32nd Congress of the International Council of the Aeronautical Sciences
September 6-10, 2021
Pudong Shangri-La, Shanghai, China

LEARNING-BASED MIDCOURSE GUIDANCE FOR VELOCITY MAXIMIZATION WITH ANGULAR CONSTRAINT

Tianyu Jin[1], Hongyan Li[2], Shaoming He[3] & Yufei Li[4]

[1,2,3,4]Beijing Institute of Technology, Beijing 100081, People's Republic of China

Abstract

This paper investigates the midcourse guidance problem for velocity maximization with constrained arrival angle and proposes a learning-based guidance algorithm to solve this problem. A full-envelope training set is first constructed by optimal state-action pairs generated by nonlinear programming (NLP) software. Then, the deep neural network (DNN) is trained based on the training set by supervised learning approach. With the well-trained DNN, optimal guidance command can be directly generated in accordance with the current states. To improve the transportability of the trained DNN, transfer learning is also used to improve the generalizability and adaptivity of the proposed algorithm. Compared with the computationally-expensive NLP algorithms, the proposed approach requires less computational power hence is more convenient for online implementation. Extensive numerical simulations are conducted to support the proposed algorithm.

Keywords: Midcourse guidance; Velocity maximization; Learning approach; Deep Neural Network

1. Introduction

The main objective of the midcourse phase is to guide the missile to approach the predicted handover point (PHP) in the required time with favorable terminal constraints. Among all the constraints, the final velocity and arrival angle are two key elements for the terminal guidance phase: high velocity enables high survivability and kill-probability in the following terminal guidance phase, and proper arrival angle can help attenuate the handover effect and guarantee target capture from the onboard seeker [1, 2]. Therefore, maximizing the final velocity with angular constraint is a worthy goal for the guidance law design of the midcourse phase.

Since the issue associated with velocity maximization is that one needs to consider state-dependent gravitational and aerodynamic forces in problem formulation [3], the mathematical model of the midcourse guidance is nonconvex and highly complicated, and finding the analytical solution is intractable. Hence, NLP methods are usually leveraged to solve this problem. However, NLP methods generally require iterative calculations to generate the optimal command at each time instant, which is computationally inefficient for an embedded computer [4–6]. To improve the computational efficiency, some researchers pursued suboptimal analytical guidance laws that maximize the terminal velocity by simplifying the mathematical model [7, 8]. However, too much assumption might make the simplified model unrealistic and unable for practical implementation. By ignoring the gravitational force and using constant aerodynamic coefficients, the authors in [3] revealed that PNG with guidance gain equal to three maximizes the velocity in the terminal guidance phase. However, it is known that the duration of the midcourse phase is much longer than that of the terminal phase. This implies that the gravity and dynamic aerodynamic coefficients pose larger influences on the flight speed, and hence the traditional PNG is no longer optimal in terms of velocity maximization.

Thanks to the rapid development of artificial intelligence, there has been an increasing attention on the learning-based algorithms in recent years. Different from the traditional model-centric approach, the learning-based method is a data-centric approach and leverages machine-learning algorithms to train a given agent to act properly based on adequate amount of training data. Since the training process can be performed offline, a well-trained agent is able to generate near-optimal actions according

to its external inputs without consuming too much computational power, which is very practical for embedded computer. The authors in [11] use deep learning approach to design guidance algorithm for hypersonic vehicle in re-entry process. This method provides fairly high impact accuracy and requires low computational power hence can be implemented online. The deep deterministic policy gradient (DDPG) algorithm in deep reinforcement learning (DRL) is used in [12] to develop missile guidance algorithm with minimum control effort. The authors in [13] uses DRL and feedback guidance methods to achieve fuel-optimal landing on Mars. In [14], deep learning methods with policy network and value function network are used to solve the orbit transfer problem of the spacecraft and proves that they enable to solve this problem with high accuracy.
Motivated by the above observations, this paper proposes a learning-based approach to generate optimal guidance command rapidly to realize velocity maximization with constrained arrival angle. The proposed guidance algorithm consists of two main functional blocks: offline training and online implementation. Numerous trajectories to various possible PHPs for velocity maximization with angular constraint are first optimized offline by NLP software to generate a complete training set enabling to cover the flight envelope of the missile. Then, a deep neural network can be trained offline based on the optimal state-action pairs of the training set by supervised learning approach. Note that the state is the variable set characterizing the relationship between missile and PHP, and the action indicates the guidance command. For online implementation, the well-trained DNN will be loaded into the onboard computer and the optimal guidance command can be directly mapped from the current states. Compared with the computationally-expensive NLP algorithms, the proposed approach requires less computational power hence is more convenient for online implementation.
Note that, in practice, especially in design phase, the aerodynamic coefficients of the missile may alter due to the modification of configuration and the original trained DNN is unsuitable for the new situation. However, retraining a new DNN from scratch consumes too much time, which is inefficient for guidance system design. To alleviate this issue, transfer learning is introduced to improve the generalization ability of the proposed algorithm. By freezing part of layers of the original DNN and retrain the remaining layers, the retraining process will significantly speed up, implying that a well-trained DNN can be efficiently generalized to different aerodynamic models with consuming relatively less time and computational power.
The reminder of the paper is organized as follows. Sec. 2. provides the mathematical model and formulates the optimization problem. Sec. 3. introduces supervised learning approach with DNN and leverages this method to recover the optimal behavior, followed by transfer learning to improve the performance of DNN in Sec. 4. Analysis and simulation results with these two different approaches are also presented in Sec. 3. and Sec. 4. respectively. Finally, some conclusions are offered in Sec. 5.

2. Problem Formulation

This paper assumes the missile is equipped with well-developed autopilot that provides roll, pitch and yaw stabilization so that the guidance command can be readily decoupled into vertical and horizontal planes. Besides, the missile is assumed to be ideal point-mass model since the autopilot delay is generally short and negligible in comparison with the time constant of the guidance loop.

2.1 Mathematical Model

Fig.1 depicts the mathematical model studied in this paper, where XOY is the inertial coordinate and X_bMY_b is the body frame of the missile. The notations of M and N denote the missile and PHP, respectively. The symbol of V represents the missile velocity. The variables of φ, and λ, α signify the flight-path angle, line-of-sight (LOS) angle and angle of attack (AoA), respectively. And the lift drag and thrust acting on the missile are denoted by L, D and T, respectively. The variable of m stands for the mass and g is the gravitational acceleration.
The governing equations describing the midcourse guidance problem in the vertical plane can be formulated as

$$\dot{x} = V\cos\varphi \tag{1}$$

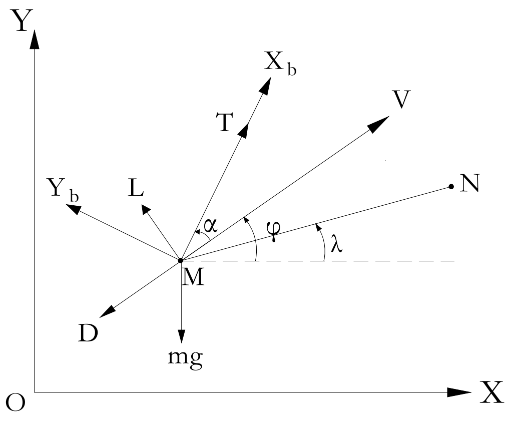


Figure 1 – Definition of notations and symbols.

$$\dot{y} = V \sin\varphi \tag{2}$$

$$\dot{\varphi} = \frac{L + T\sin\alpha}{mV} - \frac{g\cos\varphi}{V} \tag{3}$$

$$\dot{V} = \frac{T\cos\alpha - D}{m} - g\sin\varphi \tag{4}$$

$$\dot{m} = -\frac{T}{I_{sp}} \tag{5}$$

where I_{sp} is the specific impulse and the gravitational acceleration $g = 9.81 m/s^2$.
Lift and drag forces can be expressed as

$$L = C_L qS = (C_{L\alpha}\alpha)qS \tag{6}$$

$$D = C_D qS = (C_{D0} + C_{D\alpha}\alpha^2)qS \tag{7}$$

where dynamic pressure $q = \frac{1}{2}\rho V^2$, C_L is the lift coefficient, C_D is the drag coefficient and S is the reference area. $C_{L\alpha}$ represents the derivative of lift coefficient with respect to AoA, C_{D0} denotes the parasite drag coefficient and $C_{D\alpha}$ represents the induced drag coefficient. These aerodynamic coefficients are functions of Mach number Ma and can be obtained from wind tunnel tests. The required atmospheric density and local sonic speed can obtained with reference to the standard atmosphere [15].

