

Self-tuning of Servo System PID Parameters based on TD3 Algorithm

Yingzhuang Zhu^{1,3,4}, Zhigang Bing^{1,3,4}, Di Zhao^{2,4,*}, Ying Li^{1,3,4}, and
Zhanhang Lai^{1,3,4}

¹ School of Automation and Electrical Engineering Tianjin University of Technology and Education, Tianjin, China

² Engineering Teaching Practice Training Center of Tiangong University, Tianjin, China

³ Key Laboratory of Information Sensing & Intelligent Control, Tianjin, China

⁴ GongJiang Carbon Peaking and Carbon Neutrality Research Institute, Tianjin, China

*zhaodi@tiangong.edu.cn

Abstract

This paper investigates a deep reinforcement learning-based self-tuning control algorithm for PID parameters of servo motor control systems for the control problem of servo motor control systems in environments with high control accuracy requirements, using PID control parameters as the action space for deep reinforcement learning. The reward function is set according to the dynamic performance response value of the system, the action strategy network is used to select the execution action, and the value network is used to calculate the push reward. Training the control algorithm in a digital twin eliminates the tedious manual parameter tuning process and substantially improves the control performance of the controller.

Keywords

Green Intelligence; PID Parameter Self-tuning; Deep Reinforcement Learning.

1. Introduction

In the industrial field, servo motor control technology has an important value that cannot be ignored [1]. It has the advantages of fast control speed and accurate position [2]. However, the traditional PID control method has the problem of poor accuracy and stability in servo motor speed control [3]. In order to solve this problem, this paper proposes a PID parameter self-tuning and optimization algorithm based on reinforcement learning. The algorithm constructs a reward function based on the dynamic performance index of the PID control system and automatically adjusts the controller parameters by learning the empirical data of the periodic step response [4]. Compared with the traditional parameter tuning method, the algorithm does not need the tedious manual parameter tuning process and can effectively optimize the controller parameters. Experimental results show that the algorithm can significantly reduce the overshoot of the system step response, shorten the response time of the system, and substantially improve the performance of the controller.

2. Control System Control Modeling

This system uses a double closed-loop structure to control the motor speed[5]. Among them, the speed control loop adopts PID control and combines the deep reinforcement learning TD3 algorithm for adaptive adjustment of PID parameters. The regulated output value is used as the input value of the

current loop. The current loop control uses PI control, and the current loop output value is used to adjust the duty cycle of the PWM pulse wave. Finally, the power supply voltage is added to the three windings of the motor through the voltage inverter, so as to change the rotational speed of the motor and complete the control of motor speed regulation. The structure block diagram of the whole double closed-loop motor speed control system is shown below:

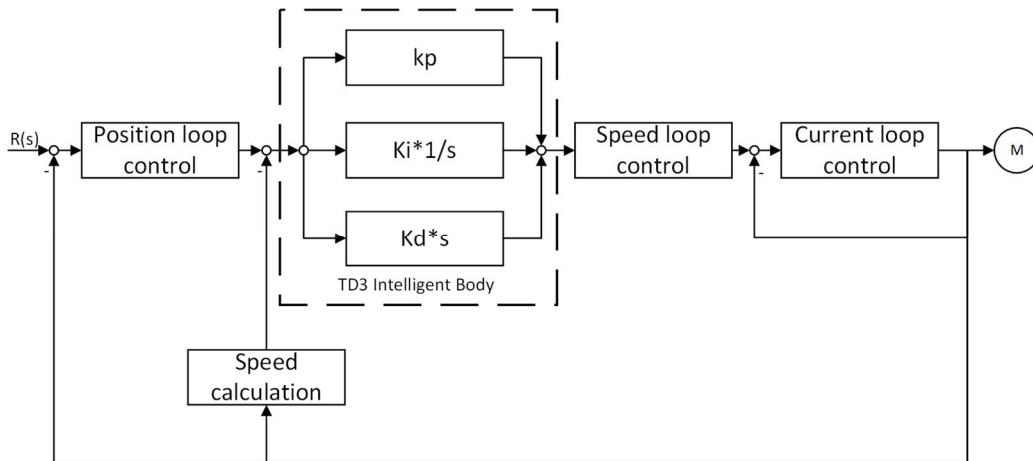


Fig. 1 Block diagram of dual closed-loop motor speed control system

In control systems, PID control algorithms are simple, adaptable, reliable, etc. The mathematical principle of PID is to regulate parameter values and perform linear control based on the deviation between the input value and the feedback.

