ISSN 2616-5775 Vol. 7, Issue 7: 17-22, DOI: 10.25236/AJCIS.2024.070703

Design of a Self-Balancing Vehicle with PID Control Based on Improved Particle Swarm Optimization Algorithm

Liu Nan[a,*], Lv Yue, You Cheng, Zhang Ranhua, Yao Qijun

School of Mechanical and Electrical Engineering and Automation, Xiamen University Tan Kah Kee College, Zhangzhou, China
[a]liunan1208@foxmail.com
[*]Corresponding author

Abstract: *As a typical nonlinear, time-varying, and strongly coupled system, the stability control of a self-balancing vehicle imposes high demands on control algorithms. Traditional PID control often relies on empiricalparameter tuning when dealing with complex dynamic environments, making it challenging to achieve optimalcontrol performance. Therefore, this study adopts an improved Particle Swarm Optimization (PSO) algorithm tooptimize the parameters of the PID controller. By enhancing the global search capability and local optimizationability of the PSO algorithm, the proportional, integral, and derivative parameters of the PID controller areautomatically adjusted to improve the system's response speed and stability. Experimental results demonstrate that the PID controller based on the improved PSO algorithm outperforms the traditional PID controller in controlling the self-balancing vehicle, effectively enhancing its stability and control accuracy.*

Keywords: *Particle Swarm, Adaptive, PID, Self-Balancing Vehicle*

1. Introduction

A self-balancing vehicle is a typical nonlinear, time-varying, and strongly coupled system[1], and its stability control demands high precision and response speed from the control algorithm. Traditional PID control methods, when applied to self-balancing vehicle control, often struggle to achieve satisfactory results due to their parameter tuning being dependent on experience, which makes them inadequate for dealing with complex and dynamic environments. Therefore, optimizing PID parameters is necessary[2].

To address this issue, this paper introduces an improved Particle Swarm Optimization (PSO) algorithm to optimize PID controller parameters. While the PSO algorithm is simple and effective in principle, its traditional form has limited local optimization capabilities in complex environments[3]. To enhance the performance of the PSO algorithm, this paper improves upon the traditional PSO algorithm by dynamically adjusting the inertia weight, adaptively tuning parameters, and adjusting population diversity, thereby enhancing its global search and local optimization capabilities. The improved PSO algorithm is then used to solve the problem of PID parameter optimization[4].

2. Self-Balancing Vehicle PID Control

The self-balancing vehicle exhibits characteristics of a traditional first-order inverted pendulum system[5]. In this paper, we study the first-order inverted pendulum model, as shown in Figure 1.

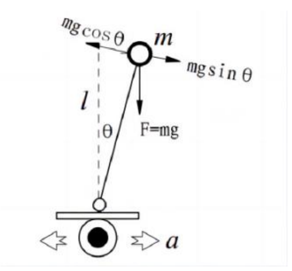

Figure 1: Establishment of the First-Order Inverted Pendulum Model for the Self-Balancing Vehicle

The establishment of the first-order inverted pendulum model as shown in the above figure includes the following parameters: the mass of the cart m, the tilt angle θ, the distance from the center of mass to the wheel axle l, the acceleration of the cart's wheels $a(t)$, and the external force applied to the self-balancing vehicle to maintain balance $X(t)$.

According to Newton's Second Law of Motion, the analysis of forces in the horizontal direction for the model in Figure 2 leads to the establishment of the force equilibrium equation[6]. The equation is as follows:

$$L\frac{\partial^2\theta(t)}{\partial t^2} = gsin\theta(t) - a(t)cos\theta(t) + Lx(t) \tag{1}$$

Generally, the tilt angle of the self-balancing vehicle is not large. Therefore, when $\theta < 10°$, linearization is performed, and let $\sin\theta(t) \approx \theta(t)$. Equation (1) can then be transformed into:

$$\mathrm{L}\frac{\partial^2\theta(t)}{\partial t^2} = g\theta(t) - a(t) + Lx(t) \tag{2}$$