2.2 Optimization Problem

The objective of this paper is to maximize the final velocity V_f with the constraints of arrival angle $\varphi = \varphi_f$ at a specific time t_f. And, at the terminal time, the missile should arrive at the PHP with zero miss distance. Thus, according to Eqs.(1)-(5), the optimization problem can be expressed as

$$\begin{aligned} &\min_{\alpha} J = -V^2(t_f) \\ s.t.\quad &\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \alpha) \\ &\mathbf{x}(t_0) = [x(t_0), y(t_0), \varphi(t_0), V(t_0), m(t_0)]^T \\ &\mathbf{x}(t_f) = [x(t_f), y(t_f), \varphi(t_f)]^T \end{aligned} \tag{8}$$

Where $\mathbf{u} = \alpha$ is the control input and $\mathbf{x} = [x, y, \varphi, V, m]^T$ is the system state. Since AoA is a small angle, the relationship between α and latax command a_c can be expressed as

$$\alpha \approx \frac{ma_c + mg\cos\varphi}{qC_{L\alpha}S + T} \tag{9}$$

Note that the problem (8) is highly nonlinear and cannot be analytically solved. Hence, its optimal solution can only be obtained by numerical methods.

3. Supervised Learning with DNN

This section introduces the supervised learning approach with DNN to realize velocity maximization with constrained arrival angle for midcourse guidance. A training set covering the full-envelope flight conditions is generated by NLP software. Then, two DNNs are trained with supervised learning approach by two methods according to different types of guidance command. And the performance of both DNNs are compared with numerical simulations to support the proposed guidance algorithm.

3.1 Generation of Training Data

3.1.1 Generating Optimal Trajectories

The initial conditions and terminal constraints of the midcourse guidance scenario are listed in Table 1. And the aerodynamic coefficients of the missile are shown in Table 2.

Table 1: **Initial conditions and terminal constraints**

Parameter	Value
Initial position, (x_0, y_0)	$(0m, 0m)$
Initial flight path angle, φ_0	$[45°, 60°]$
Initial velocity, V_0	$300m/s$
Initial mass, m_0	$400kg$
Final horizontal position, x_f	$[5000m, 30000m]$
Final vertical position, y_f	$2000m$
Final flight path angle, φ_f	$-90°$
Final flight time, t_f	$0.0045s_0 + 15$

Table 2: **Aerodynamic coefficients and derivatives**

Ma	C_{D0}	$C_{D\alpha}(rad^{-2})$	$C_{L\alpha}(rad^{-1})$
0.1	0.4298	20.9917	22.7206
0.2	0.4010	21.3565	22.6815
0.3	0.3968	21.6483	22.7522
0.4	0.4014	21.9583	22.7321
0.5	0.4016	22.0860	22.8496
0.6	0.4101	22.4690	23.1427
0.7	0.4229	23.0070	23.3920
0.8	0.4529	23.6362	23.7949
0.9	0.5546	21.0282	24.5455
1	0.6100	19.0345	25.4123

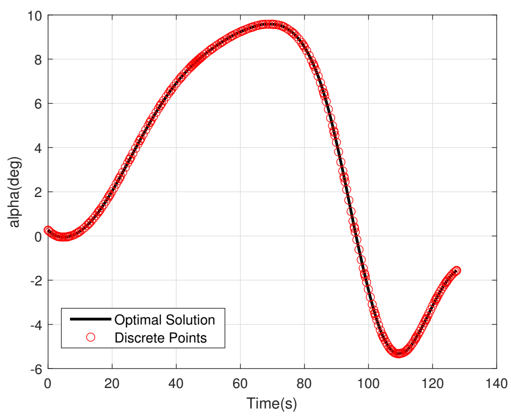
Where s_0 is the range between missile and PHP at the initial time. The characteristic area of the missile is $S = 0.06m^2$ and the thrust is $T = 2000N$ throughout the midcourse guidance. The specific impulse $I_{sp} = 2500N \cdot s/kg$, and the AoA of the missile is limited to the range $[-20°, 20°]$. Fitting nonlinear relations using a DNN requires a large amount of training data to have good results, hence 15000 initial conditions (φ_0, s_0) are generated using the average distribution. Then 15000 corresponding optimal control problems are solved by utilizing the well-known Gauss pseudospectral optimization software (GPOPS). And the maximum error tolerance is set to be 1×10^{-8}.

3.1.2 Sampling Optimal Trajectories

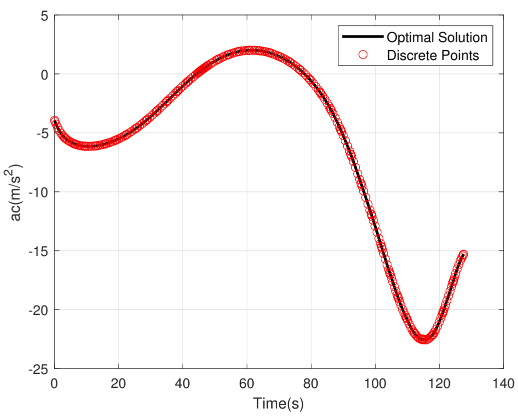
The massive data for training DNNs consists of state-action pairs are extracted from optimal trajectories. This paper uses the discrete points directly as sampling points, which are generated from the solving process of GPOPS. In detail, the core method of GPOPS for solving optimal control problems is the pseudospectral method (PSM), which discretizes both the control and state variables over the entire time range and then transforms them into NLP for solving. There are more discretization points at locations where the parameters vary drastically to achieve the predetermined maximum tolerance error, likewise, will result in more training data at these locations, which helps the DNN learn more adequately. Fig.2 shows the sampling points on the angle of attack profile and the lateral acceleration profile.

3.2 The Direct Method

The direct method means that the optimal control progress is done entirely by the DNN, i.e., let the DNN learns the mapping between missile's states $\mathbf{s}$ and the optimal lateral acceleration a_c. Therefore the input of the DNN is the state vector. The expression is $\mathbf{s} = [r, V, \varphi, \lambda, \varphi_0, s_0]^T$, where



(a) Sampling points for angle of attack



(b) Sampling points for lateral acceleration

Figure 2 – Sampling points location

$r = \sqrt{(x - x_f)^2 + (y - y_f)^2}$ is the distance between the missile and the PHP. φ_0 is the initial flight path angle. $s_0 = x_f - x_0$ is the horizontal distance between the launch point and PHP. The DNN called $\mathcal{N}_1$ in the direct method is a feedforward neural network with 3 hidden layers and 12 neurons per layer, as shown in Fig.4. The first layer $L^{(0)}$ is the input layer with 6 neurons, the same number of dimensions as $\mathbf{s}$. $L^{(1)}$ to $L^{(3)}$ is the hidden layer. $L^{(4)}$ is the output layer. The former layer is fully connected to the latter layer. The structure and internal calculations of $\mathcal{N}_1$ can be expressed as follows [14]:

$$\mathcal{N}_1 = \mathcal{N}(\mathbf{s}, a_c) : \begin{cases} L^{(0)}[\mathbf{s}, a_c] \\ L^{(i+1)} = \sigma_i(\mathbf{W}_i L^{(i)} + \mathbf{b}_i), \ \forall\ i = 0, 1, 2, 3 \\ a_c = L^{(4)} \end{cases} \tag{10}$$

where $\mathbf{W}_i$ denotes the weight matrix and $\mathbf{b}_i$ denotes the bias matrix, whose size is related to the number of neurons in each layer. σ_i is the activation function that enables the DNN to fit the nonlinear functions. All activation functions in this paper use $tanh$.

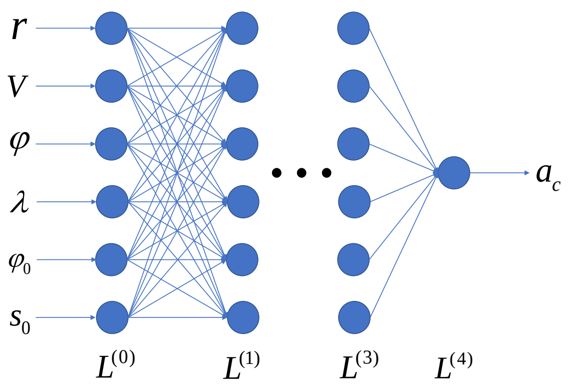


Figure 4 – Structure of $\mathcal{N}_1$

However, the direct method has the obvious drawback that it is an open-loop control method. In practice, model uncertainties and atmospheric disturbances can cause the missile to deviate from the correct optimal trajectory. At this moment, the $\mathbf{s}$ that should be entered as $\mathcal{N}_1$ becomes $\mathbf{s}'$, then $\mathcal{N}_1$ will give the wrong control command with no way to correct it. According to Bellman's optimality principle, if a process is optimal, then the current strategy must be optimal for the current state. Therefore, the error of guidance will keep accumulating. Based on this, the indirect method is proposed to overcome this drawback.

