Paper:

The Trajectory Generation via Deep Neural Network for Hypersonic Vehicle

Tirui Ma[*,†] ,Yongzhi Sheng[*],and Xiaoping Zhang[**]

[*] School of Automation, Beijing Institute of technology
No.5 South Zhongguancun Street, Haidian District, Beijing, China
E-mail: 2643375752@qq.com
[**]School of Electrical and Control Engineering, North China University of Technology
No. 5 Jinyuanzhuang Road, Shijingshan District, Beijing, China
E-mail: zhangxp@ncut.edu.cn

Abstract. In this paper, we propose a method based on deep neural networks to generate the reentry trajectory of hypersonic vehicle. There are some complex constraints in the reentry trajectory optimization problem, such as dynamic pressure constraint and no-fly zones for threat avoidance. It's hard to present the real-time controls by traditional optimal control methods, due to the limitation of onboard computer. In order to obtain the training data, we adopt gauss pseudo-spectral method (GPM) to compute the optimal trajectories, which contain the optimal states and controls. Deep neural network (DNN) are trained with the optimal states and controls. The reentry trajectory driven by the trained DNN is efficient, and the trained neural networks can present the real-time optimal controls, according to the reentry states of hypersonic vehicle.

Keywords: trajectory generation, deep neural network, hypersonic vehicle

1. Introduction

The performance of the hypersonic vehicle is directly affect by the reentry trajectory. Those typical methods to solve trajectory optimization can be divided into direct methods and indirect methods. Using the direct method to solve reentry trajectory optimization problem is a very tedious task, because it has many complex constraints. Over the past decade, Gauss Pseudo-spectral Method (GPM), one of the indirect methods, is widely adopted to deal with this problem. Yang et.al.[1] presented a GPM with the improved mesh refinement algorithm to the reentry trajectory optimization problem with multiple path and terminal constraints for the three-degree-of-freedom model of a hypersonic vehicle X-33. They prove the effectiveness of GPM to solve the reentry trajectory optimization with typical constrains. Zhang Y. et.al. use the GPM to generate an optimal trajectory for unmanned combat aerial vehicles[2]. However, it is hard to provide the real-time control for the limitation of onboard computer.

In recent years, many researchers aimed to propose new methods to solve this problem. Thanks to the success of neural networks in classification problem, the methods based on neural networks are presented to solve the optimal control problem. Effati and Pakdaman proposed the artificial neural networks (ANNs) to approximate the solution of Hamiltonian conditions based on the Pontryagin minimum principle (PMP)[3]. The weights of ANNs can be trained with an error function which contains the PMP conditions. Levine proposed a deep and recurrent neural network to represent a control policy for a continuous, high-dimensional locomotion task[4]. He has successfully trained the neural network controllers with thousands of parameters. Carlos et.al. used the deep artificial neural networks to approximate the optimal state-feedback control of continuous time, deterministic, non-linear systems[5]. They found that deep networks significantly outperform shallow networks in the ability to build an accurate functional representation of the optimal control. Zhu et.al presented a real-time optimal thrust controller during lunar landing with deep neural networks[6]. This work offers the possibility to get an approximate solution of co-state equation by the deep architectures, without time-consuming iterative process. A deep recurrent neural network (RNN) controller was presented to control a sophisticated and highly nonlinear flight vehicle[7]. Compared to a traditional gain-scheduled LQR controller, the RNN controller has better robustness. This work proved the effectiveness of RNN controller for flight control applications. A deep neural network is successfully trained to present the solution to the Hamilton–Jacobi–Bellman policy equation in four different cases of pinpoint landing[8]. Those studies provide a possibility for the application of neural network in aerospace domain.

In this paper, we attempted to propose a real-time controller based on the DNN to generate reentry trajectory for hypersonic vehicle. The network is trained with the states and controls of optimal trajectories to approximate the optimal solutions. In Section II, we introduce the reentry trajectory optimization problem, and give the typical constrains of this problem. In Section III, we describe the architecture of the deep neural network and the method to train the network. A dataset to train the network is obtained with GPM. In the following Section IV, the reentry trajectories generated by the DNN and GPM is compared with each other. And we prove the performance of the networks.

