CEAS-GNC-2024-072

EuroGNC Navigation Guidance Control CEAS Bristol, UK June 11th-13th 2024
University of BRISTOL ROYAL AERONAUTICAL SOCIETY AIAA

Flight Trajectory Generation through a Collocation Approach with Successive Linear Programming

Zhidong Lu Ph.D. Candidate, Technical University of Munich, Institute of Flight System Dynamics, 85748, Garching, Germany. zhidong.lu@tum.de

Haichao Hong Associate Professor, Shanghai Jiao Tong University, School of Aeronautics and Astronautics, 200240, Shanghai, People's Republic of China. haichao.hong@sjtu.edu.cn

Florian Holzapfel Professor, Technical University of Munich, Institute of Flight System Dynamics, 85748, Garching, Germany. florian.holzapfel@tum.de

ABSTRACT

Optimal control methods are widely used to generate flight trajectories for aerial vehicles. These optimal control problems are generally non-convex due to nonlinear flight dynamics and constraints. This study integrates a collocation framework with successive linear programming to address non-convex trajectory generation problems. A linear programming subproblem is constructed by linearizing nonlinear collocation constraints and path constraints. This subproblem aims to find optimal increments of parameters, states, and controls to refine a reference trajectory, which is subsequently re-linearized to formulate subsequent subproblems. An approximate solution to the original optimal control problem is derived through the iterative resolution of these subproblems. To address the potential unboundedness and infeasibility of the subproblem, this paper incorporates linearized constraints into the cost function via exact penalties and enforces trust region on the increments at each collocation node. The maximum allowable trust region size is dynamically adjusted based on the linearization error to assure global convergence. Practical applications in fixed-wing aircraft and quadrotor trajectory generation tasks demonstrate the effectiveness of our approach. Comparative analyses with solutions from state-of-the-art toolboxes indicate that our method achieves near-optimal and dynamically feasible trajectories more efficiently in terms of iterations and computational time. The source code for the algorithm and examples presented in this paper is available at `https://github.com/lenleo1/Colloc_SLP.git`.

Keywords: Trajectory Generation; Optimal Control; Trapezoidal Collocation; Successive Linear Programming

1 Introduction

Well-designed trajectories are essential for the safe and efficient operation of aerial vehicles. Various trajectory generation techniques have been proposed in the literature. Among these, geometric methods such as Dubins paths and clothoid curves [1, 2] have been used widely to generate flight trajectories. These methods are simple but may not fully account for the complex flight dynamics, resulting in dynamically infeasible trajectories unsuitable for flight control. In contrast, optimal control methods generate dynamically feasible trajectories and offer improved performance.

A predominant approach for solving optimal control problems (OCPs) involves direct methods, transcribing the OCP into a nonlinear programming (NLP) problem by discretizing the trajectory. Dis-

cretization techniques commonly used include shooting methods and collocation methods. The single shooting method discretizes only the control input and thus reduces the number of optimization variables, but it is sensitive to the initial guess [3]. Collocation methods, which involve full discretization of state and control histories, stand out for their sparse gradient patterns, rapid local convergence, and ability to manage unstable systems [4]. However, the computational complexity of solving the resulting NLP problem remains substantial using general solvers.

Considering the challenges in solving a non-convex NLP, recent research has increasingly focused on convex optimization for solving optimal flight control problems [5–12]. Convex Optimization is appealing due to its theoretical guarantee of obtaining a globally optimal solution within polynomial time if the problem configuration is feasible [13]. Within this context, two major approaches have been distinguished. The first is *lossless convexification*, which transforms the original non-convex problem into a convex counterpart while ensuring that solutions to the convex problem apply to the original [5, 6]. The second approach is *Sequential Convex Programming (SCP)*, which solves the original non-convex problem through a series of convex subproblems [7–11]. The convex subproblem can take the form of a Linear Program (LP), a Quadratic Program (QP), a Convex Quadratically Constrained Quadratic Program (QCQP), or a Second-Order Cone Program (SOCP). Usually, the *successive linearization* technique is applied to linearize non-convex constraints in the original problem, facilitating the formulation of a convex subproblem. After this subproblem is solved to full optimality, a new subproblem is created by re-linearizing the original problem at the updated solution. Unlike the well-known Sequential Quadratic Program (SQP) algorithms that use Hessian approximation of nonlinear constraints, the Sequential Convex Programming (SCP) methods documented in the literature generally rely solely on first-order linearization of constraints and do not necessitate Hessian approximations. The linearized constraints may render the subproblem *infeasible* or *unbounded*. To tackle the *infeasibility* issue, Ref. [11] introduced collocation residuals between every two adjacent nodes to ensure dynamical feasibility. References [8, 9] introduced virtual controls to the linearized dynamics and constraints and then added them to the cost function. Reference [14] directly penalized constraint violations by adding penalties to the cost. Regarding *unbounded* subproblems, trust-region constraints are commonly used to restrict the optimization variables to a vicinity of the reference solution. Furthermore, to enhance convergence, the size of the trust region is updated in the iterative process based on a quantitative measure of the linearization error, that is, the trust region is expanded, shrunk, or maintained according to standard trust region updating rules [15].

Many recent applications of SCP have formulated the subproblem as a SOCP. The second-order cone constraints stem from the original OCP [7], or quadratic trust region constraints [9]. However, second-order cone constraints in flight dynamics are uncommon in the specific context of aircraft flight trajectory generation. Linear constraints can also express the trust region. Moreover, in many flight trajectory generation tasks, cost functions can be modeled as linear functions of optimization variables when a collocation framework is used to transcribe the OCP. Some examples are time duration [16, 17], flight range [17, 18], fuel or energy consumption [19], and a combination of these [20]. These observations justify the use of simpler LP subproblems within SCP. We acknowledge that many previous works have examined the *Successive Linear Programming (SLP)* algorithms, highlighting their advantage in solving large-scale sparse problems [21, 22]. Our work builds upon these SLP methods and introduces several modifications, including implementing multiple trust regions, penalizing the trust region sizes, and utilizing a continuous trust region update rule. These modifications aim at enhancing the robustness and convergence of the iterative process.

In this paper, we leverage the successive linearization technique within a trapezoidal collocation framework to formulate an LP subproblem characterized by a linear cost function, linear constraints inherited from the original OCP, and linearized constraints. We address potential *infeasibility* and *unboundedness* by incorporating all linearized constraints into the cost function via exact penalties and imposing trust regions on optimization variables. Given the sparsity of the collocation framework, we assign each collocation node a separate trust region instead of using a single trust region for all

optimization variables. Furthermore, an adaptive strategy is applied to adjust the maximum size of trust regions based on the linearization error. Our contributions include:

- Incorporation of successive linearization within the trapezoidal collocation framework, aiding in addressing path constraints, formulating linear cost, and maintaining a sparse structure.
- A feasible LP subproblem is formulated and can be efficiently solved. Multiple trust regions and continuous updates of their sizes help reduce iterations.
- Application of this algorithm to generate feasible flight trajectories for a fixed-wing aircraft and a multi-rotor, achieving effective results with minimal iterations.

The remainder of this paper is structured as follows: Sec. 2 delves into the preliminaries of the trapezoidal collocation method. Sec. 3 delineates the formulation of the LP subproblem, followed by Sec. 4, which elaborates on the iterative algorithm. In Sec. 5, we apply our method to real-world applications involving a fixed-wing aircraft and a quadrotor, juxtaposing our results with leading trajectory optimization software using trapezoidal collocation. Finally, Sec. 6 concludes the paper.