3.3 The Indirect Method

There are two possible solutions to the shortcomings of the direct method: one is to add the state vector $\mathbf{s}^{(t-1)}$ or output $a_c^{(t-1)}$ of the previous moment to the input $\mathbf{s}^{(t)}$ of the next moment, forming a closed-loop control. The other is to combine DNNs with traditional guidance methods. With the stability and mathematical guarantees of traditional methods, the open-loop characteristics of direct methods can be compensated and the robustness will be better.

Inspired by the biased proportional navigation guidance (BPNG), a trajectory optimization method combining BPNG and DNN is proposed, i.e., the indirect method. BPNG is generally used in the terminal guidance process to ensure that the missile hits the target under certain angular constraints. When the target is stationary, the expression for BPNG is given as follows [19]:

$$a_c = NV\dot{\lambda} + \frac{V(\varphi_f - \varphi) - N(\varphi_f - \lambda)}{t_{go}} = NV\dot{\lambda} + a_b = \mathcal{P} + a_b \tag{11}$$

The meaning of each physical quantity in Eq.(11) is the same as that described in Fig.1. N is the artificially set coefficient, usually from 2 to 6. a_b is called bias acceleration. t_{go} is the remaining flight time, the accuracy of the estimation of this value will directly affect the guidance effect of BPNG. Although there are many estimation methods [20–22], the estimation of t_{go} is still difficult. Note that the first term $\mathcal{P} = NV\dot{\lambda}$ in Eq.(11) guarantees final miss distance tends to zero; the second term ensures arrival angle constraint. Therefore, a DNN identical to $\mathcal{N}_1$ called $\mathcal{N}_2$ is used to learn the mapping relationship between a_b and the state vector $\mathbf{s}$, i.e., $\mathcal{N}_2 = \mathcal{N}(\mathbf{s}, a_b)$. Then the coefficient N is fixed as 3 and $\mathcal{P}$ is calculated directly. Finally, $\mathcal{P}$ works together with $\mathcal{N}_2$ to output lateral acceleration and control the missile to reach the PHP with maximum terminal velocity and specified arrival angle. Note that in the midcourse guidance process, the angular velocity $\dot{\lambda}$ of the line-of-sight angle cannot be obtained directly from the onboard seeker, therefore it needs to be calculated using the following equation:

$$\dot{\lambda} = \frac{-V\sin(\varphi - \lambda)}{r} \tag{12}$$

In general, the indirect method is to let $\mathcal{N}_2$ work together with the proportional navigation guidance to complete the midcourse guidance, where $\mathcal{N}_2$ outputs the bias acceleration a_b that is summed with $\mathcal{P}$ as the lateral acceleration a_c. The specific working process is shown in Fig.5. Even if there are errors in the output of $\mathcal{N}_2$ due to external disturbances, the presence of $\mathcal{P}$ will offset part of the error to some extent. Subsequent numerical simulations also prove this.

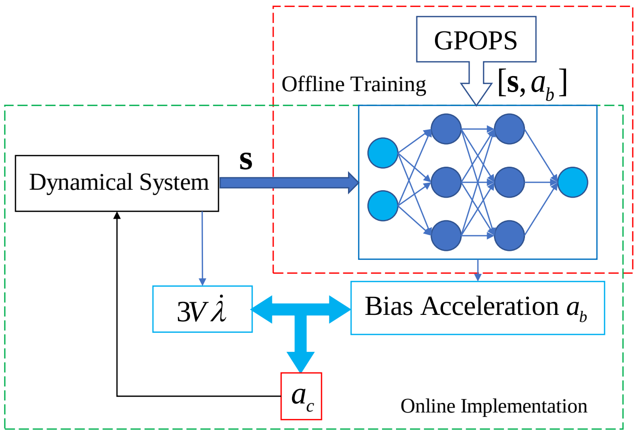


Figure 5 – Scheme of the indirect method

3.4 Training of DNNs

Using the above method, a total of 5442627 optimal state-lateral acceleration pairs $[\mathbf{s}, a_c]$ are obtained, which are used as the data set $\mathcal{D}_1$ for training $\mathcal{N}_1$. The optimal control command a_c in $\mathcal{D}_1$ are transformed into a_b according to Eq.(11) and used as the data set $\mathcal{D}_2$ for $\mathcal{N}_2$. 70% of the samples in the dataset are used as the training set, 15% as the validation set, and 15% as the test set. Due to the large order of magnitude difference between the data, normalization is required. Otherwise the training time will increase or the training is likely to fail. The normalization method used in this paper is as follows, mapping data to $(-1, 1)$:

$$x_{norm} = \frac{2}{x_{max} - x_{min}}(x - x_{min}) - 1 \tag{13}$$

where x_{norm} represents the normalized data, x_{max} is the maximum value of the data set, and similarly, x_{min} is the minimum value.
The loss function reflects how well the DNN fits the data and is also the optimization target of the DNN during training. Mean square error (MSE) is generally used in the fitting problem, calculated as follows:

$$MSE = \frac{\sum_{i=1}^{N}(u - u^*)^2}{N} \tag{14}$$

where u is the predicted value of the DNN and u^* is the target value.
The existing optimization algorithms are first-order optimization algorithms, represented by stochastic gradient descent (SGD) and Adam. And second-order optimization algorithms, represented by Levenberg-Marquardt algorithm (L-M) and conjugate gradient method (CG). The second-order optimization algorithms additionally take into account the second-order derivative information. Therefore they can adjust the parameters more accurately [23–25]. The numerical simulations reveal that the convergence is faster using L-M for smaller networks like $\mathcal{N}_1$ and $\mathcal{N}_2$. Therefore DNNs are trained by the L-M algorithm for 200 epochs. L-M uses the following expression to update the parameters:

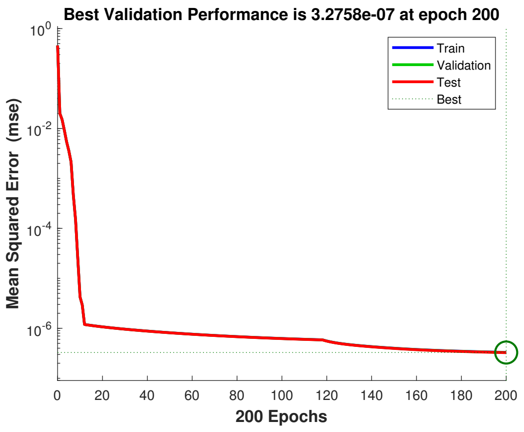
$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\mathbf{J}^T\mathbf{J} + \mu\mathbf{I})^{-1}\mathbf{J}^T\mathbf{e} \tag{15}$$

where $\mathbf{J}$ is the Jacobi matrix of the error function, $\mathbf{I}$ is the identity matrix, $\mathbf{e}$ is the error vector. μ is the trust-region radius and the initial value is 0.001. When the loss function decreases, let $\mu' = 0.1\mu$ so that the L-M algorithm is closer to the Newton method resulting in a faster convergence rate. Otherwise $\mu' = 10\mu$.

