Hindawi
International Journal of Aerospace Engineering
Volume 2022, Article ID 2935929, 18 pages
https://doi.org/10.1155/2022/2935929

Research Article

Reinforcement Learning for Computational Guidance of Launch Vehicle Upper Stage

Shiyao Li,[1] Yushen Yan,[1] Hao Qiao,[2] Xin Guan,[1] and Xinguo Li[1,3]

[1]*School of Astronautics, Northwestern Polytechnical University, Xi'an 710072, China*
[2]*Xi'an Modern Control Technology Research Institute, Xi'an 710065, China*
[3]*National Key Laboratory of Aerospace Flight Dynamics, Northwestern Polytechnical University, Xi'an 710072, China*

Correspondence should be addressed to Shiyao Li; lishiyao@mail.nwpu.edu.cn

Received 9 March 2022; Revised 5 May 2022; Accepted 25 May 2022; Published 7 June 2022

Academic Editor: Franco Bernelli-Zazzera

This manuscript investigates the use of a reinforcement learning method for the guidance of launch vehicles and a computational guidance algorithm based on a deep neural network (DNN). Computational guidance algorithms can deal with emergencies during flight and improve the success rate of missions, and most of the current computational guidance algorithms are based on optimal control, whose calculation efficiency cannot be guaranteed. However, guidance-based DNN has high computational efficiency. A reward function that satisfies the flight process and terminal constraints is designed, then the mapping from state to control is trained by the state-of-the-art proximal policy optimization algorithm. The results of the proposed algorithm are compared with results obtained by the guidance-based optimal control, showing the effectiveness of the proposed algorithm. In addition, an engine failure numerical experiment is designed in this manuscript, demonstrating that the proposed algorithm can guide the launch vehicle to a feasible rescue orbit.

1. Introduction

This manuscript studies the computational guidance algorithm based on DNN. Lu [1] proposed the concept of "computational guidance and control," in which the generation of guidance and control commands relies extensively on onboard computation and does not require a specified reference trajectory.

So far, most of the research on computational guidance is based on the optimal trajectory planning problem. The primary aim of the trajectory planning algorithm is to solve the optimal control problem (OCP), which is generally based on nonlinear dynamics and achieves specific performance indicators under the constraints of state and control variables. The solution to the problem is mainly achieved using indirect [2–4] and direct [5–7] methods. The indirect method solves the optimal control problem by using the classical variational method and Pontryagin's minimum principle to derive the necessary first-order conditions of the optimal control and transform the problem into a two-point boundary value problem (TPBVP) [8]. However, the convergence of the numerical iteration is extremely sensitive to the initial value, and the TPBVP is difficult to solve. Therefore, the indirect method cannot be directly applied to launch vehicles' guidance systems without simplification.

The direct method transforms the optimal control problem of continuous space into a nonlinear programming problem and uses a numerical method to directly optimize the performance index [9–11]. In 2007, JPL proposed lossless convexity technology for dynamic descent guidance of the Mars lander [12]. After that, a systematic summary of the research and development of lossless convexity technology was presented in [13]. Unfortunately, only a few nonconvex constraints can be used for lossless convexification. For the problem that the lossless convexification technique cannot be used, a sequential convexification method was proposed. But this method was based on the linearization technique, which required multiple iterations, and was also sensitive to the initial value. Nevertheless, considering the rapidity of the convex optimization algorithm in solving convex problems, in recent years, trajectory planning based on this algorithm, such as planetary landing [14], rocket ascent guidance [15], and entry guidance [16], has been widely studied.

In recent years, with the application of machine learning methods in various fields, researchers in the aerospace field also began to pay attention to machine learning, especially deep learning and reinforcement learning. DNNs are among the most versatile and powerful machine learning tools, thanks to their unique capability of accurately approximating complex nonlinear input-output functions when provided with a sufficiently large amount of data consisting of sample input-output pairs (i.e., a training set) [17]. The term "G&CNet" (namely, guidance, and control network) was coined by the European Space Agency [18] to refer to an onboard system that provides real-time guidance and control functionalities to the spacecraft by means of a DNN, which replaced the traditional control and guidance architectures [19]. Aimed at dealing with the sensitive problem of the initial value guess of the indirect method, a method was proposed in [20] to obtain a good initial guess through the DNN, and the numerical experimental results showed that this improved the computational speed of the indirect method. Carlos and Dario [21] directly applied the deep learning method to the optimal control problem, and the numerical experimental results showed that the trajectory obtained by the DNN architecture was close to the optimal one. This work opened up the possibility of using a DNN to directly drive the state-action selection. To solve the 2D trajectory optimization problem of a hypersonic vehicle, the authors of [22] proposed a DNN architecture. The idea in [22] was similar to that in [21], where the DNN was used to obtain the mapping relationship between state and control. Compared with the traditional optimal control problem, the DNN can ensure real-time performance of the algorithm. A fast approach to generating time-optimal asteroid landing trajectories was presented in [23], and a DNN was developed to approximate the gravitational field of asteroids, and the corresponding time consumption of gravity calculation in trajectory propagation was significantly reduced.

The above methods use the supervised learning (SL) method to train the DNN. However, the SL method needs large expert samples like state-control pairs, which are obtained by solving the OCP. But obtaining expert samples creates a heavy computational load to construct a dataset for training. Another approach to training DNN is called reinforcement learning (RL). RL does not require prior computation for generating the expert samples. In RL, samples are collected from the interaction between agent and environment, and the agent understands and improves the current performance through the reward obtained by interaction. Therefore, the reward function is the key; researchers may never get the ideal results if the reward function is not designed well. In [24], the performance of behavioral cloning (BC) and RL was investigated on a linear multi-impulsive rendezvous mission. An interactive deep reinforcement learning (DRL) algorithm with an actor-indirect method architecture was presented in [25] to train the DNN-driven controller for optimal control of the landing problem. In [26], the authors applied reinforcement learning to a Mars landing guidance system to directly generate guidance commands. In [27, 28], the authors applied the RL meta-learning framework to optimize an integrated and adaptive guidance and control system for exoatmospheric and endoatmospheric interception problems, and the numerical results showed the system was robust to the parasitic attitude loop. In [29–31], the authors used RL metalearning framework in the vehicle landing problems with distributions like sensor noise and actuator failure, and the numerical results showed that RL metalearning could deal with these distributions well and get good results. A robust trajectory design method based on reinforcement learning was proposed in [19], and the experimental results showed that good results could be obtained through different models. In [32], the image-based reinforcement metalearning was applied to solve the lunar landing task with uncertain dynamic parameters, and the numerical results showed that the resulting closed-loop guidance policy was effective even if the environment was partially observed. The image-based reinforcement metalearning was also used in the autonomous guidance of an impactor in a binary asteroid system, and the numerical results showed that the guidance system was robust and could be applied to almost all test scenarios [33].

Once a neural network is trained, it only needs to do simple matrix multiplication when in use. Compared with guidance-based optimal control which requires solving the optimal control problem, the calculation time can be ignored. Thus, the method based on machine learning has real-time performance. Aimed at the guidance of the launch vehicle ascending phase, a guidance algorithm based on reinforcement learning is proposed in this manuscript. In Section 2, the background of reinforcement learning is introduced, and in Section 3, the guidance-based reinforcement learning framework is proposed, combined with a dynamics equation. Section 4 presents the experimental results and a discussion.

2. Reinforcement Learning

2.1. Markov Decision Process. The Markov decision process (MDP) is a mathematical model of a sequential decision problem. In an environment where the system state has a Markov nature, it is used to simulate possible random strategies and rewards of agents. The complete MDP is usually described by $(\mathbf{s}, \mathbf{a}, R, P)$, where $\mathbf{s}$ represents the state set $\{s_0, s_1, \cdots, s_n\}$, $\mathbf{a}$ represents the action set $\{a_0, a_1, \cdots, a_n\}$, R represents the scalar reward, and P represents the state transition probability of the environment $P(\mathbf{s}, \mathbf{a}, \mathbf{s}')$. In reinforcement learning, the agent is the learner and decision-maker of the whole system, and the state is the description of environmental information; the action is the agent's response to the environment, and the reward is the evaluation of action by the environment. The agent observes the environment and selects the appropriate action according to the obtained state information. The environment receives the action, makes corresponding feedback, and enters a new state. The agent obtains the reward from the environment and adjusts the next action.

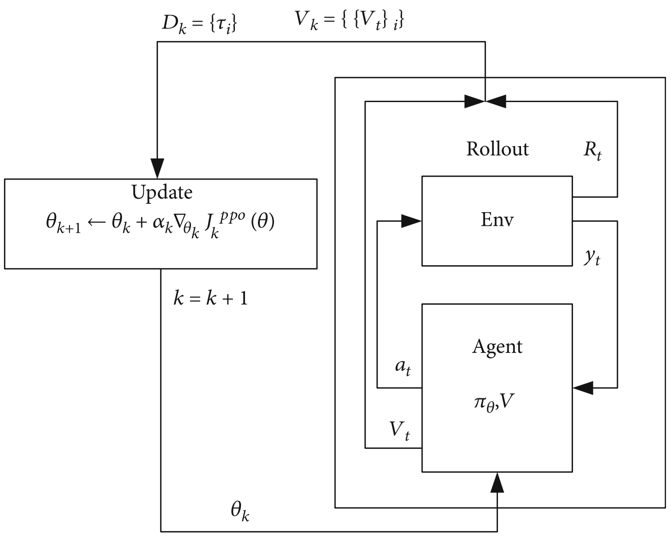

Figure 1: Training process by PPO [46]

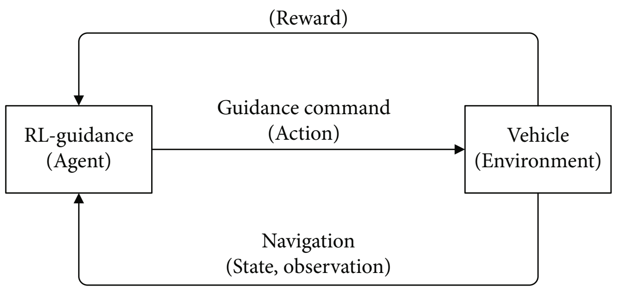

Figure 2: Guidance-based RL process.

