aerospace

MDPI

Article

Intelligent Trajectory Generation Method for Hypersonic Glide Vehicles Based on RBF Neural Networks

Feng Yang *, Ziheng Cheng and Chengyu Zhao

School of Mechanics and Aerospace Engineering, Dalian University of Technology, Dalian 116024, China; czh1107@mail.dlut.edu.cn (Z.C.); zcy0803@mail.dlut.edu.cn (C.Z.)
* Correspondence: feng_yang@dlut.edu.cn; Tel.: +86-18610502123

Abstract

In this paper, a radial basis function (RBF) neural network based trajectory generation strategy is proposed to solve the online rapid generation of initial reference trajectory for low-cost hypersonic glide vehicles (HGV) under initial state perturbation. Firstly, the feasible trajectories that constitute the sample sets are offline generated by pseudospectral method according to the possible distribution of heights and velocities. Then, the sample set is randomly divided into training subset and test subset, by which the RBF neural network is trained and verified. Moreover, the input of the RBF neural network is a vector comprised by height and velocity from the initial state, whereas the output is a discrete state-control sequence which represents the trajectory from the current state to the expected final state. The simulation results validate that the proposed method has high confidence and small errors, which can improve the on-line generation efficiency of the trajectory.

Keywords: RBF neural networks; hypersonic glide vehicles; on-line trajectory generation

1. Introduction

The research on low-cost hypersonic vehicles (HGV) has been a new trend in the research area, whose lethality is kept but manufacturing cost is greatly decreased [1,2]. For instance, it is reported that the purchase price of a HGV named Yukongji-1000 (YKJ-1000) may be merely 100,000 dollars [3], and the cost of the 'Blackbeard' hypersonic missile in the United States is much lower than that of the same type of hypersonic vehicle, which cost about 1 million dollars [4]. However, the manufacturing errors of the low-cost HGV maybe amplified during ex-tensive implementation. In another words, the errors of the booster will cause the initial state dispersion of the HGV at the beginning of glide phase. Therefore, there is a critical need to rapidly generate a reference trajectory during reentry that aligns with the current flight mission. Traditional trajectory design methodologies for HGV necessitate extensive offline simulations across numerous operational scenarios and rely on onboard storage of pre-computed trajectories [5], resulting in low computational efficiency and high costs. In recent years, the emergence of intelligent methodologies has offered new solutions for rapid, online trajectory generation. Consequently, the challenge of swiftly generating nominal trajectories for HGV has become an issue that needs to be solved.

Traditional optimal trajectory generation methods, such as the direct shooting method and the pseudospectral method [6–9], are effective tools for offline trajectory design, as they can handle complex multi-constraint problems and generate high-precision optimal solutions. However, these methods rely on a large number of numerical calculations and

Check for updates

Academic Editor: Mingzhe Hou

Received: 16 April 2026
Revised: 10 May 2026
Accepted: 14 May 2026
Published: 19 May 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

iterations, resulting in low computational efficiency that makes it difficult to meet the real-time requirements of an airborne environment and thus limits the possibility of their online deployment [10].

In recent years, the rapid development of artificial intelligence technology has provided a new technical path for online trajectory generation. Among them, the intelligent method based on neural network shows great potential through offline training and online implementation. For example, Chai [11] and Shi [12] use deep neural networks to learn the mapping from flight state to control instructions through offline generated sample sets, thereby achieving rapid trajectory generation. Wang [13] further optimized the network structure and parameter update algorithm to improve performance. Dai [14] employed the pseudospectral convex programming to accelerate the computational efficiency of the sample set, by which a BP neural network is trained to output the control variable for reusable launch vehicles. This method is also applied to rocket ascent problem [15] and hypersonic vehicle reentry problem [16]. Cheng [17] uses Deep Neural Network (DNN) to learn the mapping relationship between flight state and distance, and uses deep neural network and constraint management to realize real-time trajectory generation. Su [18] introduced whale optimization algorithm and extreme learning machine to improve the efficiency of sample generation and network training. However, the underlying implementation logic of the aforementioned methods relies on step-by-step recursive computation. At each integration step, the neural network outputs control, which is then fed into the dynamic model to propagate to the next state. Although different neural networks and different sample set generating methods are used, the essential implementation logic of the abovementioned methods keep the same. Since more computational resources is required to complete the integration step by step, the on-line performance may not satisfy the requirements of the practical vehicle.

As an alternative, reinforcement learning methods are also introduced to optimize online generation efficiency. Wang [19] combined polynomial chaos, convex optimization, and DNN for optimal robust fast trajectory generation. Shi [20] trained neural networks using state-control and state-costate pairs from high-fidelity algorithms to enhance online trajectory generation reliability. Peng [21] utilizes deep reinforcement learning and modular strategy to improve the adaptability of the trajectory generation algorithm. Bao [22,23] utilized pre-trained DNN to accelerate reinforcement learning convergence for rapid trajectory generation. Li [24] designed a method based on Twin Delayed Deep Deterministic policy gradient algorithm (TD3) to generate trajectories that satisfy multiple constraints online for hypersonic vehicles. Su [25] used distributed proximal policy optimization to train networks for optimal control-based fast trajectory generation. Zhu [26] employed deep reinforcement learning to output bank angles for real-time trajectory generation. Wu [27] proposed a TD3-based reinforcement learning framework for fast trajectory generation with no-fly zone avoidance. Compared to supervised learning methods, although the reinforcement learning approach does not depend on the sample trajectories, their practical application is hindered by several factors. Their training is inherently complex, and online trajectory generation remains computationally demanding; moreover, ensuring convergence poses a challenge.

In summary, due to the limited onboard computing resources of low-cost hypersonic glide vehicles, existing intelligent trajectory generation methods still suffer from recursive computational burdens or high training complexity in online computation. To address these issues, this paper proposes a lightweight online trajectory generation strategy based on RBF neural networks. The RBF neural network is chosen for its some advantages ideally suited to resource-constrained HGV: extremely fast forward propagation due to its simple local-activation structure; direct mapping from current state to full future state–control

sequences without recursive integration. The main contributions of this paper can be summarized as follows: firstly, a new online trajectory generation framework is proposed, in which the pseudospectral method is employed to generate optimal trajectory samples using dispersed height and velocity as initial states, thereby establishing a mapping sample library between "state dispersion" and "trajectory sequence". Secondly, based on this sample library, the RBF neural network is trained to learn the mapping from the current height and velocity states to the complete future state-control sequence. This approach avoids the step-by-step recursive computation inherent in traditional methods, significantly improving online generation efficiency. Thirdly, simulation results demonstrate that the trajectories generated by the proposed method exhibit high confidence and precision, effectively accommodating uncertainties during reentry. This provides a feasible solution for online intelligent trajectory generation of hypersonic glide vehicles.

2. Reentry Model Establishment

2.1. Establishment of Three Degree of Freedom Model

In this paper, the dynamic model of the return coordinate system is established with reference to Vinh [28]. The dynamic modeling of the aircraft in the return coordinate system is shown in Figure 1.

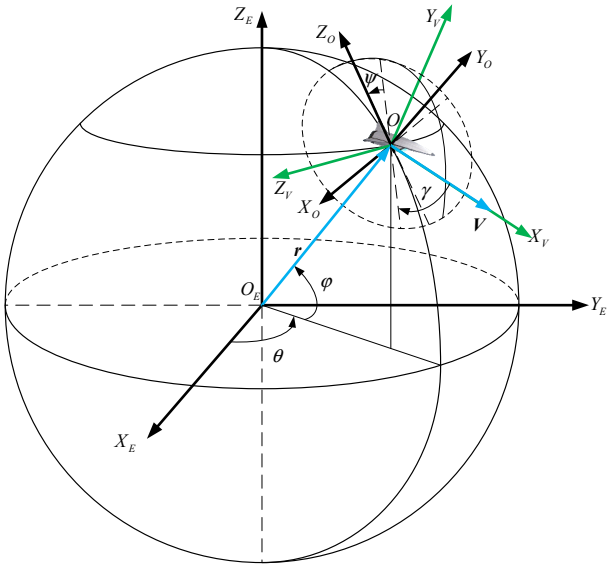


Figure 1. Dynamic modeling of aircraft in return coordinate system. where $O_E X_E Y_E Z_E$ is the geocentric coordinate system, $OX_V Y_V Z_V$ is the velocity coordinate system, and $OX_O Y_O Z_O$ is the return coordinate system, as well as the geometric relationship of key variables such as the aircraft position vector $\boldsymbol{r}$, the velocity vector $\boldsymbol{V}$, longitude θ, latitude φ, flight path angle γ and heading angle ψ during the reentry of the aircraft. They are all marked in the diagram.

