Neural Network-Accelerated Trajectory Optimization for Launch Vehicle Landing

1st Zhipeng Shen
Department of Aeronautical and Aviation Engineering
The Hong Kong Polytechnic University
Hong Kong, China
zhipeng.shen@connect.polyu.hk

2nd Shiyu Zhou
Department of Biomedical Engineering
City University of Hong Kong
Hong Kong, China
shiyuzhou9-c@my.cityu.edu.hk

3rd Jianglong Yu
School of Automation Science and Electrical Engineering
Beihang University
Beijing, China
sdjxyjl@buaa.edu.cn

4th Hailong Huang
Department of Aeronautical and Aviation Engineering
The Hong Kong Polytechnic University
Hong Kong, China
hailong.huang@polyu.edu.hk

2023 9th International Conference on Control Science and Systems Engineering (ICCSSE) | 979-8-3503-3905-5/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICCSSE59359.2023.10245228

***Abstract*—This paper presents a novel trajectory optimization method for the 6-degrees-of-freedom powered landing problem in aerospace guidance and control. The method combines machine learning and convex optimization to achieve real-time performance. Specifically, we formulate the powered landing problem as an optimal control problem and transform it into a convex optimization problem. To enhance the state-of-the-art sequential convex programming (SCP) algorithm, we use a deep neural network as an initial trajectory generator to provide a satisfactory initial guess for the SCP algorithm. Simulation results show that the proposed method achieves precise guidance of the vehicle to the landing site. Monte Carlo tests demonstrate that it can save an average of 40.8% of the computation time compared to the SCP method. Therefore, the proposed scheme is suitable for real-time applications in the aerospace industry.**

***Index Terms*—Trajectory optimization, sequential convex programming, neural networks, real-time computing, launch vehicle landing**

I. Introduction

Convex optimization-based guidance methods have become an increasingly prevalent and effective approach in the field of aerospace guidance and control [1]–[5]. By converting optimal control problems into convex optimization problems via convexification, state-of-the-art convex optimization solvers can efficiently solve them. Nonetheless, when dealing with optimal control problems featuring general nonconvexities, such as the intricate 6-degrees-of-freedom (6-DoF) dynamics, researchers frequently utilize sequential convex programming (SCP) methods [2]–[4]. The SCP method tackles the original nonconvex optimal control problem by linearizing it and resolving a series of locally convex approximations. Recent research has yielded theoretical guarantees for the locally optimal solutions obtained by SCP methods, along with assured convergence properties [6], [7]. This technique has been successfully employed in numerous aerospace guidance applications, including atmospheric reentry [1], missile guidance [8], rocket launch [9], and optimal landing [4], [10]. Furthermore, the SCP method is suitable for various nonlinear systems, as its Taylor expansion-based linearization can manage different nonlinearities. However, the SCP method's performance is heavily reliant on the quality of the initial reference trajectory. To address this issue, this paper introduces an initial trajectory generator using a deep neural network (DNN) to enhance the initial reference trajectory's quality and bolster the SCP method's performance.

The optimal landing problem represents a fundamental optimal control problem with applications in Mars exploration missions and reusable rocket missions [4], [10]. Recently, researchers have explored more general optimal landing problems that incorporate 6-DoF dynamics [2]–[4], [11]. In [2], the authors proposed an SCP algorithm for a generalized 6-DoF free-final-time powered descent guidance problem while considering state-triggered constraints. In [3], an SCP algorithm was presented for the 6-DoF powered descent guidance problem with dual quaternion-based dynamics. To further enhance the computational efficiency of SCP methods, this paper introduces a data-driven SCP approach that combines SCP with deep neural networks (DNNs).

The proposed data-driven SCP method leverages the power of DNNs to provide high-quality initial reference trajectories. By training the DNN with a diverse set of optimal trajectories obtained from various initial conditions and constraints, the method can generate accurate initial guesses, reducing the number of iterations required for SCP convergence. This approach not only improves the computational performance of SCP methods in 6-DoF powered landing problems but also demonstrates potential applicability to a wide range of aerospace guidance and control problems. Additionally, the integration of DNNs into the SCP framework paves the way for further exploration of machine learning techniques in the field of aerospace guidance and control, potentially leading to more sophisticated and efficient solutions.

