}\text{sign}(e), |e| > \delta \\ \frac{e}{\delta^{1-\alpha}}, |e| \le \delta \end{cases} \tag{9}$$

After discretization by the Euler method, the following Equation (10) is obtained:

$$\begin{cases} x_1(k+1) = x_1(k) + hx_2(k) \\ e(k) = x_1(k) - v_0 \end{cases} \tag{10}$$

In the above equation, r is a variable that controls the speed of the tracking signal and is proportional to the tracking speed, x_1 is the speed signal after arranging the transition process, v_0 is the input speed signal, δ is the filtering factor, and α_0 is the nonlinear factor.

The NLSEF mathematical model is shown in Equation (11):

$$\begin{cases} e_1 = v_1(k) - z_1(k) \\ u_0 = \beta_1 \cdot \text{fal}(e_1, \alpha_1, \delta) \\ u = u_0 - \frac{z_2(k)}{b} \end{cases} \tag{11}$$

In the Equation (11), e_1 is the error between the speed signal, $v_1(k)$, after the transition process and the observed value, $z_1(k)$, of the actual output, y, while $-\frac{z_2(k)}{b}$ is the component that compensates for disturbances.

According to Equation (7), the system equation is built, as shown in Equation (12):

$$\begin{cases} \dot{x}_1 = f(x_1, t) + bu \\ y = x_1 \end{cases} \tag{12}$$

where $f(x_1, t) = -\frac{1}{J}T_L - \frac{B}{J}w_m$ is an unknown disturbance, which is suppressed by a non-smooth feedback effect, $b = \frac{3P_n\psi_f}{2J}$, $u = i_q$. Let $x_2 = f(x_1, t)$, $\dot{x}_2 = g(x_1, t)$; then, there is the following Equation (13):

$$\begin{cases} \dot{x}_1 = x_2 + bu \\ \dot{x}_2 = g(x_1, t) \\ y = x_1 \end{cases} \tag{13}$$

Build the mathematical model of the ESO according to Equation (13), as shown in Equation (14):

$$\begin{cases} e_2 = z_1 - y \\ \dot{z}_1 = z_2 - \beta_2\text{fal}(e_2, \alpha_2, \delta) + bu \\ \dot{z}_2 = -\beta_3\text{fal}(e_2, \alpha_3, \delta) \end{cases} \tag{14}$$

After discretization by the Euler method, the following Equation (15) is obtained:

$$\begin{cases} e_2 = z_1(k) - y(k) \\ z_1(k+1) = z_1(k) + h[z_2(k) - h\beta_2\text{fal}(e_2, \alpha_2, \delta) + bu(k)] \\ z_2(k+1) = z_2(k) - h\beta_3\text{fal}(e_2, \alpha_3, \delta) \end{cases} \tag{15}$$

The PMSM active disturbance rejection control model built based on the above mathematical model is shown in Figure 2.

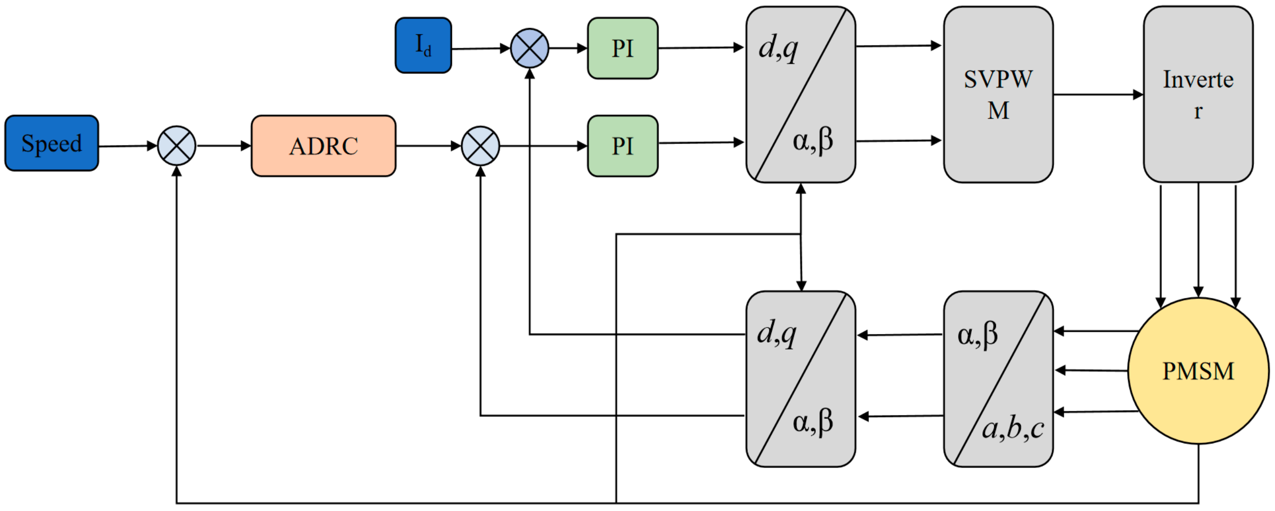

Figure 2. PMSM active disturbance rejection control model.

3. Improved PSO Algorithms

The PSO algorithm [23] is an intelligent optimization algorithm that simulates the behavior of animal groups [24,25]. It is widely used to solve optimization problems in various scenarios due to its wide search range and fast iteration speed [26–28]. However, it suffers from issues, such as premature convergence and susceptibility to local optima. Therefore, in this paper, we propose improvements to the PSO algorithm by using adaptive inertia weights, dynamic learning speed factors, and integrating genetic algorithms

3.1. Basic PSO Algorithms

Let the search space be d-dimensional, with n particles. The position of the i-th particle is x_i, and the velocity change rate of the i-th particle is v_i. The best position found by the entire particle swarm so far is p_g. For each iteration, the expression of the i-th particle's movement in the d-th dimension is shown in Equation (16):

$$v_i = \omega v_i + c_1 r_1 \left(x_{pbest} - x_i\right) + c_2 r_2 \left(x_{gbest} - x_i\right) \quad (16)$$

The particle position x_i is shown in Equation (17):

$$x_i = x_{i-1} + v_i \quad (17)$$

Here, c_1 and c_2 are cognitive factors, with c_1 being the individual cognitive factor for each particle and c_2 being the social cognitive factor for each particle. r_1 and r_2 are random numbers between 0 and 1. ω is known as the inertia weight; when the inertia weight is large, the global search capability is strong, but the local search capability is weak. Conversely, when the inertia weight is small, the global search capability is weak, but the local search capability is strong. Algorithm 1 shows the pseudocode for the basic PSO algorithm.

Algorithm 1 Particle swarm optimization algorithm.

```
Input: Population size, N, dimension, d, cognition constants, c1 and c2, inertia weight, w.
Output: The best solution, gbest(t)
1: Initialization
2: while Terminate condition has not been met do
3:     for j ← 1 to N do
4:         Compute v(j) according to Equation (16)
5:         Compute x(j) according to Equation (17)
6:         Evaluate particle fitness, f(x(j))
7:         Update the best personal solution, p(j)
8:         Update the best global solution, g(j)
9:     end for
10: end while
11: Select the best solution, gbest(t)
```

3.2. IPSO Algorithms

3.2.1. Particle Initialization Method Based on Logistic Chaos Initialization

The random initialization of the particle swarm space has high randomness and uneven distribution, leading to issues, such as a lack of population diversity and low search efficiency. To address these problems, we used the logistic mapping function for particle swarm initialization to improve the algorithm's performance. Logistic chaotic initialization has nonlinear and periodic characteristics, enabling it to generate more complex and random sequences. This helps increase population diversity and prevents the swarm from falling into local optima.

The logistic mapping function is shown in Equation (18) below:

$$Z_{n+1} = k \cdot Z_n \cdot (1 - Z_n) \tag{18}$$

In Equation (18), n is the current number of iterations, Z_n is the value after the n-th iteration, and k is the mapping parameter. K affects the evolution process of logistic mapping, and the larger the mapping parameter, k, the more uniform the mapping distribution.

The logistic chaos initialization flowchart is shown in Figure 3.