2 Preliminaries on Trapezoidal Collocation

Consider a continuous-time optimal problem in the following form:

$$\min_{\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}} \quad J(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) = \int_0^{t_f} L\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}\right) dt + \phi\left(\boldsymbol{x}\left(t_f\right),\boldsymbol{p}\right) \tag{1}$$

$$\text{s.t.} \quad \frac{d\boldsymbol{x}}{dt} = \boldsymbol{f}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}\right) \tag{2}$$

$$\boldsymbol{x}_{\text{lb}} \le \boldsymbol{x} \le \boldsymbol{x}_{\text{ub}},\ \boldsymbol{u}_{\text{lb}} \le \boldsymbol{u} \le \boldsymbol{u}_{\text{ub}},\ \boldsymbol{p}_{\text{lb}} \le \boldsymbol{p} \le \boldsymbol{p}_{\text{ub}} \tag{3}$$

$$\boldsymbol{C}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}\right) \le \boldsymbol{0} \tag{4}$$

$$\boldsymbol{\psi}\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}\right) = \boldsymbol{0} \tag{5}$$

Here, $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is the state vector, $\boldsymbol{u} \in \mathbb{R}^{n_u}$ is the control vector, $\boldsymbol{p} \in \mathbb{R}^{n_p}$ represents the vector of optimizable parameters, $J : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$ is the scalar cost function. This cost is composed of two parts: the Lagrange term $L : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}$ and the Mayer term $\phi : \mathbb{R}^{n_x} \times \mathbb{R}^{n_p} \to \mathbb{R}$. The function $\boldsymbol{f} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_x}$ characterizes the state dynamics. The inequality and equality constraints are embodied by: $\boldsymbol{C} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_c}$ and $\boldsymbol{\psi} : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R}^{n_p} \to \mathbb{R}^{n_\psi}$. The optimization variables comprise the time histories of states $\boldsymbol{x}(t)$ and controls $\boldsymbol{u}(t)$, as well as time-invariant parameters $\boldsymbol{p}$. They are confined within a lower bound *lb* and an upper bound *ub* as in Eq. (3).

The Bolza form in Eq. (1) can represent objectives prevalent in trajectory generation tasks. The Lagrange term L can be transmuted into a Mayer term via the incorporation of a new state variable into the system dynamics:

$$\frac{dx_{\text{lag}}}{dt} = L\left(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}\right),\ x_{\text{lag}}\left(0\right) = 0 \tag{6}$$

Subsequently, the integration of L materializes as $x_{\text{lag}}\left(t_f\right)$, which is a linear function of the extended states $\boldsymbol{x}_{\text{ext}} = \left[\boldsymbol{x}^T, x_{\text{lag}}\right]^T$. Frequently, the Mayer term ϕ emerges as a linear function of the states $\boldsymbol{x}$ and the parameters $\boldsymbol{p}$, for example, time duration t_f, flight distance, and fuel consumption. Incorporating these considerations, we postulate that the objective in Eq. (1) can be expressed as a linear function:

$$J(\boldsymbol{x},\boldsymbol{u},\boldsymbol{p}) = \phi\left(\boldsymbol{x}_{\text{ext}}\left(t_f\right),\boldsymbol{p}\right) \tag{7}$$

It should be noted that the two cost functions in Eq. (1) and Eq. (7) are equivalent because the Lagrange term is converted to $x_{\text{lag}}\left(t_f\right)$. For the sake of brevity, we refer to $\boldsymbol{x}_{\text{ext}}$ as $\boldsymbol{x}$ in the remainder of this paper.

Before the discretization of the continuous-time OCP, the physical time t is transformed into the normalized time τ:

$$\tau = \frac{t - t_0}{t_f - t_0}, \ t_0 = 0 \tag{8}$$

By establishing the initial time t_0 as zero (without loss of generality) and treating the final time t_f as an optimizable parameter, we derive the scaled state dynamics:

$$\frac{d\boldsymbol{x}}{d\tau} = t_f \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{p}) = \boldsymbol{g}(\boldsymbol{x}, \boldsymbol{u}\ \boldsymbol{p}) \tag{9}$$

For simplification in subsequent discussions, $\frac{d\boldsymbol{x}}{d\tau}$ will be referred to as $\dot{\boldsymbol{x}}$. Discretization of the trajectory is conducted on the normalized time grid,

$$\tau_k \in [0, 1], \ k = 1, ..., N, \ h_k = \tau_{k+1} - \tau_k > 0 \tag{10}$$

where N is the number of discretization nodes, h_k is the k-th time interval. The optimization variable vector is composed of the following:

$$\boldsymbol{z} = \left[\boldsymbol{p}^T \quad \boldsymbol{x}_1^T \quad \boldsymbol{x}_2^T \quad \cdots \quad \boldsymbol{x}_N^T \quad \boldsymbol{u}_1^T \quad \boldsymbol{u}_2^T \quad \cdots \quad \boldsymbol{u}_N^T\right]^T \in \mathbb{R}^{n_z} \tag{11}$$

where $n_z = n_p + N(n_x + n_u)$ designates the number of optimization variables. To ensure accurate propagation of the nonlinear dynamics across the time grid, we apply the trapezoidal collocation rule:

$$\mathrm{CD}_{k+1} = \boldsymbol{x}_{k+1} - \boldsymbol{x}_k - h_k \frac{(\dot{\boldsymbol{x}}_k + \dot{\boldsymbol{x}}_{k+1})}{2}, \ k = 1, ..., N-1 \tag{12}$$

Here, CD_{k+1} denotes the collocation defect, signifying the integration error within the $[\tau_k, \tau_{k+1}]$ time interval. The trapezoidal rule and the collocation defect are illustrated in Figure 1. All collocation defects should be zero and are enforced as equality constraints in the optimization problem:

$$\boldsymbol{e} = \left[\mathrm{CD}_2^T, \cdots, \mathrm{CD}_N^T\right]^T \triangleq \boldsymbol{0} \tag{13}$$

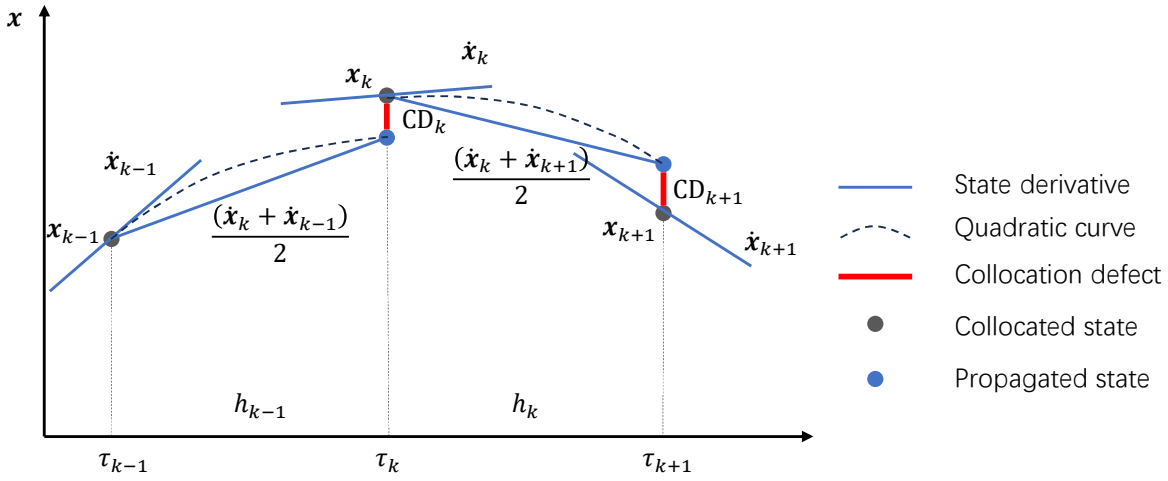

Fig. 1 An illustration of the trapezoidal rule and the collocation defect

Overall, the OCP in Eq. (1) is discretized and transformed to the following NLP:

$$\min_{\boldsymbol{z}} \quad J(\boldsymbol{z}) = \phi(\boldsymbol{x}_N, \boldsymbol{p}) \tag{14}$$

$$\text{s.t.} \quad z_{\text{lb}} \leq z \leq z_{\text{ub}} \tag{15}$$

$$\boldsymbol{e} = \boldsymbol{0} \tag{16}$$

$$\boldsymbol{C}(z) \leq \boldsymbol{0} \tag{17}$$

$$\boldsymbol{\psi}(z) = \boldsymbol{0} \tag{18}$$

3 Linear Subproblem Formulation

In this section, we employ the technique of successive linearization to transform nonlinear collocation constraints, along with other path and point constraints, into linear ones. These linearized constraints are subsequently integrated into the cost function via exact penalties. Furthermore, linear trust region constraints are applied to the parameter vector $\boldsymbol{p}$ and the state/control vectors at each collocation node. By incorporating elastic variables, we formulate a feasible and bounded LP subproblem.