2. Reentry Trajectory Optimization Problem

2.1. Mathematical model

The vehicle model considered in this paper is proposed by Shaughnessy et al[9], which is a winged-cone configuration. In order to simplify the dynamic model, we assume the earth is a nonrotating and symmetrical sphere, and the vehicle can be regarded as a point. The longitudinal dynamical model of hypersonic vehicle can be described as:

$$\begin{aligned} h &= V\sin\gamma \\ \phi &= \frac{V\cos\gamma}{R_e+h} \\ \dot{V} &= \frac{-D}{m}-\frac{\mu\sin\gamma}{\left(R_e+h\right)^2} \\ \dot{\gamma} &= \frac{L}{mV}+\left(\frac{V}{R_e+h}-\frac{\mu}{V\left(R_e+h\right)^2}\right)\cos\gamma \end{aligned} \tag{1}$$

The state vector can be defined as $\boldsymbol{x} \triangleq \left[h,\phi,V,\gamma\right]^T \in \mathbb{R}^4$,whose components are altitude, range angle, velocity and flight path angle of vehicle, respectively. R_e is the radius of earth, m is the mass of vehicle, and μ is the gravitational constant. D and L are the drag and lift of vehicle, respectively, and can be expressed as:

$$\begin{aligned} D &= \frac{1}{2}\rho V^2 S * C_D \\ L &= \frac{1}{2}\rho V^2 S * C_L \end{aligned} \tag{2}$$

where S is the reference area of hypersonic vehicle, and ρ is the atmospheric density that can be described as $\rho=\rho_0 e^{-\beta h}$, ρ_0 and β are the atmospheric density at sea level and the density coefficient, respectively. C_D and C_L are the drag and lift coefficients. They can be approximated using curve-fitting techniques and expressed respectively as:

$$\begin{aligned} C_D &= C_{D0}+C_{D1}\alpha+C_{D2}\alpha^2+C_{D3}V \\ C_L &= C_{L0}+C_{L1}\alpha+C_{L2}V \end{aligned} \tag{3}$$

The control variable u is the angle of attack α . More details can be found in[10].

2.2. Optimal control problem of Hypersonic vehicle

In this paper, we consider the following type of optimal control problem. Maximizing the velocity can improve the performance of the re-entry process of hypersonic vehicle, when the range angle is given. Therefore, it's to find the optimal control α over t in the time horizon $[t_0,t_f]$ to maximize the velocity, and the cost function J is given by:

$$\max \quad J = V(t_f) \tag{4}$$

subject to the Eq. (1), and the following constraints, the control constraint:

$$\alpha_{\min} \le \alpha \le \alpha_{\max} \tag{5}$$

the boundary conditions:

$$\begin{aligned} \boldsymbol{x}_0 &= \left[h(t_0),\phi(t_0),V(t_0),\gamma(t_0)\right]^T \\ \boldsymbol{x}_f &= \left[h(t_f),\phi(t_f),\ free,\ free\right]^T \end{aligned} \tag{6}$$

the path conditions:

$$\begin{aligned} Q &= 5.188\times10^{-9}\rho^{0.5}V^3 \le Q_{\max} \\ q &= \frac{1}{2}\rho V^2 \le q_{\max} \\ N &= \frac{\sqrt{L^2+D^2}}{mg} \le N_{\max} \end{aligned} \tag{7}$$

where $\alpha_{\min}$ and $\alpha_{\max}$ are the minimum and maximum of angle of attack. Q is the heating rate and the maximum is $Q_{\max}$. q is the dynamic pressure, and it must be less than the maximum value $q_{\max}$. N is the aerodynamic load and the maximum is $N_{\max}$. g is the gravity coefficient at the surface of the earth.

Without loss of the generality, the optimal control problem of hypersonic vehicle can be considered as the two-point boundary value problem (TPBVP). It can be summarized as the following general form:

$$\begin{aligned} \min \quad & J = \varphi\left(\boldsymbol{x}(t_0),t_0,\boldsymbol{x}(t_f),t_f\right) \\ s.t. \quad & \dot{\boldsymbol{x}} = f\left(\boldsymbol{x}(t),u(t),t\right) \\ & \mathbf{B}\left(\boldsymbol{x}(t_0),t_0,\boldsymbol{x}(t_f),t_f\right)=0 \\ & \mathbf{C}\left(\boldsymbol{x}(t),u(t),t\right)\le 0 \\ & t_0 \le t \le t_f \end{aligned} \tag{8}$$

The Eq. (8) can be gotten from the cost function, dynamical model, boundary constraints, path constraints and control constraint respectively.