The three-dimensional motion equation of the aircraft on the spherical rotating earth is:

$$\begin{aligned}
\dot{r} &= V\sin\gamma \\
\dot{\theta} &= \frac{V\cos\gamma\sin\psi}{r\cos\varphi} \\
\dot{\varphi} &= \frac{V\cos\gamma\cos\psi}{r} \\
\dot{V} &= -\frac{D}{m} - g\sin\gamma + R_V \\
\dot{\gamma} &= \frac{1}{V}\left[\left(\frac{V^2}{r} - g\right)\cos\gamma + \frac{L}{m}\cos\sigma\right] + R_\gamma \\
\dot{\psi} &= \frac{1}{V}\left[\frac{L\sin\sigma}{m\cos\gamma} + \frac{V^2}{r}\cos\gamma\sin\psi\tan\varphi\right] + R_\psi
\end{aligned} \tag{1}$$

where the left side is the time derivative of the state, and the right side includes the aerodynamic term, the gravity term, the centrifugal force term and the Coriolis force term R_V, R_γ, R_ψ.

$$R_V = r\omega_e^2\left(\cos^2\varphi\sin\gamma - \cos\varphi\sin\varphi\cos\gamma\cos\psi\right) \tag{2}$$

$$R_\gamma = \frac{1}{V}[2V\omega_e\cos\varphi\sin\psi + \omega_e^2 r\cos\varphi(\cos\varphi\cos\gamma + \sin\varphi\sin\gamma\cos\psi)] \tag{3}$$

$$R_\psi = \frac{1}{V}\left[-2V\omega_e(\tan\gamma\cos\psi\cos\varphi - \sin\varphi) + \frac{r\omega_e^2}{\cos\gamma}\sin\psi\sin\varphi\cos\varphi\right] \tag{4}$$

The expressions of R_V, R_γ, R_ψ (2)~(4) are derived from the inertial force effect caused by the rotation of the earth.

r is the geocentric distance. θ is the longitude of the projection point of the aircraft on the surface, and φ is the latitude of the projection point of the aircraft on the surface. V is the speed, γ is the flight path angle, and ψ is the heading angle. m is the mass of the aircraft, g is the acceleration of the earth's gravity, σ is the bank angle, and ω_e is the angular velocity of rotation. L and D are lift and drag respectively. Expressed as:

$$L = \frac{\rho V^2 S C_L}{2}, D = \frac{\rho V^2 S C_D}{2} \tag{5}$$

In the formula, ρ is the atmospheric density, which is a function of height, and is calculated by the fitted formula atmospheric model. S is the aerodynamic area of the aircraft, C_L and C_D are lift coefficient and drag coefficient respectively, which are calculated as functions of angle of attack and Mach number in this paper [29].

$$C_L = 0.0513\alpha + 0.2945e^{-0.1028Ma} - 0.2317$$

$$C_D = 7.24\times10^{-4}\alpha^2 + 0.406e^{-0.1028Ma} + 0.024$$

2.2. Reentry Process Constraints

Typical inequality trajectory constraints include

$$q = \frac{1}{2}\rho V^2 \le q_{\max} \tag{6}$$

$$\dot{Q} = k_Q\sqrt{\rho}V^{3.15} \le \dot{Q}_{\max} \tag{7}$$

$$n = \frac{\sqrt{L^2 + D^2}}{mg} \le n_{\max} \tag{8}$$

where Equation (6) is dynamic pressure constraint, the dynamic-pressure limit is $q_{\max}$. The dynamic pressure is a key indicator of aerodynamic load, which affects the structural load. Excessive dynamic pressure may lead to rudder instability or structural overload. Equation (7) is the heating rate constraint at a stagnation point, $k_Q = 9.4369\times10^{-5}$ is a constant, the heating-rate limit is $\dot{Q}_{\max}$. During the reentry process, the surface of the aircraft generates high heat flow due to shock compression and aerodynamic heating. Excessive heat flow may lead to the failure of thermal protection materials. The stagnation heat flow is calculated by Fay-Riddle formula. Equation (8) is overload constraint, the load factor limit is $n_{\max}$. The overload is defined as the ratio of aerodynamic force to gravity, which reflects the load borne by the aircraft structure. Excessive overload may lead to structural damage or affect the normal operation of the equipment. The constraint value is usually determined by the structural strength design of the aircraft [30].

In this paper, the quasi-equilibrium glide condition is derived for the formula of $\dot{\gamma} \approx 0$. The formula is as follows [31]

$$\frac{V}{r} + \frac{(L\cos\sigma - g)}{V} = 0 \tag{9}$$

The terminal constraint has strict position constraint, which is given by the following formula.

$$r\left(t_f\right) = r_f, \theta\left(t_f\right) = \theta_f, \varphi\left(t_f\right) = \varphi_f \tag{10}$$

where r_f, θ_f, φ_f are the terminal geocentric distance, the terminal latitude and longitude respectively.

2.3. Establishment of Trajectory Optimization Problem

The aircraft trajectory optimization problem is defined as a given initial state in the flight phase to determine an optimal control law so that the state variables meet the constraints and the performance index is minimized.

In the research of this paper, the state and control are highly nonlinear. Because the state and control in the dynamic equation are coupled with each other, the system is prone to high frequency jitter when it is continuously linearized. In order to alleviate this problem, the change rate of attack angle and the change rate of bank angle are defined as new controls, and two new additional state equations are added to the dynamic equation. The new controls defined as follows:

$$\boldsymbol{u} = \left[\dot{\alpha}, \dot{\sigma}\right]^{\mathrm{T}} \tag{11}$$

By adding (11) to the original equation of motion, the augmented equation of motion can be obtained:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{f}_{\omega}(\boldsymbol{x}) \tag{12}$$

where $\boldsymbol{x} = [r, \theta, \varphi, V, \gamma, \psi, \alpha, \sigma]^{\mathrm{T}}$ is the state of the augmented dynamic system; $\boldsymbol{u}$ is a new control; column vector $\boldsymbol{f}(\boldsymbol{x}) \in \mathbb{R}^8$ is only related to the state, $\boldsymbol{B} \in \mathbb{R}^8$ is the control matrix, and $\boldsymbol{f}_{\omega}(\boldsymbol{x}) \in \mathbb{R}^8$ is the column vector related to the earth rotation. The specific form of each vector is as follows:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} V\sin\gamma \\ V\cos\gamma\sin\psi/(r\cos\varphi) \\ V\cos\gamma\cos\psi/r \\ -D/m - g\sin\gamma \\ L\cos\sigma/(mV) + \left(V^2 - g/r\right)\cos\gamma/(Vr) \\ L\sin\sigma/(mV\cos\gamma) + V\cos\gamma\sin\psi\tan\varphi/r \\ 0 \\ 0 \end{bmatrix} \tag{13}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^{\mathrm{T}} \tag{14}$$

$$\boldsymbol{f}_{\omega}(\boldsymbol{x}) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ r\omega_{\mathrm{e}}^2\left(\cos^2\varphi\sin\gamma - \cos\phi\sin\phi\cos\gamma\cos\psi\right) \\ 2\omega_{\mathrm{e}}\cos\varphi\sin\psi + \omega_{\mathrm{e}}^2 r\cos\varphi(\cos\varphi\cos\gamma + \sin\varphi\sin\gamma\cos\psi)/V \\ -2V\omega_{\mathrm{e}}(\tan\gamma\cos\psi\cos\varphi - \sin\varphi) + r\omega_{\mathrm{e}}^2\sin\psi\sin\varphi\cos\varphi/(V\cos\gamma) \\ 0 \\ 0 \end{bmatrix} \tag{15}$$

Since the heat load is the core index of the thermal protection system design, which directly affects the thickness, weight selection and task cost of the thermal protection material. In this study, the minimization of total heat load is selected as the optimization objective. The optimal control problem in this article can be described as

$$\begin{aligned} &\min J = \int_{t_0}^{t_f} \dot{Q} dt \\ &\text{s.t. } \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{B}\boldsymbol{u} + \boldsymbol{f}_{\omega}(\boldsymbol{x}) \\ &\quad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \boldsymbol{x}(t_f) = \boldsymbol{x}_f \\ &\quad \boldsymbol{x} \in [\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}] \\ &\quad \boldsymbol{u} \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}] \\ &\quad V/r + (L\cos\sigma - g)/V = 0 \\ &\quad q = 0.5\rho V^2 \le q_{\max} \\ &\quad \dot{Q} = k_Q \sqrt{\rho} V^{3.15} \le \dot{Q}_{\max} \\ &\quad n = \sqrt{L^2 + D^2}/mg \le n_{\max} \end{aligned} \tag{16}$$

In the formula, t_0 and t_f are expressed as the initial time and the terminal time respectively. $\boldsymbol{x} = [r, \theta, \varphi, V, \gamma, \psi, \alpha, \sigma]^{\mathrm{T}}$, $\boldsymbol{u} = [\dot{\alpha}, \dot{\sigma}]^{\mathrm{T}}$. $\boldsymbol{x} \in [\boldsymbol{x}_{\min}, \boldsymbol{x}_{\max}]$ is a constraint on the state, $\boldsymbol{u} \in [\boldsymbol{u}_{\min}, \boldsymbol{u}_{\max}]$ is a constraint on the control. The specific value of (16) is shown in Section 4.1.

