# Reentry trajectory optimization based on Deep Reinforcement Learning

Jiashi Gao[1],Xinming Shi[1],Zhongtao Cheng[1],Jizhang Xiong[1],Lei Liu[2], Yongji Wang[3],Ye Yang[4]

1. Huazhong University of Science & Technology, Wuhan,430074
E-mail: uestzaoan@163.com

2. Huazhong University of Science & Technology, Wuhan,430074
E-mail: liulei@mail.hust.edu.cn

3. Huazhong University of Science & Technology, Wuhan,430074
E-mail: wangyjch@hust.edu.cn

4. Science and Technology on Aerospace Intelligent Control Laboratory, Beijing Aerospace Automatic Control Institute, No.50 Yong-ding Road, Haidian Disctrict, Beijing, 100854, China

**Abstract:** This article solved the reentry optimization problem of RLV using the Deep Reinforcement Learning-Deep Deterministic Policy Gradient (DDPG) method for continuous system decision making. Compared with the traditional intelligent optimization algorithm, the DDPG algorithm trains appropriate action values for each state value during flight by constructing the action neural network and the critic neural network, avoiding the problems caused by the improper segmentation of traditional intelligent algorithms. And through the greedy algorithm, the optimization process is prevented from falling into local optimum. By comparing the trajectory optimization results with the particle swarm optimization algorithm, the effectiveness of the DDPG algorithm is verified. At the same time, the optimized trajectory of the DDPG algorithm has better smoothness, and the optimization process is not easy to fall into the local maximum.

**Key Words:** Reentry trajectory optimization, Deep Reinforcement Learning

## 1 INTRODUCTION

Trajectory optimization is a kind of optimal control problem, which belongs to the complex large-scale nonlinear optimization category. Traditional trajectory optimization design methods mainly include indirect method based on maximum value principle and direct method based on nonlinear programming theory. Most of these methods obtain a complete control curve during flight. The various disturbances encountered during the flight caused flight state to fluctuate. Due to the low dependence of the traditional trajectory optimization method on the control value and the state value, the adaptability of the optimization algorithm in actual flight needs to be improved.

In recent years, a decision-making algorithm-enhanced learning algorithm has developed rapidly. The enhanced learning algorithm is an algorithm that selects the optimal action according to the state-action value function and updates the learning network through the rewards brought by the action selection. The decision-making mechanism for enhanced learning can well meet the requirements of adaptability for aircraft trajectory optimization.

In order to adapt reinforcement learning to solving complex, high-dimensional, sensory input problems, Krizhevsky et al. proposed a DQN algorithm combined with deep learning. Deep neural network function approximators were used to estimate the action-value function[1].Prior to DQN, it was generally believed that learning value functions using large, non-linear function approximators was difficult and unstable. DQN is able to learn value functions using such function approximators in a stable and robust way due to two innovations:1)the network is trained off-policy with samples from a replay buffer to minimize correlations between samples; 2)the network is trained with a target Q network to give consistent targets during temporal difference backups. In this work we make use of the same ideas, along with batch normalization[2], a recent advance in deep learning. However, although DQN can solve the problems of high-dimensional observation spaces, it can only be used to solve the problem of discrete, low-dimensional state spaces. Since DQN obtains the optimal action in a certain state by maximizing the state-action value function, it cannot be used for continuous problems, and many tasks, such as trajectory optimization, are continuous and even high-dimensional action spaces.

In order to solve the problem of reinforcement learning in continuous control, Timothy P. Lillicrap et al. designed an algorithm based on DPG called DDPG (deep deterministic policy gradient)[3] .The DDPG algorithm introduces actor and critic neural networks, and updates network parameters through Q update mechanism and gradient descent method. DDPG has high stability and robustness in solving multiple challenging physical control problems.

In this paper, the idea of DDPG is applied to the trajectory optimization problem of spacecraft, and the effectiveness of this idea in trajectory optimization decision is verified by simulation. The structure of the article is as follows: The second part introduces the RLV reentry segment model used. The third part introduces the application of DDPG in the trajectory optimization problem, and gives the specific process of trajectory optimization. The fourth part is the analysis of simulation results, comparing the advantages

This work was supported in part by the National Natural Science Foundation of China (Grants nos. 61873319, 61803162 and 61573161).

and disadvantages of particle swarm optimization algorithm in intelligent optimization algorithm and DDPG, and verifying the effectiveness of DDPG.