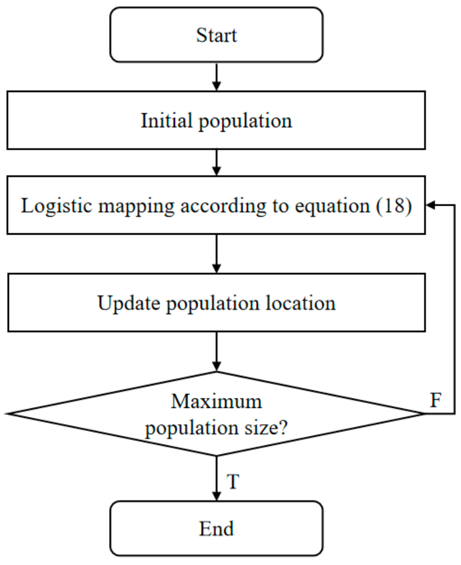
Start
Initial population
Logistic mapping according to equation (18)
Update population location
Maximum population size?
F
T
End

Figure 3. Logistic chaos initialization.

We used chaotic initialization k = 4, with a population size of 300. The random initialization and logistic chaotic initialization particle distribution within a given interval are shown in Figure 4.

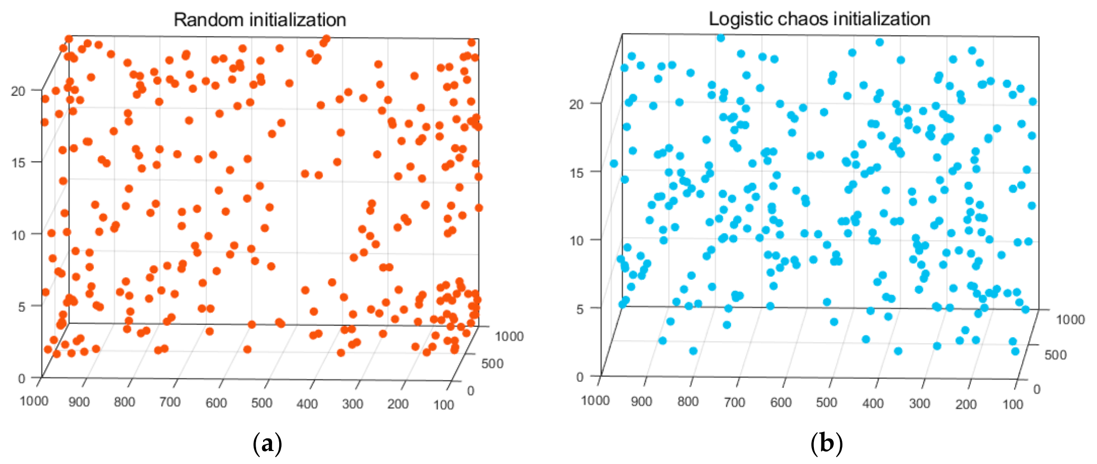

Figure 4. Particle distribution: (**a**) random initialization and (**b**) logistic chaotic initialization.

As shown in Figure 4a, the population particles under random initialization were mostly distributed at the interval edges, with uneven distribution and particle clustering in certain regions. In contrast, Figure 4b demonstrates that the chaotic initialization resulted in a more uniform distribution of population particles within the value space, enhancing population diversity.

3.2.2. Adaptive Weights and Dynamic Cognitive Factors

Inertia weight and cognitive factors are crucial components of the PSO algorithm. The inertia weight significantly affects the performance of the PSO algorithm. During the initial iterations, particles have higher velocities, which enhances global search capabilities but weakens local search abilities. As the number of iterations increases, the inertia weight decreases, resulting in reduced particle velocity and improved local search capabilities. The size of the inertia weight influences the search range and convergence speed of the algorithm. Therefore, we introduced an adaptive inertia weight function that adjusts the inertia weight of the particle swarm based on the iterative fitness.

When $f_i < f_a$, the inertia weight is shown in Equation (19) below:

$$w_i = w_l - \frac{(w_h - w_l) \cdot (f_i - f_{min})}{f_a - f_{min}} \tag{19}$$

When $f_i \geq f_a$, the inertia weight is shown in Equation (20) below:

$$w_i = w_h \tag{20}$$

In Equation (19), f_i is the fitness of the i-th particle, f_a is the average fitness, w_i is the inertia weight of the i-th particle, w_h is the maximum inertia weight set to 0.9, and w_l is the minimum inertia weight set to 0.4.

The cognitive factors, c_1 and c_2, represent the individual and social cognitive factors, respectively. When c_1 is large and c_2 is small, the algorithm exhibits better global search capabilities. Conversely, when c_1 is small and c_2 is large, the algorithm demonstrates better local optimization abilities. Therefore, we introduced dynamic cognitive factors, adjusting the cognitive factors of the particle swarm based on the number of iterations.

The dynamic cognitive factors are shown in Equation (21):

$$\begin{cases} c_1 = 2\sin^2\left[\frac{\pi}{2}\left(1 - \frac{g_i}{g_{max}}\right)\right] \\ c_2 = 2\sin^2\left(\frac{\pi g_i}{g_{max}}\right) \end{cases} \tag{21}$$

In Equation (21), g_i represents the i-th iteration and g_{max} represents the total number of iterations.

3.2.3. Particle Update Method Based on Genetic Algorithm Cross-Mutation

The PSO algorithm is prone to falling into local optima during iterations, resulting in the failure to obtain a global optimal solution. To address this issue, we integrated the genetic algorithm (GA) into PSO to enhance its performance. The genetic algorithm [29,30] is a classical global optimization algorithm inspired by the theory of natural evolution, which finds optimal solutions by mimicking the processes of natural selection and reproduction. In this study, crossover and mutation were introduced during the population iteration process. The randomness of crossover and mutation in the genetic algorithm helps resolve the problem of local optima. Additionally, a Gaussian probability function was used to optimize the mutation effect, thereby improving the accuracy of the algorithm.

The obtained crossover positions are shown in Equation (22):

$$cx_j = r \cdot x(j_1) + (1 - r) \cdot x(j_2) \tag{22}$$

In Equation (22), cx_j represents the position of the j-th child, and $x(j_1)$ and $x(j_2)$ represent the particle positions of the hybrid parent.

The Gaussian probability function is shown in Equation (23):

$$f(x_j) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x_j - \mu)^2}{2\sigma^2}\right) \tag{23}$$

In Equation (23), x_j is the j-th particle, μ is the global best, and σ is the Gaussian mutation parameter.

The position of the mutated particles is shown in Equation (24):

$$mx_j = x_j + f(x_j) .* \left(x_j - x_{gbest}\right) .* \text{ rand} \tag{24}$$

In Equation (24), mx_j represents the position of the mutated particle.

4. Implementation of IPSO Algorithm in ADRC

The specific steps for solving the ADRC adaptive dynamic optimization model using the IPSO algorithm are as follows:

1. Initialize the IPSO algorithm and determine the inertia weight, dynamic learning factor function, and population interval. Perform logistic chaotic initialization for the three-dimensional population particles within the given parameter range. Calculate each particle's velocity, current position, and fitness, then determine the initial optimal position and optimal fitness of the population.
2. Calculate the population's average fitness, f_a, and minimum fitness, f_{min}, and assess whether the particle's fitness is less than the average fitness. Compute the corresponding inertia weight for the particle, and update its velocity and position based on the calculated fitness.
3. Select particles for crossover operations, evaluate both parent and offspring particles, update their respective positions, calculate the particle fitness, and update the individual and population historical optimal fitness.
4. Perform particle mutation using a Gaussian probability function, replace the parent particles with mutated particles, calculate their fitness, and update the individual and population historical optimal fitness.
5. Check if the maximum number of iterations has been reached. If reached, terminate the iteration; otherwise, return to step 2.
6. Output the optimal fitness value and position of the particles and obtain the parameters required for ADRC.

The process flowchart is shown in Figure 5.

