doi: 10.82124/CEAS-GNC-2026-053 CEAS-GNC-2026-053


# Engineerable Onboard Guidance Based on Successive Convex Programming for Powered Landing

**Jia Zhuo** — **Deputy Chief Engineer**, Beijing Galactic Energy Space Technology Co., Beijing, China. jiazhuo332@gmail.com

**Zihang Song** — **GNC Engineer**, Beijing Galactic Energy Space Technology Co., Beijing, China. jaimeski.song@gmail.com

**Baiqi Liu** — **Chief Engineer**, Beijing Galactic Energy Space Technology Co., Beijing, China. lbq@buaa.edu.cn

*ABSTRACT*

**This paper describes an onboard guidance strategy which can be applied to powered landing phase of reusable launch vehicle. This method addresses the non-convex dynamic problems induced by aerodynamic forces and non-constant specific impulse variations with successive convex programming (SCP). By introducing thrust acceleration as control variable, smooth thrust command is obtained. By means of treatments such as reducing the number of trust region constraints and normalizing variables, this method significantly reduces the computational cost of the algorithm and ensures its applicability to real-time onboard planning computations. Through numerical simulations, we demonstrate the convergence robustness of the algorithm under a wide range of flight time inputs. Tests conducted on the engineering-grade flight control computer demonstrate that the algorithm exhibits low computational complexity, which can meet the requirements of onboard guidance.**

**Keywords:** Successive Convex Programming, Onboard Guidance, Powered Landing, Reusable launch vehicle

## Nomenclature

| | | |
|---|---|---|
| $\theta$, $\psi$ | = | Euler angles (pitch, yaw) |
| $x, y, z$ | = | position in surface fixed coordinate |
| $v_x$, $v_y$, $v_z$ | = | velocity in surface fixed coordinate |
| $t$ | = | time |
| $t_f$ | = | total flight time |
| $a$ | = | acceleration |
| $T$ | = | thrust magnitude |
| $m$ | = | mass |
| $\boldsymbol{F}$ | = | aerodynamic force vector |
| $\boldsymbol{g}$ | = | gravitational acceleration vector |
| N | = | number of discrete segments |
| $I_{sp}$ | = | specific impulse |

# 1 Introduction

In recent years, reusable rockets technology has achieved rapid development. Within the realm of reusable technologies, the powered landing guidance algorithm stands as one of the core technologies enabling the accurate and reliable return of launch vehicles to landing sites [1]. Owing to various deviations and disturbance factors during the rocket's return process, the initial state of the powered descent phase is difficult to pre-estimate and pre-set. Consequently, soft-landing guidance requires real-time trajectory planning and feedforward control.

Extensive studies employ convex optimization-based methods for powered descent trajectory planning. Refs. [2, 3] transform the planetary landing problem into a convex programming problem that can be reliably solved using second-order cone programming (SOCP) through lossless convexification. However, In the powered descent process of an Earth-bound rocket, aerodynamic effects cannot be neglected.

To address the non-convex problems caused by aerodynamic effects and nonlinear dynamics, a series of studies have adopted successive convex programming (SCP) methods to tackle the non-convexity arising from non-linearity. The decoupled thrust in the ground-fixed Cartesian coordinate system is used as the control input [4, 5], which exhibit excellent global convergence characteristics and have low dependence on the initial reference trajectory. Refs. [6, 7] further adopt Euler attitude angular rates, thrust rates, rates of angle of attack and sideslip angle as control variables, enabling convenient constraints on the corresponding control variables and state variables. However, abrupt changes in rates imply infinite acceleration, which is unachievable in practical engineering applications.

Refs. [8, 9] ensure the smoothness and engineering feasibility of trajectories by introducing six-degree-of-freedom (6-DOF) model. The accompanying introduction of quaternion kinematics and attitude dynamics may increase the complexity of the optimization problem. The nonlinear equality constraints such as the unit quaternions condition may cause the successive convex programming (SCP) to fall into a "crawling" phenomenon [10]. To address this issue, the Augmented Convex-Concave Decomposition (ACCD) needs to be further adopted [11, 12], ACCD decomposes nonlinear equality constraints into a cone constraint and a linearized inequality constraint, which heightens the problem’s complexity, introduces more uncertainties for engineering applications. This study proposes a successive convex programming (SCP) algorithm base on 3-DOF model. By adopting thrust acceleration as the control input and introducing an attitude angular rate penalty term, the algorithm can generate smooth thrust and attitude commands while being more concise than the 6-DoF method. In terms of computational efficiency, this study finds that it is not necessary to apply trust region constraints to all design variables; instead, constraining only the highly nonlinear quantities in the dynamic equations can avoid unbounded problems, which reduces the problem dimension while improving the algorithm's convergence speed. Another measure is the nondimensionalization of design variables, which can also significantly reduce the number of iterations in a single convex optimization. These two measures enhance the real-time performance of the algorithm and ensure its engineering applicability.

This paper is organized as follows: In Section 2, the problem considered in this study is formulated as the optimization problem. In Section 3, convexification of the problem formalized in Section 2 is performed for sequential convex optimization. In Section 4, the proposed techniques are applied to typical scenario and the comparison with nonlinear programming results is presented. Finally, in Section6, the conclusion is given.

# 2 Problem Formulation

## 2.1 Coordinate Frame

This study focuses on the final powered descent phase of the rocket's return process. The translation dynamics of the rocket will be expressed in a surface fixed coordinate frame. The frame is showed in Fig.1 Definitions of pitch angle $\theta$ and yaw angle $\psi$ are showed in Fig. 2. Assuming the rocket body is axisymmetric, the roll angle does not affect the directions of thrust, lift, and drag in the ground-fixed coordinate system. It is set to a constant value 0 in this study.

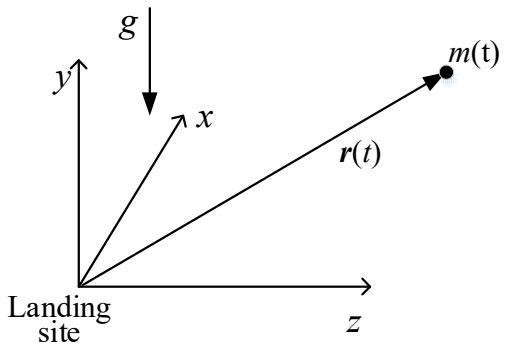


**Fig. 1 Surface fixed coordinate frame**

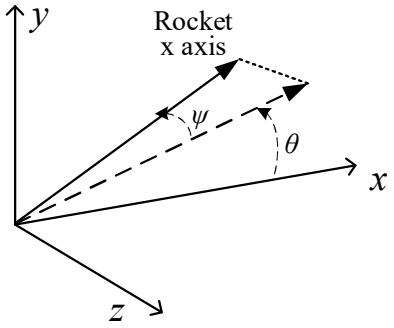


**Fig. 2 Euler angles definition**

## 2.2 Equations of motion

The vehicle is modeled as a point mass subject to 3-DOF translational motion. We denote the vehicle's mass, position, and velocity as $m$(t), $\boldsymbol{r}$(t), and $\boldsymbol{v}$(t), respectively. Since the powered descent phase covers only a short portion of the entire rocket return process, the gravity $g$, and air density $\rho$ are assumed to be constant over the course of the trajectory.

Based on the aerodynamic axisymmetry of the rocket body, the aerodynamic coefficients are considered to be functions of only the Mach number, angle of attack, and sideslip angle. This study uses un-trimmed aerodynamic data. If trimmed aerodynamic data were used, the trajectories obtained would be more accurate. However, this difference does not affect the conclusions of this study. The aerodynamic coefficient data are stored in the on-board computer in the form of a multi-dimensional table.

$$\boldsymbol{C}_{BODY}^{aero} = \begin{bmatrix} C_x(Ma,\ \alpha,\ \beta) \\ C_y(Ma,\ \alpha,\ \beta) \\ C_z(Ma,\ \alpha,\ \beta) \end{bmatrix} \tag{1}$$

The aerodynamic force in body frame can be computed as

$$\begin{bmatrix} F_x \\ F_y \\ F_z \end{bmatrix} = \frac{1}{2}\rho v_a^2 S_r \boldsymbol{C}_{BODY}^{aero} \tag{2}$$

$S$r is the reference area, and $\rho$ denotes atmospheric density, which is assumed to be a constant in this study. This is acceptable when the altitude variation during the powered descent phase is small. The $v_a$ is the speed of the vehicle with respect to the air.

the 3-DOF translational motion of the vehicle is presented below. The $G_{BODY}^{SF}$ in Eq. (5) is the rotation matrix from body fixed frame to surface fixed frame. We assume that the thrust acts only along the axial direction of the rocket body, meaning the effect of nozzle gimbaling on the thrust is neglected. Due to the short flight distance during the powered descent phase, we assume that the gravitational acceleration remains constant throughout this process.

