Real-time computational powered landing guidance using convex optimization and neural networks
Zhipeng Shen, Shiyu Zhou, Jianglong Yu

Abstract—Computational guidance is an emerging and accelerating trend in aerospace guidance and control. Combining machine learning and convex optimization, this paper presents a real-time computational guidance method for the 6-degrees-of-freedom powered landing guidance problem. The powered landing guidance problem is formulated as an optimal control problem, which is then transformed into a convex optimization problem. Instead of brutally using the neural networks as the controller, we use neural networks to improve the state-of-the-art sequential convex programming (SCP) algorithm. Based on the deep neural network, an initial trajectory generator is designed to provide a satisfactory initial guess for the SCP algorithm. Benefiting from designing the initial trajectory generator as a sequence model predictor, the proposed data-driven SCP architecture is capable of improving the performance of any state-of-the-art SCP algorithm in various applications, not just powered landing guidance. The simulation results show that the proposed method can precisely guide the vehicle to the landing site. Moreover, through Monte Carlo tests, the proposed method can averagely save 40.8% of the computation time compared with the SCP method, while ensuring higher terminal states accuracy. The proposed computational guidance scheme is suitable for real-time applications.

Index Terms—Computational guidance, sequential convex programming, neural networks, real time, powered landing.

I. INTRODUCTION

In recent years, computational guidance is an emerging trend in aerospace guidance and control [1]–[6]. Thanks to the improved computing power and advanced algorithms, computational guidance methods can leverage the increased onboard computational capability to complete the intensive online computation [1], [5]. However, instead of simply using computing power to numerically solve the guidance problem via brute force, computational guidance achieves online computation through upfront investment in problem formulation, modeling, and analysis [3], [5]. Motivated by the computational guidance philosophy [1]–[6], this paper introduces a data-driven method to further enhance the upfront investment and achieve more efficient online computation.

As a popular and powerful way, the guidance method based on convex optimization is a significant computational guidance method [4], [7]–[18]. The convex optimization-based guidance method transforms the optimal control problem into a convex optimization problem by convexification and then utilizes the state-of-the-art convex optimization solver to efficiently solve the convex optimization problem. Lossless convexification theory bridges the gap between nonconvex optimal control problems and efficient convex optimization methods [4], [7]–[9]. The results of lossless convexification have been extended to optimal control problems with nonconvex control constraints [7], linear state constraints [8], and annular control constraints [4]. For optimal control problems with more general nonconvexities, such as the complex 6-degrees-of-freedom (DoF) dynamics, researchers turn to sequential convex programming (SCP) methods [13]–[15]. Through linearization, SCP solves the original nonconvex optimal control problem by solving a sequence of locally convex approximations. The current researches provide theoretical guarantees for locally optimal solutions obtained by SCP methods with guaranteed convergence properties [19]–[21]. This method has been extended to many applications of aerospace guidance, such as atmospheric reentry [12], missile guidance [10], rocket launch [11], and

Fig. 1. The landing scenario.

This work was supported by the National Natural Science Foundation of China under Grants 62103016, 61922008, the Foundation Strengthening Program Technology Field Fund under Grant 2021-JCJQ-JJ-1237, China National Postdoctoral Program for Innovative Talents under Grant BX202000034, and the China Postdoctoral Science Foundation under Grant 2020M680297.
Z. Shen is with the Department of Aeronautical and Aviation Engineering, the Hong Kong Polytechnic University, Hong Kong, China (E-mail: zhipeng.shen@connect.polyu.hk).
S. Zhou is with the Department of Biomedical Engineering, City University of Hong Kong, Hong Kong, China.
J. Yu is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China (E-mail: sdjxyjl@buaa.edu.cn).

This work has been submitted to the IEEE for possible publication. Copyright may be transferred without notice, after which this version may no longer be accessible.
optimal landing [2], [3], [15]. Since the linearization based on Taylor expansion can make the SCP method suitable for different nonlinear systems, the SCP method can be applied in many different applications. However, the linearization in the SCP algorithm also makes its performance significantly dependent on the initial reference trajectory. One of the contributions of this paper is to design an initial trajectory generator through the deep neural network (DNN) to improve the quality of the initial reference trajectory and further improve the performance of the SCP method.

The optimal landing problem is one of the optimal control problems that has been widely concerned because it is widely used in Mars exploration missions and reusable rocket missions [2]–[4], [15]. Since lossless convexification can transform the nonconvex 3-DoF landing problem into a more easily solved and equivalent convex optimization problem, lossless convexification was widely used in optimal landing tasks and has been verified in a series of flight experiments [4], [9], [22]–[24]. More recently, researchers begin to consider more general optimal landing problems with 6-DoF dynamics [13]–[15], [25]. In [13], an SCP algorithm was presented for a generalized 6-DoF free-final-time powered descent guidance problem while considering the state-triggered constraints. An SCP algorithm was also introduced for the 6-DoF powered descent guidance problem with dual quaternion-based dynamics in [14]. To achieve more efficient computational guidance methods, this paper presents a data-driven SCP approach by combining SCP with DNN. The proposed approach can significantly reduce the number of iterations required for the convergence of SCP methods and further improve the computational performance.