The agent and environment interact at each time step. This mapping from state to action is called the policy, which is expressed as:

$$\pi(a|s) = P[\mathbf{a}_t = a|\mathbf{s}_t = s]. \tag{1}$$

The goal of reinforcement learning is to find the optimal action policy. The more positive feedback an agent receives in the learning process, the better the policy it learns. Therefore, the weighted cumulative sum of the reward value of each step overtime is defined as the return, which is expressed as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \tag{2}$$

where γ represents the discount factor.

By maximizing the long-term return G_t, the corresponding best action policy can be obtained. To describe the long-term value when executing the policy at the state s, the expectation of return at this time is defined as the state-value function:

$$V_\pi(s) = \mathbb{E}_\pi[G_t \mathbf{s}_t| = s]. \tag{3}$$

To measure the value of executing action a at the state s, the action-value function can be defined as:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|\mathbf{s}_t = s, \mathbf{a}_t = a]. \tag{4}$$

According to the Bellman equation, the value function can be decomposed as follows:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi\left[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots |\mathbf{s}_t = s\right] \\ &= \mathbb{E}_\pi[R_{t+1} + \gamma V_\pi(\mathbf{s}_{t+1})|\mathbf{s}_t = s]. \end{aligned} \tag{5}$$

Similarly, the Bellman equation form of the action-value function can be obtained:

$$Q_\pi(s, a) = \mathbb{E}_\pi[R_{t+1} + \gamma Q_\pi(s_{t+1}, a_{t+1})|\mathbf{s}_t = s, \mathbf{a}_t = a]. \tag{6}$$

According to Bellman's principle of optimality, if the value function is the max, the corresponding policy is the optimal policy. Therefore, the Bellman equations of the optimal state-value and action-value functions can be expressed as:

$$\begin{aligned} V^*(s) &= \max_a R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a V^*\left(s'\right), \\ Q^*(s, a) &= R_s^a + \gamma \sum_{s'} P_{ss'}^a \max_{a'} Q^*\left(s', a'\right). \end{aligned} \tag{7}$$

According to whether the environment model (state transition probability) is known or not, reinforcement learning can be divided into model-based and model-free methods. Generally speaking, because the model-free method does not make full use of the empirical knowledge obtained in learning, the convergence speed is slower than in the model-based method. However, the model-free method is one of the most important learning techniques

Table 1: Parameters of boundary conditions.

Parameter	Value
Min initial position (m)	(371973.739, 114644.849, -13899.978)
Max initial position (m)	(373973.739, 116644.849, -10899.978)
Min initial velocity (m/s)	(3652.033, 556.843, -32.666)
Max initial velocity (m/s)	(3852.033, 756.843, -12.666)
Initial pitch and yaw angle range (°)	[50,60], [-20,-10]
Terminal position (m)	(1912866.558, -73986.648, 2551.256)
Terminal velocity (m/s)	(7457.930, -2220.619, 178.661)

Table 2: Hyperparameters.

Parameter	Value	Parameter	Value
Shape reward coefficient	0.01	Epochs per update	30
Final reward coefficient	1000	Episodes per rollouts	50
Constant positive reward	0.001	Number of iterations	10000
Discount factor	0.995	Total episode	500000
GAE factor	0.98		

in reinforcement learning because of its small amount of calculation per iteration and good adaptability to dynamic unknown environments.

2.2. Policy Gradient Method. Reinforcement learning algorithms can be divided into value function and policy gradient-based according to the optimization objectives. The algorithm based on the value function finds the optimal policy by maximizing the state-value function or action-value function, such as Q-learning [34] and sarsa [35]. The algorithm based on policy gradient parameterizes the policy using a nonlinear function and maximizes the cumulative reward by directly iterating the policy, such as policy gradient (PG) [36] and REINFORCE [37].

In 2015, Mnih et al. [38] first proposed the deep Q network (DQN), which achieved end-to-end learning by introducing an experience replay mechanism and constructing an independent target network. DQN was directly learned from high-dimensional perceptual input to a successful policy, and the algorithm was applied to Atari games with great success. However, there were still some unavoidable problems, such as overestimation of Q value, low sample utilization, and poor learning stability. In 2016, Hasselt et al. designed a double Q network structure [39], which was responsible for selecting and evaluating actions through two Q networks; double DQN effectively avoided the overestimation phenomenon caused by the greedy strategy in the DQN algorithm and had better performance. Schaul et al. proposed a DQN algorithm based on the priority experience replay mechanism, which used priority sampling instead of uniform sampling, and improved the convergence speed by increasing the frequency of resampling in the important transition process [40]. Wang et al. proposed a dueling DQN algorithm [41], which separately handled the evaluation of states and actions on two branches of a network, and finally combined them on the output layer for Q-value estimation, which could obtain a better evaluation policy than the traditional DQN. Aiming at the problem of partially observable scenes, Hausknecht and Stone proposed a deep recurrent Q network [42] algorithm, which used long-short-term memory (LSTM) in the DQN structure and could be applied in partially observable scenes. DeepMind proposed the rainbow algorithm [43] in 2017, which integrated six DQN-based methods, including double DQN and dueling DQN, and could achieve better results than any one of them. The algorithms mentioned above are optimized from different perspectives, and the requirements for discrete action spaces are not changed.

In the reinforcement learning task of continuous action space, to obtain the value function, the continuous action space needs to be discretized, which will cause an action dimension disaster. Moreover, the value function iteration method usually uses a greedy strategy to update the value function, which will make the agent learn a fixed policy. To solve the above problems, a policy-based method was proposed, which estimated the gradient of the objective function relative to the policy parameters, then used the gradient ascent algorithm to optimize the parameters and finally obtained the optimal policy. The approximate expression of the policy function can be written as:

$$\pi_\theta(a|s) = P\{\mathbf{a}_t = a|\mathbf{s}_t = s, \theta_t = \theta\}, \tag{8}$$

where θ represents the parameter of the policy, to solve this parameter, the expectation of the agent about the reward $J(\theta)$ is introduced as the objective function, and the following expression is used to update:

$$\theta_{t+1} = \theta_t + \alpha \nabla \widehat{J}(\theta_t), \tag{9}$$

where α represents the learning rate and $\nabla \widehat{J}(\theta_t)$ is the gradient value of the objective function.

According the policy gradient theory, $\nabla \widehat{J}(\theta_t)$ is rewritten as:

$$\nabla_\theta J(\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{k=0}^{N-1} \nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta,t}(s,a)\right], \tag{10}$$

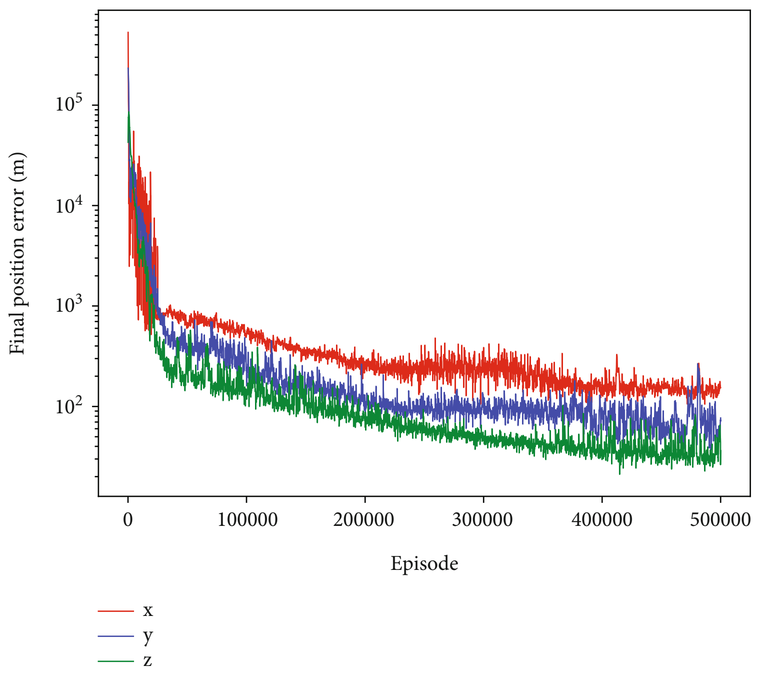

Figure 3: Learning curves for final position error.

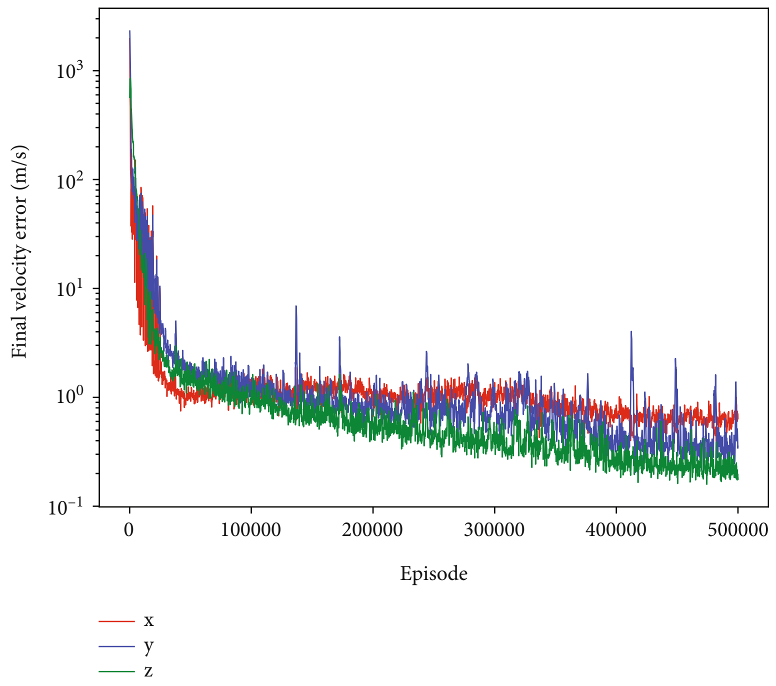

Figure 4: Learning curves for final velocity error.

where $Q_{\pi_\theta,k}(s,a)$ is the action-value function, and the expression is:

$$Q_{\pi_\theta,t}(s,a) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{t'=t}^{N-1}\gamma^{t'-t}R_{t'}|\mathbf{s}_t = s, \mathbf{a}_t = a\right]. \tag{11}$$

$Q_{\pi_\theta,k}(s,a)$ can be estimated without bias by the Monte Carlo method. Although this will reduce the deviation from the target value, it will also make a large variance and affect the convergence speed of the algorithm.