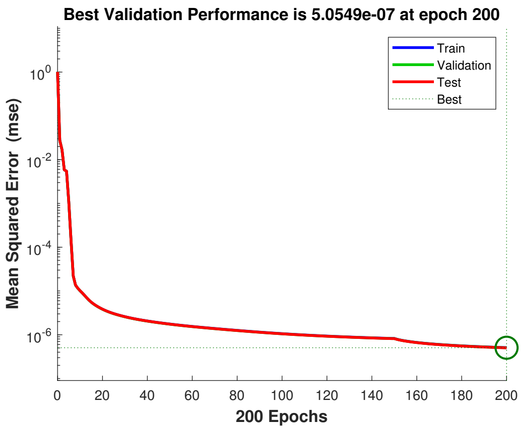
3.5 Simulation Results

3.5.1 Performance of DNNs

The training histories of $\mathcal{N}_1$ and $\mathcal{N}_2$ are shown in Fig.6. It can be seen that DNNs converge very fast using the L-M algorithm. Both mean square errors are very low at the end of training progresses. The histograms of the error distribution of DNNs on the validation and test sets are shown in Fig.8.
To evaluate the performance of DNNs in midcourse guidance, 500 different initial conditions in the range presented in Table 1 are randomly generated as the test conditions. Then, using the well-trained $\mathcal{N}_1$ and $\mathcal{N}_2$ to complete the whole midcourse guidance process. Finally, the performances are statistically measured in four aspects: horizontal distance error $\Delta x = |x_{fn} - x^*|$ (the midcourse guidance is considered to be finished when the vertical height y of the missile drops to $2km$), final velocity error $\Delta V = |V_{fn} - V^*|$, arrival angle error $\Delta\varphi = |\varphi_{fn} - \varphi^*|$ and time error $\Delta t = |t_{fn} - t^*|$. The superscript $*$ represents the optimal value and the subscript fn represents the final state value of the missile under the control of DNNs. The mean, maximum, median, and standard deviation of the four types of errors are shown in Tables 3. Since the difference between the performance of $\mathcal{N}_1$ and $\mathcal{N}_2$ cannot be seen from the pictures, only 20 simulations of $\mathcal{N}_2$ are selected to show in Fig.10.
It can be seen that both $\mathcal{N}_1$ and $\mathcal{N}_2$ can complete midcourse guidance. But the horizontal distance error of the missile under control of $\mathcal{N}_1$ is much larger than that of $\mathcal{N}_2$. The two DNNs are very close in velocity error, but $\mathcal{N}_1$ is slightly better than $\mathcal{N}_2$. The arrival angle error under control of $\mathcal{N}_2$ is slightly better than that of $\mathcal{N}_1$, but the time error is significantly larger than that of $\mathcal{N}_1$. Since Δx is larger for $\mathcal{N}_1$ and Δt is larger for $\mathcal{N}_2$, more experiments are conducted to investigate these two errors.

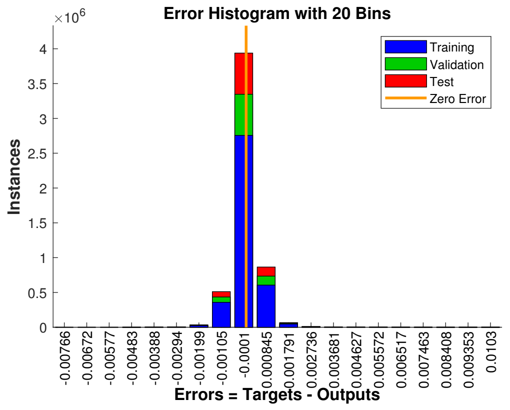


(a) Loss functions for $\mathcal{N}_1$

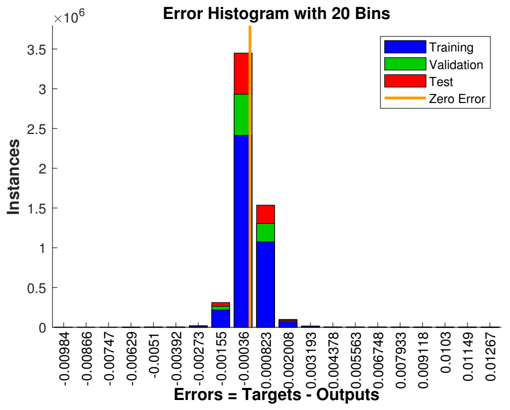


(b) Loss functions for $\mathcal{N}_2$

Figure 6 – Loss functions for $\mathcal{N}_1$ and $\mathcal{N}_2$



(a) Error of $\mathcal{N}_1$



(b) Error of $\mathcal{N}_2$

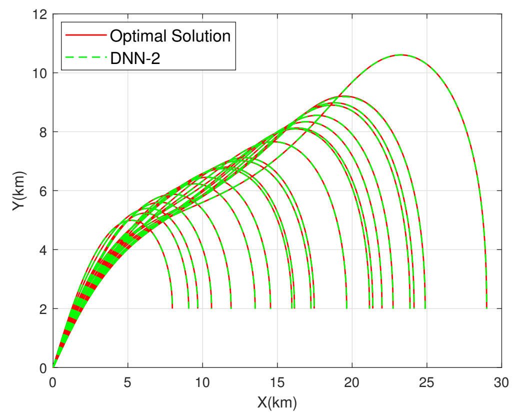
Figure 8 – Error histograms of $\mathcal{N}_1$ and $\mathcal{N}_2$

By making the distribution of Δx and Δt with s_0, it is found that as s_0 increases, the Δx for $\mathcal{N}_1$ tends to increase, while the Δt for $\mathcal{N}_2$ also has a trend to increase, as shown in Fig.12. As discussed in Chapter 3.2using $\mathcal{N}_1$ for midcourse guidance, errors will accumulate with the total distance, leading to a continuous decrease in the accuracy of missiles, which seriously affects the use of long-range missiles in practice. Using $\mathcal{N}_2$ substantially reduces the horizontal distance error but the time error also increases meanwhile. This shows that although $\mathcal{N}_2$ can correct for errors during flight, the time error caused by the missile deviating from its optimal trajectory cannot be compensated for in subsequent flights. This problem also exists in conventional methods. Therefore the indirect method is effective. Its time error can be reduced by minimizing the prediction error of $\mathcal{N}_2$. While the prediction error of $\mathcal{N}_2$ can be realized by increasing the size of the training set, optimizing the network structure, and adjusting hyperparameters.

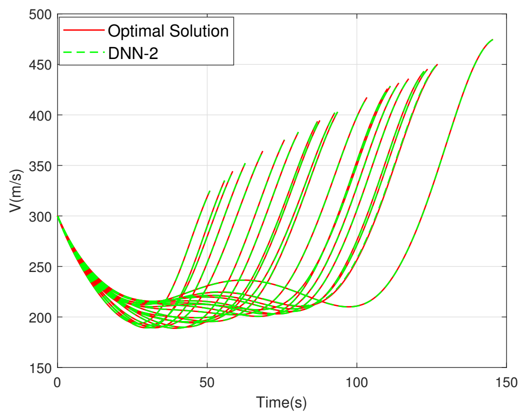
3.5.2 *Robustness Analysis*

Robustness describes a system's ability to survive adverse situations. Therefore a reliable system must have a high level of robustness. Although DNN-based optimal control methods have excellent performance, the flight process is ideally noiseless. So the performances of $\mathcal{N}_1$ and $\mathcal{N}_2$ under noise are tested as a measure of their robustness.

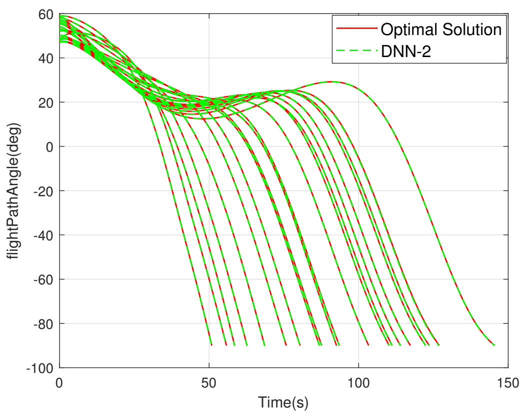
For the DNNs input $\mathbf{s} = [r,\ V,\ \varphi,\ \lambda,\ \varphi_0,\ s_0]^T$, the last two parameters φ_0 and s_0 are initially given and will not be disturbed during the flight. The first four parameters are measured and calculated by the sensors and GPS. So there will be noise interference in actual use. The White Gaussian Noise is added with mean 0 and standard deviation 0.5, 0.5, 0.005, 0.005 to the four parameters r, V, φ and