3. Establishment of Online Trajectory Generation Method

In this paper, an online trajectory generation method based on RBF is proposed. The whole process is shown in Figure 2. In the offline part, the pseudo-spectral method is used to generate the optimal trajectory data according to the flight task. The height and velocity dispersion in the initial stage of gliding are used as the initial state to generate the state-action sequence reaching the terminal point. Based on this, the RBF neural network is trained. The input is the height and velocity dispersion data of the current state, and the output is the state-control sequence from the current state to the terminal point. In the online part, the state error is used to determine whether the trajectory needs to be generated, and the actual flight state is used to quickly generate the trajectory of the current state until the expected final state.

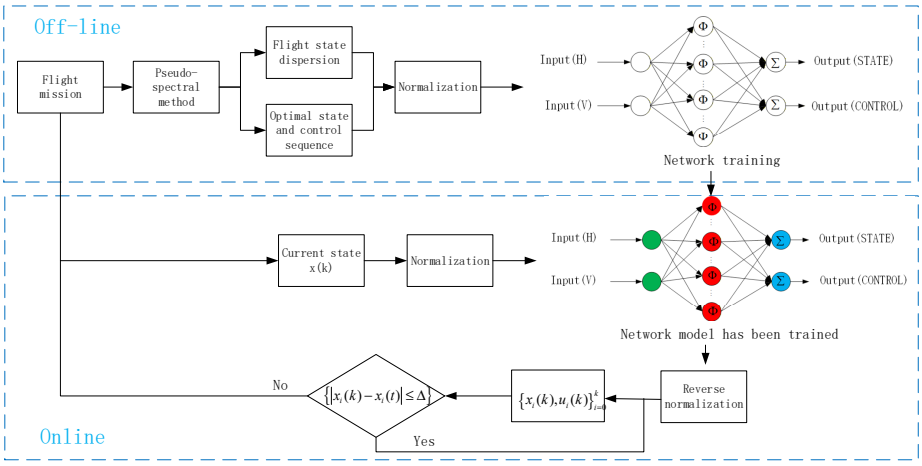


Figure 2. The total diagram of the research plan.

3.1. Generation of Sample Set

Due to the high production volume of low-cost hypersonic glide vehicles, random deviations exist between the actual thrust and the nominal thrust of different booster engines during the boost phase, leading to certain uncertainty in the initial glide state. In this paper, based on the modeling in Section 2.1, the hp adaptive pseudospectral method [32] is

used to solve the trajectory optimization problem established in Section 2.3 with height and speed as the initial dispersion conditions.

The hp comes from the h method (refined mesh element) and the p method (increasing the order of polynomials in the element) in the finite element method. The hp adaptive pseudospectral method is mainly divided into two parts: discretization and mesh refinement. Firstly, the optimal control problem in Section 2.3 is discretized and parameterized.

The time t is divided into K subintervals, $t_0 < t_1 < t_2 \ldots t_K = t_f$ is used to represent the $K+1$ grid, $N_k (k = 1, 2, \ldots, K-1)$ is the number of collocation points of the k-th subinterval, and then the time interval is converted to $[-1, 1]$, so the following time domain transformation is made.

$$\tau = \frac{2t}{t_k - t_{k-1}} - \frac{t_k + t_{k-1}}{t_k - t_{k-1}} \tag{17}$$

The state variable in the k-th subinterval can be expressed as

$$\boldsymbol{x}^{(k)}(\tau) \approx \boldsymbol{X}^{(k)}(\tau) = \sum_{i=0}^{N_k+1} {L_i}^{(k)}(\tau) {\boldsymbol{X}_i}^{(k)} \tag{18}$$

Among them, the Lagrange interpolation polynomial is:

$${L_i}^{(k)}(\tau) = \prod_{j=0, j\neq i}^{N_k+1} \frac{\tau - {\tau_j}^k}{{\tau_i}^k - {\tau_j}^k} \tag{19}$$

In the formula: $\boldsymbol{X}^{(k)}(\tau)$ is the approximate value of the state on the k-th grid, which is a function of τ. N_k is the number of Lagrange-Gauss collocation points on the k-th grid. τ_i is the node of the k-th grid, $\boldsymbol{X}_i$ is the value of the state variable at τ_i, and L_i is the Lagrange polynomial at τ_i.

Then the system state equation can be expressed as

$$\sum_{i=0}^{N_k+1} {\boldsymbol{X}_i}^{(k)} {D_{ji}}^{(k)} = \frac{t_k - t_{k-1}}{2} {f_j}^{(k)}(\boldsymbol{X}_j^{(k)}, \boldsymbol{U}_j^{(k)}) \tag{20}$$

$${D_{ji}}^{(k)} = {\dot{L}_i}^{(k)}({\tau_j}^{(k)}),\ (j = 1, \ldots N_k, i = 1, \ldots N_k + 1) \tag{21}$$

The Formula (21) is the Gauss pseudospectral differential matrix in the subinterval n, which is a matrix of $N_k \times (N_k + 1)$, and ${\boldsymbol{U}_j}^{(k)}$ is the value of the control at τ_i.

In this paper, we only consider the performance index of integral type. The integral term of the performance index function is approximated by Gauss integral, and the objective function can be expressed as follows:

$$J = \sum_{k=1}^{K} \sum_{j=1}^{N_k} \frac{t_k - t_{k-1}}{2} {W_j}^{(k)} {l_j}^{(k)}({\boldsymbol{X}_j}^{(k)} {\boldsymbol{U}_j}^{(k)}) \tag{22}$$

where ${w_j}^{(k)}$ is the Gaussian weight and ${l_j}^{(k)}$ is the original integral term.

After discretization, the optimal control problem is transformed into a nonlinear programming problem. The core of the hp adaptive pseudospectral method is to determine the values of h and p, that is, the number of subintervals and the order of the interpolation polynomial.

The degree of satisfaction of the constraint conditions is evaluated. The maximum allowable deviation is set to ε, and whether the error is less than ε at the collocation point is tested. If it is satisfied, the solution obtained in this interval is considered to be feasible. If the deviation is greater than ε, it is necessary to subdivide the subinterval or increase the order of the interpolation polynomial.

When the grid needs to be re-refined, it is first necessary to determine whether p or h should be increased. Let the curvature function of the j-th state component in the mesh be

$$\kappa_j^{(k)}(\tau) = \frac{\left|\ddot{X}_j^{(k)}(\tau)\right|}{\left|\left[1+\left(\dot{X}_j^{(k)}(\tau)\right)^2\right]^{3/2}\right|} \tag{23}$$

The maximum and average values of the curvature function are ${\kappa^{(k)}}_{\max}$ and $\overline{\kappa}^{(k)}$, respectively. Let

$$r_k = \frac{{\kappa^{(k)}}_{\max}}{\overline{\kappa}^{(k)}} \tag{24}$$

Define judgment indicators $(r_k)_{\max}$. If $r_k < (r_k)_{\max}$, then p is increased, otherwise h is increased.

After discretizing the original optimal control problem, the sequential quadratic programming (SQP) algorithm is used to solve it. The core idea is that at each iteration point, the original problem is approximated as a quadratic programming sub-problem, and the search direction is obtained by solving the sub-problem, and then one-dimensional search and iteration are carried out until convergence. The solution process reference [33].

This paper takes the trajectory optimization problem as the starting point, and clarifies the optimal control problem to be solved and its constraints. On this basis, the height and velocity parameters of the aircraft are scattered to construct a variety of different initial states to cover the actual tasks. Then, for each dispersion case, the pseudospectral method is used for numerical solution. By discretizing the continuous optimal control problem into a nonlinear programming problem, the optimal trajectory satisfying the accuracy requirement is obtained through the GPOPS [34] in MATLAB. After solving, the state sequence and control sequence are extracted respectively to form a single trajectory data $\left[x_j, u_j\right]_{j=1}^{m}$. Finally, the state and control sequences of all distributed cases are summarized to form a trajectory data set $\left\{\left[x_j^{(i)}, u_j^{(i)}\right]_{j=1}^{m}\right\}_{i=1}^{n}$. where m represent the data points contained in a single trajectory, and n represents the total number of trajectories. The solution process is shown in Figure 3.