At this point, the self-balancing vehicle has reached equilibrium, $a(t)=0$.Therefore, we can obtain:

$$\mathrm{L}\frac{\partial^2\theta(t)}{\partial t^2} = g\theta(t) + Lx(t) \tag{3}$$

According to control theory, applying the Laplace transform to Equation (3) and simplifying it yields the system transfer function as follows:

$$H(S) = \frac{\Theta(s)}{X(s)} = \frac{1}{s^2 - \frac{g}{L}} \tag{4}$$

The system's two poles can be obtained as:

$$S = \pm\sqrt{\frac{g}{L}} \tag{5}$$

From the above equation, it can be seen that the system's two poles are not on the same side. According to Nyquist's theorem, one of the poles is on the positive real axis, making the overall system of the self-balancing vehicle unstable and unable to maintain balance[7]. The stability of the self-balancing vehicle, represented by $a(t)$, is determined by both θ and $a(\theta)$. To maintain balance, a feedback differential control loop needs to be established[8].

The position closed-loop control adopts:

$$PWM = k_p e(k) + k_i \sum e(k) + k_d[e(k) - e(k-1)] \tag{6}$$

Here, $e(k)$ is the position deviation at time k, and the position information is read by the encoder. k_p is the proportional coefficient, k_i is the integral coefficient, and k_d is the differential coefficient.

The velocity closed-loop control adopts incremental PID control:

$$PWM{+}{=}\ k_p[h(k) - h(k-1)] + k_i h(k) + k_d[h(k) - 2h(k-1) - h(k-2) \tag{7}$$

Here, $h(k)$ is the velocity deviation at time k, obtained from the encoder readings within a unit of time. Different combinations of k_p,k_ih, and $k_d$will affect the performance of the PID controller. Traditionally, PID parameters are determined through empirical tuning methods. In this paper, an improved Particle Swarm Optimization (IPSO) algorithm is used to optimize the PID parameters.

3. Standard Particle Swarm Optimization (PSO) Algorithm

The Particle Swarm Optimization (PSO) algorithm, proposed by Kennedy and Eberhart in 1995, is an intelligent optimization algorithm based on the foraging behavior of birds. This algorithm uses information sharing among individuals in the swarm to gradually approach the optimal solution to the problem[9].

Treating a single bird as a particle, let the position of the i-th particle at time t be $X_i(t)$ and its velocity be $V_i(t)$. The position and velocity update formulas at time t+1 are as follows[10]:

$$V_i(t+1) = \omega V_i(t) + c_1 r_1[P_i(t) - X_i(t)] + c_2 r_2[g(t) - x_i(t)] \tag{8}$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{9}$$

Here, ω is the inertia weight, controlling the influence of the particle's previous velocity on the current velocity; c_1 and c_2 are the cognitive (self-learning) and social learning factors, respectively; r_1 and r_2 are random numbers between 0 and 1; $P_i(t)$ is the best position found by the i-th particle so far; and $g(t)$ is the best position found by the entire swarm.

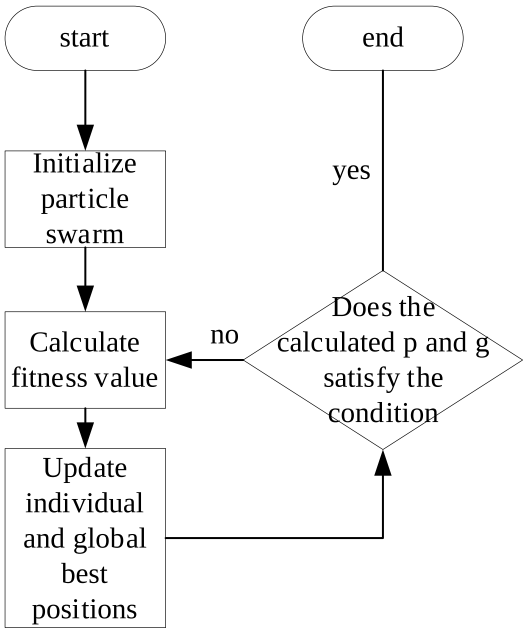
start
end
Initialize particle swarm
yes
no
Calculate fitness value
Does the calculated p and g satisfy the condition
Update individual and global best positions