## 2 RLV REENTRY TRAJECTORY MODEL

The simplified reentry motion equation in the windless environment is[4]:

$$\begin{aligned} \dot{r} &= V\sin\gamma \\ \dot{V} &= -g\sin\gamma - \frac{D}{M} \\ \gamma &= \left(\frac{V}{r} - \frac{g}{V}\right)\cos\gamma + \frac{L}{MV}\cos\sigma \end{aligned} \tag{1}$$

The control vector is $U = [\alpha, \sigma]$, control vector is the action vector in reinforcement learning. The drag, the lift, the density and the gravity acceleration are assumed to have standard form:

$$\begin{aligned} D &= \frac{1}{2}\rho V^2 C_D S \\ L &= \frac{1}{2}\rho V^2 C_L S \\ \rho &= \rho_0 e^{-\beta h} \\ g &= g_0\left(\frac{r_0}{r}\right)^2 \\ r &= r_0 + h \end{aligned} \tag{2}$$

The aerodynamic parameters can be approximated by the following polynomial:

$$\begin{aligned} C_D &= C_{D0} + kC_L^2 \\ C_{D0} &= 0.4, k = 0.18 \\ C_L &= C_{L\alpha}\alpha \\ C_{L\alpha} &= 7.0 / rad \end{aligned}$$

The state quantity of the flight model is $S = [r, V, \gamma]$, the control quantity is $U = [\alpha, \sigma]$, corresponding to the state vector and action vector in reinforcement learning respectively. Since the aircraft reentry problem is a terminal fixed problem, the satisfaction of the terminal constraint is taken as the reward value required by the critic network. this article, the terminal constraints are $h_f = 12000m, V_f = 450m/s$.

The flight process of the reentry segment must meet the process constraints. In this paper, consider the dynamic pressure constraints and heat flow constraints during flight. The heat flux is given by:

$$q = \frac{18300}{\sqrt{R_N}}(\rho)^{0.5}\left(\frac{V}{10^4}\right)^{3.05}, \quad q_{\max} = 18.5W/cm^2 \tag{3}$$

The dynamic pressure is given by:

$$Q = \frac{\rho V^2}{2}, \quad Q_{\max} = 50kPa \tag{4}$$

The reward is obtained by:

$$reward = \begin{cases} \dfrac{-|q - q_{\max}| - |Q - Q_{\max}|}{10}, & \text{do not satisfy processconstraints} \\ 0, & \text{Meet process constraints and do not satisfy terminal constraints} \\ 100 - \dfrac{|h - h_f|}{\Delta H} - \dfrac{|V - V_f|}{\Delta V}, & \text{Meet terminal constraints and process constraints} \end{cases}$$

Where $\Delta H$, $\Delta V$ are the allowed terminal error thresholds and their selection must guarantee:

$$100 - \frac{|h - h_f|}{\Delta H} - \frac{|V - V_f|}{\Delta V} \geq 0$$

## 3 DDPG CONSTRUCTION AND TRAINING

The main point of the DDPG algorithm is the introduction of the actor neural network and the critic neural network, which are used to select actions and approximate Q. Among them, the actor neural network is updated by the policy gradient method, and the Q method is updated by the Bellman equation. This is also the origin of the DDPG algorithm naming. Figure 1 shows the overview of RLV reentry trajectory optimization under DDPG:

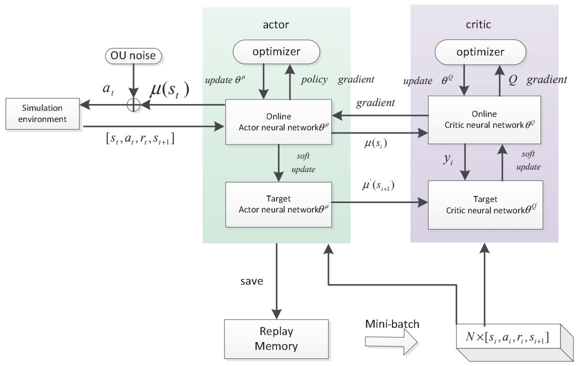

Fig 1. The overview of RLV reentry trajectory optimization under DDPG

Define Q as the cumulative reward on the specified state and the specified action:

$$Q(s_t, a_t) = \int_t^\infty \gamma^{-(\kappa - t)} r(s_t, a_t) d\kappa$$

Where $r$ is the real-time reward for performing the action a in the states. According to the Bellman equation:

$$Q(s_t, a_t) = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1})$$

We approximate $Q(s_t, a_t)$ with a critic neural network approximation $Q(s_t, a_t | \theta^Q)$ parameterized by $\theta^Q$, which we optimize by minimizing the loss $L(\theta^Q)$:

$$L(\theta^Q) = \frac{1}{2}(y_t - Q(s_t, a_t | \theta^Q))^2$$

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, a_{t+1} | \theta^Q)$$

Since $Q(s_t, a_t | \theta^Q)$ and $Q(s_{t+1}, a_{t+1} | \theta^Q)$ use a neural network output in the update formula of $L(\theta^Q)$, this practice has proved to be unstable in many environments, so we established target network, which is modified for actor-critic and using "soft" or "hard" target updates, rather than directly copying the weights. Establish neural networks $Q'(s_t, a_t | \theta^{Q'})$ and $\mu'(s_t | \theta^{\mu'})$ with the same structure as $Q(s_t, a_t | \theta^Q)$, $\mu(s_t | \theta^\mu)$, respectively, for calculating the target value. In this paper, the weights of the target neural network are updated by the soft method, as follows:

$$\theta' \leftarrow \tau\theta + (1-\tau)\theta', \tau \leq 1$$

The soft method causes the target value to be updated slowly, although this may reduce the learning rate, but it improves the stability of learning.

When we get the critic function through the critic neural network, we use it to update the actor neural network. The actor is updated by following the applying the chain rule to the expected return from the start distribution $J$ with respect to the actor parameters:

$$\nabla_{\theta^\mu} J = \frac{\partial Q(s_t, \mu(s_t \mid \theta^\mu) \mid \theta^Q)}{\partial a_t} \cdot \frac{\partial \mu(s_t \mid \theta^\mu)}{\partial \theta^\mu}$$

Due to the large difference in magnitude between state quantities in the trajectory optimization task of the reentry segment, this will cause difficulty in neural network learning. Therefore, before the neural network training, the state quantity $[r, V, \gamma]$ in the aircraft motion model should be normalized by:

$$T = \sqrt{\frac{R_0^{\ 3}}{\mu_E}}$$

Since the training process of most optimization algorithms assumes that the training samples are independently and identically distributed, if the training is performed according to the real-time state value of the flight process output, the above assumption is violated, in order to solve the above problem, it is essential to learn in mini-batches, rather than online. In this paper, we have established a replay buffer of 20,000 in size, taking into account the diversity and complexity of flight conditions. State transition training samples were sampled from replay buffer whose compositions are $[s\_old, action, reward, s\_new]$ according to the environment. When the replay buffer was full, the oldest samples were discarded. At each time-step the actor and critic are updated by sampling a mini-batch uniformly from the buffer.

***Algorithm 1 DDPG in reentry***

```
● Initialize $[r_0, V_0, \gamma_0]$ and normalized
● Initialize $Q(s, a \mid \theta^Q)$ and $\mu(s \mid \theta^\mu)$ with weights $\theta^Q$ and $\theta^\mu$
● Initialize target network $Q'(s, a \mid \theta^{Q'})$ and $\mu'(s \mid \theta^{\mu'})$ with $\theta^{Q'} \leftarrow \theta^Q$ , $\theta^{\mu'} \leftarrow \theta^\mu$
● Initial replay buffer size R and sample size N
● For episodes =1,…,M do:
    ➢ Initialize $[r_0, V_0, \gamma_0]$ and normalized
    ➢ For steps=1,T,do:
        ✧ Choose action according to greedy policy: $a_t = \mu(s_t \mid \theta^\mu) + \mathrm{N}_t$
        ✧ Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$ according to the equation of motion and terminal constraints
        ✧ Store transition tuple $[s_t, a_t, r_t, s_{t+1}]$ in replay buffer
        ✧ Sample a mini-batch of N transitions $[s_i, a_i, r_i, s_{i+1}]$ from relay buffer randomly
        ✧ Compute:
          $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1} \mid \theta^{\mu'}) \mid \theta^{Q'})$
        ✧ Update critic neural network by minimizing:
          $L = \frac{1}{N}\sum_{i=1}^{N}(y_i - Q(s_i, a_i \mid \theta^Q))^2$
        ✧ Compute:
          $\nabla_{\theta^\mu} J = \frac{1}{N}\sum_{i}^{N} \frac{\partial Q(s_i, \mu(s_i \mid \theta^\mu) \mid \theta^Q)}{\partial a_i} \cdot \frac{\partial \mu(s_i \mid \theta^\mu)}{\partial \theta^\mu}$
        ✧ Update the action neural network using the sampled policy gradient $\nabla_{\theta^\mu} J$
        ✧ Update critic and action target network:
          $\theta^{Q'} \leftarrow \tau\theta^Q + (1-\tau)\theta^{Q'}$
          $\theta^{\mu'} \leftarrow \tau\theta^\mu + (1-\tau)\theta^{\mu'}$
    ➢ End for
● End for
```

## 4 SIMULATION

To verify the performance of DDPG in trajectory optimization, in this section, we present the optimization results of the DDPG algorithm and the PSO algorithm in a simplified reentry model under the Tensorflow/Python environment.