3.1 Optimization Variables and Cost Function

In the context of an iterative process, we assume an extant solution to the original problem, denoted as $z^{(j)}$. As illustrated in Fig. 2, we define the increments to this existing solution as optimization variables:

$$d\boldsymbol{z}^{(j)} = \left[d\boldsymbol{p}^{(j)T} \quad d\boldsymbol{x}_1^{(j)T} \quad d\boldsymbol{x}_2^{(j)T} \quad \cdots \quad d\boldsymbol{x}_N^{(j)T} \quad d\boldsymbol{u}_1^{(j)T} \quad d\boldsymbol{u}_2^{(j)T} \quad \cdots \quad d\boldsymbol{u}_N^{(j)T} \right]^T \tag{19}$$

The solution in the next iteration is:

$$z^{(j+1)} = z^{(j)} + dz^{(j)} \tag{20}$$

Similarly, the cost function Eq. (14) can be written as:

$$J^{(j+1)} = \phi\left(\boldsymbol{x}_N^{(j)} + d\boldsymbol{x}_N^{(j)}, \boldsymbol{p}^{(j)} + d\boldsymbol{p}^{(j)} \right) \tag{21}$$

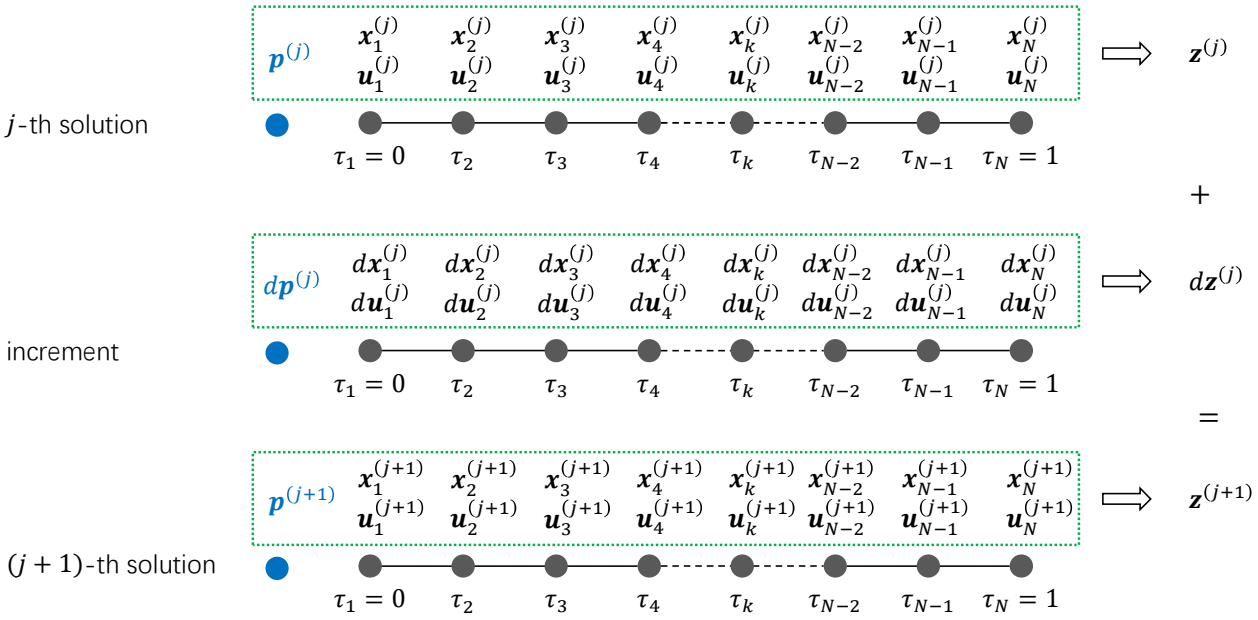

Fig. 2 Discretization grid and optimization variables

3.2 Successive Linearization of Nonlinear Dynamics

Within the iterative process, the nonlinear dynamics Eq. (9) at the $(j+1)$-th iteration can be linearized with respect to the j-th solution, yielding the following approximation:

$$\begin{aligned}\dot{\boldsymbol{x}}^{(j+1)} \approx & \dot{\boldsymbol{x}}^{(j)} + \frac{\partial \boldsymbol{g}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)}, \boldsymbol{p}^{(j)}\right)}{\partial \boldsymbol{x}}\left(\boldsymbol{x}^{(j+1)} - \boldsymbol{x}^{(j)}\right) + \frac{\partial \boldsymbol{g}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)}, \boldsymbol{p}^{(j)}\right)}{\partial \boldsymbol{u}}\left(\boldsymbol{u}^{(j+1)} - \boldsymbol{u}^{(j)}\right) \\ & + \frac{\partial \boldsymbol{g}\left(\boldsymbol{x}^{(j)}, \boldsymbol{u}^{(j)}, \boldsymbol{p}^{(j)}\right)}{\partial \boldsymbol{p}}\left(\boldsymbol{p}^{(j+1)} - \boldsymbol{p}^{(j)}\right) \\ & = \dot{\boldsymbol{x}}^{(j)} + \boldsymbol{G}_x^{(j)} d\boldsymbol{x}^{(j)} + \boldsymbol{G}_u^{(j)} d\boldsymbol{u}^{(j)} + \boldsymbol{G}_p^{(j)} d\boldsymbol{p}^{(j)}\end{aligned} \tag{22}$$

where $\boldsymbol{G}_x^{(j)}$, $\boldsymbol{G}_u^{(j)}$, and $\boldsymbol{G}_p^{(j)}$ denote gradients of $\boldsymbol{g}$ concerning the states, controls, and parameters, respectively. Based on Eq. (22), we examine two adjacent nodes at τ_k and τ_{k+1}:

$$\hat{\dot{\boldsymbol{x}}}_k^{(j+1)} = \dot{\boldsymbol{x}}_k^{(j)} + (\boldsymbol{G}_x)_k^{(j)} d\boldsymbol{x}_k^{(j)} + (\boldsymbol{G}_u)_k^{(j)} d\boldsymbol{u}_k^{(j)} + (\boldsymbol{G}_p)_k^{(j)} d\boldsymbol{p}^{(j)} \tag{23}$$

$$\hat{\dot{\boldsymbol{x}}}_{k+1}^{(j+1)} = \dot{\boldsymbol{x}}_{k+1}^{(j)} + (\boldsymbol{G}_x)_{k+1}^{(j)} d\boldsymbol{x}_{k+1}^{(j)} + (\boldsymbol{G}_u)_{k+1}^{(j)} d\boldsymbol{u}_{k+1}^{(j)} + (\boldsymbol{G}_p)_{k+1}^{(j)} d\boldsymbol{p}^{(j)} \tag{24}$$

Here the symbol $\hat{\dot{\boldsymbol{x}}}$ denotes the linear approximation of $\dot{\boldsymbol{x}}$. By combing Eq. (12) with Eq. (23-24), the collocation defect emanated from the approximated dynamics are:

$$\begin{aligned}\widehat{\mathrm{CD}}_{k+1}^{(j+1)} = & \boldsymbol{x}_{k+1}^{(j+1)} - \boldsymbol{x}_k^{(j+1)} - h_k \frac{\left(\hat{\dot{\boldsymbol{x}}}_k^{(j+1)} + \hat{\dot{\boldsymbol{x}}}_{k+1}^{(j+1)}\right)}{2} \\ = & \boldsymbol{x}_{k+1}^{(j)} + d\boldsymbol{x}_{k+1}^{(j)} - \boldsymbol{x}_k^{(j)} - d\boldsymbol{x}_k^{(j)} - \frac{h_k}{2}\left[\dot{\boldsymbol{x}}_k^{(j)} + (\boldsymbol{G}_x)_k^{(j)} d\boldsymbol{x}_k^{(j)} + (\boldsymbol{G}_u)_k^{(j)} d\boldsymbol{u}_k^{(j)} + (\boldsymbol{G}_p)_k^{(j)} d\boldsymbol{p}^{(j)}\right] \\ & - \frac{h_k}{2}\left[\dot{\boldsymbol{x}}_{k+1}^{(j)} + (\boldsymbol{G}_x)_{k+1}^{(j)} d\boldsymbol{x}_{k+1}^{(j)} + (\boldsymbol{G}_u)_{k+1}^{(j)} d\boldsymbol{u}_{k+1}^{(j)} + (\boldsymbol{G}_p)_{k+1}^{(j)} d\boldsymbol{p}^{(j)}\right] \\ = & \boldsymbol{x}_{k+1}^{(j)} - \boldsymbol{x}_k^{(j)} - \frac{h_k}{2}\left(\dot{\boldsymbol{x}}_k^{(j)} + \dot{\boldsymbol{x}}_{k+1}^{(j)}\right) - \left[\frac{h_k}{2}(\boldsymbol{G}_x)_k^{(j)} + \boldsymbol{I}\right] d\boldsymbol{x}_k^{(j)} - \frac{h_k}{2}(\boldsymbol{G}_u)_k^{(j)} d\boldsymbol{u}_k^{(j)} \\ & - \left[\frac{h_k}{2}(\boldsymbol{G}_x)_{k+1}^{(j)} - \boldsymbol{I}\right] d\boldsymbol{x}_{k+1}^{(j)} - \frac{h_k}{2}(\boldsymbol{G}_u)_{k+1}^{(j)} d\boldsymbol{u}_{k+1}^{(j)} - \frac{h_k}{2}\left[(\boldsymbol{G}_p)_k^{(j)} + (\boldsymbol{G}_p)_{k+1}^{(j)}\right] d\boldsymbol{p}^{(j)}\end{aligned} \tag{25}$$