Figure 2: Standard Particle Swarm Optimization Algorithm Flowchart

4. Improved Particle Swarm Optimization Algorithm

4.1 Dynamic Adjustment of Inertia Weight

In the Particle Swarm Optimization algorithm, the value of ω affects the particle's velocity. When ω is fixed, it may lead to local optima[11]. Therefore, this paper adopts a cosine annealing decrement strategy.

$$\omega(t) = \omega_{\min} + 0.5 * (\omega_{\max} - \omega_{\min}) * (1 + \cos(\pi t / T)) \tag{10}$$

Where: $\omega(t)$ is the inertia weight at generation t; $\omega_{\max}$ is the initial inertia weight; $\omega_{\min}$ is the final inertia weight; T is a constant that controls the decay rate; t is the current iteration number.

The cosine annealing variation strategy provides a larger weight in the early and middle stages to promote global search and avoid premature convergence to local optima. In the later stages, it provides a smaller weight to promote local search. The cosine annealing decrement strategy offers a smooth cosine change curve, which helps maintain the stability of the particle swarm and reduces oscillations during the search process.

4.2 Adaptive Parameter Adjustment Strategy Based on Population Diversity

In the Particle Swarm Optimization algorithm, the learning factors c_1 and c_2 control the particle's movement towards its own best position and the global best position, respectively. Therefore, when the population diversity is large, more global search should be conducted; when the population diversity is small, more local search should be performed[12].

Population diversity is represented by the population standard deviation:

$$\sigma(t) = \sqrt{1/N \sum_{i=1}^{N} (X_i(t) - \overline{X}(t))^2} \tag{11}$$

Where: $\sigma(t)$ is the standard deviation of the population positions at generation t ; N is the population size; $X_i(t)$ is the position of the i-th particle at generation t; $\overline{X}(t)$ is the mean position of all particles at generation t.

Adaptive Learning Factors:

$$c_1 = c_{1\max} + (\sigma(t)/\sigma_{\max}) \cdot (c_{1\max} - c_{1\min})$$
$$c_2 = c_{1\max} - (\sigma(t)/\sigma_{\max}) \cdot (c_{2\max} - c_{2\min}) \tag{12}$$

Where: $\sigma_{\max}$ is the maximum value of the population position standard deviation; $c_{1\max}$ and $c_{1\min}$ are the maximum and minimum values of c_1, respectively; $c_{2\max}$ and $c_{2\min}$ are the maximum and minimum values of c_2, respectively.

5. Experimental Process and Results Analysis

5.1 Experimental Process

To verify the system performance after optimizing PID parameters using the improved Particle Swarm Optimization algorithm, the following Simulink model was established.