3. Deep Neural Networks for the Optimal Control

In this section, we provide the network architecture, training data and training method to approximate the optimal solutions. The optimal control system based on deep neural networks is shown in Fig.1. The deep neural networks are trained with the data of optimal trajectories. And it will produce a control command, according to the states of hypersonic vehicle.

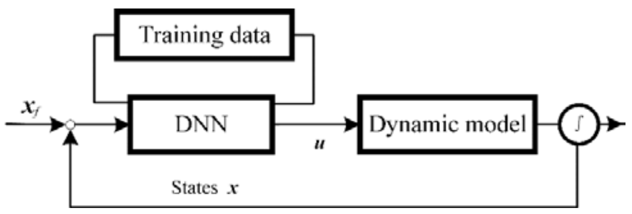

Fig. 1. Optimal control policy

3.1. DNN Architecture

As shown in Fig.2, the deep neural networks architecture is proposed. It's a simple feedforward architecture and has seven layers, which include five hidden layers. There are six inputs, one output and five hundred units in the hidden layer. The next section describes the inputs $\tilde{x}$ in more detail.

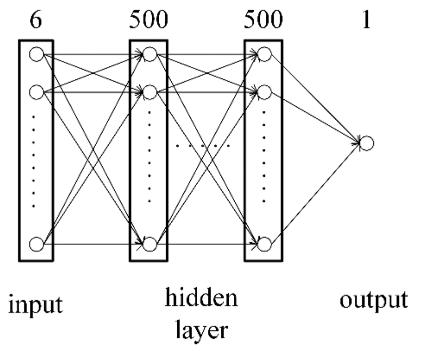

Fig. 2. Deep neural networks architecture

The output z_{ij} of the unit i in the hidden layer j can be described as the following form:

$$z_{ij} = \sigma\left(\boldsymbol{w}_{ij}\mathbf{z}_{j-1} + b_{ij}\right) \tag{9}$$

where $\boldsymbol{w}_{ij}$ is the vector of weights, $\mathbf{z}_{j-1}$ is the output vector of the previous layer, b_{ij} the bias corresponding to this unit, $\sigma(\bullet)$ is the activation function. The activation function is the important component of DNN architecture, and it has many different kinds.

The rectified linear unit (ReLU) and sigmoid function are common function. The ReLU can avoid the vanishing gradient problem, compared to sigmoid function[11]. In this paper, we select two different kinds unit for the hidden layers and output layer. For the hidden layers, we use the sigmoid function as the activation function for units. And we select the ReLUs at the output layers.

3.2. Generating Optimal Trajectories and Learning the Optimal Control

3.2.1. Generating Optimal Trajectories

In order to training the DNN, we build a dataset that contain optimal trajectories. Each optimal trajectory consists of the optimal states $\boldsymbol{x}^*$ and optimal control $\boldsymbol{u}^*$. For the reentry trajectory optimization problem, we give a lots of different initial and terminal conditions. Table. 1 is shows the range of some initial and terminal conditions. The initial range angle is constant zero. The boundary constraints of the optimal control problem will be given, when the initial conditions and terminal conditions are determined.

Table. 1 The range of initial and terminal conditions.

Initial altitude	*Initial velocity*	*Initial flight path*	*Terminal altitude*	*Terminal range angle*
55~ 65*km*	5~ 7*km/s*	0 ~ -10°	30~ 35*km*	80 ~ 110°

Then we use GPM to compute the optimal trajectory. As a numerical technique, GPM transforms the optimal control problem to a nonlinear programming problem (NLP) by using discrete approximation[12]. The Eq. (8) can be described as the Bolza form as follows:

$$\begin{aligned}
\min \quad & J = \varphi\left(\boldsymbol{x}(\tau_0), \tau_0, \boldsymbol{x}(\tau_f), \tau_f\right) \\
s.t. \quad & \dot{\boldsymbol{x}} = \frac{t_f - t_0}{2} f\left(\boldsymbol{x}(\tau), u(\tau), \tau, t_0, t_f\right) \\
& \mathbf{B}\left(\boldsymbol{x}(\tau_0), t_0, \boldsymbol{x}(\tau_f), \tau_f\right) = 0 \\
& \mathbf{C}\left(\boldsymbol{x}(\tau), u(\tau), \tau, t_0, t_f\right) \le 0 \\
& \tau_0 = -1 \le \tau \le \tau_f = 1
\end{aligned} \tag{10}$$