In Eq. (7) the specific impulse of the engine is no longer constant, but a function that changes with the thrust amplitude. As the thrust changes, the combustion condition of the rocket engine deviates from the standard design state, causing the engine's specific impulse to vary with the magnitude of the thrust. This study models the variable specific impulse and incorporates its impact on propellant consumption into trajectory optimization. Furthermore, thrust acceleration and Euler angle rates are introduced into the model as control variables, which allows for constraint of these variables based on the practical capabilities of thrust adjustment and attitude control.

$$\dot{\boldsymbol{r}}(t) = \boldsymbol{v}(t) \tag{3}$$

$$\dot{\boldsymbol{v}}(t) = \boldsymbol{a}(t) \tag{4}$$

$$\boldsymbol{a}(t) = \frac{1}{m(t)} G_{BODY}^{SF} \begin{bmatrix} T(t) + F_x(t) \\ F_y(t) \\ F_z(t) \end{bmatrix} + \begin{bmatrix} g_x \\ g_y \\ g_z \end{bmatrix} \tag{5}$$

$$G_{BODY}^{SF} = \begin{bmatrix} \cos\theta\cos\psi & -\sin\theta & \cos\varphi\sin\psi \\ \sin\theta\cos\psi & \cos\theta & \sin\theta\sin\psi \\ -\sin\psi & 0 & \cos\psi \end{bmatrix} \tag{6}$$

$$\dot{m}(t) = -\frac{T(t)}{I_{sp}(T) g_0} \tag{7}$$

$$\ddot{T}(t) = \mathrm{u}_T \tag{8}$$

$$\dot{\theta}(t) = \mathrm{u}_\theta \tag{9}$$

$$\dot{\psi}(t) = \mathrm{u}_\psi \tag{10}$$

## 2.3 Boundary Conditions

The initial and terminal conditions are given fixed values.

$$\boldsymbol{r}(t_0) = \boldsymbol{r}_0, \boldsymbol{v}(t_0) = \boldsymbol{v}_0, \theta(t_0) = \theta_0, \psi(t_0) = \psi_0, m(t_0) = m_0 \tag{11}$$

$$\boldsymbol{r}(t_f) = \boldsymbol{r}_f, \boldsymbol{v}(t_f) = \boldsymbol{v}_f, \theta(t_f) = \theta_f, \psi(t_f) = \psi_0, T(t_f) = T_f \tag{12}$$

## 2.4 Constraints

We impose constraints on the position states and control-related quantities.

$$U_{\min} \le U \le U_{\max} \quad U = T, \theta, \psi, \dot{T}, u_T, u_\theta, u_\psi \tag{13}$$

$$\begin{bmatrix} x_{\min} \\ y_{\min} \\ z_{\min} \end{bmatrix} \le \begin{bmatrix} x \\ y \\ z \end{bmatrix} \le \begin{bmatrix} x_{\max} \\ y_{\max} \\ z_{\max} \end{bmatrix} \tag{14}$$

## 2.5 Objective Function

The objective function maximizes the final mass of the vehicle, thus minimizing the propellant consumption during the powered descent.

$$\min \quad -m(t_f) \tag{15}$$

# 3 Convex Formulation

## 3.1 Discretization

To discretize the above problem, we divide the trajectory into N segments of equal length, $\Delta t$. Thus, there are N+1(0~N) trajectory state points that need to be calculated. N is then determined by balancing computational cost and computational accuracy. The discretization method also significantly affects computational cost and numerical accuracy. This study adopts the first-order hold (FOH) discretization method, whose computational accuracy is significantly higher than that of the zero-order hold method. Compared with pseudospectral methods, it is simpler and involves lower computational load, which is important in real time onboard planning. The discretized versions of the optimal problem is presented below.

The virtual control variables $v_{cx,y,z}$ in Eq. (22) are used to prevent artificial infeasibility [4].

Objective Function:

$$\min \quad -m[\mathrm{N}] \tag{16}$$

Boundary Conditions:

$$\boldsymbol{r}[0]=\boldsymbol{r}_0, \boldsymbol{v}[0]=\boldsymbol{v}_0, \theta[0]=\theta_0, \psi[0]=\psi_0, m[0]=m_0 \tag{17}$$

$$\boldsymbol{r}[\mathrm{N}]=\boldsymbol{r}_f, \boldsymbol{v}[\mathrm{N}]=\boldsymbol{v}_f, \theta[\mathrm{N}]=\theta_f, \psi[\mathrm{N}]=\psi_0, T[\mathrm{N}]=T_f \tag{18}$$

Dynamics:

$$m[\mathrm{k}+1]=m[\mathrm{k}]-\frac{1}{2}\Delta t\left[\frac{T[\mathrm{k}]}{I_{sp}(T[\mathrm{k}])g_0}+\frac{T[\mathrm{k}+1]}{I_{sp}(T[\mathrm{k}+1])g_0}\right] \tag{19}$$

$$\boldsymbol{r}[\mathrm{k}+1]=\boldsymbol{r}[\mathrm{k}]+\boldsymbol{v}[\mathrm{k}]\Delta t+\frac{1}{3}\Delta t^2(\boldsymbol{a}[\mathrm{k}]+\frac{1}{2}\boldsymbol{a}[\mathrm{k}+1]) \tag{20}$$

$$\boldsymbol{v}[\mathrm{k}+1]=\boldsymbol{v}[\mathrm{k}]+0.5\Delta t(\boldsymbol{a}[\mathrm{k}]+\boldsymbol{a}[\mathrm{k}+1]) \tag{21}$$

$$\boldsymbol{a}[\mathrm{k}]=\frac{1}{m[\mathrm{k}]}G_{BODY}^{SF}(\theta[\mathrm{k}],\psi[\mathrm{k}])\begin{bmatrix}T[\mathrm{k}]+F_x[\mathrm{k}]\\F_y[\mathrm{k}]\\F_z[\mathrm{k}]\end{bmatrix}+\begin{bmatrix}v_{cx}[\mathrm{k}]\\v_{cy}[\mathrm{k}]\\v_{cz}[\mathrm{k}]\end{bmatrix}+\boldsymbol{g} \quad \mathrm{k}=0\sim\mathrm{N} \tag{22}$$

$$U[\mathrm{k}+1]=U[\mathrm{k}]+0.5\Delta t(\dot{U}[\mathrm{k}]+\dot{U}[\mathrm{k}+1]) \quad U=T,\theta,\psi,\dot{T} \tag{24}$$

Except for Eq. (22), in all other equations, the value of k ranges from 0 to N-1.

Control and State Constraints:

The value of k ranges from 0 to N.

$$U_{\min} \le U[\mathrm{k}] \le U_{\max} \quad U = T, \theta, \psi, \dot{T}, u_T, u_\theta, u_\psi \tag{25}$$

$$\begin{bmatrix} x_{\min} \\ y_{\min} \\ z_{\min} \end{bmatrix} \le \begin{bmatrix} x[\mathrm{k}] \\ y[\mathrm{k}] \\ z[\mathrm{k}] \end{bmatrix} \le \begin{bmatrix} x_{\max} \\ y_{\max} \\ z_{\max} \end{bmatrix} \tag{26}$$

## 3.2 Linearization, trust region and virtual control

All dynamic equations in Section 3.1 are nonlinear and are linearized using the first-order Taylor expansion. To address the issues of potential unbounded problem and artificial infeasibility caused by linearization, it is necessary to introduce trust region constraints and virtual control variables for resolution.

Compared with many studies [6-9] that impose trust region constraints on all states and control variables, we find that satisfactory results can be achieved by only applying trust region constraints to highly nonlinear objects in the dynamics, while significantly reducing the constraint dimension.