The mathematical expression for the PID algorithm is:

$$e(t) = y^*(t) - y(t) \tag{1}$$

$$u(t) = K_p[e(t) + \frac{1}{T_i} \int_0^t e(t)dt + T_d \frac{d}{dt} e(t)] \tag{2}$$

$$= K_p e(t) + K_i \int_0^t e(t)dt + K_d \frac{d}{dt} e(t) \tag{3}$$

In the above equation, $u(t)$ is the output value of the PID controller; $\{k_p, k_i, k_d\}$ are proportional, integral, and differential control coefficients, respectively; T_i and T_d are the integral and differential time constants, respectively. The relationship between the time constant and the control coefficient is: $K_i = K_p / T_i$, $K_d = K_p \cdot T_d$. PID parameter tuning takes into account the interrelationships between the three parameters of $\{k_p, k_i, k_d\}$, as well as regulation in different response states.

DC servo motor internal and external influencing factors are variable and complex, while the system is loaded with unknown motor parameters, want to directly obtain the model of DC servo motor is more difficult. Therefore, this paper seeks the mathematical model of DC motor through the identification of DC motor system.

The mathematical model of the DC motor system is known to be a second-order system by the mechanistic analysis method, which is specified as:

$$\frac{n(s)}{U_{d_0}(s)} = \frac{1/C_e}{T_m T_l s^2 + T_m s + 1} \quad (4)$$

where the ratio of rotational speed $n(s)$ to input voltage $U_{d_0}(s)$ is the transfer function of the DC motor. C_e is the coefficient of reverse electromotive force; T_m is the electromechanical time constant of the motor; T_l is the electromagnetic time constant of the armature circuit.

For DC motors, the armature circuit electromagnetic time constant T_l is much smaller than the motor's electromechanical time constant T_m , so the above equation can be approximately equal to the following equation:

$$\frac{n(s)}{U_{d_0}(s)} = \frac{1/C_e}{T_m s + 1} \quad (5)$$

So the motor system can be viewed as a first order system with a delay link, which in turn can be obtained from the unit step response of the system:

$$G(s) = e^{-0.052s} \frac{1}{0.12s + 1} = \frac{1}{0.12s + 1} \cdot \frac{1}{0.052s + 1} \quad (6)$$

The servo motor system is then added to the PID controller, so the final system transfer function is:

$$\frac{C(s)}{R(s)} = \frac{G(s)}{1 + G(s)H(s)} = \frac{(K_p + K_i s + K_d / s) \cdot G(s)}{1 + G(s)} \quad (7)$$

Substitute $G(s)$ into the above equation:

$$\frac{C(s)}{R(s)} = \frac{K_i s^2 + K_p s + K_d + 1}{0.00624s^3 + 0.172s^2 + 2s} \quad (8)$$

3. Markov Decision Process

Markov Decision Processes are actions added to Markov Reward Processes and can be used to model reinforcement learning problems. Usually defined as a tuple $\langle S, A, P, R, \gamma \rangle$, where S denotes the set of environmental states; A represents the set of all actions; P is the state transfer probability matrix. $P_{ss'}^a = P[S_t = s' | S_t = s, A_t = a]$; R is the reward function, Performing action $A_t \in A$ in state $S_t \in S$ yields reward $R_S^A = E[R_{t+1} | S_t = S, A_t = A]$; γ is the discount factor, $\gamma \in [0, 1]$.

Reinforcement learning is a learning method in which an "intelligent body" learns an optimal strategy based on a reward function R , i.e., a sequence of actions that yields the highest cumulative reward,

by continuously interacting with the environment. The strategy is defined as when the given state of strategy π , With respect to the probability distribution of action a , i.e., $\pi(a | s) = P[S_t = s, A_t = a]$, Reinforcement learning uses the value function to evaluate the merits of strategy π . The value function includes the state value function $V(s)$ and the action value function $Q(s, a)$. The state value function $V_\pi(s)$ of the MDP starts from state s , and the target gain obtained by executing strategy π is used to measure the value size of the intelligent body when it is in state s . The value function $V_\pi(s)$ of the MDP is used to measure the value size of the intelligent body when it is in state s ; The action-value function $Q_\pi(s, a)$ of the MDP is the expectation of the cumulative future harvest obtained after taking action a , starting from state s , and is used to measure the magnitude of the value of performing action a on the current state. To wit:

$$V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) | S_t = s] = \sum_{a \in A} \pi(a | s) \left(R(s, a) + \gamma \sum_{s' \in S} P(s' | s, a) V_\pi(s') \right) \quad (9)$$

$$Q_\pi(s, a) = E_\pi[R_{t+1} + \gamma Q_\pi(S_{t+1}, A_{t+1}) | S_t = s, A_t = a] = R(s, a) + \gamma \sum_{s' \in S} P(s' | s, a) \sum_{a' \in A} \pi(a' | s') Q_\pi(s', a') \quad (10)$$