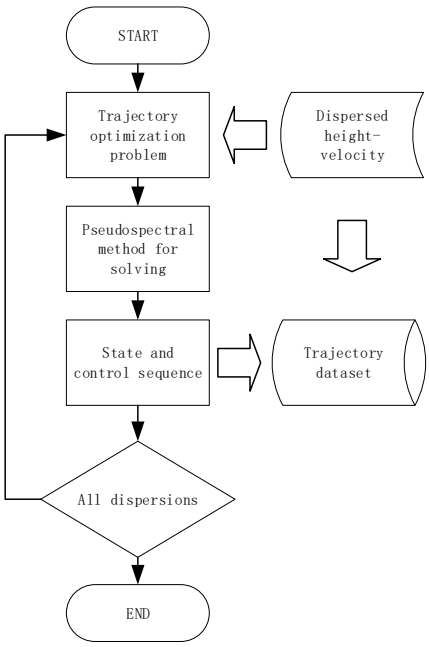


Figure 3. Sample set generation.

3.2. Establishment and Training of RBF Neural Network Model

Radial basis function neural network (RBFNN) is a machine learning model with strong nonlinear mapping ability. The core idea of RBF neural network is to simulate complex nonlinear functions by local response neurons. It adopts a three-layer feedforward structure: first, the input layer sends the original data to the network; subsequently, the hidden layer uses the radial basis function to perform nonlinear transformation on the data. Each hidden layer neuron represents a center. Only when the input is close to the center, the neuron will be activated and respond. The response intensity decays rapidly with increasing distance, reflecting the characteristics of local perception. Finally, the output layer performs a linear weighted sum of the responses of all neurons in the hidden layer to obtain the final result. Figure 4 shows the structure of the trajectory generation neural network.

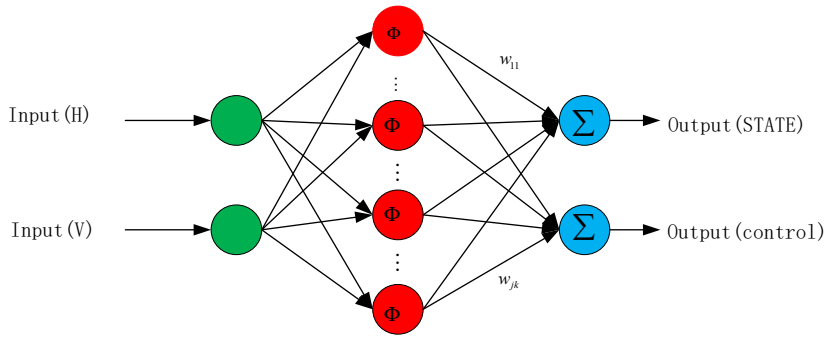


Figure 4. Trajectory Generation Neural Network.

The unit of the hidden layer is activated by the basis function. In this paper, the Gaussian basis function is used. The output of the j-th hidden layer is:

$$\phi_{ij} = e^{\left(-\frac{\|x^{(i)}-\mu_j\|^2}{2\eta_j^2}\right)} \tag{25}$$

where $x^{(i)}$ is the i-th input data, μ_j is the basis function center of the hidden node, and η_j is the width of the radial basis function, also known as the spread factor. The center point μ_j determines the function position, and the width η_j determines the range of action. The radial basis function image is shown in Figure 5.

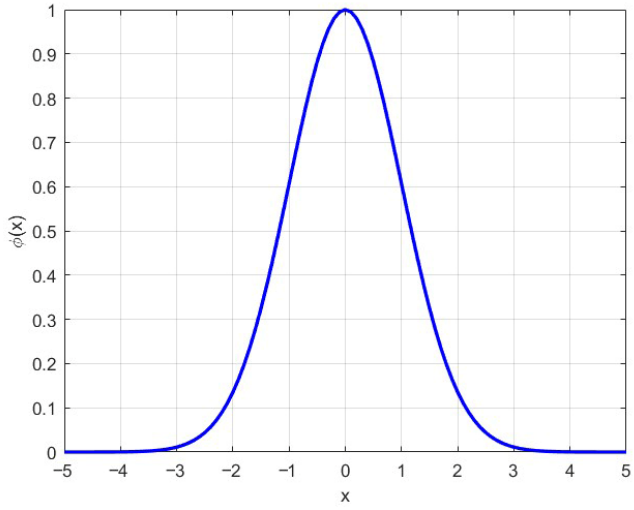


Figure 5. Radial Basis Function image.

The output layer linearly combines the output of the radial basis function hidden layer to generate the expected output. The k-th output layer.

$$y_k^{(i)} = \sum_{i=1}^{M} w_{jk}\phi_{ij} \tag{26}$$

where w is the weight.

Randomly extract 70% of the total number of samples from the database as a training set to train the RBF neural network. The remaining 30% of the samples are used as test samples to test the accuracy of neural network calculation. In the training of this paper, in order to avoid the loss of accuracy of the trajectory data caused by the large difference in the order of magnitude of the input and output data, the input data and the output data are normalized.

The key to the training of RBF neural network lies in the selection of its function center point and the calculation of parameter weight. In this paper, the network is trained by referring to the orthogonal least square method in Chen [35], and the parameters are updated according to the output data matrix.

The hidden layer output matrix $\boldsymbol{\Phi} \in R^{N\times N}$, the target output can be linearly represented by the N column vector of $\boldsymbol{\Phi}$. However, because the contribution ratio of N column vectors of $\boldsymbol{\Phi}$ to $\boldsymbol{y}$ is obviously different, $M \leq N$ vectors can be found in turn according to the contribution size to form $\hat{\boldsymbol{\Phi}} \in R^{M\times N}$, so as to meet the error requirements, that is

$$\left\|\boldsymbol{y} - \boldsymbol{W}\hat{\boldsymbol{\Phi}}\right\| \leq \varepsilon \tag{27}$$

In the formula, $\boldsymbol{W}$ is the optimal weight that satisfies the error ε requirement. If different values of $\boldsymbol{\Phi}$ are selected, the approximation errors are different. Once $\boldsymbol{\Phi}$ is determined, the RBFNN data center μ will be determined. The weight matrix $\boldsymbol{W}$ from the hidden layer to the output layer can be obtained by solving the inverse.

The training process of RBF neural network: Normalize the data by subtracting the mean value from the real value and dividing it by the standard deviation. The basic principle of the orthogonal least squares method is to add neurons one by one, and select the center point μ with the largest error reduction each time; the training termination condition is that the network output error is less than $\varepsilon = 0.00001$ or the number of neurons reaches the upper limit.

In this paper, cross-validation is used to search the grid in the range of $\eta \in (0.1, 1.0)$ and the number of neurons $N \in (50, 300)$. When $\eta < 0.2$, the network exhibits overfitting and the test set error increases; When $\eta > 0.8$, the network tends to be smooth and cannot capture complex dynamic characteristics. Similarly, when the number of neurons is less than 100, the model fitting is insufficient; after more than 250, the training time increases significantly and the accuracy improvement is limited. By analogy, considering the accuracy of the model and the efficiency of online generation, this paper selects $\eta = 0.4$ and the number of neurons MN $=$ 200 as the final configuration. This combination ensures that the network has the smallest generalization error for the test set while maintaining a lightweight structure.

The input layer of the above RBF neural network is set to the dispersion data of height and speed, namely $[H_i, V_i]_{i=1}^{n}$, n denotes the total number of discrete points, i denotes each discrete point. and the output layer of the neural network is set to the state sequence $\left\{\left\{H_i^{(j)}, \lambda_i^{(j)}, \varphi_i^{(j)}, V_i^{(j)}, \gamma_i^{(j)}, \psi_i^{(j)}\right\}_{j=1}^{m}\right\}_{i=1}^{n}$ control sequence $\left\{\left\{\alpha_i^{(j)}, \sigma_i^{(j)}\right\}_{j=1}^{m}\right\}_{i=1}^{n}$ from the current state to the terminal point, m represent the data points contained in a single trajectory, and n represents the total number of trajectories. The training pseudo-

code of the radial basis neural network trajectory generation model is shown in Table 1, where $\boldsymbol{I} = [H_i, V_i]_{i=1}^{n}$, $\boldsymbol{O} = \left\{ \left\{ H_i^{(j)}, \lambda_i^{(j)}, \varphi_i^{(j)}, V_i^{(j)}, \gamma_i^{(j)}, \psi_i^{(j)}, \alpha_i^{(j)}, \sigma_i^{(j)} \right\}_{j=1}^{m} \right\}_{i=1}^{n}$ are the input and output matrices for unnormalized training respectively; $\boldsymbol{I}_{\text{norm}}$, $\boldsymbol{O}_{\text{norm}}$ are the normalized input and output matrices for training; $\boldsymbol{I}_{\text{mean}} = \frac{1}{n}\sum_{i=1}^{n} i_i$, $\boldsymbol{O}_{\text{mean}} = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m} \boldsymbol{O}_i^{(j)}$ are the mean values of the input and output matrix for training; $\boldsymbol{I}_{\text{std}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(i_i - i_{\text{mean}})^2}$, $\boldsymbol{O}_{\text{std}}\sqrt{\frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(\boldsymbol{O}_i^{(j)} - \boldsymbol{O}_{\text{mean}}\right)^2}$ are the standard deviation of the input and output matrix for training; $\boldsymbol{I}_{\text{TEST}}$ is the input matrix for the normalized test; $\boldsymbol{O}_{\text{pre}}$ is the neural network output matrix, which is the normalized value; $\boldsymbol{O}_{\text{real}}$ is the output matrix of the neural network after inverse normalization; $\boldsymbol{O}_{\text{TEST}}$ is the output matrix for testing; η is the width of the radial basis function, and MN is the maximum number of neurons.