We denote the difference between a quantity in the i$^{\text{th}}$ and its counterpart in the (i-1)$^{\text{th}}$ iteration as

$$\delta x_i[\mathrm{k}] \triangleq x_i[\mathrm{k}] - x_{i-1}[\mathrm{k}] \quad i > 1 \tag{27}$$

We choose $\delta\Delta t$, $\delta\theta$, $\delta\psi$ to applying trust region constraint. The $\Delta t$ appears in the form of a quadratic power in Eq. (20), while the Euler angles appear in trigonometric functions. These highly nonlinear terms must be subject to trust region constraints to avoid excessive degradation of approximation accuracy and unbounded problems. Other terms such as $T$ and $m$ are highly coupled with the aforementioned $\Delta t$ and Euler angles, and will not present unbounded problems under the joint constraints such as the equality constraints of dynamic equations.

$$\left\| \delta\Delta t_i \right\| \le \lambda_{\Delta t} \tag{28}$$

$$\delta u_i^c[\mathrm{k}] = u_i^c[\mathrm{k}] - u_{i-1}^c[\mathrm{k}], \mathrm{c} = \theta, \psi \tag{29}$$

$$\left\| \delta u_i^\theta[0], \cdots, \delta u_i^\theta[\mathrm{N}], \delta u_i^\psi[0], \cdots, \delta u_i^\psi[\mathrm{N}] \right\| \le \lambda_{\Delta E} \tag{30}$$

Regarding the artificial infeasibility caused by linearization, there is also no need to add virtual control terms to all dynamic equations [6-9]. Adding virtual control term only to Eq. (22) can influence the position and velocity equations, which is sufficient to avoid artificial infeasibility while reducing the problem dimension.

With the introduction of thrust acceleration and Euler angle rates, the thrust, thrust rate and Euler angles, are treated as state variables, which are presented as follows.

$$\boldsymbol{s} = [\boldsymbol{r}\ \boldsymbol{v}\ m\ T\ \theta\ \psi\ \dot{T}\ ]^{\mathrm{T}} \tag{31}$$

The control variables are showed below.

$$\boldsymbol{u} = [u_T\ u_\theta\ u_\psi\ ]^{\mathrm{T}} \tag{32}$$

The virtual control variables defined as below, $m_v$ is the upper bound on the magnitude of the virtual control vector.

$$\boldsymbol{v}_c = \begin{bmatrix} v_{ax} & v_{ay} & v_{az} & m_v \end{bmatrix}^{\mathrm{T}} \tag{33}$$

We define a combined variable $\boldsymbol{\chi}[\mathrm{k}]$ that includes the k and k+1 notes of a certain quantity, which is used to describe the linearized dynamic equations.

$$\boldsymbol{\chi}_c[\mathrm{k}] \triangleq \begin{bmatrix} c^{\mathrm{T}}[\mathrm{k}] & c^{\mathrm{T}}[\mathrm{k+1}] \end{bmatrix}^{\mathrm{T}} \tag{34}$$

$$\boldsymbol{\chi}[\mathrm{k}] \triangleq \begin{bmatrix} \Delta t & \chi_s^{\mathrm{T}}[\mathrm{k}] & \chi_{\boldsymbol{u}}^{\mathrm{T}}[\mathrm{k}] & \chi_{v_c}^{\mathrm{T}}[\mathrm{k}] \end{bmatrix}^{\mathrm{T}} \quad \mathrm{k} = 0\sim\mathrm{N\text{-}1} \tag{35}$$

In the discretized dynamic equations, the nonlinear terms are denoted as *f*. A first-order Taylor expansion is performed on all *f* terms, resulting in the linearized discrete dynamic equations (41) ~ (44).

$$f_m(\boldsymbol{\chi}[\mathrm{k}]) = -\frac{1}{2}\Delta t\left[\frac{T[\mathrm{k}]}{I_{sp}(T[\mathrm{k}])g_0} + \frac{T[\mathrm{k+1}]}{I_{sp}(T[\mathrm{k+1}])g_0}\right] \tag{36}$$

$$\boldsymbol{f}_r(\boldsymbol{\chi}[\mathrm{k}]) \triangleq \boldsymbol{v}[\mathrm{k}]\Delta t + \frac{1}{3}\Delta t^2(\boldsymbol{a}[\mathrm{k}] + \frac{1}{2}\boldsymbol{a}[\mathrm{k+1}]) \tag{37}$$

$$\boldsymbol{f}_v(\boldsymbol{\chi}[\mathrm{k}]) \triangleq 0.5\Delta t(\boldsymbol{a}[\mathrm{k}] + \boldsymbol{a}[\mathrm{k+1}]) \tag{38}$$

$$\boldsymbol{a}[\mathrm{k}] = \frac{1}{m[\mathrm{k}]} G_{BODY}^{SF}(\theta[\mathrm{k}], \psi[\mathrm{k}]) \begin{bmatrix} T[\mathrm{k}] + F_x[\mathrm{k}] \\ F_y[\mathrm{k}] \\ F_z[\mathrm{k}] \end{bmatrix} + \begin{bmatrix} v_{cx}[\mathrm{k}] \\ v_{cy}[\mathrm{k}] \\ v_{cz}[\mathrm{k}] \end{bmatrix} + \boldsymbol{g} \tag{39}$$

$$f_U(\chi[\mathrm{k}]) = 0.5\Delta t(\dot{U}[\mathrm{k}] + \dot{U}[\mathrm{k+1}]), \quad \mathrm{U}=T, \theta, \psi, \dot{T} \tag{40}$$

$$m[\mathrm{k+1}] = \mathrm{m}[\mathrm{k}] + f_m(\boldsymbol{\chi}_{i-1}[\mathrm{k}]) + \left.\frac{\partial f_m}{\partial \boldsymbol{\chi}}\right|_{\boldsymbol{\chi}_{i-1}[\mathrm{k}]} \delta\boldsymbol{\chi}_i[\mathrm{k}] \tag{41}$$

$$\boldsymbol{r}[\mathrm{k+1}] = \boldsymbol{r}[\mathrm{k}] + \boldsymbol{f}_r(\boldsymbol{\chi}_{i-1}[\mathrm{k}]) + \left.\frac{\partial \boldsymbol{f}_r}{\partial \boldsymbol{\chi}}\right|_{\boldsymbol{\chi}_{i-1}[\mathrm{k}]} \delta\boldsymbol{\chi}_i[\mathrm{k}] \tag{42}$$

$$\boldsymbol{v}[\mathrm{k+1}] = \boldsymbol{v}[\mathrm{k}] + \boldsymbol{f}_v(\boldsymbol{\chi}_{i-1}[\mathrm{k}]) + \left.\frac{\partial \boldsymbol{f}_v}{\partial \boldsymbol{\chi}}\right|_{\boldsymbol{\chi}_{i-1}[\mathrm{k}]} \delta\boldsymbol{\chi}_i[\mathrm{k}] \tag{43}$$

$$U[\mathrm{k+1}] = U[\mathrm{k}] + f_U(\boldsymbol{\chi}_{i-1}[\mathrm{k}]) + \left.\frac{\partial f_U}{\partial \boldsymbol{\chi}}\right|_{\boldsymbol{\chi}_{i-1}[\mathrm{k}]} \delta\boldsymbol{\chi}_i[\mathrm{k}], \quad \mathrm{U}=T, \theta, \psi, \dot{T} \tag{44}$$

In the calculation of partial derivatives, the partial derivatives of aerodynamic forces and specific impulse are obtained through numerical differentiation, such as $\frac{\partial F_x[\mathrm{k}]}{\partial v_x[\mathrm{k}]}$, $\frac{\partial F_x[\mathrm{k}]}{\partial \theta[\mathrm{k}]}$ and $\frac{\partial I_{sp}[\mathrm{k}]}{\partial T[\mathrm{k}]}$, the other partial derivatives use analytical solutions.

## 3.3 Initialization

For the first optimization calculation, initialization is required to provide a reference trajectory. This paper adopts the initialization method from Reference 4, where the inputs that need subjective guessing are the flight time $t_{\mathrm{f}}$ (equivalent $\mathrm{N}\Delta t$) and propellant consumption. Subsequent numerical simulations

show that the proposed method is not sensitive to different input values of flight time and propellant consumption.

The command obtained from the initialization calculation is the thrust vector in the ground-fixed coordinate system. This solution is converted into the forms of thrust magnitude and Euler angles; meanwhile, numerical differentiation is performed on the thrust magnitude and Euler angles to obtain the thrust rate and angular velocity.

