Hindawi
International Journal of Aerospace Engineering
Volume 2022, Article ID 2969809, 18 pages
https://doi.org/10.1155/2022/2969809

Research Article

PySCP: A Multiple-Phase Optimal Control Software Using Sequential Convex Programming

Daxi Zhang[1] **and Yulin Zhang**[2,3]

[1]*School of Aerospace Science and Technology, National University of Defense Technology, Changsha 410073, China*
[2]*School of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China*
[3]*Huzhou Institute of Zhejiang University, Huzhou 310027, China*

Correspondence should be addressed to Daxi Zhang; zdx011580@126.com

Received 7 December 2021; Revised 7 March 2022; Accepted 10 March 2022; Published 13 April 2022

Academic Editor: Chen Pengyun

Optimal control problems are common in aerospace engineering. A Python software program called PySCP is described for solving multiple-phase optimal control problems using sequential convex programming methods. By constructing a series of approximated second-order cone programming subproblems, PySCP approaches to the solution of the original optimal control problem in an iterative way. The key components of the software are described in detail, including convexification, discretization, and the adaptive trust region method. The convexification of the first-order differential dynamic equation is implemented using successive linearization. Six discretization methods, including zero-order hold, first-order hold, Runge-Kutta, and three hp pseudospectral collocation methods, are implemented so that different types of optimal control problems can be tackled efficiently. Adaptive trust region method is employed, and robust convergence is achieved. Both free-final-time problem and fixed-final-time problem can be solved by the software. The application of the software is demonstrated on three optimal control problems with varying complexity. PySCP provides researchers a useful toolkit to solve a wide variety of optimal control problems using sequential convex programming.

1. Introduction

Many aerospace engineering problems require to solve an optimal control problem (OCP) [1], such as the ascent trajectory of launch vehicles [2, 3], hypersonic vehicle reentry [4], spacecraft rendezvous [5, 6], and extraterrestrial objects soft landing [7]. Traditionally, there are two methods that can be used to solve OCPs, the indirect methods and direct methods [8]. The former derives the optimality condition based on the Pontryagin maximum principle and classical calculus of variation theories, resulting in a two-point boundary value problem (TPBVP). The costate variables in the TPBVP have no explicit meaning and are extremely sensitive to initial guesses, making the TPBVP very difficult to solve. In contrast, the direct methods discrete the original continuous-time OCP into a nonlinear programming (NLP) problem, which can be then solved by an NLP solver. The direct methods are more often used in practice [9] because the analytical optimality condition is generally very complicated.

In most aerospace engineering problems, the NLP problems obtained by direct methods are large-scale sparse and nondeterministic polynomial-time hard, and the computation time to reach the expected accuracy is not guaranteed or limited. It is possible that the computation time is so long that no result can be reached. Although the dramatic increase in computing power in past decades makes it possible for direct methods to be widely used in aerospace engineering, rapid computation and guaranteed convergence are still in pursuit, especially in multidisciplinary design optimization [10] studies and online guidance and control [11] where computation efficiency is essential.

Convex optimization problems are computationally tractable and globally convergent and can be solved in polynomial time [12, 13]. These outstanding characteristics have attracted many researchers to try to solve OCPs by convex

approaches [14]. There are two ways to do this. The first way is called lossless convexification. The word "lossless" indicates that the convexification process does not lose the problem's characterization, and the convexified convex problem is equivalent to the original OCP. However, most optimal control problems cannot be approached using lossless convexification due to the intrinsic nonconvexity. In this case, the second way is applied, which is called sequential convex programming (SCP), also known as successive convex programming. Since equivalent convex problem cannot be constructed, approximate convex problem is constructed in the local neighbourhood of a reference solution (i.e., the reference trajectory). By solving the approximate convex problem, new reference trajectory is obtained, and new approximate convex problem can be generated. The solution to the optimal control problem can then be approached in an iterative way.

In recent years, significant effort has been devoted into this topic, and many aerospace engineering problems have been solved by convex approaches. Açıkmeşe et al. [15–18] proposed lossless convexification to solve the single-phase OCP of Mars pinpoint landing, which was then extended to onboard guidance for vertical landing launch vehicles and asteroid landing [19, 20]. Many advanced techniques have been developed and combined with SCP, for instance, problem reformulation by equivalent transformation [21, 22] and pseudospectral methods [23]. Both aerodynamic control and thrust control for powered landing have also been tackled using SCP [24, 25]. Sequential convex approaches have achieved significant progress in this area, and even six-degree-of-freedom free-final-time powered landing can be solved in real time [26].

Hypersonic reentry is another difficult problem that has been well addressed. Subject to significant aerodynamic forces and strict path constraints, trajectory optimization of hypersonic reentry is a highly nonlinear problem. Liu et al. [27] first introduced new control variables and obtained corresponding linear dynamics plus additional nonconvex control constraints which were then relaxed into convex constraints. By successive linearization, the original nonconvex problem is converted into a sequence of second-order convex programming (SOCP) problems. Line search and trust region methods [28], adaptive mesh refinement [29], pseudospectral discretization methods [30], and equivalent transformation [31] have also been employed associated with SCP to cope with this highly nonlinear OCP. SCP has also been extensively applied in many other problems, such as unmanned arial vehicle formation flight [32] and spacecraft rendezvous guidance [33, 34].

In past decades, there has been many scientific computing software that implemented direct methods and widely used in aerospace engineering. The first well-known direct collocation software was Optimal Trajectories by Implicit Simulation [35] (OTIS), a FORTRAN software that has general-purpose capabilities for problems in aeronautics and astronautics. Commercial general-purpose optimal control software GPOPS II [36, 37] is a Matlab toolkit that uses hp-adaptive pseudospectral methods to discrete OCPs and large-scale NLP solvers (SNOPT [38] and IPOPT [39]) for the transcribed NLP problems; DIDO [40, 41] uses pseudospectral methods and sequential quadratic programming (SQP) to solve OCPs, which are widely used in orbit maneuvering studies. Program to optimize simulated trajectory II (POST II) [42] developed by NASA uses direct shooting methods and gradient descent/SQP and is applicable in various ascent and descent trajectory optimization problems. DLR developed first European tool based on flipped Radau pseudospectral methods, referred as SPARTAN [43, 44]. IClocs2 [45, 46] is another comprehensive software suite for solving OCPs implemented in Matlab and Simulink, aiming at providing a user-friendly interface. However, there has been no public software that implements SCP to boost the research on optimal control problems using convex optimization.

This paper presents a Python software program for solving multiple-phase optimal control problems using sequential convex programming methods. PySCP first maps the time horizon of each phase onto a normalized time horizon $\tau \in [-1, 1]$ so that both fixed-final-time and free-time-time cases can be handled. The nonconvex functions in the optimal control problem need to be convexified. PySCP provides a template to convexify the dynamic equation using successive linearization. After convexification, the continuous-time problem is discretized to form a discrete SOCP problem. To do this, PySCP implements six discretization methods, including zero-order hold (ZOH), first-order hold (FOH), Runge-Kutta (RK), and three Legendre pseudospectral methods. The trust region size has a deterministic influence on the convergence property of the iteration process. An adaptive trust region method is employed to adjust the trust region size. To demonstrate the utility of PySCP, three examples of different complexity are illustrated. We hope this software can help to provide researchers a platform to deal with optimal control problems with less effort and promote the application of SCP in aerospace engineering.

2. Multiple-Phase Optimal Control Problem

The general multiple-phase optimal control problems that can be solved by PySCP are given as follows.

Problem P0. Minimize the objective function

$$J = \sum_{p=1}^{P} \left\{ \Phi^{(p)} \left[\mathbf{x}^{(p)}(t_0), \mathbf{x}^{(p)}(t_f) \right] + \int_{t_0^{(p)}}^{t_f^{(p)}} g^{(p)} \left[\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t) \right] dt \right\}, (p = 1, 2, \cdots, P), \tag{1}$$

subject to the dynamic constraints

$$\frac{d}{dt}\mathbf{x}^{(p)}(t) = \mathbf{f}^{(p)} \left[\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t) \right], \tag{2}$$

the path constraints

$$\mathbf{C}^{(p)} \left[\mathbf{x}^{(p)}(t), \mathbf{u}^{(p)}(t) \right] \leq \mathbf{0}, \tag{3}$$

the boundary constraints

$$\varphi^{(p)}\left[\mathbf{x}^{(p)}(t_0), \mathbf{x}^{(p)}\left(t_f\right)\right] \leq \mathbf{0}, \tag{4}$$

and the linkage constraints

$$\mathscr{L}_{p_{l,a}}^{p_{l,b}}\left[\mathbf{x}^{(p_i)}\left(t_f\right), \mathbf{x}^{(p_j)}(t_0)\right] \leq \mathbf{0}, \begin{cases} p_{l,a}, p_{l,b} \in [1, \cdots, P], \\ l = 1, \cdots, n_L. \end{cases} \tag{5}$$

The above functions are defined by the following mappings:

$$\begin{aligned} \Phi^{(p)} &: \mathbb{R}^{2n_x^{(p)}} \longrightarrow \mathbb{R}, \\ \mathbf{g}^{(p)} &: \mathbb{R}^{n_x^{(p)}+n_u^{(p)}} \longrightarrow \mathbb{R}, \\ \mathbf{f}^{(p)} &: \mathbb{R}^{n_x^{(p)}+n_u^{(p)}} \longrightarrow \mathbb{R}^{n_x^{(p)}}, \\ \mathbf{C}^{(p)} &: \mathbb{R}^{n_x^{(p)}+n_u^{(p)}} \longrightarrow \mathbb{R}^{n_c^{(p)}}, \\ \varphi^{(p)} &: \mathbb{R}^{2n_x^{(p)}} \longrightarrow \mathbb{R}^{n_\varphi^{(p)}}, \\ \mathscr{L}_{p_{l,a}}^{p_{l,b}} &: \mathbb{R}^{n_x^{(p_i)}+n_x^{(p_j)}} \longrightarrow \mathbb{R}^{n_{\mathscr{L}}^{(l)}}. \end{aligned} \tag{6}$$

$\mathbf{x}^{(p)} \in \mathbb{R}^{n_x^{(p)}}$ is the state variable vector in the p-th phase, and $\mathbf{u}^{(p)} \in \mathbb{R}^{n_u^{(p)}}$ is the control variable vector. $n_x^{(p)}$, $n_u^{(p)}$, $n_c^{(p)}$, and $n_\varphi^{(p)}$ are the state variable dimension, control variable dimension, number of path constraints, and number of boundary constraints in the p-th phase. n_L is the number of linkage couples, and $n_{\mathscr{L}}^{(l)}$ is the linkage constraints in the l-th linkage couple. An example of how phases can be linked is given in Figure 1. There are four phases and three linkage couples in total.

Phases can be either free-final-time or fixed-final-time. The time domain of each phase is mapped onto a normalized time domain $\tau \in [-1, 1]$ based on the following expression:

$$t^{(p)} = \frac{t_f^{(p)} - t_0^{(p)}}{2}\tau + \frac{t_f^{(p)} + t_0^{(p)}}{2}, \tau \in [-1, 1]. \tag{7}$$

Denote $\sigma^{(p)} = (t_f^{(p)} - t_0^{(p)})/2$. The multiple-phase optimal control problem can be rewritten as follows.

