Hindawi
International Journal of Aerospace Engineering
Volume 2023, Article ID 7924190, 16 pages
https://doi.org/10.1155/2023/7924190

Research Article

Intercept Guidance of Maneuvering Targets with Deep Reinforcement Learning

Zhe Hu,[1] Liang Xiao,[1] Jun Guan,[2] Wenjun Yi,[1] and Hongqiao Yin[1]

[1]*National Key Laboratory of Transient Physics, Nanjing University of Science and Technology, Nanjing 210094, China*
[2]*Jiangsu University of Science and Technology, Zhenjiang 212000, China*

Correspondence should be addressed to Zhe Hu; hz@njust.edu.cn

Received 17 October 2022; Revised 20 July 2023; Accepted 26 August 2023; Published 13 September 2023

Academic Editor: E Jiaqiang

In this paper, a novel guidance law based on a reinforcement learning (RL) algorithm is presented to deal with the maneuvering target interception problem using a deep deterministic policy gradient descent neural network. We take the missile's line-of-sight (LOS) rate as the observation of the RL algorithm and propose a novel reward function, which is constructed with the miss distance and LOS rate to train the neural network off-line. In the guidance process, the trained neural network has the capacity of mapping the missile's LOS rate to the normal acceleration of the missile directly, so as to generate guidance commands in real time. Under the actor-critic (AC) framework, we adopt the twin-delayed deep deterministic policy gradient (TD3) algorithm by taking the minimum value between a pair of critics to reduce overestimation. Simulation results show that the proposed TD3-based RL guidance law outperforms the current state of the RL guidance law, has better performance to cope with continuous action and state space, and also has a faster convergence speed and higher reward. Furthermore, the proposed RL guidance law has better accuracy and robustness when intercepting a maneuvering target, and the LOS rate is converged.

1. Introduction

Among modern missile missions, improving the accuracy of the guidance system is the most important and difficult process. The main issue is designing a guidance law, which plays an important role in the missile guidance system, since it will directly affect the relative motion of the missile and target, and also has great effects on the final miss distance. Proportional navigation guidance (PNG) law has been widely used in many aircrafts of different types for a long time. Although PNG algorithms have achieved excellent performances in many works, they still have the disadvantage of insufficient terminal guidance capability due to their own properties. However, PNG makes it easy to produce a divergent acceleration command in the terminal guidance stage since the acceleration is proportional to the LOS rate. As a result, researchers have developed various sophisticated guidance strategies such as sliding-mode guidance law [1–3] and finite-time convergent guidance law [4–6] to further improve the performance. Some researchers are attempting to merge early artificial intelligence hypotheses in order to design new intelligent algorithms. For instance, Hossain et al. [7] employ a genetic algorithm to generate training data for neural networks and then apply neural networks to optimize guidance commands based on the current states and terminal conditions. For irregular and difficult guidance situations, Kasmaiee et al. [8] couple two types of computational intelligence algorithms, including neural networks and genetic algorithms for optimization. In [9, 10], a genetic algorithm was implemented as the optimization method; since a large number of numerical simulations were required for this purpose, an artificial neural network was employed for training a function between the control parameters and the airfoil aerodynamic coefficients. Kasmaiee and Tadjfar's use of image processing in aerospace applications and improving the efficiency of the spraying system is fully described in the papers [11, 12]. Reference [13] proposes a new guidance law based on the fuzzy logic method. Li et al. [14] consider target acceleration as a bounded disturbance, then use a primal-dual neural network to solve the optimal solution under this constraint, and develop a guidance law based on model predictive control.

Nowadays, with the powerful nonlinear approximation and data representation, data-driven-based RL methods have attracted considerable attention in the design of guidance law, and model-free RL has been implemented challenging constrained, uncertain, multidimensional, and nonphysical systems to estimate the long-term behaviours of the systems under the current policy and to determine the optimal future policies under unknown disturbances [15]. In [16–18], Yang et al. proposed three novel algorithm frameworks for optimal control problems based on the Hamilton-Jacobi-Bellman (HJB) equation. As a computational method of learning through environmental interaction, the Q-learning algorithm is first presented to solve the problem of noncontinuous control. In reference [19], an improved path planning method is proposed, in which the authors combined a fuzzy Q-learning method with a simulated annealing algorithm in the action search policy to balance action exploration and utilization, and the guidance and obstacle avoidance information is also used to design a reward function for the problem of UAV local path planning in an unknown environment. In [20, 21], the authors have applied a Q-learning algorithm to provide a solution for the path planning of multiple aircraft formation flights. The authors in [22, 23] use the Q-learning algorithm, respectively, to design a reinforcement learning guidance law algorithm in two-dimensional and three-dimensional simulation environments and compare it with proportional guidance. It is proved that the guidance law based on the RL algorithm has better accuracy. However, both of them regard the discretized LOS rate as the state space and the discretized normal acceleration as the action space, which is obviously inconsistent with the actual situation. In [24], the authors successfully solved the problem of discontinuity of normal acceleration by discretizing the proportional coefficients as action space; however, such a guidance law has the same defects as PNG, and it cannot solve the problem of divergence of LOS rate in specific circumstances. In order to solve the discontinuity problem of the Q-learning algorithm, in [25], the authors use a convolutional neural network to approximate the behavior value function, which can solve the problem of continuous state space and achieve better results than Q-learning in Atari games. In [26], the authors propose a time-controllable reentry guidance law based on the Deep Q-Network (DQN) algorithm. That is, the neural network is used to generate the bank angle command online and then combined with the amplitude information to form the final bank angle command so that the designed reentry guidance law has good performance in task adaptability, robustness, and time controllability. Schulman et al. [27] propose a Proximal Policy Optimization (PPO) algorithm based on the AC framework, which can solve the problem of continuous action space very well. Gaudet et al. [28] improve the PPO algorithm and propose a three-dimensional guidance law design framework based on reinforcement learning for interceptors trained with the PPO algorithm. Comparative analysis shows that the deduced guidance law also has better performance and efficiency than the extended Zero-Effort Miss (ZEM) policy. This is a good solution to the problem of continuous control, but the PPO algorithm uses random policy to explore and utilize, and this method estimates the accurate gradient and requires a large number of random actions to explore the accurate gradient. Therefore, such a random operation deduced the algorithm's convergence speed. Lillicrap et al. [29] also propose the deep deterministic policy gradient (DDPG) algorithm based on the AC framework. Compared with the PPO algorithm, the DDPG algorithm adopts a deterministic policy to explore and uses Hindsight Experience Replay (HER) to improve the efficiency of training samples and greatly improve the convergence speed of the algorithm. By improving the DDPG algorithm, in [30], the authors propose an online path planning method based on deep reinforcement learning for the problem of UAV maneuvering, target tracking, and obstacle avoidance control. The authors [31] propose a computational guidance algorithm for missile target interception based on the DDPG algorithm. In [32, 33], the authors also proposed a deep reinforcement learning guidance algorithm based on the DDPG algorithm by improving the reward function. In [34], the authors propose a terminal guidance law based on the DDPG algorithm; by designing the environment state and action of the interception problem, the guidance law with optimal learning reward from the interactive data of the simulation environment is realized. In [35], the authors design a missile guidance law using DDPG for maneuvering target interception, but which only considers a maneuvering mode. However, in reinforcement learning algorithms based on value learning, such as DQN [25], the existence of function approximation errors will lead to the problem of Q value overestimation and suboptimal strategies. Fujimoto et al. showed in the paper [36] that the structure that causes the overestimation of the Q value and the accumulation of errors exists in the AC framework, so the authors proposed the "Twin-Delayed Deep Deterministic policy gradient algorithm." The algorithm limits the overestimation of the Q value by selecting the minimum value of the estimates generated in the two critic estimation networks and uses a delayed update policy to reduce the error of each update; this algorithm has been verified on a set of tasks on OpenAI, and it has demonstrated the highest level in each task environment. In [37], the authors propose autonomous navigation of UAV in multiobstacle environments based on TD3. Therefore, we apply TD3 instead of DDPG to improve the estimation accuracy and robustness.

The main contributions are summarized as follows:

(i) This paper presents a new framework for missile guidance law training. According to the problem that may be caused by overestimation bias in the previous RL guidance law, the minimum value between two critics is adopted to reduce overestimation and improve the robustness and accuracy of action output

(ii) According to the analysis of the PNG algorithm, the reward function is designed, which is beneficial for the missile to move towards the direction of decreasing LOS rate and higher accuracy

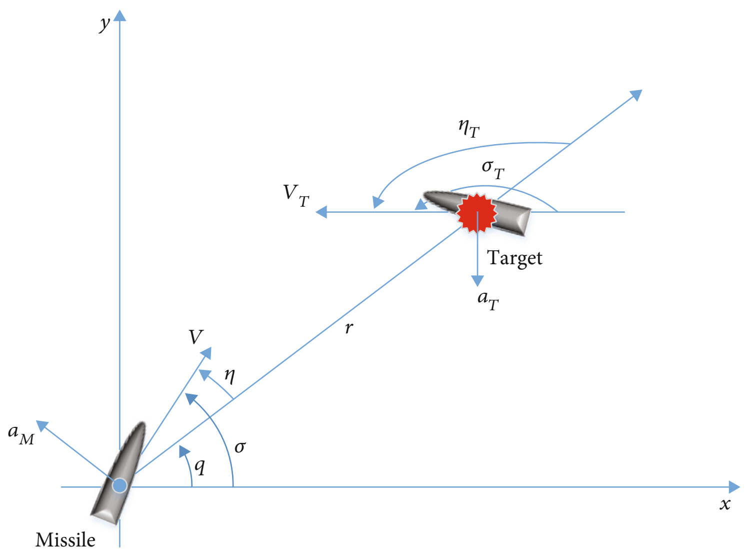


FIGURE 1: Relative motion model (a_M is perpendicular to V, a_T is perpendicular to V_T).