To solve the above problems, an actor-critic method was proposed. The actor is responsible for updating the policy gradient and executing the actions calculated by the policy.

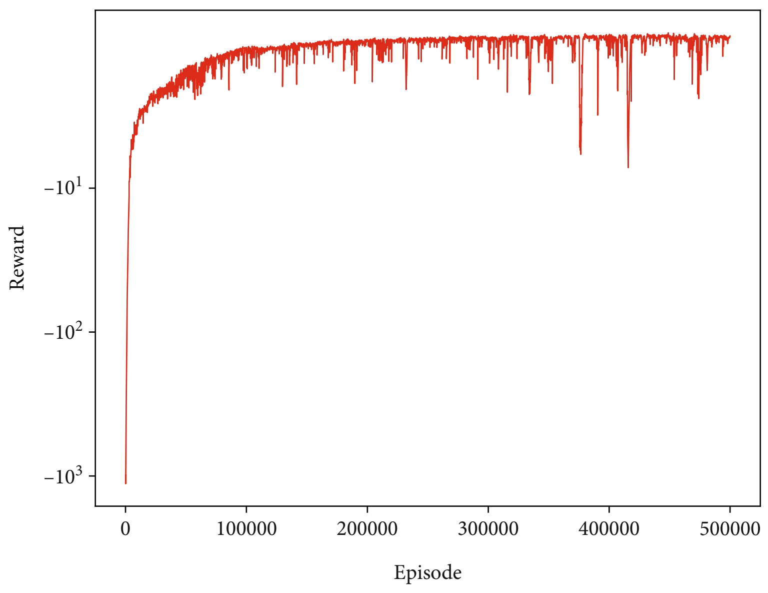

Figure 5: Learning curves of reward.

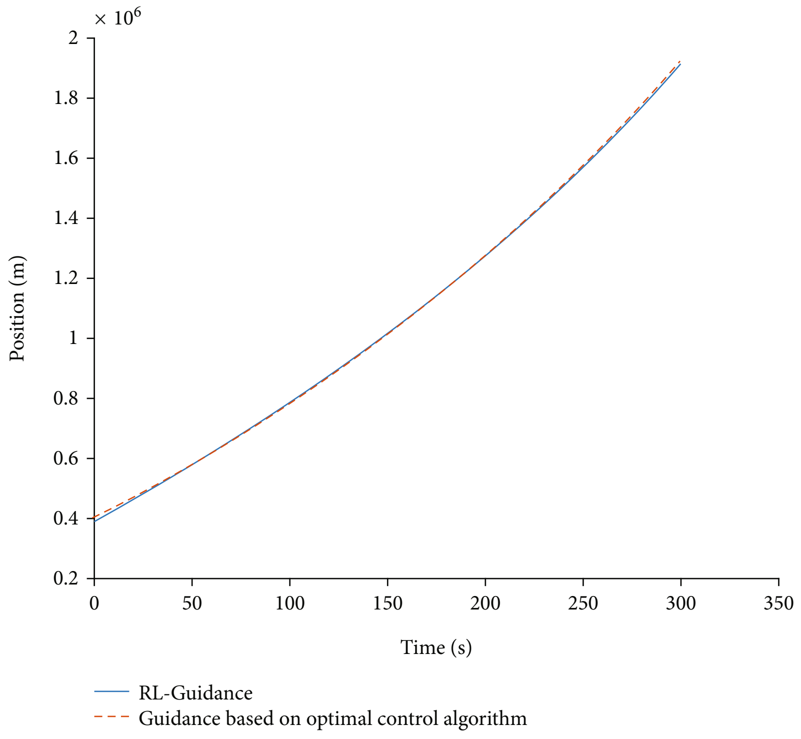

Figure 6: Position comparison of RL-guidance and guidance-based optimal control.

The critic is responsible for scoring the actor through the evaluation mechanism and then feeding the score back to the actor to guide it to update the policy gradient.

Trust region policy optimization (TRPO) [44], proposed by Schulman et al., is a kind of actor-critic method. According to this method, the gradient of the reward objective function can be transformed into the following expression:

$$\nabla_\theta J(\theta) = \mathbb{E}\left[\sum_{t=0}^{\infty} \Psi \nabla_\theta \log \pi_\theta(a|s)\right]. \quad (12)$$

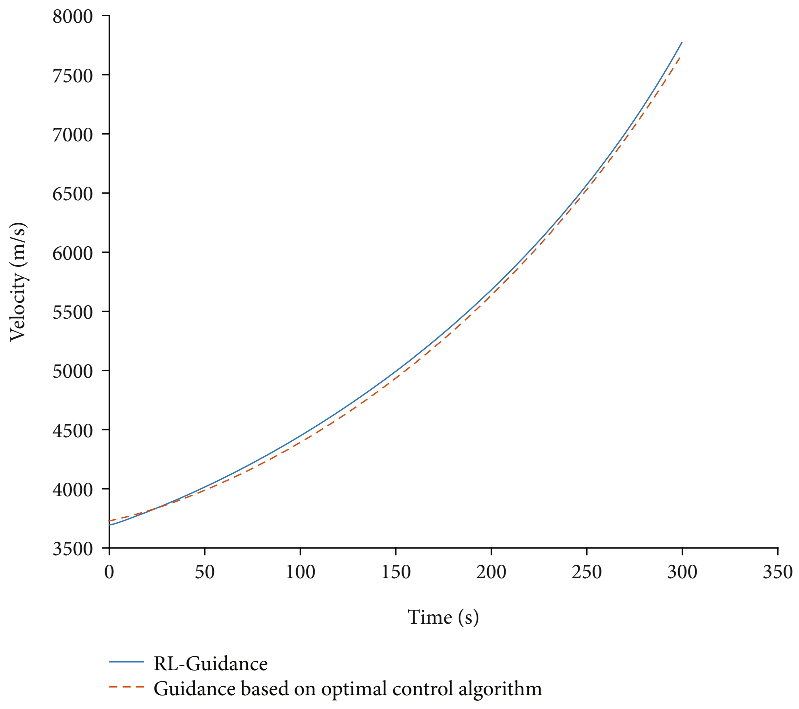

Figure 7: Velocity comparison of RL-guidance and guidance-based optimal control.

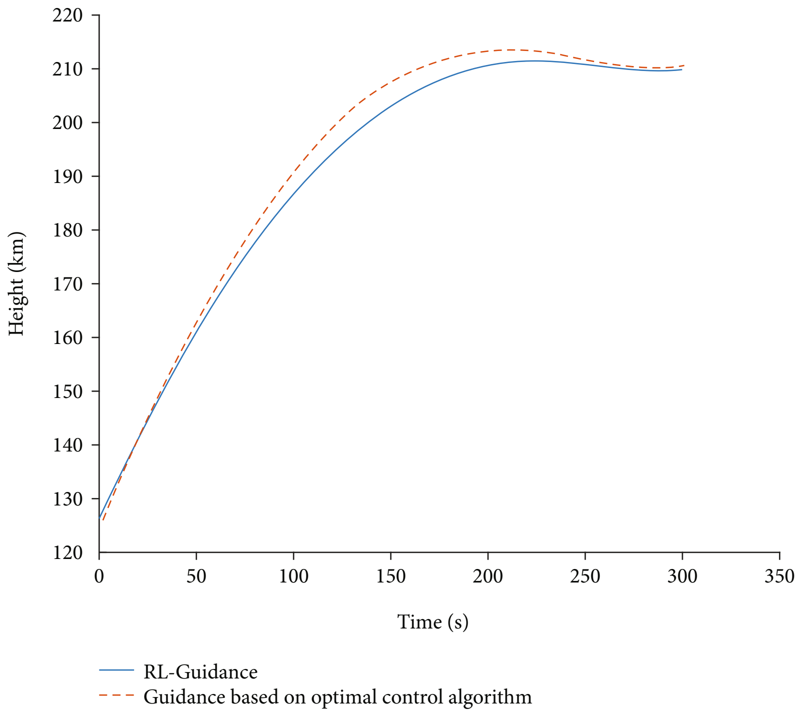

Figure 8: Height comparison of RL-guidance and guidance-based optimal control.

The above expression can be regarded as a generalized actor-critic framework, where Ψ is the evaluator.

To further improve the stability of the learning process and reduce the variance in the policy gradient estimation, a baseline function without changing the deviation is considered. Generally, the state-value function $V_{\pi_\theta}(s)$ is selected as the baseline function. By using the baseline function, an advantage function $A_{\pi_\theta}(s, a)$ can be obtained, and the expression is:

$$A_{\pi_\theta}(s, a) = Q_{\pi_\theta}(s, a) - V_{\pi_\theta}(s). \tag{13}$$

Table 3: Results of RL-guidance and guidance-based optimal control.

Parameter	RL-guidance	Guidance-based optimal control
Final time (s)	299.94	301.15
Final position error (m)	(2.767, 15.614, 128.149)	(47.263, -0.145, -16.306)
Final velocity error (m/s)	(-0.642, -0.259, -0.107)	(0.543, -0.007, -0.101)