The corresponding optimal value function is:

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} P_{ss'}^a [R_s^a + \gamma V^*(s')] \quad (11)$$

$$Q^*(s, a) = \sum_{s' \in S} P_{ss'}^a [R_s^a + \gamma \max_{a' \in A} Q^*(s', a')] \quad (12)$$

4. The Main Framework of the TD3 Algorithm

TD3 algorithm is a reinforcement learning algorithm for continuous action control, TD3 is an improved version of the DDPG algorithm, which mainly solves the over-estimation problem and the stability problem that tends to occur in the training process of the DDPG algorithm.

The TD3 algorithm updates the original DDPG algorithm with a dual-Q network and a delayed update strategy.

The TD3 algorithm is structurally the same as the DDPG algorithm and is structurally based on the Actor-Critic (AC) framework as shown in Fig. 2.

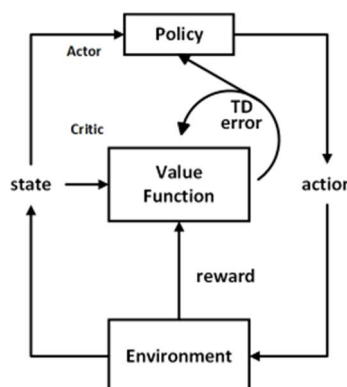


Fig. 2 Schematic diagram of the AC framework algorithm

At the algorithmic level DDPG algorithm combines the advantages of DPG algorithm and DQN algorithm. DDPG absorbs the advantages of single-step updating of the strategy gradient in Actor-Critic, as well as DQN's skill in estimating the Q-value. DDPG can be divided into two large networks, the strategy network and the value network. DDPG continues the idea of DQN fixed target network. On the strategy network Actor, DDPG adopts a deterministic strategy, which is determined directly by the function Determine, Actor outputs a deterministic action, and the network that produces this deterministic action is defined as $a = \mu(s | \theta^\mu)$, Actor network updates the parameters according to Eq. (13):

$$\nabla_{\theta^\mu} J \approx \frac{1}{N} \sum_i \nabla_a Q(s, a | \theta^Q) \Big|_{s=s_i, a=\mu(s_i)} \nabla_{\theta^\mu} \mu(s | \theta^\mu) \Big|_{s_i} \quad (13)$$

On the value network Critic, unlike the traditional AC structure, DDPG predicts the Q -value instead of the V -value in the Critic network, where the expression for the output target value y_i is:

$$y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} | \theta^{\mu'}) | \theta^Q) \quad (14)$$

Update Critic by minimizing the loss function L :

$$L = \frac{1}{N} \sum_i (y_i - Q(s_i, a_i | \theta^Q))^2 \quad (15)$$

The parameters of the DDPG target network are updated according to equation (16):

$$\begin{cases} \theta^Q \leftarrow \tau \theta^Q + (1 - \tau) \theta^{Q'} \\ \theta^{\mu'} \leftarrow \tau \theta^{\mu'} + (1 - \tau) \theta^{\mu} \end{cases} \quad (16)$$

where the range of values of τ is (0,1).

TD3 algorithm is to improve on the value network Critic by adding a dual Critic network, in the DDPG algorithm, the training objective of the value network Critic is to minimize the gap between the Q-value and the target Q-value, but since the target Q-value is calculated from the same value network Critic, it is prone to over-estimation, TD3 algorithm is used to compute the target Q-value by introducing two different value networks Critic, and then take the minimum of the two target Q-values as the final target Q-value.

Another improvement of the TD3 algorithm over the DDPG algorithm is the introduction of a delayed update strategy, i.e., the parameters of the Actor and Critic networks are updated again at regular intervals. In the DDPG algorithm, each time the parameters of the Actor and Critic networks are updated, they are updated using the current experience samples. However, since the effect of each action takes a certain amount of time to manifest itself in the problem of continuous action control, updating using the current experience samples is prone to overfitting, and the introduction of a delayed update strategy can solve this problem.