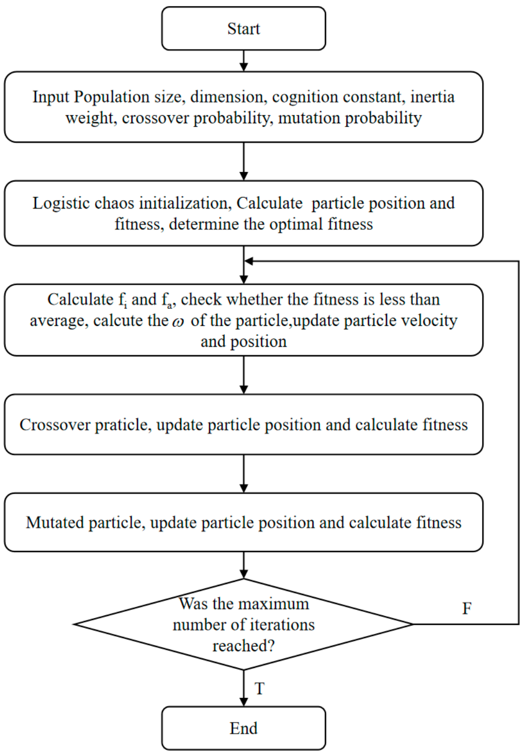
Start
Input Population size, dimension, cognition constant, inertia weight, crossover probability, mutation probability
Logistic chaos initialization, Calculate particle position and fitness, determine the optimal fitness
Calculate f_i and f_a, check whether the fitness is less than average, calcute the ω of the particle,update particle velocity and position
Crossover praticle, update particle position and calculate fitness
Mutated particle, update particle position and calculate fitness
Was the maximum number of iterations reached?
F
T
End

Figure 5. IPSO algorithm operation process.

Algorithm 2 displays the pseudocode of the IPSO algorithm.

Algorithm 2 IPSO algorithm.

```
Input: Population size, N, dimension, d, cognition constants, c1 and c2, inertia weight, w,
crossover probability, pc, mutation probability, pm.
Output: The best solution, gbest(t)
1: Logistic chaos initialization according to Equation (18)
2: while Terminate condition has not been met do
3:     Compute c1(j) and c2(j) according to Equation (21)
4:     for i ← 1 to N do
5:         if (f(x(i))<fa) then
6:             Compute w(i) according to Equation (19)
7:         end if
8:         Compute v(i) according to Equation (16)
9:         Compute x(i) according to Equation (17)
10:        Evaluate particle fitness, f(x(i))
11:        Update the best personal solution, p(i)
12:        Update the best global solution, g(i)
13:    end for
14:    for j ← 1 to cross-pool do
15:        Crossover (seed1, seed2) according to (22)
16:        Evaluate particle fitness, f(x(j))
17:        Update the best global solution, g(j)
18:    end for
19:    for k ← 1 to mutation pool do
20:        Mutation (seed3) according to (24)
21:        Evaluate particle fitness, f(x(k))
22:        Update the best global solution, g(k)
23:    end for
24: end while
25: Select the best solution, gbest(t)
```

We applied the IPSO algorithm to the parameter optimization of the ADRC in the PMSM control system. Specifically, considering the stability and accuracy of the system, IPSO optimizes the adjustable key parameters β_1, β_2, and β_3. The fitness function determines the trend of particle position changes. Therefore, an appropriate fitness function is needed to connect IPSO and ADRC to obtain suitable ADRC parameters. The performance indicators for PMSM ADRC include both stability and dynamic performance metrics. When the system input is a step response, the integral of time-weighted absolute error (ITAE) is used to measure system performance. We used ITAE to evaluate the observation error of the total disturbance in the system by the ESO. The ITAE expression is shown in Equation (25):

$$\mathrm{ITAE} = \int_0^T t|e_i(t)|dt \tag{25}$$

In Equation (25), T is the system response period, $e_i(t)$ is the total disturbance observation error of the system, and $e_i(t)$ is expressed as follows:

$$e_i(t) = v_1(k) - z_1(k) \tag{26}$$

In Equation (26), $e_i(t)$ is the error between the speed signal, $v_1(k)$, after the transition process and the observed value, $z_1(k)$, of the actual output, y.

The effectiveness of the proposed improved PSO algorithm, traditional PSO algorithm, chaotic PSO algorithm, and adaptive weight PSO algorithm was verified through the fitness values and speed waveforms of different algorithms. Figure 6 shows the fitness value iteration curves for the different algorithms, and Table 1 presents the ITAE values of the fitness functions for each algorithm.

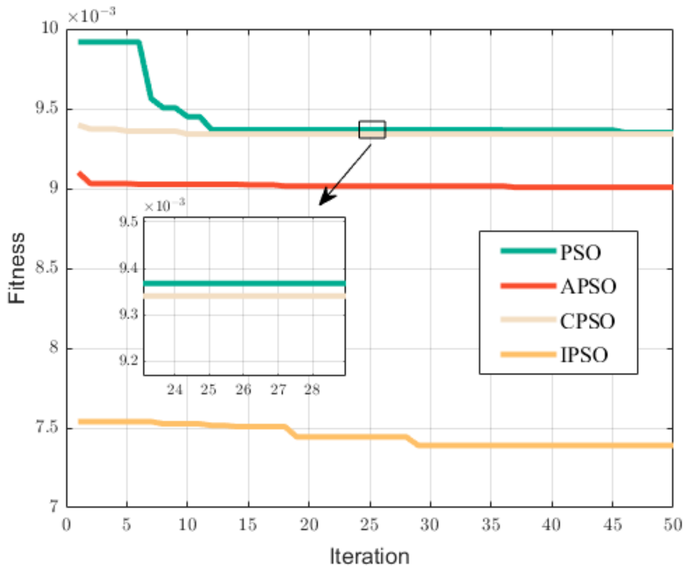

Figure 6. Different algorithms' fitness iteration curves.

As shown in Figure 6, compared to the PSO algorithm, the IPSO algorithm described in this paper achieved the greatest optimization in fitness value. Both the APSO and CPSO algorithms also showed slight improvements in the fitness value.

Table 1. Fitness function value.

	PSO	APSO	CPSO	IPSO
ITAE (10^{-3})	9.4	9.0	9.3	7.4

5. Experimental Results and Analysis of PMSM Control System

5.1. Simulation Model Construction

The ADRC model for the PMSM based on the improved PSO algorithm used in this paper is shown in Figure 7. The motor used was a SPMSM. The speed loop used an ADRC to control the motor speed, and the ADRC parameters were tuned using the IPSO algorithm to obtain the optimal parameter ratios and the PMSM output waveform. The current loop used PI control, and under the ideal condition of $i_d = 0$, the SVPWM module controlled i_q to achieve the desired speed for PMSM operation.

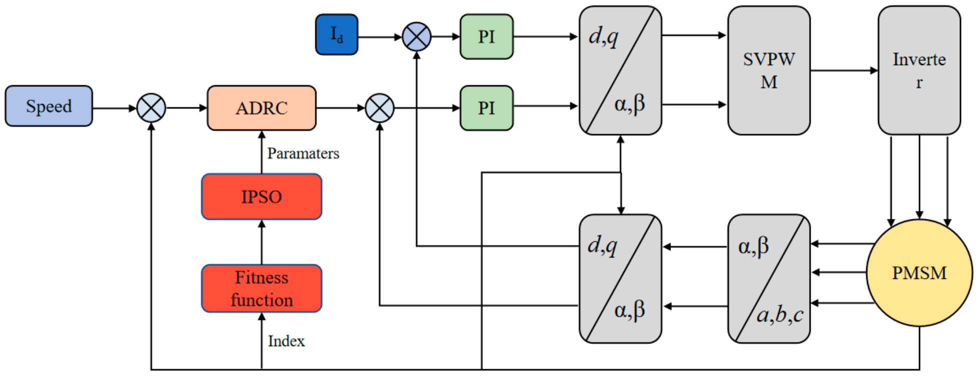

Figure 7. The ADRC model for the PMSM based on the improved PSO algorithm.

The PMSM control system flowchart is shown in Figure 8 below.