Problem P1. Minimize the objective function

$$J = \sum_{p=1}^{P}\left\{\Phi^{(p)}\left[\mathbf{x}^{(p)}(-1), \mathbf{x}^{(p)}(1)\right] + \sigma\int_{-1}^{1}\mathbf{g}^{(p)}\left[\mathbf{x}^{(p)}(-1), \mathbf{u}^{(p)}(1)\right]d\tau\right\}, \tag{8}$$

subject to the dynamic constraints

$$\frac{d}{d\tau}\mathbf{x}^{(p)}(\tau) = \sigma^{(p)}\mathbf{f}\left[\mathbf{x}^{(p)}(\tau), \mathbf{u}^{(p)}(\tau)\right], \tag{9}$$

the path constraints

$$\mathbf{C}^{(p)}\left[\mathbf{x}^{(p)}(\tau), \mathbf{u}^{(p)}(\tau)\right] \leq \mathbf{0}, \tag{10}$$

the boundary constraints

$$\varphi^{(p)}\left[\mathbf{x}^{(p)}(-1), \mathbf{x}^{(p)}(+1)\right] \leq \mathbf{0}, \tag{11}$$

and the linkage constraints

$$\mathscr{L}_{p_{l,a}}^{p_{l,b}}\left[\mathbf{x}^{(p_{l,a})}(1), \mathbf{x}^{(p_{l,b})}(-1)\right] \leq \mathbf{0}. \tag{12}$$

The dynamic equations are a first-order differential equation set, and all the other functions are algebraic. For simplification, the symbol for phase number (p) is omitted in the remaining part of this paper, except for the functions in the objective and linkage constraints.

2.1. Workflow of PySCP. The workflow of PySCP is illustrated in Figure 2. The basic idea behind this is that "construct the approximated SOCP and iterate to approach the original problem."

Step 1. Convexification. If all functions and constraints of the optimal control problem are convex, the optimal control problem can be reformulated as convex optimization problem without requirement for approximation, and the solution to the convex problem is also the optimal solution to the original problem. Otherwise, all nonconvex functions and constraints must be transformed into convex ones. There are many different convexification methods, and most of them require a reference trajectory for approximation.

Step 2. Discretization. The convexified convex problem is still continuous-time problem which cannot be directly handled by computers. Discretization methods approximate the functions in the continuous-time problem using state variables and control variables at the discrete points, after which a SOCP problem is constructed. The discretization method determines the unknown variable number and the computation efficiency of the transcribed convex optimization problem. In PySCP, six discretization methods are implemented, including ZOH, FOH, Runge-Kutta, and three Legendre pseudospectral methods.

Step 3. Solve the convex optimization problem. The SOCP problem can be solved efficiently in bounded computation time [12, 47]. There are many mature software or toolkit that implements SOCP solvers, such as MOSEK [48], CPLEX, SDPT3 [49], SeDuMi [50], and ECOS [51]. There is also a variety of software that provides interfaces to formulate SOCP problems, such as CVX [52], CVXPY [53], CVXOPT [54], and CVXGEN [55], to name a few. PySCP uses CVXPY to address the SOCP problem, which is solved by the primal-dual internal point method using ECOS by default.

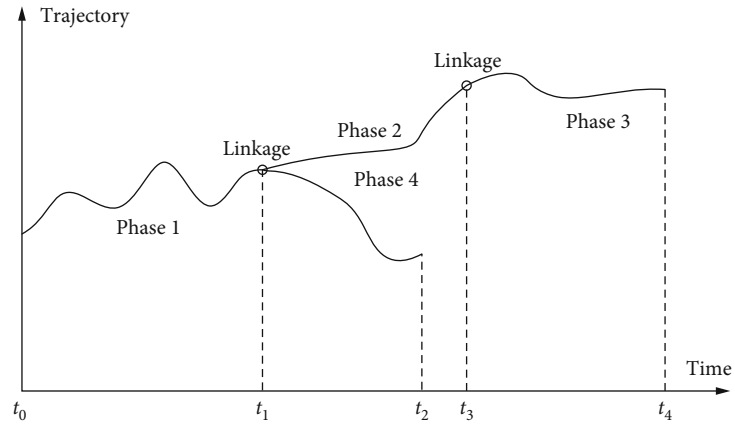

Figure 1: Schematic of linkage conditions for multiphase optimal control problem.

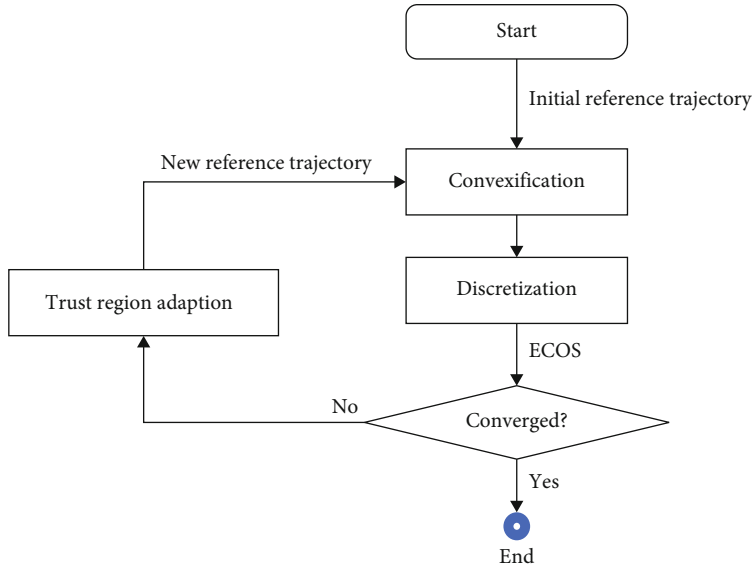

Figure 2: Workflow of PySCP.

In each iteration, the solution to the SOCP problem is checked whether it meets the convergence requirement. If positive, the algorithm stops. Otherwise, go to Step 4.

Step 4. Trust region adaption. In mathematical optimization, those methods that iterate to approach to the solution by constructing approximated problems are usually called local descent methods [56], including trust region method and line search method. The former first determines the step size and then finds the search direction. In contrast, the latter method determines the search direction first and then the step size. The basic idea behind SCP is the same as the local descent methods. Both the trust region method and line search method can be applied for iterative optimization [57]. PySCP adopts a method called adaptive trust region method. The solution to the SOCP serves as the new reference trajectory for constructing a new SOCP problem.

In the next three sections, the convexification, discretization, and adaptive trust region method will be discussed one by one.

2.2. Convexification Methods. Convexification and discretization are aimed at constructing a discrete SOCP problem in the local neighbourhood of a reference trajectory. SOCP is a kind of convex optimization problems defined as

$$\begin{aligned} \min \quad & a_i^T x \\ \text{s.t.} \quad & A_0 x = b_0 \\ & \|A_i x + b_i\| \le c_i^T x + d, i = 1, 2, \cdots, m, \end{aligned} \tag{13}$$

where $\|\cdot\|$ is 2-norm operator. In a SOCP problem, the equality function must be linear, and inequality function is convex in the sense that the constrained feasible space is a second-order cone. All the function in Problem P1 must be convexified into functions with the same form as Equation

(13). There are many convexification methods, including equivalent transformation, change of variables, successive linearization, successive approximation, and relaxation. We refer the readers to [14] for more details.

The convexification method depends on the specific problem. In this paper, we focus on the convexification of the dynamic equation. The successive linearization is employed, which refers to the process of repeatedly linearizing a nonconvex function around a reference trajectory by Taylor expansion. Denote the reference trajectory of the k -th iteration as $\{\mathbf{x}^{(k)}, \mathbf{u}^{(k)}, \sigma^{(k)}\}$. For a first-order differential equation in the following form

$$\frac{d\mathbf{x}}{d\tau} = \sigma \mathbf{f}(\mathbf{x}, \mathbf{u}), \tag{14}$$

the right-hand side can be linearized by Taylor expansion. If the final time is fixed, σ is known and a constant value; then, the linearized differential equation can be expressed as

$$\begin{aligned}\left.\frac{d\mathbf{x}}{d\tau}\right|_{\left\{\mathbf{x}^{(k)},\mathbf{u}^{(k)},\sigma^{(k)}\right\}} &= \sigma \mathbf{f}(\mathbf{x}, \mathbf{u})\big|_{\left\{\mathbf{x}^{(k)},\mathbf{u}^{(k)},\sigma^{(k)}\right\}} \approx \sigma^{(k)} \mathbf{A}\left[\mathbf{x}^{(k)}, \mathbf{u}^{(k)}\right] \\ &\cdot \left(\mathbf{x} - \mathbf{x}^{(k)}\right) + \sigma^{(k)} \mathbf{B}\left[\mathbf{x}^{(k)}, \mathbf{u}^{(k)}\right] \\ &\cdot \left(\mathbf{u} - \mathbf{u}^{(k)}\right) + \sigma^{(k)} \mathbf{f}\left[\mathbf{x}^{(k)}, \mathbf{u}^{(k)}\right].\end{aligned} \tag{15}$$

When the final time is free, σ is an unknown variable. The dynamic equation can be approximated as

$$\begin{aligned}\left.\frac{d\mathbf{x}}{d\tau}\right|_{\left\{\mathbf{x}^{(k)},\mathbf{u}^{(k)},\sigma^{(k)}\right\}} &= \sigma \mathbf{f}(\mathbf{x}, \mathbf{u})\big|_{\left\{\mathbf{x}^{(k)},\mathbf{u}^{(k)},\sigma^{(k)}\right\}} \approx \sigma^{(k)} \mathbf{A}\left[\mathbf{x}^{(k)}, \mathbf{u}^{(k)}\right] \\ &\cdot \left(\mathbf{x} - \mathbf{x}^{(k)}\right) + \sigma^{(k)} \mathbf{B}\left[\mathbf{x}^{(k)}, \mathbf{u}^{(k)}\right]\left(\mathbf{u} - \mathbf{u}^{(k)}\right) + \sigma \mathbf{f}\left[\mathbf{x}^{(k)}, \mathbf{u}^{(k)}\right].\end{aligned} \tag{16}$$

In Equations (15) and (16),

$$\mathbf{A} = \mathbf{f}_{\mathbf{x}} = \frac{\partial}{\partial \mathbf{x}} \mathbf{f}[\mathbf{x}(\tau), \mathbf{u}(\tau), \tau] \in \mathbb{R}^{n_x \times n_x} \tag{17}$$

is the Jacobian matrix of $\mathbf{f}$ respect to $\mathbf{x}$, and

$$\mathbf{B} = \mathbf{f}_{\mathbf{u}} = \frac{\partial}{\partial \mathbf{u}} \mathbf{f}[\mathbf{x}(\tau), \mathbf{u}(\tau), \tau] \in \mathbb{R}^{n_x \times n_u} \tag{18}$$

is the Jacobian matrix of $\mathbf{f}$ respect to $\mathbf{u}$. Denote

$$\mathbf{R}\left[\mathbf{x}^{(k)}, \mathbf{u}^{(k)}, \sigma^{(k)}\right] = -\sigma^{(k)}\left[\mathbf{A}\mathbf{x}^{(k)} + \mathbf{B}\mathbf{u}^{(k)}\right]. \tag{19}$$

Equation (15) can be rewritten as

$$\frac{d}{d\tau}\mathbf{x} = \sigma^{(k)}\mathbf{A}\mathbf{x} + \sigma^{(k)}\mathbf{B}\mathbf{u} + \sigma^{(k)}\mathbf{f} + \mathbf{R}, \tag{20}$$

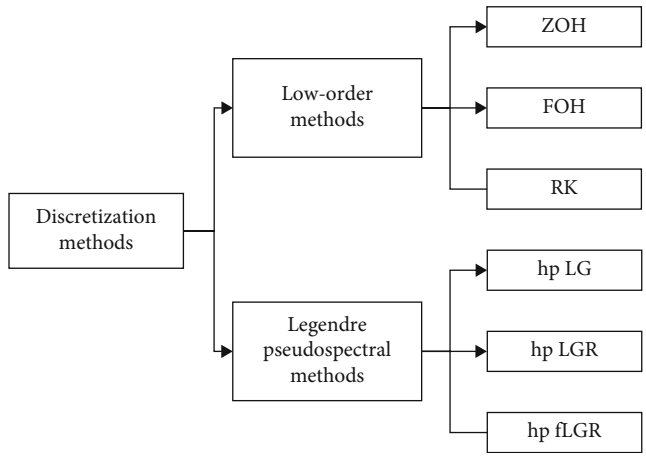

Figure 3: Discretization methods.

and Equation (16) is rewritten as

$$\frac{d}{d\tau}\mathbf{x} = \sigma^{(k)}\mathbf{A}\mathbf{x} + \sigma^{(k)}\mathbf{B}\mathbf{u} + \sigma\mathbf{f} + \mathbf{R}. \tag{21}$$

In this way, the right-hand side of the first-order differential equation is linearized. The only difference between Equation (20) and Equation (21) lies at the third term on the right-hand side. To keep it simple, only Equation (21) will be discussed, and the case for fixed-final-time phases can be obtained in a similar way.