Overall, the TD3 algorithm offers significant performance and stability improvements over the DDPG algorithm.

The DDPG has four network structures, while the TD3 algorithm has six network structures, namely Actor and Target-Actor networks, and two sets of Critic and Target-Critic networks.

The Actor network uses a 3-layer neural network using a fully connected structure, the network takes the state space as input; the network takes $\{k_p, k_i, k_d\}$ i.e. the action space as output and the output is the control parameters of the PID controller. The purpose of the Actor network is to output the action a_t that maximizes $Q(s_t, a_t)$ based on the current state s_t .

Adjust the PID parameters according to the system state Actor network to realize the precise control of the servo system.

The Critic network uses a 3-layer neural network using a fully connected structure, where the network takes state-action pair (s_t, a_t) as input and the output is $Q(s_t, a_t)$. The purpose of the Critic network is to output the most accurate $Q(s_t, a_t)$ for (s_t, a_t) based on the state action.

The training process used is the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm to train the neural network and get the optimal valuation network and policy network. The structure of the controller is shown in Fig. 3.

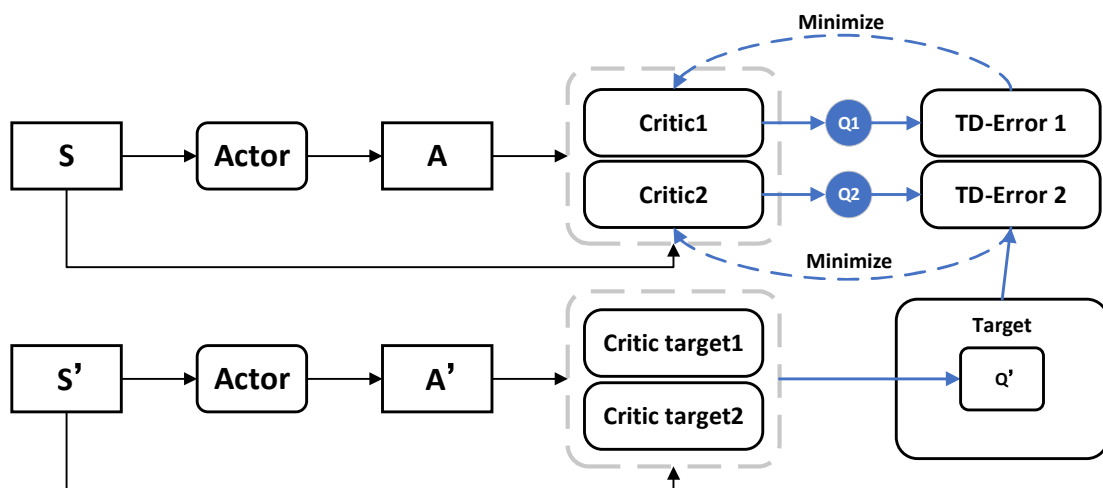


Fig. 3 TD3 Algorithm A-C Network

5. Softmax Strategy Selection

The advantages and disadvantages of the AGENT's choice of action are determined based on the gains, after which the strategy selection method is improved to enable the AGENT to choose the optimal strategy as much as possible.

Selection strategies often used in reinforcement learning are ϵ -greedy method, UCB method and Softmax, ϵ -greedy method first set a very small value of ϵ , $1-\epsilon$ probability of choosing the current action with the largest gain, ϵ probability to explore, the disadvantage is that it can not well distinguish between sub-optimal action and negative or low gain action; ϵ -greedy method first set a very small value of ϵ , $1-\epsilon$ probability of choosing the current action with the largest gain, ϵ probability to explore.

UCB, the action with the optimal upper confidence limit is selected each time, which improves the selection probability of suboptimal and higher yielding actions, with the disadvantage of greatly reducing the selection probability of the optimal action. Replacing the probability of an action being selected with weights, the mathematical description of action selection is:

$$W_j = \frac{Q(j)}{\sum_{i=1}^N Q(i)} \quad (17)$$

There are a total of N actions, $Q(j)$ denotes the estimated average benefit of the j th action, and W_j denotes the weights assigned to the actions.