(iii) The open-source Python code of Gym is used to build a digital simulation environment with continuous action space and state space. The simulation results show that compared with the RL guidance law based on the DDPG algorithm, our guidance law has better convergence speed and higher returns under the same knowing environment and has higher accuracy than PNG, and normal acceleration converges better

Section 2 discusses the origin of the problem and previews the basics, including the relative motion model and the Markov decision process. Section 3 establishes the guidance problem under the RL framework and deduces the algorithm principle. Section 4 discusses and analyzes the simulation results of this paper. Section 5 discusses the conclusions from this work.

2. Preliminaries and Problem Setup

2.1. Relative Kinematics and Guidance Problem. To simplify the problem, our work only focuses on the two-dimensional environment of the relative motion between the missile and the target and uses the dynamic analysis method to study the guidance trajectory. The simulated model does not consider gravity, thrust, and drag.

The engagement scenario between the missile and the target in this paper is shown in Figure 1.

In Figure 1, XY denotes the inertial coordinate, M and T refer to the interceptor and the target, V and V_T represent the velocity of the missile and the target, respectively, q is the line-of-sight angle between the missile and the target, $-Q$ and σ_T are the flight path angles of the missile and the target, respectively, η and η_T are the parameters of the leading angles of the missile and the target, respectively, and s_0 and a_T represent the normal acceleration of the missile and the target. In this paper, it is assumed that both the missile and the target are only subjected to a normal overload which is perpendicular to the direction of velocity; that is, the overload is only used to change the direction of each speed.

In a 2D coordinate system, the dynamic equations of motion between the missile and the target are as follows:

$$
\begin{aligned}
r &= \sqrt{(x_t - x_m)^2 + (y_t - y_m)^2},\\
q &= \arctan\left(\frac{y_t - y_m}{x_t - x_m}\right),\\
\dot{r} &= -V_T \cos\eta_T - V\cos\eta,\\
r\dot{q} &= V\sin\eta - V_T\sin\eta_T,\\
q &= \eta + \sigma,\\
q &= \eta_T + \sigma_T,\\
\frac{d\sigma}{dt} &= \frac{a_M}{V},
\end{aligned}
\tag{1}
$$

where $a_{0\_predict}$ is the relative distance between the missile and the target, $\dot{r}$ is the relative speed of the missile and the target, $\dot{q}$ is the LOS rate, x_t, y_t is the position coordinate of the target, and x_m, y_m is the position coordinate of the missile.

At present, the most commonly used guidance method is the method of PNG, in which it usually commands the missile to move to the target by calculating the normal acceleration of the missile. And the missile's normal acceleration is proportional to the change in the line of sight angle rate. To this end, the output of PNG is the normal acceleration a_M, and its description is shown in the following:

$$a_M = KV\dot{q}, \tag{2}$$

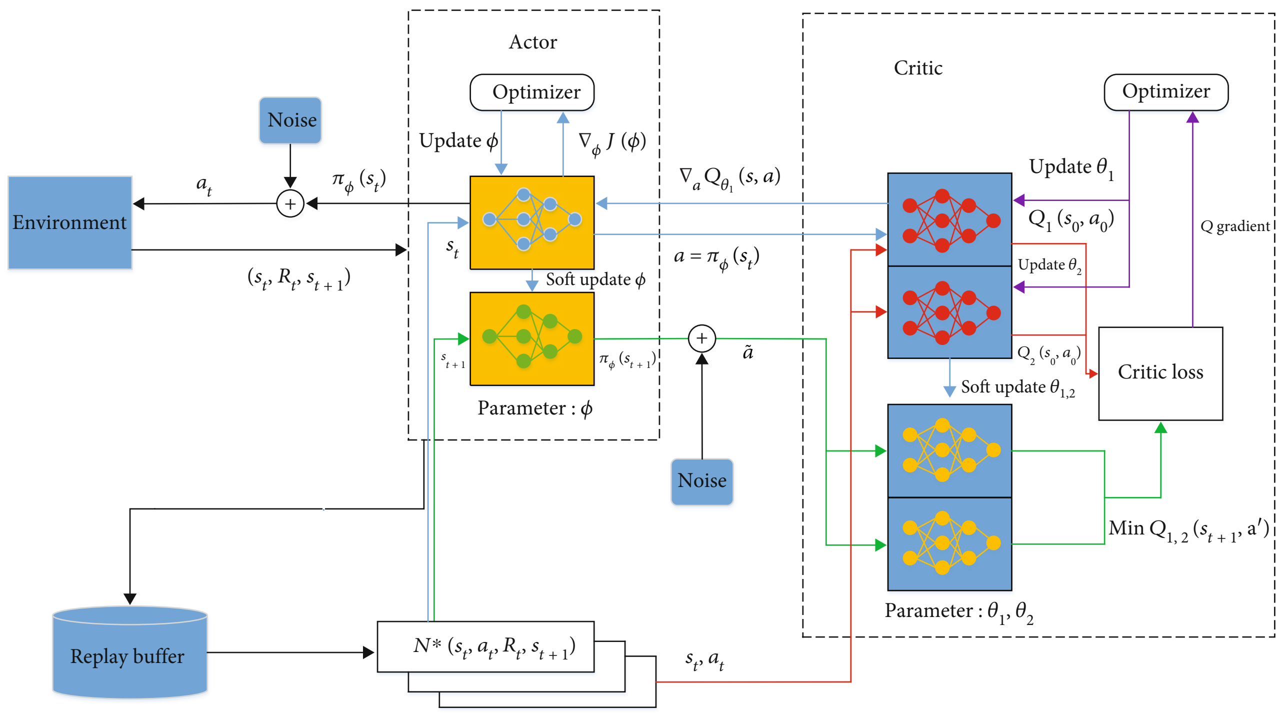


Figure 2: Schematic of the TD3-based RL guidance law (a_t represents the guidance command to be output at time t and s_t represents the LOS rate ϕ captured by the seeker at time t).

where a_M denotes the normal acceleration command, K denotes the proportionality coefficient, and V denotes the relative speed between the missile and the target.

We differentiate the 4th formula in equation (1) with respect to time to obtain the following dynamics:

$$\dot{r}\dot{q} + r\ddot{q} = \dot{V}\sin\eta + V\dot{\eta}\cos\eta - \dot{V}_T\sin\eta_T - V_T\dot{\eta}_T\cos\eta_T. \quad (3)$$

Substitute equation (2) into the final formula in equation (1), we can get $\dot{\sigma} = K\dot{q}$, then substitute it into equation (3).

$$\begin{aligned} \dot{\eta} &= \dot{q} - \dot{\sigma} = (1-K)\dot{q}, \\ \dot{\eta}_T &= \dot{q} - \dot{\sigma}_T, \\ \dot{r} &= V_T\cos\eta_T - V\cos\eta. \end{aligned} \quad (4)$$

Substitute equation system (4) into equation (3)

$$r\ddot{q} = -(KV\cos\eta + 2\dot{r})(\dot{q} - \dot{q}^*), \quad (5)$$

where

$$\dot{q}^* = \frac{\dot{V}\sin\eta - \dot{V}_T\sin\eta_T + V_T\dot{\sigma}_T\cos\eta_T}{KV\cos\eta + 2\dot{r}}. \quad (6)$$

Suppose the target moving in a straight line with a constant speed and the missile velocity is also kept constant, $\dot{V}$, $\dot{V}_T$ and $\dot{\sigma}_T$ are going to be equal to 0. So it can be known from equation (6): $\dot{q}^* = 0$.

Thus, equation (5) can be written as

$$\ddot{q} = -\frac{1}{r}(KV\cos\eta + 2\dot{r})\dot{q}. \quad (7)$$

From equation (7), it is easy to see when $-Q$, i.e., $K > 2|\dot{r}|/V\cos\eta$, then $\ddot{q}\cdot\dot{q} < 0$. In this case, the sign of $\ddot{q}$ will be opposite to that of $\mathrm{L} = -Q(s, a_{0\_predict})$. Therefore, the missile's required normal acceleration decreases as $|\dot{q}|$ decreases, and the missile trajectory becomes flat; this property is the same as PNG.

However, the above conclusion holds in the assumption of a missile with a constant speed moving and a target keeping in a straight line and maneuvering at a constant speed. In real word, the reality will be very complex, so it is obviously different.

2.2. Markov Decision Process. Most algorithms for reinforcement learning are based on the Markov decision process (MDP). When an agent conducts reinforcement learning, the MPD model is often used to establish corresponding mathematical models for decision problems with uncertain state transition probability, state space, and action space and solve the reinforcement learning problem by solving the mathematical model.

A MDP consists of a five-tuple: $M = <S, A, P, R, \gamma>$. The elements of the five-tuple can be described in the following.