Table 1 gives the parameters of the reentry model.

Table1. Parameters of RLV

| Parameter name | Value |
|---|---|
| Mass | 1750kg |
| S | 0.785m$^2$ |
| $R_0$ | 6378000m |
| $\rho_0$ | $1.225\ kg/m^3$ |
| $R_N$ | 0.25m |
| $[r_0, v_0, \gamma_0]$ | [150000,1500,0] |

Considering the high dimension of the state vector and the input vector in the trajectory optimization problem, the neural network parameters are selected as follows: both the actor and critic neural networks have 3 hidden layers. All layers are fully connected, he activate function between the hidden layers is Relu and the layers that between the hidden layer and the output layer is tanh. Each layer has 300 neurons and 300 biases.

The learning rates of actor and critic networks are set at 0.001 which is proper for the reentry trajectory optimization. Reward attenuation factor is 0.99.The capacity of replay memory is set to 20000. In order to improve the computational efficiency of the computer, the batch size is set to 256.

Reentry trajectory planning requires time within 300s, so the training time is set at 300 for each training episode. Too few training steps will result in too few states, making it impossible to train the neural network effectively, and too high training time will slow down the training and even stop learning.

The parameters of the particle swarm algorithm are set as follows: The number of population is 100 and the number of iterations is 100.The angle of attack and the angles of inclination are divided into 20 segments for interpolation.

The following figure 2-4 shows the optimization state curve of reentry trajectory optimization using DDPG and PSO, respectively. Compare with the simulation results in [4], simulation results confirm the effectiveness of DDPG in trajectory optimization. Figure 5 indicates reward in relation to training episodes. When the reward is above 80, the trajectory optimization can be considered to meet the requirements. It can be seen that as the training progresses, the optimization success rate is improved. And when the number of training is greater than 800, it has a good training success rate. Compared with the PSO algorithm, DDPG takes more time to optimize the trajectory than PSO because of the many training states. However, since DDPG selects the action value in real time in trajectory optimization, it has higher optimization precision than PSO. At the same time, DDPG can better avoid local optimization than PSO and parameters can be reused.

Table2. Simulation results of DDPG and PSO

| *Index Algorithm* | *DDPG* | *PSO* |
|---|---|---|
| Training time | 421s | 243s |
| Terminal height | 11987m | 12039m |
| Terminal speed | 451.2m/s | 461.3m/s |