If the average return is negative, the above weights are likely to be negative and improved that is:

$$W_j = \frac{Q_n(j) - \min(Q_n)}{\sum_{i=1}^N [Q_n(i) - \min(Q_n)]} \quad (18)$$

where $\min(Q_n)$ is the current minimum return.

If the difference between the mean value of the action returns is very large in some cases and very small in others, the above case is not a good way to distinguish the actions with a very small difference in the mean value of the returns, which is improved to be the expression of the probability of action selection for the Softmax strategy:

$$P(A_j) = \frac{\exp\left(\frac{Q(A_j)}{\tau}\right)}{\sum_{i=1}^N \exp\left(\frac{Q(A_i)}{\tau}\right)} \quad (19)$$

Where A_j refers to action j , τ is the temperature coefficient, and the actual problem can be adjusted if τ is smaller, the probability of selection of the optimal action is close to 1, and the larger τ is, the more the selection of each action converges to the average.

6. Action Space, State Space and Reward Function Design

The servo system of the storage robotic arm adopts PID control, and the TD3 algorithm is added to the PID parameter adjustment process, and the deep reinforcement learning is utilized to complete the PID parameter self-tuning of the servo system.

The experiments in this paper use a PWM pulse-controlled servo motor as the controlled object and use the TD3 algorithm for self-tuning of the PID parameters. The control objective is to accurately feed the material into the storage bin, which requires very high control accuracy because the objects to be stored are very small. In order to realize this goal, the dynamic response index of the system is chosen as the state observation quantity in this paper.

The algorithm uses the PID control parameter $\{k_p, k_i, k_d\}$ as the action space for deep reinforcement learning.

As the size of the PID control parameters can produce different control effects even with small changes, in order to allow the action to be fully explored, this paper sets the range of the three PID parameters at [6,20,6]. Also the PWM pulse control signal is converted into 3 stages i.e. the first stage of 0R~1000R, the second stage of 1000R~-1000R, and the third stage of -1000R~1000R, each stage is set up with a 10-step exploring process, and the actions within a stage are accumulated, and the range of the three parameters of the PID for each stage is [60,200,60], and every time the switching stages, the accumulation of actions in the previous stage will be refreshed.

Therefore, the dynamic indicators of the step response of the control system, the overshoot σ , the regulation time t_s , the steady state error e_{ss} , and the current signal phase *times* in which the controlled system is located are used as the observation space of the algorithm.

The overshoot σ is defined as the percentage by which the highest point value of the step response of the controlled system exceeds the steady state value of the step response, as in the following equation:

$$\sigma\% = \frac{y(t_p) - y(t_\infty)}{y(t_\infty)} \times 100\% \quad (20)$$

Where t_p is the peak time, i.e., the time when the system step response reaches the highest point; t_∞ is the infinite time, i.e., the time when the system step response reaches the steady state.

The conditioning time t_s is defined as the minimum time for which the system step response enters and remains within the $\pm 3\%$ error band of $C(\infty)$. The conditioning time t_s is defined as the minimum time for which the system step response enters and remains within the $\pm 3\%$ error band of $C(\infty)$.

The steady state error e_{ss} is defined as the difference between the steady state $y(t_\infty)$ and $C(\infty)$ of the step response of the system as in the following equation:

$$e_{ss} = |y(t_\infty) - C(\infty)| \quad (21)$$

Reward function plays a crucial role in reinforcement learning, which is used to guide the intelligent body to learn according to the reward signal and finally achieve the desired goal. The reward function can be divided into two types, discrete and continuous, and the discrete reward function is suitable for solving the reinforcement learning problem of some simple actions, and for the complex reinforcement learning problem, the discrete reward function is difficult to guide the intelligent body to explore the optimal action due to the sparse distribution of rewards. Therefore, this paper adopts the combined reward function, taking the inverse of the dynamic response index as the base reward function, with the discrete reward for reaching the expected index.

With such a designed reward function, the intelligent body can learn according to the reward signal and gradually adjust its strategy to achieve the desired goal. Compared with the discrete reward function, the combinatorial reward function can better guide the intelligent body to explore the optimal action, thus improving the efficiency and stability of reinforcement learning.

According to this situation, the reward function designed in this paper is as follows:

(1) The basis reward function is constructed from the inverse of the dynamic step response metrics {overshooting amount σ , regulation time t_s , steady state error e_{ss} } after the decomposition:

$$R = \frac{1}{\sigma + t_s + e_{ss} + k} \quad (22)$$

k in the above equation is a parameter that avoids a zero denominator.