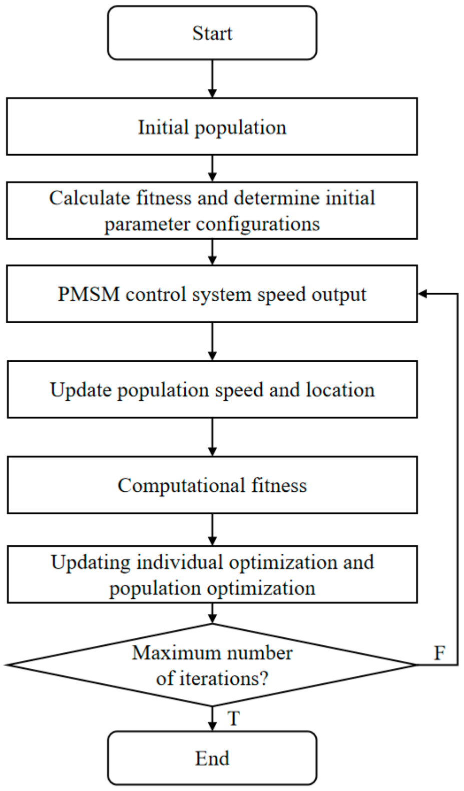
Start
Initial population
Calculate fitness and determine initial parameter configurations
PMSM control system speed output
Update population speed and location
Computational fitness
Updating individual optimization and population optimization
Maximum number of iterations?
F
T
End

Figure 8. PMSM control system.

The relevant parameters of PMSM are shown in Table 2 below.

Table 2. Parameter table of PMSM.

Motor Parameters	Parameter Value	Unit
Flux linkage	0.175	Wb
Moment of inertia	0.003	kg·m^2
Damping coefficien	0.008	N·m·s
Stator resistance	2.875	Ω
Stator inductance	8.500	mH
Number of pole pairs	4	

5.2. Experimental Results and Analysis

In the simulation system, an expected speed signal was applied to the PMSM. The motor speed output waveform was observed to measure the time required to reach the target speed from zero, the overshoot, and the oscillation amplitude. Once the motor reached a steady state, a load of 10 N · m was applied at 0.1 s. The speed change of the PMSM under different control strategies was recorded after the external load was applied. The total simulation duration was 0.2 s. Figure 9 shows the output speed waveforms of the PMSM using the ADRC strategy under no-load conditions, and Figures 10–12 show the output speed waveform when 8 N · m, 10 N · m, and 12 N · m loads were applied at 0.1 s using the ADRC strategy. In Figure 9, the horizontal axis is time, and the vertical axis is speed.

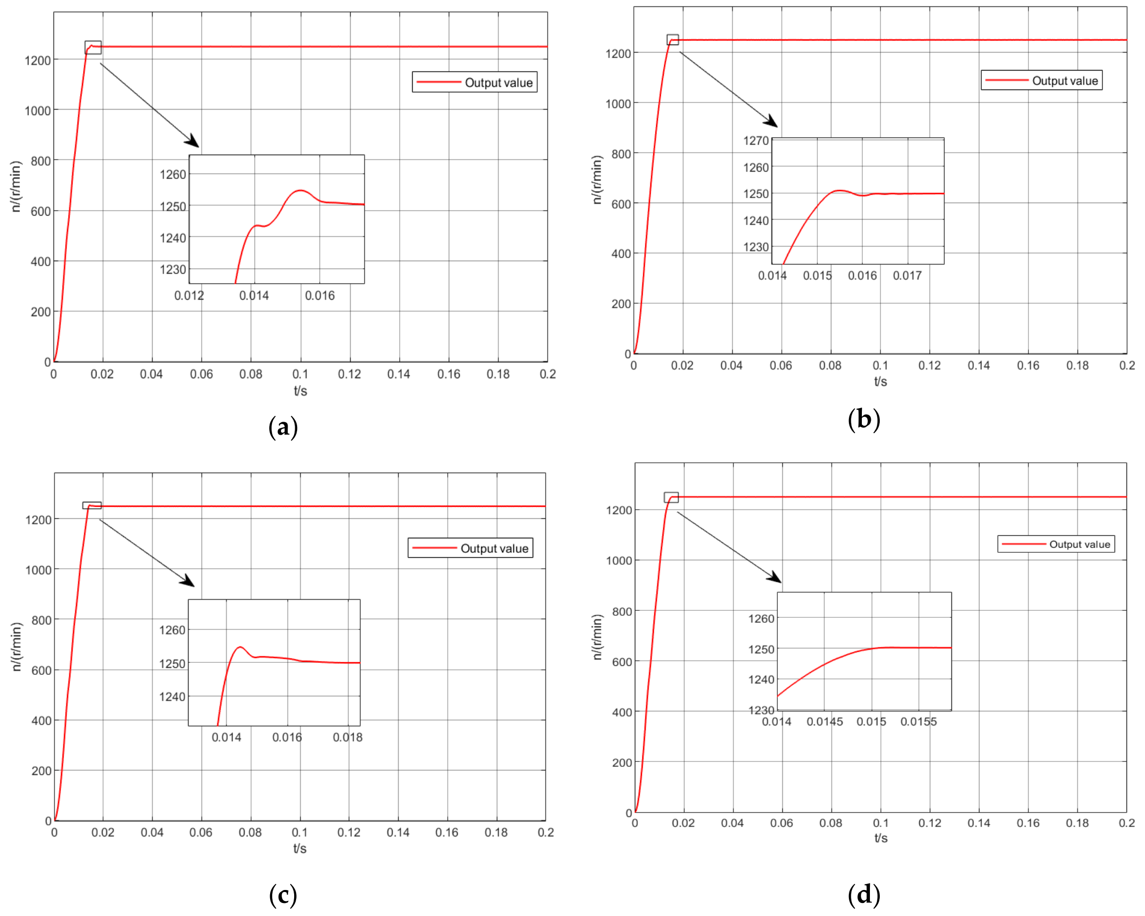

Figure 9. Waveforms of no-load speed. (**a**) PSO, (**b**) APSO, (**c**) CPSO, and (**d**) IPSO.

Figure 10 shows the output speed waveforms when a 8 N · m load was applied at 0.1 s using the ADRC strategy.