For notational clarity, we proffer the following abbreviations:

$$(\boldsymbol{F}_{xs})_k^{(j)} = \left[\frac{h_k}{2}(\boldsymbol{G}_x)_k^{(j)} + \boldsymbol{I}\right] \tag{26}$$

$$(\boldsymbol{F}_{xe})_k^{(j)} = \left[\frac{h_k}{2}(\boldsymbol{G}_x)_{k+1}^{(j)} - \boldsymbol{I}\right] \tag{27}$$

$$(\boldsymbol{F}_u)_k^{(j)} = \frac{h_k}{2}(\boldsymbol{G}_u)_k^{(j)} \tag{28}$$

$$(\boldsymbol{F}_u)_{k+1}^{(j)} = \frac{h_k}{2}(\boldsymbol{G}_u)_{k+1}^{(j)} \tag{29}$$

$$(\boldsymbol{F}_p)_k^{(j)} = \frac{h_k}{2}\left[(\boldsymbol{G}_p)_k^{(j)} + (\boldsymbol{G}_p)_{k+1}^{(j)}\right] \tag{30}$$

This facilitates the expression of Eq. (25) as:

$$\widehat{\mathrm{CD}}_{k+1}^{(j+1)} = \mathrm{CD}_{k+1}^{(j)} - (\boldsymbol{F}_{xs})_k^{(j)} d\boldsymbol{x}_k^{(j)} - (\boldsymbol{F}_u)_k^{(j)} d\boldsymbol{u}_k^{(j)} - (\boldsymbol{F}_{xe})_k^{(j)} d\boldsymbol{x}_{k+1}^{(j)} - (\boldsymbol{F}_u)_{k+1}^{(j)} d\boldsymbol{u}_{k+1}^{(j)} - (\boldsymbol{F}_p)_k^{(j)} d\boldsymbol{p}^{(j)} \tag{31}$$

Notably, $\widehat{\mathrm{CD}}_{k+1}^{(j+1)}$ is the predicted collocation defect at the $(j+1)$-th iteration, while $\mathrm{CD}_{k+1}^{(j)}$ stands as the true collocation defect at the j-th iteration. This equation indicates that the predicted collocation defect at the next iteration is the sum of the true collocation defect at the current iteration and some linear terms concerning the increments of states, control, and parameters. A matrix expression encompassing all predicted collocation defects across the collocation nodes is established as:

$$\begin{pmatrix} \widehat{\mathrm{CD}}_2^{(j+1)} \\ \widehat{\mathrm{CD}}_3^{(j+1)} \\ \vdots \\ \widehat{\mathrm{CD}}_N^{(j+1)} \end{pmatrix} = \begin{pmatrix} \mathrm{CD}_2^{(j)} \\ \mathrm{CD}_3^{(j)} \\ \vdots \\ \mathrm{CD}_N^{(j)} \end{pmatrix} -$$

$$\begin{pmatrix} (\boldsymbol{F}_p)_1^{(j)} & (\boldsymbol{F}_{xs})_1^{(j)} & (\boldsymbol{F}_{xe})_2^{(j)} & 0 & \cdots & 0 & 0 & (\boldsymbol{F}_u)_1^{(j)} & (\boldsymbol{F}_u)_2^{(j)} & 0 & \cdots & 0 & 0 \\ (\boldsymbol{F}_p)_2^{(j)} & 0 & (\boldsymbol{F}_{xs})_2^{(j)} & (\boldsymbol{F}_{xe})_3^{(j)} & \cdots & 0 & 0 & 0 & (\boldsymbol{F}_u)_2^{(j)} & (\boldsymbol{F}_u)_3^{(j)} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ (\boldsymbol{F}_p)_{N-1}^{(j)} & 0 & 0 & 0 & \cdots & (\boldsymbol{F}_{xs})_{N-1}^{(j)} & (\boldsymbol{F}_{xe})_N^{(j)} & 0 & 0 & 0 & \cdots & (\boldsymbol{F}_u)_{N-1}^{(j)} & (\boldsymbol{F}_u)_N^{(j)} \end{pmatrix} \begin{pmatrix} d\boldsymbol{p}^{(j)} \\ d\boldsymbol{x}_1^{(j)} \\ \vdots \\ d\boldsymbol{x}_N^{(j)} \\ d\boldsymbol{u}_1^{(j)} \\ \vdots \\ d\boldsymbol{u}_N^{(j)} \end{pmatrix} \tag{32}$$

The above matrix expression is simplified as:

$$\hat{\boldsymbol{e}}^{(j+1)} = \boldsymbol{e}^{(j)} - \boldsymbol{A}^{(j)} d\boldsymbol{z}^{(j)} \triangleq \boldsymbol{0} \tag{33}$$

In this way, the nonlinear collocation defect constraints about $\boldsymbol{z}$ in Eq. (16) are approximated by linear equality constraints about $d\boldsymbol{z}$ in Eq. (33).

3.3 Other Constraints

The box bounds in Eq. (15) can be directly transformed to upper and lower bounds on $d\boldsymbol{z}^{(j)}$:

$$\boldsymbol{z}_{\mathrm{lb}} - \boldsymbol{z}^{(j)} \leq d\boldsymbol{z}^{(j)} \leq \boldsymbol{z}_{\mathrm{ub}} - \boldsymbol{z}^{(j)} \tag{34}$$

The equality constraints in Eq. (18) are divided into a linear part $\bar{\boldsymbol{\psi}}(\boldsymbol{z})$ and a nonlinear part $\tilde{\boldsymbol{\psi}}(\boldsymbol{z})$. This nonlinear part undergoes linearization at $\boldsymbol{z}^{(j)}$:

$$\tilde{\boldsymbol{\psi}}\left(\boldsymbol{z}^{(j+1)}\right) \approx \left(\frac{\partial \tilde{\boldsymbol{\psi}}}{\partial \boldsymbol{z}}\right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{\psi}}\left(\boldsymbol{z}^{(j)}\right) = \boldsymbol{0} \tag{35}$$

In a parallel fashion, the inequality constraints in Eq. (17) are also partitioned into a linear part $\bar{\boldsymbol{C}}(\boldsymbol{z})$ and a nonlinear part $\tilde{\boldsymbol{C}}(\boldsymbol{z})$. The latter is linearized at $\boldsymbol{z}^{(j)}$:

$$\tilde{\boldsymbol{C}}\left(\boldsymbol{z}^{(j+1)}\right) \approx \left(\frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{z}}\right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{C}}\left(\boldsymbol{z}^{(j)}\right) \leq \boldsymbol{0} \tag{36}$$

3.4 Linear subproblem

Upon integrating these linear cost and constraints, we derive a linear subproblem as follows:

$$\min_{d\boldsymbol{z}} \quad J^{(j+1)} = \phi\left(\boldsymbol{x}_N^{(j)} + d\boldsymbol{x}_N^{(j)}, \boldsymbol{p}^{(j)} + d\boldsymbol{p}^{(j)}\right) \tag{37}$$

$$\text{s.t.} \quad \boldsymbol{z}_{\mathrm{lb}} - \boldsymbol{z}^{(j)} \leq d\boldsymbol{z}^{(j)} \leq \boldsymbol{z}_{\mathrm{ub}} - \boldsymbol{z}^{(j)} \tag{38}$$

$$\boldsymbol{e}^{(j)} - \boldsymbol{A}^{(j)} d\boldsymbol{z}^{(j)} = \boldsymbol{0} \tag{39}$$

$$\bar{\boldsymbol{\psi}}\left(d\boldsymbol{z}^{(j)}\right) + \bar{\boldsymbol{\psi}}\left(\boldsymbol{z}^{(j)}\right) = \boldsymbol{0} \tag{40}$$

$$\left(\frac{\partial \tilde{\boldsymbol{\psi}}}{\partial \boldsymbol{z}}\right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{\psi}}\left(\boldsymbol{z}^{(j)}\right) = \boldsymbol{0} \tag{41}$$

$$\bar{\boldsymbol{C}}\left(d\boldsymbol{z}^{(j)}\right) + \bar{\boldsymbol{C}}\left(\boldsymbol{z}^{(j)}\right) \leq \boldsymbol{0} \tag{42}$$

$$\left(\frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{z}}\right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{C}}\left(\boldsymbol{z}^{(j)}\right) \leq \boldsymbol{0} \tag{43}$$