## 3.4 Nondimensionalization

Another method to improve computational efficiency is nondimensionalization. In the original problem, the orders of magnitude of different physical quantities vary significantly, and the loss of numerical precision caused by these magnitude differences will lead to difficulties in optimizer convergence and an increase in the number of iterations. This problem can be significantly alleviated after nondimensionalization. The following nondimensionalization is used:

1) The distances are normalized by $R_0$, it can be set as the distance between the current position and the landing site or a nearby integer value.
2) Time is normalized by $\sqrt{R_0 / g_0}$ .
3) The velocities are normalized by $\sqrt{R_0 g_0}$ .
4) The mass is normalized by initial mass: $m_0$.
5) The forces are normalized by $m_0 g_0$ .

## 3.5 Optimization problem formulation

After discretization and linearization, the original problem in Section 2 is transformed into a convex optimization problem. In this problem we treat the states and control variables of all nodes as design variables in the optimization problem, and convert the dynamic equations into equality constraints at the nodes. Since all the newly added equality constraints are linear constraints, their impact on the computational load is relatively small.

### *3.5.1 Design Variables*

Based on the definitions in Eqs. (31) ~ (33), the augmented discrete design variables are defined as

$$\boldsymbol{X} = [\boldsymbol{s}_0^{\mathrm{T}}\ \boldsymbol{u}_0^{\mathrm{T}}\ \boldsymbol{v}_{c0}^{\mathrm{T}}\ \cdots \boldsymbol{s}_k^{\mathrm{T}}\ \ \boldsymbol{u}_k^{\mathrm{T}}\ \boldsymbol{v}_{ck}^{\mathrm{T}}\ \cdots\ \boldsymbol{s}_N^{\mathrm{T}}\ \ \boldsymbol{u}_N^{\mathrm{T}}\ \boldsymbol{v}_{cN}^{\mathrm{T}}\ \Delta t\ \lambda_{\Delta t}\ \lambda_{\Delta E}\ \ \lambda_{u_T}\ \lambda_{u_\theta}\ \lambda_{u_\psi}]^{\mathrm{T}} \tag{45}$$

### *3.5.2 Constraints*

**Equality Constraints**

The equality constraints include the boundary conditions (17) and (18), as well as the discrete dynamic equations (41) to (44) for k = 0 to N-1.

**Inequality Constraints**

The inequality constraints include the constraints on states and controls from Eqs (25) and (26). In addition, there are the following second order cone constraints. Eqs. (46) and (47) are the trust region constraints. Eq. (48) is command smoothing constraint, which serve to keep the fluctuations of control variables as smooth as possible. All the λ terms will be incorporated into the objective function as penalty terms.

$$\left\|\delta\Delta t_i\right\| \le \lambda_{\Delta t} \tag{46}$$

$$\left\|\delta u_i^{\theta}[0],\cdots,\delta u_i^{\theta}[\mathrm{N}],\delta u_i^{\psi}[0],\cdots,\delta u_i^{\psi}[\mathrm{N}]\right\| \le \lambda_{\Delta E} \tag{47}$$

$$\left\|u_c[0],\cdots,u_c[\mathrm{N}]\right\| \le \lambda_{u_c} \quad c = T,\theta,\psi \tag{48}$$

### *3.5.3 Objective Function*

All trust region constraints and smooth accumulation terms are added to the objective function as penalty terms.

$$\min \ -\mathrm{w}_{mf} m[\mathrm{N}] + \mathrm{w}_{v_c} \sum_{i=0}^{N} m_{vi} + \mathrm{w}_{\lambda\Delta t}\,\lambda_{\Delta t} + \mathrm{w}_{\lambda\Delta E}\,\lambda_{\Delta E} + \mathrm{w}_{\lambda u_T}\,\lambda_{u_T} + \mathrm{w}_{\lambda u_\theta}\,\lambda_{u_\theta} + \mathrm{w}_{\lambda u_\psi}\,\lambda_{u_\psi} \tag{49}$$

The weights $\mathrm{w}_{v_c}$ $\mathrm{w}_{\lambda\Delta t}$ $\mathrm{w}_{\lambda\Delta E}$ $\mathrm{w}_{\lambda u_T}$ $\mathrm{w}_{\lambda u_\theta}$ $\mathrm{w}_{\lambda u_\psi}$ measure the relative importance of the penalization of trust region and virtual controls with respect to the true objective function. To ensure the rapid contraction of the virtual control, the weight coefficient $\mathrm{w}_{v_c}$ should be two to three orders of magnitude larger than that of the others.

## 3.6 Successive Convex Programming Algorithm

The steps of the algorithm are described as follows:

1) Specify vehicle and environmental parameters, boundary conditions. The algorithm parameters are specified in algorithm design phase and cannot be changed during flight.
2) Specify a time of flight guess, $t_{f0}$, and the propellant consumption, solve the initial optimization problem.
3) Convert the solution into thrust and Euler angles format.
4) Solve the optimization problem described in Section 3.5 until convergence is achieved. The convergence criterion is the sum of the squares of the differences between all states and control variables and their respective results from the previous calculation.

The SCP process is shown in Fig 3.

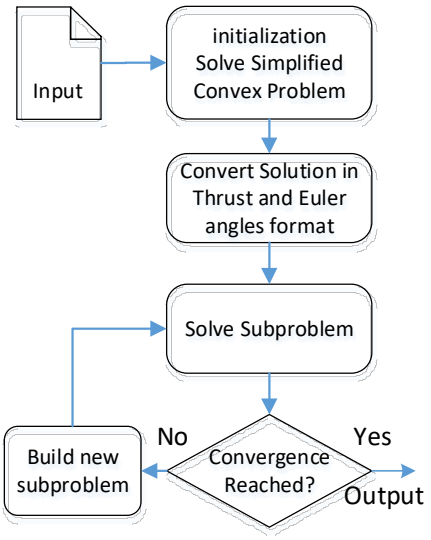


**Fig. 3 Successive Convex Programming Process**

# 4 Numerical Simulation

In this section we present an example scenario of a powered landing, subject to the constraints discussed in this paper. The algorithm uses the open-source solver ECOS for solution.

## 4.1 Scenario Settings and Vehicle Parameters

The initial and terminal conditions for the simulation are set as shown in Table 1, and the main characteristic parameters of the vehicle are presented in Table 2. Table 3 presents the settings of control constraints. The number of discrete points N is set to 40.

**Table 1 Powered Landing Initial and Final states**

| Quantity | Value | Unit | Quantity | Value | Unit |
|---|---|---|---|---|---|
| $x_0$ | 400 | m | $x_f$ | 0 | m |
| $y_0$ | 3,600 | m | $y_f$ | 0 | m |
| $z_0$ | 200 | m | $z_f$ | 0 | m |
| $v_{x0}$ | -40 | m/s | $v_{xf}$ | 0 | m/s |
| $v_{y0}$ | -250 | m/s | $v_{yf}$ | -1 | m/s |
| $v_{z0}$ | -30 | m/s | $v_{zf}$ | 0 | m/s |

**Table 2 Characteristic parameters of the vehicle**

| $m_0$ (kg) | $T_{\text{std}}$ (kN) | $T_{\max}$ | $T_{min}$ |
|---|---|---|---|
| 30000 | 490 | $0.9T_{\text{std}}$ | $0.4T_{\text{std}}$ |

**Table 3 Rates and control Constraints**

| $\dot{\theta}_{\max}$ (deg/s) | $\dot{T}_{\max}$ (1/s) | $u_{T_{\max}}$ ($1/s^2$) | $u_{\theta_{\max}}$ $u_{\psi_{\max}}$ (deg /s) |
|---|---|---|---|
| 3.0 | $0.3T_{\text{std}}$ | $0.1T_{\text{std}}$ | 0.3 |

## 4.2 Converged Solution

First, we present the optimization results under the input $t_{f0}$ = 44s, and discuss the constraint satisfaction and convergence of the generated trajectory. A 44 seconds flight time guess is much larger than the final converged flight time of 31.6 seconds. Therefore, the planned trajectory needs to cover a longer distance to consume the excess time, resulting in a significant discrepancy between the reference trajectory and the final iterative result, which increases the difficulty of subsequent planning calculations. The convergence results under this condition demonstrate the global convergence capability of the algorithm to some extent.Figure 4 shows the position and velocity curves of the three channels in the fixed coordinate system. Figure 7 shows the variation of the 3D spatial position. The dots along the trajectory indicate discretization points. Fig. 5 and Fig. 6 show the thrust, Euler angles, and their higher-order quantities. It can be seen that the thrust magnitude remains within the upper and lower limits, while the other quantities do not reach the constraint boundaries. All control quantities exhibit smooth curves.