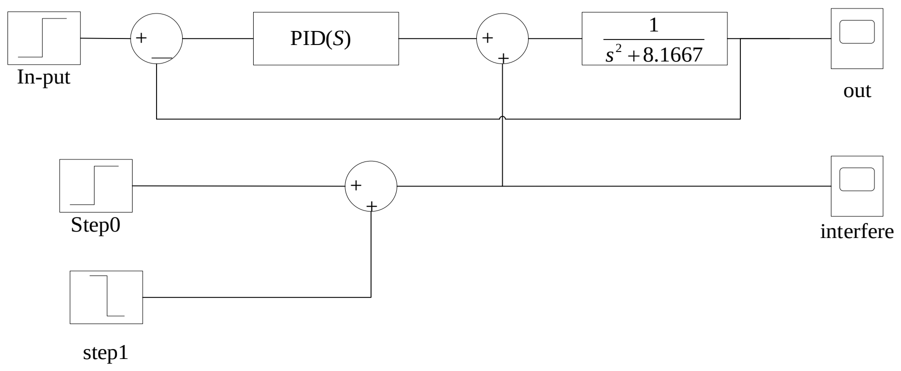

Figure 3: PID Simulation Control

In the model shown in Figure 3, the PID controller first calls a step signal and sets the initial parameters to 0. Then, it calls two step signals to merge their outputs, which are used to simulate the external force signals affecting the balance state of the cart. Finally, the PID controller is used to restore the cart to its balanced state[13].

In the process of optimizing PID parameters using the improved Particle Swarm Optimization algorithm, the number of particles is set to 30, and the maximum number of iterations is 100. Determining the Value of Inertia Weight ω According to Equation (8). ω_{max} and ω_{min} are set to 0.9 and 0.4, respectively. According to Equation (10), the learning factors c_1 and c_2 are determined, where $c_{1max}$and$c_{1min}$are set to 2.5 and 1.5, respectively, and c_{2max} and c_{2min} are also set to 2.5 and 1.5, respectively. The time t for the cart to restore balance is used as the fitness function.

5.2 Experimental Results

The fitness function curve of the Particle Swarm Optimization algorithm is shown in Figure 4. The horizontal axis represents generations, and the vertical axis represents fitness values. The solid line depicts the function curve of the improved Particle Swarm Optimization algorithm, while the dashed line represents the function curve of the standard Particle Swarm Optimization algorithm.

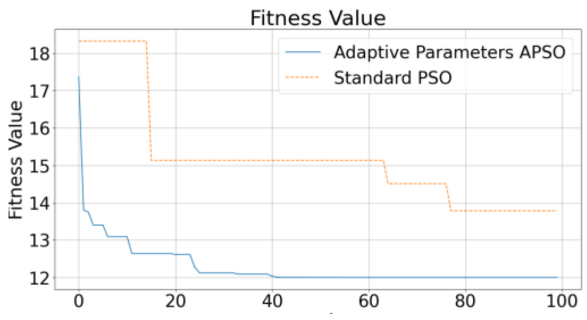

Figure 4: Fitness Curves of the Improved Particle Swarm Optimization and Standard Particle Swarm Optimization Algorithms

As shown in Figure 4, the fitness value of the IPSO algorithm upon convergence is lower than that of the standard algorithm, and it tends to converge at around 40 generations, while the standard PSO algorithm converges at around 80 generations. The IPSO algorithm outperforms the standard PSO algorithm in both speed and quality.

The PID parameters optimized by the IPSO algorithm, the standard PSO algorithm, and the empirical tuning method were used to control the self-balancing vehicle. After stabilization, an acceleration of 2.1g was applied. The adjustment time *t* is the interval between applying the acceleration to the vehicle and the vehicle reaching balance. The results are shown in Table 1.

Table 1: Comparison of Adjustment Times for Different Algorithms

Performance Indicator	Empirical Tuning	Standard PSO	Improved PSO
k_p	-400	-512	-480
k_i	-0.0024	-0.0021	-0.0029
k_d	-1.95	-2.14	-1.92
Adjustment Time *t*	1.71s	1.21s	0.86s

As shown in the table, the adjustment time for the experimental group with empirically tuned parameters is 1.71 seconds, which is the longest among all experimental groups. The adjustment time for the experimental group with parameters tuned using the standard PSO algorithm is slightly shorter, at 1.21 seconds. The experimental group with parameters tuned using the improved PSO algorithm has the shortest adjustment time, at 0.86 seconds. This indicates that optimizing PID parameters using the improved PSO algorithm can enhance system performance, enabling the self-balancing vehicle to reach equilibrium more quickly.