Recently, researchers are also interested in new techniques for onboard algorithms leveraging advancements in machine learning [25]–[36]. Many researchers have combined traditional methods with neural networks to improve computational performance [25]–[33]. Shi et al. in [26] proposed a method based on deep learning to realize real-time online trajectory planning of hypersonic vehicles. Chen et al. used DNN to generate the optimal asteroid landing trajectory [27]. Cheng et al. used DNN to improve the predictor-corrector entry guidance law of lifting hypersonic vehicles [28]. The optimal planetary orbit transfer was studied by combining physics-informed neural networks and Pontryagin’s minimum principle in [29]. You et al. in [30] studied the fuel-optimal powered decent guidance by incorporating Pontryagin’s minimum principle with neural network. In [31], the free final-time fuel-optimal powered landing guidance problem was studied by combining the lossless convex optimization and the DNN, where the DNN was used to predict the final time. Chai et al. developed a real-time optimal command generator for reentry problems by using the DNN in [32] and further proposed a DNN-based method to achieve the integration of trajectory planning and attitude control for 6-DOF reentry flight [25]. Since neural networks are usually considered black boxes, it is not a guaranteed method to directly use them as control command generators. Instead of using neural networks to output control commands, this paper combines DNN with SCP methods. In this paper, a DNN-based trajectory generator is proposed to generate initial guesses required by the SCP method. This method not only improves the performance of the SCP method through DNN but also provides the same theoretical guarantee as the SCP method. Similarly, a warm-start method was proposed in [37], which used neural networks to predict the initial trajectory for the SCP method. However, the neural network in [37] was used to predict a trajectory parameterized as a polynomial for each state and control. In this case, it is difficult to determine the order of each polynomial in different applications. In [38], a neural network policy was updated through the trajectories obtained from a one-step SCP iteration and a feedback control law. The control law in [38] needs to be carefully designed according to the specific application. Inspired by the sequence model in natural language processing [39], [40], we design the initial trajectory generator as a sequence model predictor. Therefore, the initial trajectory generator does not need to be specially designed for the problem. Since the trajectories of different problems can be considered as the same sequence model with different dimensions on each discrete point, this makes the proposed architecture applicable to SCP algorithms in various applications.

The primary contributions of this paper are highlighted as follows.

1) A real-time computational guidance approach is proposed for the powered landing guidance. Compared with the state-of-the-art SCP algorithm [13]–[15], the proposed algorithm saves 40.8% time and has better accuracy. The proposed method can meet real-time requirements better.

2) Instead of brutally using the neural networks to output optimal solutions [25], [32], [33], a data-driven SCP algorithm is presented by combining SCP and DNN, such that the proposed method can improve the performance while providing the same theoretical guarantee as the SCP algorithm.

3) The initial trajectory generator is designed as a sequence model predictor, which makes the proposed computational guidance architecture a more general approach that can be applied to improve the computational performance of any SCP algorithm in various applications. Meanwhile, the related works [37], [38] requires special design according to the specific applications.

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 6DoF powered landing guidance problem is formulated as a nonconvex optimal control problem, while considering the aerodynamic effects and the multiple constraints. The landing scenario is shown in Fig. 1.
A. Notation

We use · to denote the vector dot product, × to represent the vector cross product, and ‖ ‖ to denote the Euclidean norm. We denote time as \( t \in \mathbb{R} \). The initial time \( t_0 \) is defined as the time at which the guidance problem begins, and the terminal time \( t_f \) is defined as the time at which the vehicle reaches the terminal conditions. Subscripts \( I \) and \( B \) represent parameters expressed in the inertial frame \( F_I \) and body-fixed frame \( F_B \), respectively. The frame \( F_I \) is an east-north-up coordinate frame, and its origin is fixed at the landing site. The origin of the frame \( F_B \) coincides with the vehicle’s center of mass, the \( z \) axis of \( F_B \) points along the vertical axis of the vehicle, the \( x \) axis of \( F_B \) points out the side of the vehicle, and the \( y \) axis of \( F_B \) completes the right-handed system. Some notations associated with the guidance problem are defined below.

Notations:

- \( g_0 \): Standard gravitational acceleration
- \( \rho \): Ambient atmospheric density
- \( m(t) \in \mathbb{R}_{++} \): Vehicle mass
- \( r_I(t) \in \mathbb{R}^3 \): Inertial position of the vehicle
- \( v_I(t) \in \mathbb{R}^3 \): Inertial velocity of the vehicle
- \( g_I \): Gravitational acceleration
- \( T_I(t) \in \mathbb{R}^3 \): Thrust vector
- \( A_I(t) \in \mathbb{R}^3 \): Aerodynamic force
- \( S_A \in \mathbb{R}_{++} \): Reference area of the vehicle
- \( C_A \in \mathbb{R}_{++}^{3} \): Aerodynamic coefficient matrix
- \( \otimes \): Quaternion multiplication
- \( q_i \): Identity quaternion
- \( q_{IB}(t) \in S^3 \subset \mathbb{R}^4 \): Unit quaternion that parameterizes the transformation from \( F_I \) to \( F_B \)
- \( C_{IB}(t) \in \text{SO}(3) \): Direction cosine matrix corresponding to \( q_{IB}(t) \)
- \( \omega_B(t) \in \mathbb{R}^3 \): Angular velocity of the vehicle in \( F_B \)
- \( J_B \in \mathbb{S}_{++}^3 \): Moment of inertia of the vehicle
- \( d_{TB} \): Position of the engine gimbal pivot point in \( F_B \)
- \( d_{A,B} \): Position of the center of pressure in \( F_B \)
- \( M_B(t) \in \mathbb{R}^3 \): Torque acting on the vehicle, \( M_B(t) \triangleq d_{TB}(t) \times T_B(t) + d_{A,B} \times A_B(t) \)
- \( I_{sp} \): Vacuum specific impulse of the engine
- \( P_{atm} \): Ambient atmospheric pressure
- \( S_{ne} \): Nozzle exit area of the engine

B. Dynamics

Since most powered landing maneuvers are far below the orbital velocities, and the initial position of the powered landing is only a few kilometers away from the landing site. The effects of planetary rotation are neglected, and we assume the gravitational field is uniform. Moreover, higher-order phenomena such as fuel slosh and elastic structural modes are not considered. As a rigid body, the vehicle has a constant center of mass and moment of inertia. Additionally, the density and pressure of the ambient atmosphere are assumed to be constant, and the effects of winds are not taken into account. We also assume that the center of mass is fixed to \( F_B \). In this paper, the attitude dynamics are established based on quaternion, and the scalar-first quaternion convention is used.