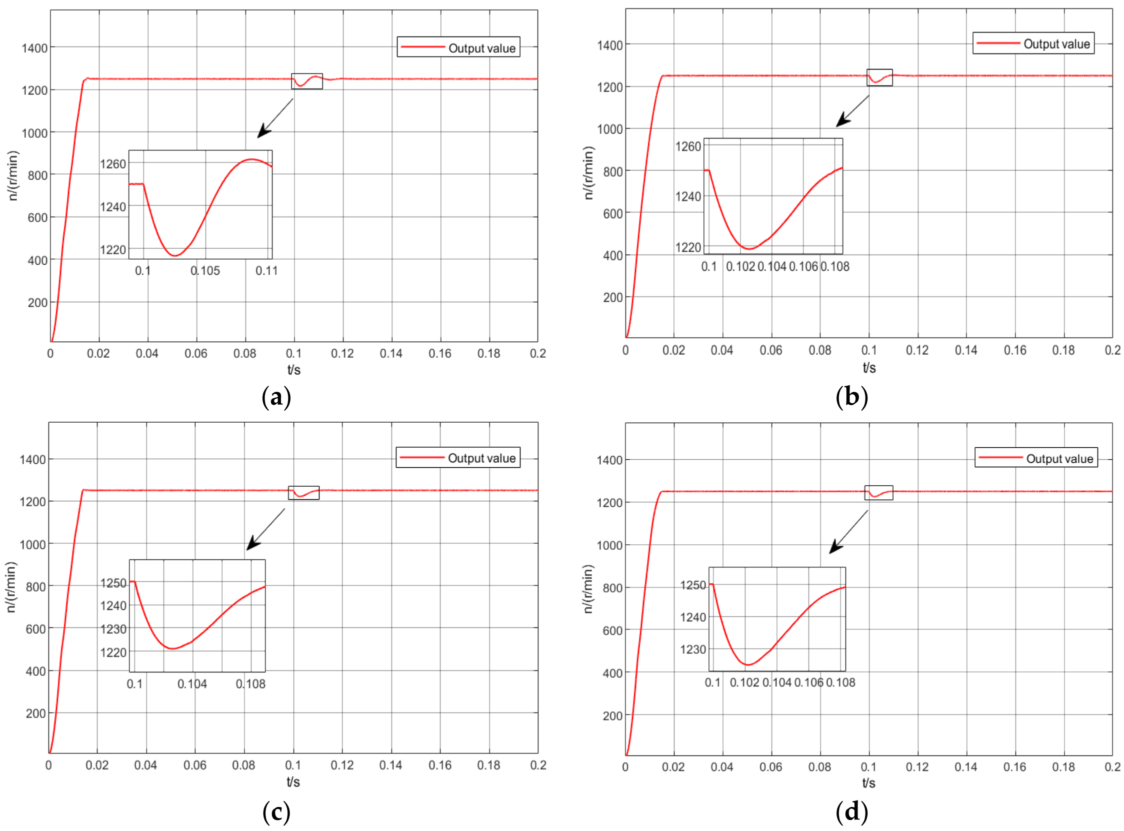

Figure 10. The 8 N · m load speed waveforms. (**a**) PSO, (**b**) APSO, (**c**) CPSO, and (**d**) IPSO.

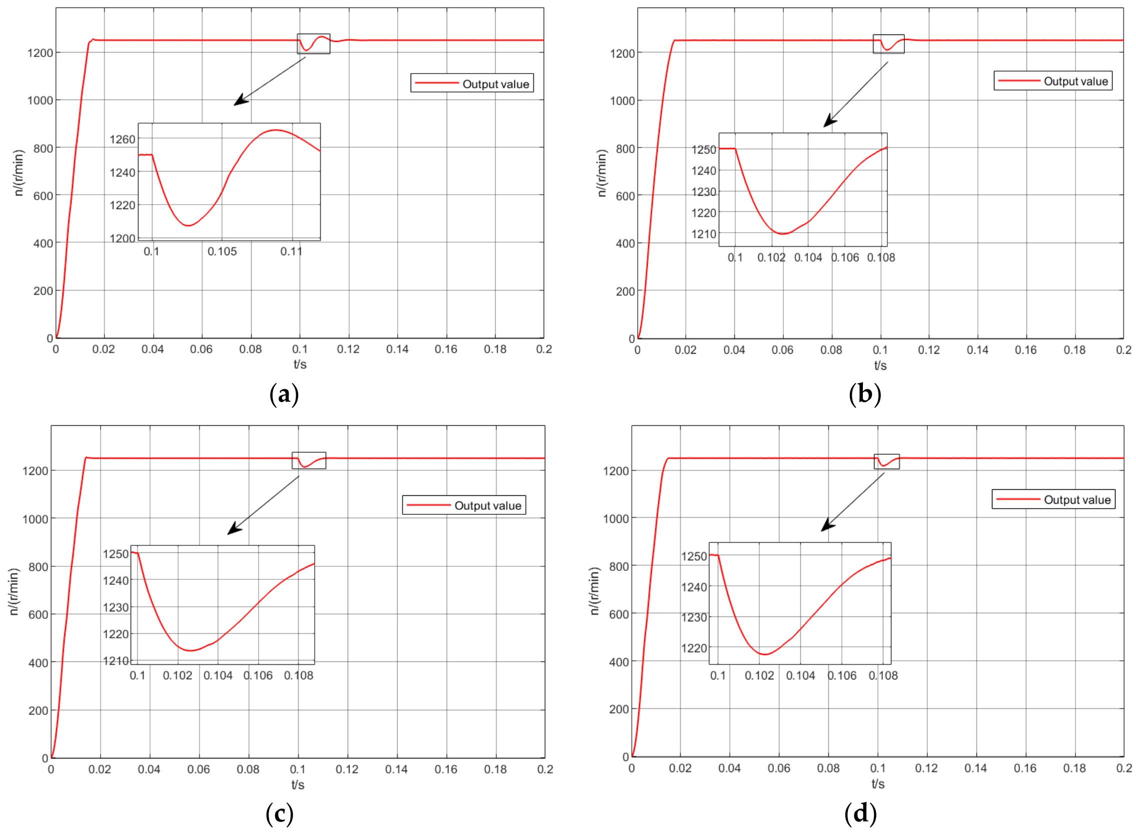

Figure 11. The 10 N · m load speed waveforms. (**a**) PSO, (**b**) APSO, (**c**) CPSO, and (**d**) IPSO.

Figure 11 shows the output speed waveforms when a 10 N · m load was applied at 0.1 s using the ADRC strategy.

Figure 12 shows the output speed waveforms when a 12 N · m load was applied at 0.1 s using the ADRC strategy.

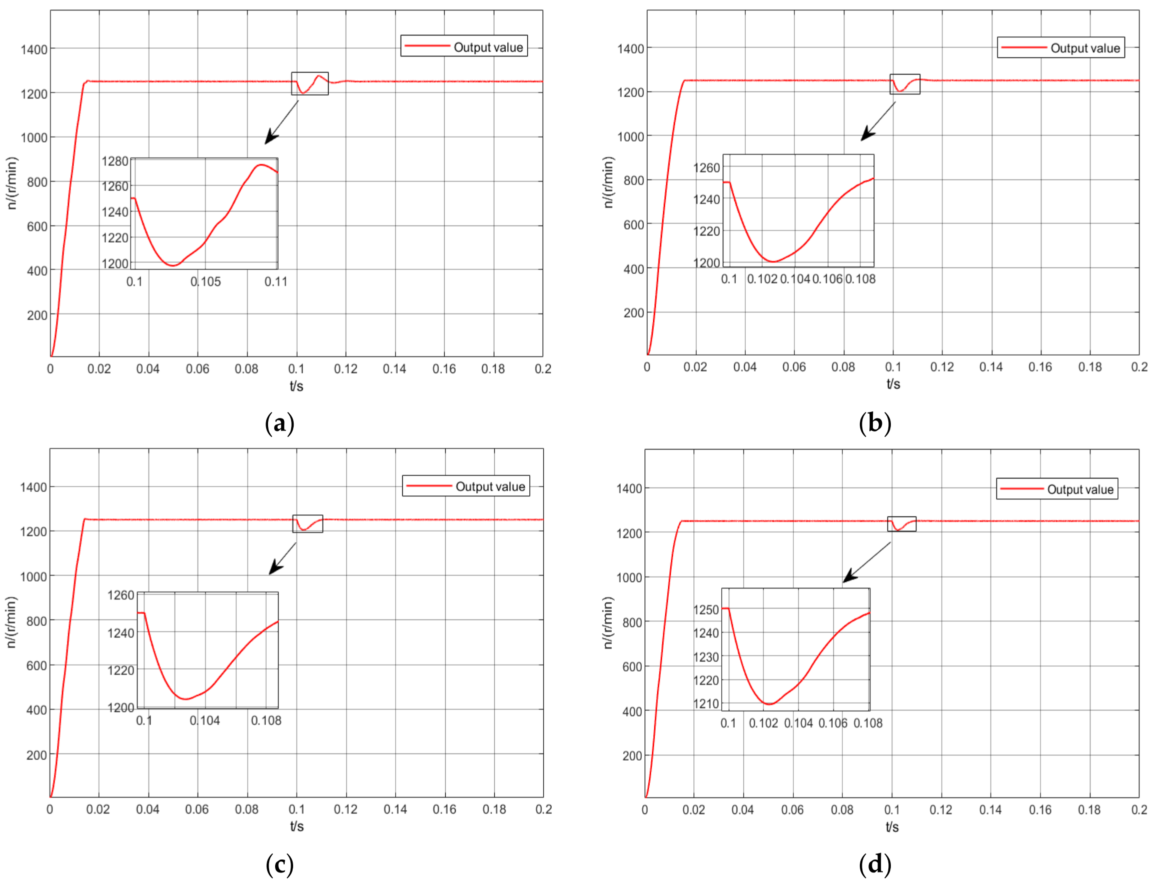

Figure 12. The 12 N · m load speed waveforms. (**a**) PSO, (**b**) APSO, (**c**) CPSO, and (**d**) IPSO.