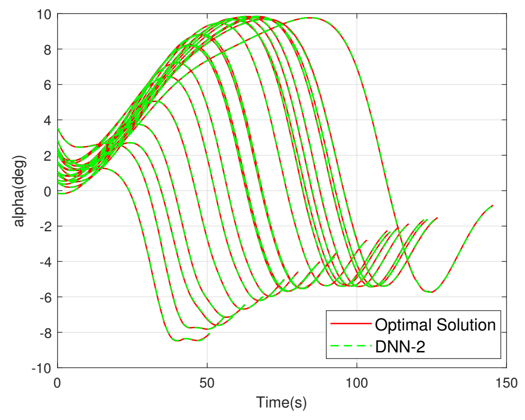
(a) Trajectory



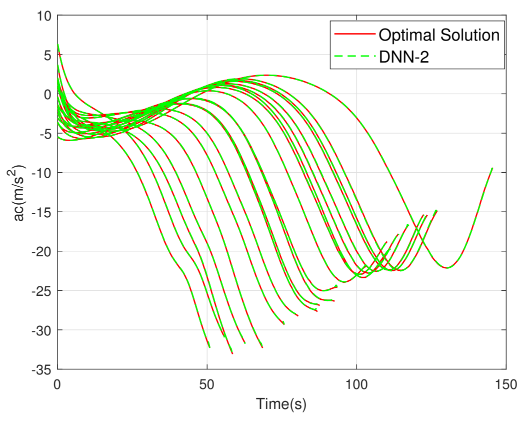
(b) Velocity



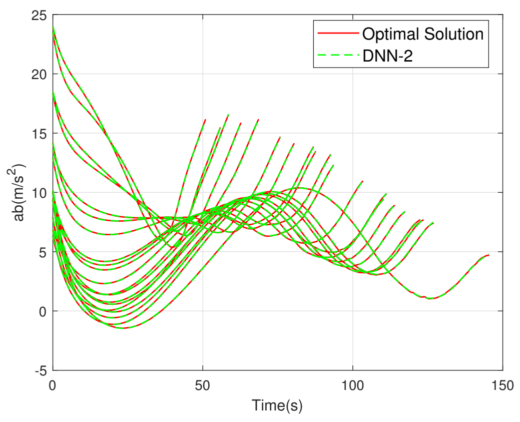
(c) Flight path angle



(d) Angle of attack



(e) Lateral acceleration

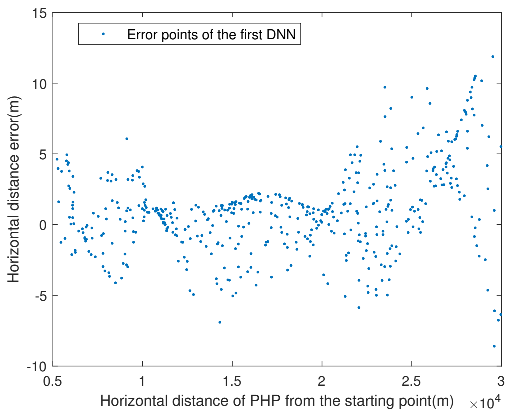


(f) Bias acceleration

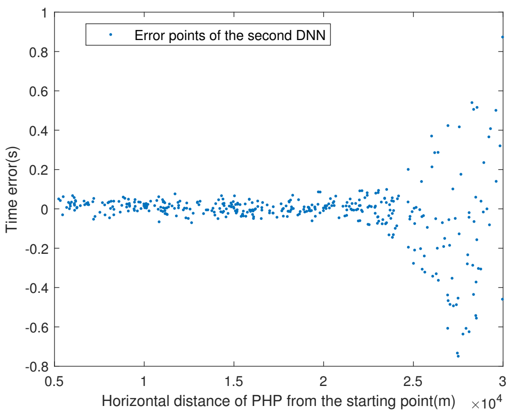
Figure 10 – The midcourse guidance performance of $\mathcal{N}_2$

Table 3 – **The absolute errors of the two DNNs after the midcourse guidance**

DNN	Error	Mean	Max	Median	Standard Deviation
$\mathcal{N}_1$	$\Delta x\ (m)$	2.3638	10.5005	1.5304	2.1827
	$\Delta V\ (m/s)$	0.1176	0.5069	9.2802×10^{-2}	9.5298×10^{-2}
	$\Delta\varphi(rad)$	2.2218×10^{-3}	1.0113×10^{-2}	1.9481×10^{-3}	1.7564×10^{-3}
	$\Delta t\ (s)$	2.1413×10^{-2}	7.0500×10^{-2}	1.9978×10^{-2}	1.4635×10^{-2}
$\mathcal{N}_2$	$\Delta x(m)$	1.4916×10^{-2}	0.1482	6.7419×10^{-3}	2.1154×10^{-2}
	$\Delta V\ (m/s)$	0.1563	0.8726	0.1292	0.1307
	$\Delta\varphi(rad)$	1.8503×10^{-3}	7.5579×10^{-3}	1.4152×10^{-3}	1.5307×10^{-3}
	$\Delta t\ (s)$	7.9789×10^{-2}	0.8736	3.1015×10^{-2}	0.1388



(a) Distribution of position error with distance under $\mathcal{N}_1$



(b) Distribution of time error with distance under $\mathcal{N}_2$

Figure 12 – Distribution of errors with distance

λ, respectively. Then 500 test cases are randomly initialized to evaluate the performance of the two DNNs under noise and the comparison results with the optimal solution are shown in Fig.14, where the error is obtained by subtracting the optimal solution from the actual performance of the DNNs.
It can be seen that there is a considerable increase in the position error of $\mathcal{N}_1$ after adding noise, indicating that the direct method is not very resistant to interference. However, excluding some individual angular errors that are large, the last three errors are kept at a very small value. This indicates that $\mathcal{N}_1$ can maintain the overall shape of the optimal trajectory, although it cannot correct the in-flight errors. All four errors of $\mathcal{N}_2$ change very little after adding noise, indicating that the indirect method is more resistant to interference than the direct method. The proportional guidance term $\mathcal{P}$ plays an important role which can compensate the errors of the output of $\mathcal{N}_2$ to a large extent.
In general, both the direct method and the indirect method have certain anti-interference ability as well as perform very well in the final velocity and arrival angle. However, for tasks requiring high positional accuracy, the indirect method is much better than the direct method. On the contrary, for tasks with high time accuracy requirements, it is better to choose the direct method.

4. Transfer Learning under Different Aerodynamic Coefficients

4.1 Deep Transfer Learning Overview

The performance of deep neural networks outside the scope of the training set is generally uncertain. In practice, not only the noise, but also the DNN working environment or initial conditions may change. For example, in the missile design phase, the aerodynamic parameters will change due to the modification of the missile configuration, and the previously trained DNN will be unsuitable to the current situation. Fig.15 shows the performance of DNNs after changing the aerodynamic parameters in Table 2, which are increased by 10%. $\mathcal{N}_1$ deviates severely from the optimal solution with very

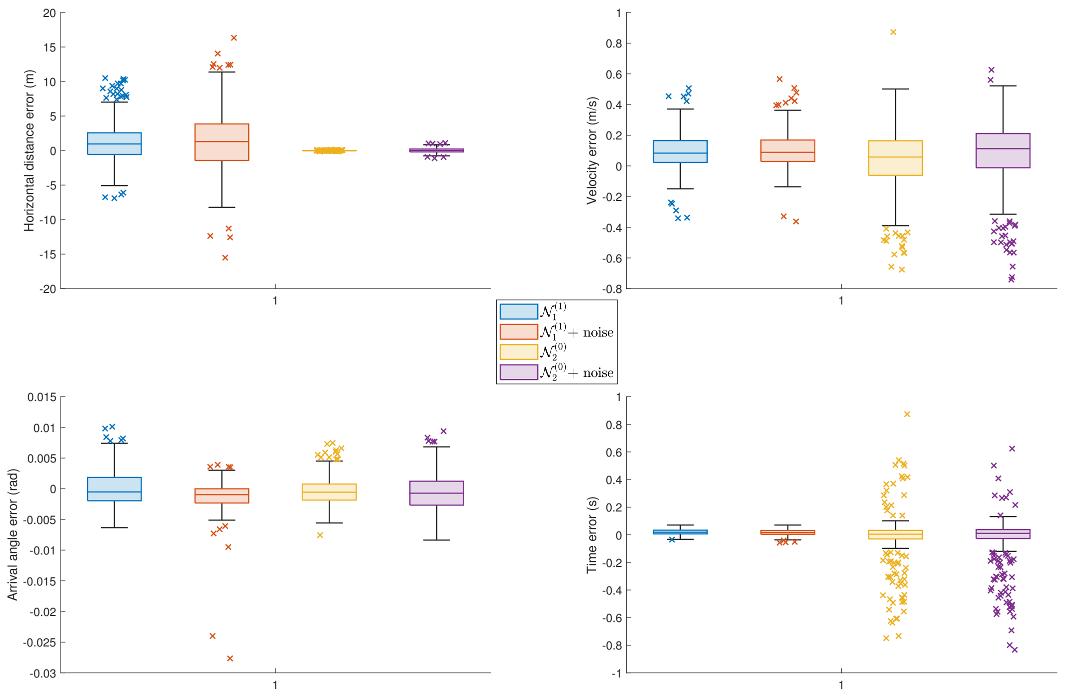
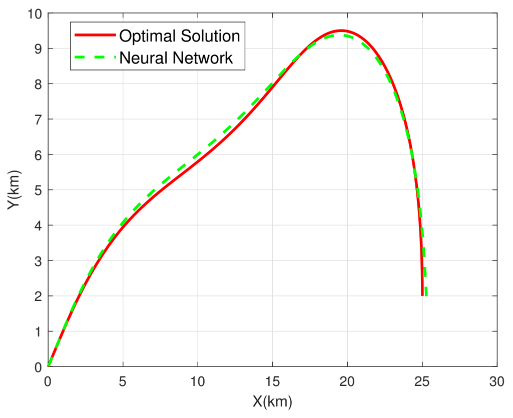