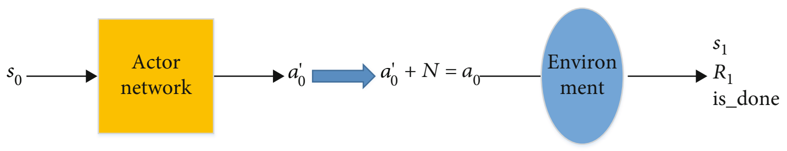
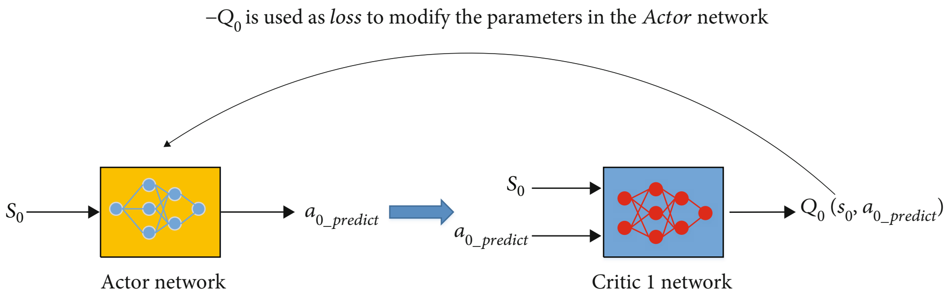


Figure 3: Generating experience procedure.



$-Q_0$ is used as *loss* to modify the parameters in the *Actor* network
S_0
$a_{0\_predict}$
S_0
$a_{0\_predict}$
$Q_0(s_0, a_{0\_predict})$
Actor network
Critic 1 network

Figure 4: Actor network update procedure.

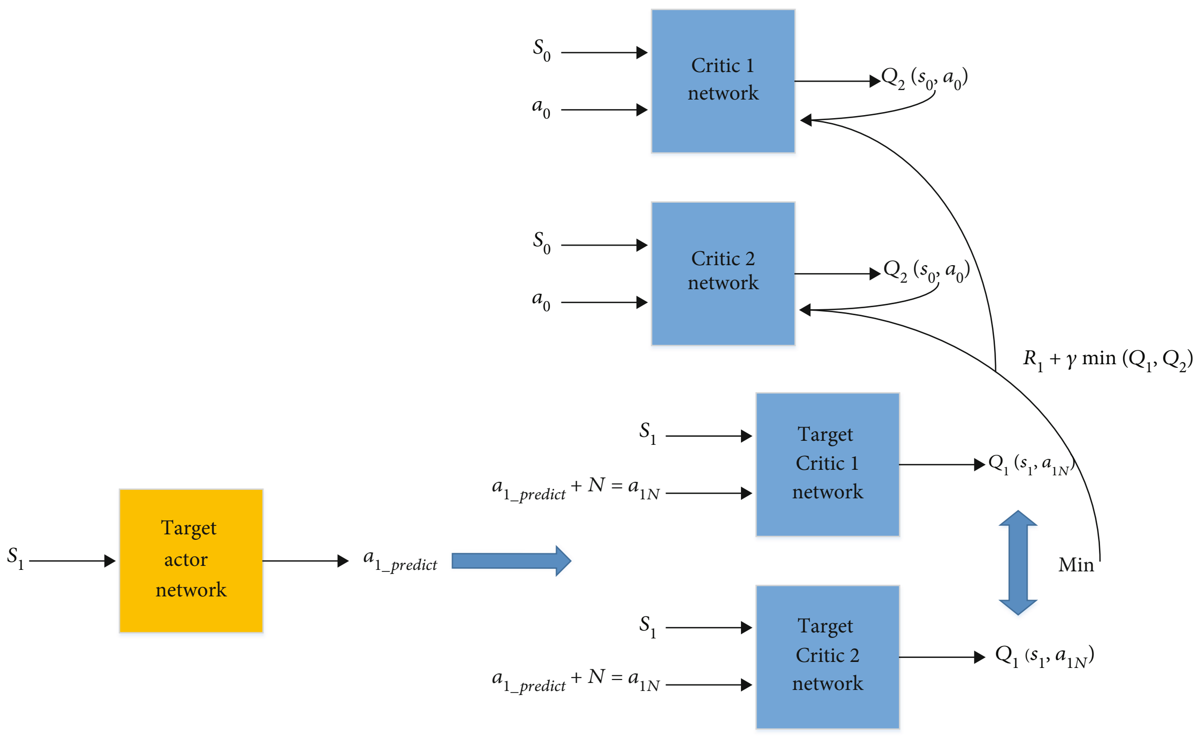


Figure 5: Critic network update procedure.

S represents the state set of the environment, and the state refers to the information that the agent can obtain useful for decision-making. In the reinforcement learning framework, the agent relies on the current state to make decisions.

A represents the action set of the agent. It is the set of actions the agent can choose from in the current reinforcement learning task.

P represents the probability of state transition. $P^a_{ss'}$ represents the probability that in the current state $s (s \in S)$, it will be transferred to another state $s' (s' \in S)$ after action $a (a \in A)$.

$$P^a_{ss'} = P\left(S_{t+1} = s' \middle| S_t = s, A_t = a\right). \tag{8}$$

Given a policy and an MDP: $M = <S, A, P, R, \gamma>$, the probability $P^\pi_{ss'}$ of a state transfer from s to s' when executing policy π is equal to the sum of a series of probabilities, which refers to the product of the probability $\pi(a|s)$

```
Initialize critic networks $Q_{\theta 1}, Q_{\theta 2}$ , and actor network $\pi_\phi$
with random parameters $\theta_1, \theta_2, \phi$
Initialize target networks $\theta_1' \longleftarrow \theta_1, \theta_2' \longleftarrow \theta_2, \phi' \longleftarrow \phi$
Initialize replay buffer $\mathscr{D}$
for t=1 to T do:
    $ENV.reset()$, reset the environment initial value
    While True:
      Select action with exploration noise$a \sim \pi(s) + \varepsilon$, $\varepsilon \sim \mathscr{N}(0, \sigma)$ and
    observe reward $r$ and new state $s'$.
      Store transition experience tuple $(s, a, R, s')$ in $\mathscr{D}$
      Sample mini-batch of $N$ transitions $(s, a, R, s')$ from $\mathscr{D}$
      $\tilde{a} \longleftarrow \pi_{\phi'}(s) + \varepsilon$ , $\varepsilon \sim \text{clip}(\mathscr{N}(0, \tilde{\sigma}), -c, c)$
      $y \longleftarrow R + \gamma \min_{e=1,2} Q_{\theta_e'}(s', \tilde{a})$
      Calculate the TD error
      $\delta_i = R_i + \gamma Q_{\theta'}(s', \tilde{a}) - Q_\theta(s, a)$
      Calculate the loss function $L(\theta_{e=1,2})$
      $L_{e=1,2} = N^{-1} \sum_{i=1}^{N} (y - Q_{\theta_{e=1,2}}(s, a))^2$
      Update the critic network using gradient descent as(where, $e$=1,2)
      $\nabla_{\theta_e} L(\theta_e) = 1/N \sum_{i=1}^{N} \delta_i \nabla_{\theta_e} Q_{\theta_e}(s_i, a_i)$
          ${\theta_e}' = \theta_e + \alpha_{\theta_e} \nabla_{\theta_e} L(\theta_e)$
      if $t$ mod $d$ then
        Update $\phi$ by the deterministic policy gradient:
        $\nabla_\phi J(\pi_\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$
        $\phi' = \phi + \alpha_\phi \nabla_\phi J(\pi_\phi)$
        Update target networks:
        $\theta_i' \longleftarrow \tau \theta_i + (1 - \tau) \theta_i'$
        $\phi' \longleftarrow \tau \phi + (1 - \tau) \phi'$
    end while
end for
```

Algorithm 1: Guidance law learning algorithm based on TD3 algorithm.

of executing an action a and the probability $P_{ss'}^a$ that the action can cause a state transfer from ϕ to s' when executing the current policy π. The specific mathematical expression is as follows:

$$P_{ss'}^{\pi} = \sum_{a \in A} \pi(a|s) P_{ss'}^a, \tag{9}$$

where $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$ is the reward function. R_s^a represents the reward obtained after taking action $a(a \in A)$ in the current state s_0, s_1, a_0. The specific mathematical expression is as follows:

$$R_s^a = \mathrm{E}[R_{t+1} | S_t = s, A_t = a], \tag{10}$$

where γ is the discount factor, take $\gamma = 0.95$. The purpose of using the discount factor is to take into account the immediate return in the future when calculating the cumulative return in the current state.

Since this article uses the model-free reinforcement learning method, MDP does not involve the part of state transition probability. Therefore, we adopt a simplified MDP consisting of a four-tuple: (s_0, a_0, s_1, r_1). Next, we design corresponding items based on the background of solving the problem.

3. Guidance Law with TD3 Algorithm

3.1. Markov Decision Process Design. The environmental information observed by the agent can describe the environmental state to a certain extent. If the simulation covers all the state space encountered by the guidance law in practice and there is enough sampling density, the obtained guidance law will be the optimal guidance law relative to the missile and environmental dynamics modeled. However, in order to better solve the problems in this paper, some information that may interfere with the decision-making task should not be included in the state set denoted by, so we only choose the LOS rate as the state space, which can also cover the whole process of guidance. To this end, we set $S : \langle \dot{q} \rangle$. Taking into account that the characteristics of the projectile itself have certain restrictions on overload, the LOS rate is usually in the range of (-0.5, 0.5) rad/s.

Traditional PNG takes the relative speed and the LOS rate as input, and the output is the normal acceleration. Our goal is to use the neural network to directly map the LOS rate to the normal acceleration so that our actions denoted by A may select the normal acceleration, i.e., $A : \langle N \rangle g$. Here, g is the acceleration of gravity. However, in order to ensure the normal operation of various components, the actual aircraft needs to consider the impact of overload and limit it to a certain range. The overload is limited to 10 times the acceleration of gravity.