The remainder of this paper is organized as follows. Sec. II

shows the formulation of the 6-DoF landing problem. Sec. III introduces the SCP algorithm. Sec. IV gives the details of the proposed initial trajectory generator, along with the proposed SCP algorithm. Sec. V presents the test results of the proposed guidance approach. Sec. VI concludes the whole paper.

II. Problem Statement

In this section, the free-final-time 6-DoF powered landing guidance problem is formulated as a nonconvex optimal control problem, while considering the aerodynamic effects and the multiple constraints.

A. Notation

In this paper, we use the following notations: the vector dot product is denoted by $\cdot$, the vector cross product is represented by $\times$, and the Euclidean norm is denoted by $|\cdot|$. Time is denoted by $t \in \mathbb{R}$, with the initial time t_0 defined as the time at which the guidance problem begins and the terminal time t_f defined as the time at which the vehicle reaches the terminal conditions. Subscripts $\mathcal{I}$ and $\mathcal{B}$ represent parameters expressed in the inertial frame $\mathcal{F}_\mathcal{I}$ and the body-fixed frame $\mathcal{F}_\mathcal{B}$, respectively.

B. Dynamics

In this paper, we make the following simplifying assumptions for the powered landing problem: since most powered landing maneuvers occur at velocities far below orbital speeds and the initial landing position is typically only a few kilometers away from the landing site, we neglect the effects of planetary rotation and assume a uniform gravitational field. We also do not consider higher-order phenomena such as fuel slosh and elastic structural modes [2]–[4]. As a rigid body, the vehicle has a constant center of mass and moment of inertia. Additionally, we assume that the ambient atmosphere has constant density and pressure, and we do not account for wind effects. Moreover, we assume that the center of mass is fixed to $\mathcal{F}_\mathcal{B}$. The attitude dynamics in this paper are established based on quaternions, using the scalar-first convention.

The mass-depletion dynamics are given by

$$\dot{m}(t) = -\alpha \|\boldsymbol{T}_\mathcal{B}(t)\| - \beta, \tag{1}$$

where $\alpha \triangleq 1/(I_{\mathrm{sp}} g_0)$ and $\beta \triangleq \alpha P_{\mathrm{atm}} S_{\mathrm{ne}}$.

The translational dynamics are given as follows

$$\dot{\boldsymbol{r}}_\mathcal{I}(t) = \boldsymbol{v}_\mathcal{I}(t), \tag{2}$$

$$\dot{\boldsymbol{v}}_\mathcal{I}(t) = \frac{1}{m(t)}(C_{\mathcal{BI}}(t)\boldsymbol{T}_\mathcal{B}(t) + \boldsymbol{A}_\mathcal{I}(t)) + \boldsymbol{g}_\mathcal{I}, \tag{3}$$

where $C_{\mathcal{BI}}(t) \triangleq C_{\mathcal{IB}}^T(t) = C_{\mathcal{IB}}\left(\boldsymbol{q}_{\mathcal{IB}}^*(t)\right)$, and $\boldsymbol{q}_{\mathcal{IB}}^*(t)$ is the conjugate of $\boldsymbol{q}_{\mathcal{IB}}(t)$. The aerodynamic force $\boldsymbol{A}_\mathcal{I}(t)$ is modeled as follows

$$\boldsymbol{A}_\mathcal{I}(t) = -\frac{1}{2}\rho \|\boldsymbol{v}_\mathcal{I}(t)\| S_A C_A \boldsymbol{v}_\mathcal{I}(t), \tag{4}$$

where C_A is a diagonal matrix for most vehicles that are approximately axisymmetric. The attitude dynamics are given by

$$\dot{\boldsymbol{q}}_{\mathcal{IB}}(t) = \frac{1}{2}\boldsymbol{\Omega}\left(\boldsymbol{\omega}_\mathcal{B}(t)\right)\boldsymbol{q}_{\mathcal{IB}}(t), \tag{5}$$

$$J_\mathcal{B}\dot{\boldsymbol{\omega}}_\mathcal{B}(t) = \boldsymbol{M}_\mathcal{B}(t) - \boldsymbol{\omega}_\mathcal{B}(t) \times J_\mathcal{B}\boldsymbol{\omega}_\mathcal{B}(t), \tag{6}$$

where $\boldsymbol{\Omega}(\cdot)$ is a skew-symmetric matrix defined for the quaternion kinematics (5).