3. Discretization Methods

The objective function, path constraints, boundary constraints, and linkage constraints are all algebraic functions, and the discrete form of these functions is the same as the original functions. However, the dynamic constraints are first-order differential functions and require to be approximated and transformed to algebraic functions.

To do this, PySCP implements six discretization methods, as shown in Figure 3. ZOH, FOH, and RK belong to low-order discretization methods.

Global Legendre pseudospectral methods benefit from two outstanding characteristics [58, 59]: (1) when the solution is smooth, the pseudospectral methods have quasi-exponential convergence when the number of discrete points is increased; (2) Runge phenomenon is avoided. However, if the solution is not smooth, the quasi-exponential convergence is invalid. In order to overcome this phenomenon, hp Legendre pseudospectral methods are adopted. Low-order methods only have quasi-linear convergence [60] when the number of discrete points is increased. If the solution accuracy is high, the required number of discrete points becomes very large. These two kinds of discretization methods have their own advantages and disadvantages. Users can select appropriate methods for their problems.

3.1. Legendre Pseudospectral Methods. Global Legendre pseudospectral methods use the roots of orthogonal Legendre polynomials as the collocation points and construct a

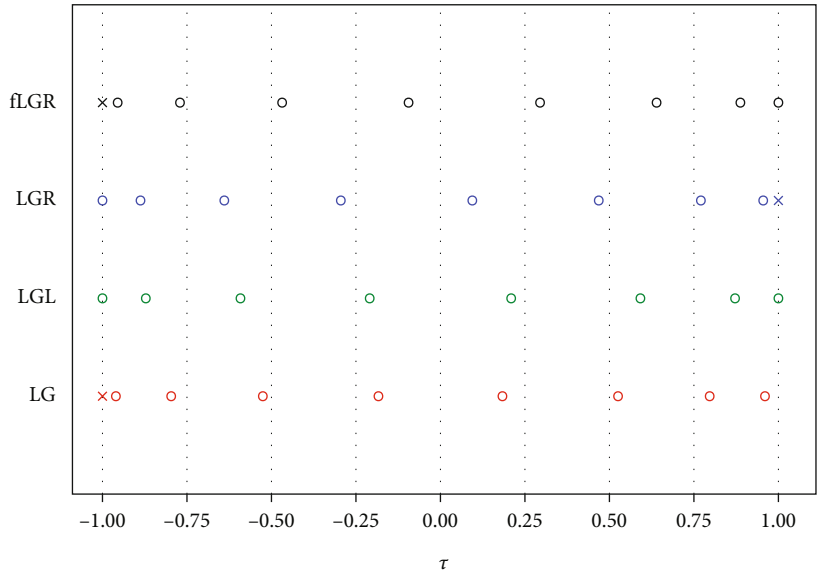

Figure 4: The distribution of collocation points and nodes in LG, LGL, LGR, and fLGR ($N = 8$).

set of Lagrangian polynomials to approximate the functions. Depending on the difference of collocation points, global Legendre pseudospectral methods can be divided into Legendre-Gauss (LG) pseudospectral method, Legendre-Gauss-Radau (LGR) pseudospectral method, and Legendre-Gauss-Lobatto (LGL) pseudospectral method.

The discretization points, also known as nodes, are the points where to approximate the variables. An illustration of the pseudospectral collocation points and nodes is shown in Figure 4 (the circles represent the collocation points, and the nodes include both the circles and the crosses). Denote the N-th-order Legendre polynomial as $P_N(\tau)$. The collocation points of the LG are the root of $P_N(\tau)$, which contains neither $\tau = -1$ nor $\tau = +1$. The LGR points are the root of $P_{N-1}(\tau) + P_N(\tau)$, which contains the left bound of the time interval $\tau = -1$ but does not contain $\tau = +1$. The LGL points are the root of $\dot{P}_{N-1}(\tau)$ together with $\tau = -1$ and $\tau = +1$. The LG and LGL points are symmetric about the origin whereas the LGR points are not. There is another version of LGR, which is called flipped LGR (fLGR). The fLGR points are obtained by flipping the LGR points with respect to the y axis.

The global Legendre pseudospectral methods approximate the functions on the time interval of $\tau \in [-1, 1]$ using global polynomials. They lose pseudoexponential convergence when the optimal control is not smooth. To overcome this problem, hp pseudospectral methods are implemented in PySCP. hp methods first divide the time interval into several subintervals and then approximate the function on the subintervals with local Lagrangian polynomials. "h" represents the number of the subintervals, and "p" represents the polynomial degree in the subinterval. The global pseudospectral methods can be considered as special cases of hp pseudospectral methods in the sense that $h = 1$.

Take hp fLGR as example. The time interval $\tau \in [-1, 1]$ is divided into $\mathscr{K}$ subintervals. Denote the h-th $(h = 1, \cdots, \mathscr{K})$ subinterval as $\mathcal{S}_h = [\tau_{h,l}, \tau_{h,r}]$ and the left bound and right bound of this subinterval as $\tau_{h,l}, \tau_{h,r} \in [-1, 1]$. Assume that there are N_h collocation points in this subinterval. A set of Lagrangian polynomials are constructed using the variables at the nodes

$$\ell_i(\tau) = \prod_{\substack{j=0 \\ j \neq i}}^{N_h} \frac{\tau - \tau_j}{\tau_i - \tau_j}, (i = 0, \cdots, N_h). \tag{22}$$

The functions in the subinterval $\mathcal{S}_h$ can be approximated as

$$\mathbf{x}(\tau) \approx \mathbf{X}(\tau) = \sum_{i=0}^{N_h} \mathbf{X}_{[h],i} \ell_{[h],i}(\tau), \tau \in \mathcal{S}_h, \tag{23}$$

where $\mathbf{X}_{[h]}$ and $\mathbf{U}_{[h]}$ are the state variables and control variables at the nodes in $\mathcal{S}_h$ and $\ell_i(\tau)$ have the following property

$$\ell_i(\tau_j) = \delta_{ij} = \begin{cases} 1, & i = j, \\ 0, & i \neq j. \end{cases} \tag{24}$$

δ_{ij} is the Kronecker Delta function. Differentiating Equation (23), we have

$$\frac{d}{d\tau}\mathbf{x}(\tau_j) \approx \sum_{i=0}^{N_h} \mathbf{X}_i \dot{\ell}_i(\tau_j) = \sum_{i=0}^{N_h} \mathbf{D}_{[N_h],ji} \mathbf{X}_i, (j = 1, \cdots, N_h), \tag{25}$$

where $\mathbf{D}_{[N_h]} \in \mathbb{R}^{N_h \times (N_h+1)}$ is the pseudospectral differential matrix and $\mathbf{D}_{[N_h],ji} = \dot{\ell}_{[h],i}(\tau_{[h],j})$. Thus, the dynamic equation

can be discretized by the following form:

$$\mathrm{D}_{[N_h]}\mathbf{X}_{[h]} = \sigma^{(k)}\mathbf{A}\mathbf{X}_{[h]} + \sigma^{(k)}\mathbf{B}\mathbf{U}_{[h]} + \sigma\mathbf{f}\left(\mathbf{X}_{[h]}, \mathbf{U}_{[h]}, \sigma^{(k)}\right) + \mathbf{R}\left(\mathbf{X}_{[h]}, \mathbf{U}_{[h]}, \sigma^{(k)}\right). \tag{26}$$

Denote the state variables in all $\mathscr{H}$ subintervals as follows:

$$\mathbf{X} = \left[\mathbf{X}_{[1]}, \mathbf{X}_{[2]}, \cdots, \mathbf{X}_{[\mathscr{H}]}\right]^T. \tag{27}$$

The dynamic equation can be approximated by

$$\mathbf{D}\mathbf{X} = \sigma^{(k)}\mathbf{A}\mathbf{X} + \sigma^{(k)}\mathbf{B}\mathbf{U} + \sigma\mathbf{f}\left(\mathbf{X}, \mathbf{U}, \sigma^{(k)}\right) + \mathbf{R}\left(\mathbf{X}, \mathbf{U}, \sigma^{(k)}\right). \tag{28}$$

This is called the pseudospectral differential form. Corresponding to this form, the dynamic equation can also be approximated in another form called the pseudospectral integral form. In the h-th subinterval, the state variable is approximated by

$$\mathbf{X}_{[h]} = \mathbf{X}_{[h],0} + \sum_{i=0}^{N_h} \mathbf{I}_{[N_h],ji}\left\{\sigma^{(k)}\mathbf{A}\mathbf{X}_{[h],i} + \sigma^{(k)}\mathbf{B}\mathbf{U}_{[h],i} + \sigma\mathbf{f}\left(\mathbf{X}_{[h],i}, \mathbf{U}_{[h],i}, \sigma^{(k)}\right) + \mathbf{R}\left(\mathbf{X}_{[h],i}, \mathbf{U}_{[h],i}, \sigma^{(k)}\right)\right\}, \tag{29}$$

where $\mathbf{I}_{[N_h]}$ is $N_h \times (N_h + 1)$ matrix and $\mathbf{I}_{[N_h],ji} = \int_{\tau_{h,l}}^{\tau_{[h],j}} \ell_{[h],i}(\xi)\, \mathrm{d}\xi$. Combing all subintervals, the integral form can be expressed as

$$\mathbf{X} = \mathbf{X0} + \sum_{h=1}^{\mathscr{H}} \sum_{i=0}^{N_h} \mathbf{I}_{[N_h],ji}\left\{\sigma^{(k)}\mathbf{A}\mathbf{X}_{[h],i} + \sigma^{(k)}\mathbf{B}\mathbf{U}_{[h],i} + \sigma\mathbf{f}\left(\mathbf{X}_{[h],i}, \mathbf{U}_{[h],i}, \sigma^{(k)}\right) + \mathbf{R}\left(\mathbf{X}_{[h],i}, \mathbf{U}_{[h],i}, \sigma^{(k)}\right)\right\}. \tag{30}$$

X0 is a vector containing the state variables on the left bound of all subintervals. The differential form and integral form of hp LG pseudospectral method and hp LGR are similar. It should be noted that the differential form and the integral form are equivalent [61]. However, this is not the case for hp LGL. Therefore, hp LGL is not implemented in PySCP.

Apart from the dynamic equation, the integral term in the objective is approximated by

$$\int_{-1}^{1} \mathbf{g}(\mathbf{x}, \mathbf{u}, \tau)\mathrm{d}\tau = \sum_{i=1}^{N} w_i \mathbf{g}(\mathbf{X}, \mathbf{U}). \tag{31}$$

w_i is the integral weights of the pseudospectral methods. We refer the readers to [62] for the computation of the differential matrix $\mathbf{D}$, the integral matrix $\mathbf{I}$, and the integral weights w_i.

3.2. Low-Order Discretization Methods. The convexified dynamic equation

$$\frac{d}{d\tau}\mathbf{x} = \sigma^{(k)}\mathbf{A}(\tau)\mathbf{x} + \sigma^{(k)}\mathbf{B}(\tau)\mathbf{u} + \sigma\mathbf{f}(\tau) + \mathbf{R}(\tau) \tag{32}$$

can be considered as a continuous-time linear control system. In control theory, this control system can be discretized using ZOH, FOH, and RK methods. Assume that the normalized time horizon is discretized into N subintervals by $N + 1$ discretization points, which can also be called nodes. The subintervals can be either uniformly distributed or nonuniformly distributed.