The accuracy of these linearized dynamics and constraints is contingent upon $d\boldsymbol{z}^{(j)}$ being sufficiently small. Linearized constraints may render the subproblem unbounded or yield an empty feasible region [23]. To address the *unboundedness* issue, trust region constraints on $d\boldsymbol{z}^{(j)}$ are employed. It is a common practice in both NLP and SCP methods to use a single scalar trust region radius for all optimization variables [15, 23]. Considering the inherent sparsity pattern of our problem, where states and controls at temporally distant collocation nodes exhibit no interconnection, we allocate trust regions distinctively for parameters and states/controls at individual discretization nodes. This multiple trust regions scheme is illustrated in Fig. 3. These trust region constraints are written as follows:

$$\left\|d\boldsymbol{p}^{(j)}\right\|_\infty \leq \Delta_p^{(j)} \leq \Delta_{\max}, \quad \left\|d\boldsymbol{x}_k^{(j)}, d\boldsymbol{u}_k^{(j)}\right\|_\infty \leq \Delta_k^{(j)} \leq \Delta_{\max} \tag{44}$$

Here, the infinity norm is used because it introduces linear constraints. An upper bound $\Delta_{\max}$ is imposed on all trust region sizes. The sizes of all trust regions are further written into a vector:

$$\boldsymbol{\Delta}^{(j)} = \left[\Delta_p^{(j)}, \Delta_1^{(j)}, \Delta_2^{(j)}, \cdots, \Delta_N^{(j)}\right] \tag{45}$$

This trust region vector is updated in each iteration—a facet to be elaborated upon.

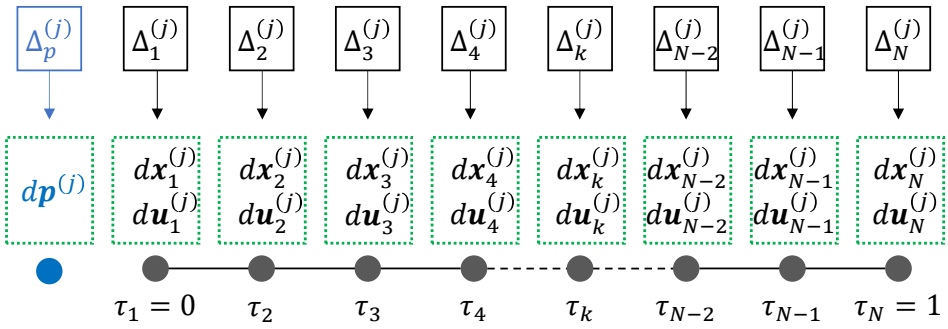

Fig. 3 Multiple trust regions for the parameters, states, and controls

To rectify potential infeasibility caused by linearization, the linearized constraints are amalgamated into the cost function via the following penalty:

$$\begin{aligned}\hat{J}_{\text{aug}}^{(j+1)} =& J^{(j+1)} + \gamma \left\|\boldsymbol{e}^{(j)} - \boldsymbol{A}^{(j)} d\boldsymbol{z}^{(j)}\right\|_1 + \gamma \left\|\left(\frac{\partial \tilde{\boldsymbol{\psi}}}{\partial \boldsymbol{z}}\right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{\psi}}\left(\boldsymbol{z}^{(j)}\right)\right\|_1 \\ &+ \gamma \left\|\max\left(\boldsymbol{0}, \left(\frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{z}}\right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{C}}\left(\boldsymbol{z}^{(j)}\right)\right)\right\|_1\end{aligned} \tag{46}$$

Here, the ℓ_1 penalty is exact, which implies that for γ larger than a certain value, the local minimizer of $J^{(j+1)}$ occurs with the local minimizer of $\hat{J}_{\text{aug}}^{(j+1)}$ [15, p. 507]. Besides, to drive the linearization error small, the trust region sizes are also penalized through an ℓ_1 penalty term:

$$\hat{J}_{\text{total}}^{(j+1)} = \hat{J}_{\text{aug}}^{(j+1)} + \gamma_\Delta \left\| \boldsymbol{\Delta}^{(j)} \right\|_1 \tag{47}$$

The weight γ_Δ should be much less than 1 to reduce trust region sizes without unduly prioritizing this objective over the minimization of $\hat{J}_{\text{aug}}^{(j+1)}$.

We further apply the elastic variable strategy [15, p. 536][22] to reformulate the subproblem into the subsequent form:

$$\min_{dz} \quad J^{(j+1)} + \gamma \left[\sum (\boldsymbol{v}_1 + \boldsymbol{w}_1) + \sum (\boldsymbol{v}_2 + \boldsymbol{w}_2) + \sum \boldsymbol{y} \right] + \gamma_\Delta \sum \boldsymbol{\Delta}^{(j)} \tag{48}$$

$$\text{s.t.} \quad \text{Eq. (38), (40), (42), (44)} \tag{49}$$

$$\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{y}, \boldsymbol{\Delta}^{(j)} \geq \mathbf{0} \tag{50}$$

$$\boldsymbol{e}^{(j)} - \boldsymbol{A}^{(j)} d\boldsymbol{z}^{(j)} = \boldsymbol{v}_1 - \boldsymbol{w}_1 \tag{51}$$

$$\left(\frac{\partial \tilde{\boldsymbol{\psi}}}{\partial \boldsymbol{z}} \right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{\psi}} \left(\boldsymbol{z}^{(j)} \right) = \boldsymbol{v}_2 - \boldsymbol{w}_2 \tag{52}$$

$$\left(\frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{z}} \right)^{(j)} d\boldsymbol{z}^{(j)} + \tilde{\boldsymbol{C}} \left(\boldsymbol{z}^{(j)} \right) \leq \boldsymbol{y} \tag{53}$$

where $\boldsymbol{v}_1$ and $\boldsymbol{w}_1$ are artificial elastic variables for the linearized collocation defect constraints, $\boldsymbol{v}_2$ and $\boldsymbol{w}_2$ are elastic variables for other linearized equality constraints, $\boldsymbol{y}$ is the elastic variable for the linearized inequality constraints. The extended optimization variables of Problem (48) are:

$$\boldsymbol{Z} = \left[d\boldsymbol{z}^{(j)^T}, \boldsymbol{v}_1^T, \boldsymbol{w}_1^T, \boldsymbol{v}_2^T, \boldsymbol{w}_2^T, \boldsymbol{y}^T, \boldsymbol{\Delta}^{(j)^T} \right]^T \tag{54}$$

The cost function and all equality and inequality constraints are linear functions about $\boldsymbol{Z}$; therefore, Problem (48) is a linear programming problem.

4 Successive Linear Programming Algorithm

The preceding section formulates a feasible LP subproblem that approximates the original OCP. After solving this subproblem, the reference trajectory is updated and re-linearized to develop a new subproblem. We aspire to get a solution to the original OCP by solving a series of subproblems. This section presents the comprehensive methodology to achieve this objective.