Table 1. NN training pseudo-code.

RBFNN Trajectory Generation Model Training Pseudo-Code	
1:	Data loading: Input $[H_i, V_i]_{i=1}^{n}$; output $\left\{ \left\{ H_i^{(j)}, \lambda_i^{(j)}, \varphi_i^{(j)}, V_i^{(j)}, \gamma_i^{(j)}, \psi_i^{(j)}, \alpha_i^{(j)}, \sigma_i^{(j)} \right\}_{j=1}^{m} \right\}_{i=1}^{n}$
2:	Data normalization: $\boldsymbol{I}_{\text{norm}} = \frac{\boldsymbol{I} - \boldsymbol{I}_{\text{mean}}}{\boldsymbol{I}_{\text{std}}}$; $\boldsymbol{O}_{\text{norm}} = \frac{\boldsymbol{O} - \boldsymbol{O}_{\text{mean}}}{\boldsymbol{O}_{\text{std}}}$
3:	Set: $\varepsilon = 0.00001$; $\eta = 0.4$; MN $= 200$
4:	For $k = 1$; Iterative addition of neurons
5:	Calculate the radial basis function values: $\phi_{ij} = \exp\left(-\frac{\left\| x^{(i)} - \mu_j \right\|^2}{2\eta_j^2} \right)$
6:	Determine hidden layer output matrix $\boldsymbol{\Phi}$
7:	Calculate output weight: $\boldsymbol{W} = \boldsymbol{\Phi}^* \boldsymbol{y}$
8:	Update network error: LOSS $= \|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{\Phi}\|$
9:	If $\|\boldsymbol{y} - \boldsymbol{W}\boldsymbol{\Phi}\| \leq \varepsilon$; Training end
10:	Else $k = k + 1$; Return to 5
11:	Model evaluation: Using the test set for generation: $\boldsymbol{O}_{\text{pre}} = \text{RBF}(\mathbf{I}_{\text{TEST}})$
12:	Anti- normalization: $\boldsymbol{O}_{\text{real}} = \boldsymbol{O}_{\text{pre}} * \boldsymbol{O}_{\text{std}} + \boldsymbol{O}_{\text{mean}}$
13:	Calculation performance indicators: RMSE $= \text{sqrt}(\text{mean}(\boldsymbol{O}_{\text{real}} - \boldsymbol{O}_{\text{TEST}})^2)$

The establishment of the trajectory generation method in this paper ensures the ability of the aircraft to select the trajectory during flight, which can reduce the amount of calculation and quickly generate the trajectory online compared with the traditional optimization method.

4. Simulation Analysis

4.1. Generation of Trajectory Sample Set

In this paper, CAV (Common Aero Vehicle) is used for simulation verification. The aircraft is proposed by the U.S. Space Commands. This parameter is widely used in the research and verification of trajectory optimization and guidance methods. The mass is 907 kg and the aerodynamic area is 0.4839 m^2 [36]. The sample generation uses the pseudospectral method to optimize the trajectory, and the GPOPS-‖ [34] in MATLAB is used to solve the problem. Since the trajectory optimization problem is solved by using the angle of attack and the bank angle as the control variables, the trajectory will not be smooth enough. Therefore, the angle of attack and the heeling angle are augmented as the state variables, and the change rate of the angle of attack and the bank angle is solved as the control variables.

The initial states and initial control are shown in Table 2, the constraint conditions are shown in Table 3.

Table 2. Initial State.

Parameter	Data
Initial height h_0	65~70 km
initial longitude θ_0	0°
Initial latitude φ_0	0°
Initial velocity V_0	5500~6000 m/s
Initial flight path angle γ_0	0°
Initial heading angle ψ_0	90°
Initial angle of attack α_0	30°
Initial bank angle σ_0	0°

Table 3. Constraint Conditions.

Parameter	Data
Angle of attack α	30~40°
Bank angle σ	−70~70°
Angle of attack rate $\dot{\alpha}$	−1~1°
Bank angle rate $\dot{\sigma}$	−1~1°
Dynamic pressure q	0~20 kPa
Heat flow $\dot{Q}$	0~1000 kW/m^2
Overload n	0~2.5

The constraints are shown in Table 3.

The initial state is dispersed by the height and velocity. The state point is taken every 100 m at the height, and the state point is taken every 10 m/s at the speed. According to the latitude and longitude of the reachable domain, the data for determining the terminal status point is shown in Table 4. The terminal speed range (800–1000 m/s) is not a hard constraint, but a certain speed range is required to enter the energy management stage at a height of 30 km.

Table 4. Terminal State.

Parameter	Data
Terminal height h_f	30 km
Terminal longitude θ_f	28° E
Terminal latitude φ_f	12° N
Terminal velocity V_f	800~1000 m/s

A total of 2500 initial dispersion samples are generated according to the pseudo-spectral method, and all samples are interpolated according to time to obtain the state-control data every 1 s interval, which includes height, longitude, latitude, speed, flight path angle, heading angle, angle of attack, bank angle. Each of these states or control contain about 2.5 million data points.

Figures 6 and 7 shows the envelope boundary of the whole trajectory cluster under the discrete conditions of initial height (65 km~70 km) and initial velocity (5500 m/s~6000 m/s), which are height-velocity corridor, latitude and longitude corridor, flight path angle corridor, heading angle corridor, angle of attack corridor, bank angle corridor. In addition, we select the trajectories generated under the maximum and minimum initial height and velocity states as representative trajectories for display, and each

trajectory is within the envelope range. The dispersion of initial height and velocity leads to the diversity of reentry trajectories, which reflects the influence of uncertainty. Therefore, a database based on a large number of reentry trajectories is established for neural network training.

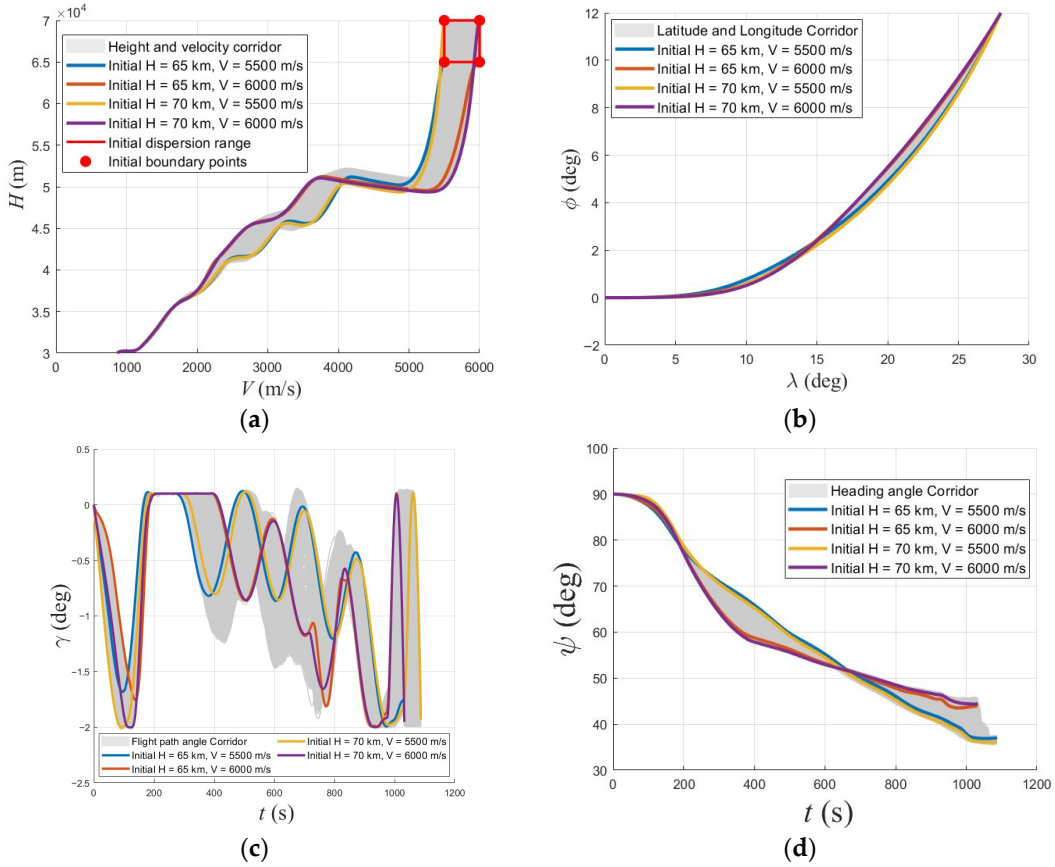


Figure 6. State sample figure (**a**) Height-velocity corridor; (**b**) Latitude and Longitude corridor; (**c**) Flight path angle corridor; (**d**) Heading angle corridor.