where τ is defined as the following form:

$$\tau = \frac{2t}{t_f - t_0} - \frac{t_f + t_0}{t_f - t_0} \tag{11}$$

The state $\boldsymbol{x}(\tau)$ and control $\boldsymbol{u}(\tau)$ can be approximated by the discrete states $\mathbf{x}(\tau_i)$ $(i = 0, 1, \cdots, n)$ and the discrete controls $\mathbf{u}(\tau_i)$ $(i = 1, 2, \cdots, n)$ as follows:

$$\begin{aligned}
& \boldsymbol{x}(\tau) \approx \mathbf{x}(\tau) = \sum_{i=0}^{n} L_i(\tau)\mathbf{x}(\tau_i) \\
& L_i(\tau) = \prod_{j=0, j\ne i}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j} \qquad (i = 0, 1, 2 \cdots, n) \\
& \boldsymbol{u}(\tau) \approx \mathbf{u}(\tau) = \sum_{i=0}^{n} L_i'(\tau)\mathbf{u}(\tau_i) \\
& L_i'(\tau) = \prod_{j=1, j\ne i}^{n} \frac{\tau - \tau_j}{\tau_i - \tau_j} \qquad (i = 1, 2 \cdots, n)
\end{aligned} \tag{12}$$

Where $L_i(\tau)$ and $L_i'(\tau)$ are the basic of Lagrange polynomials with degree n and $n+1$. The discrete states $\mathbf{x}(\tau_i)$ $(i=0,1,\cdots,n)$ consists the initial states and discrete states at the Legendre–Gauss nodes $\tau_k(k=1,2,\cdots,n)$, and the discrete controls $\mathbf{u}(\tau_i)$ $(i=1,2,\cdots,n)$ is the controls at the Legendre–Gauss nodes. The Legendre–Gauss nodes are the roots of Legendre polynomial with degree *n*. And the differential equation of states at the Legendre–Gauss node are described as the following form:

$$\dot{\mathbf{x}}(\tau_k) \approx \dot{\mathbf{x}}(\tau_k) = \sum_{i=0}^{n} \dot{L}_i(\tau_k)\mathbf{x}(\tau_i) = \sum_{i=0}^{n} M_{ki}\mathbf{x}(\tau_i) \quad (13)$$
$$k=1,2,\cdots,n$$

where M_{ki} is the elements of differentiation matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n+1}$, and it's given by:

$$M_{ki} = \begin{cases} \dfrac{(1+\tau_k)\dot{\tilde{L}}_n(\tau_k) + \tilde{L}_n(\tau_k)}{(\tau_k-\tau_i)\left[(1+\tau_i)\dot{\tilde{L}}_n(\tau_i) + \tilde{L}_n(\tau_i)\right]} & i \neq k \\ \dfrac{(1+\tau_k)\ddot{\tilde{L}}_n(\tau_i) + 2\dot{\tilde{L}}_n(\tau_i)}{2\left[(1+\tau_i)\dot{\tilde{L}}_n(\tau_i) + \tilde{L}_n(\tau_i)\right]} & i = k \end{cases} \quad (14)$$

the $\tilde{L}_n(\tau)$ is the Legendre polynomial with degree *n*. Consider the boundary constraints, the terminal states should increase an additional constraint[13]:

$$\mathbf{x}(\tau_f) - \mathbf{x}(\tau_0) - \frac{t_f - t_0}{2}\sum_{k=1}^{n}\omega_k f\left(\mathbf{x}(\tau_k), \mathbf{u}(\tau_k), \tau_k, t_0, t_f\right) = 0 \quad (15)$$

where ω_k is the weights of Gauss-Legendre quadrature. To solve optimal control problem, the Eq.(10) can be expressed as a nonlinear programming problem with the state and control approximation equation:

$$\begin{aligned} \min \quad & J = \varphi\left(\mathbf{x}(\tau_0), \tau_0, \mathbf{x}(\tau_f), \tau_f\right) \\ s.t. \quad & \mathbf{x}(\tau_f) - \mathbf{x}(\tau_0) - \frac{t_f - t_0}{2}\sum_{k=1}^{n}\omega_k f\left(\mathbf{x}(\tau_k), \mathbf{u}(\tau_k), \tau_k, t_0, t_f\right) = 0 \\ & \sum_{i=0}^{n} M_{ki}\mathbf{x}(\tau_k) - \frac{t_f - t_0}{2} f\left(\mathbf{x}(\tau_k), \mathbf{u}(\tau_k), \tau_k, t_0, t_f\right) = 0 \\ & \mathbf{B}\left(\mathbf{x}(\tau_0), t_0, \mathbf{x}(\tau_f), t_f\right) = 0 \\ & \mathbf{C}\left(\mathbf{x}(\tau_k), \mathbf{u}(\tau_k), \tau_k, t_0, t_f\right) \le 0 \qquad (k=1,2\cdots,n) \end{aligned} \quad (16)$$

By solving the nonlinear programming problem, we obtain three thousand optimal trajectories. There are some of the optimal trajectories shown in Fig.3. Three hundred flight states and corresponding controls are selected from each optimal trajectory.

Because the terminal location has been determined, we introduce two new state parameters, inspired by the guidance law form. The new state parameters are residual altitude dh and residual range angle $d\phi$, and they can be expressed as follows:

$$\begin{aligned} dh &= h - h_f \\ d\phi &= \phi - \phi_f \end{aligned} \quad (17)$$

where h and ϕ are the current altitude and range angle, respectively, h_f and ϕ_f are the terminal altitude and range angle. Therefore, we define a new state vector $x' = \left[h, \phi, V, \gamma, dh, d\phi\right]^T \in \mathbb{R}^6$. According to the optimal states $\boldsymbol{x}^*$, we can get the new optimal states $\boldsymbol{x}'^*$.

The range of optimal states are different with each other, and it will affect the rate of convergence. In order to improve the convergence speed of network training, the optimal states $\boldsymbol{x}'^*$ are normalized to obtain the network inputs $\tilde{\boldsymbol{x}}$:

$$\tilde{\boldsymbol{x}} = \frac{\boldsymbol{x}'^* - \boldsymbol{x}'^*_{\min}}{\boldsymbol{x}'^*_{\max} - \boldsymbol{x}'^*_{\min}} \quad (18)$$

where $\boldsymbol{x}'^*_{\min}$ and $\boldsymbol{x}'^*_{\max}$ are the minimum and maximum of each optimal state, respectively. Then the train sets are set up with the optimal states $\tilde{\boldsymbol{x}}$ and optimal control $\boldsymbol{u}^*$.

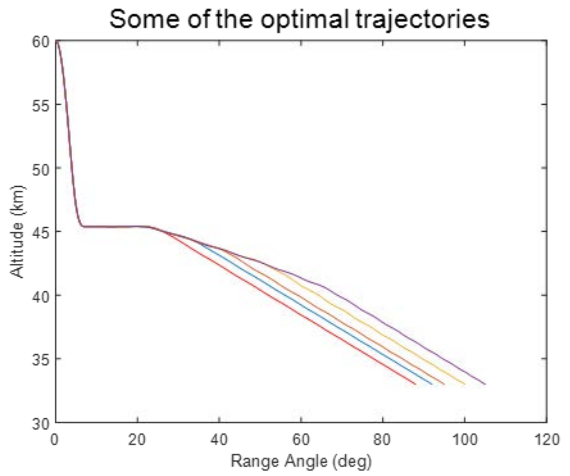

Fig. 3. Some of the optimal trajectories

3.2.2. Training the DNN

Along the training process, we add a dropout layer between each layer to improve training efficiency. Then with dropout layer, the Eq. (9) can be described as:

$$\begin{aligned} \tilde{\mathbf{z}}_{j-1} &= \mathbf{r}_j\tilde{\mathbf{z}}_{j-1} \\ z_{ij} &= \sigma\left(\boldsymbol{w}_{ij}\tilde{\mathbf{z}}_{j-1} + b_{ij}\right) \end{aligned} \quad (19)$$

The vector $\mathbf{r}_j$ contains Bernoulli random variables, which have probability *p* to be 1. More details about dropout layer can be found in[14]. The dropout layer has a good effect on reducing overfitting.