C. Nonconvex Optimal Control Problem

We conclude this section by completing the statement of the nonconvex optimal control problem. The objective function and boundary conditions of the optimal control problem can be designed according to the scenario and mission requirements. In this work, we mainly focus on the minimum-fuel problem, which is equivalent to maximizing the terminal mass. The initial conditions can be given as

$$m(t_0) = m_0, \boldsymbol{r}_\mathcal{I}(t_0) = \boldsymbol{r}_{\mathcal{I}0}, \boldsymbol{v}_\mathcal{I}(t_0) = \boldsymbol{v}_{\mathcal{I}0}, \tag{7}$$

$$\boldsymbol{q}_{\mathcal{IB}}(t_0) = \boldsymbol{q}_0, \boldsymbol{\omega}_\mathcal{B}(t_0) = \boldsymbol{\omega}_{\mathcal{B}0}, \tag{8}$$

where m_0, $\boldsymbol{r}_{\mathcal{I}0}$, $\boldsymbol{v}_{\mathcal{I}0}$, $\boldsymbol{q}_0$, $\boldsymbol{\omega}_{\mathcal{B}0}$ are the prescribed mass, position, velocity, quaternion, and angular velocity at the initial time, respectively. At the terminal time, the goal is to land the vehicle at the landing site steadily and safely. The terminal conditions are given by

$$\boldsymbol{r}_\mathcal{I}(t_f) = \boldsymbol{0}, \boldsymbol{v}_\mathcal{I}(t_f) = \boldsymbol{0}, \boldsymbol{q}_{\mathcal{IB}}(t_f) = \boldsymbol{q}_i, \boldsymbol{\omega}_\mathcal{B}(t_f) = \boldsymbol{0}. \tag{9}$$

The state constraints and control constraints can be found in [2].

III. Sequential Convex Programming

This section introduces an SCP algorithm to solve optimal landing problem. In Sec. III-A, optimal control is converted into a discrete-time convex optimization subproblem. In Sec. III-B, the SCP algorithm is presented, which iteratively solves a sequence of subproblems to get a converged solution.

A. Convex Formulation

1. Normalization

The dynamics equation in the original problem is a nonconvex factor. Thus, it should be converted into a convex formulation. The continuous-time dynamics can be represented as

$$\dot{\boldsymbol{x}}(t) = f(\boldsymbol{x}(t), \boldsymbol{u}(t)), \quad \forall t \in [t_0, t_f], \tag{10}$$

where $\boldsymbol{x}(t) \in \mathbb{R}^{n_x}$ and $\boldsymbol{u}(t) \in \mathbb{R}^{n_u}$ denote the state and control vectors, respectively, and $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ denotes the continuous-time nonconvex dynamics. To begin, the scaled time $\tau \in [0, 1]$ is defined to equivalently convert the original problem into a free-final-time problem. By applying the chain rule, the dynamics can be rewritten as

$$\boldsymbol{x}'(\tau) = t_f f(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau)) \triangleq F(\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), t_f). \tag{11}$$