The initial input signal was 1250 r/min. As shown in Figure 9, the response times of the four different algorithms were roughly the same. However, in terms of overshoot and oscillation amplitude, the IPSO algorithm proposed in this paper performed better than the other three algorithms. After applying external loads when the system was stable, as seen in Figures 10–12, in the case of the same-load PSO algorithm, after the rectification of the ADRC controller in the face of speed fluctuations of the longest recovery time, and when speed overshooting occurred, the APSO and CPSO algorithms' recovery time was similar to the phenomenon of when overshooting did not occur significantly. The IPSO algorithm was calibrated to provide a ADRC controller with better robustness and dynamic recovery, minimal speed fluctuations after sudden load application, and providing better immunity to interference, effectively solving the issue of excessive speed fluctuation due to the external load. This allows the controlled motor's output speed to more accurately match the desired value. Comparative analysis showed that the IPSO algorithm proposed in this paper had better accuracy and robustness.

Under the same conditions, the input was changed to a step input to test the control effect. Initially, the transmission value remained at 1250 r/min, and the step signal changed to 800 r/min at 0.2 s. At the system steady-state time of 0.3 s, a 10 N · m load was applied. The control effect is shown in Figure 13, comparing the PMSM speed output under different control strategies.

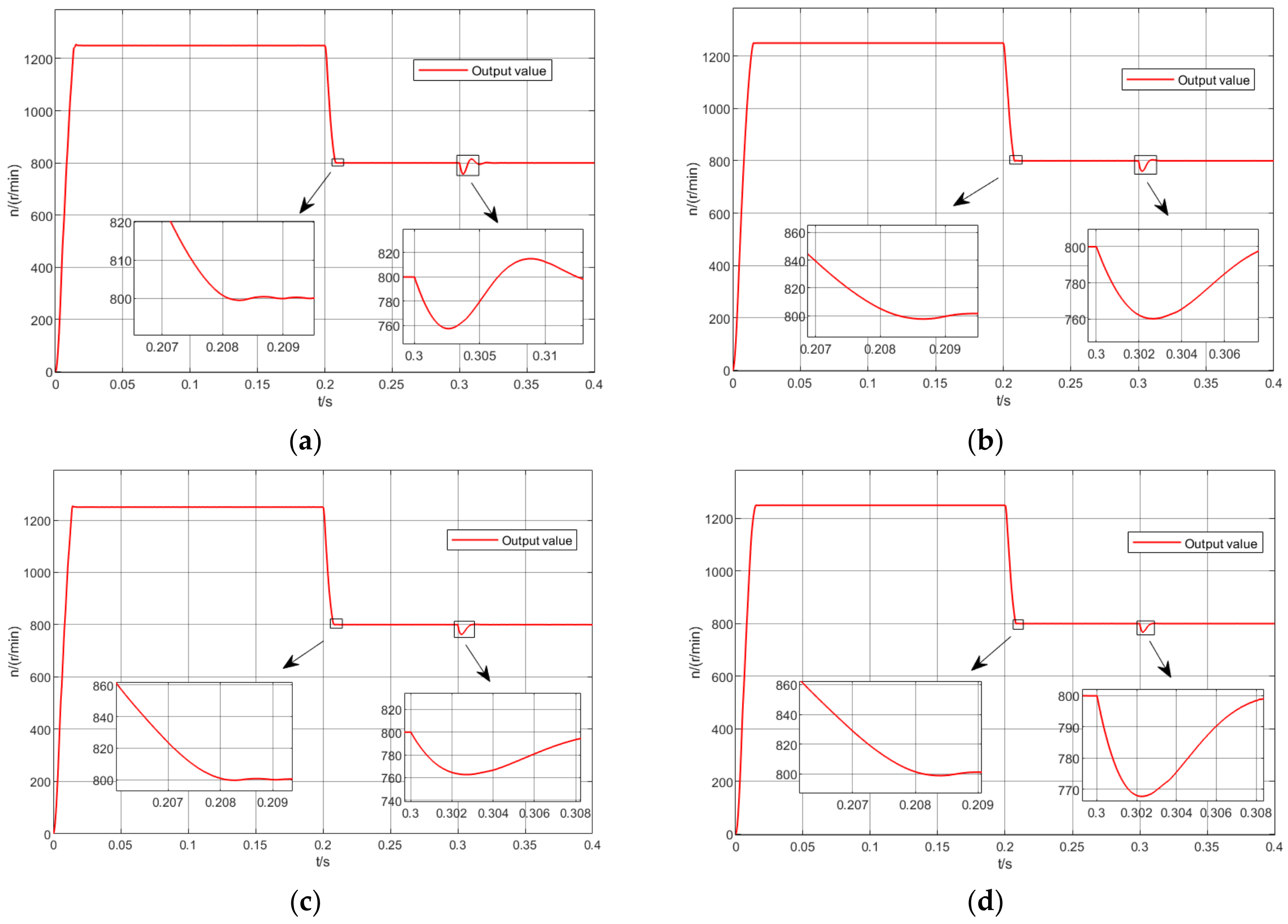

Figure 13. Different control strategy speed waveforms. (**a**) PSO, (**b**) APSO, (**c**) CPSO, and (**d**) IPSO.

Figure 13a shows that the ADRC controller optimized by the PSO algorithm had small fluctuations in the output curve in PMSM control in the face of a suddenly decreasing speed input, but the speed fluctuation was larger and there were overshooting and oscillations after the load was applied. Figure 13b shows the ADRC controller optimized by the APSO algorithm, with a smoother curve in the face of a suddenly decreasing speed input in PMSM control, and the speed recovery curve was smooth without overshooting and oscillations, compared to the PSO algorithm. Figure 13c shows the ADRC controller optimized by the CPSO algorithm, with reduced speed fluctuations after load and no overshooting and oscillations. Compared with the PSO algorithm, the speed fluctuation was reduced after adding load, and the speed recovery curve was smooth, without overshooting oscillation. Figure 13d shows the IPSO method compared to the PSO, APSO, and CPSO algorithms. Facing the sudden change in speed input, speed adjustment was more stable, there was basically no overshooting, and after increasing the load, the speed dynamics of the drop were still the smallest, the speed recovery was fast, there was basically no vibration phenomenon, and the stability of the PMSM and the anti-jamming was more powerful.

The values of speed fluctuation for different methods facing sudden changes are shown in Table 3 below.

Table 3. Speed Fluctuations of Different Particle Swarm Optimization Algorithms.

	PSO	APSO	CPSO	IPSO
Speed value (r/min)	757	760	762	768
Fluctuated value	43	40	38	32

From Table 3, it can be seen that the ADRC controller tuned by the IPSO algorithm proposed in this paper was optimized to cope with speed fluctuation in the PMSM control system in response to a sudden torque increase situation, by 26% compared to the conventional PSO algorithm, by 20% compared to the APSO algorithm, and by 16% compared to the CPSO algorithm.

5.3. Experiment Verification

In order to test the PMSM control system's performance and anti-interference performance, the PMSM experimental platform was built, as shown in Figure 14 below, the rotational speed-tracking as well as external load tests were carried out, respectively, and the experimental data were obtained and plotted by connecting with the host computer.