The reward function serves as a feedback system that indicates to the missile from the environment whether its performance is good or bad. In order to solve the two problems of achieving LOS rate convergence to 0 and improving guidance accuracy, the reward function designed in this work can be divided into two parts:

$$R_1 = \begin{cases} \dfrac{0.1}{|\dot{q}|}, & \text{otherwise}, \\ 100, & \text{if } \dfrac{0.1}{|\dot{q}|} > 100, \end{cases} \tag{11}$$

$$R_2 = \begin{cases} \dfrac{100}{|r|}, & \text{if } r < 10, \\ 10000, & \text{if } \dfrac{100}{|r|} > 10000. \end{cases} \tag{12}$$

The first part is shown in equation (11). Here, R_1 is the reward generated at the current moment of t. The smaller the $|\dot{q}|$ is, the higher the reward is, and the highest one is 100. And the second part R_2 is a terminal reward. The reward can only be generated when the distance error $r < 10$m. It is expected that the estimated minimum miss can reach 0.01, and the upper limit of the reward will be controlled. The upper limit of the reward will be controlled, and the smaller the r is, the higher the terminal reward the agent gets, which can promote the agent to explore with higher accuracy.

To this end, the final reward function can be defined as follows:

$$R = \begin{cases} R_1 + R_2 & \text{if end \& if } r < 10, \\ R_1 & \text{otherwise}, \end{cases} \tag{13}$$

where end is when the termination condition is true.

3.2. Reinforcement Learning. Reinforcement learning considers the paradigm of the agent's interaction with its environment, and its purpose is to make the agent take the behavior that maximizes the benefits, so as to get an optimal strategy. Based on the above MDP design, the continuous time can be divided into multiple discrete moments t. At each moment, the agent adopting policy π selects the corresponding action a according to the state s and then interacts with the environment to obtain the reward R of this step and the state s'. The return is defined as the discounted sum of the reward.

$$G_t = \sum_{T}^{i=t} \gamma^{i-t} R(s_i, a_i). \tag{14}$$

Table 1: Dynamic model initial simulation parameters.

Parameter	Value
Missile position (m)	(0, 0)
Missile speed (m/s)	1200
Target speed (m/s)	800
Relative range (m)	Random (30000, 40000)
LOS angle (°)	Random (40, 70)
Missile's flight path angle (°)	LOS angle + random (-10, 10)
Target's flight path angle (°)	Random (150, 180)

Table 2: Hyperparameters for TD3 algorithm.

Parameter	Value
Number of hidden layers	2
BATCH_SIZE	32
Replay buffer size	50000
Actor learning rate	10^{-5}
Critic learning rate	2×10^{-5}
Policy noise	0.2
Noise bound	0.5
Soft update factor τ	0.01
Discounting factor γ	0.95
Delay steps	5
Gradient optimizer	Adam

Table 3: Policy and value function network architecture.

Layer	Policy network	Value network
Input layer	1	2
Hidden1	30	10
Hidden2	40	15
Output	1	1

Here is a discount factor $\gamma \in (0, 1)$ used to determine the priority of short-term rewards, R is the result of the return function at time t, and G_t is the cumulative sum of subsequent rewards after the time point.

The action value is defined as the $Q^\pi(s_t, a_t)$ will be obtained when the agent takes action a by following the policy π in the state s at the time t. This value is obtained through the critic function.

$$Q^\pi(s_t, a_t) = \mathrm{E}[G_t | s_t, a_t]. \tag{15}$$

According to the meaning of accumulative rewards above, reinforcement learning proposes the definition of value function calculated by the Bellman equation, which

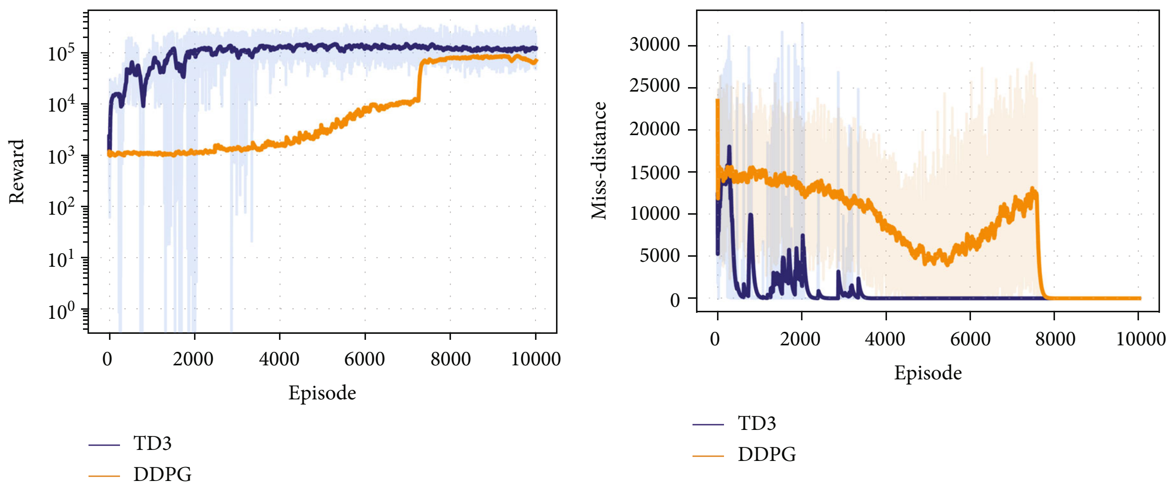


Figure 6: Comparison of learning curves of the two algorithms. (a) The convergence curves of average reward. (b) The convergence curves of the miss distance.

can provide an evaluation value for a certain state of the agent, as shown in the following:

$$Q^\pi(s_t, a_t) = \mathrm{E}_{r_t, s_{t+1}\sim E}\left[R(s_t, a_t) + \gamma \mathrm{E}_{a_{t+1}\sim\pi}\left[Q^\pi(s_{t+1}, a_{t+1})\right]\right]. \tag{16}$$

The deterministic policy gradient (DPG) algorithms [38] map the state to a deterministic action by expressing the policy as a policy function. When the policy is deterministic, the equation for calculating the behavior value function using Bellman's equation will become the following equation:

$$Q^\pi(s_t, a_t) = \mathrm{E}_{R_t, s_{t+1}\sim E}\left[R(s_t, a_t) + \gamma Q^\pi(s_{t+1}, \pi(s_{t+1}))\right]. \tag{17}$$

In the RL framework, the agent's goal is to find an optimal policy Q with parameter ϕ to maximize the total reward received over a sequence of time steps:

$$\phi^* = \arg\max_{\phi} Q(s, \pi_\phi). \tag{18}$$

Supposing policy $y = r + \gamma \min\{Q_1, Q_2\}$ is a deterministic policy, the gradient of the behavior value function to the parameter can easily be calculated by the rule of chain derivation as follows:

$$\frac{\partial Q(s, a)}{\partial \phi} = \frac{\partial Q(s, a)}{\partial a} \cdot \left.\frac{\partial \pi}{\partial \phi}\right|_{a=\pi_\phi} = \nabla_\phi \pi_\phi \cdot \nabla_a Q(s, a)\big|_{a=\pi_\phi}. \tag{19}$$

TD3 uses off-policy to calculate the deterministic policy gradient. Because the deterministic policy gradient itself is not exploratory, the data it generates lacks diversity, so there is no way to learn. The off-policy method is that the agent uses exploratory policy to generate data and calculates the policy gradient based on these data. We still use π to represent the behavior policy, and the policy function parameter ϕ is updated by the following gradient terms:

$$\nabla_\phi J(\phi) = \mathrm{E}_{s\sim p_\pi}\left[\nabla_a Q^\pi(s, a)\big|_{a=\pi(s)} \nabla_\phi \pi_\phi(s)\right]. \tag{20}$$

3.3. Network Structure of TD3. The network structure of the TD3 algorithm can be described in the following:

As illustrated in Figure 2, the actor-network input is environment states, and the output is agent action. Critic network input is the state and action, and the output is the corresponding Q value. The purpose of the actor network is to output the action a_t that maximizes $Q(s_t, a_t)$ according to the state s_t. The larger $Q(s_t, a_t)$ the calculated by a_t is, the better the network training. The purpose of the critic network is to output its action value $Q(s_t, a_t)$ according to the state action (s_t, a_t). The difference between the actor network and the target actor network is that the actor network is updated in the replay buffer at each step, while the target actor network copies the actor's network parameters at intervals to achieve its update. This "lag update" is to ensure the stability of training when training the actor network. The purpose of the target critic network is the same as that of the target actor network. It also wants to use a network that is not frequently updated to make the critic network converge stably. The soft update method between them is as follows:

$$\begin{aligned} \theta' &\longleftarrow \tau\theta + (1-\tau)\theta', \\ \phi' &\longleftarrow \tau\phi + (1-\tau)\phi', \end{aligned} \tag{21}$$

where θ and ϕ, respectively, are the parameters of the critic and the actor network.