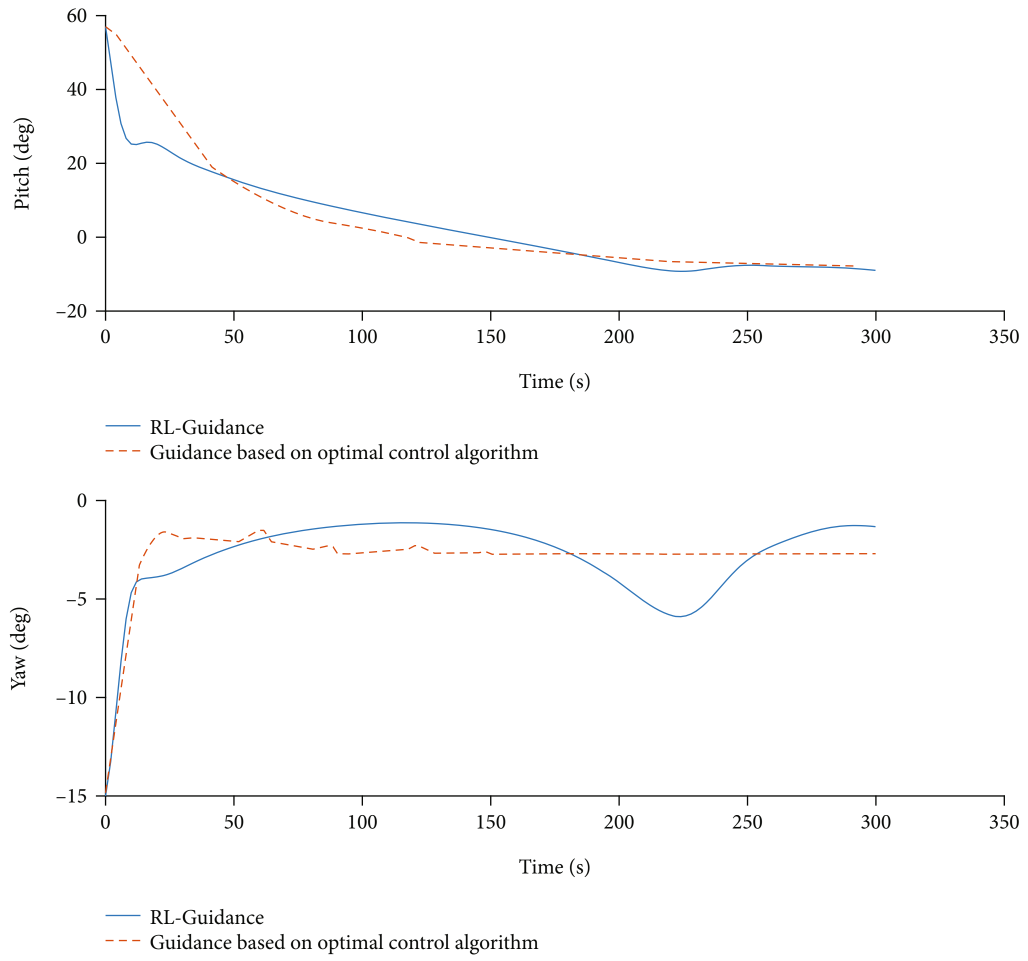

Figure 9: Attitude comparison of RL-guidance and guidance-based optimal control.

Next, the advantage function is estimated by using temporal-difference error (td-error); the expression is:

$$\delta_{\pi_\theta} = R_{s,a} + \gamma V_{\pi_\theta}\left(s'\right) - V_{\pi_\theta}(s). \tag{14}$$

It can be proved that td-error is an unbiased estimator of the advantage function by the following expression:

$$\begin{aligned}\mathbb{E}_{\pi_\theta}\left[\delta_{\pi_\theta} | s, a\right] &= \mathbb{E}_{\pi_\theta}\left[R_{s,a} + \gamma V_{\pi_\theta}\left(s'\right) | s, a\right] - V_{\pi_\theta}(s) \\ &= Q_{\pi_\theta}(s, a) - V_{\pi_\theta}(s) = A_{\pi_\theta}(s, a).\end{aligned} \tag{15}$$

Therefore, by estimating the advantage function by td-error, the policy gradient can be obtained as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) A_{\pi_\theta}(s, a)\right]. \tag{16}$$

In the above expression, the advantage function is brought into the reward objective function, and the effect caused by the change of the state-value function is removed from the action-value function, thereby the variance is reduced. In this method, a neural network can be set up to approximate the policy and evaluation functions.

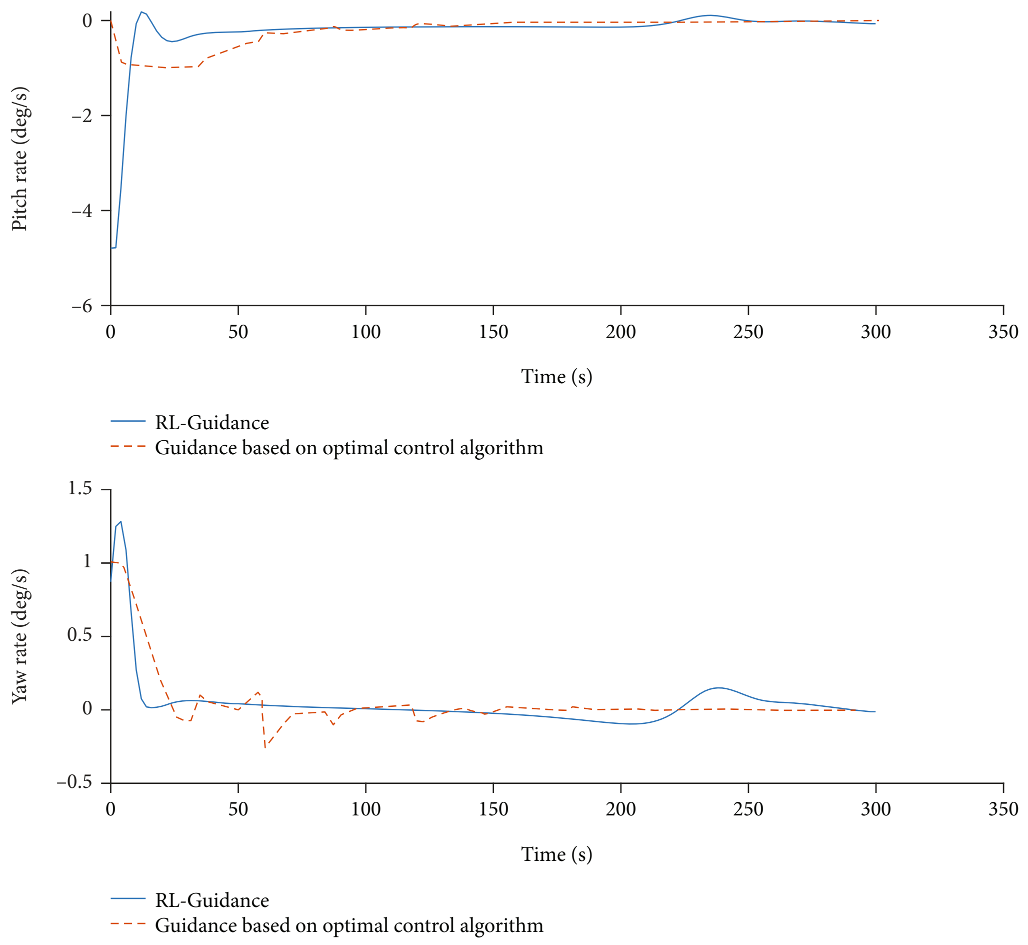

Figure 10: Control comparison of RL-guidance and guidance-based on optimal control.

2.3. Proximal Policy Optimization. The proximal policy optimization (PPO) algorithm [45] is a policy gradient algorithm that is derived from the TRPO algorithm. At present, the PPO algorithm is one of the recommended algorithms in the field of reinforcement learning. The policy gradient algorithm is sensitive to the learning step size. To solve this problem, Schulman et al. proposed the TRPO algorithm. The TRPO algorithm adopts a monotonic maximum step size method to update the policy, while using KL divergence to express the special constraint that the new policy is better than the old policy. The algorithm does not aim to update the step size, but uses an alternative loss function, which finally transforms the reinforcement learning policy update problem into the following optimization problem:

$$\max \mathbb{E}_{a,\pi_{\text{old}}}\left[\frac{\pi(a|s)}{\pi_{\text{old}}(a|s)}\left(Q_{\pi_{\text{old}}}(s,a)-V_{\pi_{\text{old}}}(s)\right)\right], \quad \text{s.t.}\bar{\mathrm{D}}_{KL}^{\rho_{\pi_{\text{old}}}}(\pi_{\text{old}},\pi)\le\delta. \tag{17}$$

The TRPO algorithm uses Taylor expansion to expand the constraints and uses the conjugate gradient method to optimize the network parameters, which can ensure monotonic improvement of the policy model during optimization. However, the theory of the algorithm is complex and not easy to implement and debug by coding. To solve this problem, Schulman et al. made a first-order approximation of the TRPO algorithm and proposed the PPO algorithm. The expected approximation is completed by using the Monte Carlo method, so the objective function becomes:

$$\max \frac{1}{N}\sum_{t=1}^{N}\left[\frac{\pi_t(a|s)}{\pi_{\text{old},t}(a|s)}\left(Q_{\pi_{\text{old}},t}(s,a)-V_{\pi_{\text{old}},t}(s)\right)\right], \tag{18}$$

where $r_t(\theta)$ represents the ratio of old and new policies $\pi_t(a|s)/\pi_{\text{old},t}(a|s)$ in the expression, and the objective function is transformed into:

$$J(\theta)=\mathbb{E}_t\left[r_t(\theta)\left(Q_{\pi_{\text{old}},t}(s,a)-V_{\pi_{\text{old}},t}(s)\right)\right]. \tag{19}$$

The PPO algorithm rewrites the objective function in the TRPO algorithm as:

$$J^{\text{CLIP}}(\theta)=\mathbb{E}_t[\min\,(r_t(\theta)A_t,\text{clip}(r_t(\theta),1-\epsilon,1+\epsilon)A_t)], \tag{20}$$

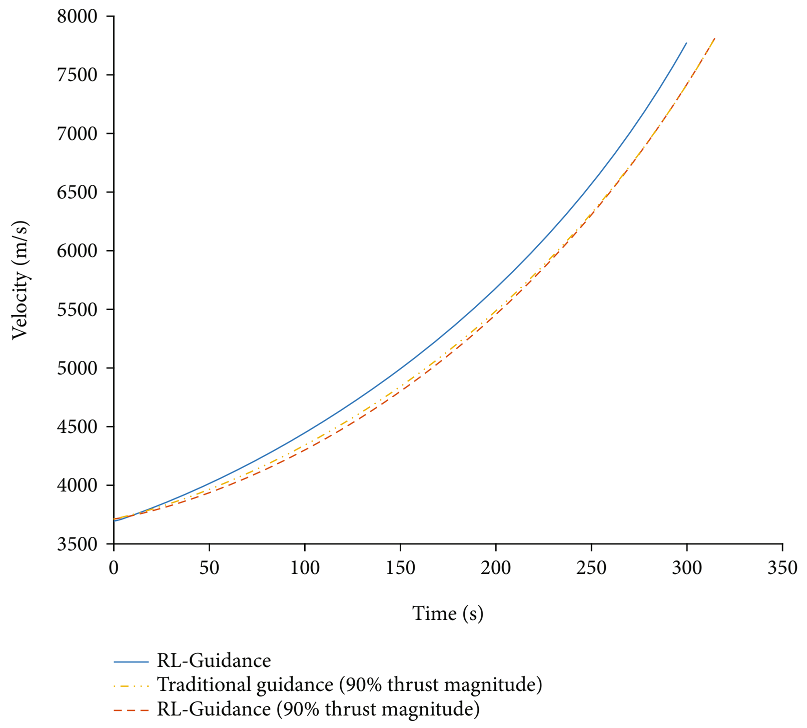

FIGURE 11: Velocity comparison of RL-guidance and traditional guidance in experiment 2.