According to the control theory [63], the state transition of Equation (32) over an subinterval can be expressed as

$$\mathbf{x}(\tau) = \Psi(\tau, \tau_i)\mathbf{x}(\tau_i) + \int_{\tau_i}^{\tau} \Psi(\tau, \xi)\mathbf{B}\mathbf{u}d\xi + \sigma\int_{\tau_i}^{\tau} \Psi(\tau, \xi)\mathbf{f}d\xi + \int_{\tau_i}^{\tau} \Psi(\tau, \xi)\mathbf{R}d\xi, \tag{33}$$

where $\Psi(\tau, \tau_i)$ is the state transition matrix which has the following properties:

(1) $\Psi(\tau, \tau_i)$ is first-order continuous

(2) Nonsingular and invertible: $\Psi(\tau, \tau_i) = \Psi^{-1}(\tau_i, \tau)$ and $\Psi(\tau, \tau_i)\Psi(\tau_i, \tau) = I$, where I is the unit matrix

(3) For all τ, $\Psi(\tau, \tau) = I$

(4) Ψ is the unique solution to the linear matrix ordinary differential equation

$$\frac{d}{d\tau}\Psi(\tau, \tau_k) = \sigma^{(k)}\mathbf{A}(\tau)\mathbf{x} + \sigma^{(k)}\mathbf{B}(\tau)\mathbf{u} + \sigma\mathbf{f}(\tau) + \mathbf{R}(\tau). \tag{34}$$

ZOH assumes that the control variables keep constant during each subinterval and are always equal to the control variable on the left bound of each subinterval, i.e.,

$$u(\tau) = u(\tau_i), \tau \in [\tau_i, \tau_{i+1}). \tag{35}$$

FOH assumes that the control variables are linearly interpolated by the control variables on the left bound and right bound of each subinterval, i.e.,

$$u(\tau) = \frac{-\tau + \tau_{k+1}}{\tau_{k+1} - \tau_k}u(\tau_k) + \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k}u(\tau_{k+1}), \tau \in [\tau_k, \tau_{k+1}). \tag{36}$$

The illustration of ZOH and FOH is shown in Figure 5.

In the ZOH method, the state variables $\mathbf{X}_{i+1}$ at $\tau = \tau_{i+1}$ can be approximated by

$$\mathbf{X}_{i+1} = \widehat{\mathbf{A}}_i\mathbf{X}_i + \widehat{\mathbf{B}}_i\mathbf{U}_i + \sigma\widehat{\mathbf{f}}_i + \widehat{\mathbf{R}}_i, \tag{37}$$

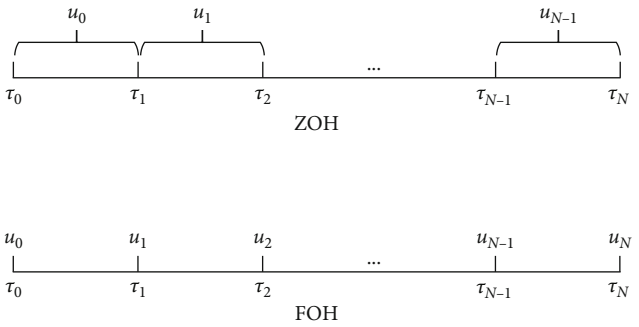

Figure 5: Illustration of ZOH and FOH.

where the matrices are given by

$$\begin{aligned}
\widehat{\mathbf{A}}_i &= \Psi(\tau_{i+1}, \tau_i),\\
\widehat{\mathbf{B}}_i &= \Psi(\tau_{i+1}, \tau_i)\int_{\tau_i}^{\tau_{i+1}} \Psi^{-1}(\xi, \tau_i)\sigma^{(k)}\mathbf{B}\left[\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \xi\right] d\xi,\\
\widehat{\mathbf{f}} &= \Psi(\tau_{i+1}, \tau_k)\int_{\tau_i}^{\tau_{i+1}} \Psi^{-1}(\xi, \tau_i)\mathbf{f}\left[\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \xi\right] d\xi,\\
\widehat{\mathbf{R}} &= \Psi(\tau_{i+1}, \tau_k)\int_{\tau_i}^{\tau_{i+1}} \Psi^{-1}(\xi, \tau_i)\mathbf{R}\left[\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \xi\right] d\xi.
\end{aligned} \tag{38}$$

Thus, the state variables on adjacent points are associated with each other by an algebraic equation. In the FOH method, the state variables $\mathbf{X}_{i+1}$ at $\tau = \tau_{i+1}$ can be associated with $\mathbf{X}_i$ by the following equation:

$$\mathbf{X}_{i+1} = \widehat{\mathbf{A}}_i\mathbf{X}_i + \widehat{\mathbf{B}}_{i-}\mathbf{U}_i + \widehat{\mathbf{B}}_{i+}\mathbf{U}_{i+1} + \sigma\widehat{\mathbf{f}}_i + \widehat{\mathbf{R}}_i, \tag{39}$$

where the matrices are given by

$$\begin{aligned}
\widehat{\mathbf{A}}_i &= \Psi(\tau_{i+1}, \tau_i),\\
\widehat{\mathbf{B}}_{i-} &= \Psi(\tau_{i+1}, \tau_i)\int_{\tau i}^{\tau_{i+1}} \Psi^{-1}(\xi, \tau_i)\sigma^{(k)}\mathbf{B}(\mathbf{x}, \mathbf{u}, \xi)\lambda_- d\xi,\\
\widehat{\mathbf{B}}_{i+} &= \Psi(\tau_{i+1}, \tau_i)\int_{\tau_i}^{\tau_{i+1}} \Psi^{-1}(\xi, \tau_i)\sigma^{(k)}\mathbf{B}(\mathrm{x}, \mathrm{u}, \xi)\lambda_+ \mathrm{d}\xi,\\
\widehat{\mathbf{f}} &= \Psi(\tau_{i+1}, \tau_k)\int_{\tau_i}^{\tau_{i+1}} \Psi^{-1}(\xi, \tau_i)\mathbf{f}(\mathbf{x}, \mathbf{u}, \xi) d\xi,\\
\widehat{\mathbf{R}} &= \Phi(\tau_{i+1}, \tau_k)\int_{\tau_i}^{\tau_{i+1}} \Phi^{-1}(\xi, \tau_i)\mathbf{R}(\mathbf{x}, \mathbf{u}, \xi) d\xi.
\end{aligned} \tag{40}$$

In the RK method, the transcribed dynamic equation is given as follows:

$$\mathbf{X}_{i+1} = \mathbf{X}_i + \frac{\Delta\tau}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4), \tag{41}$$

Table 1: Simulation parameters.

Parameter	Value
ω_v	1×10^3
ω_f	1×10^2
ρ_1	0.1
ρ_2	0.2
ρ_3	0.75
c_1	0.5
c_2	3.2

where

$$\begin{aligned}
\mathbf{k}_1 &= \sigma^{(k)}\mathbf{A}_i\mathbf{X}_i + \sigma^{(k)}\mathbf{B}_i\mathbf{U}_i + \mathbf{f}_i\sigma + \mathbf{R}_i,\\
\mathbf{k}_2 &= \sigma^{(k)}\mathbf{A}_{i+1/2}\left(\mathbf{X}_i + \frac{\Delta\tau}{2}\mathbf{k}_1\right) + \mathbf{B}_{i+1/2}\mathbf{U}_{i+1/2} + \mathbf{f}_{i+1/2}\sigma + \mathbf{R}_{i+1/2},\\
\mathbf{k}_3 &= \sigma^{(k)}\mathbf{A}_{i+1/2}\left(\mathbf{X}_i + \frac{\Delta\tau}{2}\mathbf{k}_2\right) + \mathbf{B}_{i+1/2}\mathbf{U}_{i+1/2} + \mathbf{f}_{i+1/2}\sigma + \mathbf{R}_{i+1/2},\\
\mathbf{k}_4 &= \sigma^{(k)}\mathbf{A}_{i+1}(\mathbf{X}_i + \Delta\tau\mathbf{k}_3) + \mathbf{B}_{i+1}\mathbf{U}_{i+1} + \mathbf{f}_{i+1}\sigma + \mathbf{R}_{i+1}.
\end{aligned} \tag{42}$$

The subscript $i + 1/2$ indicates the value is evaluated at the middle point of $\tau = \tau_i$ and $\tau = \tau_{i+1}$. Subscribing Equation (42) into Equation (41), the latter equation can be reformatted into the same form as that of Equation (39).

4. Adaptive Trust Region Method

After convexification and discretization, the multiple-phase optimal control problem is transcribed to a SOCP problem around a reference trajectory $\{\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \sigma^{(k)}\}$, which is summarized as follows:

Problem P2. Minimize the objective

$$L = \sum_{p=1}^{P}\left\{\Phi^{(p)}[\mathbf{X}_0, \mathbf{X}_N] + \sigma\sum_{i=1}^{N}\mathbf{w}_i^{(p)}\mathbf{g}^{(p)}\left[\mathbf{X}_i^{(p)}, \mathbf{U}_i^{(p)}, t\right]\right\}, \tag{43}$$

subject to the transcribed dynamic constraints (Equation (28), Equation (30), Equation (37), or Equation (39) depending on the discretization method), path constraints

$$\mathbf{C}[\mathbf{X}_i, \mathbf{U}_i, \tau_i] \le \mathbf{0}, \tag{44}$$

the boundary constraints

$$\varphi[\mathbf{X}_i] \le \mathbf{0}, \tag{45}$$

the linkage constraints

$$\mathscr{L}_{p_{l,a}}^{p_{l,b}}\left[\mathbf{X}_N^{\left(p_{l,a}\right)}, \mathbf{X}_0^{\left(p_{l,b}\right)}\right] \le \mathbf{0}, \tag{46}$$

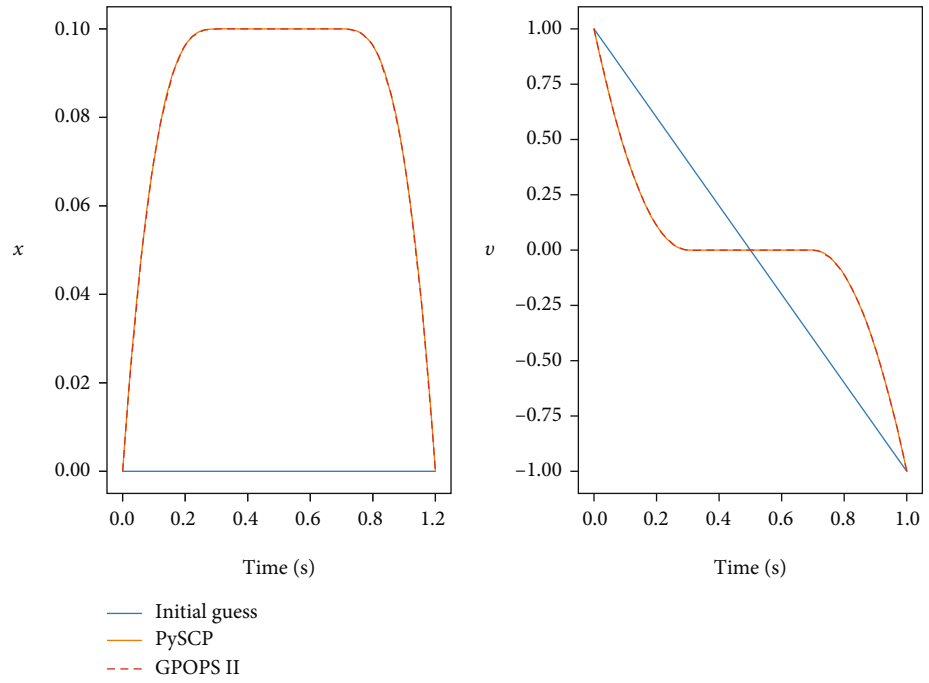

Figure 6: Optimal states of the Breakwell problem.