Figure 3 shows the process of generating experience. Given a state s_0, get action a_0' through the actor network, and then add noise N to get action $a_0 = a_0' + N$ (noise is to ensure certain exploration), and then we input a_0 into the

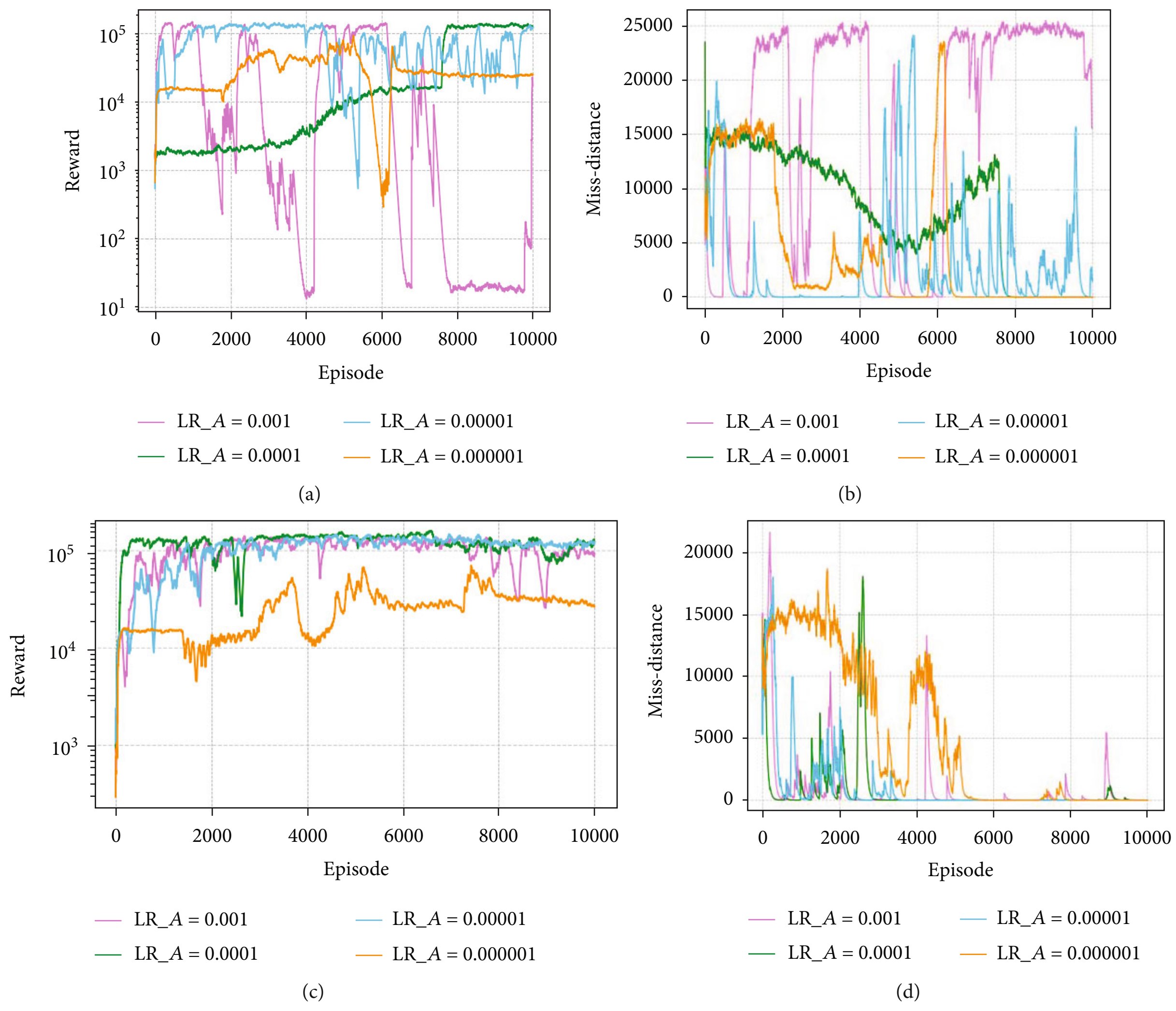


Figure 7: Comparison of convergence performance between the DDPG and TD3 model: (a) cumulative reward comparison of several different learning rates of DDPG, (b) miss-distance comparison of several different learning rates of DDPG, (c) cumulative reward comparison of several different learning rates of our guidance algorithm, (d) miss-distance comparison of several different learning rates of our guidance algorithm.

environment to get s_1 and R_1, thus get an experience like this: (s_0, a_0, s_1, R_1), and then put the experience in the replay buffer.

The significance of the existence of the replay buffer is to eliminate the correlation of experience because, in reinforcement learning, the adjacent action is usually strongly correlated, if we break it up and put it into the replay buffer, the neural network can be better trained when we randomly select a batch of experiences from the replay buffer to train it.

The actor-network update procedure is shown in Figure 4. The s_0 represents known items, we take out a batch of experience from the replay buffer; here is an experience: (s_0, a_0, s_1, R_0) as an example to describe the process of training a neural network.

According to the description of the actor network in Section 3.1, the loss function of the actor network is $-Q$, and the smaller the $-Q$ is, the better the performance. This $-Q$ needs to be obtained from the critic1 network, as shown in Figure 5.

Input the s_0 in the experience into the actor network to get the predicted action a_{0_predict} without noise, directly input s_0 and a_{0_predict} into the critic1 network to get the value of Q, and then use $-Q$ as a loss function to modify the actor network. The loss function equation is as follows:

$$\mathrm{L} = -Q\left(s, a_{0\_\text{predict}}\right). \tag{22}$$

As mentioned in Section 3.2, the network parameters ϕ of the actor network are updated through a deterministic gradient, so equation (19) becomes

$$\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s, a)\Big|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s). \tag{23}$$

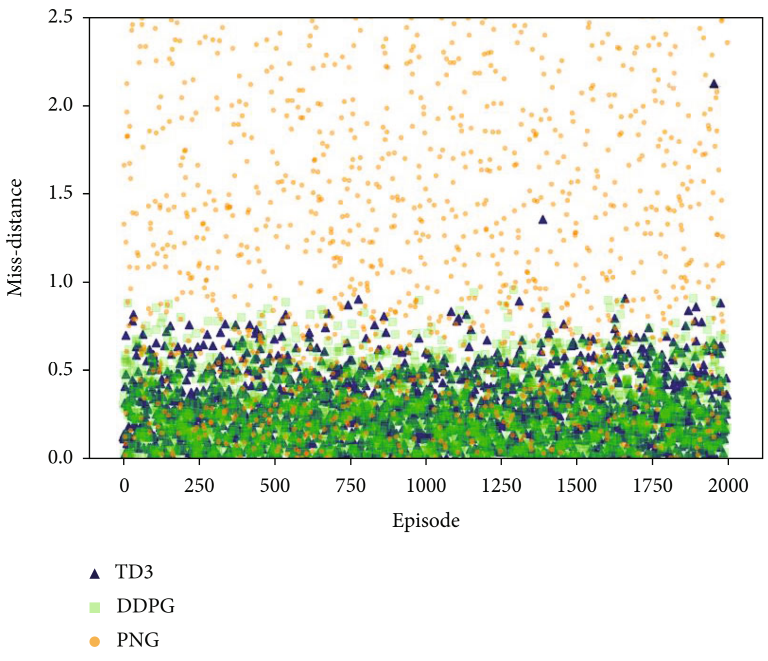


Figure 8: Miss-distance comparison.

As shown in Figure 5, s_0, s_1, a_0 represent known items. This section describes the critic network update procedure, which we still use experience: (s_0, a_0, s_1, R_1) as an example to describe the process of training the critic network.

As shown in Figure 5, the TD3 algorithm uses two target critic networks considering that the critic network always overestimates the Q value in actual applications. It borrows the idea of dueling double DQN (DDQN) [39] and adopts two networks to estimate the Q value and choose the smaller one to avoid overestimating the value as much as possible.

Because two target critic networks are used, the frequently updated critic network needs a corresponding number and is finally updated with the smaller of the two Q values:

$$y = r + \gamma \min \{Q_1, Q_2\}. \tag{24}$$

The above equation is used as the mean square error of $Q_1(s_0, a_0)$ and $Q_2(s_0, a_0)$, respectively. And then we use the mean square error as the loss function for gradient descent. The loss function is defined as follows:

$$L_{e=1,2} = N^{-1} \sum_{i=1}^{N} \left(y - Q_{\theta_{e=1,2}}(s, a)\right)^2. \tag{25}$$

By calculating equation (24), the gradient $\nabla_a Q_{\theta_{i=1,2}}(s, a)$ of Q value with respect to the parameter $\theta_{e=1,2}$ is obtained, which will be used in the calculation of equation (22).

In addition, note that the entire critic module actually trains the parameters of two critic neural networks. The parameters of the target actor network and two target critic networks are updated by the actor network and two critic networks (as shown in equation (21)), and noise N is added to the predicted action a_{1_predict} of the target actor network, which becomes action a_{1N} before being used as input to the two target critic networks, thus making the Q value of the next step more precise.

Table 4: Miss-distance comparison.

	TD3	DDPG	PNG
Number of effective hits	2000	2000	1986
Number of direct hits	1999	2000	993
Average miss distance (m)	0.27	0.28	2.58

The detailed pseudocode of TD3 is summarized in Algorithm 1.

4. Simulation and Analysis

In this section, we consider two target maneuvering modes: sinusoidal maneuvering and constant maneuvering, and then test them from three aspects: flight trajectory, normal acceleration, and LOS rate. Finally, we analyze the performance of the three guidance laws from the miss distance.

First, when the target is in sinusoidal maneuver, TD3 and DDPG algorithms were used to train the neural network with the parameters in Tables 1 and 2 for 10,000 episodes. The actor and critic functions are implemented by four-layer fully connected neural networks; critic1 and critic2 use the same network architectures, and the network architectures are as shown in Table 3. Except for the critic output layer, each neuron in other layers is activated by the Relu function, which is

Table 5: Test result.