We use the datasets to train the DNN with stochastic gradient descent (SGD) and a batch size of $n=16$. The loss function to be minimized at the training process is given by:

$$J_n = \sum_{i=0}^{n}\frac{1}{n}\left(\mathbf{N}(\tilde{\boldsymbol{x}}_i) - \boldsymbol{u}^*\right)^2 \quad (20)$$

$\mathbf{N}(\tilde{\boldsymbol{x}}_i)$ is the output of DNN, and $\boldsymbol{u}^*$ is the optimal control. Every weight w_i is updated with a learning rate $\kappa = 0.003$ and a momentum $\varepsilon = 0.9$:

$$v_i^t \to v_i^{t+1} = \varepsilon v_i^t - \kappa \frac{\partial J_n}{\partial w_i^t} \qquad (21)$$
$$w_i^t \to w_i^{t+1} = w_i^t + v_i^{t+1}$$

We set 15 epochs to train the DNN with all training data.

4. Results

To verify the validity of the DNN after training, we generate the reentry trajectory with DNN. The initial and terminal conditions is included in Table. 2. It's consistent with the range that we proposed before, but not included in the train sets of optimal trajectories.

Table. 2 The initial and terminal conditions.

Initial altitude	*Initial velocity*	*Initial flight path*	*Terminal altitude*	*Terminal range angle*
60 *km*	5.5*km/s*	0	33*km*	89.5°

As is shown in Fig.4, the control value $\boldsymbol{u}_n$ obtained by the trained DNN successfully approximates to the optimal control $\boldsymbol{u}_g$ obtained by GPM. The maximum error between $\boldsymbol{u}_n$ and $\boldsymbol{u}_g$ is less than 5%. The reentry trajectories generated by the trained DNN and GPM are shown in Fig.5. The terminal states of DNN-driven trajectory are $\boldsymbol{x}_{nf} = \left[33.3km, 89.5^\circ, 2.67\, km/s, -0.1^\circ\right]^T$, and The terminal states of GPM-driven trajectory are $\boldsymbol{x}_{gf} = \left[33.01km, 89.5^\circ, 2.63\, km/s, -0.09^\circ\right]^T$. The error of terminal altitude is 0.3*km*, it's acceptable for the reentry trajectory problem.

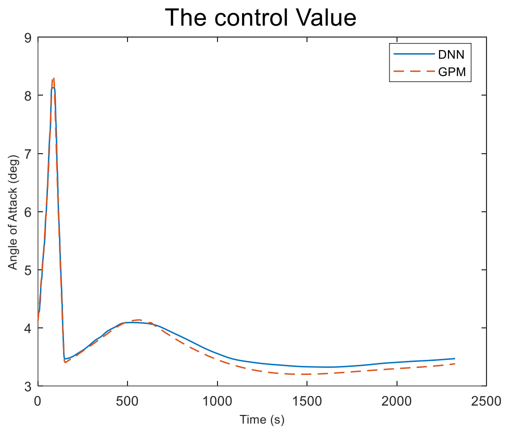

Fig. 4. Angle of Attack

In order to study the performance of the trained DNN, we introduced 10% aerodynamic interference into the simulation of flight system. Then the trained DNN present the control with the real-time states of vehicle. We obtain the trajectories with the trained DNN and the optimal control obtained by GPM, as is shown in Fig.6. The DNN-driven trajectory can still meet the quasi-equilibrium glide condition under the influence of aerodynamic interference, compared with the GPM-driven trajectory. The terminal states of trajectories are $\boldsymbol{x}'_{nf} = \left[33.9km, 89.5^\circ, 2.66\, km/s, -0.11^\circ\right]^T$ and $\boldsymbol{x}'_{gf} = \left[33.7km, 89.35^\circ, 2.62\, km/s, -0.08^\circ\right]^T$, respectively. According to the simulation results, the trained DNN is an effective strategy for generating the reentry trajectory, and it also ensures the robustness of flight system.