2. Linearization and Discretization

To formulate a convex problem, the equality constraint functions must be affine. Hence, Eq. (11) is linearized as follows

$$\boldsymbol{x}'(\tau) \approx A(\tau)\boldsymbol{x}(\tau) + B(\tau)\boldsymbol{u}(\tau) + \boldsymbol{s}(\tau)t_f + \boldsymbol{c}(\tau), \tag{12}$$

$$A(\tau) \triangleq \left.\frac{\partial F}{\partial \boldsymbol{x}}\right|_{\tilde{z}(\tau)}, B(\tau) \triangleq \left.\frac{\partial F}{\partial \boldsymbol{u}}\right|_{\tilde{z}(\tau)}, \boldsymbol{s}(\tau) \triangleq \left.\frac{\partial F}{\partial t_f}\right|_{\tilde{z}(\tau)}, \quad (13)$$

$$\boldsymbol{c}(\tau) \triangleq -A(\tau)\tilde{\boldsymbol{x}}(\tau) - B(\tau)\tilde{\boldsymbol{u}}(\tau), \quad (14)$$

where $\tilde{\boldsymbol{z}}(\tau) \triangleq \begin{bmatrix} \tilde{t}_f & \tilde{\boldsymbol{x}}^T(\tau) & \tilde{\boldsymbol{u}}^T(\tau) \end{bmatrix}^T$ is the reference trajectory.

The linearization of the thrust magnitude lower bound constraint can be found in [4], and the discretization method is as same as [2].

The trust-region constraint is introduced to ensure that the optimized trajectory lies in a region where the linearization is valid. The trust-region constraint is expressed in a quadratic form as

$$\|\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k\|^2 + \|\boldsymbol{u}_k - \tilde{\boldsymbol{u}}_k\|^2 \leq \sigma_k, \quad (15)$$

where $k \in \{1, 2, \ldots, N\}$, and σ_k is a trust-region radius. To allow the optimization process to select the trust region, the cost function is augmented with the trust-region radii $\boldsymbol{\sigma} \in \mathbb{R}^N_+$ as

$$J_{\mathrm{tr}} = w_{\mathrm{tr}} \|\boldsymbol{\sigma}\|_1, \quad (16)$$

where $w_{\mathrm{tr}} \in \mathbb{R}_{++}$ is a weighing term, and $\|\cdot\|_1$ denotes one-norm.

The linearization-based method also suffers from the artificial infeasibility issue [2]. To solve this issue, a virtual control term $\boldsymbol{\mu}_k \in \mathbb{R}^{n_x}$ is added into the discrete-time linearized dynamics. The virtual control should be only used for constraint satisfaction. Thus, the cost function is augmented with a large weighing term $w_{\mathrm{vc}} \in \mathbb{R}_{++}$ as

$$J_{\mathrm{vc}} = w_{\mathrm{vc}} \|V\|_1, \quad (17)$$

where $V \triangleq \begin{bmatrix} \boldsymbol{\mu}_1 & \boldsymbol{\mu}_2 & \ldots & \boldsymbol{\mu}_{N-1} \end{bmatrix} \in \mathbb{R}^{n_x \times (N-1)}$.

B. Sequential Convex Programming Algorithm

As mentioned in Sec. III-A, the SCP algorithm is an iterative algorithm. In each iteration, the algorithm uses a solver for convex optimization to efficiently solve the convex subproblem [2]–[5], [9]. In this subsection, we will introduce how the algorithm is initialized and terminated.

1. Initialization

In this paper, an initial trajectory generator is proposed in Sec. IV. The proposed method is based on DNN and uses the pre-computed results to train the neural network so that it can give a satisfactory initial trajectory guess. The details of the proposed initial trajectory generator will be introduced in Sec. IV.

2. Convergence Criteria

The iteration process terminates when convergence criteria are met. The convergence criteria are given as

$$J_{\mathrm{tr}}(\boldsymbol{\sigma}) \leq \epsilon_{\mathrm{tr}}, \quad (18a)$$

$$J_{\mathrm{vc}}(V) \leq \epsilon_{\mathrm{vc}}, \quad (18b)$$

where $\epsilon_{\mathrm{tr}} \in \mathbb{R}_{++}$ and $\epsilon_{\mathrm{vc}} \in \mathbb{R}_{++}$ are the convergence tolerances, which can be user-specified. The convergence criterion (18a) measures the difference between the solutions of two consecutive iterations. Additionally, the criterion (18b) guarantees that the solution meets the dynamics.