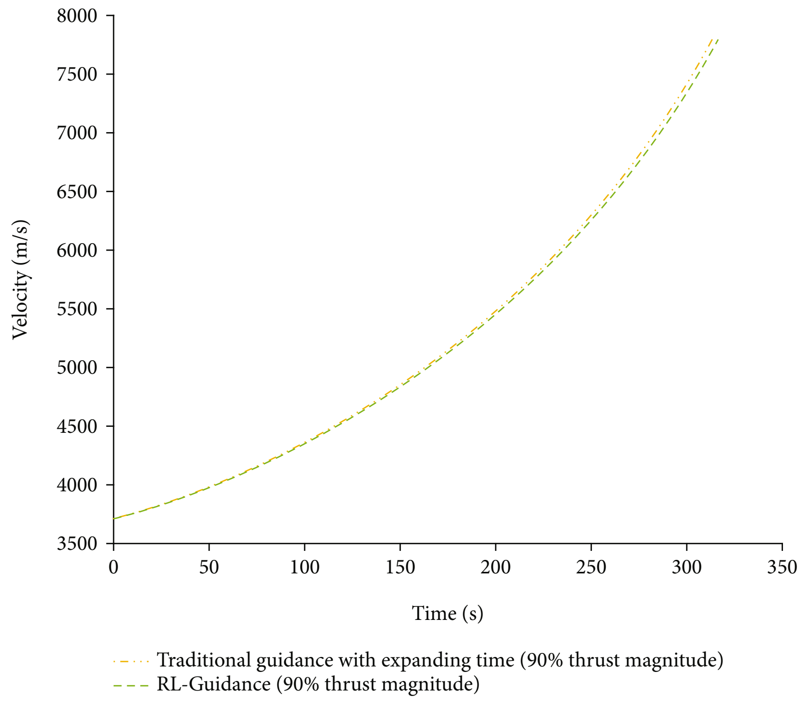

FIGURE 12: Velocity comparison of RL-guidance and traditional guidance with expanding flight time in experiment 2.

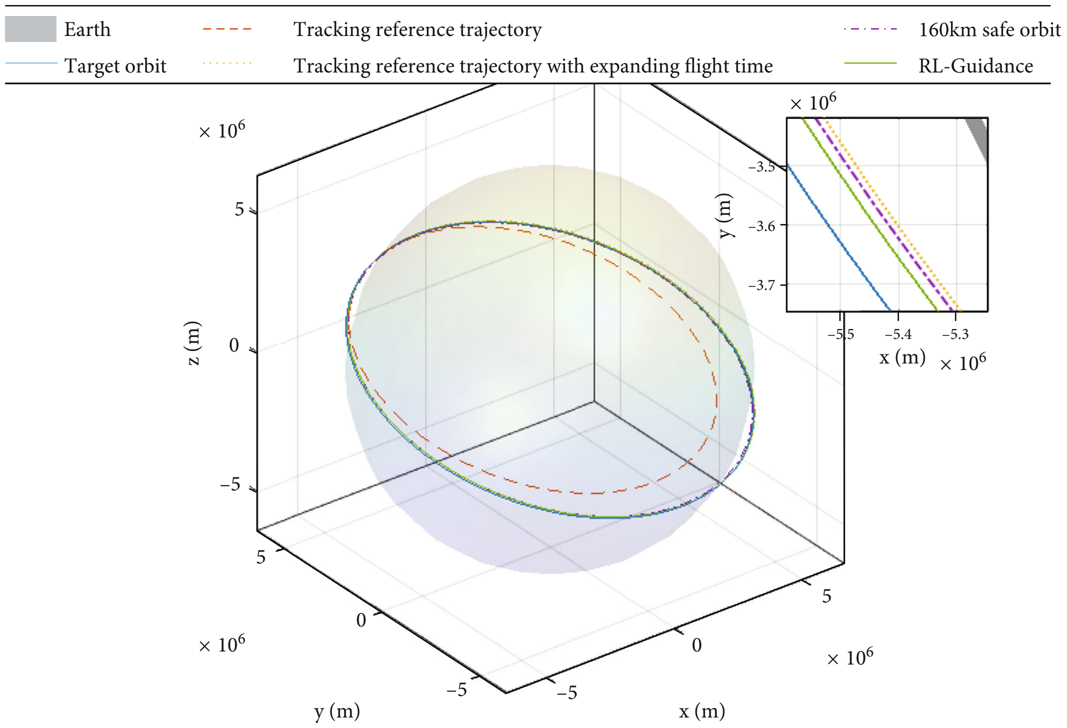

Figure 13: Orbit reached by using different guidance in experiment 2.

where $A_t = Q_{\pi_{\text{old}},t}(s, a) - V_{\pi_{\text{old}},t}(s)$. The clip function limits the range of $r_t(\theta)$ to $[1-\epsilon, 1+\epsilon]$, which ensures that each update will not fluctuate too much, and ϵ is a hyperparameter. The PPO algorithm adds the objective of the value function to the optimization objective, and the expression is:

$$J_t^{\text{PPO}}(\theta) = \mathbb{E}_t\left[J_t^{\text{CLIP}}(\theta) - cJ_t^V(\theta)\right], \tag{21}$$

where c is the value function coefficient and $J_t^V(\theta)$ is the mean squared error between the current value function estimation and the obtained reward to go, which is expressed as:

$$J_t^{\ V}(\theta) = \mathbb{E}_t\left[\left(V_t - \sum_{t=0}^{t} R_{t'}\right)^2\right]. \tag{22}$$

In practice, the process of PPO is as follows (shown in Figure 1):

(1) Rollouts phase. First, train n episodes in the environment through the current policy, and generate a batch of trajectories; each trajectory associated with a single episode, including the corresponding states, actions, and rewards

(2) Update phase. The policy optimization algorithm updates the policy using a batch of trajectories (rollouts). Then, the network's parameter θ is updated by the following expression:

$$\theta_{k+1} \longleftarrow \theta_k + \alpha_k \nabla_{\theta_k} J_k^{\text{PPO}}(\theta). \tag{23}$$

(3) The training is stopped when a user-defined iteration number is reached

3. Problem Statement

3.1. Dynamics Model. In the ascent process of the launch vehicle, the flight time in the atmosphere is short, and deviation in the atmosphere can be corrected by the guidance of the upper stage. Thus, the guidance of the upper stage determines the orbit insert accuracy of the launch vehicle. Therefore, this manuscript focuses on the guidance of the upper stage of the launch vehicle. The dimensionless equations of motion of a three-dimensional (3D) launch vehicle can be expressed in launch-inertial coordinate as follows:

$$\begin{aligned}
\dot{x} &= v_x, \quad \dot{y} = v_y, \quad \dot{z} = v_z,\\
\dot{v}_x &= \frac{T\cos\varphi\cos\psi}{m} - \frac{(x+R_{ex})}{\|\mathbf{r}_e\|^3},\\
\dot{v}_y &= \frac{T\sin\varphi\cos\psi}{m} - \frac{(y+R_{ey})}{\|\mathbf{r}_e\|^3},\\
\dot{v}_z &= -\frac{T\sin\psi}{m} - \frac{(z+R_{ez})}{\|\mathbf{r}_e\|^3},\\
\dot{m} &= -\frac{T}{I_{\text{sp}} g_0},
\end{aligned} \tag{24}$$

Table 4: Results of experiment 2.

Parameter	Target orbit	Tracking reference trajectory	RL-guidance	Tracking reference trajectory with expanding flight time
Semimajor (m)	6588753	5953963	6588753	6588753
Eccentricity	0.0052	0.1046	0.0054	0.0102
Inclination (°)	20.009	19.991	20.009	20.019
Altitude of perigee (km)	176	-1047	175	143

in which $\mathbf{r} = [x, y, z]$ is the position of the launch vehicle in the launch-inertial coordinate, which is normalized by the radius of Earth $R_0 = 6378145$ m, and $\mathbf{v} = [v_x, v_y, v_z]$ is the velocity of the launch vehicle in the launch-inertial coordinate, which is normalized by $\sqrt{R_0 g_0}$, in which $g_0 = 9.81$ m/s^2. The position from Earth's center to the vehicle $\mathbf{r}_e = [x + R_{ex}, y + R_{ey}, z + R_{ez}]$ is normalized by R_0, and $\mathbf{R}_e = [R_{ex}, R_{ey}, R_{ez}]$ is the dimensionless position from Earth's center to the launch-inertial coordinate's origin which is the launch point. m is the mass of the launch vehicle. T is the thrust magnitude; as in most launch vehicles, the mass flow is uncontrollable; therefore, the thrust magnitude T is uncontrollable during the same flight phase [15, 47]. I_{sp} is the specific impulse of the engine. φ and ψ are the pitch and yaw angle, respectively, measured in the thrust vector in the launch-inertial coordinate. The differentiation of equations in Equation (24) is with respect to dimensionless time normalized by $\sqrt{R_0/g_0}$.