6. Conclusion

This paper presents improvements to the standard Particle Swarm Optimization (PSO) algorithm. The inertia weight values in the standard PSO algorithm are adjusted using a cosine annealing strategy to prevent the algorithm from getting trapped in local optima when inertia weights are fixed. The learning factors are optimized based on the population standard deviation to accelerate the algorithm's convergence speed. The improved PSO algorithm is then used to optimize the PID parameters of the self-balancing vehicle system. The results show that the adjustment time using the improved PSO algorithm is reduced by 0.85 seconds compared to the empirical tuning method and by 0.35 seconds compared to the standard PSO algorithm, demonstrating good effectiveness.

References

[1] Chen Yuanwei. Research on Self-Balancing Vehicle Control System Based on PID Algorithm [J]. Instrumentation Technology, 2024, (01): 64-65+82.
[2] Sun Chao, Guo Naiyu, Yan Mingdie, Ding Jianjun. PID Parameter Optimization Using Improved Adaptive Particle Swarm Optimization Algorithm [J]. Journal of China Construction Machinery, 2023, 21(05): 377-382.
[3] Yang Rongkun, Zhu Youcheng, Fan Rui. Wind Turbine Gearbox Temperature Control System Based on PID Optimization with Adaptive Particle Swarm Algorithm [J]. Automation and Instrumentation,

2024, 39(05): 64-67.

[4] Feng Le, Tang Huachun, Gao Liang, Zou Hongmei, Wang Lin, Tan Mian. Adaptive Fine-Tuning Algorithm Based on Particle Swarm Optimization [J]. Intelligent Computing and Applications, 2024, 14(04): 232-237.

[5] Lin Shouguang. Design of Two-Wheel Self-Balancing Vehicle Based on LQR Control [J]. Journal of Tonghua Normal University, 2024, 45(02): 9-17.

[6] Xue Fan, Sun Jinggao, Yan Huaicheng. Modeling and Control Research of Two-Wheel Self-Balancing Vehicle [J]. Chemical Engineering Automation and Instrumentation, 2012, 39(11): 1450-1454+1497.

[7] Pang Xinyu. Design of Two-Stage Inverted Pendulum Control System Based on Two-Wheel Self-Balancing Vehicle [D]. Harbin Normal University, 2020.

[8] Weng Wenwen, Yin Chenbo, Feng Hao, Zhou Junjing. Application of Improved Particle Swarm Optimization Algorithm in Excavator Bucket Position Control [J]. Mechanical Design and Manufacturing, 2020, (02): 166-169.

[9] Nguyen Quy Dang, Milani Sina, Marzbani Hormoz, Jazar Reza Nakahie. Tire-Road Separation Time Reduction by an Adaptive PID Controller Utilizing Particle Swarm Optimization Algorithm [J]. SAE International Journal of Commercial Vehicles, 2021, 14(4).

[10] Mohamed Elhaj Ahmed Mohamed, Yanling Guo. Separately Excited DC Motor Speed Tracking Control Using Adaptive Neuro-Fuzzy Inference System Based on Genetic Algorithm Particle Swarm Optimization and Fuzzy Auto-Tuning PID [J]. IOP Conference Series: Earth and Environmental Science, 2019, 300(4): 2114-2115.

[11] Ren Jin, Li Yibo, Min Chang. Coverage Optimization of Wireless Sensor Networks Based on Improved Particle Swarm Optimization Algorithm [J]. Radio Engineering, 2024,(01):1-8.

[12] Zhou Hengtai, Hao Jinqing, Li Long, Liu Jing. Double Closed-Loop Temperature Control System Based on Particle Swarm Optimization Fuzzy PID [J]. Journal of Taiyuan Normal University (Natural Science Edition), 2024, 23(01): 53-59.

[13] Han Shuai, Liu Manlu, Zhang Junjun, Zhang Hua. Research on Fuzzy Adaptive Compensation Algorithm for Two-Wheel Self-Balancing Vehicle [J]. Mechanical Design and Manufacturing, 2020, (09): 197-200.