	1th	2th	3th	4th	5th	6th
Distance (m)	32148	32857	33924	36696	30000	40000
LOS angle (°)	68	42	46	57	40	70
Missile's flight path angle (°)	61	35	47	50	30	80
Target's flight path angle (°)	162	178	166	157	180	180
PNG miss distance (m)	4019.92	15.92	1.37	5.43	16.45	11.64
DDPG miss distance (m)	0.0003	0.87	0.05	0.11	0.56	0.29
TD3 miss distance (m)	0.34	0.26	0.60	0.30	2.44	0.44

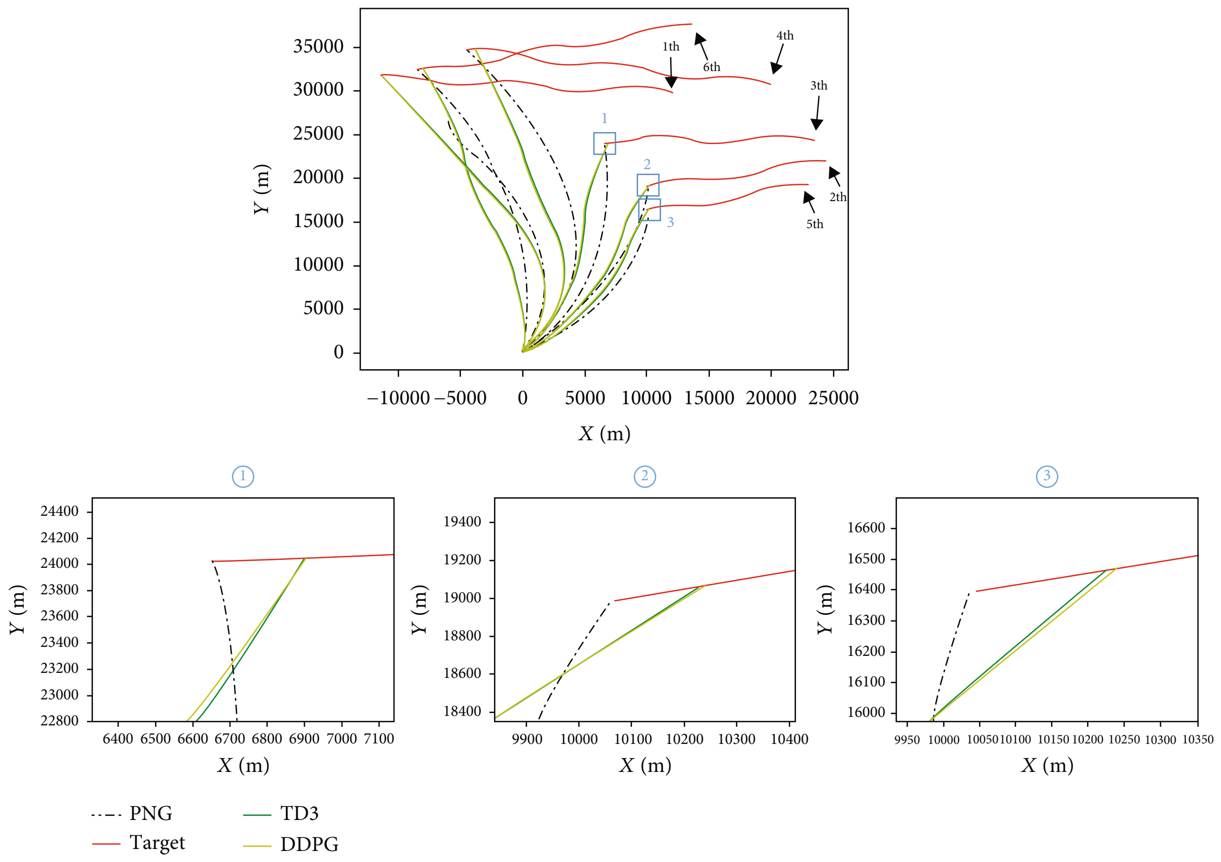


Figure 9: Ballistic trajectory.

$$\mathrm{Relu}(x) = \begin{cases} x, & \text{if } x \geq 0, \\ 0, & \text{if } x < 0, \end{cases} \tag{26}$$

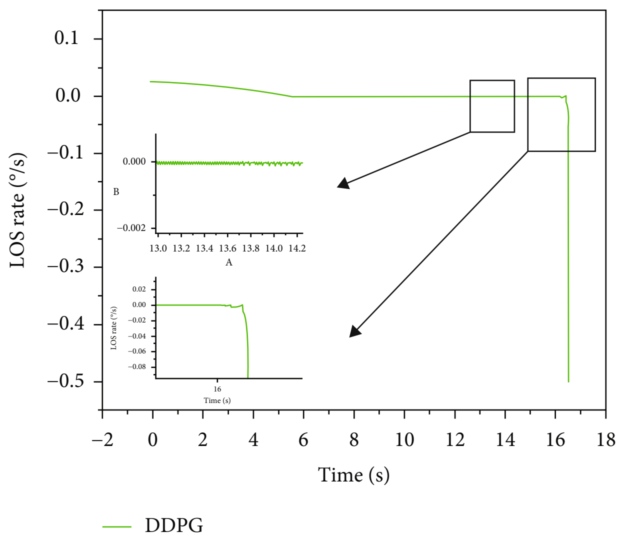
The output layer of the actor network is activated by the tanh function, which is defined as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{27}$$

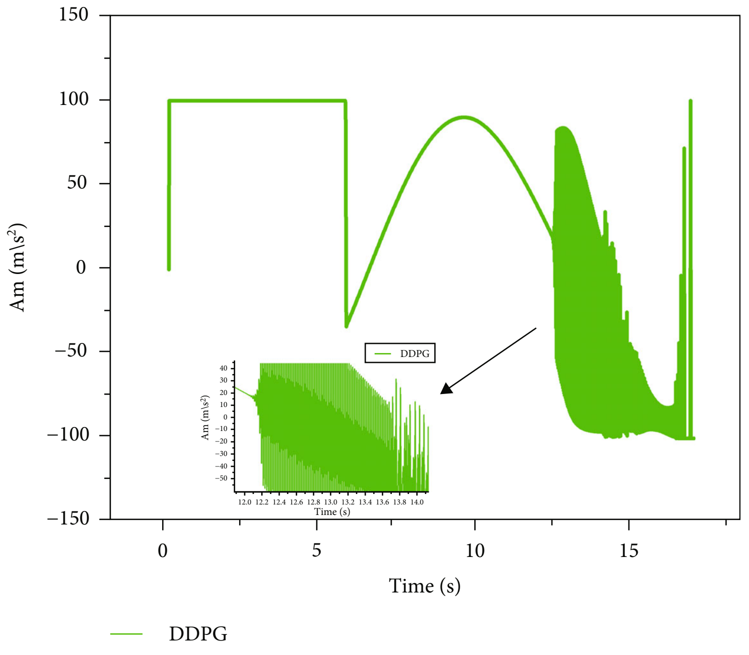
The policy and value functions are periodically updated during optimization after accumulating trajectory rollouts of replay buffer size.

As shown in Figure 6(a), the TD3 algorithm has converged when it is close to 4000 episodes, while the DDPG algorithm has converged when it is close to 8000 training rounds, and the reward obtained after TD3 is stable is higher than that of the DDPG algorithm. Figure 6(b) shows the variation trend of the final miss distance during the training of the two algorithms.

To test the convergence of the two algorithms, the agent is trained at several learning rates, as presented in Figure 7. The learning rate of the critic network in the two algorithms is twice that of the actor network. From Figure 7, one can note that our guidance algorithm is more stable than the DDPG guidance law; the DDPG algorithm cannot even converge at learning rates between 0.001 and 0.00001.



(a)



(b)

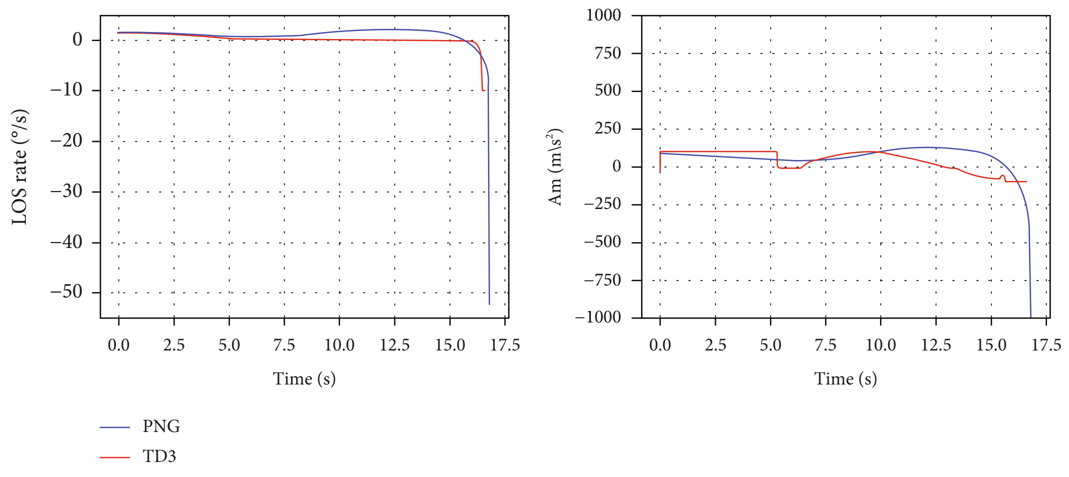
FIGURE 10: The normal acceleration.

When the trained TD3 algorithm is compared to the PNG, the two are randomly tested 2000 times. Because the LOS rate converges only when K is large enough, the PNG method takes the minimum value under the three conditions of the proportionality coefficient $K = 3, 4, 5$. We call the miss distance <10 m as an effective hit, and the miss distance <2 m as a direct hit.