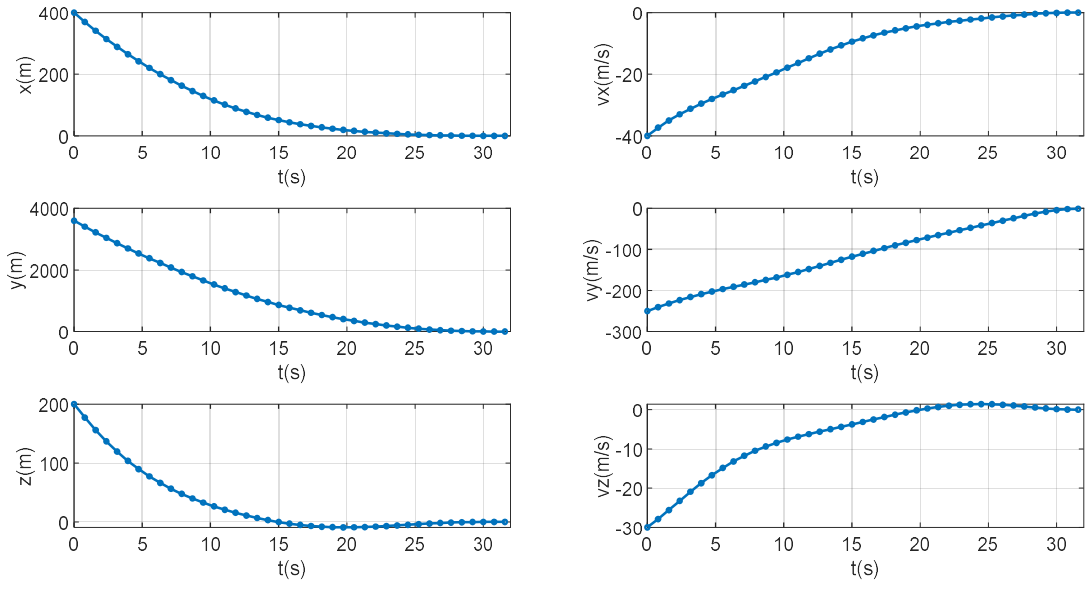


**Fig. 4 x, y and z components of the positions and velocities**

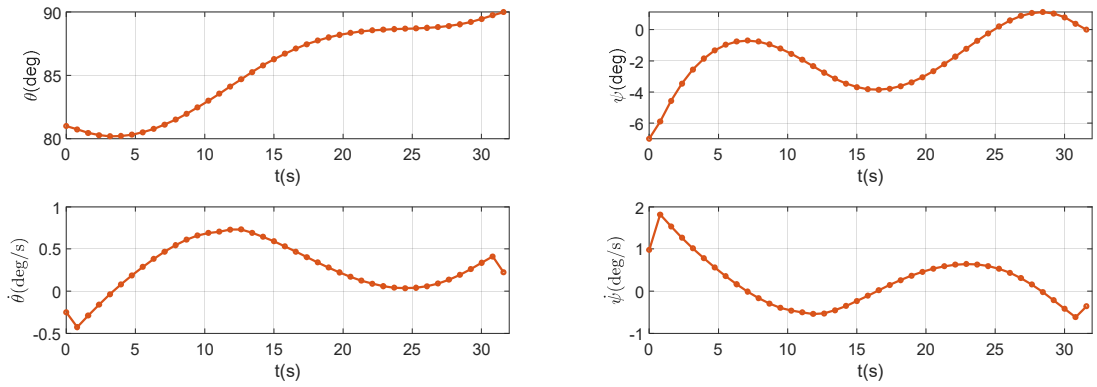


**Fig.5 pitch and yaw angel profile of the converged trajectory**

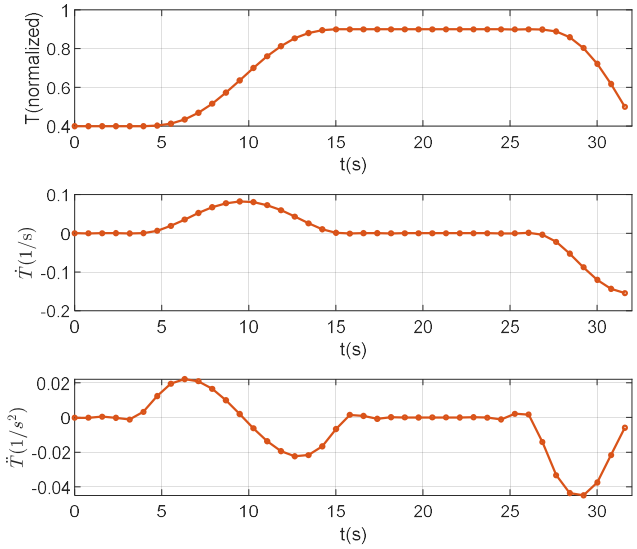


**Fig.6 Thrust profile**

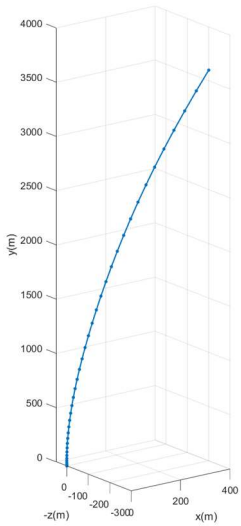


**Fig.7 3D trajectory**

## 4.3 Convergence Behavior

Figure. 8~11 show the iteration histories of state and control quantities. It can be seen that the results of the early iterations differ significantly from the final results, but after three iterations, they are already close to the final trajectory (marked in red). Fig. 12 presents the iteration histories of the virtual control upper bound, flight time difference, trust region upper bound, and convergence flag.