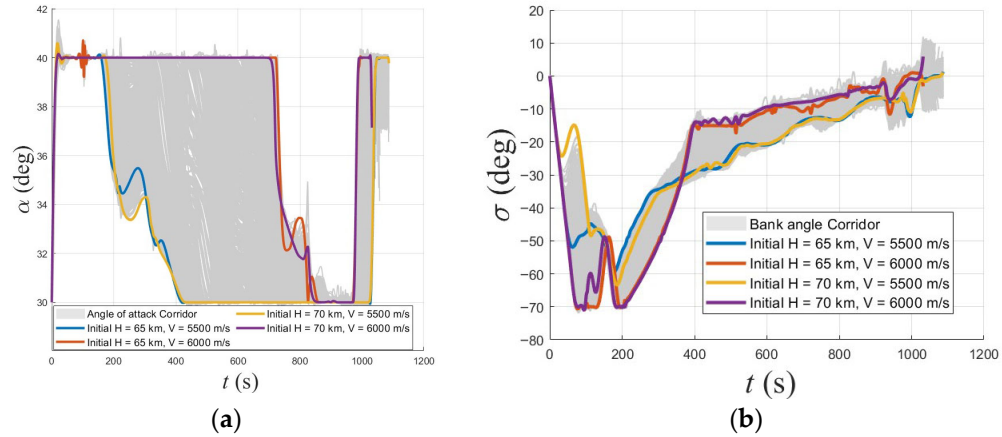


Figure 7. Control sample figure (**a**) Angle of attack corridor; (**b**) Bank angle corridor.

4.2. Neural Network Training Simulation Results

Due to the difference between the order of magnitude of each state and the control, in order to ensure the accuracy of the application of the neural network after training, this paper selects the neural network for training the state and the control respectively. It can be seen from Section 3.2 that the maximum number of neurons is selected as MN $=$ 200, and the cut-off accuracy (mean square error, MSE) is selected as $\varepsilon = 0.00001$. The training

termination condition is that the network output error is less than the cut-off accuracy or the number of neurons reaches the upper limit. The training process is shown in Figure 8. The red dash line in the figure is the MSE value required for training.

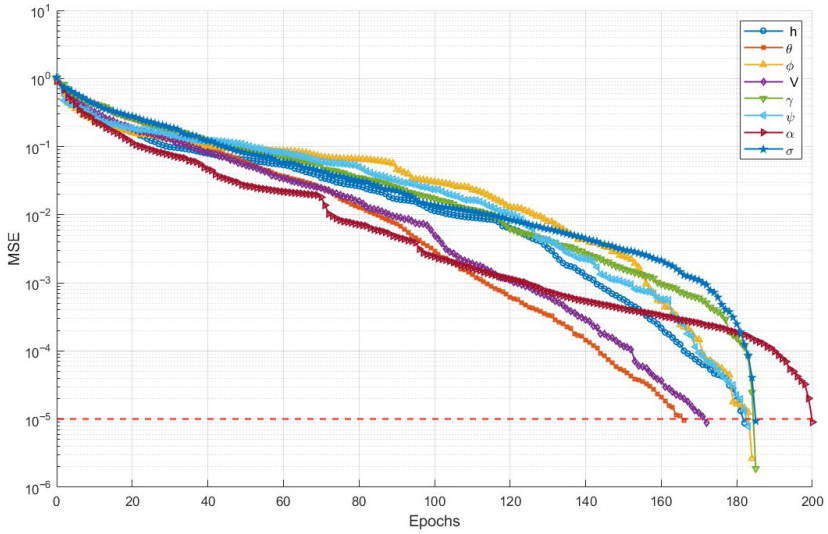


Figure 8. Neural network training process diagram.

The total offline training time for the eight RBF networks (one for each state/control sequence) on the specified hardware is 550 s. This includes data normalization, OLS center selection, and final weight calculation. The training is not part of the online runtime. Over all 2500 trajectories, the constraint satisfaction rate is 100%.

Table 5 shows the epoch of iterations of neural network for outputting each parameter and the terminal MSE that meets the accuracy requirements. It can be seen from the table that the number of iterations of each network is between 150 and 200, and the final iteration accuracy is below 0.00001. The trained network meets the accuracy requirements. After the training is completed, a total of 8 neural networks are formed, which output state sequences and control sequences respectively.

Table 5. The epoch of iterations of each neural network and the terminal MSE.

Neural Network Output Parameters	Epochs	Terminal MSE
height h	182	8.784×10^{-6}
longitude θ	166	9.466×10^{-6}
latitude φ	184	2.645×10^{-6}
velocity V	172	8.745×10^{-6}
flight path angle γ	185	1.878×10^{-6}
heading angle ψ	183	7.799×10^{-6}
angle of attack α	200	9.043×10^{-6}
bank angle σ	185	9.330×10^{-6}

4.3. Online Trajectory Generation Simulation Results

In order to verify the superiority of the method in this paper compared with the deep neural networks (DNN), the existing database is used to train the deep neural networks. The trajectory generation method based on the deep neural networks generates the control according to the state, and then the whole trajectory is obtained by integration.

Referring to [13], the DDN architecture and hyperparameter design in this paper are shown in Tables 6 and 7.

Table 6. DNN Network Structure.

Component	Specification
Input layer	18 neurons (initial state, terminal state sf, current state $\left[s_0, s_f, s_{current}\right]$)
Number of hidden layers	8 fully-connected layers
Neurons per hidden layer	500
Activation function	Tanh (used as a sigmoid-type activation)
Output layer	2 neurons (angle of attack α, bank angle σ)

Table 7. Training Hyperparameters.

Hyperparameter	Value
Optimizer	Adam
learning rate	0.0001
Batch size	256
epochs	30
Loss function	Mean squared error (MSE)
Early stopping condition	Training loss (MSE) < 0.001

The training loss function convergence curve is shown in Figure 9.

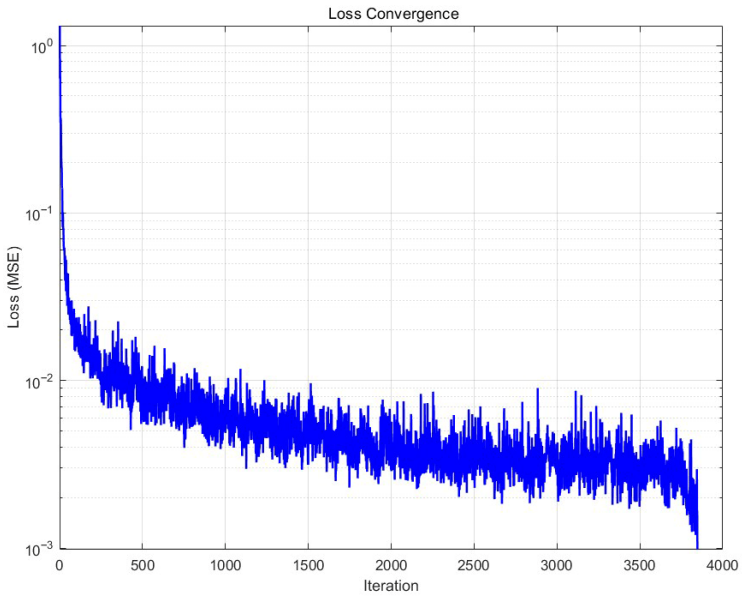


Figure 9. DNN loss function convergence curve.

According to the size of the sample, each epoch consists of 377 iterations. It can be seen from Figure 9 that the MSE reaches the condition to complete the training in the 11th epoch of iteration.

Randomly select the initial state for trajectory generation, the initial height of 65 km and the initial speed of 5500 m/s are selected for simulation analysis. In order to verify the accuracy of the data generated by the neural network, the control generated by RBF neural network is interpolated, and the state trajectory is obtained by fourth-order Runge-Kutta integration.

The sample trajectory, the trajectory generated by RBF neural network, the trajectory obtained by integrating the control obtained by RBF, and the trajectory obtained by recursively generating the control with DNN and integrating step by step are compared as shown in Figure 10, where (a)–(e) is the state, (f) and (g) are the control, and (h)–(j) are the dynamic pressure, heat flow and overload obtained by the DNN step-by-step integration and the integral of the control generated by the RBF, respectively.