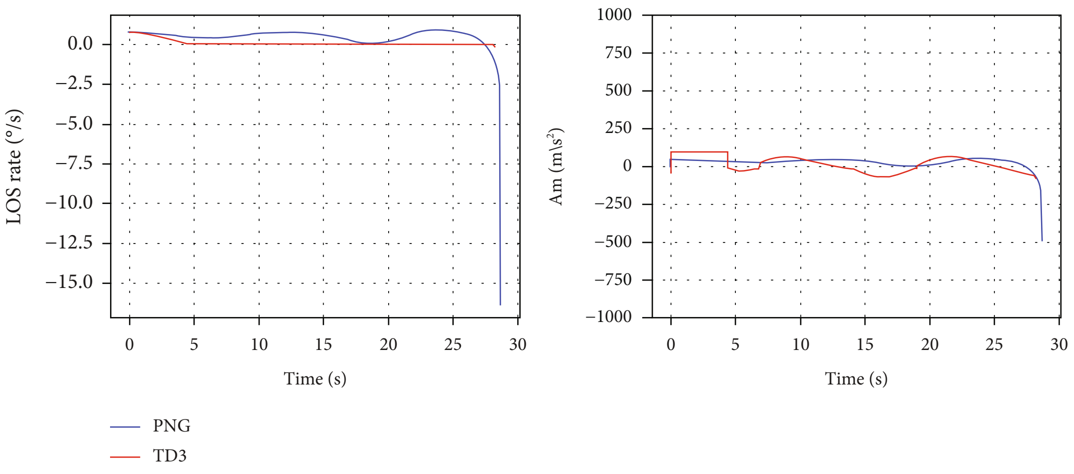
As shown in Figure 8 and Table 4, in 2000 tests, the RL guidance law based on the TD3 algorithm is much better than PNG in guidance accuracy, but there is not much difference between it and DDPG. In the number of miss distance less than 10 m, TD3 is slightly higher than PNG, but if the miss distance is reduced to less than 2 m, the gap is obvious, TD3 is almost 100% successful, but PNG is only 50%.

We take 4 of these tests and then take 2 boundary values for testing. The proportional coefficient of these 6 tests is $K = 3$, and the target performs sinusoidal maneuvers. Table 5 shows the recorded 6 initial dynamic model simulation parameters and the final miss distance comparison:

As can be seen from Figure 9, all 6 tests of the RL guidance law have a quick-hitting time compared with PNG. In the first test, it can be seen that PNG guidance accuracy has a large error. The second and fifth tests clearly show that the PNG algorithm did not hit the target of a locally enlarged image.



(a) The 5th test



(b) The 6th test

Figure 11: Comparison results of the 5th and 6th tests.

Finally, take the fifth and sixth times as examples to compare the LOS rate and normal acceleration. The changes in the LOS rate and normal acceleration of the RL guidance law based on the DDPG algorithm in the fifth test are shown in Figure 10. They have relatively unstable oscillations at the guidance end, and the oscillations will have a greater impact on the stability of the projectile. It can be seen that the RL-based guidance law implemented in the DDPG algorithm has not achieved a good training effect after ten thousand training times.

As shown in Figure 10(b), the robustness of the RL-based guidance law implemented in DDPG is poor, so it will not participate in the subsequent comparison of the LOS rate and the normal acceleration.

As shown in Figure 11, the LOS rate of the two algorithms is generally stable in the early stage. However, as mentioned in Section 2.1, PNG's LOS rate fails to converge in the final trajectory stage when the target presents non-straight motion, and the RL guidance law based on the TD3 algorithm has better convergence than PNG.

The PNG algorithm follows the change rule of equation (2), so the normal acceleration diverges at the last moment, and the normal overload required at the terminal stage is too large. In contrast to the reinforcement learning algorithm, although the curve is not as smooth as the normal acceleration, it does not oscillate as violently as Figure 10(b), and its normal acceleration is better than PNG in terms of terminal convergence, which is more realistic.

In addition, we also tried the second scenario to see if it has a good generalization to the engagement scenarios we did not experience in the training process. As a result, the fifth and sixth initial state parameters in Table 5 were left unchanged, while the target maneuvering mode was changed to constant maneuvering $20\,\mathrm{m/s^2}$.

As shown in Figures 12 and 13, when the target maneuvering mode is changed to continual maneuvering, the performance of the two guidance laws improves, but the guidance law still outperforms PNG in the terminal stage, and its accuracy is still higher, as presented in Table 6. This demonstrates that the guidance law has a high degree of generalization.

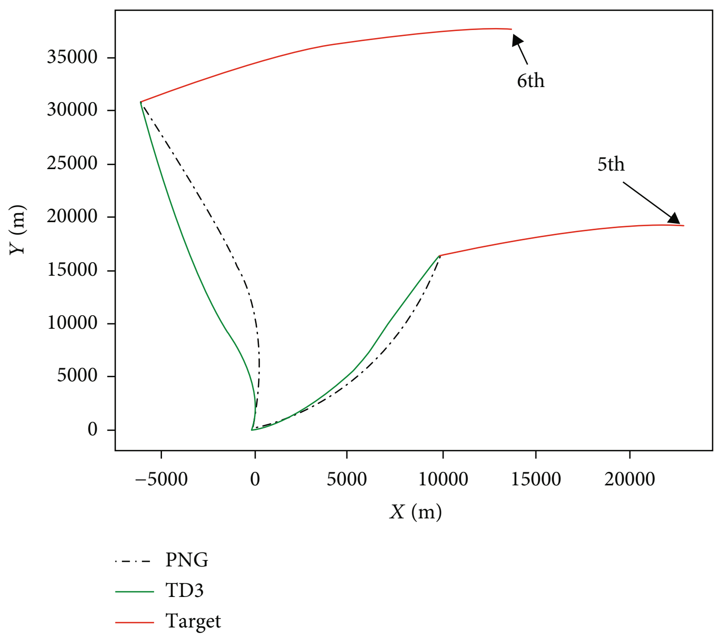
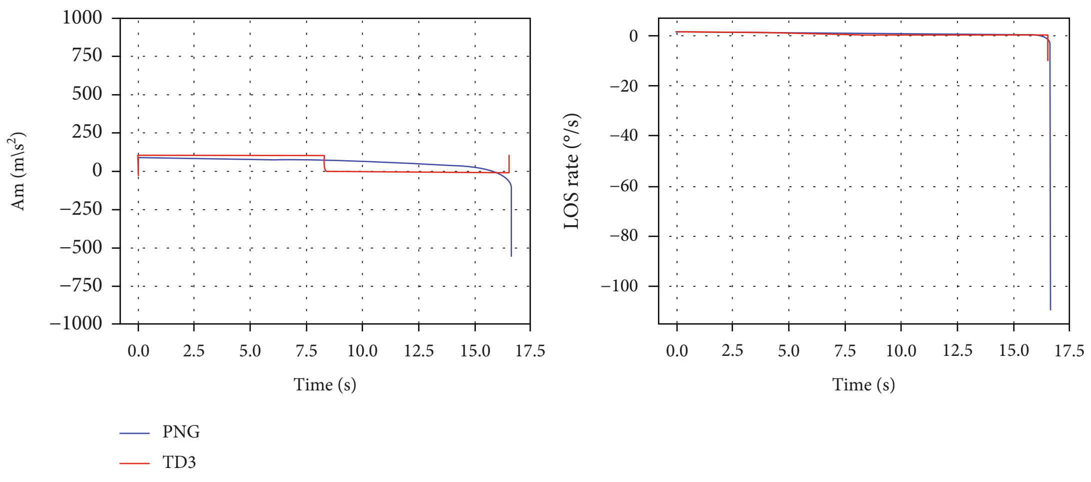
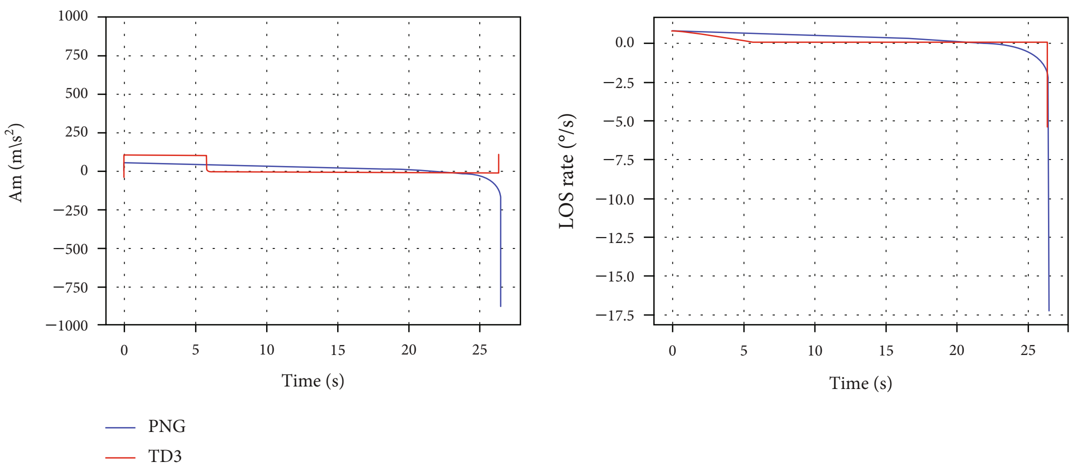


Figure 12: Ballistic trajectory.



(a) The 5th test



(b) The 6th test

Figure 13: Comparison results of constant maneuver.

Table 6: The constant maneuvering miss-distance comparison.