(2) The reward accumulation term is a discrete reward when the control objective is completed within the specified number of exploration steps to the next stage:

$$R = \frac{1}{\sigma + t_s + e_{ss} + k} + r(\sigma, t_s, e_{ss}) \quad (23)$$

where $r(\sigma, t_s, e_{ss})$ is the segmentation function, and the discrete reward is obtained when $\{\sigma, t_s, e_{ss}\}$ reaches a set threshold:

$$r(\sigma, t_s, e_{ss}) = \begin{cases} 300 \times stage & \sigma < 0.02, t_s < 0.03, e_{ss} < 2 \\ 0 & \text{else} \end{cases} \quad (24)$$

where *stage* is the current stage of the controlled system, and when the set threshold is reached, the discrete reward for the corresponding stage is harvested.

(3) The required number of steps in each phase is 10, with discrete reward decay for each step performed:

$$R = \frac{1}{\sigma + t_s + e_{ss} + k} + r(\sigma, t_s, e_{ss}) - 10 \times \text{step} \quad (25)$$

Where *step* is the number of action steps in each phase, the attenuation reward is growing in the form of an equidistant series, each action step will be rewarded for attenuation, the attenuation of the magnitude of the response to the growth of 10, when *step* reaches 10 when the mandatory termination of the action, so that the object to be controlled to enter into the next phase of control, and at the same time will be *step* set to zero.

7. Analysis of Experimental Results

The experimental results in this paper are based on Deep Reinforcement Learning TD3 algorithm in a customized environment. Where the hardware and software environment required for the experimental environment is shown in the table:

Table 1. Hardware and software environment table

Parameters	Configure
Operating system	Windows 11
CPU	12th Gen Intel(R) Core(TM) i5-12400F 2.50 GHz
Running memory	16G
Deep Learning Framework	Pytorch 0.6
Computer language environment	Python3.6.4

In the customized environment, the action is a 3-dimensional data and the state is a 4-dimensional continuous data, and the experimental parameter list is shown in the following table:

Table 2. Experimental parameter table

Parameters	Value
Experience playback pool size	3000
Dimension of Action	3
State dimension	4
Learning rate	0.0003
Discount factors	0.9
Sample size	200

In the specific experiments, the strategy for the first 3000 time steps is a randomized strategy, where the initial training data is collected through random exploration of the intelligences. (s, a, s', d) Markov decision trajectory contained in the intelligent body from the initial state to the termination state is called a round or an episode. the termination state of this training environment is when the 4th value of the state is equal to 3. The termination state of this training environment is when the 4th value of the state equals 3. The experience pool playback pool size refers to the number of quaternions A stored in the experience playback pool; the learning rate is a parameter by which the algorithm evaluates the network in terms of updating using the loss function; and the reward discount rate is a measure of the importance of future rewards, and is used in reinforcement learning algorithms to balance the weights of immediate and future rewards.

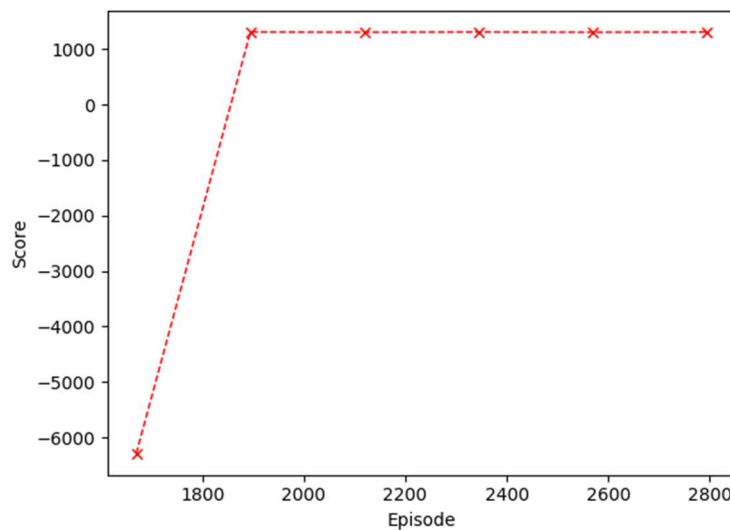


Fig. 4 Reward Change Chart

The variation of $\{k_p, k_i, k_d\}$ during the simulation is shown below and the final parameter variation at each stage is shown below.