4.1 Trust Region Update

The trust region constraints are paramount to ensure the subproblem is bounded and feasible. Small region sizes imply low linearization error but allow little progress in improving the current solution. Conversely, large region sizes allow significant progress but reduce the confidence of obtaining a helpful solution. By integrating the trust region sizes as optimization variables, as illustrated in Eq. (54), we allow the linear programming solver to concurrently determine the optimal trust region sizes along with other variables in a manner that minimizes the overall cost, as outlined in Eq. (48). There are two main issues within this self-adaptive process of updating trust regions. First, there is a risk that the trust regions may not converge, potentially perpetuating linearization errors. Second, the imposition of a penalty on trust region sizes, denoted by $\gamma_\Delta \left\| \boldsymbol{\Delta}^{(j)} \right\|_1$, may inadvertently shift the subproblem's solution

away from the local minimum of $J^{(j+1)}$. To address these concerns, we implement a strategy where both the maximum allowable size of the trust regions, $\Delta_{\max}$, and the penalty weight, γ_Δ, are progressively reduced to approach zero beyond a specified iteration threshold, j_1:

$$\Delta_{\max}^{(j)} = \beta_1^{\max(0,j_1-j)} \Delta_{\max}^{(0)} \tag{55}$$

$$\gamma_\Delta^{(j)} = \beta_2^{\max(0,j_1-j)} \gamma_\Delta^{(0)} \tag{56}$$

The pivotal iteration, j_1, is identified as a positive integer, setting a phase within which the linear solver is encouraged to explore the local minimizer of $J^{(j+1)}$. Following this phase, the gradual reduction of both the trust region sizes and their associated penalty weight guides the solver toward attaining an optimal local solution. The ratio β_1 and β_2 are scalars between 0 and 1; larger values are used to allow smooth and gradual reduction of $\Delta_{\max}^{(j)}$ and $\gamma_\Delta^{(j)}$.

In addition to the enforced decrement of the maximum trust region size as a function of iteration number, the trust region is also permitted to expand or contract based on a defined measure of *linearization error*. The measure, denoted as $\rho^{(j)}$, is calculated as the ratio of the actual reduction in the total cost to the predicted reduction, where the total cost is a composite measure comprising both the actual cost, the violation of constraints, and the trust region sizes.

$$J_{\text{total}}^{(j+1)} = J^{(j+1)} + \gamma \left\| \boldsymbol{e}^{(j+1)} \right\|_1 + \gamma \left\| \tilde{\boldsymbol{\psi}} \left(\boldsymbol{z}^{(j+1)} \right) \right\|_1 + \gamma \left\| \max \left(\boldsymbol{0}, \tilde{\boldsymbol{C}} \left(\boldsymbol{z}^{(j+1)} \right) \right) \right\|_1 + \gamma_\Delta \left\| \boldsymbol{\Delta}^{(j)} \right\|_1 \tag{57}$$

$$\rho^{(j)} = \frac{J_{\text{total}}^{(j)} - J_{\text{total}}^{(j+1)}}{J_{\text{total}}^{(j)} - \hat{J}_{\text{total}}^{(j+1)}} \tag{58}$$

A value of $\rho^{(j)}$ close to 1 indicates a small linearization error, warranting the expansion of the trust region. In contrast, a small $\rho^{(j)}$ necessitates a contraction of the trust region. This study adopts and modifies the trust region update rule in Ref. [24], particularly adjusting the rule for instances when $\rho^{(j)} < 0$ to permit further contraction of the trust region and allowing expansion only when the boundary of the trust region is reached. The modified rule is:

$$\Delta_{\max}^{(j+1)} = \begin{cases} \max\left(\alpha_0, \alpha_1 + \eta_1 \rho^{(j)}\right) \Delta_{\max}^{(j)}, & \text{if } \rho^{(j)} < 0 \\ \left[\alpha_1 + (1-\alpha_1)\left(\frac{\rho^{(j)}}{\eta_2}\right)^2\right] \Delta_{\max}^{(j)}, & \text{if } 0 \le \rho^{(j)} < \eta_2 \\ \Delta_{\max}^{(j)}, & \text{if } \rho^{(j)} \ge \eta_2, \left\| d\boldsymbol{z}^{(j)} \right\|_\infty < \Delta_{\max}^{(j)} \\ \left[\alpha_3 + (\alpha_2 - \alpha_3)\, e^{-\left(\frac{\rho^{(j)}-1}{\eta_2-1}\right)^2}\right] \Delta_{\max}^{(j)}, & \text{if } \rho^{(j)} \ge \eta_2, \left\| d\boldsymbol{z}^{(j)} \right\|_\infty = \Delta_{\max}^{(j)} \end{cases} \tag{59}$$

In this study, we set $\alpha_0 = 0.1$, $\alpha_1 = 0.5$, $\alpha_2 = 2$, $\alpha_3 = 1.01$, $\eta_1 = 0.01$, and $\eta_2 = 0.95$. This update rule is illustrated in Fig. 4, where the dotted green line denotes the case when $\rho^{(j)} \ge \eta_2$ and $\left\| d\boldsymbol{z}^{(j)} \right\|_\infty < \Delta_{\max}^{(j)}$. In this case, the maximum trust region size remains unchanged because it was not reached in the previous iteration. The described strategy for updating trust regions aligns with the principle of standard rules [15]. A rigorous analysis of its convergence has not been included in this study.

4.2 Stopping Criteria

The following practical stopping criteria are applied, and the algorithm stops if any criterion is met:

1) The relative change of the actual cost satisfies: $\left| J^{(j)} - J^{(j+1)} \right| \le \epsilon_{\text{opt}} \left| J^{(j)} \right|$.
2) The maximum trust region size has shrunk to less than a predefined tolerance: $\Delta_{\max}^{(j)} \le \epsilon_\Delta$.

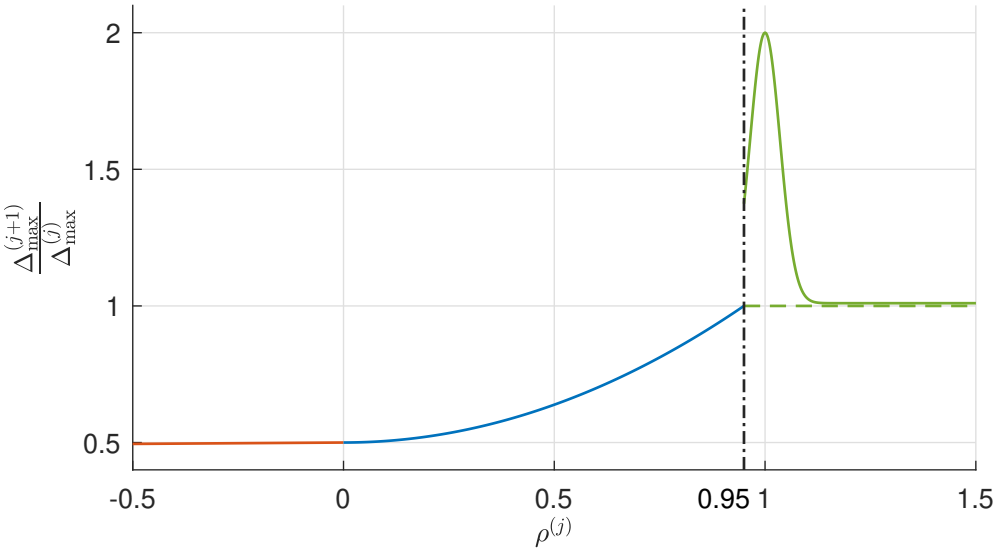

Fig. 4 Update rule for the maximum trust region size

3) The step size is lower than a predefined tolerance: $\left\|dz^{(j)}\right\|_\infty \leq \epsilon_\Delta$.
4) The maximum iteration number is reached.

4.3 Algorithm

The methodology of solving the original OCP via iterative resolution of LP subproblems is expounded in Algorithm 1. User-defined parameters and their default values are listed in Table 1.

Table 1 A summary of the user-defined parameters for Algorithm 1

Parameter	Meaning	Note	Default value
$\Delta_{\max}^{(0)}$	initial maximum trust region	$\Delta_{\max}^{(0)} > 0$	10
γ	penalty weight for linearized constraints	$\gamma > 1$	100
γ_Δ	penalty weight for trust regions	$0 < \gamma_\Delta < 1$	0.1
j_1	threshold iteration	$j_1 > 1$	6
β_1	$\Delta_{\max}$ shrinking ratio	$0 < \beta_1 < 1$	0.7
β_2	γ_Δ shrinking ratio	$0 < \beta_2 < 1$	0.7
ϵ_{opt}	local optimality tolerance	$0 < \epsilon_{\text{opt}} < 1$	1e-5
ϵ_{fea}	feasibility tolerance	$0 < \epsilon_{\text{fea}} < 1$	1e-6
ϵ_Δ	step size and trust region size tolerance	$0 < \epsilon_\Delta < 1$	1e-5

5 Applications

This section presents the application of the above CSLP algorithm on two different flight systems. The first is a three-degree-of-freedom fixed-wing aircraft model; the second is a six-degree-of-freedom quadrotor model. Both models have nonlinear dynamics and are subject to nonlinear path constraints. The performance of our proposed CSLP algorithm is compared with two other state-of-the-art trajectory optimization toolboxes that also use the trapezoidal collocation scheme, namely, the *OptimTraj* toolbox [25] and the *FALCON.m* toolbox [26].