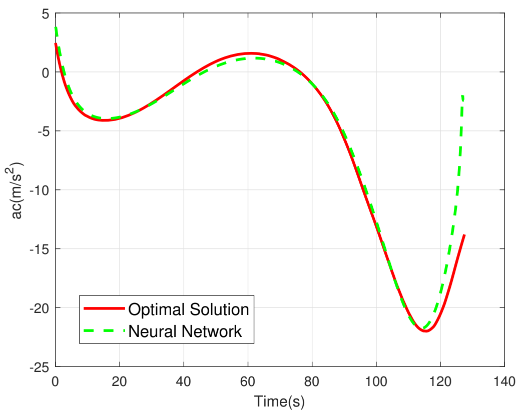
Figure 14 – Performance of two DNNs under noise

large position errors (over $250m$) and arrival angle errors (over $7deg$). While $\mathcal{N}_2$ reaches the PHP more accurately but has large errors in other parameters as well. The best way is to retrain the DNN, but the production of a new training set and the learning process are quite time-consuming, which can seriously slow down the design efficiency. And for more complex tasks, generating a training set that encompasses all cases will become extremely huge [14]. Transfer learning can reduce training samples and learning time by combining knowledge learned from previous tasks to new tasks, where DNN-based transfer learning is called deep transfer learning (DTL). Its most notable application is in the field of computer vision. Large networks require days to weeks of training time which ordinary users cannot afford. DTL on these networks allows them to learn new classification tasks in a day. Thus the advantages of DTL are obvious and significant.
A more precise definition of DTL is as follows: The domain $D = \{\mathcal{X}, P(X)\}$, where $\mathcal{X}$ is the feature space, $P(X)$ is the edge probability distribution, and $X = \{x_1, ..., x_n\} \in \mathcal{X}$. The task $T = \{y, f(x)\}$, consists of two parts: label space y and target prediction function $f(x)$, which is a non-linear function that reflected DNN. Next, given the source domain D_s and the learning task T_s, the target domain D_t and the learning task T_t. DTL utilizes the knowledge in D_s and T_s to improve the predictive function $f_T(\cdot)$. Note that $D_s = D_t$ and $T_s = T_t$ do not hold simultaneously, and generally D_s is much larger than D_t [26, 27].
There are currently four categories of DTL:

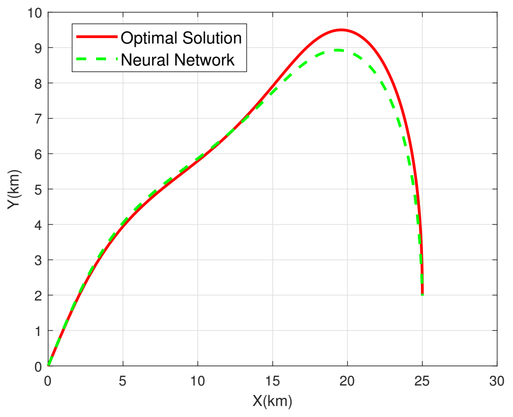
- Instances-based. When there is some similarity between two tasks, the method assigns certain weights to some instances in the source domain and utilizes them as part of the target domain. The representative method is TrAdaBoost.
- Mapping-based. This method considers that the instances in the source and target domains will be more similar when mapped to some new data space and transfer learning is performed on this basis. The representative method is transfer component analysis (TCA).
- Network-based. This method retains the trained part of the network structure and parameters, and trains on this basis to complete transfer learning. A method known as Fine-tune is widely



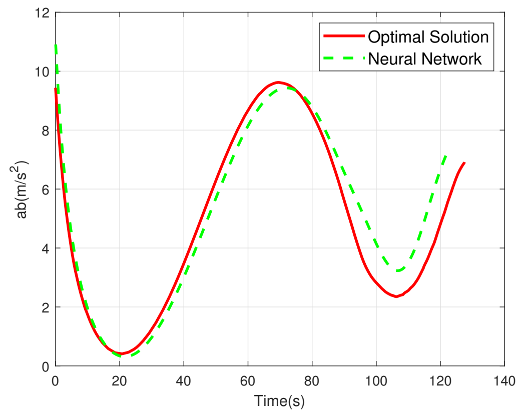
(a) Trajectory of $\mathcal{N}_1$



(b) Lateral acceleration of $\mathcal{N}_1$



(c) Trajectory of $\mathcal{N}_2$



(d) Bias acceleration of $\mathcal{N}_2$

Figure 15 – Performance of $\mathcal{N}_1$ and $\mathcal{N}_2$ with new aerodynamic coefficients

used in the field of computer vision. It enables very good transfer learning between two tasks with similar data features [28].

- Adversarial-based. This method uses DNNs to extract features from two domains firstly and then uses the generative adversarial network (GAN) to find migratable features between two domains.

Changing the aerodynamic coefficients, the flight trajectory and optimal control profile of the missile are similar to the previous shapes. This indicates a high similarity of midcourse guidance missions with different aerodynamic coefficients. The Fine-tune method is chosen for DTL which considers that the structure and parameters of the well-trained DNN contain the knowledge of the previous task. Therefore some structure and parameters of the DNN are used as part of the new network and are not involved in the subsequent back propagation process, i.e., the parameters in it are frozen and only the rest are trained. The implementation of Fine-tune is shown in Fig.17.

4.2 Preparation for Deep Transfer Learning

All the aerodynamic coefficients in Table 2 are increased by 10% as a new task scenario. Using $\mathcal{N}_1$ and $\mathcal{N}_2$ from the previous chapter as the initial networks for DTL. We attempt all possibilities without changing structures of the original networks, with a fixed number of layers from 0 to 3, using Fine-tune for DTL respectively. The number of fixed layers is always counted from front to back, i.e., the fixed layers must be in front of the free layers, otherwise, backpropagation is not available. Being fixed means that the $\mathbf{W}_i$ and $\mathbf{b}_i$ in these layers are no longer updated.
Then, 2000 initial conditions are randomly generated and solved the corresponding optimal control problems by GPOPS. The optimal state-action pairs are extracted as in the previous chapter to form

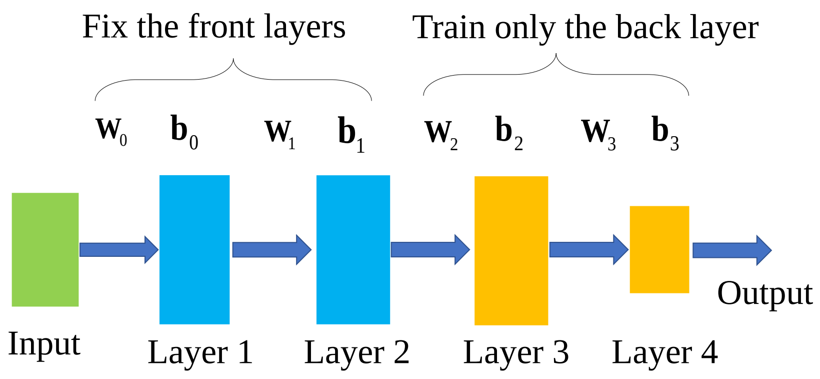

Fix the front layers
Train only the back layer
$\mathbf{W}_0$ $\mathbf{b}_0$ $\mathbf{W}_1$ $\mathbf{b}_1$ $\mathbf{W}_2$ $\mathbf{b}_2$ $\mathbf{W}_3$ $\mathbf{b}_3$
Input
Layer 1
Layer 2
Layer 3
Layer 4
Output