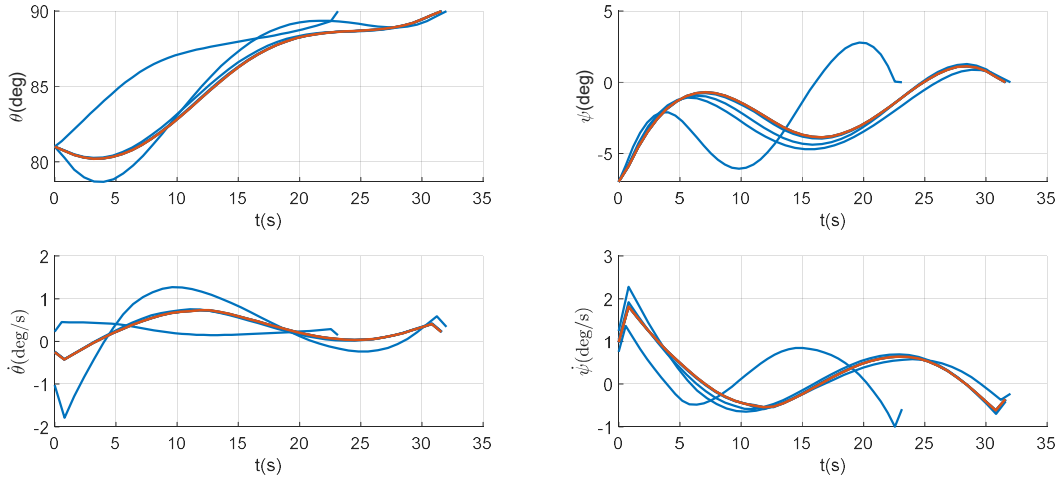


**Fig.8 Iteration history of Euler angles, and angles rates**

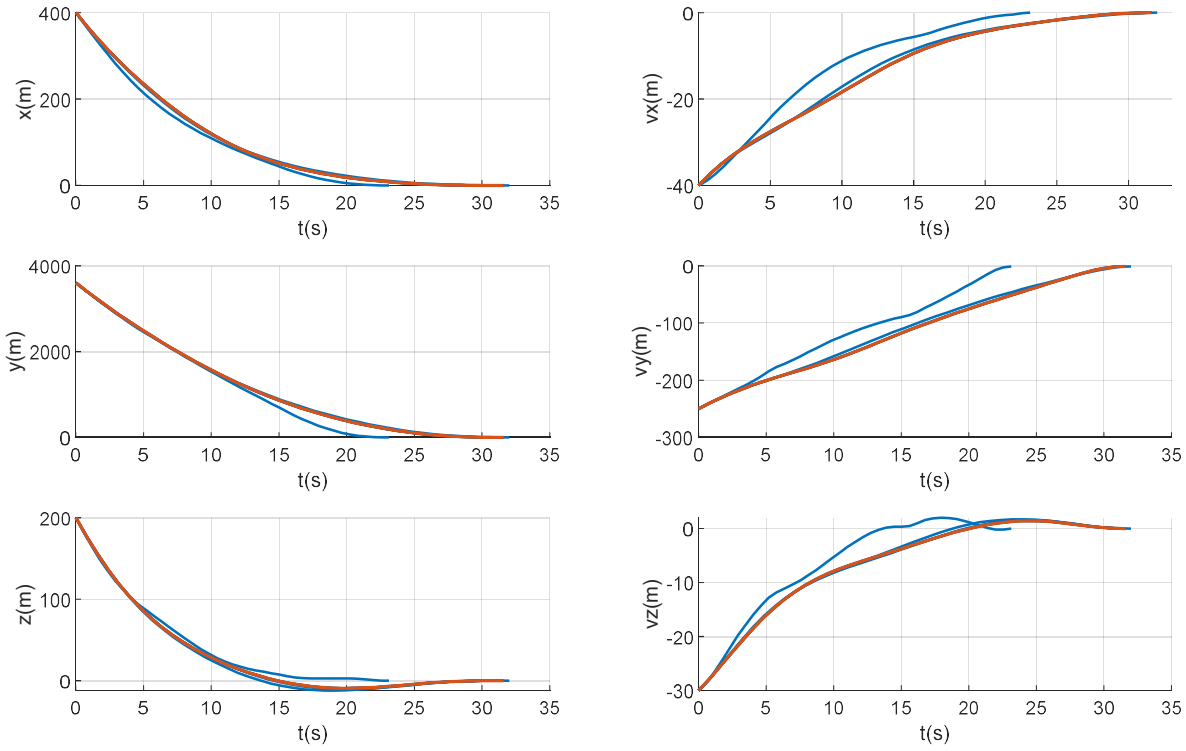


**Fig.9 Iteration history of translational states**

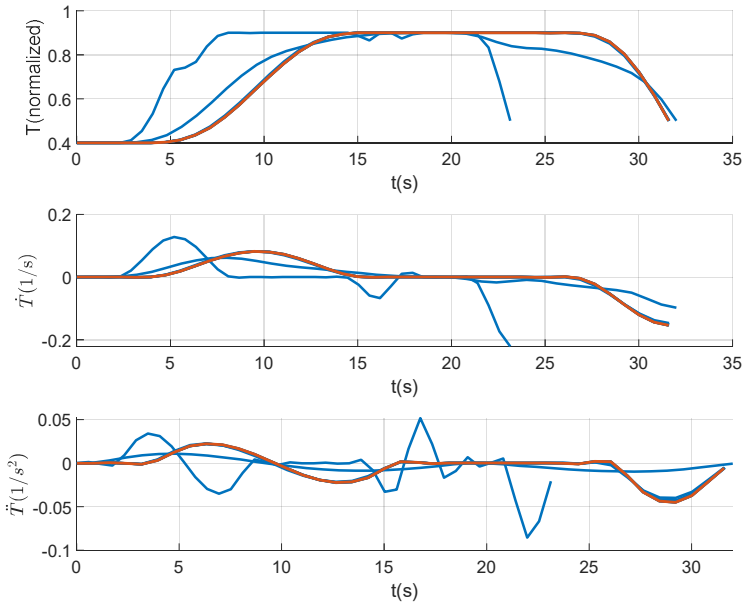


**Fig.10 Iteration history of thrust**

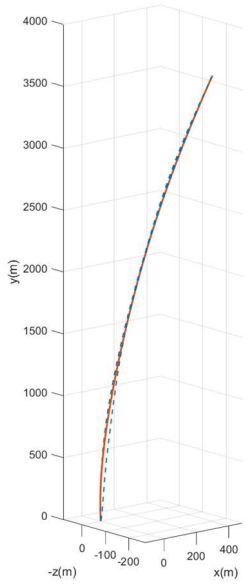


**Fig.11 3D position history**

We select four different input flight time guesses as test cases, 24s, 32s, 40s, and 44s, to investigate the global convergence of the algorithm. The results are shown in Figs. 13 and 14. It can be seen that except for the 24s case, all other cases basically converge to the same point after 4 iterations. The 24s case also reaches the same point after 10 iterations, indicating that the algorithm has good adaptability to inputs with significant differences. Based on the convergence speed corresponding to different guessed flight times, in practical engineering algorithms, setting a relatively larger input value for the flight time is more conducive to the rapid convergence of the algorithm.

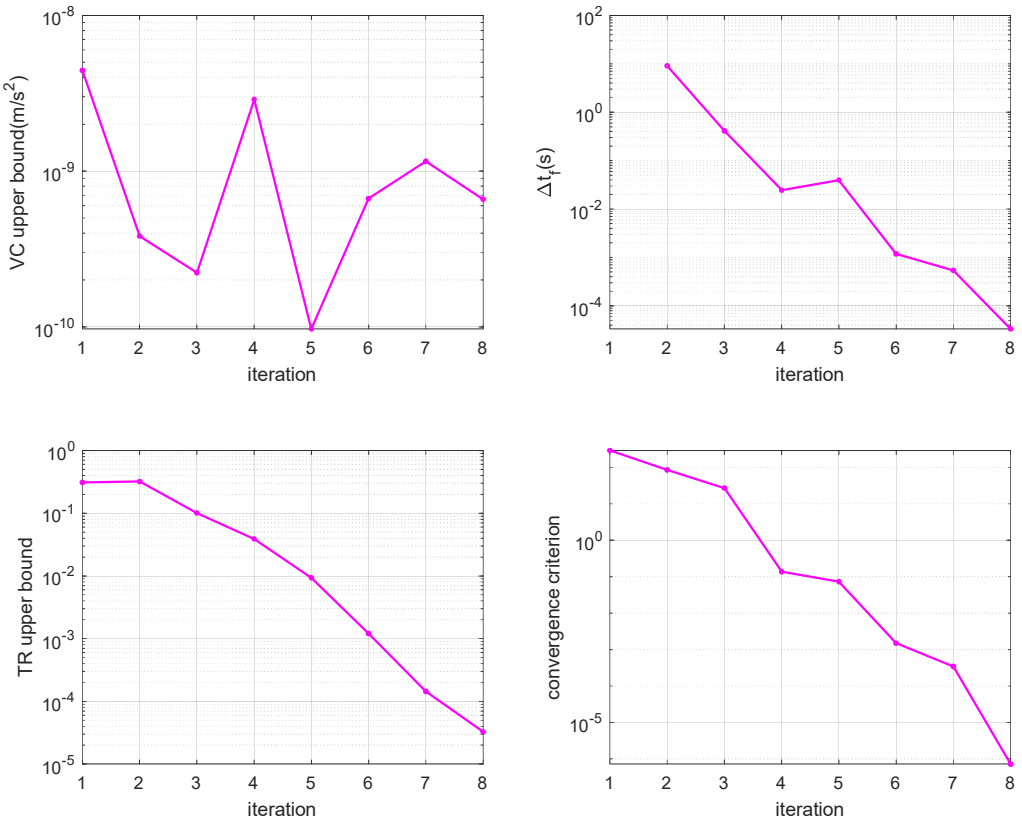


**Fig.12 Iteration history of convergence criterion, virtual control upper bound, tf difference and trust region upper bound**