IV. Initial Trajectory Generator

In this section, the initial trajectory generator based on the DNN is proposed, which can significantly reduce the computation time of the SCP algorithm. In general, we use straight-line initialization to initialize the SCP algorithm. The algorithm is used to solve guidance problems with various initial conditions. The obtained solutions are collected to construct a data set. The data set is used to train the DNN-based generator so that the generator can give a satisfactory initial guess trajectory. In Sec. IV-A, the construction of the data set is introduced. The structure of the DNN-based generator is presented in Sec. IV-B.

A. Data Set

1. Data Set Construction

Disturbing the initial state of the vehicle with the uniformly distributed stochastic parameter $\boldsymbol{\xi} \in \mathbb{R}^{n_x}$, the various guidance problems with various initial states can be obtained. The disturbed initial state can be expressed as

$$\boldsymbol{x}_1^n = \boldsymbol{x}_1 + \boldsymbol{\xi}^n, \quad \forall n \in \{1, 2, \ldots, N_{\mathrm{tra}}\}, \quad (19)$$

where the superscript n denotes the n-th perturbation, N_{tra} is the number of trajectories in the data set, and $\boldsymbol{x}_1 \triangleq \begin{bmatrix} m_0 & \boldsymbol{r}_{\mathcal{I}0}^T & \boldsymbol{v}_{\mathcal{I}0}^T & \boldsymbol{q}_0^T & \boldsymbol{\omega}_{\mathcal{B}0}^T \end{bmatrix}^T$. Each perturbation generates a guidance problem. The SCP algorithm presented in Sec. III is used to solve the various guidance problems. As a result, the data set contains N_{tra} optimized trajectories.

B. Trajectory Generator Structure

Motivated by the sequence model prediction in natural language processing [12], the initial trajectory is modeled as a sequence model. A frame of the trajectory is defined as $(\boldsymbol{x}_k, \boldsymbol{u}_k)$, for $k \in \{1, 2, \ldots, N\}$. The DNN is used to recurrently predict each frame of the trajectory until an N-frame trajectory is generated. Given the last frame, the DNN can predict the next frame of the trajectory according to

$$(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1}) = Net_{\mathrm{tra}}(\boldsymbol{x}_k, \boldsymbol{u}_k), \ \forall k \in \{1, 2, \ldots, N-1\}. \quad (20)$$

The generated trajectory can be used as the initial trajectory of SCP.

The structure of the DNN is important for the performance of the generator. In this paper, a variety of structures are tested in Sec. V. According to the test results, a five-hidden-layers structure with 256 units per layer is finally selected. Further, the Rectified Linear Unit (ReLU) is adopted as the activation function, and the DNN has N_L layers in total.

V. Results

In this section, the test results are presented to demonstrate the effectiveness and performance of the proposed guidance method. In Sec. V-A, the constructed data set is presented. In Sec. V-B, the details of the training process are introduced. In Sec. V-C, the performance of the proposed guidance method is illustrated via simulations. The Monte Carlo analysis is

performed to test the performance of the algorithm. The proposed algorithm is also compared with the state-of-the-art SCP algorithm to highlight the improvement.

A. Data Set

The data set is constructed via the SCP method mentioned in Sec. III and IV-A. The weight parameters in Eqs. (16) and (17) are selected as $w_{\text{tr}} = 0.5$ and $w_{\text{vc}} = 1 \times 10^5$, respectively. The convergence tolerances ϵ_{tr} and ϵ_{vc} in Eqs. (18a) and (18b) are both selected as 5×10^{-4}. The initial conditions and parameters of the landing problem are listed in Table I. The vehicle's moment of inertia is $\text{diag}([4 \times 10^6, 4 \times 10^6, 1 \times 10^5])$ kg $\cdot$ m^2.