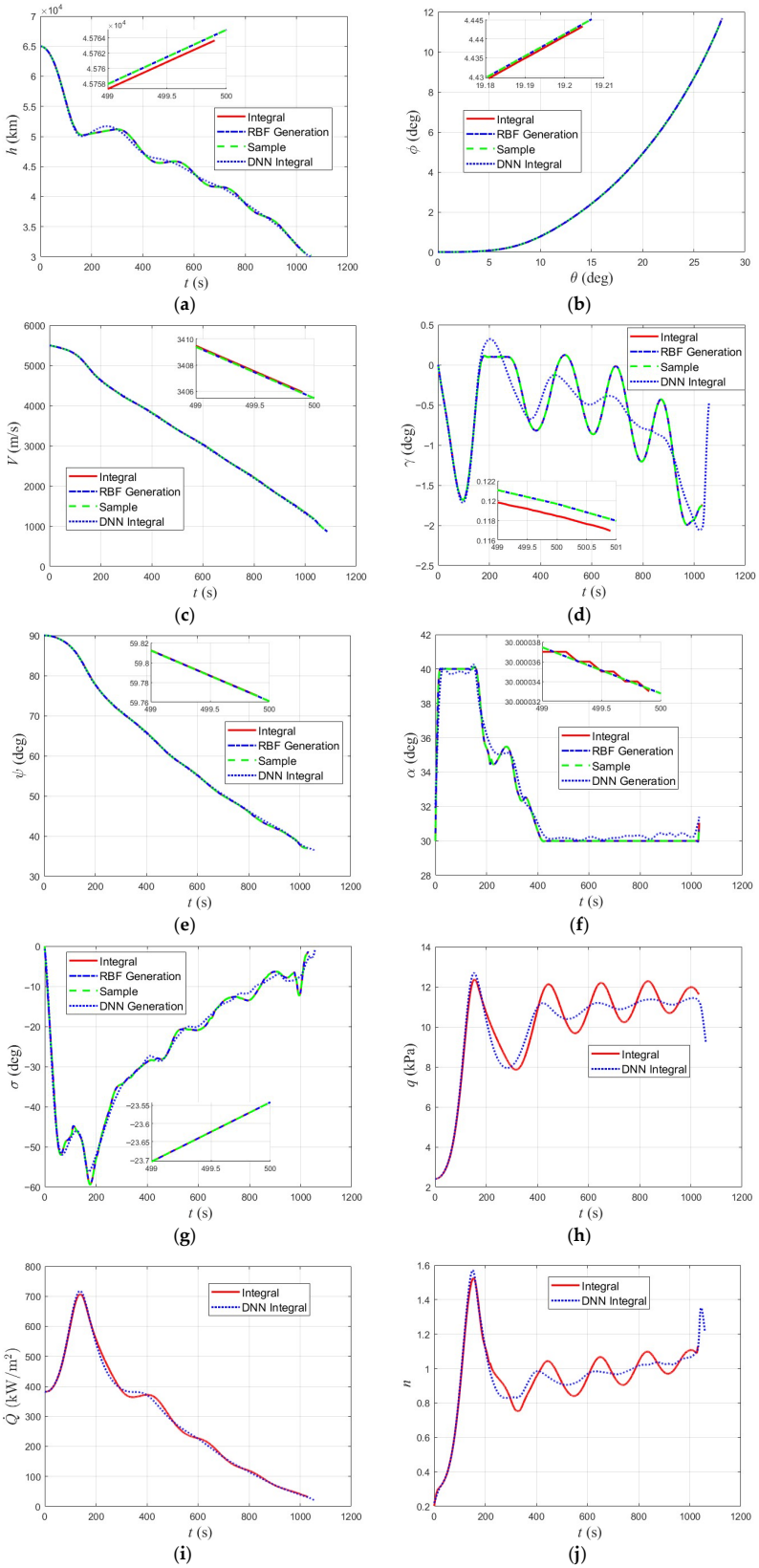


Figure 10. Integral results, RBF neural network results, sample data, Deep neural network results, comparison figure. (**a**) Curve of height changing with time; (**b**) The latitude and longitude change curve; (**c**) Curve of Speed changing with time; (**d**) Curve of Flight path angle changing with time; (**e**) Curve of heading angle changing with time; (**f**) Curve of angle of attack changing with time; (**g**) Curve of bank changing with time; (**h**) Curve of dynamic pressure changing with time; (**i**) Curve of heat flow with time; (**j**) Curve of overload changing with time.

All timings were obtained on a desktop computer with an Intel Core i7-11800H CPU (2.3 GHz) (Intel Corporation, Santa Clara, CA, USA), 16 GB RAM, running MATLAB R2024b under Windows 11. From the comparison between the trajectory generated by the RBF neural network in Figure 10 and the trajectory generated by the pseudospectral method, it can be seen that the deviation of the state and the control is below 1×10^{-5}. Therefore, the optimality of the neural network is better guaranteed. In terms of computation time, the time for the trained neural network to generate the trajectory is 1/100 of the trajectory generated by the pseudospectral method, which ensures the rapidity of its online trajectory generation. The control (angle of attack, bank angle) generated by the network is linearly interpolated, and the trajectory after integration is obtained by integrating the dynamic equation. The deviation from the trajectory is also very small, which ensures the feasibility of the subsequent guidance command. It can be seen from the dynamic pressure, heat flow, overload curve of (h)–(j) in Figure 10 that all process constraints strictly meet the constraints in Table 4.

According to the results of Figure 10, based on the GPOPS results, the root mean square error (RMSE) of the trajectory generated by DNN and RBFNN is calculated, which includes the height, speed, latitude and longitude, flight path angle and control angle of attack and bank angle. As shown in Table 8, the RMSE of angle of attack and bank angle in the trajectory generated by RBFNN are all below 10^{-5}, which is four orders of magnitude lower than that generated by DNN. The RMSE of height and velocity is also below 0.1, which is four and three orders of magnitude higher than the trajectory accuracy obtained by DNN integration. The CPU calculation time of RBF network is 0.115 s, which is only 1/10 of DNN and 1/100 of GPOPS. As shown in the Table 9. Since the DNN-based method uses the output of the current control and uses the inherent recursive scheme for state calculation, causes cumulative regression error in a long flight. The RBF network directly maps the initial state to the entire state-control sequence, solving a structured output problem with global approximation characteristics, so the error is small.

Table 8. The RMSE of the trajectory generated by RBFNN and DNN.

Parameter	RBFNN RMSE	DNN RMSE
height h	0.05 m	1648 m
longitude θ	0.003°	16.5747°
latitude φ	0.00615°	6.2899°
velocity V	0.02 m/s	25 m/s
flight path angle γ	0.00259°	0.7788°
angle of attack α	0.00000178°	0.3949°
bank angle σ	0.0000004°	1.2085°

Table 9. The flight time and CPU time of the trajectory generated by RBFNN, DNN and GPOPS.

Method	Flight Time (s)	CPU Time (s)
GPOPS	1031	29
DNN	1031	1.6
RBFNN	1031	0.115

5. Conclusions

In this paper, an online trajectory generation method based on the RBF neural network is proposed for the reentry phase of the hypersonic glide vehicle, aiming to address the real-time requirements of online trajectory generation. This method tackles the challenges posed by the low computational efficiency of traditional trajectory optimization methods, as well as the recursive computational burden and training complexity of existing intel-

ligent methods. The key innovation lies in replacing recursive step-by-step computation with a direct mapping from the current state to the complete future state-control sequence. Simulation results demonstrate that the proposed method significantly improves online generation efficiency while keeping trajectory errors within a small range. Moreover, its computation time is only 1/100 of that required by the pseudospectral method. A single sequence prediction makes the RBF-based method particularly suitable for low-cost hypersonic glide vehicles with limited airborne computing power. In summary, the proposed RBFNN-based method demonstrates high precision and efficiency for onboard trajectory generation under initial state dispersion. Nevertheless, several avenues remain for future investigation. The current method only considers height and velocity perturbations; future work will incorporate dispersions in longitude and latitude. The framework can be extended to handle no-fly zone constraints.

Author Contributions: Methodology, Z.C. and F.Y.; software, Z.C. and C.Z.; formal analysis, Z.C. and F.Y.; investigation, Z.C. and C.Z.; writing—original draft preparation, Z.C.; writing—review and editing, F.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Research Funds for the Central Universities, grant number DUT25GF203.