Algorithm 1 Collocation based Successive Linear Programming (CSLP) Algorithm

Require: N, γ, γ_Δ, $\Delta_{\max}^{(0)}$, β_1, β_2, ϵ_{opt}, ϵ_{fea}, ϵ_Δ, j_{m}.

Generate initial guess $\boldsymbol{z}^{(0)}$
for $j = 0, 1, 2, \cdots, j_m$ **do**
    Calculate $\boldsymbol{e}^{(j)}$, $\boldsymbol{A}^{(j)}$, $\left(\frac{\partial \tilde{\boldsymbol{\psi}}}{\partial \boldsymbol{z}}\right)^{(j)}$, $\tilde{\boldsymbol{\psi}}\left(\boldsymbol{z}^{(j)}\right)$, $\left(\frac{\partial \tilde{\boldsymbol{C}}}{\partial \boldsymbol{z}}\right)^{(j)}$, and $\tilde{\boldsymbol{C}}\left(\boldsymbol{z}^{(j)}\right)$.
    Calculate $\Delta_{\max}^{(j)}$ and $\gamma_\Delta^{(j)}$ according to Eq. (55) and Eq. (56).
    Set up the subproblem in Eq. (48).
    Solve the subproblem in Eq. (48) and get the solution $d\boldsymbol{z}^{(j)}$.
    Compute $\rho^{(j)}$ as in Eq. (58).
    Update $\Delta_{\max}^{(j+1)}$ according to Eq. (59)
    if $\rho^{(j)} > 0.01$ **then**
        $\boldsymbol{z}^{(j+1)} = \boldsymbol{z}^{(j)} + d\boldsymbol{z}^{(j)}$ ▷ Accept the new solution
    else
        $\boldsymbol{z}^{(j+1)} = \boldsymbol{z}^{(j)}$ ▷ Keep the old solution
    end if
    Compute the local optimality measure $opt = \left|\boldsymbol{J}^{(j)} - \boldsymbol{J}^{(j+1)}\right| / \left|\boldsymbol{J}^{(j)}\right|$
    Compute the feasibility measure $fea = \left\|\boldsymbol{e}^{(j+1)}\right\|_\infty + \left\|\boldsymbol{\psi}\left(\boldsymbol{z}^{(j+1)}\right)\right\|_\infty + \left\|\max\left(\boldsymbol{0}, \boldsymbol{C}\left(\boldsymbol{z}^{(j+1)}\right)\right)\right\|_\infty$
    if $opt \le \epsilon_{\text{opt}}$ and $fea \le \epsilon_{\text{fea}}$ **then**
        ExitFlag = 1 ▷ The feasibility and the optimality tolerances are met
        break
    end if
    if $\Delta_{\max}^{j} \le \epsilon_\Delta$ or $\left\|d\boldsymbol{z}^{(j)}\right\|_\infty \le \epsilon_\Delta$ **then**
        if $fea \le \epsilon_{\text{fea}}$ **then**
            ExitFlag = 2 ▷ The step and feasibility tolerances are met
        else
            ExitFlag = 3 ▷ The step tolerance is met, but the feasibility tolerance is not
        end if
        break
    end if
    if $j \ge j_m$ **then**
        ExitFlag = 0 ▷ Maximum number of iterations reached
        break
    end if
end for
Return $\boldsymbol{z}^{(j+1)}$, $\boldsymbol{J}^{(j+1)}$, and ExitFlag.

5.1 Fixed-wing Aircraft Flight Trajectory

The first example is the generation of time-minimum flight trajectory for a fixed-wing aircraft [27]. The following point-mass model describes the nonlinear flight dynamics:

$$\dot{X} = V\cos\chi\cos\gamma \tag{60}$$
$$\dot{Y} = V\sin\chi\cos\gamma \tag{61}$$
$$\dot{H} = V\sin\gamma \tag{62}$$
$$\dot{V} = \frac{1}{m}\left(T - D - mg\sin\gamma\right) \tag{63}$$
$$\dot{\chi} = \frac{1}{mV\cos\gamma}L\sin\mu \tag{64}$$
$$\dot{\gamma} = \frac{1}{mV}\left(L - mg\cos\gamma\right) \tag{65}$$

with the aerodynamic forces and the thrust modeled as:

$$L = \frac{1}{2}\rho V^2 S C_L = \frac{1}{2}\rho V^2 S\left(C_{L0} + C_{L\alpha}\alpha\right) \tag{66}$$
$$D = \frac{1}{2}\rho V^2 S C_D = \frac{1}{2}\rho V^2 S\left(C_{D0} + kC_L^2\right) \tag{67}$$
$$T = \delta_T T_{\max} \tag{68}$$

where the wing area $S = 110\ \mathrm{m}^2$, mass $m = 70000$ kg, $C_{L0} = 0.2$, $C_{L\alpha} = 4\ \mathrm{rad}^{-1}$, $C_{D0} = 0.03$, $k = 0.04$, and $T_{\max} = 0.3mg$. The model subsumes six states: horizontal position X, lateral position Y, vertical position H, airspeed V, course angle χ, and climb angle γ. The control inputs are throttle command δ_T, angle of attack α, and bank angle μ. The normal load factor $n_z = \frac{L}{mg}$ is a nonlinear output.

The aircraft mission is to fly from the waypoint $(x, y, z) = (0, 0, 1000\mathrm{m})$ to $(5000\mathrm{m}, 2000\mathrm{m}, 1000\mathrm{m})$, leaving and arriving the waypoints with a speed of 100 m/s and zero course and climb angles. During the flight, the states, controls, and output are limited. All constraints are summarized below:

$$X(0) = Y(0) = 0, H(0) = 1000\ \mathrm{m}, V(0) = 100\ \mathrm{m/s}, \chi(0) = \gamma(0) = 0 \tag{69}$$
$$X(t_f) = 5000\ \mathrm{m}, Y(t_f) = 2000\ \mathrm{m}, H(t_f) = 1000\ \mathrm{m}, V(t_f) = 100\ \mathrm{m/s}, \chi(t_f) = \gamma(t_f) = 0 \tag{70}$$
$$80\ \mathrm{m/s} \le V \le 120\ \mathrm{m/s},\ -\frac{\pi}{6} \le \gamma \le \frac{\pi}{6},\ 0 \le \alpha \le \frac{\pi}{12},\ 0 \le \delta_T \le 1,\ -\frac{\pi}{6} \le \mu \le \frac{\pi}{6} \tag{71}$$
$$0.8 \le n_z \le 1.2 \tag{72}$$

We adopted a collocation grid with $N = 31$ equidistant nodes and the default parameters in Table 1 for the CSLP algorithm. The *linprog* solver [28] was used to solve the linear subproblems. The *FALCON.m* and *OptimTraj* toolboxes were used to generate reference solutions for comparative analysis. Both toolboxes formulate NLPs employing the trapezoidal collocation method and solve them through the *IPOPT* solver [29] and the *fmincon* solver [28], respectively. The same scaling to optimization variables was applied when deriving all three solutions.

Figure 5 and Fig. 6 depict the trajectories devised by our CSLP algorithm alongside the reference solutions. All solutions complied with the prescribed constraints on states and controls. During the initial ten seconds, the aircraft performed a dive while accelerating to its maximum velocity $V = 120$ m/s using maximum thrust and concurrently executed a right bank to $\mu = 30°$. This was followed by approximately 25 seconds, during which the aircraft maintained its velocity with minimal throttle. In the final phase, the aircraft leveraged its kinetic energy to ascend back to its original altitude while decelerating to its initial velocity and executing a left bank to $\mu = -30°$, before finally leveling its wings. The two reference

solutions were almost identical, while the CSLP solution showed some unsteady lateral motions during the constant-speed phase but less altitude loss.