TABLE I
INITIAL CONDITIONS AND PARAMETERS

Parameter	Value	Parameter	Value
m_0 (kg)	30000	$T_{\max}$ (N)	800000
$\boldsymbol{r}_{\mathcal{I}0}$ (m)	$[0\ 0\ 1500]^T$	$T_{\min}$ (N)	320000
$\boldsymbol{v}_{\mathcal{I}0}$ (m/s)	$[0\ 0\ -80]^T$	$\tilde{t}_f$ (s)	18
$\boldsymbol{q}_0$	$\boldsymbol{q}_i$	$\boldsymbol{g}_{\mathcal{I}}$ (m/s^2)	$[0\ 0\ -9.81]^T$
$\boldsymbol{\omega}_{\mathcal{B}0}$ (deg/s)	$[0\ 0\ 0]^T$	S_A (m^2)	10
$\boldsymbol{T}_{\mathcal{B}0}$ (N)	$[0\ 0\ T_{\min}]^T$	C_A	$\text{diag}([3, 3, 1])$
$m_{\min}$ (kg)	22000	ρ (kg/m^3)	1.225
N	30	I_{sp} (s)	282
$\omega_{\max}$ (deg/s)	30	P_{atm} (Pa)	0
γ_c (deg)	20	S_{ne} (m^2)	0
$\theta_{\max}$ (deg)	80	$\boldsymbol{d}_{T,\mathcal{B}}$ (m)	$[0\ 0\ -14]^T$
$\vartheta_{\max}$ (deg)	20	$\boldsymbol{d}_{A,\mathcal{B}}$ (m)	$[0\ 0\ 2]^T$

As discussed in Sec. IV-A, a uniformly distributed term $\boldsymbol{\xi} \triangleq \left[\begin{array}{ccccc} \boldsymbol{\xi}_m{}^T & \boldsymbol{\xi}_r{}^T & \boldsymbol{\xi}_v{}^T & \boldsymbol{\xi}_q{}^T & \boldsymbol{\xi}_\omega{}^T \end{array} \right]^T \in \mathbb{R}^{n_x}$ is added to the initial state of the vehicle to generate various guidance problems. The ranges of the random parameters are listed in Table II.

TABLE II
RANGES OF RANDOM PARAMETERS

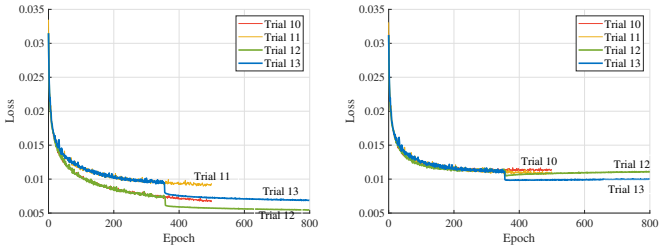
Parameter	Range
$\boldsymbol{\xi}_m$ (kg)	0
$\boldsymbol{\xi}_r$ (m)	$[[-500, 500]\ \ [-500, 500]\ \ 0]^T$
$\boldsymbol{\xi}_v$ (m/s)	$[[-40, 40]\ \ [-40, 40]\ \ [-20, 20]]^T$
$\boldsymbol{\xi}_q$	$Euler2Quater^*([[-30, 30]\ \ [-30, 30]\ \ 0]^T) - \boldsymbol{q}_i$
$\boldsymbol{\xi}_\omega$ (deg/s)	$[[-20, 20]\ \ [-20, 20]\ \ 0]^T$

* Function that converts Euler angles (deg) to quaternions.

The data set can be obtained by solving various guidance problems. In this paper, the data set contains 48333 trajectories, 45000 of which are used for training and 3333 for testing.

B. Training Results

To determine the optimal hyperparameters for our training process, we evaluated the performance of different hyperparameters. We conducted a total of 13 trials, and their details are listed in Table III. The loss functions during the training process are shown in Figs. 1a-1b. Notably, trials 12 and 13 utilized learning rate decay to improve convergence. Trials 1-11 were trained for 500 epochs, while trials 12 and 13 were trained for 800 epochs to test the effect of learning rate decay.



(a) Training loss (b) Test loss

Fig. 1. Training process of trials 10-13. 500 epochs of training are carried out in trials 10 and 11, and 800 epochs of training are carried out in trials 12 and 13.