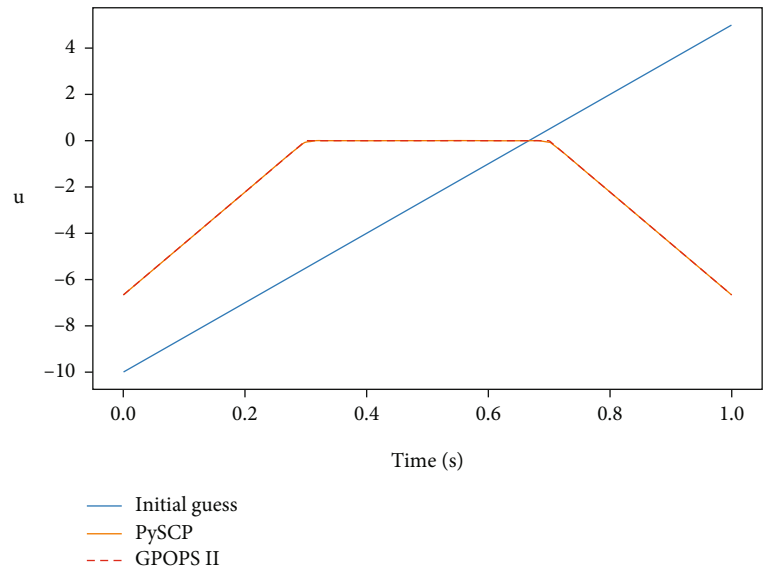

Figure 7: Optimal control of the Breakwell problem.

and the trust region constraints

$$\begin{aligned}\left|\mathbf{X}-\mathbf{X}^{(k)}\right| &\le \boldsymbol{\delta}_{\mathbf{x}},\\ \left|\mathbf{U}-\mathbf{U}^{(k)}\right| &\le \boldsymbol{\delta}_{\mathbf{u}},\\ \left|\sigma-\sigma^{(k)}\right| &\le \delta_{\sigma}.\end{aligned} \tag{47}$$

The dynamic constraints are equality functions, and the feasible space is very small in most situations, or even null, especially in the first several iterations. This phenomenon is usually called artificial infeasibility. To avoid the situation that no solution exists, a slack variable $\mathbf{v}_f \in \mathbb{R}_+^{n_x \times n_{cp}}$ is introduced. In pseudospectral methods, n_{cp} is the number of collocation points. For low-order discretization methods, n_{cp} is the number of the subintervals. Take Equation (37) as an example, the dynamic constraints are converted to an inequality function

$$\left|\mathbf{X}_{i+1}-\left(\widehat{\mathbf{A}}_i\mathbf{X}_i+\widehat{\mathbf{B}}_i\mathbf{U}_i+\sigma\mathbf{f}_i+\widehat{\mathbf{R}}_i\right)\right| \le \mathbf{v}_f. \tag{48}$$

Table 2: Comparison of accuracy and CPU times.

Method	Nodes	h	p	CPU time, ms	Objective	Objective relative error	Max relative error, x_1	Max relative error, x_2
ZOH	10	-	-	7.59	4.57452	2.93×10^{-2}	9.46×10^{-4}	7.73×10^{-2}
	20	-	-	16.87	4.47573	7.03×10^{-3}	4.04×10^{-4}	3.23×10^{-2}
	50	-	-	52.99	4.44945	1.11×10^{-3}	2.07×10^{-4}	1.64×10^{-2}
	100	-	-	144.31	4.44578	2.78×10^{-4}	1.04×10^{-4}	8.26×10^{-2}
FOH	10	-	-	2.49	4.66915	5.06×10^{-2}	1.24×10^{-7}	6.55×10^{-8}
	20	-	-	7.85	4.50242	1.30×10^{-3}	1.98×10^{-7}	4.30×10^{-8}
	50	-	-	29.69	4.45408	2.16×10^{-3}	1.89×10^{-7}	7.14×10^{-8}
	100	-	-	61.58	4.44694	5.46×10^{-4}	1.71×10^{-7}	7.88×10^{-8}
RK	10	-	-	8.01	4.66912	5.06×10^{-2}	3.03×10^{-8}	1.20×10^{-7}
	20	-	-	17.52	4.50255	1.30×10^{-2}	6.34×10^{-8}	1.38×10^{-8}
	50	-	-	57.71	4.45403	2.16×10^{-3}	1.00×10^{-7}	6.69×10^{-8}
	100	-	-	129.11	4.44692	5.47×10^{-4}	7.08×10^{-8}	7.71×10^{-8}
hp LG	20	1	18	36.5	4.44068	8.47×10^{-4}	8.91×10^{-3}	1.80×10^{-2}
	42	1	40	135.1	4.44428	3.60×10^{-5}	2.34×10^{-3}	4.76×10^{-3}
	56	5	10	96.8	4.44393	1.15×10^{-4}	1.61×10^{-3}	3.19×10^{-3}
	106	5	20	274.8	4.44437	1.58×10^{-5}	4.31×10^{-4}	8.68×10^{-4}
hp LGR	21	1	20	33.4	4.44067	8.50×10^{-4}	9.61×10^{-3}	1.81×10^{-2}
	41	1	40	119.0	4.44404	9.06×10^{-5}	2.38×10^{-3}	4.94×10^{-3}
	51	5	10	84.9	4.44406	8.62×10^{-5}	1.67×10^{-3}	3.43×10^{-3}
	101	5	20	252.4	4.44439	1.19×10^{-5}	4.62×10^{-4}	9.46×10^{-4}
hp fLGR	21	1	20	32.0	4.44067	8.50×10^{-4}	9.53×10^{-3}	1.86×10^{-2}
	41	1	40	127.1	4.44404	9.06×10^{-5}	2.40×10^{-3}	4.66×10^{-3}
	51	5	10	84.2	4.44406	8.62×10^{-5}	1.72×10^{-3}	3.49×10^{-3}
	101	5	20	247.7	4.44439	1.19×10^{-5}	4.59×10^{-4}	9.06×10^{-4}

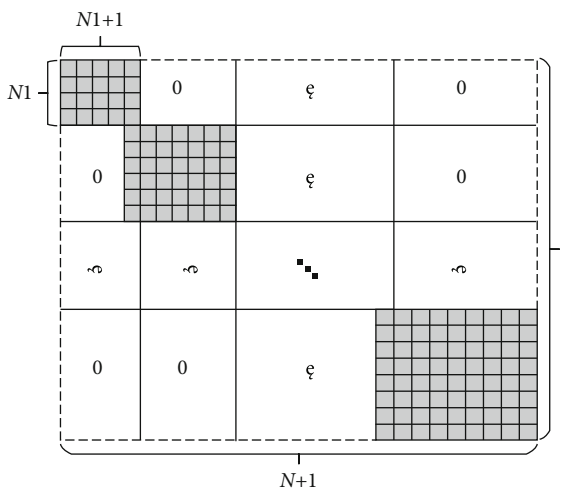

Figure 8: The sparsity of the pseudospectral differential matrix for hp fLGR method.

The slack variable enlarges the feasible space of the dynamic constraints. The norm of $\mathbf{v}_f$ represents how much the dynamic constraints are violated; therefore, $\|\mathbf{v}_f\|$ should be as small as possible to make sure that the solution to the SOCP problem is also the solution to the original OCP. In addition to the dynamic constraints, a slack variable $\mathbf{v}_c \in \mathbb{R}_+^{n_c \times n_{cp}}$ for the path constraints is also introduced. The path constraints are transformed to the following inequality function:

$$\mathtt{C}[\mathbf{X}, \mathbf{U}] \leq \mathbf{v}_c. \tag{49}$$

To ensure that two slack variables approach to zero at convergence, they are penalized in the objective function. Construct an augmented objective function of the original optimal control problem, defined as follows:

$$\widehat{J} = J + \omega_f \left\| \mathbf{v}_f \right\| + \omega_c \|\mathbf{v}_c\|. \tag{50}$$

The penalty on the slack variables is also introduced onto

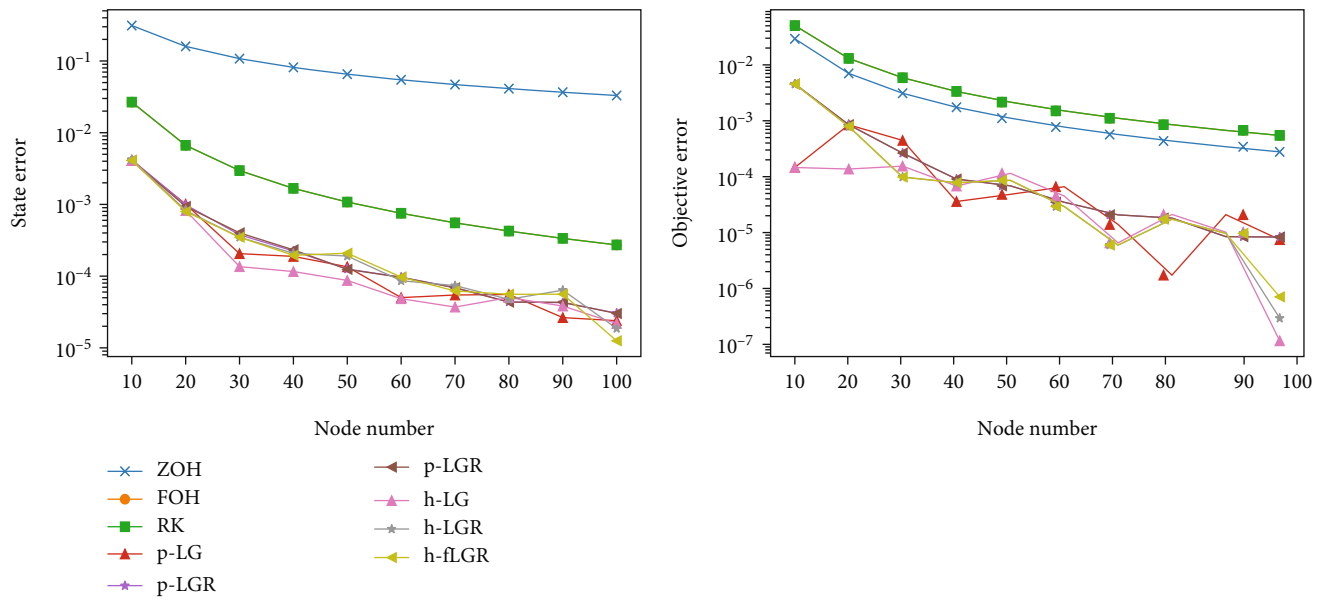

Figure 9: Accuracy with the increase of node number.

the objective function of Problem 2, as follows:

$$\widehat{L}\left[\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \sigma^{(k)}\right] = L + \omega_f \left\|\mathbf{v}_f\right\| + \omega_c \|\mathbf{v}_c\|, \tag{51}$$

where ω_f and ω_c are the penalty factors on the slack variables.

Define a ratio parameter

$$\rho^{(k)} = \frac{\widehat{J}\left(\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \sigma^{(k)}\right) - \widehat{J}\left(\mathbf{X}^{(k-1)}, \mathbf{U}^{(k-1)}, \sigma^{(k-1)}\right)}{\widehat{J}\left(\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \sigma^{(k)}\right) - \widehat{L}\left(\mathbf{X}^{(k)}, \mathbf{U}^{(k)}, \sigma^{(k)}\right)}, \tag{52}$$

to measure the accuracy of approximation. The numerator is the improvement in the performance index of the original optimal control problem, and the denominator is the improvement in the performance index of the SOCP problem. If $\rho^{(k)}$ is close to 1, the approximation is very accurate, and the trust region can be enlarged. If $\rho^{(k)} \ll 1$, the approximation is poor, and the trust region must be scaled down.

Based on the above analysis, define the following parameters: $0 < \rho_1 < \rho_2 < 1$ and $c_1 < 1 < c_2$. If $\rho^{(k)} < \rho_1$, then adjust the trust region by $\boldsymbol{\delta}^{(k)} = c_1 \boldsymbol{\delta}^{(k-1)}$. If $\rho_1 \le \rho^{(k)} < \rho_2$, then the trust region keeps unchanged, $\boldsymbol{\delta}^{(k)} = \boldsymbol{\delta}^{(k-1)}$. Otherwise. if $\rho^{(k)} \ge \rho_2$, enlarge the trust region by $\boldsymbol{\delta}^{(k)} = c_2 \boldsymbol{\delta}^{(k-1)}$. The PySCP algorithm is terminated when the nonlinear cost reduction $\Delta\widehat{J}$ goes below a tolerance 1×10^{-4}.