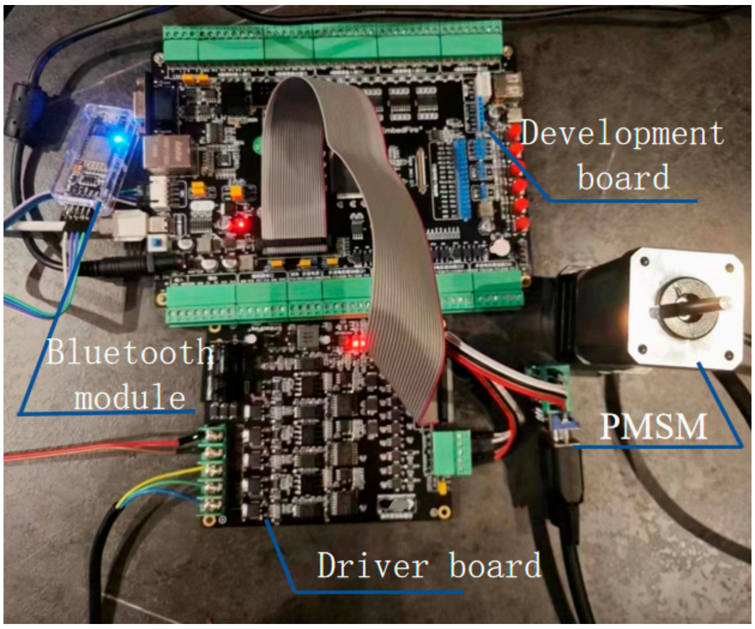

Figure 14. PMSM experimental platform.

The PMSM experimental platform in Figure 14 communicates wirelessly with the host computer via a Bluetooth module to enable remote commands to be given to control the PMSM.

(1) Speed-tracking test

Firstly, the PMSM control system carried out speed-tracking experiments. Starting the motor under no-load condition, the horizontal coordinate in the following Figure 15 indicates the number of sampling points, the interval time of each sampling point is 0.1 s, and the vertical coordinate indicates the rotational speed. The initial speed of the motor was set to 1250 r/min, the speed was changed to reduce it to 800 r/min at 20 s, and the speed-tracking effect of PMSM was obtained, as shown in Figure 15. From the figure, it can be seen that the maximum fluctuation amplitude of rotational speed reached 100 r/min when the PMSM system adopted the ADRC controller for the rotational speed loop, but when the rotational speed loop adopted the ADRC controller of the IPSO algorithm, the maximum fluctuation amplitude decreased to 49 r/min, and the amplitude of fluctuation of rotational speed was still relatively small, even when the rotational speed was changed and could reach the given rotational speed quickly.

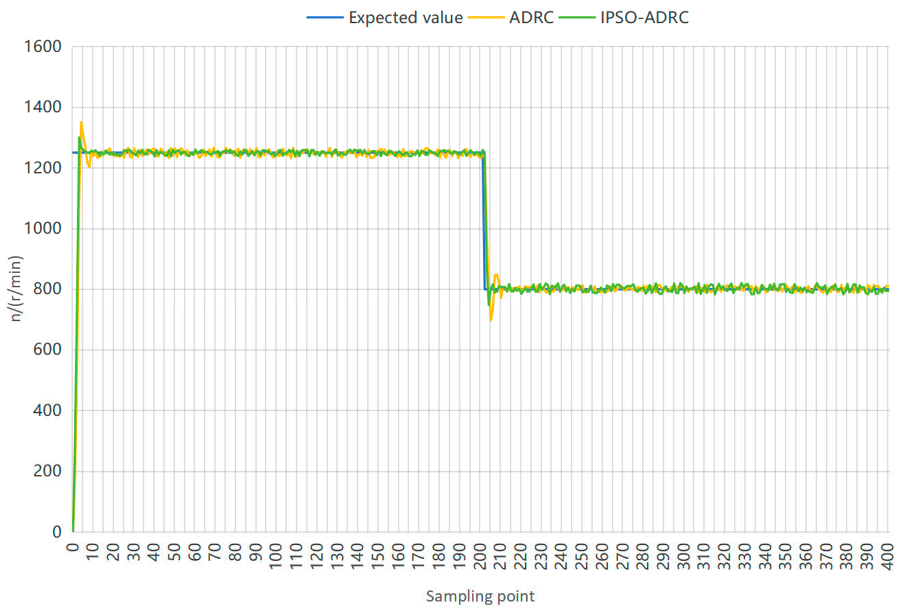

Figure 15. Effect of the speed-tracking test.

(2) Anti-turbulence test

In the PMSM control system anti-disturbance experiment, the motor was started under the no-load condition, the initial speed of the motor was set to 1250 r/min, and 20 N · m load was applied to the motor at 17 s. The PMSM speed graph is shown in Figure 16 below, from which we can see that the speed ring reached 1250 r/min very quickly when the two control methods were used, respectively, but its maximum overshoot was 221 r/min when the ADRC controller was used. After applying perturbation, its overshooting amount reached 221 r/min at maximum, while its overshooting amount was 165 r/min at maximum when using the self-oscillation controller optimized by the IPSO algorithm. It can be seen that its anti-perturbation ability was obviously better than the former, and the recovery time of the ADRC controller optimized by the IPSO algorithm was faster compared with that of the traditional self-oscillation controller.

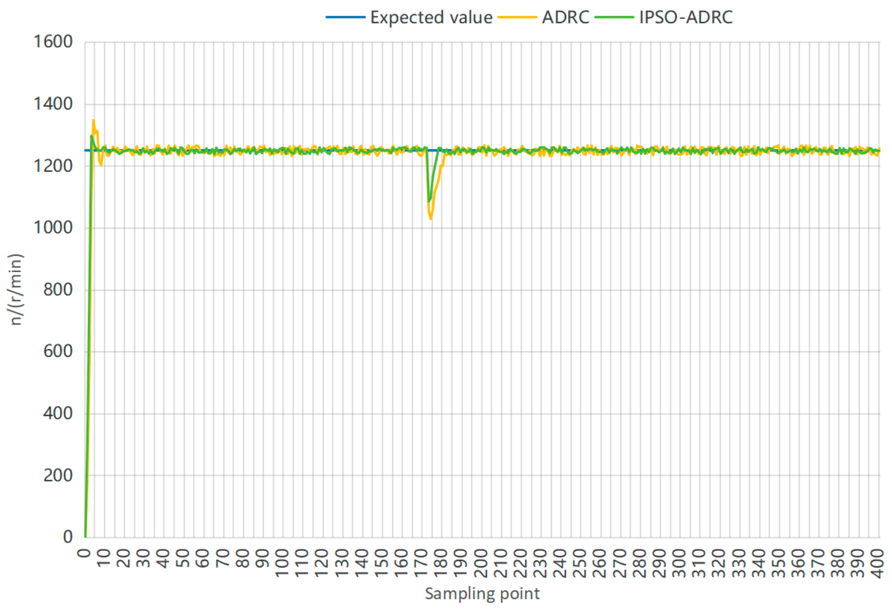

Figure 16. Effect diagram of the anti-disturbance test.

Through the above test experiments on the PMSM control system, the optimized ADRC controller of the IPSO algorithm proved that the speed loop control of the IPSO algorithm had a better speed-tracking ability as well as anti-jamming ability in the PMSM control system, and this verified the feasibility of the ADRC controller based on the improved PSO algorithm proposed in this paper in the PMSM control system.

6. Conclusions

To improve the control performance of the ADRC, this paper addressed the issues of limited search range and susceptibility to local optima in the PSO algorithm. An improved algorithm was proposed based on chaotic initialization, adaptive inertia coefficients, dynamic cognitive factors, and crossover mutation methods. This study introduced an intelligent optimization method based on the improved PSO algorithm for tuning the parameters of the ADRC. A model of the ADRC system for a PMSM was established, and the parameters of the ADRC were optimized. To validate the parameter tuning effect of the proposed optimization algorithm on the ADRC, simulations were conducted using MATLAB2022. The experimental results showed that this method significantly enhanced the performance of the PMSM ADRC system, and compared to the PI control, speed fluctuation was optimized by 26%, improving both stability and robustness.

In the IPSO algorithm, chaotic initialization broadens the search range and prevents local optima, while adaptive inertia coefficients and dynamic cognitive factors effectively accelerate the convergence speed of the particle swarm. The introduction of crossover mutation methods further optimized the local optima issues, enhancing accuracy. Compared to the PSO, APSO, and CPSO optimization algorithms, the IPSO-based ADRC for the PMSM proposed in this paper demonstrated superior tracking and control performance. In the future, the ADRC controller can be further applied and optimized on the PMSM current loop to improve the performance of the entire PMSM control system

Author Contributions: Conceptualization, Z.Z., L.W. and Y.W.; methodology, Z.Z.; writing—original draft preparation, Z.Z.; writing—review and editing, L.W. and Y.W.; visualization, X.Z. and Y.T.; project administration, L.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Key R&D Program of China (Key Special Project for Marine Environmental Security and Sustainable Development of Coral Reefs 2022-3.1).