TABLE III
INFORMATION OF TRIALS

Trial	Learning Rate	Unit[1]	Layer[2]	Batch[3]	WD[4]
1	1×10^{-3}	128	5	128	–
2	1×10^{-4}	128	5	128	–
3	1×10^{-5}	128	5	128	–
4	1×10^{-4}	64	5	128	–
5	1×10^{-4}	256	5	128	–
6	1×10^{-4}	256	5	256	–
7	1×10^{-4}	256	5	64	–
8	1×10^{-4}	256	5	32	–
9	1×10^{-4}	256	4	128	–
10	1×10^{-4}	256	6	128	–
11	1×10^{-4}	256	6	128	1×10^{-5}
12	1×10^{-4}	256	6	128	–
13	1×10^{-4}	256	6	128	1×10^{-5}

[1] Number of units in each layer of the DNN.
[2] Number of DNN layers.
[3] Batch size.
[4] Constant λ in weight decay.

Among the trials we conducted, trial 10 exhibited the lowest training and test losses. Due to limited space, we omitted the results of trials 1-9. From Figs. 1a and 1b, we observed that trial 12, which used the same hyperparameters as trial 10, achieved a lower training loss but had a test loss similar to that of trial 10, indicating overfitting. To address this, we employed weight decay. Trial 13 achieved the best test loss and acceptable training loss, suggesting that the DNN has satisfactory generalization ability. Therefore, in Sec. V-C, we will use the DNN obtained from trial 13 as the initial trajectory generator.

C. Guidance Performance

We validated the proposed guidance method through two missions with different initial conditions. The solutions for these two missions are shown in Fig. 2. As demonstrated, the proposed algorithm successfully guided the vehicle to the landing site, satisfying the approach cone constraint, and ensuring the minimum mass constraint. The computation time for Mission 1 and Mission 2 was 0.62s and 0.94s, respectively, meeting the requirement of solving the trajectory online in less than 1s [2]–[4]. Specifically, in Mission 1, the DNN took 0.11s to generate the initial trajectory, and the SCP algorithm took 0.51s to solve the guidance problem after two iterations.

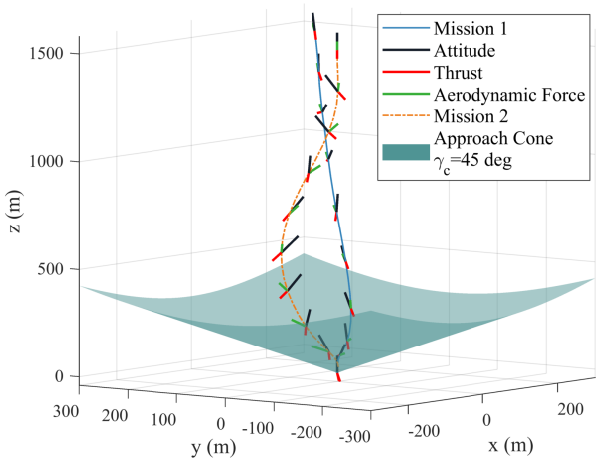


Fig. 2. Solutions of the missions.

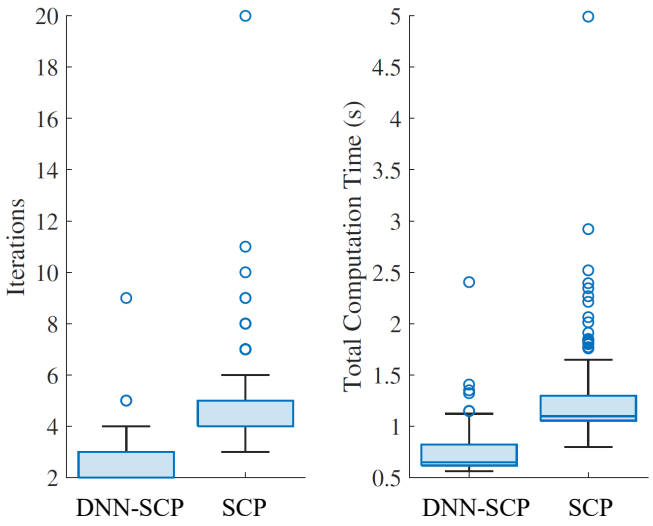


Fig. 3. Iterations and total computation time.