5. Examples

In this section, PySCP is demonstrated on three examples. The first example is the Breakwell problem and demonstrates the ability of PySCP to solve single-phase smooth OCPs with fixed final time. The second example is the lunar landing problem to show its ability to cope with nonsmooth free-final-time problems. The third example is the ascent trajectory optimization of two-stage-to-orbit launch vehicle, which is a nonsmooth multiple-phase optimal control problem.

The simulation parameters in all the following examples are listed in Table 1, including the penalty factors and parameters for the adaptive trust region method. Suitable values of these parameters here are selected according to a trial-and-error process. The virtual control weights are typically 1-3 orders of magnitude larger to ensure that the corresponding terms in the right-hand side of Equation (50) go to 0.

5.1. Breakwell Problem. The Breakwell problem [60] is a single-phase fixed-final-time optimal control problem whose optimum control variable is smooth. The objective is given by

$$J = \frac{1}{2}\int_0^1 u^2 dt, \tag{53}$$

subject to a convex dynamic equation

$$\dot{x} = v, \dot{v} = u, \tag{54}$$

fixed boundary constraints

$$x(0) = 0, x(1) = 1, v(0) = 1, v(1) = -1, \tag{55}$$

and state constraints

$$x(t) \le l = 0.1. \tag{56}$$

In this problem, all functions and constraints are convex. PySCP achieves the solution on the first iteration because no approximation is made when transcribing the problem to

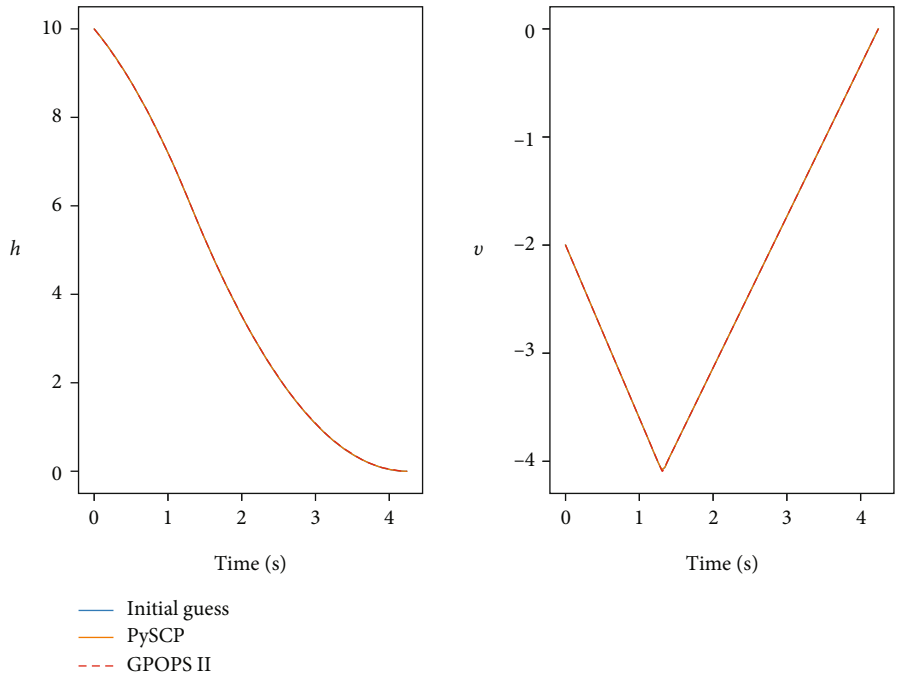

Figure 10: The optimal states of the lunar landing problem.

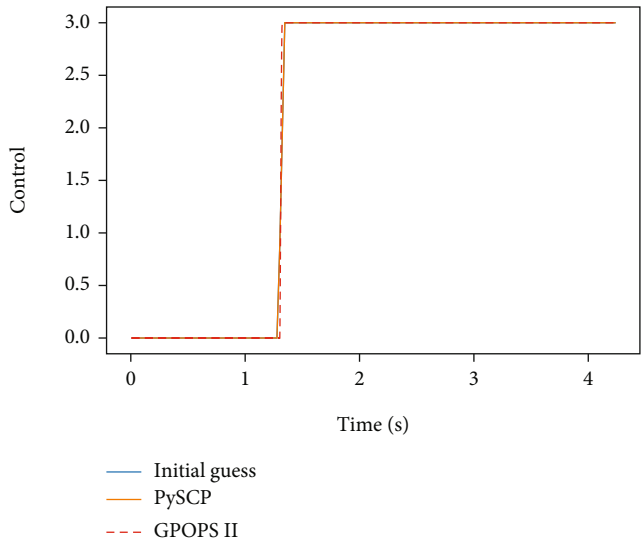

Figure 11: The optimal control variable of the lunar landing problem.

the SOCP problem. The analytical optimal objective is $J = 4/(9l) \approx 0.44444$.

The optimal state variables are shown in Figure 6, and the optimal control variable is shown in Figure 7. The discretization method is FOH, and the subinterval number is 40. All other discretization methods lead to the same optimal solution. The straight lines are the initial guess, which are far away from the final solution.

To assess the precision of the implemented PySCP method, the state variables are integrated by open-loop propagation given the computed control signals. The integrated state vector is denoted as $\widehat{\mathbf{X}}$. The open-loop propagation is implemented and integrated in PySCP, which is automatically called after iteration stops. The relative error between the integrated state and the computed state is defined as

$$\mathbf{e}_j(\tau_i) = \frac{\left|\widehat{\mathbf{X}}_j(\tau_i) - \mathbf{X}_j(\tau_i)\right|}{1 + \max\left|\mathbf{X}_j(\tau_i)\right|}, i = 1, \cdots, N, j = 1, \cdots, n_x. \quad (57)$$

The max relative error of each state variable is

$$e_{\max,j} = \max_{i=1,\cdots,N} \mathbf{e}_j(\tau_i), j = 1, \cdots, n_x. \quad (58)$$

Multiple discretization methods are implemented in PySCP. All these methods are employed, and the CPU times and max relative errors for different methods are listed in Table 2.

The CPU times for the low-order methods become larger when the node number grows. However, this is not the case for the pseudospectral methods. For instance, hp LG with 42 nodes is faster than that with 56 nodes. This is mainly because of the structure of the differential matrix $\mathbf{D}$ and integral matrix $\mathbf{I}$, as illustrated in Figure 8. The nonzero elements only exist in the diagonal blocks, and the number of nonzero elements in each block is proportional to the square of the polynomial degrees in corresponding interval. Higher polynomial degree leads to a longer solution time.

Optimal objective can be obtained by all methods, but the pseudospectral methods outperform the low-order methods. ZOH has a much worse accuracy and objective value than other methods. FOH and RK have a very small relative error compared to other methods, which is probably

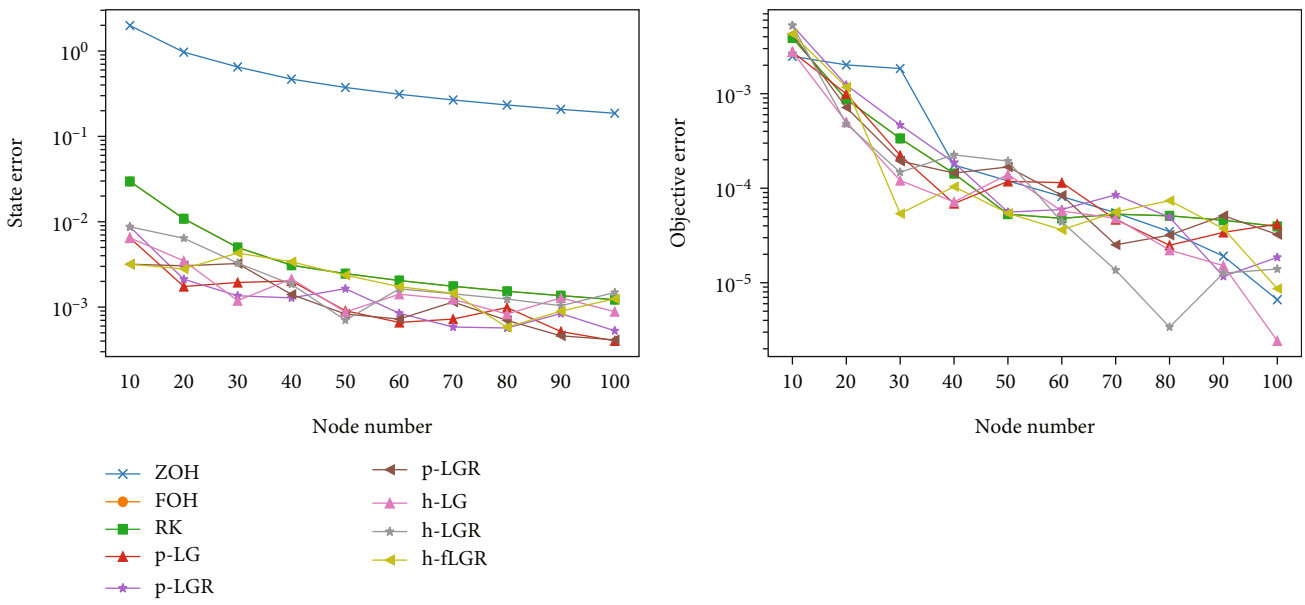

Figure 12: Accuracy with the increase of node number.

due to the way that FOH and RK discrete the dynamic functions. They approximate the dynamic functions by integrating from one node to the next, similar to open-loop propagation. However, they are characterized with larger objective error. The pseudospectral methods are better in obtaining the optimal objective. Even low degree can lead to an optimal objective close to the analytical value.

The state relative error and objective error are plotted in Figure 9. The hp pseudospectral methods are implemented using both h-scheme and p-scheme. The character "h" indicates that a fixed low-degree polynomial (in this case, $p = 10$) is used in each interval, and the number of subintervals $\mathscr{K}$ is changed, while "p" indicates that $\mathscr{K}$ is fixed to be 1, and the polynomial degree is changed.

This is an example problem whose state and control variables are all smooth. The pseudospectral methods outperform the low-order methods by 1-2 orders of magnitude. h-scheme and p-scheme have a similar behaviour. They both have a quasi-exponential convergence behaviour as stated before when solving smooth problems.

5.2. Lunar Landing Problem. The lunar landing problem [64] is a single-phase free-final-time bang-bang control problem. The objective is

$$J = \int_{t_0}^{t_f} u dt. \tag{59}$$

The state variables are the flight altitude h and velocity v, subject to convex dynamic equations

$$\begin{aligned} \dot{h} &= v, \\ \dot{v} &= -g + u, \end{aligned} \tag{60}$$

where $g = 1.6\text{m/s}^2$ is the gravitational acceleration at the Moon surface. The boundary condition is given by

$$\begin{aligned} h(t_0) = 10, v(t_0) = -2, \\ h(t_f) = 0, v(t_f) = 0. \end{aligned} \tag{61}$$

The control variable is bounded by $u \in [0, 3]$. After normalization, the objective function is given by

$$J = \int_{-1}^{1} \sigma u dt, \tag{62}$$

which is not convex. Therefore, successive linearization is employed to approximate the objective function.

$$J = \sum_{i=0}^{N} \left(\sigma^{(k)} w_i \mathbf{U}_i + \sigma w_i \mathbf{U}_i^{(k)} - \sigma^{(k)} w_i \mathbf{U}_i^{(k)} \right). \tag{63}$$

The optimal states are shown in Figure 10, and the optimal control is shown in Figure 11. The analytical optimal objective is 8.7831. The plotted curve by PySCP is calculated using the hp fLGR method with three subintervals, each with ten collocation points.