Data Availability Statement: All the data used during the study period of this paper appear in the submitted articles.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wei, C.Z.; Zhang, Y.K.; Pu, J.L.; Zhang, F. Online entry trajectory planning with simultaneous constraints on terminal full-states, flight time and no-fly zones. *Aerosp. Sci. Technol.* **2026**, *168*, 111019. [CrossRef]
2. Johnson, E.N.; Calise, A.J.; Curry, M.D.; Mease, K.D.; Corban, J.E. Adaptive guidance and control for autonomous hypersonic vehicles. *J. Guid. Control Dyn.* **2006**, *29*, 725–737. [CrossRef]
3. Beijing Lingkongtianxing Technology Co., Ltd. Available online: https://www.spacetransportation.com.cn/ (accessed on 13 March 2026).
4. Huang, Y.W.; Wang, P.; Wang, J.W.; Xie, J. Research on the low-cost and scaled development of U.S. hypersonic missiles. *Tact. Missile Technol.* **2026**, 1–16. (In Chinese)
5. Sun, H.Q.; Zhang, S.G. Skip Re-Entry Trajectory Detection and Guidance for Maneuvering Vehicles. *Sensors* **2020**, *20*, 2976. [CrossRef] [PubMed]
6. Zhao, J.; Zhou, R.; Jin, X. Progress in reentry trajectory planning for hypersonic vehicle. *J. Syst. Eng. Electron.* **2014**, *25*, 627–639. [CrossRef]
7. Sun, Q.C.; Xu, J.J.; Zhang, H.S. Guidance for Hypersonic Reentry Using Nonlinear Model Predictive Control and Radau Pseudospectral Method. *Optim. Control Appl. Methods* **2025**, *46*, 1417–1428. [CrossRef]
8. Mao, Y.F.; Zhang, D.L.; Wang, L. Reentry trajectory optimization for hypersonic vehicle based on improved Gauss pseudospectral method. *Soft Comput.* **2017**, *21*, 4583–4592. [CrossRef]
9. Zhao, J.; Zhou, R.; Jin, X.L. Reentry Trajectory Optimization Based on a Multistage Pseudospectral Method. *Sci. World J.* **2014**, *2014*, 878193. [CrossRef]
10. Huang, G.; Lu, Y.; Nan, Y. A survey of numerical algorithms for trajectory optimization of flight vehicles. *Sci. China Tech. Sci.* **2012**, *55*, 2538–2560. [CrossRef]
11. Chai, R.; Tsourdos, A.; Savvaris, A.; Xia, Y.Q.; Chai, S.C. Real time Reentry Trajectory Planning of Hypersonic Vehicles: A Two-Step Strategy Incorporating Fuzzy Multi-Objective Transcription and Deep Neural Network. *IEEE Trans. Ind. Electron.* **2020**, *67*, 6904–6915. [CrossRef]
12. Shi, Y.; Wang, Z.B. Onboard Generation of Optimal Trajectories for Hypersonic Vehicles Using Deep Learning. *J. Spacecr. Rockets* **2021**, *58*, 400–414. [CrossRef]
13. Wang, J.Y.; Wu, Y.P.; Liu, M.; Yang, M.; Liang, H.Z. A Real time Trajectory Optimization Method for Hypersonic Vehicles Based on a Deep Neural Network. *Aerospace* **2022**, *9*, 188. [CrossRef]
14. Dai, P.; Feng, D.Z.; Feng, W.H.; Cui, J.S.; Zhang, L.H. Entry trajectory optimization for hypersonic vehicles based on convex programming and neural network. *Aerosp. Sci. Technol.* **2023**, *137*, 108259. [CrossRef]

15. Wang, X.; Dai, P.; Cheng, X.M.; Liu, Y.Z.; Cui, J.S.; Zhang, L.H.; Feng, D.Z. An online generation method of ascent trajectory based on feedforward neural networks. *Aerosp. Sci. Technol.* **2022**, *128*, 107739. [CrossRef]
16. Li, H.; Chen, H.; Tan, C.; Jiang, Z.; Xu, X. Fast Trajectory Generation with a Deep Neural Network for Hypersonic Entry Flight. *Aerospace* **2023**, *10*, 931. [CrossRef]
17. Cheng, L.; Jiang, F.H.; Wang, Z.B.; Li, J.F. Multi-constrained Real-Time Entry Guidance Using Deep Neural Networks. *IEEE Trans. Aerosp. Electron. Syst.* **2021**, *57*, 325–340. [CrossRef]
18. Su, Y.; Liu, Y. Onboard generation of reentry trajectory for RLV via regularized extreme learning machine and marine predator whale optimizer. *Adv. Space Res.* **2024**, *74*, 5023–5043. [CrossRef]
19. Wang, T.J.; Wang, R.; Meng, F.Y.; Wang, J.N.; Chen, G. Fast optimal robust trajectory generation based on polynomial chaos and deep neural networks. *Nonlinear. Dyn.* **2025**, *113*, 34485–34509. [CrossRef]
20. Shi, J.L.; Wang, J.B.; Su, L.F.; Ma, Z.W.; Chen, H.B. A Neural Network Warm-Started Indirect Trajectory Optimization Method. *Aerospace* **2022**, *9*, 435. [CrossRef]
21. Peng, G.X.; Wang, B.; Liu, L.; Fan, H.J.; Cheng, Z.T. Real-time adaptive entry trajectory generation with modular policy and deep reinforcement learning. *Aerosp. Sci. Technol.* **2023**, *142*, 108594. [CrossRef]
22. Bao, C.Y.; Wang, P.; He, R.Z.; Tang, G.Z. Autonomous trajectory planning method for hypersonic vehicles in glide phase based on DDPG algorithm. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2023**, *237*, 1855–1867. [CrossRef]
23. Bao, C.Y.; Zhou, X.; Wang, P.; He, R.Z.; Tang, G.Z. A deep reinforcement learning-based approach to onboard trajectory generation for hypersonic vehicles. *Aeronaut. J.* **2023**, *127*, 1638–1658. [CrossRef]
24. Li, J.F.; Song, S.M.; Shi, X.P. Deep reinforcement learning based trajectory real-time planning for hypersonic gliding vehicles. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2024**, *238*, 1665–1682. [CrossRef]
25. Su, L.F.; Wang, J.B.; Chen, H.B.; Pezzella, G. A Real time and Optimal Hypersonic Entry Guidance Method Using Inverse Reinforcement Learning. *Aerospace* **2023**, *10*, 948. [CrossRef]
26. Zhu, J.W.; Zhang, H.; Zhao, S.B.; Bao, W.M. Multi-Constrained intelligent gliding guidance via optimal control and DQN. *Sci. China-Inf. Sci.* **2023**, *66*, 132202. [CrossRef]
27. Wu, T.C.; Wang, H.L.; Liu, Y.H.; Li, T.R.; Yu, Y. Learning-based interfered fluid avoidance guidance for hypersonic reentry vehicles with multiple constraints. *ISA Trans.* **2023**, *139*, 291–307. [CrossRef] [PubMed]
28. Vinh, N.X.; Busemann, A.; Culp, R.D. *Hypersonic and Planetary Entry Flight Mechanics*, 1st ed.; University of Michigan Press: Ann Arbor, MI, USA, 1980; pp. 26–27.
29. Sun, Y.; Duan, G.R.; Zhang, M.R.; Zhang, Z. Modified aerodynamic coefficient models of hypersonic vehicle in reentry phase. *J. Syst. Eng. Electron.* **2011**, *33*, 134–137. (In Chinese)
30. Lu, P. Entry Guidance: A Unified Method. *J. Guid. Control Dyn.* **2014**, *37*, 713–728. [CrossRef]
31. Shen, Z.J.; Lu, P. Onboard generation of three-dimensional constrained entry trajectories. *J. Guid. Control Dyn.* **2003**, *26*, 111–121. [CrossRef]
32. Han, P.; Shan, J.Y.; Meng, X.Y. Re-entry trajectory optimization using an hp-adaptive Radau pseudospectral method. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2013**, *227*, 1623–1636. [CrossRef]
33. Yokoyama, N.; Suzuki, S.; Tsuchiya, T. Convergence acceleration of direct trajectory optimization using novel Hessian calculation methods. *J. Optim. Theory Appl.* **2008**, *136*, 297–319. [CrossRef]
34. Rao, A.V.; Benson, D.A.; Darby, C.; Patterson, M.A.; Francolin, C.; Sanders, I.; Huntington, G.T. Algorithm 902: GPOPS, A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method. *ACM Trans. Math. Softw.* **2010**, *37*, 22. [CrossRef]
35. Chen, E.S.; Yang, H.H.; Bos, S. Orthogonal least-squares learning algorithm with local adaptation process for the radial basis function networks. *IEEE Signal Process. Lett.* **1996**, *3*, 253–255. [CrossRef]
36. Phillips, T.H. A Common Aero Vehicle (CAV) Model, Description, and Employment Guide. *Schafer Corp. AFRL AFSPC* **2003**, *27*, 1–12.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.