Figure 17 – Implementation of Fine-tune

Table 4 – **Performance of $\mathcal{N}_1$ after transfer learning**

Error	Fixed number of layers	Mean	Max	Median	Standard Deviation
$\Delta x\ (m)$	3	39.1595	78.0597	39.8762	21.9910
	2	16.4219	77.6405	14.0368	11.6863
	1	**4.8678**	**25.1094**	**3.0978**	**5.0036**
	0	5.4615	51.6793	4.1743	6.3752
$\Delta V\ (m/s)$	3	8.9060	15.0183	9.2083	2.8790
	2	0.1792	0.5486	0.1494	0.1340
	1	0.1371	0.3637	0.1204	9.6052×10^{-2}
	0	$\mathbf{9.3082\times10^{-2}}$	**0.3158**	$\mathbf{7.8056\times10^{-2}}$	$\mathbf{7.2538\times10^{-2}}$
$\Delta\varphi\ (rad)$	3	6.1871×10^{-3}	1.7720×10^{-2}	4.6691×10^{-3}	5.0217×10^{-3}
	2	4.8114×10^{-3}	1.5359×10^{-2}	4.3848×10^{-3}	3.2006×10^{-3}
	1	$\mathbf{2.0876\times10^{-3}}$	$\mathbf{5.7058\times10^{-3}}$	$\mathbf{1.8368\times10^{-3}}$	$\mathbf{1.3646\times10^{-3}}$
	0	2.1257×10^{-3}	1.0644×10^{-2}	1.9517×10^{-3}	1.6188×10^{-3}
$\Delta t\ (s)$	3	0.1060	0.4408	9.8758×10^{-2}	7.2904×10^{-2}
	2	8.2548×10^{-2}	0.3204	6.7232×10^{-2}	6.6772×10^{-2}
	1	3.8367×10^{-2}	0.2592	3.5833×10^{-2}	3.1030×10^{-2}
	0	$\mathbf{3.0445\times10^{-2}}$	**0.2092**	$\mathbf{2.8112\times10^{-2}}$	$\mathbf{2.5597\times10^{-2}}$

a training set with a total of 717723 samples. The optimization algorithm uses Adam and the learning rate is set to 0.0001, with 1000 epochs trained.

4.3 Simulation Results

4.3.1 Performance of DNNs

After training, 500 initial conditions are randomly generated for numerical simulations. The errors between the simulation results and the optimal solution are shown in Table 4 and Table 5. A total of 8 networks are trained and tested, and these networks are denoted by $\mathcal{N}_1^{(i)}$ and $\mathcal{N}_2^{(i)}$, with i representing the fixed number of layers. Also to demonstrate the effectiveness of transfer learning, two brand new DNNs (randomly initialized with all parameters) without fixing any parameters, using the same training set and initial conditions for training and testing. It is found that the midcourse guidance results of the new DNNs are far from the optimal solution, regardless of whether the direct or indirect method is used. This indicates that the new DNNs cannot adequately learn the mapping relationship between states and optimal actions at the size of the training set used for transfer learning.

It can be seen that $\mathcal{N}_1^{(1)}$ has the smallest position error and arrival angle error, and conversely $\mathcal{N}_1^{(0)}$ has

Table 5 – **Performance of $\mathcal{N}_2$ after transfer learning**

Error	Fixed number of layers	Mean	Max	Median	Standard Deviation
$\Delta x\ (m)$	3	5.5535×10^{-2}	0.2235	4.0610×10^{-2}	5.2726×10^{-2}
	2	3.6694×10^{-2}	0.1556	2.5375×10^{-2}	3.6728×10^{-2}
	1	2.6827×10^{-2}	0.2498	1.3022×10^{-2}	3.6085×10^{-2}
	0	$\mathbf{1.4306 \times 10^{-2}}$	$\mathbf{9.7927 \times 10^{-2}}$	$\mathbf{6.6408 \times 10^{-3}}$	$\mathbf{1.7571 \times 10^{-2}}$
$\Delta V\ (m/s)$	3	0.8291	3.6870	0.6645	0.6042
	2	0.2927	0.9092	0.1861	0.2503
	1	0.2376	1.0197	0.1951	0.2120
	0	$\mathbf{0.1650}$	$\mathbf{0.5625}$	$\mathbf{0.1599}$	$\mathbf{0.1110}$
$\Delta \varphi\ (rad)$	3	6.0194×10^{-3}	2.2771×10^{-2}	5.3680×10^{-3}	4.3127×10^{-3}
	2	4.5723×10^{-3}	1.5008×10^{-2}	4.3258×10^{-3}	3.0499×10^{-3}
	1	3.0562×10^{-3}	1.2350×10^{-2}	2.5263×10^{-3}	2.3292×10^{-3}
	0	$\mathbf{1.9629 \times 10^{-3}}$	$\mathbf{8.1339 \times 10^{-3}}$	$\mathbf{1.2980 \times 10^{-3}}$	$\mathbf{1.6473 \times 10^{-3}}$
$\Delta t\ (s)$	3	0.5541	3.2908	0.4118	0.6230
	2	0.2286	0.9159	0.1275	0.2397
	1	0.1689	1.0159	8.7714×10^{-2}	0.2174
	0	$\mathbf{8.4987 \times 10^{-2}}$	$\mathbf{0.5659}$	$\mathbf{5.3646 \times 10^{-2}}$	$\mathbf{0.1067}$

a smaller velocity error and time error. Overall, $\mathcal{N}_1^{(1)}$ outperforms $\mathcal{N}_1^{(0)}$ because the position error is the smallest and the other three errors are within acceptable limits. In contrast, the transfer learning of $\mathcal{N}_2$ presents a different result. As the number of training layers increases, the error gradually decreases. $\mathcal{N}_2^{(0)}$ reaches the minimum error and even performs comparably to $\mathcal{N}_2$. The training sets used in the transfer learning of both DNNs are the same except for the different outputs, but the performance of $\mathcal{N}_1$ after DTL is considerably worse than that of $\mathcal{N}_2$, probably because both DNNs are in underfitting states under the current training data volume, but the special mechanism of the indirect method leads to a good performance when the learning of $\mathcal{N}_2$ is not so adequate.
Numerical simulations demonstrate that DTL with Fine-tune is feasible and that DTL with $\mathcal{N}_2$ is more practical and reliable. Therefore, whether using DNN for midcourse guidance or transfer learning, the indirect method performs no worse than the direct method. Thus, it is necessary to study how to combine DNN with existing methods, rather than giving the task to DNNs exclusively.

4.3.2 Robustness Analysis

DNNs are expected to be practically feasible after transfer learning, so it must be as robust as a retrained network. Fig.18 shows the performance of $\mathcal{N}_1^{(1)}$ and $\mathcal{N}_2^{(0)}$ before and after the addition of noise. These tests are modeled after the method in Chapter 3, and the number of tests is still 500. It can be seen that the $\mathcal{N}_1^{(1)}$, although performing well overall, has some cases of large errors. On the contrary, $\mathcal{N}_2^{(0)}$ still manages to maintain very high accuracy. This result proves that the transfer learning of $\mathcal{N}_2$ is more feasible and practical than that of $\mathcal{N}_1$.