The mass-depletion dynamics are given by

\[
\dot{m}(t) = -\alpha \|T_B(t)\| - \beta \tag{1}
\]

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

The translational dynamics are given as follows

\[
\dot{r}_I(t) = v_I(t) \tag{2}
\]

\[
\dot{v}_I(t) = \frac{1}{m(t)}(C_{IB}(t)T_B(t) + A_I(t)) + g_I \tag{3}
\]

where \( C_{IB}(t) \triangleq C^T_{IB}(t)C_{IB}(t) = C_{IB}(q_{IB}(t)) \) is the conjugate of \( q_{IB}(t) \). The aerodynamic force \( A_I(t) \) is modeled as follows

\[
A_I(t) = -\frac{1}{2} \rho \|v_I(t)\| S_A C_A v_I(t) \tag{4}
\]

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

\[
\dot{q}_{IB}(t) = \frac{1}{2} \Omega(q_{IB}(t)) q_{IB}(t) \tag{5}
\]

\[
J_B \omega_B(t) = M_B(t) - \omega_B(t) \times J_B \omega_B(t) \tag{6}
\]

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

C. State Constraints

Since the fuel on the vehicle is limited, the mass of the vehicle should be greater than a minimum mass \( m_{\text{min}} \in \mathbb{R}_{++} \). The mass is constrained by enforcing

\[
m_{\text{min}} \leq m(t) \tag{7}
\]

Next, the trajectory of the vehicle is constrained in an approach cone to ensure that the vehicle has sufficient elevation on the planetary surface for landing. The approach cone constraint can be expressed as

\[
\|H_t r_I(t)\| \leq \cot \gamma_c e \cdot r_I(t) \tag{8}
\]

\[
e \triangleq \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T, H_t \triangleq \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}
\]

The approach cone is centered at the landing point, and \( \gamma_c \in [0^\circ, 90^\circ] \) is the angle between the cone and the horizontal.

The vehicle’s tile angle, which is the angle between the \( z \) axes of \( F_B \) and \( F_I \), should also be constrained by enforcing

\[
2\|H_t q_{TB}(t)\|^2 \leq 1 - \cos \theta_{\text{max}} \tag{10}
\]

\[
H_t \triangleq \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{11}
\]

where \( \theta_{\text{max}} \in (0^\circ, 90^\circ] \) is the maximum tilt angle.

The allowable angular velocity is constrained by enforcing

\[
\|\omega_B(t)\|_\infty \leq \omega_{\text{max}} \tag{12}
\]

where \( \omega_{\text{max}} \in \mathbb{R}_{++} \) is a maximum angular velocity, and \( \|\cdot\|_\infty \) denotes the \( \infty \)-norm.
D. Control Constraints

We assume that a single gimbaled engine is equipped on the vehicle. The engine can rotate symmetrically, but is constrained to a maximum gimbal angle $\vartheta_{\text{max}} \in (0^\circ, 90^\circ)$. Hence, the first control constraint is given by

$$
\|H \cdot T_B(t)\| \leq \tan \vartheta_{\text{max}} \mathbf{e} \cdot T_B(t)
$$

(13)

The thrust magnitude of the engine can vary between a fixed minimum value and a fixed maximum value. The second control constraint is given as

$$
T_{\text{min}} \leq \|T_B(t)\| \leq T_{\text{max}}
$$

(14)

where $[T_{\text{min}}, T_{\text{max}}] \subset \mathbb{R}^{+}$ is the permitted thrust interval. We assume that the rocket engine starts at the initial time $t_0$, and only the minimum thrust magnitude with a zero-degree gimbal angle can be used when the engine starts. Then, an equality constraint can be given as

$$
T_B(t_0) = T_{B0} = \begin{bmatrix} 0 & 0 & T_{\text{min}} \end{bmatrix}^T
$$

(15)

E. 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

$$
m(t_0) = m_0, \mathbf{r}_I(t_0) = \mathbf{r}_{I0}, \mathbf{v}_I(t_0) = \mathbf{v}_{I0}
$$

(16)

$$
\mathbf{q}_{\text{IB}}(t_0) = \mathbf{q}_0, \mathbf{\omega}_B(t_0) = \mathbf{\omega}_{B0}
$$

(17)

where $m_0, \mathbf{r}_{I0}, \mathbf{v}_{I0}, \mathbf{q}_0, \mathbf{\omega}_{B0}$ 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

$$
\mathbf{r}_I(t_f) = 0, \mathbf{v}_I(t_f) = 0, \mathbf{q}_{\text{IB}}(t_f) = \mathbf{q}_t, \mathbf{\omega}_B(t_f) = 0
$$

(18)

The nonconvex optimal control problem is summarized as Problem 1.

III. SEQUENTIAL CONVEX PROGRAMMING

This section introduces a SCP algorithm to solve Problem 1. In Sec. III-A, Problem 1 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 Problem 1 is a nonconvex factor. Thus, it should be converted into a convex formulation. The continuous-time dynamics can be represented as

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

(19)

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

$$
\dot{x}'(\tau) \triangleq \frac{d}{d\tau} x(\tau) = \frac{dt}{d\tau} f(x(\tau), u(\tau))
$$

(21)

We assume that $t_0 = 0$. Thus, one has $t = t_f \tau$. Further, the Eq. (21) can be written as

$$
x'(\tau) = t_f f(x(\tau), u(\tau)) \triangleq F(x(\tau), u(\tau), t_f)
$$

(22)

2. Linearization

To formulate a convex problem, the equality constraint functions must be affine [41]. Hence, Eq. (22) is linearized as follows

$$
x'(\tau) \approx A(\tau)x(\tau) + B(\tau)u(\tau) + s(\tau)t_f + e(\tau)
$$

(23)

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

(24)

$$
c(\tau) \triangleq -A(\tau)\ddot{x}(\tau) - B(\tau)\ddot{u}(\tau)
$$

(25)

where $\ddot{x}(\tau) \triangleq \left[ I_f \ddot{x}^T(\tau) \ddot{u}^T(\tau) \right]^T$ is the reference trajectory.