To apply reinforcement learning to launch vehicle guidance problems and satisfy the constraints of the flight process, this manuscript uses the optimal control expression, which can represent initial, terminal, and process constraints. The guidance problem of the upper stage of the launch vehicle can be written as follows:

Problem:

$$\min\ J = -m(t_f), \quad \text{s.t.} \tag{25}$$

Equation (24), t_f free

$$\mathbf{r}(t_0) = \mathbf{r}_0, \quad \mathbf{v}(t_0) = \mathbf{v}_0, \tag{26}$$

$$\mathbf{r}(t_f) = \mathbf{r}_f, \quad \mathbf{v}(t_f) = \mathbf{v}_f, \tag{27}$$

$$\dot{\varphi}_{\min} \le \dot{\varphi} \le \dot{\varphi}_{\max}, \quad \dot{\psi}_{\min} \le \dot{\psi} \le \dot{\psi}_{\max}, \tag{28}$$

$$m(t_f) \ge m_{\text{dry}}, \tag{29}$$

in which $\mathbf{r} = [x\, y\, z]$ and $\mathbf{v} = [v_x \quad v_y \quad v_z]$. $\mathbf{r}_0$ and $\mathbf{r}_f$ represent the initial and terminal position, respectively. $\mathbf{v}_0$ and $\mathbf{v}_f$ represent the initial and terminal velocity, respectively. t_0 and t_f are the start and final time, respectively. And m_{dry} is the dry mass of the launch vehicle. Equation (25) is the cost function. Equation (26) and Equation (27) represent the initial and terminal constraints. The minimum and maximum values of pitch and yaw angle rates are presented in Equation (28). In this manuscript, the angle constraint is regarded as a hard constraint. Once the constraint is violated, the current episode is stopped, and a large negative reward is returned, then a new episode is started. Equation (29) represents the fuel constraint. When the fuel runs out, the current episode is stopped.

3.2. Implementation Details. This section describes the techniques we use in using reinforcement learning. For the network of policy and value, we design a neural network with tanh activations on each hidden layer. In the policy network, the input layer has $n_s^p = 8$ neurons, the output layer has $n_o^p = 2$ neurons, and the number of hidden layers is three, and the sizes of the hidden layers are $h_1^p = 10n_s$, $h_2^p = \sqrt{h_1^p h_3^p}$, and $h_3^p = 10n_o^p$, respectively, where h_1^p, h_2^p, and h_3^p represent the size of each hidden layer. In the value network, the input layer has $n_s^v = n_s^p$ neurons, the output layer has $n_o^v = 1$ neuron, the number of hidden layers is three, and the sizes of the hidden layers are $h_1^v = 10n_s^v$, $h_2^v = \sqrt{h_1^v h_3^v}$, and $h_3^v = 5$, respectively, where h_1^v, h_2^v, and h_3^v represent the size of each hidden layer. This structure has been studied in aerospace trajectory optimization, such as Mars landing [26] and Earth-Mars transfer orbit [19]. To generate the corresponding action, the policy uses Gaussian distribution with mean $\pi_\theta(a_k|s_k)$ and a diagonal covariance matrix for action distribution. Moreover, the Adam optimizer is used to adjust the learning rate of policy and value networks. A method similar to the PPO2 algorithm [45] in OpenAI baselines is used to approximate KL divergence. The expression is as follows:

$$\epsilon = \begin{cases} \min\left(\epsilon_{\max}, 1.5\epsilon\right), & \text{if kl} < \dfrac{1}{2}\text{kl}_{\text{targ}}, \\ \max\left(\epsilon_{\min}, \dfrac{1}{1.5}\epsilon\right), & \text{if kl} > 2\text{kl}_{\text{targ}}, \end{cases} \tag{30}$$

$$\zeta = \begin{cases} 1.5\zeta, & \text{if kl} < \dfrac{1}{2}\text{kl}_{\text{targ}} \text{ and } \epsilon \dfrac{1}{2}\epsilon_{\max} \text{and } \zeta < \zeta_{\max}, \\ \dfrac{1}{1.5}\zeta, & \text{if kl} > 2\text{kl}_{\text{targ}} \text{ and } \epsilon << 2\epsilon_{\min} \text{and} \zeta > \zeta_{\min}. \end{cases} \tag{31}$$

According to the suggestions of [26], we adjust the parameters according to the KL divergence between policy updates, represented by kl. In addition, $\epsilon_{\max}$ and $\epsilon_{\min}$ are designed to be 0.5 and 0.01, respectively. We also adjust the parameters according to Equation (31), in which $\zeta_{\max}$ and $\zeta_{\min}$ are designed as 10 and 0.1, respectively.

To apply the reinforcement learning method to launch vehicle ascending guidance, in combination with the dynamics model, the observation, action, and reward are designed. In the research of reinforcement learning for aerospace guidance, there is no unified choice for observation. In [48], the authors designed $s = \{\mathbf{r} - \mathbf{r}_{\text{ref}}, \mathbf{v} - \mathbf{v}_{\text{ref}}\}$ for learning, in which the subscript ref represents the reference trajectory. In [26], the authors used a similar idea: they designed a velocity field $\mathbf{v}_{\text{targ}}$ that mapped the lander's position to a target velocity for learning, which achieved good results. Unfortunately, the construction of $\mathbf{v}_{\text{targ}}$ is not general, and it cannot be applied to all problems. However, this method provides an inspiration: if a good reference state can be designed, good learning efficiency and final results can be obtained. In [19], the authors did not use the reference trajectory, the state of the aircraft was regarded as observation, and good results were obtained. Combined with the motion equations introduced in Section 3.1, the expression of observation designed in this manuscript is as follows:

$$\text{obs} = \{\mathbf{r}, \mathbf{v}, \varphi, \psi\}. \tag{32}$$

The guidance commands of the launch vehicle are generally the pitch angle and yaw angle, but the angular rate is limited. If these angles are used as the action of the neural network, the angular rate is not easy to control. To satisfy the angular rate constraint, we use the angular rate as the action and the attitude angle as a part of the observation.

In addition, it should be noted that stop conditions need to be designed in reinforcement learning. In the research of reinforcement learning guidance algorithms [24, 30], terminal velocity or position constraints are usually used as stop conditions for each episode. However, in low earth orbit (LEO) missions studied in this manuscript, the semimajor axis is one of the indicators of engine shutdown. If the semimajor axis of the orbit at the current time exceeds the semimajor axis of the target orbit, the guidance system sends an engine shutdown command. In this manuscript, the current episode is terminated if the semimajor axis of the orbit at the current time exceeds the semimajor axis of the target orbit.

3.3. Reward Function. In [49], the authors presented the hypothesis that the maximization of total reward may be enough to understand intelligence and its associated abilities. A suitable reward can make the agent learn knowledge faster and better, but how to design a suitable reward function is one of the difficulties of reinforcement learning, especially in the aerospace guidance field. In the launch vehicle guidance problem, the thrust magnitude cannot be controlled, and the thrust direction can only be controlled by the attitude of the vehicle, which makes the problem difficult to solve. Thus, although many scholars now use mathematical optimization algorithms as the basis for computational guidance, there are few engineering applications because the problem is not easy to solve, and the calculation time is too long to be applied online.

A common practice is to give a reward after running an episode. However, reinforcement learning randomly selects the control during the training. If the reward is only based on the final result, it is very likely that the terminal condition will never be satisfied. This is called the sparse reward problem. This problem is generally solved using inverse reinforcement learning, where the reward function for each step is learned through expert representations. In this problem, solutions obtained by mathematical optimization algorithms such as convex optimization can be used in expert representations, but the calculation time of the mathematical optimization algorithm is uncertain, and therefore, it cannot be well applied to inverse reinforcement learning.

In the ascending flight of the launch vehicle, the velocity and position increase gradually. At each time step, we can reward the agent if the agent drives it toward the target point. This method called shaping reward was proposed by Ng [50]. Gaudet et al. used this method in the Mars landing guidance [26], but the shaping reward constructed by Gaudet et al. cannot be well applied in other fields. Therefore, a simple but effective shaping reward is proposed in this manuscript. The reward function expression is as follows:

$$r_{\text{shape}} = \lambda_{\text{track}}(\|\mathbf{r}(t_k) - \mathbf{r}_f\|), \tag{33}$$

where r_{shape} is a negative reward, which represents the distance between the current position $\mathbf{r}(t_k)$ and the terminal position $\mathbf{r}_f$, and λ_{track} is a shaping reward coefficient. The way to minimize the shaping reward is to move the vehicle toward the target point directly. Moreover, because the shaping reward is related to the number of steps, the fewer steps, the fewer negative rewards. For a launch vehicle with constant mass flow, the minimum number of steps means the optimal energy. Therefore, the shaping reward designed in this manuscript can not only guide the vehicle to the target but also minimize the number of steps, to achieve the optimal energy.

When an episode is stopped, the final reward will be given. We refer to the reward function in [19], and the expression is as follows:

$$r_{\text{done}} = \lambda_d \max\{0, e_{r,v} - \varepsilon\}, \tag{34}$$

where r_{done} is a negative reward and called the final reward, λ_d is the final reward coefficient, and ε is tolerance on terminal violation. The expression of ε is as follows [46]:

$$\varepsilon = \begin{cases} \varepsilon_0, & k \le k_i, \\ \varepsilon_i \left[\dfrac{\varepsilon_f}{\varepsilon_i}\right]^{(k-k_i)/(k_f-k_i)}, & k_i < k < k_f, \\ \varepsilon_f, & k \ge k_f, \end{cases} \tag{35}$$

where k is the current episode number, $\varepsilon_0 = 0.0005$, $\varepsilon_f = 1e - 6$, $k_i = 0$, and $k_f = 450000$.

The expression of $e_{\mathrm{r,v}}$ is as follows:

$$e_{\mathrm{r,v}} = \begin{cases} \max\{e_{\mathrm{r}}, e_{\mathrm{v}}\}, & \text{if done}, \\ 0, & \text{if not done}, \end{cases} \tag{36}$$

where e_{r} and e_{v} are represent the final position error and the final velocity error, respectively. The expression of e_{r} and e_{v} are as follows:

$$\begin{aligned} e_{\mathrm{r}} &= \frac{\|\mathbf{r}(t_{\mathrm{f}}) - \mathbf{r}_{\mathrm{f}}\|}{\|\mathbf{r}_{\mathrm{f}}\|}, \\ e_{\mathrm{v}} &= \frac{\|\mathbf{v}(t_{\mathrm{f}}) - \mathbf{v}_{\mathrm{f}}\|}{\|\mathbf{v}_{\mathrm{f}}\|}. \end{aligned} \tag{37}$$

In addition, considering the process constraints on attitude during flight, a penalty function is designed. When the process constraints are violated, the current episode stops immediately and returns the penalty. The penalty function r_{penalty} is given by the following:

$$r_{\mathrm{penalty}} = \begin{cases} -200, \text{if Constraint violated}, \\ 0, \text{if Constraint is not violated}. \end{cases} \tag{38}$$

To sum up, the design of the total reward function for the guidance problem of the launch vehicle is given by the following:

$$r = r_{\mathrm{shape}} + r_{\mathrm{penalty}} + r_{\mathrm{done}} + \eta, \tag{39}$$

where η is a constant positive reward. In the numerical experiment, we found that without this positive reward, the agent will immediately violate the constraint at the beginning of the training. This positive reward is the key to encouraging the agent to continue to move forward.