	5th	6th
PNG miss distance (m)	2.79	15.29
TD3 miss distance (m)	0.01	0.50

5. Conclusion

We have demonstrated that when facing a maneuvering target, the performance of the resulting guidance strategy is better than that of the traditional proportional guidance method by directly mapping the LOS rate to the overload instruction via a neural network. The simulation results show that (1) RL guidance law based on TD3 has better accuracy and convergence of normal acceleration compared with PNG; (2) when designing guidance law based on RL algorithm framework, the TD3 algorithm outperforms DDPG in terms of robustness; and (3) the guidance law also has a good generalization to new engagement scenarios that have not been experienced during training. Furthermore, because the RL algorithm is a model-free framework, it can optimize guidance laws in more complicated environmental frameworks. Therefore, we believe that designing guidance law based on the RL framework will be an effective way to intercept maneuvering targets in the future. The following work may consider RL guidance law performance in more realistic combat environments that contain acceleration of gravity, aerodynamic forces, thrust, and air humidity.

Nomenclature

MPD:	Markov decision process
RL:	Reinforcement learning
s:	State vector in the MDP
a:	Action vector in the MDP
θ:	The parameter of the critic network
ϕ:	The parameter of the actor network
$R(s_t, a_t)$:	Reward for being in state s when selecting the corresponding action a at the time t
G_t:	The cumulative sum of subsequent rewards after the time point t
$p(x)$:	The probability density p associated with the random variable x
$\mathrm{E}[f(x)]$:	Expectation of $f(x)$, i.e., $\int_{-\infty}^{+\infty} p(x)f(x)dx$
$Q^{\pi}(s_t, a_t)$:	The action value for agent taking action a by following the policy π in the state s at the time t.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work has been supported by the National Natural Science Foundation of China (Grant No. 1147213611402117).

References

[1] J. Moon, K. Kim, and Y. Kim, "Design of missile guidance law via variable structure control," *Journal of Guidance, Control, and Dynamics*, vol. 24, no. 4, 2001.

[2] S. R. Kumar, S. Rao, and D. Ghose, "Sliding-mode guidance and control for all-aspect interceptors with terminal angle constraints," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1230–1246, 2012.

[3] B. Zhang and Z. Di, "Optimal predictive sliding-mode guidance law for intercepting near-space hypersonic maneuvering target," *Chinese Journal of Aeronautics*, vol. 35, no. 4, pp. 320–331, 2022.

[4] Z. Zhang, C. Man, S. Li, and S. Jin, "Finite-time guidance Laws for three-dimensional missile-target interception, proceedings of the Institute of Mechanical Engineers," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 230, no. 2, 2016.

[5] Y. J. Si and S. M. Song, "Adaptive reaching law based three-dimensional finite-time guidance law against maneuvering targets with input saturation," *Aerospace Science and Technology*, vol. 70, pp. 198–210, 2017.

[6] D.-Y. Lee, Y.-W. Kim, C.-H. Lee, and M.-J. Tahk, "Impact angle control guidance laws using finite-time-convergence control methods," in *29th Mediterranean Conference on Control and Automation (MED)*, pp. 997–1002, Puglia, Italy, 2021.

[7] M. A. Hossain, A. A. M. Madkour, K. P. Dahal, and L. Zhang, "A real-time dynamic optimal guidance scheme using a general regression neural network," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1230–1236, 2013.

[8] S. Kasmaiee, M. Tadjfar, and S. Kasmaiee, "Optimization of blowing jet performance on wind turbine airfoil under dynamic stall conditions using active machine learning and computational intelligence," *Arabian Journal for Science and Engineering*, vol. 2, 2023.

[9] M. Tadjfar, S. Kasmaiee, and S. Noori, "Continuous blowing jet flow control optimization in dynamic stall of NACA0012 airfoil," in *ASME 2020 Fluids Engineering Division Summer Meeting*, p. 8, 2020.

[10] M. Tadjfar, S. Kasmaiee, and S. Noori, "Optimization of NACA 0012 airfoil performance in dynamics stall using continuous suction jet," in *ASME 2020 Fluids Engineering Division Summer Meeting*, p. 9, 2020.

[11] S. Kasmaiee and M. Tadjfar, "Experimental study of the injection angle impact on the column waves: wavelength, frequency and drop size," *Experimental Thermal and Fluid Science*, vol. 148, article 110989, 2023.

[12] S. Kasmaiee and M. Tadjfar, "Influence of injection angle on liquid jet in crossflow," *International Journal of Multiphase Flow*, vol. 153, article 104128, 2022.

[13] H. Wang, Z. Y. Guan, L. J. Yu, and B. B. Zhang, "Research on fuzzy guidance law for unmanned aerial vehicle," in *Proceedings of the 2013 International Conference on Advanced Computer Science and Electronics Information*, p. 41, 2013.

[14] Z. Li, Y. Xia, C. Y. Su, J. Deng, J. Fu, and W. He, "Missile guidance law based on robust model predictive control using neural-network optimization," *IEEE Transactions on Neural*

Networks and Learning Systems, vol. 1, no. 99, pp. 1803–1809, 2015.

[15] O. Tutsoy, "COVID-19 epidemic and opening of the schools: artificial intelligence-based long-term adaptive policy making to control the pandemic diseases," *IEEE Access*, vol. 9, pp. 68461–68471, 2021.

[16] Y. L. Yang, H. Modares, K. G. Vamvoudakis, W. He, C. Z. Xu, and D. C. Wunsch, "Hamiltonian-driven adaptive dynamic programming with approximation errors," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13762–13773, 2022.

[17] Y. L. Yang, Y. Pan, C. Z. Xu, and D. C. Wunsch, "Hamiltonian-driven adaptive dynamic programming with efficient experience replay," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022.

[18] Y. Yang, B. Kiumarsi, H. Modares, and C. Xu, "Model-free λ-policy iteration for discrete-time linear quadratic regulation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 34, no. 2, pp. 635–649, 2023.

[19] F. Kamrani and R. Ayani, "Using on-line simulation for adaptive path planning of UAVs," in *11th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT'07)*, pp. 1550–6525, Chania, Greece, 2007.

[20] G. Lei, M. Z. Dong, T. Xu, and L. Wang, "Multi-agent path planning for unmanned aerial vehicle based on threats analysis," in *2011 3rd International Workshop on Intelligent Systems and Applications*, pp. 1–4, Wuhan, China, 2011.

[21] P. Rui, "Multi-UAV formation maneuvering control based on Q-learning fuzzy controller," in *2010 2nd International Conference on Advanced Computer Control*, pp. 252–257, Shenyang, China, 2010.

[22] C. Chithapuram, Y. Jeppu, and C. A. Kumar, "Artificial intelligence learning based on proportional navigation guidance," in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, pp. 1140–1145, Mysore, India, 2013.

[23] C. U. Chithapuram, A. K. Cherukuri, and Y. V. Jeppu, "Aerial vehicle guidance based on passive machine learning technique," *International Journal of Intelligent Computing and Cybernetics*, vol. 9, no. 3, pp. 255–273, 2016.

[24] Q. H. Zhang, B. Q. Ao, and Q. X. Zhang, "Reinforcement learning guidance law of Q-learning," *Systems Engineering and Electronics*, vol. 2, no. 42, pp. 414–419, 2020.

[25] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] K. Fang, Q. Z. Zhang, K. Ni, and L. Cui, "Reentry guidance law with flight time constraint," *Journal of Harbin Institute of Technology*, vol. 51, no. 10, pp. 90–97, 2019.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017, https://arxiv.org/abs/1707.06347.

[28] B. Gaudet, R. Furfaro, and R. Linares, "Reinforcement learning for angle-only intercept guidance of maneuvering targets," *Aerospace Science and Technology*, vol. 99, article 105746, 2020.

[29] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., "Continuous control with deep reinforcement learning," 2019, https://arxiv.org/abs/1509.02971.

[30] B. Li, Z. P. Yang, D. Q. Chen, S. Y. Liang, and H. Ma, "Maneuvering target tracking of UAV based on MN-DDPG and transfer learning," *Defence Technology*, vol. 17, no. 2, pp. 457–466, 2021.

[31] S. He, H. Shin, and A. Tsourdos, "Computational missile guidance: a deep reinforcement learning approach," *Journal of Aerospace Information Systems*, vol. 18, no. 8, pp. 571–582, 2021.

[32] D. Hong, M. Kim, and S. Park, "Study on reinforcement learning-based missile guidance law," *Applied Sciences*, vol. 10, no. 18, p. 6567, 2020.

[33] C. Li, B. Deng, and T. Zhang, "Terminal guidance law of small anti-ship missile based on DDPG," in *2020 International Conference on Image, Video Processing and Artificial Intelligence*, pp. 450–456, 2020.

[34] Y. Liu, Z. Z. He, C. Y. Wang, and M. Z. Guo, "Terminal guidance law design based on DDPG algorithm," *Chinese Journal of Computers*, vol. 44, no. 9, pp. 1854–1865, 2021.

[35] M. J. Du, C. Peng, and J. J. Ma, "Deep reinforcement learning based missile guidance law design for maneuvering target interception," in *40th Chinese Control Conference (CCC)*, pp. 3733–3738, 2021.

[36] S. Fujimoto, H. V. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," 2018, https://arxiv.org/abs/1802.09477.

[37] S. Zhang, Y. Li, and Q. Dong, "Autonomous navigation of UAV in multi-obstacle environments based on a deep reinforcement learning approach," *Applied Soft Computing*, vol. 115, article 108194, 2022.

[38] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *The 31st International Conference on Machine Learning*, pp. 387–395, 2014.

[39] H. V. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," 2015, https://arxiv.org/abs/1509.06461.