The other nonconvex factor in Problem 1 is the thrust magnitude lower bound constraint, which can be given as

$$
h_u(u(\tau)) \triangleq T_{\text{min}} - \|u(\tau)\| \leq 0
$$

(26)

Further, Eq. (26) can be approximated using a first-order Taylor series:

$$
h_u(u(\tau)) \approx h_u(\ddot{u}(\tau)) + \frac{dh_u(u(\tau))}{du(\tau)} \ddot{u}(\tau)
$$

(27)

= $T_{\text{min}} - \ddot{u}(\tau) / \|u(\tau)\| = T_{\text{min}} - H_u(\tau)u(\tau)

3. Discretization

The optimal control problem is a continuous-time problem, which leads to infinite dimensional space. By introducing $N$ evenly spaced temporal nodes, the problem is discretized into $N - 1$ subintervals. The control command is approximated by the affine interpolation given as

$$
\bar{u}(\tau) = \tilde{u}(\tau) + \dot{\bar{u}}(\tau), \quad \forall \tau \in [\tau_k, \tau_{k+1}]
$$

(28)

$$
\tilde{u}(\tau) = \frac{\tau_{k+1} - \tau}{\tau_{k+1} - \tau_k} - \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k}, \quad \ddot{\bar{u}}(\tau) = \frac{\tau_{k+1} - \tau}{\tau_{k+1} - \tau_k} - \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k}
$$

(29)

where $k \in \{1, 2, \ldots, N - 1\}$, $\tau_k = (k - 1)/(N - 1)$, and $u_k \triangleq u(\tau_k)$. Then, the dynamics Eq. (23) can be shown as follows for each time interval.

$$
x'(\tau) = A(\tau)x(\tau) + \tilde{\eta}(\tau)B(\tau)u_k + \dot{\eta}(\tau)B(\tau)u_{k+1} + s(\tau)t_f + e(\tau), \quad \forall \tau \in [\tau_k, \tau_{k+1}]
$$

(30)
**Problem 1:** Find the control commands profile $T_B(t), \forall t \in [t_0, t_f]$ to solve the nonconvex optimal control problem.

| Cost function | $\min_{t_f, T_B(t)} -m(t_f)$ | Convex |
|---------------|-------------------------------|--------|

| Boundary conditions | $m(t_0) = m_0, \dot{r}_X(t_0) = r_{x0}, \dot{v}_X(t_0) = v_{x0}$ | See Eqs. (15-18) Convex |
| $q_{zB}(t_0) = q_{0}, \dot{\omega}_B(t_0) = \omega_{B0}, T_B(t_0) = T_{B0}$ | |
| $\dot{r}_X(t_f) = 0, \dot{v}_X(t_f) = 0, q_{zB}(t_f) = q_{f}, \omega_B(t_f) = 0$ | |

| Dynamics | $\dot{m}(t) = -\alpha |T_B(t)| - \beta$ | See Eq. (1) Nonconvex |
| $\dot{r}_X(t) = v_{xX}(t)$ | See Eq. (2) Nonconvex |
| $\dot{v}_X(t) = \frac{1}{m(t)} \{ C_{zB}(t) T_B(t) + A_X(t) \} + g_I$ | See Eq. (3) Nonconvex |
| $\dot{q}_{zB}(t) = \frac{1}{2} \Omega (\omega_B(t)) q_{zB}(t)$ | See Eq. (5) Nonconvex |
| $J_B \omega_B(t) = M_B(t) - \omega_B(t) \times J_B \omega_B(t)$ | See Eq. (6) Nonconvex |

| State constraints | $m_{min} \leq m(t)$ | See Inequality (7) Convex |
| $\| H_r x(t) \| \leq \cot \gamma_c e \cdot r_X(t)$ | See Inequality (8) Convex |
| $2 \| H_t q_{zB}(t) \|^2 \leq 1 - \cos \theta_{max}$ | See Inequality (10) Convex |
| $\| \omega_B(t) \|_{\infty} \leq \omega_{max}$ | See Inequality (12) Convex |

| Control constraints | $\| H_r T_B(t) \| \leq \tan \theta_{max} e \cdot T_B(t)$ | See Inequality (13) Convex |
| $T_{min} \leq \| T_B(t) \| \leq T_{max}$ | See Inequality (14) Convex |

The state transition matrix $\Phi(\tau, \tau_k), \forall \tau \in [\tau_k, \tau_{k+1}]$ associated with Eq. (30) with zero input is given by

$$\Phi(\tau, \tau_k) = I + \int_{\tau_k}^{\tau} A(\zeta) \Phi(\zeta, \tau_k) \, d\zeta$$  (31)

where $I$ is an identity matrix with appropriate dimension. The discrete-time state vector can be denoted as $x_k \triangleq x(\tau_k)$. By applying the inverse and transitive properties of the state transition matrix, the discrete-time linearized dynamics can be presented as

$$x_{k+1} = A_k x_k + \dot{B}_k u_k + B_k u_{k+1} + s_k t_f + c_k$$  (32)

where

$$A_k \triangleq \Phi(\tau_{k+1}, \tau_k)$$  (33a)

$$\dot{B}_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_k) \hat{v}(\tau) B(\tau) \, d\tau$$  (33b)

$$B_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_k) \eta(\tau) B(\tau) \, d\tau$$  (33c)

$$s_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_k) s(\tau) \, d\tau$$  (33d)

$$c_k \triangleq A_k \int_{\tau_k}^{\tau_{k+1}} \Phi^{-1}(\tau, \tau_k) c(\tau) \, d\tau$$  (33e)

The propagation method described in Eqs. (31-33e) is analogous to a multiple shooting method, which can improve the convergence performance of the algorithm [13], [14]. Note that Eq. (31) and Eqs. (33a-33e) are computed only based on $\hat{v}(\tau)$, the propagation method can be computed simultaneously in implementation, which is computationally efficient.