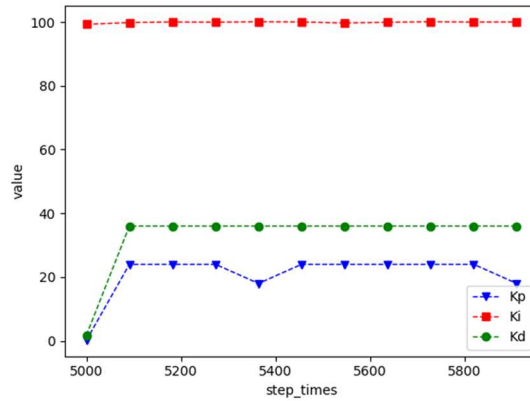


Fig. 5 Parameter variation chart

At about 5100 iterations, the parameters converge.

The model obtained from the self-tuning of PID parameters based on the TD3 algorithm proposed in this paper is used to carry out the experiments on the motor speed following of the servo system, and the experimental results are shown in Fig:

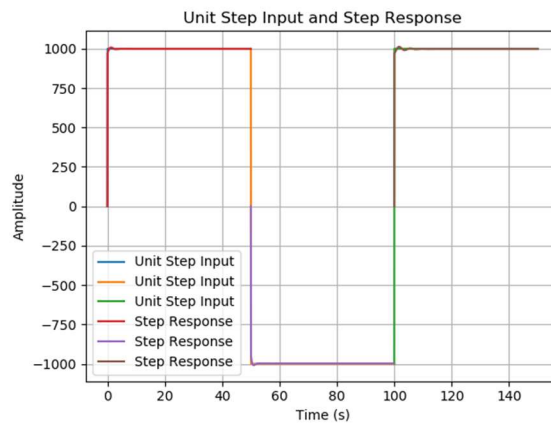


Fig. 6 Experimental results graph

The dynamic response values of the final system are given in the table below:

Table 3. Dynamic Response Table

Dynamic Response Indicator	Point	Response value
Overshoot σ	1	0.0138
	2	0.0138
	3	0.0136
Adjustment time t_s	1	0.0225
	2	0.0187
	3	0.0216
Steady-state error e_{ss}	1	1.28
	2	1.40
	3	1.92

8. Conclusion

In this paper, for the control problem of servo motor system, TD3 algorithm in deep reinforcement learning algorithm is chosen to complete the self-tuning problem of PID parameters of the control system, using deep reinforcement learning algorithm to adjust the action value of the intelligent body, and designing combinatorial reward function to guide the intelligent body to learn towards the optimal strategy. Compared with the human-tuned PID controller, the deep reinforcement learning algorithm learns and fits a better control strategy through neural networks, and the state of the system adaptively adjusts the PID parameters during the control process, which effectively improves the dynamic response performance and stability of the system.

References

- [1] Fan, X. N., "A Simulation Study of PSO-Based Self-Tuning of Brushless DC Motor PI Parameters", Journal of Taiyuan College (Natural Science Edition), Vol. 40, No. 3, pp. 45-50, 2022, doi: 10.14152/j.cnki.2096-191X.2022.03.008.C. Li, W.Q. Yin, X.B. Feng, et al. Brushless DC motor stepless speed regulation system based on fuzzy adaptive PI controller, Journal of Mechanical & Electrical Engineering, vol. 29 (2012), 49-52.
- [2] Zhang Quan and Bai Jigang, "Research related to servo motor control technology", Southern Agricultural Machinery, Vol. 54, No. 2, pp. 35-37, 2023.
- [3] Zongtao Luan, Tao Tao, Xuesong Mei and Nogang Sun, "Research on Pulse Sequence Position Control of AC Servo System", Journal of Xi'an Jiaotong University, Vol. 43, No. 12, pp. 35-39, 2009.
- [4] J.Z. Yan and X.T. Zhuan, "Parameter self-tuning and optimization algorithm based on reinforcement learning", Journal of Intelligent Systems, Vol. 17, No. 2, pp. 341-347, 2022.
- [5] M. Chen, J. Wang and C. Liang, "Brushless DC motor speed control system based on ISSA-Fuzzy-PID algorithm", Computing Technology and Automation, Volume 42, Issue 1, Pages 15-21, 2023, doi: 10.16339/j.cnki.jsjsyzdh.202301003.