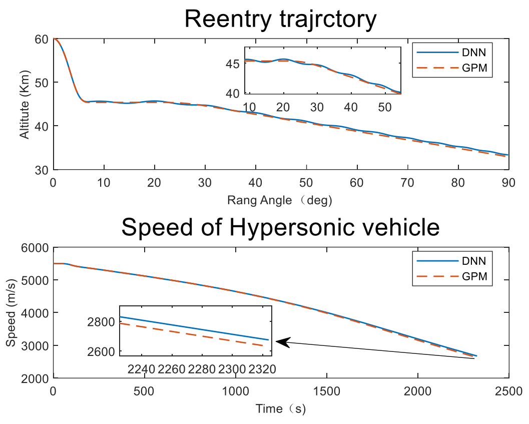

Fig. 5. The Reentry Trajectory

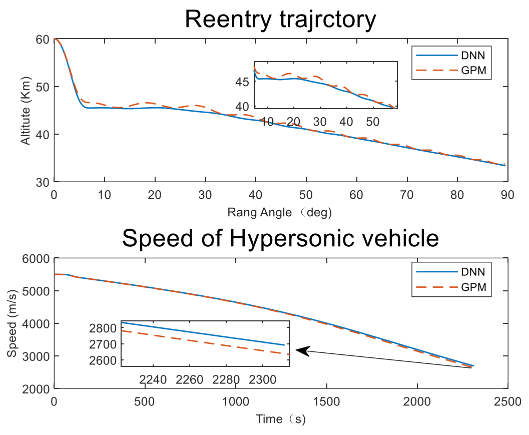

Fig.6. The Reentry Trajectories with Interference

5. Conclusions

In this paper, we have proposed a deep neural network to present a real-time control for generating the reentry trajectory. The performance of the trained DNN is demonstrated with the different initial and terminal states. The trained DNN has good robustness against the aerodynamic interference. We satisfy the reentry trajectory generated by the train DNN. And it is a practicable policy to use the DNN to generate the reentry trajectory.

References:

[1] P. Yang and R. Qi, “Reentry Trajectory Optimization for Hypersonic Vehicle Based on Improved Mesh Refinement Techniques,” 35th Chinese Control Conference (CCC), Chengdu, China, 2016

[2] Y. Zhang, W. Zhang, and J. Chen, “Air-to-ground Weapon Delivery Trajectory Planning for UCAVs Using Gauss Pseudospectral Method,” Acta Aeronautica Et Astronautica Sinica, 32(7), pp. 1240-1251, 2011.

[3] S . Effati and M. Pakdaman, “Optimal Control Problem via Neural Networks,” Neural Computing and Applications, 23, pp. 2093–2100, 2013.

[4] S. Levine, “Exploring Deep and Recurrent Architectures for Optimal Control,”CoRR,Vol. abs/1311.1761, 2013

[5] S. S. Carlos, I. Dario, and D. Hennes . "Learning the optimal state-feedback using deep networks," Computational Intelligence IEEE, 2017.

[6] L. Zhu, J. Ma , and S. Wang . "Deep Neural Networks Based Real-time Optimal Control for Lunar Landing," IOP Conference Series Materials Science and Engineering, 2019

[7] A. N. Scott and P. K. Pramod,” Development of a Robust Deep Recurrent Neural Network Controller for Flight Applications,” American Control Conference, Seattle, USA,pp.5336-5342,2017.

[8] S. S. Carlos and I. Dario, ” Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems,” JOURNAL of GUIDANCE, CONTROL, AND DYNAMICS, 41(5), pp. 1122-1135, 2018

[9] S. Keshmiri, R. Colgren and M. Mirmirani “Development of an aerodynamic database for a generichypersonic air vehicle,”. Proceedings of AIAA Guidance. Navigation. and Control Conference and Exhibit. San Francisco. Califomia. AIAA, 2005.

[10] M. Tava, and S. Suzuki . "Multidisciplinary design optimization of a re-entry vehicle shape and trajectory," Aiaa Journal, 2013.

[11] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks." Journal of Machine Learning Research, 15, pp.315-323, 2011

[12] G. W. Reddien, “Collocation at Gauss Points as a Discretization in Optimal Control,” SIAM Journal on Control and Optimization, 17(2),pp. 298–306, 1979.

[13] J. Zhao, and R. Zhou, "Reentry trajectory optimization for hypersonic vehicle satisfying complex constraints," Chinese Journal of Aeronautics, 26(6), pp. 1544-1553, 2013

[14] N. Srivastava, G. Hinton, A. Krizhevsky et al. “Dropout: A simple way to prevent neural networks from overfitting,” Journal of Machine Learning Research, 15(1), pp.1929-1958, 2014.