4. Trust Region and Virtual Control

The SCP algorithm is an iterative algorithm based on linearization. Thus, 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

$$\| x_k - \bar{x}_k \|^2 + \| u_k - \bar{u}_k \|^2 \leq \sigma_k$$  (34)

where $k \in \{1, 2, \ldots, N\}$, and $\sigma_k$ 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 $\sigma \in \mathbb{R}^+_N$ as

$$J_{tr} = w_{tr} \| \sigma \|_1$$  (35)

where $w_{tr} \in \mathbb{R}^+$ is a weighting term, and $\| \cdot \|_1$ denotes one-norm. The linearization-based method also suffers from the artificial infeasibility issue [13]. The multiple constraints, e.g. state constraints, linearized constraints, and the trust-region constraint, may cause infeasibility. For example, if the problem is linearized based on an unrealistic reference trajectory, the linearized constraints and the trust-region constraint cannot be simultaneously satisfied. To solve this issue, a virtual control term $\mu_k \in \mathbb{R}^n$ is added into the discrete-time linearized dynamics (32) according to

$$x_{k+1} = A_k x_k + \dot{B}_k u_k + B_k u_{k+1} + s_k t_f + c_k + \mu_k$$  (36)

The virtual control should be only used for constraint satisfaction. Thus, the cost function is augmented with a large weighting term $w_{vc} \in \mathbb{R}^+$ as

$$J_{vc} = w_{vc} \| V \|_1$$  (37)

where $V \triangleq [ \mu_1 \mu_2 \ldots \mu_{N-1} ] \in \mathbb{R}^{n_x \times (N-1)}$.

The convex optimization subproblem is summarized as Problem 2.

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 subproblem Problem 2 [11], [13]-[16]. In this subsection, we will introduce how the algorithm is initialized and terminated.

1. Initialization

The initialization approach generates an initial reference trajectory to start the iterative process. The trajectory generated by the initialization approach is only used in the first iteration, and the solution of the previous iteration is used as the reference trajectory in the next iteration to iteratively formulate the convex optimization subproblem.
There are already some proposed initialization approaches, e.g., the straight-line initialization and the 3-DoF initialization [13], [14]. Although the SCP algorithm can handle a wide range of initialization guesses, the quality of guesses greatly affects the convergence performance. Poor guesses may lead to more iterations, which can increase computation time. In this paper, an initial trajectory generator is proposed in Sec. IV. The proposed method is based on DNN and uses the precomputed results to train the neural network so that it can give a satisfactory initial trajectory guess. By giving a good initial trajectory guess, the convergence of the iterative algorithm can be accelerated to reduce the computation time. The details of the proposed initial trajectory generator will be introduced in Sec. IV.

2. Nondimensionalization

A numerical solver may encounter issues associated with machine precision and sensitivity due to the different magnitudes of variables in the optimization problem. A nondimensionalization is performed by scaling the optimization variables via

\[ m = \frac{m}{U_M}, \quad \bar{r}_T = \frac{r_T}{U_L}, \quad \bar{v}_T = \frac{v_T}{U_L}, \quad \bar{v}_I = \frac{v_I}{U_T/U_L} \]  

(38a)

\[ \bar{T}_B = \frac{U_T^2}{U_M U_L} \]  

(38b)

where \( U_M \equiv m_0, \) \( U_L \equiv ||r_{ZO}||, \) and \( U_T \equiv 1 \) s are the scaling units.

3. Convergence Criteria

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

\[ J_{\mu}(\sigma) \leq \epsilon_{\mu} \]  

(39a)

\[ J_{vc}(V) \leq \epsilon_{vc} \]  

(39b)

where \( \epsilon_{\mu} \in \mathbb{R}_{++} \) and \( \epsilon_{vc} \in \mathbb{R}_{++} \) are the convergence tolerances, which can be user-specified. The convergence criterion (39a) measures the difference between the solutions of two consecutive iterations. Additionally, the criterion (39b) guarantees that the solution meets the dynamics.

Comparing the maximum difference in the solutions between two consecutive iterations is another approach. The algorithm terminates when the difference is less than a tolerance. The criterion can be given as

\[ \max_{\{1,2,\ldots,N\}} \| (x_k - \bar{x}_k) \|_{\infty} < \epsilon_x \]  

(40)

where \( \epsilon_x \in \mathbb{R}_{++} \) is the tolerance. Using different numerical solvers, even solving the same problem, can result in different values for criteria (39a-39b) [14]. The criterion (40) is uniformly functional across different solvers. Criteria (39) are more conducive to obtaining the dynamically feasible solution, but require more iterations to converge, while criterion (40) is more suitable for online application. In Sec. V, the criteria (39) are used to construct the data set, and the criterion (40) is used to test guidance performance.

In this paper, we combine SCP algorithms with DNN to improve the performance of SCP algorithms, instead of using neural networks as controllers brutality. The proposed algorithm is detailed as Algorithm 1 in Sec. IV together with the proposed initial trajectory generator.

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 the 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. The overview of the proposed scheme is shown in Fig. 2. 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. In Sec. IV-C, the details on how to train the DNN-based generator are given.

A. Data Set

1. Straight-line Initialization
The data set is constructed by solving various guidance problems. Thus, a baseline initialization approach is introduced. The straight-line initialization method is presented as

\[ \tilde{x}_k = \left( \frac{N - k}{N - 1} \right) \tilde{x}_{ini} + \left( \frac{k - 1}{N - 1} \right) \tilde{x}_{fin} \] (41a)

\[ \tilde{u}_k = \frac{T_{\max} - T_{\min}}{2} e \] (41b)

where \( \tilde{x}_{ini} \triangleq \left[ m_0 \ r_{f_0}^T \ v_{f_0}^T \ q_i^T \ \omega_{B0}^T \right]^T \), \( \tilde{x}_{fin} \triangleq \left[ m_{\min} \ 0^T \ 0^T \ q_i^T \ 0^T \right]^T \), and \( k \in \{1, 2, \ldots, N\} \).