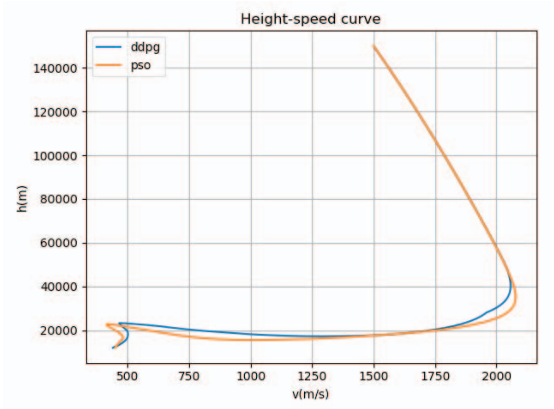

Fig 2. h-v relationship diagram of reentry trajectory based on DDPG and PSO

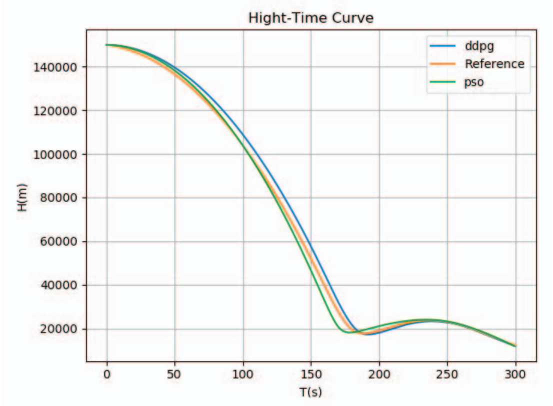

Fig 3. h-t relationship diagram of reentry trajectory based on DDPG and PSO

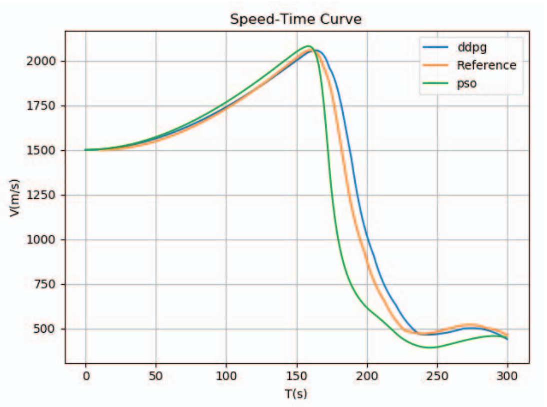

Fig 4. v-t relationship diagram of reentry trajectory based on DDPG and PSO

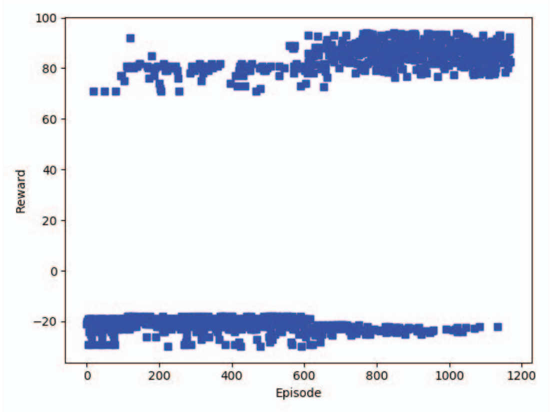

Fig 5.Average reward value of DDPG-iteration number scatter plot

## 5 CONCLUSION

This paper solves the trajectory problem in reentry flight by DDPG algorithm. By comparing the simulation results with PSO, the feasibility of DDPG in continuous nonlinear optimization problem is verified. The simulation results show that DDPG requires a long training time in order to obtain sufficient flight states, but its training result has good precision. At the same time, since DDPG realizes state-action mapping, the disadvantage of being easily trapped in local optimum can be easily avoid and also has good robustness when flight state deviates due to fluctuations.

## REFERENCES

[1] Runsheng Yu*, Zhenyu Shi*, Chaoxing Huang*, Tenglong Li*, Qiongxiong Ma. ”Deep Reinforcement Learning Based Optimal Trajectory Tracking Control of Autonomous Underwater Vehicle”. Proceedings of the 36th Chinese Control Conference July 26-28, 2017, Dalian, China

[2] Lillicrap, Timothy P., et al.”Contious control with DRL.”arXiv preprint arXic:1509.02971(2015)

[3] Timothy P. Lillicrap,et al. “CONTINUOUS CONTROL WIT H DEEP REINFORCEMENT LEARNIN-G”.Google Deepmind

[4] Rajesh Kumar Arora.”Reentry Trajectory Optimization : Evolutionary Approach”.9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization 4-6 September 2002

[5] C.R. Margraves*.”Direct Trajectory Optimization Using Nonlinear Programming and Collocation”.J. GUIDANCE.VOL. 10, NO. 4

[6] Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[7] Silver, David, et al. ”Deterministic policy gradient algorithms.” Proceedings of the 31st International Conference on Machine Learning . 2014.

[8] Konda, Vijay R., and John N. Tsitsiklis. ”Actor-Critic Algorithms.” NIPS. Vol. 13. 1999

[9] Kingma, Diederik, and Jimmy Ba. ”Adam: A method for stochastic optimization.” arXiv preprint arXiv:1412.6980 (2014).

[10] Dean, Jeffrey, et al. ”Large scale distributed deep networks.” Advances in neural information processing systems. 2012.

[11] Balduzzi, David and Ghifary, Muhammad. Compatible value gradients for reinforcement learning of continuous deep policies. arXiv preprint arXiv:1509.03005, 2015.

[12] Heess, N., Hunt, J. J, Lillicrap, T. P, and Silver, D. Memory-based control with recurrent neural networks. NIPS Deep Reinforcement Learning Workshop (arXiv:1512.04455), 2015.

[13] Levine, Sergey, Finn, Chelsea, Darrell, Trevor, and Abbeel, Pieter. End-to-end training of deep visuomotor policies. arXiv preprint arXiv:1504.00702, 2015.

[14] Curtis Zimmerman,”Automated Method to Compute Orbital Reentry Trajectories with Heating Constraints ”,JOURNAL OF GUIDANCE, CONTROL, AND DYNAMICS Vol. 26, No. 4, July–August 2003