The state relative error and objective error for this nonsmooth problem are plotted in Figure 12. Different from the previous smooth problem, the pseudospectral methods have a similar behaviour with that of FOH and RK when the node number is increased. This is mainly due to the noncontinuity in the control variable. The breaking point in control variable makes the pseudospectral methods loss its convergence superiority over the low-order methods. An adaptive mesh refinement might help to mitigate this phenomenon, and faster convergence might be obtained for the pseudospectral methods.

5.3. Ascent Trajectory Optimization of a Two-Stage-to-Orbit Launch Vehicle. This problem is a multiple-phase optimal control problem. The state can be described by four

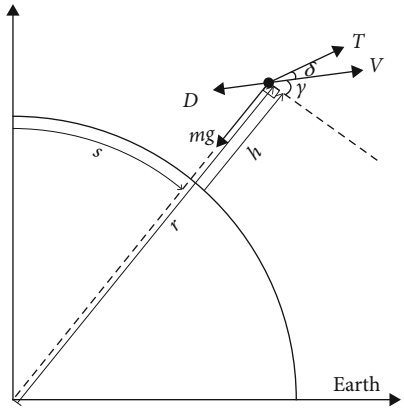

Figure 13: Flight dynamics of the launch vehicle.

Table 3: Data of Falcon 9 full thrust rocket.

Stage	First	Second
Dry mass (t)	22.2	4.0
Propellant mass (t)	410.9	107.5
Vacuum thrust (kN)	8227	934
Specific impulse (s)	300	350
Nozzle exit area (m^2)	11.039	N/A

variables: the altitude h, downrange s, velocity v, and horizontal flight path angle γ, as depicted in Figure 13.

The objective is to maximize the payload mass, equivalent to the minimization of the opposite value of the final mass

$$J = -m_f, \tag{64}$$

subject to the dynamic equation

$$\begin{aligned} \dot{h} &= v \sin\gamma, \\ \dot{d} &= v \cos\gamma \frac{R_e}{h + R_e}, \\ \dot{v} &= \frac{T_v \cos\delta}{m} - D - \frac{\mu}{(h + R_e)^2} \sin\gamma, \\ \dot{\gamma} &= \frac{T_\gamma \sin\delta}{mv} + \left(\frac{v}{h + R_e} - \frac{\mu}{(h + R_e)^2 v}\right) \cos\gamma, \\ \dot{m} &= -\frac{T}{g_0 I_{sp}}. \end{aligned} \tag{65}$$

The control variables are T_v, T_γ, and thrust magnitude T . They satisfy the following equation:

$$T_v^2 + T_\gamma^2 = T^2, \tag{66}$$

which is nonconvex. This constraint is convexified by simply changing the equality into inequality

Table 4: Boundary conditions of the flight mission.

State variables	Initial condition	Final condition
Altitude (km)	0	500
Downrange (km)	0	[2000, 5000]
Velocity (km)	50	7905
Flight path angle (deg)	[85, 90]	0

$$T_v^2 + T_\gamma^2 \le T^2. \tag{67}$$

In the optimal solution, the equality always holds. The aerodynamic drag acceleration$D$ is calculated by

$$D = \frac{\rho V^2 C_D S_{\text{ref}}}{2m}, \tag{68}$$

where S_{ref} is the reference area and ρ is calculated by the exponential atmospheric model

$$\rho = \rho_0 \exp\left(-\frac{h}{H_0}\right). \tag{69}$$

$\rho_0 = 1.225\text{kg/m}^3$ is the sea level atmosphere density, and $H_0 = 7200\text{m}$ is the reference altitude. By successive linearization, the following are determined as follows:

$$\mathbf{A}(\mathbf{x}) = \begin{bmatrix} 0 & 0 & \sin\gamma & v\cos\gamma & 0 \\ a_{21} & 0 & a_{23} & a_{24} & 0 \\ a_{31} & 0 & 0 & a_{34} & \frac{D}{m^2} \\ a_{41} & 0 & a_{43} & a_{44} & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & \frac{1}{v} & 0 \\ 0 & 0 & -\frac{1}{g_0 I_{sp}} \end{bmatrix}, \tag{70}$$

with

$$\begin{aligned} a_{21} &= -v\cos\gamma \frac{R_e}{r^2}, \\ a_{23} &= \cos\gamma \frac{R_e}{r}, \\ a_{24} &= -v\sin\gamma \frac{R_e}{r}, \\ a_{31} &= \frac{\mu}{r^3} \sin\gamma, \\ a_{34} &= -\frac{\mu}{r^2} \cos\gamma \\ a_{41} &= \left(-\frac{v}{r^2} + \frac{2\mu}{r^3 v}\right) \cos\gamma, \\ a_{43} &= \left(\frac{1}{r} + \frac{\mu}{r^2 v^2}\right) \cos\gamma, \\ a_{44} &= -\left(\frac{v}{r} - \frac{\mu}{r^2 v}\right) \sin\gamma. \end{aligned} \tag{71}$$

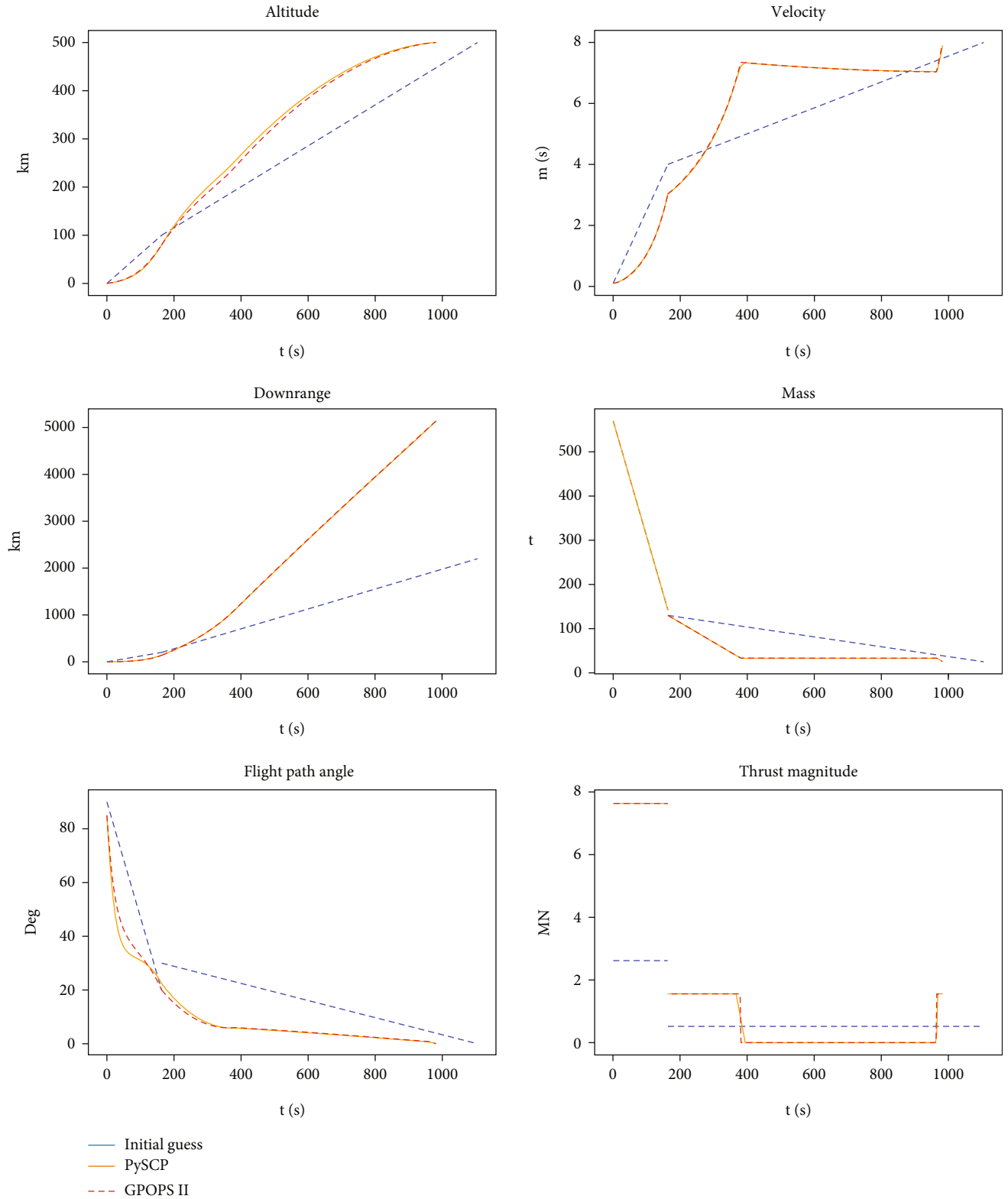

Figure 14: The optimal ascent trajectory.

The dynamic pressure path constraints must be met, which is $1/2\rho v^2 \le q_{\max}$. This is a nonconvex constraint, which can be approximated by

$$v^2 \le \frac{2q_{\max}}{\rho_0 \exp\left(-h^{(k)}/H_0\right)}. \tag{72}$$

The linkage condition between two flight phases is given by the following equality functions:

$$\begin{aligned} h^{(1)}\left(t_f\right) &= h^{(2)}(t_0), \\ s^{(1)}\left(t_f\right) &= s^{(2)}(t_0), \\ v^{(1)}\left(t_f\right) &= v^{(2)}(t_0), \\ \gamma^{(1)}\left(t_f\right) &= \gamma^{(2)}(t_0), \\ m^{(1)}\left(t_f\right) &= m^{(2)}(t_0) - m_{s,1}, \end{aligned} \tag{73}$$

where $m_{s,1}$ is the structural mass of the first stage. The trajectory optimization is based on the parameter of the available data of an existing launch vehicle, Falcon 9, as listed in Table 3.

The boundary conditions of the flight mission are listed in Table 4. The parameters of the final downrange and the initial flight path angle are not constant values; instead, they can take values in reasonable ranges and are determined by the optimization result.

The optimized states and control are plotted in Figure 14, including the altitude, downrange, velocity, flight path angle, mass, and the thrust magnitude, respectively. The results from GPOPS II are also shown. It can be seen that the results by SCP and NLP-based method coincide with each other, but slight difference exists in the thrust magnitude. GPOPS II obtains a perfect bang-bang control profile, while that of PySCP has a slight slope. This is because the GPOPS II implements hp-adaptive mesh refinement so that it can capture the discontinuity in the control, whereas PySCP does not implement mesh refinement yet.

Twelve iterations are performed before the algorithm converges, and all constraints are satisfied. The trajectory obtained by two methods is almost identical. PySCP only costs 1/6 CPU time of GPOPS II, which proves the high computational efficiency of the PySCP. As optimization theory points out, there is no theoretical guarantee on the maximum iteration and convergence rate for NLP-based methods. In contrast, the SOCP subproblems can be solved in a limited iteration number.

6. Conclusions

A Python software called PySCP has been described for multiple-phase optimal control problems using sequential convex programming. The software uses successive linearization to convexify nonconvex dynamic constraints. The software employs zero-order hold, first-order hold, Runge-Kutta, hp Legendre-Gauss, hp Legendre-Gauss-Radau, and hp flipped Legendre-Gauss-Radau to convert the continuous-time optimal control problem into discrete second-order cone programming problem, which is then solved by primal-dual internal point method. Soft penalty method and adaptive trust region method are employed to manage the iteration process. The utility of the software is demonstrated on three optimal control problems. The software described in this article provides a useful toolkit to solve a wide variety of optimal control problems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

This research is supported by the Huzhou Institute of Zhejiang University under the Huzhou Distinguished Scholar Program (ZJIHI-KY0016).

References

[1] N. Berend and C. Talbot, "Overview of some optimal control methods adapted to expendable and reusable launch vehicle trajectories," *Aerospace Science and Technology*, vol. 10, no. 3, pp. 222–232, 2006.

[2] X. Yang and W. Zhang, "Rapid optimization of ascent trajectory for solid launch vehicles based on Gauss pseudospectral method," *Journal of Astronautics*, vol. 32, no. 1, pp. 15–21, 2011.