2. **Data Set Construction**

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

\[ x_1^n = x_1 + \xi^n, \ \forall n \in \{1, 2, \ldots, N_{tra}\} \] (42)

where the superscript \( n \) denotes the \( n \)-th perturbation, \( N_{tra} \) is the number of trajectories in the data set, and \( x_1 \triangleq \left[ m_0 \ r_{f_0}^T \ v_{f_0}^T \ q_i^T \ \omega_{B0}^T \right]^T \). 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. The constructed data set is shown in Sec. V.

### B. Trajectory Generator Structure

Motivated by the sequence model prediction in natural language processing [39], [40], the initial trajectory is modeled as a sequence model as shown in Fig. 3. A frame of the trajectory is defined as \( (x_k, u_k) \), 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 next frame of the trajectory according to

\[(x_{k+1}, u_{k+1}) = Net_{tra}(x_k, u_k), \ \forall k \in \{1, 2, \ldots, N - 1\} \] (43)

The generated trajectory can be used as the initial trajectory of SCP. In Sec. V, one can see that this initialization method can significantly reduce the computation time. The SCP algorithm together with the initial trajectory generator is summarized in Algorithm 1.

It is important to remark that since the initial trajectory generator is a sequence model predictor, it is independent of the dynamics of the vehicle itself, i.e., the generator only recurrently predicts the state and control on each time stamp. Thus, the generator can be applied to other problems by only adjusting the input and output dimensions of the network. In this paper, although we have only studied the powered landing guidance problem, which contains 6-DoF dynamics with nonlinearity, multiple nonlinear constraints, etc., the generator can be applied to any SCP-based guidance method to reduce the computation time.

The structure of the DNN is very important for the performance of the generator. A DNN consists of an input layer, several hidden layers, and an output layer. In this paper, a variety of structures are tested in Sec. V. According to the test results, a five-hidden-layers structure of 256 units per layer is finally selected. Further, the Rectified Linear Unit (ReLU) is adopted as the activation function, which can be expressed as

\[ \kappa_j(x) = \max(x, 0), \ \forall j \in \{1, 2, \ldots, N_L - 1\} \] (44)

where \( \kappa_j \) means the activation function of the \( j \)-th layer, and the DNN has \( N_L \) layers in total.

In the original input vector, there is a wide variety of data types, e.g., mass, velocities, and angle rates. Since there is no prior knowledge about which features will be more relevant, it should be avoided that the weights assigned to some features

---

**Algorithm 1** SCP with Initial Trajectory Generator.

**INITIAL TRAJECTORY GENERATOR** \( (\tilde{x}_1, \tilde{u}_1) \)

for \( k \in \{1, 2, \ldots, N - 1\} \) do

compute \( (\tilde{x}_{k+1}, \tilde{u}_{k+1}) = Net_{tra}(\tilde{x}_k, \tilde{u}_k) \)

end for

give initial guess \( \tilde{t}_f \)

return \( (\tilde{t}_f, \tilde{x}, \tilde{u}) \)

**SEQUENTIAL CONVEX PROGRAMMING** \( (\tilde{t}_f, \tilde{x}, \tilde{u}) \)

while not converged do

use \( (\tilde{t}_f, \tilde{x}, \tilde{u}) \) to construct Problem 2

solve Problem 2 to get \( (t_f, x, u, V, \sigma) \)

if criterion is satisfied according to (39a-40) converged

end if

\( (\tilde{t}_f, \tilde{x}, \tilde{u}) \leftarrow (t_f, x, u) \)

end while

return \( (t_f, x, u) \)
are greater than that of other features. Thus, data normalization is required to achieve a satisfactory result in the training process [26], [39]. The input data $x$ is normalized as

$$x \leftarrow \frac{x - E(x)}{De(x)}$$

(45)

where $E(x)$ and $De(x)$ denote the mean and standard deviation of $x$. The input features are rescaled to zero mean and unit variance. The input of the DNN is a normalized trajectory frame, and the output is the normalized trajectory frame in the next discrete point. Hence, the input and output of the DNN are both a $17 \times 1$ vector.

C. Trajectory Generator Training

In the training process, the mean squared error (MSE) is utilized as the loss function, which can be expressed as

$$L = \frac{1}{N_b} \sum_{i=1}^{N_b} (\hat{y}_i - y^*)^2$$

(46)

where $N_b$ is the number of training samples, $\hat{y}$ is the final output of the DNN, and $y^*$ is the corresponding target output. Further, the Adam method is employed as the optimizer [42]. Feeding the DNN with the training data set, the optimizer can update the DNN in each epoch to reduce the loss function.

The weight decay technique is also introduced in the training process. The weight decay technique is a regularization technique, which mitigates overfitting by adding a penalty term to the loss function (46) [39]. The loss function is now given by

$$L_{\text{new}} = L + \lambda \|w_{\text{net}}\|^2$$

(47)

where $w_{\text{net}}$ is the weight vector, and $\lambda$ is the non-negative regularization constant, which characterizes the tradeoff.

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. The test results are carried out on a desktop computer with an Intel Core i7-12700F processor. The test results are obtained using Python implementation. The CVXPY and MOSEK solver are used to solve the convex optimization problem [43]–[45].

A. Data Set

The data set is constructed via the SCP method mentioned in Sec. III and IV-A. The weight parameters in Eqs. (35) and (37) are selected as $w_L = 0.5$ and $w_{vc} = 1 \times 10^5$, respectively. The convergence tolerances $\epsilon_{tr}$ and $\epsilon_{vc}$ in Eqs. (39a) and (39b) are both selected as $5 \times 10^{-4}$. 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⋅m².
As discussed in Sec. IV-A, a uniformly distributed term \( \xi \sim [\xi_m^T \xi_v^T \xi_i^T \xi_q^T \xi_o^T]^T \in \mathbb{R}^{n_x} \) 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.

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. The randomly sampled trajectories with various initial conditions are shown in Fig. 4.