In Mission 2, the DNN took 0.17s to generate the initial trajectory, and the SCP algorithm took 0.77 s to solve the guidance problem after three iterations.

To further analyze the performance of the proposed method, we conducted Monte Carlo analysis using random parameter selections similar to those used to generate the dataset in Sec. V-A. We compared the proposed method with the state-of-the-art SCP method and conducted 1000 Monte Carlo simulations. Fig. 3 shows the results, with the proposed method labeled DNN-SCP. As demonstrated, the proposed method's real-time performance was better than that of the SCP method, with 991 tests out of 1000 meeting the real-time requirements. Moreover, the median computation time of the proposed method was almost half that of the SCP method, with an average computation time of 0.7035 s, reduced by 40.8% compared to the SCP method.

VI. Conclusion

In this paper, we propose a real-time trajectory optimization method that leverages data-driven initialization. Our method combines the SCP algorithm with neural networks, considering aerodynamic effects within the framework of 6-DoF dynamics. Rather than using neural networks as the controller, we use a DNN as an initial trajectory generator to generate the required initial trajectory for the SCP algorithm. This significantly reduces computation time compared to state-of-the-art SCP methods, saving 40.8% computation time. Moreover, 99.1% of the test cases take less than 1 s, making the proposed method more suitable for online real-time applications. In extensive Monte Carlo tests, our proposed approach outperforms state-of-the-art SCP methods.

References

[1] X. Liu, Z. Shen, and P. Lu, "Entry trajectory optimization by second-order cone programming," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 227–241, 2016.

[2] M. Szmuk, T. P. Reynolds, and B. Açıkmeşe, "Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 8, pp. 1399–1413, 2020.

[3] T. P. Reynolds, M. Szmuk, D. Malyuta, M. Mesbahi, B. Açıkmeşe, and J. M. Carson, "Dual quaternion-based powered descent guidance with state-triggered constraints," *Journal of Guidance, Control, and Dynamics*, vol. 43, no. 9, pp. 1584–1599, 2020.

[4] M. Szmuk and B. Açıkmeşe, "Successive convexification for 6-DoF mars rocket powered landing with free-final-time," in *2018 AIAA Guidance, Navigation, and Control Conference*, 2018, pp. 1–14.

[5] Z. Shen, J. Yu, X. Dong, Y. Hua, and Z. Ren, "Penetration trajectory optimization for the hypersonic gliding vehicle encountering two interceptors," *Aerospace Science and Technology*, vol. 121, p. 107363, 2022.

[6] Y. Mao, D. Dueri, M. Szmuk, and B. Açıkmeşe, "Successive convexification of non-convex optimal control problems with state constraints," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 4063–4069, 2017.

[7] Y. Mao, M. Szmuk, X. Xu, and B. Açıkmeşe, "Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems," *arXiv preprint arXiv:1804.06539*, 2019.

[8] X. Liu, Z. Shen, and P. Lu, "Exact convex relaxation for optimal flight of aerodynamically controlled missiles," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1881–1892, 2016.

[9] B. Benedikter, A. Zavoli, G. Colasurdo, S. Pizzurro, and E. Cavallini, "Convex approach to three-dimensional launch vehicle ascent trajectory optimization," *Journal of Guidance, Control, and Dynamics*, vol. 44, no. 6, pp. 1116–1131, 2021.

[10] X. Liu, "Fuel-optimal rocket landing with aerodynamic controls," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 1, pp. 65–77, 2019.

[11] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açıkmeşe, "Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently," *IEEE Control Systems Magazine*, vol. 42, no. 5, pp. 40–113, 2022.

[12] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *arXiv preprint arXiv:2106.11342*, 2021.