Figure 2 shows how reinforcement learning can be applied to the guidance of the launch vehicle. It can be seen that the DNN obtained by reinforcement learning is called RL-guidance, which outputs the guidance commands, that is, the actions in reinforcement learning. The vehicle flies Δt time according to the guidance command, and then, the state of $t_k + \Delta t$ is obtained by the navigation system, and reward is obtained by the reinforcement learning model feedback to the RL-guidance system.

4. Experimental Results and Discussion

In this section, we apply the proposed algorithm to the ascent problem of the launch vehicle to verify its validity. All numerical simulations are implemented on a computer with a 4-core Intel Core E3-1230 V5 CPU @3.4 GHz, and the RL-guidance and the guidance-based optimal control are implemented in Python and Matlab environments, respectively.

The launch vehicle thrust is 2843425 N, and the specific impulse is 3365 m/s. The initial and dry mass are 350306 kg and 83090 kg, respectively. Maximum pitch and yaw angle rate is 5°.

Table 1 shows the initial and terminal parameters of the numerical experiment. The fourth-order Runge–Kutta integration is used by integrating with a 0.5 s step, and the guidance step is 1 s.

4.1. Policy Optimization.

4.1. Policy Optimization. This section presents the training process of reinforcement learning.

Table 2 lists the reward coefficients and the hyperparameters. Rollouts are generated by the interaction between the agent and the environment for 50 episodes, the advantages, the value, and policy function approximators are computed and updated by the resulting trajectories. The total episode is 500000, which took nearly 30 hours.

Figures 3 and 4 show the final position and velocity error curves, respectively, it can be seen that with increased training episodes, and the final error gradually decreases and converges after 400 thousand episodes. As can be seen from Figure 5, the reward gradually increases as the training progresses; the value of reward increases rapidly in the early stage of training and gradually converges after 400 thousand episodes.

4.2. Policy Test. At present, in the research of aerospace computational guidance, online trajectory planning is mostly performed by optimal control solvers such as GPOPS or CVX, which replace the traditional offline planning and online tracking mode. It should be noted that if the distance between the current and final point is less than 10,000 m, the integration step size is reduced from 0.5 s to 0.02 s. This method is also usually used in practice, that is, when vehicle approaches the target, the integration step size is reduced to improve the final accuracy.

4.2.1. Experiment 1. Figures 6 and 7 show comparisons of position and velocity, and Figure 8 shows the comparison of flight height. It can be seen that the results obtained by the two methods are basically the same. The final results of the two methods are listed in Table 3. As can be seen in the table, the accuracy of the proposed algorithm is consistent with guidance-based optimal control, which fully proves the effectiveness of the proposed algorithm. In addition, as mentioned before, although the training time is very long, once the training is completed, it only needs to perform some matrix multiplication operations when in use. In this experiment, the average and standard deviation time of a generated guidance command are 0.00055 s and 0.00008 s, respectively, and the median and maximum time are 0.00052 s and 0.0017 s, respectively. As a comparison, the current guidance period in engineering applications is about 0.002 s, and it can be seen that the computational efficiency of RL-guidance allows it to be fully applied online. In contrast, the guidance-based optimal control takes 20 s. Considering the difference in the application environment of the two methods, the calculation speed is still much slower than the proposed algorithm, and it is difficult to be applied online.

Figures 9 and 10 show the attitude and control curves, respectively, of the vehicle. It can be seen that the solutions of the two algorithms are very close. As mentioned before, because the thrust magnitude of the launch vehicle is not adjustable, the thrust direction can only be adjusted through limited attitude changes, which leads to a small solution space, so the solutions of the two methods are very close. It can be seen in [26] that there is an obvious difference between the solution of GPOPS and reinforcement learning; because the thrust magnitude of landing vehicle is adjustable and the solution space is large, reinforcement learning may learn other solutions that satisfy the terminal conditions. For problems with a small solution space, on the one hand, once the DNN is trained, the solution obtained by the DNN will be very close to the optimal solution, like the results obtained in this experiment. On the other hand, it will be difficult to find a suitable solution during training, resulting in a failure to train a suitable network.

4.2.2. Experiment 2. In the mission of launch vehicles, the decline of thrust is one of the fatal faults. If the thrust loss is small, the trajectory can be reconstructed to guide the vehicle to the target orbit. However, if the thrust loss is too large, the trajectory planning problem becomes an infeasible problem, and the optimal control algorithm cannot directly give the feasible solution, which means the guidance-based optimal control cannot give new guidance commands. Many scholars have studied that [15, 51, 52], in that situation, the primary goal of the mission changes from accurately entering the target orbit to moving in the orbit waiting for rescue. And the basic idea is to change the terminal constraints to make the new problems feasible, the new terminal constraint represents the new orbit, which is called the rescue orbit.

The following experiment is that the thrust is reduced by 10%, which is very likely to happen when the upper-stage engine was started.

In the case mentioned above, the remaining energy of the launch vehicle may not be able to send the payload such as a satellite into the target orbit. The guidance-based optimal control transforms the guidance problem into a nonlinear programming problem. If the launch vehicle cannot reach the target orbit because the thurst drops, it means that the original problem is infeasible, and there is no solution. Therefore, the guidance-based optimal control will not work during the flight. In this case, we assume that the guidance algorithm will switch to the method of tracking the reference trajectory, and the reference trajectory is preplanned under nominal conditions. There are two tracking methods, the first method is that the vehicle flies along the reference trajectory and shut down at the reference final time; this method is the traditional guidance. But we know that there will be some surplus fuel in the launch vehicle. Therefore, the second method will expand the flight time until the fuel is completely exhausted or some other indicators meet the requirements. However, there is a problem that when the final time of the reference trajectory is exceeded, there is no new guidance command. In this case, the last group of guidance commands in the reference trajectory can only be regarded as subsequent guidance commands.

In the first tracking method, Figure 11 shows the flight curves of the tracking reference trajectory after the failure, which is compared with RL-guidance. It can be seen from Figure 11 that the vehicle loses a lot of velocity due to the decline of thrust. And to satisfy the semimajor axis requirement, the RL-guidance expands the flight time.

In the second tracking method, Figure 12 shows the velocity curves; the green line indicates that the vehicle flies along the reference trajectory with expanding flight time until the semimajor exceeds the target semimajor. The final velocity of the launch vehicle is basically consistent with the reference terminal velocity. However, we find that this method still cannot put the launch vehicle into orbit through the following analysis.

From Figure 13 and Table 4, it can be seen that the new orbit obtained by RL-guidance is very close to the target orbit and far better than the result of traditional guidance. It should be noted that if the altitude of perigee of an orbit is less than 160 km, it is considered that this orbit is inappropriate, and the payload on this orbit will gradually fall into the atmosphere [52], the purple dash-dotted line indicates the safe orbit mentioned in [52], and the safe orbit is a circular orbit with 160 km orbit altitude that is abovementioned. As can be seen from Figure 13, the red dashed line indicates that the traditional guidance cannot guide the vehicle in an orbit. In addition, even if the flight time is increased, the reached orbit indicated by the yellow dotted line still cannot meet the requirements because the altitude of part of the orbit is less than 160 km. The green solid line indicates the orbit reached by RL-guidance when the thrust drops. It can be seen that this orbit can be used as the rescue orbit. Therefore, neither the first tracking method nor the second tracking method can put the payload into orbit. As a result, although expanding the flight time can increase the velocity, inappropriate guidance commands cannot make the vehicle enter the appropriate orbit. However, RL-guidance can generate the new guidance commands according the current state and guide the launch vehicle to a suitable orbit after the thrust drops.

For computational guidance-based optimal control, it needs an extra strategy to find a new orbit [52], the strategy takes into account various factors, such as the appropriate orbit inclination or longitude of ascending node, so the orbit obtained by this strategy is called the optimal rescue orbit. However, the optimal rescue orbit requires many iterations and takes a lot of time to find.

It still needs to discuss whether it is worth taking so many iterations to obtain the optimal rescue orbit in case of thrust failure and which rescue orbit is more important, optimal, or feasible. The proposed RL-guidance algorithm can quickly get a feasible rescue orbit, which may not be optimal, but feasible. The proposed RL-guidance algorithm continuously generates guidance commands according to the mapping of states and controls that the DNN trained. If the thrust drops, the proposed RL-guidance algorithm can generate new guidance commands according to the current state and guide the vehicle to a feasible rescue orbit. The proposed RL-guidance algorithm is autonomous, can be used as an alternative method, and is worthy of further research in case of thrust failure of launch vehicle mission.

According to the two experimental results given in this section, the results of the proposed RL-guidance are consistent with guidance-based optimal control. In addition, the proposed RL-guidance has higher computational efficiency and can be applied online. In terms of the thrust decline, the guidance-based optimal control transforms the guidance problem into an optimization problem; if the thrust drops, the original problem becomes infeasible because the target orbit cannot be reached, and the optimization algorithm cannot give a solution, which means that the guidance will not work during the flight. In this case, if the guidance system is switched to track the reference trajectory, the results show that it cannot make the vehicle in a suitable orbit. But the proposed RL-guidance can generate the new guidance commands according to the current state and guide the vehicle to a feasible orbit, which makes rescue possible.

5. Conclusions

This manuscript proposes a guidance-based reinforcement learning method and intends to demonstrate that reinforcement learning is a viable approach to developing a guidance algorithm for launch vehicles. In the research of computational guidance, most methods are based on optimal control algorithms, and the proposed guidance method is based on DNN. First, the reward function was designed to cover all constraints. After that, the mapping from state to control is trained by the state-of-the-art proximal policy optimization algorithm.