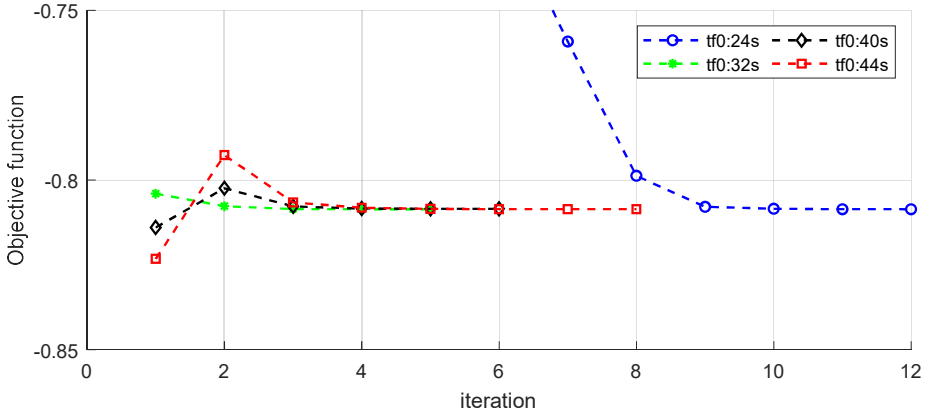


**Fig. 13 Objective function convergence history of different flight time input**

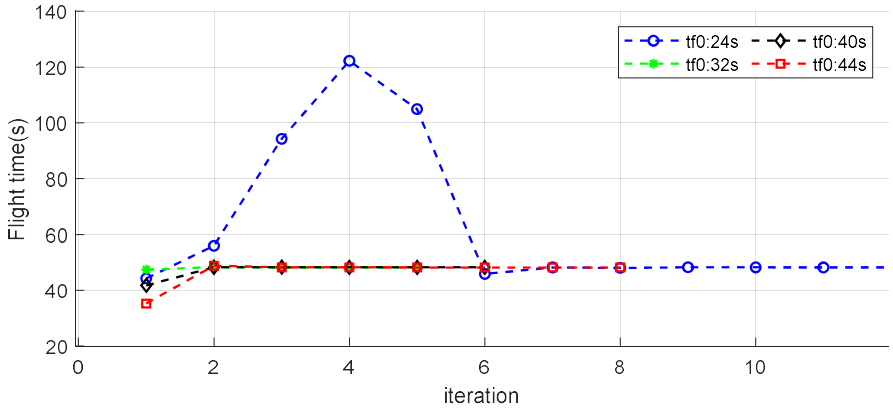


**Fig. 14 Flight time convergence history of different flight time input**

## 4.4 Numerical Precision

To verify the numerical accuracy of the discretization method and linearization, the 4th-order Runge-Kutta method is used to calculate the trajectory. The comparison between this calculation result and the optimization result is shown in Figs. 15, 16 and 17. With a flight time of approximately 31.6 seconds, the position deviation is no more than 2 meters, and compared with the initial distance, the relative deviation is no more than 0.07%. The velocity deviation is no greater than 0.1 m/s, indicating that the accuracy meets the guidance requirements.