### Table I

**INITIAL CONDITIONS AND PARAMETERS**

| Parameter | Value | Parameter | Value |
|-----------|-------|-----------|-------|
| \( m_0 \) (kg) | 300000 | \( T_{\text{max}} \) (N) | 800000 |
| \( r_{\text{f,0}} \) (m/s) | 0–1500 | \( T_{\text{min}} \) (N) | 320000 |
| \( v_{\text{f,0}} \) (m/s) | 0–50 | \( t_f \) (s) | 18 |
| \( \omega_0 \) (deg/s) | 0–5 | \( g_T \) (m/s²) | 0–9.81 |
| \( T_B0 \) (N) | 0–225 | \( C_A \) | diag(3,3,1) |
| \( m_{\text{min}} \) (kg) | 22000 | \( \rho \) (kg/m³) | 1.225 |
| \( \omega_{\text{max}} \) (deg/s) | 30 | \( \delta_{\text{tip}} \) (s) | 282 |
| \( \gamma_{\text{max}} \) (deg) | 20 | \( S_m \) (m²) | 0 |
| \( \theta_{\text{max}} \) (deg) | 80 | \( d_{T,\text{g}} \) (m) | 0–14 |
| \( \nu_{\text{max}} \) (deg) | 20 | \( d_{A,\text{g}} \) (m) | 0–2 |

### Table II

**RANGES OF RANDOM PARAMETERS**

| Parameter | Range |
|-----------|-------|
| \( \xi_m \) (kg) | 0 |
| \( \xi_v \) (m) | [−500,500] [−500,500] 0 | |
| \( \xi_i \) (m/s) | [−40,40] [−40,40] [−20,20] |
| \( \xi_q \) | Euler2Quater*([−30,30] [−30,30] 0) \( - q_i \) |
| \( \xi_o \) (deg/s) | [−20,20] [−20,20] 0 |

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

### Table III

**INFORMATION OF TRIALS**

| Trial | Learning Rate | Unit* | Layer* | Batch | WD * |
|-------|---------------|-------|--------|-------|------|
| 1     | \( 1 \times 10^{-3} \) | 128   | 5      | 128   | –    |
| 3     | \( 1 \times 10^{-4} \) | 128   | 5      | 128   | –    |
| 4     | \( 1 \times 10^{-5} \) | 64    | 5      | 128   | –    |
| 5     | \( 1 \times 10^{-6} \) | 256   | 5      | 128   | –    |
| 6     | \( 1 \times 10^{-6} \) | 256   | 5      | 128   | –    |
| 7     | \( 1 \times 10^{-7} \) | 256   | 5      | 128   | –    |
| 8     | \( 1 \times 10^{-8} \) | 256   | 5      | 32    | –    |
| 9     | \( 1 \times 10^{-9} \) | 256   | 4      | 128   | –    |
| 10    | \( 1 \times 10^{-10} \) | 256   | 6      | 128   | –    |
| 11    | \( 1 \times 10^{-11} \) | 256   | 6      | 128   | \( 1 \times 10^{-5} \) |
| 12    | \( 1 \times 10^{-12} \) | 256   | 6      | 128   | \( 1 \times 10^{-5} \) |

1. Number of units in each layer of the DNN.
2. Number of DNN layers.
3. Batch size.
4. Constant A in weight decay (see Eq. (47)).

As shown in Fig. 5, the training loss and test loss of trial 10 are minimal throughout the test. Further, learning rate decay is used to make the loss converge better. In Fig. 6, although trial 12, which uses the same hyperparameters as trial 10, achieves lower training loss, the test loss of trial 12 is close to that of trial 10. Hence, there is overfitting. Then, the weight decay

---

Fig. 5. Training process of trials 1-10. 500 epochs of training are carried out. (a) Training loss. (b) Test loss.

B. **Training Results**

In the training process, to determine the better hyperparameters, the training performance of different hyperparameters is tested. A total of 13 trials are tested, and the information on these trials is listed in Table III. The loss functions during training are shown in Figs. 5 and 6. It should be pointed out that in trials 12 and 13, learning rate decay is used to make the training process converges better. When the training loss is not reduced in 25 epochs of training, the learning rate is reduced ten times. The minimum learning rate is limited to \( 1 \times 10^{-6} \). In trails 1-11, 500 epochs of training are carried out. But 800 epochs of training are carried out in trials 12 and 13 to test the effect of learning rate decay.
mentioned in Sec. IV-C is used in the training process to alleviate the overfitting phenomenon. Trial 13 achieves the best test loss and acceptable training loss results, which indicates that the DNN may have satisfactory generalization ability. Therefore, in Sec. V-C, the DNN obtained from trial 13 will be used as the initial trajectory generator.

C. Guidance Performance

The proposed guidance method is validated by two missions with different initial conditions. The initial conditions and parameters are listed in Table IV, and the parameters not listed are the same as those in Table I. The convergence tolerance \( \epsilon_x \) in Eq. (40) is selected as \( 1 \times 10^{-2} \). The solutions of these two missions are shown in Fig. 7. As displayed in Fig. 7a, the proposed algorithm can guide the vehicle to the landing site. And the approach cone constraint can be satisfied, which can be observed in Fig. 7b. In addition, the details of the state constraints and control constraints are shown in Fig. 8. One can ensure that these constraints are all satisfied. And the minimum mass constraint can be ensured by observing that the final mass of Mission 1 is 26403.8 kg, and the final mass of Mission 2 is 26697.4 kg. In Fig. 8b, it should also be noted that the obtained solutions exhibit behavior that is close to bang-bang control, while early works noted that the solution to 3-DoF fuel-optimal powered landing guidance also exhibited bang-bang behavior [46], [47].