Two numerical experiments are designed to test the proposed algorithm. In the first numerical experiment, the results of the proposed algorithm are consistent with guidance-based optimal control. It shows that the proposed algorithm is effective and fast and has the potential for online application. The second numerical experiment aims to demonstrate the ability of the proposed algorithm under thrust drops. The current guidance algorithm research is based on the optimal control algorithm. If the original problem becomes infeasible because thrust drops, the guidance cannot generate commands; therefore, it needs an extra strategy to find a new orbit to make the programming problem feasible, and then, the guidance-based optimal control can output commands, the orbit obtained through the strategy is called an optimal rescue orbit, and it takes a lot of computational time. Not aiming to get the optimal rescue orbit, the proposed algorithm can guide launch vehicles to a feasible orbit and wait for rescue without any extra strategy. Moreover, the numerical experimental results indicate that the traditional guidance that uses offline planning and online tracking mode cannot deal with this kind of emergency. Therefore, the proposed algorithm can be used as an alternative guidance algorithm, especially in the case of thrust decline fault. In future research, guiding the launch vehicle to different rescue orbits under different faults will be considered, as well as adding various disturbances to the training. Since the mission is more complex, more training epochs may be required, and therefore, parallel computing techniques will be considered.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] P. Lu, “Introducing computational guidance and control,” *Journal of Guidance Control & Dynamics*, vol. 40, no. 2, pp. 193–193, 2017.

[2] P. Lu and B. Pan, “Highly constrained optimal launch ascent guidance,” *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 404–414, 2010.

[3] M. J. Grant and R. D. Braun, “Rapid indirect trajectory optimization for conceptual design of hypersonic missions,” *Journal of Spacecraft and Rockets*, vol. 52, no. 1, pp. 177–182, 2015.

[4] P. Lu, S. Hongsheng, and T. Bruce, “Closed-loop endoatmospheric ascent guidance,” *Journal of Guidance, Control, and Dynamics*, vol. 26, no. 2, pp. 283–294, 2003.

[5] F. Fariba and R. I. Michael, “Advances in pseudospectral methods for optimal control,” in *AIAA guidance, navigation and control conference and exhibit*, Honolulu, Hawaii, USA, 2008.

[6] M. A. Patterson and A. V. Rao, “GPOPS-II,” *ACM Transactions on Mathematical Software*, vol. 41, no. 1, pp. 1–37, 2014.

[7] R. I. Michael and K. Mark, “A review of pseudospectral optimal control: from theory to flight,” *Annual Reviews in Control*, vol. 36, no. 2, pp. 182–197, 2012.

[8] A. E. Bryson and Y. Ho, “Applied optimal control,” *Technometrics*, 1975.

[9] P. Haijun, X. Wang, L. Mingwu, and C. Biaosong, “An _hp_ symplectic pseudospectral method for nonlinear optimal control,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 42, pp. 623–644, 2017.

[10] P. Haijun, X. Wang, Z. Sheng, and C. Biaosong, “An iterative symplectic pseudospectral method to solve nonlinear state-delayed optimal control problems,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 48, pp. 95–114, 2017.

[11] X. Wang, P. Haijun, Z. Sheng, C. Biaosong, and Z. Wanxie, “A symplectic pseudospectral method for nonlinear optimal control problems with inequality constraints,” *ISA Transactions*, vol. 68, pp. 335–352, 2017.

[12] A. Behcet and S. R. Ploen, “Convex programming approach to powered descent guidance for Mars landing,” *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[13] H. M. Wade, *Lossless Convexification of Optimal Control Problems*, Doctor of Philosophy, 2014.

[14] A. Behcet, J. M. Carson, and B. Lars, “Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem,” *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2113, 2013.

[15] L. Yuan, P. Baojun, W. Changzhu, C. Naigang, and L. Yongbei, "Online trajectory optimization for power system fault of launch vehicles via convex programming," *Aerospace Science and Technology*, vol. 98, p. 105682, 2020.

[16] Z. Wang and M. J. Grant, "Constrained trajectory optimization for planetary entry via sequential convex programming," *Journal of Guidance, Control, and Dynamics*, vol. 40, no. 10, pp. 2603–2615, 2017.

[17] K. Hornik, M. Stinchcombe, and H. White, "Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks," *Neural Networks*, vol. 3, no. 5, pp. 551–560, 1990.

[18] I. Dario, M. Marcus, and B. Pan, "A survey on artificial intelligence trends in spacecraft guidance dynamics and control," *Astrodynamics*, vol. 3, no. 4, pp. 287–299, 2019.

[19] Z. Alessandro and F. Lorenzo, "Reinforcement learning for robust trajectory design of interplanetary missions," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 8, pp. 1440–1453, 2021.

[20] L. Cheng, Z. Wang, J. Fanghua, and L. Junfeng, "Fast generation of optimal asteroid landing trajectories using deep neural networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, 2019.

[21] S.-S. Carlos and I. Dario, "Real-time optimal control via deep neural networks: study on landing problems," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 5, pp. 1122–1135, 2018.

[22] S. Yang and Z. Wang, "Onboard generation of optimal trajectories for hypersonic vehicles using deep learning," *Journal of Spacecraft and Rockets*, vol. 58, no. 2, pp. 400–414, 2021.

[23] L. Cheng, L. Hengnian, Z. Wang, and J. Fanghua, "Fast solution continuation of time-optimal asteroid landing trajectories using deep neural networks," *Acta Astronautica*, vol. 167, pp. 63–72, 2020.

[24] F. Lorenzo, B. Boris, and Z. Alessandro, "Deep learning techniques for autonomous spacecraft guidance during proximity operations," *Journal of Spacecraft and Rockets*, vol. 58, no. 6, pp. 1774–1785, 2021.

[25] L. Cheng, Z. Wang, and J. Fanghua, "Real-time control for fuel-optimal moon landing based on an interactive deep reinforcement learning algorithm," *Astrodynamics*, vol. 3, no. 4, pp. 375–386, 2019.

[26] G. Brian, L. Richard, and F. Roberto, "Deep reinforcement learning for six degree-of-freedom planetary landing," *Advances in Space Research*, vol. 65, no. 7, pp. 1723–1741, 2020.

[27] B. Gaudet, I. Charcos, and R. Furfaro, "Integrated and adaptive guidance and control for endoatmospheric missiles via reinforcement learning," 2021, http://arxiv.org/abs/2109.03880.

[28] G. Brian, F. Roberto, L. Richard, and S. Andrea, "Reinforcement metalearning for interception of maneuvering exoatmospheric targets with parasitic attitude loop," *Journal of Spacecraft and Rockets*, vol. 58, pp. 1–14, 2021.

[29] G. Brian, L. Richard, and F. Roberto, "Adaptive guidance and integrated navigation with reinforcement meta-learning," *Acta Astronautica*, vol. 169, pp. 180–190, 2020.

[30] G. Brian, L. Richard, and F. Roberto, "Six degree-of-freedom body-fixed hovering over unmapped asteroids via LIDAR altimetry and reinforcement meta-learning," *Acta Astronautica*, vol. 172, pp. 90–99, 2020.

[31] G. Brian, L. Richard, and F. Roberto, "Terminal adaptive guidance via reinforcement meta-learning: applications to autonomous asteroid close-proximity operations," *Acta Astronautica*, vol. 171, pp. 1–13, 2020.

[32] A. Scorsoglio, A. D'Ambrosio, L. Ghilardi, B. Gaudet, F. Curti, and R. Furfaro, "Image-based deep reinforcement meta-learning for autonomous lunar landing," *Journal of Spacecraft and Rockets*, vol. 59, no. 1, pp. 153–165, 2022.

[33] L. Federici, A. Scorsoglio, L. Ghilardi et al., "Image-based meta-reinforcement learning for autonomous terminal guidance of an impactor in a binary asteroid system," *AIAA SCI-TECH 2022 Forum*, 2021.

[34] C. J. C. H. Watkins and P. Dayan, *Technical Note: Q-Learning*, Machine Learning, 1992.

[35] G. A. Rummery and M. Niranjan, "On-line Q-learning using connectionist systems," http://www.researchgate.net/publication/2500611_On-Line_Q-Learning_Using_Connectionist_Systems.

[36] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems*, vol. 12, 1999.

[37] R. J. Williams, *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning*, Machine Learning, 1992.

[38] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, 2015.

[39] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-learning," *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, 2016http://arxiv.org/abs/1509.06461.

[40] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," http://arxiv.org/abs/1511.05952.

[41] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of The 33rd International Conference on Machine Learning*, pp. 1995–2003, New York, NY, USA, 2016, https://proceedings.mlr.press/v48/wangf16.html.

[42] M. Hausknecht and P. Stone, "Deep recurrent Q-learning for partially observable MDPs," *Aaai Fall Symposium Series*, 2015, http://arxiv.org/abs/1507.06527v1.

[43] M. Hessel, J. Modayil, H. Van Hasselt et al., "Rainbow: combining improvements in deep reinforcement learning," in *Thirty-second AAAI conference on artificial intelligence*, New Orleans, Louisiana USA, 2018http://arxiv.org/abs/1710.02298.

[44] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust region policy optimization," in *International conference on machine learning*, pp. 1889–1897, Lille, France, 2018, http://arxiv.org/abs/1502.05477v5.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2022, http://arxiv.org/abs/1707.06347.

[46] F. Lorenzo, S. Andrea, Z. Alessandro, and F. Roberto, "Meta-reinforcement learning for adaptive spacecraft guidance during multi-target missions," *IAF Astrodynamics Symposium 2021 at the 72nd International Astronautical Congress*, 2021.

[47] X. Cheng, L. Huifeng, and Z. Ran, "Efficient ascent trajectory optimization using convex models based on the Newton-Kantorovich/pseudospectral approach," *Aerospace Science and Technology*, vol. 66, pp. 140–151, 2017.

[48] N. B. LaFarge, D. Miller, K. C. Howell, and R. Linares, "Guidance for closed-loop transfers using reinforcement learning with application to libration point orbits," *AIAA Scitech 2020 Forum*, 2020.

[49] D. Silver, S. Singh, D. Precup, and R. S. Sutton, "Reward is enough," *Artificial Intelligence*, vol. 299, p. 103535, 2021.

[50] A. Y. Ng, *Shaping and Policy Search in Reinforcement Learning*, University of California, Berkeley, 2003.

[51] H. Zeming, H. Hu, Y. He, Z. Ran, and L. Huifeng, *Feasible-Orbit-Set Generation for Launch Vehicles*, Advances in Guidance, Navigation and Control, Singapore, 2022.

[52] S. Zhengyu, C. Wang, and G. Qinghai, "Joint dynamic optimization of the target orbit and flight trajectory of a launch vehicle based on state-triggered indices," *Acta Astronautica*, vol. 174, pp. 82–93, 2020.