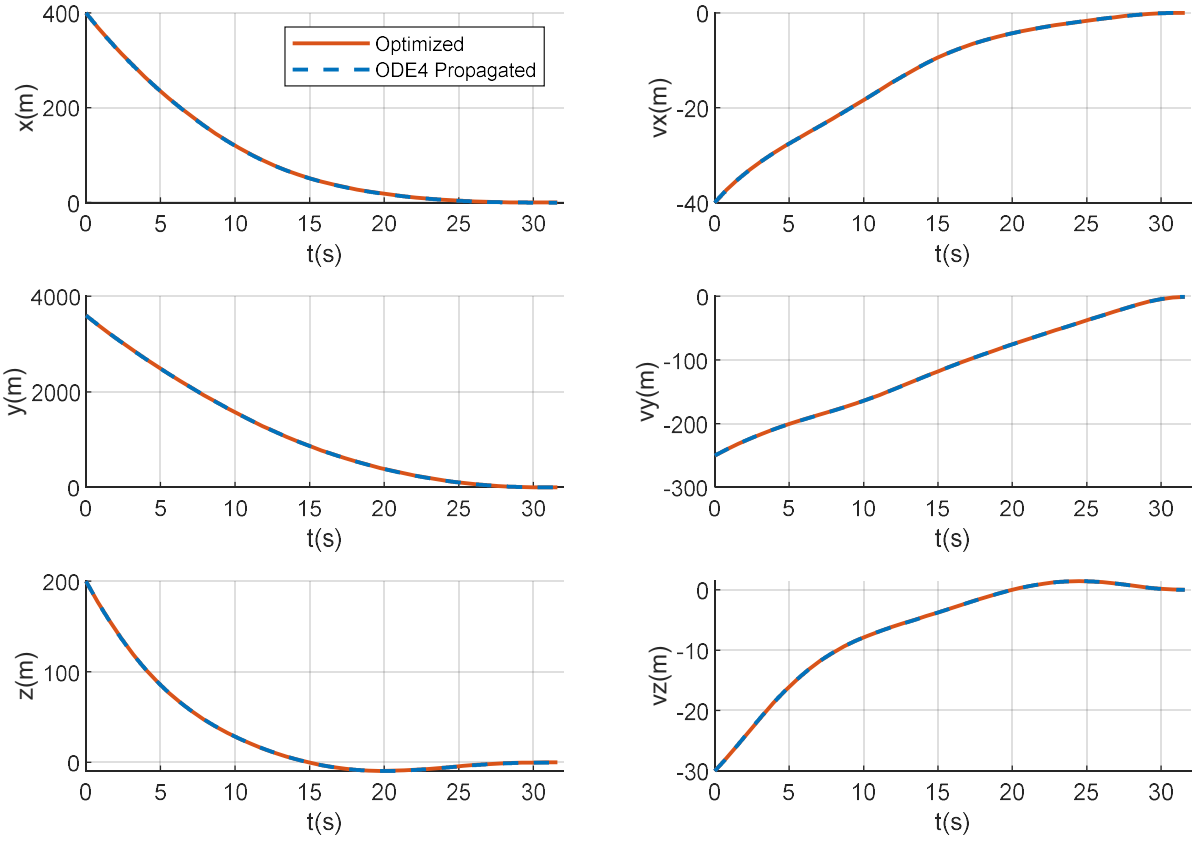


**Fig.15 Trajectory comparison with Runge-Kutta propagation**

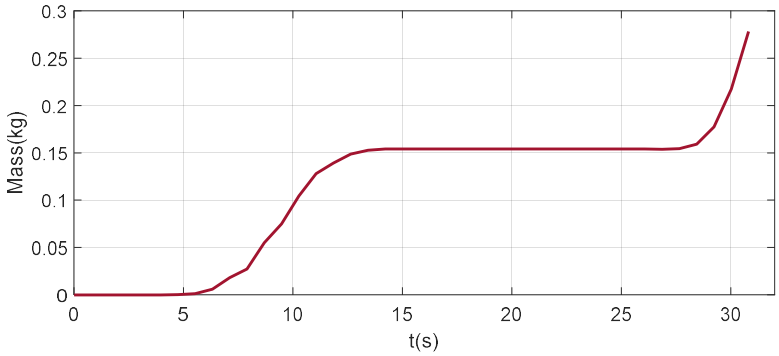


**Fig.16 Mass error with respect Runge-Kutta propagation**

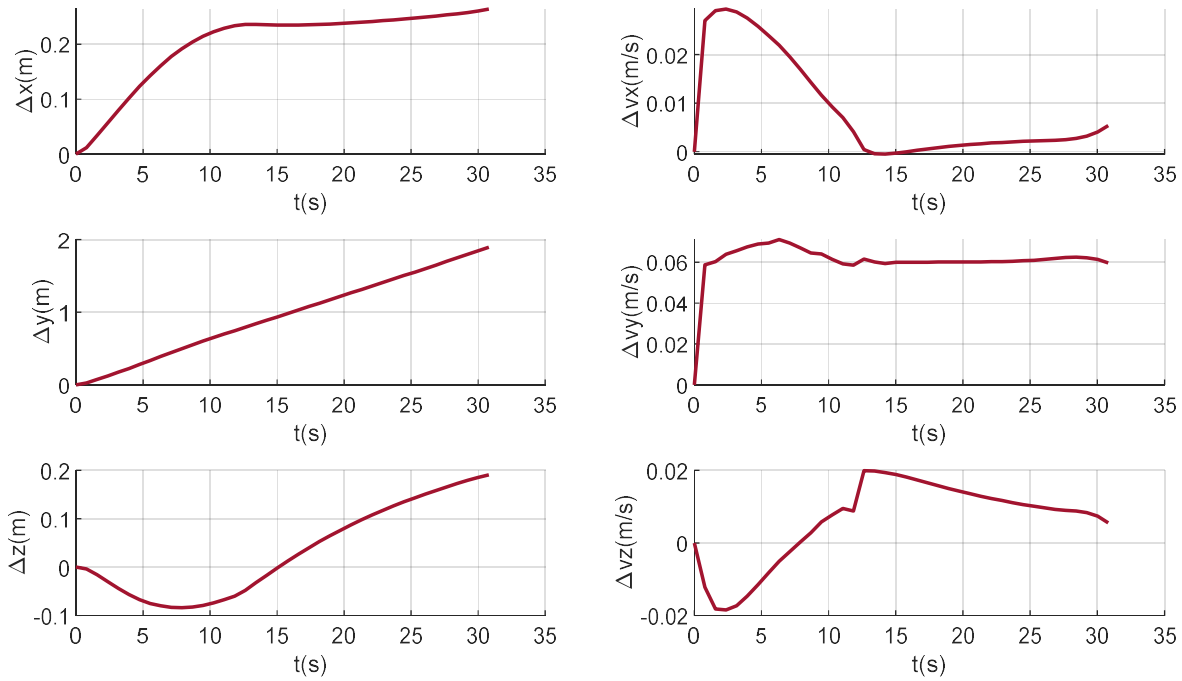


**Fig.17 Position and velocity error with respect Runge-Kutta propagation**

## 4.5 Computational Time Test

We configured the algorithm on a Rockchip RK3588 embedded processor for computational time testing. This chip is equipped with four high-performance ARM A76 computing cores, whose clock frequency is reduced to 2.05 GHz to enhance reliability. The flight control computer used in this test is shown in Fig. 18.

**Fig.18 Flight control computer with high performance processers**

We conducted 500 Monte Carlo random tests, all of which converged to the acceptable criterion (convergence indicator < $1\times10^{-4}$). The maximum number of iterations in one run was 9 (including the initialization iteration), with a maximum computation time of 346 ms. The detailed results are presented in Fig.19. In practical engineering, trajectory planning for the guidance algorithm is generally performed every 2 seconds, so the computational time can meet the requirements of real-time onboard trajectory planning.

As mentioned in Section 3.3, the algorithm requires initial guesses for the flight time t_f and propellant consumption. Numerical tests indicate that the flight time has a more pronounced effect on the

convergence rate of the iterations. An initial guess close to and slightly larger than the final converged flight time can reduce the number of iterations by 2 to 3. The initial guess for flight time can be determined through extensive offline testing. Additionally, the weighting coefficients of the objective function also influence the convergence characteristics, which need to be fine-tuned to achieve a trade-off among performance, constraints, and convergence behavior.

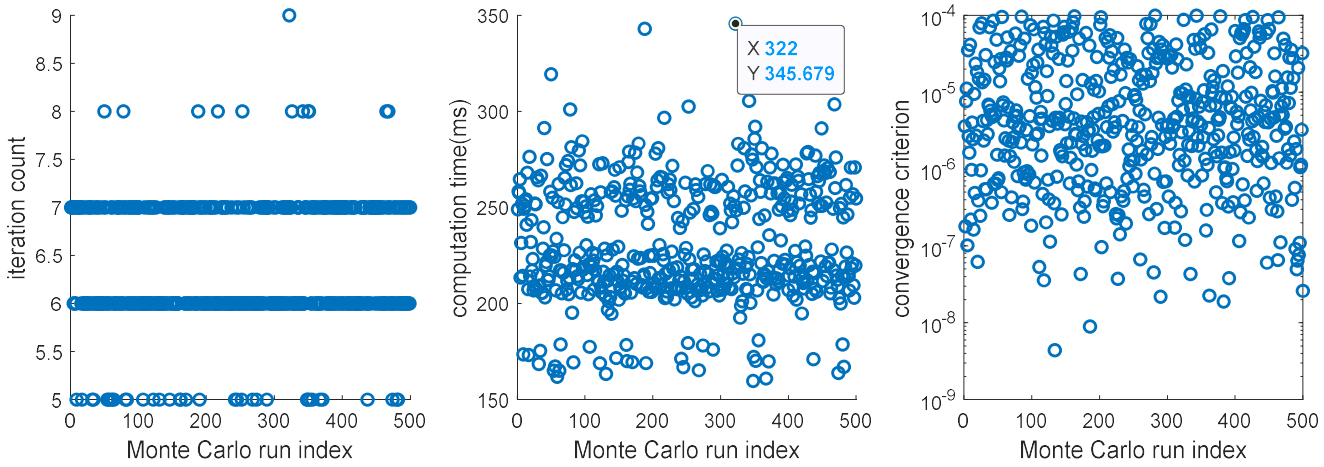


**Fig.19 Monte Carlo Test: Iteration Count and Computational Time**

# 5 Conclusions

In this paper, we presented a successive convex programming algorithm designed for the launch vehicle powered landing problem. This method addresses the non-convex problems caused by aerodynamic effects and non-constant engine specific impulse through sequential convex programming. It also introduces thrust acceleration as the control variable to generate smoother thrust commands, making it more suitable for engineering applications. In addition, by means of treatments such as reducing the number of trust region constraints and normalizing variables, this method significantly reduces the computational cost of the algorithm and ensures its applicability to real-time onboard planning operations. Numerical simulations show that the algorithm can generate trajectories satisfying constraint conditions under different inputs. Its good global convergence and low computational cost enable it to meet the conditions for practical engineering application during flight.

# Declaration of Use of Artificial Intelligence

Artificial intelligence (AI) was only used to translate Chinese into English and proofread for some grammatical errors.

# References


[1] Blackmore L. Autonomous Precision landing of space rockets. *The Bridge*, 46(4), 2016.

[2] Behçet Açıkmeşe, and Scott R. Ploen. Convex programming approach to powered descent guidance for mars landing. *Journal of Guidance, Control, and Dynamics*, 30(5), September 2007. doi: 10.2514/1.27553 .

[3] Behçet Açıkmeşe, John M. Carson III, and Lars Blackmore. Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem. In *Proceedings of the IEEE Transactions on Control Systems Technology*, 21(6), Nov. 2013, doi: 10.1109/TCST.2012.2237346

[4] Michael Szmuk and Behçet Açıkmeşe. Successive convexification for fuel-optimal powered landing with aerodynamic drag and non-convex constraints. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, San Diego, California, USA, January 2016. doi: 10.2514/6.2016-0378

[5] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC)*, Las Vegas, NV, USA, December, 2016. doi: 10.1109/CDC.2016.7798816

[6] Marco Sagliano, Ansgar Heidecker, José Macés Hernández, Stefano Farì, Markus Schlotterer, Svenja Woicke, David Seelbinder and Etienne Dumont. Onboard guidance for reusable rockets: aerodynamic descent and powered Landing. In *Proceedings of the AIAA Scitech 2021 Forum*, January 2021. doi: 10.2514/6.2021-0862

[7] Marco Sagliano, Ansgar Heidecker, Stefano Farì, José Macés Hernández, Markus Schlotterer, Svenja Woicke, David Seelbinder and Etienne Dumont. Powered atmospheric landing guidance for reusable rockets: the CALLISTO studies. In *Proceedings of the AIAA SCITECH 2024 Forum*, Orlando, FL, USA, January 2024. doi: 10.2514/6.2024-1761

[8] Michael Szmuk and Behçet Açıkmeşe. Successive convexification for 6-DoF mars rocket powered landing with free-final-time. In *Proceedings of the 2018 AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida, USA, 8-12 January 2018. doi: 10.2514/6.2018-0617

[9] Michael Szmuk, Taylor P. Reynolds and Behçet Açıkmeşe. Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. *Journal of Guidance, Control, and Dynamics*, 43(8), August 2020. doi: 10.2514/1.27553

[10] Taylor P. Reynolds and Mehran Mesbahi. The crawling phenomenon in sequential convex programming In *Proceedings of the 2020 American Control Conference (ACC),* Denver, CO, USA, July 2020. doi: 10.23919/ACC45564.2020.9147550

[11] Ping Lu. Convex–concave decomposition of nonlinear equality constraints in optimal control. *Journal of Guidance, Control, and Dynamics*, 44(1), January 2021. doi: 10.2514/1.G005443

[12] . Marco Sagliano, David Seelbinder, Stephan Theil and Ping Lu. Six-degrees-of-freedom rocket landing optimization via augmented convex-concave decomposition. *Journal of Guidance, Control, and Dynamics*, 47(1), September 2023. doi: 10.2514/1.G007570