Data Availability Statement: All data used to support the findings of this study are available from the corresponding author upon request.

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

References

1. Li, X.; Xue, Z.; Yan, X.; Zhang, L.; Ma, W.; Hua, W. Low-Complexity Multivector-Based Model Predictive Torque Control for PMSM with Voltage Preselection. *IEEE Trans. Power Electron.* **2021**, *36*, 11726–11738. [CrossRef]
2. Yang, J.; Chen, W.-H.; Li, S.; Guo, L.; Yan, Y. Disturbance/Uncertainty Estimation and Attenuation Techniques in PMSM Drives—A Survey. *IEEE Trans. Ind. Electron.* **2016**, *64*, 3273–3285. [CrossRef]
3. Wang, Z.; Tian, H.; Huang, L.; Wan, Z. Kalman Observer-Based Active Disturbance Rejection Dead-Beat Predictive Control Algorithm for Electric Machine Emulator Interface Current. *Trans. Beijing Inst. Technol.* **2023**, *43*, 912–925.
4. Xia, C.; Liu, N.; Zhou, Z.; Yan, Y.; Shi, T. Steady-State Performance Improvement for LQR-Based PMSM Drives. *IEEE Trans. Power Electron.* **2018**, *33*, 10622–10632. [CrossRef]
5. Wang, H.; Xu, S.; Hu, H. PID Controller for PMSM Speed Control Based on Improved Quantum Genetic Algorithm Optimization. *IEEE Access* **2023**, *11*, 61091–61102. [CrossRef]
6. Gao, P.; Zhang, G.; Ouyang, H.; Mei, L. An Adaptive Super Twisting Nonlinear Fractional Order PID Sliding Mode Control of Permanent Magnet Synchronous Motor Speed Regulation System Based on Extended State Observer. *IEEE Access* **2020**, *8*, 53498–53510. [CrossRef]
7. Zhan, B.; Zhang, L.; Liu, Y.; Gao, J. Model Predictive and Compensated ADRC for Permanent Magnet Synchronous Linear Motors. *ISA Trans.* **2023**, *136*, 605–621. [CrossRef]
8. Wang, G.; Liu, R.; Zhao, N.; Ding, D.; Xu, D. Enhanced Linear ADRC Strategy for HF Pulse Voltage Signal Injection-Based Sensorless IPMSM Drives. *IEEE Trans. Power Electron.* **2018**, *34*, 514–525. [CrossRef]

9. Xue, W.; Bai, W.; Yang, S.; Song, K.; Huang, Y.; Xie, H. ADRC with Adaptive Extended State Observer and Its Application to Air–Fuel Ratio Control in Gasoline Engines. *IEEE Trans. Ind. Electron.* **2015**, *62*, 5847–5857. [CrossRef]
10. Meng, Y.; Liu, B.; Wang, L. Speed Control of PMSM Based on an Optimized ADRC Controller. *Math. Probl. Eng.* **2019**, *2019*, 1074702. [CrossRef]
11. Xu, Z.; Zhang, T.; Bao, Y.; Zhang, H.; Gerada, C. A Nonlinear Extended State Observer for Rotor Position and Speed Estimation for Sensorless IPMSM Drives. *IEEE Trans. Power Electron.* **2019**, *35*, 733–743. [CrossRef]
12. Wei, W.; Xue, W.; Li, D. On Disturbance Rejection in Magnetic Levitation. *Control Eng. Pract.* **2019**, *82*, 24–35. [CrossRef]
13. Chen, Z.; Qin, B.; Sun, M.; Sun, Q. Q-Learning-Based Parameters Adaptive Algorithm for Active Disturbance Rejection Control and Its Application to Ship Course Control. *Neurocomputing* **2020**, *408*, 51–63. [CrossRef]
14. Chen, Z.; Gao, Q. Linear/Nonlinear Switching Extended State Observer. *Control Theory Appl.* **2019**, *36*, 902–908.
15. Lu, W.; Li, Q.; Lu, K.; Lu, Y.; Guo, L.; Yan, W.; Xu, F. Load Adaptive PMSM Drive System Based on an Improved ADRC for Manipulator Joint. *IEEE Access* **2021**, *9*, 33369–33384. [CrossRef]
16. Huang, M.; Ma, Y.; Wan, J.; Chen, X. A Sensor-Software Based on a Genetic Algorithm-Based Neural Fuzzy System for Modeling and Simulating a Wastewater Treatment Process. *Appl. Soft Comput.* **2015**, *27*, 1–10. [CrossRef]
17. Jin, H.; Chen, X.; Wang, L.; Yang, K.; Wu, L. Adaptive Soft Sensor Development Based on Online Ensemble Gaussian Process Regression for Nonlinear Time-Varying Batch Processes. *Ind. Eng. Chem. Res.* **2015**, *54*, 7320–7345. [CrossRef]
18. Pisa, I.; Santin, I.; Morell, A.; Vicario, J.L.; Vilanova, R. LSTM-Based Wastewater Treatment Plants Operation Strategies for Effluent Quality Improvement. *IEEE Access* **2019**, *7*, 159773–159786. [CrossRef]
19. Yang, X.; Huang, Q.; Jing, S.; Zhang, M.; Zuo, Z.; Wang, S. Servo System Control of Satcom on the Move Based on Improved ADRC Controller. *Energy Rep.* **2022**, *8*, 1062–1070. [CrossRef]
20. Zhan, Z.-H.; Zhang, J.; Li, Y.; Chung, H.S.-H. Adaptive Particle Swarm Optimization. *IEEE Trans. Syst. Man Cybern. Part B (Cybern.)* **2009**, *39*, 1362–1381. [CrossRef]
21. Zeng, R.; Zhao, J.; Xiong, Y.; Luo, X. Active Disturbance Rejection Control of Five-Phase Motor Based on Parameter Setting of Genetic Algorithm. *Processes* **2023**, *11*, 1712. [CrossRef]
22. Song, Y.; Liu, Y.; Chen, H.; Deng, W. A Multi-Strategy Adaptive Particle Swarm Optimization Algorithm for Solving Optimization Problem. *Electronics* **2023**, *12*, 491. [CrossRef]
23. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; IEEE: Piscataway, NJ, USA, 1995; Volume 4, pp. 1942–1948.
24. Eberhart, R.; Kennedy, J. A New Optimizer Using Particle Swarm Theory. In *MHS'95: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 4–6 October 1995*; IEEE: Piscataway, NJ, USA, 1995; pp. 39–43.
25. Shi, Y.; Eberhart, R. A Modified Particle Swarm Optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), Anchorage, AK, USA, 4–9 May 1998; IEEE: Piscataway, NJ, USA, 1998; pp. 69–73.
26. Zhang, F.; Fan, W.; Wu, X.; Pedersen, G.F. Performance Testing of MIMO Device with the Wireless Cable Method Based on Particle Swarm Optimization Algorithm. In Proceedings of the 2018 International Workshop on Antenna Technology (iWAT), Nanjing, China, 5–7 March 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–4.
27. Babu, T.S.; Ram, J.P.; Dragičević, T.; Miyatake, M.; Blaabjerg, F.; Rajasekar, N. Particle Swarm Optimization Based Solar PV Array Reconfiguration of the Maximum Power Extraction under Partial Shading Conditions. *IEEE Trans. Sustain. Energy* **2017**, *9*, 74–85. [CrossRef]
28. Delgarm, N.; Sajadi, B.; Kowsary, F.; Delgarm, S. Multi-Objective Optimization of the Building Energy Performance: A Simulation-Based Approach by Means of Particle Swarm Optimization (PSO). *Appl. Energy* **2016**, *170*, 293–303. [CrossRef]
29. Holland, J.H. Genetic Algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
30. Srinivas, M.; Patnaik, L.M. Genetic Algorithms: A Survey. *Computer* **1994**, *27*, 17–26. [CrossRef]