[3] H. Bei and W. Xin, "Trajectory optimization of solid launch vehicle based on Hp-adaptive pseudospectral method," *Aerospace Control*, vol. 30, no. 4, pp. 18–22, 2012.

[4] Z. Wang, "Optimal trajectories and normal load analysis of hypersonic glide vehicles via convex optimization," *Aerospace Science and Technology*, vol. 87, pp. 357–368, 2019.

[5] J. T. Betts and S. O. Erb, "Optimal low thrust trajectories to the moon," *SIAM Journal on Applied Dynamical Systems*, vol. 2, no. 2, pp. 144–170, 2003.

[6] E. M. Yong, L. Chen, and G. J. Tang, "A survey of numerical methods for trajectory optimization of spacecraft," *Journal of Astronautics*, vol. 29, no. 2, pp. 397–406, 2008.

[7] X. Yuan and S. Y. Zhu, "Small body descent trajectory optimization based on pseudospectral method," *Journal of Deep Space Exploration*, vol. 3, no. 1, pp. 51–55, 2016.

[8] J. T. Betts, "Survey of numerical methods for trajectory optimization," *Journal of Guidance Control & Dynamics*, vol. 21, no. 2, pp. 193–207, 1998.

[9] J. T. Betts, "Practical methods for optimal control using nonlinear programming," *Society for Industrial and Applied Mathematics*, vol. 55, no. 4, p. B68, 2002.

[10] J. de Muelenaere, *High-Fidelity Trajectory-Based Multidisciplinary Design Optimization for the Conceptual Design of a Reusable Air-Launched Spaceplane*, Stanford University, 2018.

[11] J. Juan, S. Merkli, S. Bennani, and H. Strauch, "FORCES-RTTO: a tool for on-board real-time autonomous trajectory

planning," in *10th International ESA Conference on Guidance, Navigation and Control Systems*, Salzburg, Austria: ESA, 2017.

[12] S. Boyd, S. T. Boyd, and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, New York, 2013.

[13] D. Malyuta, T. P. Reynolds, M. Szmuk et al., "Convex optimization for trajectory generation," 2021, http://arxiv.org/abs/2106.09125.

[14] X. Liu, P. Lu, and P. Pan, "Survey of convex optimization for aerospace applications," *Astrodynamics*, vol. 1, no. 1, pp. 23–40, 2017.

[15] B. Acikmese and P. R. Scott, "Convex programming approach to powered descent guidance for Mars landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, 2007.

[16] B. Acikmese, D. Scharf, L. Blackmore, and A. Wolf, "Enhancements on the convex programming based powered descent guidance algorithm for Mars landing," in *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference and Exhibit,*, Honulolo, Hawaii, August 2008.

[17] L. Blackmore, B. Açikmeşe, and D. P. Scharf, "Minimum-landing-error powered-descent guidance for Mars landing using convex optimization," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 4, pp. 1161–1171, 2010.

[18] B. Açıkmeşe, J. M. Carson, and L. Blackmore, "Lossless convexification of nonconvex control bound and pointing constraints of the soft-landing optimal control problem," *IEEE Transactions on Control Systems Technology*, vol. 21, no. 6, pp. 2104–2113, 2013.

[19] R. Pinson and P. Lu, "Trajectory design employing convex optimization for landing on irregularly shaped asteroids," in *Proceedings of the AIAA/AAS Astrodynamics Specialist Conference*, AIAA 2016-5378, Long Beach, California, 2016.

[20] R. Pinson and P. Lu, "Rapid generation of optimal asteroid powered descent trajectories," in *Proceedings of the AAS/AIAA Astrodynamics Specialist Conference*, AAS 15-616, Vail, Colorado, 2015.

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

[22] M. Szmuk, T. Reynolds, B. Acikmese, M. Mesbahi, and J. M. Carson, *Successive convexification for 6-dof powered descent guidance with compound state-triggered constraints*, In AIAA Scitech Forum, San Diego California, 2019.

[23] M. Sagliano, "Generalized hp pseudospectral-convex programming for powered descent and landing," *Journal of Guidance, Control, and Dynamics*, vol. 42, no. 7, pp. 1562–1570, 2019.

[24] R. Yang and X. Liu, "Fuel-optimal powered descent guidance with free final-time and path constraints," *Acta Astronautica*, vol. 172, pp. 70–81, 2020.

[25] M. Sagliano, A. Heidecker, J. M. Hernández et al., *Onboard guidance for reusable rockets: aerodynamic descent and powered landing*, AIAA Scitech 2021 Forum, 2021.

[26] Y. Mao, M. Szmuk, X. Xu, and B. Açikmese, "Successive convexification: a superlinearly convergent algorithm for nonconvex optimal control problems," 2018, http://arxiv.org/abs/1804.06539.

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

[28] Z. Wang and M. J. Grant, "Improved sequential convex programming algorithms for entry trajectory optimization," *Journal of Spacecraft and Rockets*, vol. 57, no. 6, pp. 1373–1386, 2020.

[29] X. Zhao, R. He, H. Zhang, G. Tang, and W. Bao, "Sequential convex programming method using adaptive mesh refinement for entry trajectory planning problem," *Aerospace Science and Technology*, vol. 109, article 106374, 2021.

[30] M. Sagliano and E. Mooij, "Optimal drag-energy entry guidance via pseudospectral convex optimization," in *AIAA Guidance, Navigation, and Control Conference*, Kissimmee, Florida, 2018.

[31] C. Wang, C. Pei, R. Dai, G. Jing, and J. R. Rea, *Six-dimensional atmosphere entry guidance based on dual quaternion*, In AIAA Scitech 2021 Forum, Virtual Event, 2021.

[32] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot navigation information via sequential convex programming," *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 4634–4641, Hamburg, Germany, Oct. 2015.

[33] J. Grzymisch and W. Fichter, "Optimal rendezvous guidance with enhanced bearings-only observability," *Journal of Guidance, Control, and Dynamics*, vol. 38, pp. 1131–1140, 2015.

[34] C. Louembet, D. Arzelier, and G. Deaconu, "Robust rendezvous planning under maneuver execution errors," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 1, pp. 76–93, 2015.

[35] W. G. Vlases, S. W. Paris, R. M. Lajoie, and M. J. Martens, "Optimal trajectories by implicit simulation," in *Technical report WRDC-TR-90-3056, Boeing Aerospace and Electronics*, Wright-Patterson Air Force Base, Ohio, 1990.

[36] A. V. Rao, D. A. Benson, C. L. Darby et al., "Algorithm 902: GPOPS, A MATLAB software for solving multiple-phase optimal control problems using the gauss pseudospectral method," *ACM Transactions on Mathematical Software*, vol. 37, no. 2, pp. 22–39, 2010.

[37] M. A. Patterson and A. V. Rao, "GPOPS-II: a MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming," *ACM Transactions on Mathematical Software*, vol. 41, no. 1, pp. 1–37, 2014.

[38] P. E. Gill, W. Murray, and M. A. Saunders, "SNOPT an SQP algorithm for large-scale constrained optimization," *SIAM Review*, vol. 47, no. 1, 2001.

[39] L. T. Biegler and V. M. Zavala, "Large-scale nonlinear programming using IPOPT: an integrating framework for enterprise-wide dynamic optimization," *Computers & Chemical Engineering*, vol. 33, no. 3, pp. 575–582, 2009.

[40] I. M. Ross and F. Fahroo, *User's Manual for DIDO 2001 α: A MATLAB Application for Solving Optimal Control ProbleTechnique report ID AAS-01-03*, Department of Aeronautics and Astronautics, Naval Postgraduate School, Monterey, CA, 2001.

[41] I. M. Ross, "Enhancements to the DIDO optimal control toolbox," p. 17, 2020, http://arxiv.org/abs/2004.13112.

[42] G. L. Brauer, D. Cornick, and R. Stevenson, "Capabilities and Applications of the Program to Optimize Simulated Trajectories," in *Program Summary Document*, No. NASA-CR-2770. NASA, 1977.

[43] L. Huneker, M. Sagliano, and Y. Arslantas, "Spartan: an improved global pseudospectral algorithm for high-fidelity entry-descent-landing guidance analysis," in *30th*

International Symposium on Space Technology and Science, ISTS and IEPC Paper 2015-d-43, Kobe Japan, 2015.

[44] M. Sagliano, S. Theil, V. D. Onofrio, and M. Bergsma, "Spartan: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis," in *Advances in Aerospace Guidance, Navigation and Control*, pp. 669–688, Springer, Cham, 2018.

[45] Y. Nie, O. Faqir, and E. C. Kerrigan, "ICLOCS2: try this optimal control problem solver before you try the rest," in *2018 UKACC 12th International Conference on Control*, Sheffield, UK, Sept. 2018.

[46] Y. Nie, O. Faqir, and E. C. Kerrigan, "ICLOCS2: solve your optimal control problems with less pain," in *6th IFAC Conference on Nonlinear Model Predictive Control*, Wisconsin, 2018.

[47] S. J. Wright, *Primal-dual interior-point methods*, Society for Industrial and Applied Mathematics, 1997.

[48] E. D. Andersen, C. Roos, and T. Terlaky, "On implementing a primal-dual interior-point method for conic quadratic optimization," *Mathematical Programming*, vol. 95, no. 2, pp. 249–277, 2003.

[49] K. C. Toh, M. J. Todd, and R. H. Tutuncu, "SDPT3 — a Matlab software package for semidefinite programming, version 1.3," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 545–581, 1999.

[50] J. F. Sturm, "Using SeDuMi 1.02: a Matlab toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11, no. 1-4, pp. 625–653, 1999.

[51] A. Domahidi, E. Chu, and S. Boyd, "ECOS: an SOCP solver for embedded system," *Proceedings of the European Control Conference*, 2013, pp. 3071–3076, Zurich, Switzerland, July 2013.

[52] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," 2016, http://cvxr.com/cvx.

[53] S. Diamond and S. Boyd, "CVXPY: a python-embedded modeling language for convex optimization," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2909–2913, 2016.

[54] M. Andersen, J. Dahl, and L. Vandenberghe, "CVXOPT: Python software for convex optimization, version 1.1," 2015, http://cvxopt.org.

[55] E. Chu, P. Neal, A. Domahidi, and S. Boyd, "Code generation for embedded second-order cone programming," in *Proceedings of the European Control Conference*, Zurich, Switzerland, July 2013.

[56] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for Optimization*, The MIT Press, London, England, 2019.

[57] X. Liu, Z. Shen, and P. Lu, "Solving the maximum-crossrange problem via successive second-order cone programming with a line search," *Aerospace Science and Technology*, vol. 47, pp. 10–20, 2015.

[58] D. Garg, M. Patterson, W. W. Hager, A. V. Rao, D. A. Benson, and G. T. Huntington, "A unified framework for the numerical solution of optimal control problems using pseudospectral methods," *Automatica*, vol. 46, no. 11, pp. 1843–1851, 2010.

[59] M. Sagliano, "Pseudospectral convex optimization for powered descent and landing," *Journal of Guidance Control and Dynamics*, vol. 41, no. 2, pp. 1–15, 2015.

[60] I. M. Ross and F. Fahroo, *Legendre Pseudospectral Approximations of Optimal Control Problems*, Springer, Berlin Heidelberg, 2004.

[61] D. Garg, M. A. Patterson, W. Hager, A. V. Rao, D. Benson, and G. Huntington, "An overview of three pseudospectral methods for the numerical solution of optimal control problems," *Advances in the Astronautical Sciences*, vol. 135, pp. 1–17, 2017.

[62] X. Tang, Z. Liu, and Y. Hu, "New results on pseudospectral methods for optimal control," *Automatica*, vol. 65, pp. 160–163, 2016.

[63] C. A. Rabbath and N. Lechevin, *Discrete-Time Continuous System Design with Applications*, Springer Science & Business Media, New York, 2014.

[64] J. S. Meditch, "On the problem of optimal thrust programming for a lunar soft landing," *IEEE Transactions on Automatic Control*, vol. 9, no. 4, pp. 477–484, 1964.