\[
\begin{array}{|c|c|c|}
\hline
\text{Parameter} & \text{Mission 1} & \text{Mission 2} \\
\hline
m_0 (kg) & 30000 & 30000 \\
\mathbf{r}_{Z0} (m) & [200 200 1500] & [0 0 1500] \\
\mathbf{v}_{Z0} (m/s) & [-20 -20 -80] & [0 0 -80] \\
\mathbf{q}_0 & \text{Euler2Quater}([[-20 20 0] \text{ ]^T}) & [0 0 0 1]^T \\
\omega_{Z0} (deg/s) & [0 0 0] & [0 0 0] \\
\gamma_c (deg) & 45 & 45 \\
\hat{\gamma}_c (deg) & 30 & 30 \\
\hline
\end{array}
\]

\( ^* \) Function that convert Euler angles (deg) to quaternions.
The computation time of Mission 1 and Mission 2 is 0.62 s and 0.94 s, respectively. In Mission 1, the DNN takes 0.11 s to generate the initial trajectory, and the SCP algorithm takes 0.51 s to solve the guidance problem after 2 iterations. In Mission 2, the DNN takes 0.17 s to generate the initial trajectory, and the SCP algorithm takes 0.77 s to solve the guidance problem after 3 iterations. For real-time landing guidance, it is required to solve the trajectory online in less than 1 s [13]–[15]. In work [13], it was claimed that the propagation step (see Eq. 33) consumes time on the order of 10 ms by using C++ and Eigen matrix library. Thus, the computation time of the propagation step was omitted in [13]. However, all of our work is implemented in Python in this paper, and the computation time of the propagation step is almost the same as that of the solver. To provide more complete test results, the computation time of the propagation step is not omitted in this paper.

To further analyze the performance of the proposed method, the Monte Carlo analysis is used to test it. The random parameter selection for Monte Carlo analysis is the same as that used to generate the data set in Sec. V-A. In Monte Carlo analysis, the obtained thrust solution is used as the open-loop control command. The nonlinear dynamics of the vehicle are propagated using the open-loop control command. The proposed method is compared with the state-of-the-art SCP method. 1000 Monte Carlo simulations have been carried out, and Figs. 9 and 10 show the results. To simplify, the proposed method is labeled DNN-SCP in the Figs. 9 and 10. The error is defined as the 2-norm of each state error, which can be expressed as

$$ err(y) \triangleq \|y(t_f) - y_f\| $$

where $y \in \{r_I, v_I, q_{IB}, \omega_B\}$, and $y_f$ is the terminal constraint value of variable $y$. The purpose of the landing guidance is to successfully land the vehicle on the landing site. Thus, the accuracy of the terminal states is significant. The errors of the terminal states are shown in Fig. 9. According to Fig. 9a, the proposed method performs better than the state-of-the-art SCP method in position accuracy and velocity accuracy.
According to Fig. 9b, the proposed method is slightly better than the SCP method in orientation accuracy. And the angular velocity accuracy performance is almost the same. Even if the SCP method seems barely better than the proposed method in angular velocity accuracy, the angular velocity errors of the two methods are both small enough. The reason why the proposed method can obtain better accuracy is that, although the convergence criteria of the two methods are the same, the trained DNN provides a better initial trajectory, making it easier to obtain a dynamically feasible solution.

Another important performance for the proposed method is its computational performance. To meet the real-time requirements, the computation time needs to be less than 1 s [13]–[15]. According to Fig. 10a, the real-time performance of the proposed method is also better than that of the SCP method. Although a few tests do not meet the real-time requirements, most of the tests can meet the requirements. In this Monte Carlo simulation of 1000 tests, 991 tests meet the real-time requirements. The median computation time is almost half of the SCP method. Specifically, the average computation time of the proposed method is 0.7035 s, and that of the SCP method is 1.1887 s, which is reduced by 40.8%. Because the proposed method has a better initial trajectory provided by the DNN, it can converge faster. The median number of iterations of the proposed method is 2, while that of the SCP method is 4. Even if the DNN needs an average of 0.1523 s to generate the initial trajectory, the computation time of the proposed method is far less than that of the SCP method. As this paper studies the minimum-fuel guidance problem, Fig. 10b shows the final mass of the vehicle. The greater the mass remaining, the more fuel remaining. The proposed method generally consumes slightly more fuel than the SCP method. The trajectory of the proposed method depends largely on the initial trajectory generated by the DNN, so its optimality is also limited by the initial trajectory.

In general, a good initial trajectory based on historical experience can improve the performance of iterative algorithms such as the SCP algorithm. However, it is very inefficient to use historical experience data directly. A better initial trajectory generator can be obtained by training the neural network with historical experience data. The initial trajectory generator can make the algorithm converge faster by giving a satisfactory initial trajectory based on the training data. In addition, the trained neural network does not occupy much memory and has high computational efficiency.

VI. CONCLUSION

This paper proposes a real-time computational guidance method using data-driven initialization. We present a guidance method that combines the SCP method with neural networks in the framework of 6-DoF dynamics considering aerodynamic effects. Instead of brutally using neural networks as the controller, we use a DNN as an initial trajectory generator to generate the initial trajectory required by the SCP method. This makes the proposed method less time-consuming than the state-of-the-art SCP method. And compared with using neural networks as the controller, the SCP algorithm ensures reliability. In extensive Monte Carlo tests, the proposed approach outperforms a state-of-the-art SCP method. The terminal states accuracy of the proposed method is better. And the proposed method consumes less time, saving 40.8% computation time compared with the SCP method. 99.1% of the test cases take less than 1 s, which means that the proposed method is more suitable for online real-time applications. Furthermore, our DNN-based SCP scheme opens up interesting lines of future research. The DNN-based trajectory generator is designed as a sequence model predictor, which makes the proposed architecture can be used to improve the performance of the SCP algorithm in various applications.

REFERENCES

[1] P. Lu, “What is guidance?” Journal of Guidance, Control, and Dynamics, vol. 44, no. 7, pp. 1237–1238, 2021.
[2] J. Wang, N. Cui, and C. Wei, “Optimal rocket landing guidance using convex optimization and model predictive control,” Journal of Guidance, Control, and Dynamics, vol. 42, no. 5, pp. 1078–1092, 2019.