4.3.3 Time consumption analysis

To demonstrate the real-time characteristics of DNN-based guidance and the effectiveness of transfer learning, we counted the time consumed by the related work, displayed in Table 6. All simulations are performed in Matlab 2020b and PyTorch 1.7.1, with a laptop using Intel i7-7700HQ processor at 2.8 GHz and GTX1050Ti. It takes close to 2 hours to train a DNN that can accurately complete midcourse guidance, but transfer learning takes less than 10 minutes, which is more than 10 times faster than retraining a new DNN. When DNN works, it takes about 0.9 ms to generate an optimal control instruction. Throughout this paper, our simulations use a time step of 0.05s, which is far from

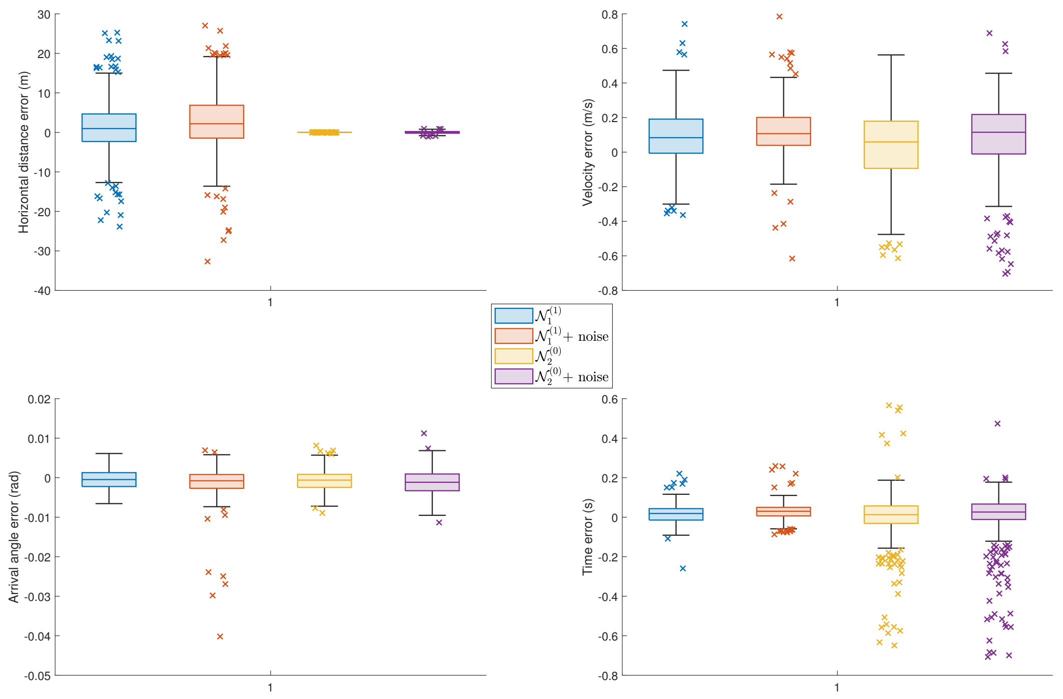


Figure 18 – Performance of two DNNs under noise

the response time of DNN. If a shorter time step is used, the accuracy of the guidance will be further improved.

Table 6 – **Performance of $\mathcal{N}_2$ after transfer learning**

Tasks	Approaches	Time consumption
Training of $\mathcal{N}_1$ and $\mathcal{N}_2$	L-M, 200 epoch	110 min
Fine-tune ($\mathcal{N}_1^{(0)}$ and $\mathcal{N}_2^{(0)}$)	Adam, 1000 epoch	10 min
Fine-tune ($\mathcal{N}_1^{(0)}$ and $\mathcal{N}_2^{(0)}$)	L-M, 100 epoch	7 min
Solving the optimal control problem	GPOPS	Average 0.85 sec
$\mathcal{N}_1$ and $\mathcal{N}_2$ generate an optimal instruction	/	0.0009 sec

5. Conclusion

This paper proposes a deep learning-based approach to solve the missile midcourse guidance problem for velocity maximization with a constrained arrival angle. A training set covering the full-envelope conditions is first established by NLP software. Then, a supervised-learning approach is utilized to train the DNN based on the training set. To improve the transportability of a trained DNN, transfer learning is introduced to promote the adaptivity of the proposed algorithm. The main advantage of the proposed approach lies in its ability of generating optimal behaviors with requiring relatively lower computational power, which is suitable for practical implementation. Numerical simulations are performed to support the proposed algorithm.

6. Contact Author Email Address

hongyanli@bit.edu.cn

7. Copyright Statement

References

[1] Nahshon Indig, Joseph Z Ben-Asher, and Nathan Farber. Near-optimal spatial midcourse guidance law with an angular constraint. *Journal of Guidance, Control, and Dynamics*, 37(1):214–223, 2014.

[2] Hongyan Li, Shaoming He, Jiang Wang, Hyo-Sang Shin, and Antonios Tsourdos. Near-optimal midcourse guidance for velocity maximization with constrained arrival angle. *Journal of Guidance, Control, and Dynamics*, 44(1):172–180, 2021.

[3] In-Soo Jeon, Mark Karpenko, and Jin-Ik Lee. Connections between proportional navigation and terminal velocity maximization guidance. *Journal of Guidance, Control, and Dynamics*, 43(2):383–388, 2020.

[4] David A Benson, Geoffrey T Huntington, Tom P Thorvaldsen, and Anil V Rao. Direct trajectory optimization and costate estimation via an orthogonal collocation method. *Journal of Guidance, Control, and Dynamics*, 29(6):1435–1440, 2006.

[5] Krishna Sarkar, P Kar, Arijit Mukherjee, and Radhakant Padhi. Range extension of an air-to-air engagement by offline trajectory optimization. In *AIAA Atmospheric Flight Mechanics Conference*, page 6339, 2011.

[6] Prem Kumar, Abhijit Bhattacharya, and Radhakant Padhi. Minimum drag optimal guidance with final flight path angle constraint against re-entry targets. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 1320, 2018.

[7] CF Lin and LL Tsai. Analytical solution of optimal trajectory-shaping guidance. *Journal of Guidance, Control, and Dynamics*, 10(1):60–66, 1987.

[8] Shih-Ming Yang. Analysis of optimal midcourse guidance law. *IEEE Transactions on Aerospace and Electronic Systems*, 32(1):419–425, 1996.

[9] Xinfu Liu, Zuojun Shen, and Ping Lu. Entry trajectory optimization by second-order cone programming. *Journal of Guidance, Control, and Dynamics*, 39(2):227–241, 2016.

[10] Guo Jie Tang Shengjing, Wang Xiao. Trajectory optimization and guidance for hypersonic gliding vehicles based on hp pseudospectral convex programming. *Tactical Missile Technology*, pages 66–75, 2020.

[11] Yang Shi and Zhenbo Wang. A deep learning-based approach to real-time trajectory optimization for hypersonic vehicles. In *AIAA Scitech 2020 Forum*, page 0023, 2020.

[12] Hyo-Sang Shin, Shaoming He, and Antonios Tsourdos. Computational flight control: A domain-knowledge-aided deep reinforcement learning approach. *arXiv preprint arXiv:1908.06884*, 2019.

[13] Roberto Furfaro, Andrea Scorsoglio, Richard Linares, and Mauro Massari. Adaptive generalized zem-zev feedback guidance for planetary landing via a deep reinforcement learning approach. *Acta Astronautica*, 171:156–171, 2020.

[14] Dario Izzo and Ekin Öztürk. Real-time guidance for low-thrust transfers using deep neural networks. *Journal of Guidance, Control, and Dynamics*, 44(2):315–327, 2021.

[15] United States. National Oceanic, Atmospheric Administration, and United States. Air Force. *US standard atmosphere, 1976*, volume 76. National Oceanic and Atmospheric Administration, 1976.

[16] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[17] Joseph F Horn, Eric M Schmidt, Brian R Geiger, and Mark P DeAngelo. Neural network-based trajectory optimization for unmanned aerial vehicles. *Journal of Guidance, Control, and Dynamics*, 35(2):548–562, 2012.

[18] Alexis Duburcq, Yann Chevaleyre, Nicolas Bredeche, and Guilhem Boéris. Online trajectory planning through combined trajectory optimization and function approximation: Application to the exoskeleton atalante. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3756–3762. IEEE, 2020.

[19] Feng Gao, SJ Tang, Jiao Shi, and J Guo. A bias proportional navigation guidance law based on terminal impact angle constraint. *Transactions of Beijing Institute of Technology*, 34(3):277–282, 2014.

[20] Hyo-Sang Shin, Hangju Cho, and Antonios Tsourdos. Time-to-go estimation using guidance command history. *IFAC Proceedings Volumes*, 44(1):5531–5536, 2011.

[21] Ick-Ho Whang and Won-Sang Ra. Time-to-go estimation filter for anti-ship missile application. In *2008 SICE Annual Conference*, pages 247–250. IEEE, 2008.

[22] Y Zhang and G Ma. A biased png law with impact time constraint for antiship missiles. In *2012 Chinese guidance, navigation and control conference*, pages 874–878, 2012.

[23] Quoc V Le, Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, and Andrew Y Ng. On optimization methods for deep learning. In *ICML*, 2011.

[24] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[25] Siti Noorain Mohmad Yousoff, Amirah Baharin, and Afnizanfaizal Abdullah. A review on optimization algorithm for deep learning method in bioinformatics field. In *2016 IEEE EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pages 707–711. IEEE, 2016.

[26] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[27] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[28] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014.