Figure 7 details the progression of cost values and collocation errors across iterations. The CSLP algorithm solution exhibits a slightly prolonged duration (t_f = 47.27 s) compared to the reference solutions (t_f = 47.22 s), yet achieves convergence to a feasible solution in merely five iterations, in contrast to 72 and 101 iterations required by *FALCON.m* and *OptimTraj*, respectively. Figure 8 illustrates the progression of the CSLP algorithm: the first subplot records the actual and total cost values $J^{(j)}$ and $J^{(j)}_{\text{total}}$, and the second subplot records the linearization error measure $\rho^{(j)}$ and the maximum allowable trust region size $\Delta^{(j)}_{\max}$. The initial trust regions are set to 5, an overlarge value resulting in a poor approximation of the original OCP, as evidenced by the rise in $J^{(1)}_{\text{total}}$ and a negative $\rho^{(1)}$ of -1.066. Consequently, the maximum trust region size was reduced to 2.233 in the second iteration, according to the update rule in Eq. (59). Subsequent iterations demonstrated continuous reductions in J_{total} and a ρ value approaching 1, indicating minimal linearization error. The third subplot details the trust region vector $\boldsymbol{\Delta}^{(j)}$, with the first element being the trust region size for dt_f and the remaining 31 elements corresponding to the sizes at all collocation nodes. The optimal trust region sizes vary, particularly at the initial and final nodes. Although the maximum allowable trust region size $\Delta_{\max}$ remained nearly unchanged in the last three iterations, the actual trust region vector $\boldsymbol{\Delta}$ exhibits a gradual decrease among all collocation nodes to the order of 10^{-5}. This gradual decrease is not driven by the update rule for $\Delta_{\max}$ but through the penalization of $\boldsymbol{\Delta}$ in the cost function. Therefore, our CSLP algorithm provides more flexibility to expedite convergence across iterations.

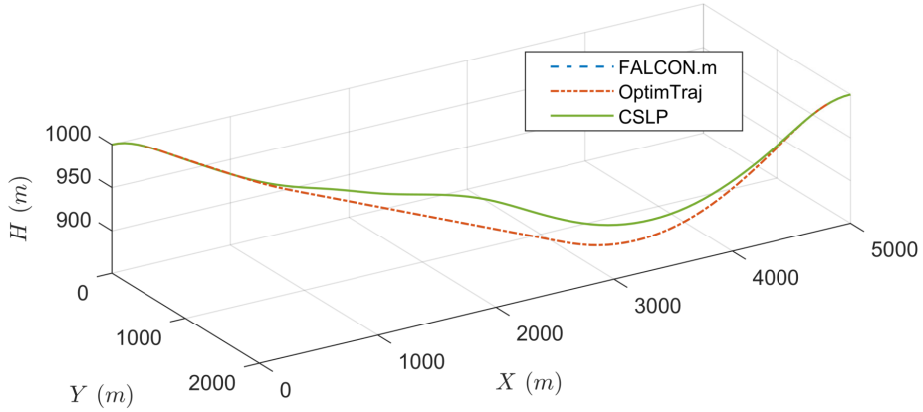

Fig. 5 Minimum time trajectories of the fixed-wing aircraft

5.2 Quadrotor Flight Trajectory

The second example describes the generation of a control-minimum flight trajectory for a quadrotor. This example is adapted from the *OptimTraj* toolbox [25]. The six-degree-of-freedom equations of motion of the quadrotor are as follows:

$$\dot{\boldsymbol{X}}_E = \boldsymbol{V}_E \tag{73}$$

$$\dot{\boldsymbol{\Psi}} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix} \boldsymbol{\omega} \tag{74}$$

$$\dot{\boldsymbol{V}}_E = M_{EB}\frac{\sum_{i=1}^{4}\boldsymbol{F}_i}{m} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{75}$$

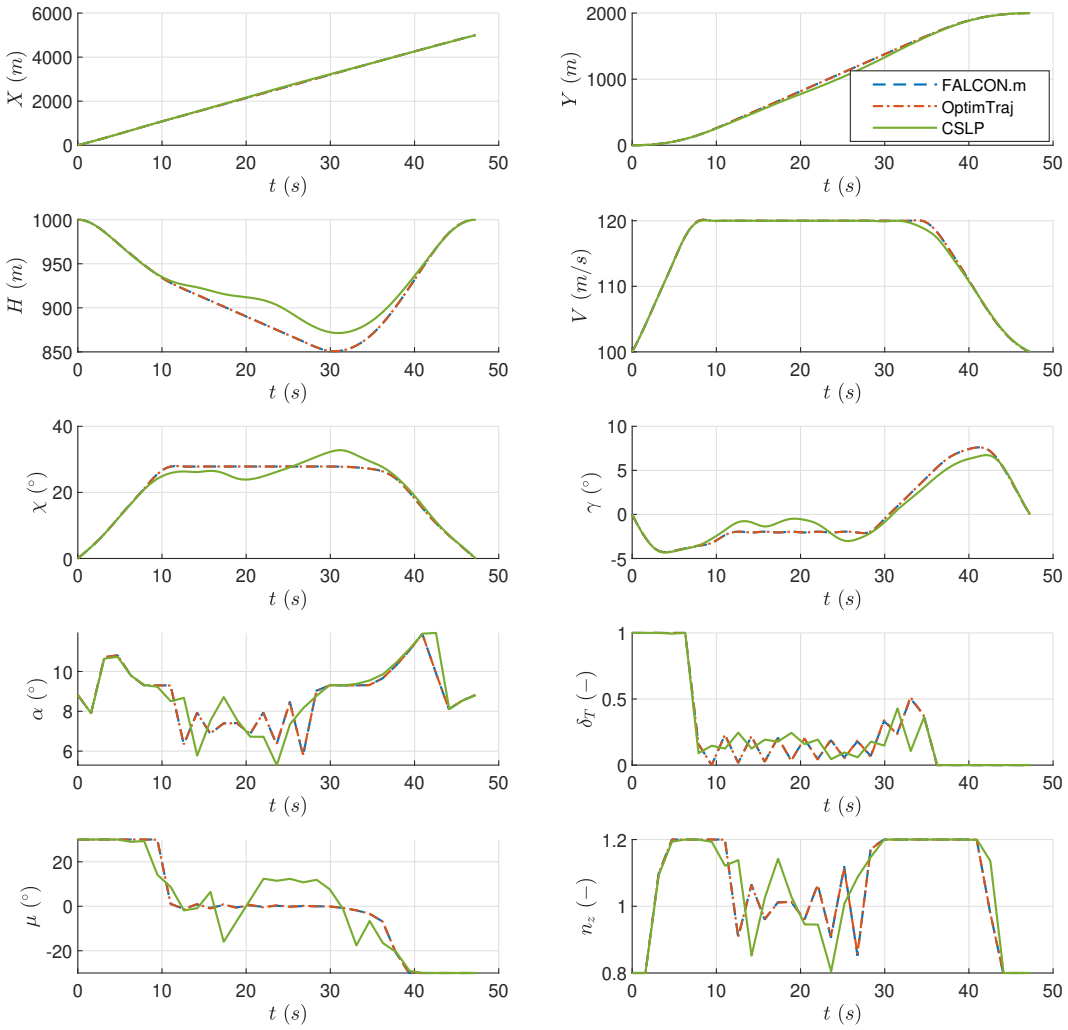

Fig. 6 Time histories of states and controls of the fixed-wing aircraft

$$\dot{\boldsymbol{\omega}} = \frac{\sum_{i=1}^{4} \boldsymbol{M}_i - \boldsymbol{\omega} \times \boldsymbol{I}\boldsymbol{\omega}}{\boldsymbol{I}} \tag{76}$$

where $\boldsymbol{X}_E$ is the three-dimensional location, $\boldsymbol{\Psi} = [\phi, \theta, \psi]^T$ is the Euler attitude, $\boldsymbol{V}_E = [u_E, v_E, w_E]^T$ is the velocity vector in the NED frame, $\boldsymbol{\omega} = [p, q, r]^T$ is the angular rate vector in the body frame, M_{EB} is the transformation matrix from the body frame to the NED frame, $\boldsymbol{I}$ is the inertial matrix, $\boldsymbol{F}_i$ and $\boldsymbol{M}_i$ are the force and moment vector generated by each rotor, which are calculated as follows:

$$\boldsymbol{F}_i = \begin{bmatrix} 0 \\ 0 \\ f_i \end{bmatrix}, \ \boldsymbol{M}_i = \begin{bmatrix} 0 \\ 0 \\ f_i \end{bmatrix} \times \boldsymbol{r}_i + \begin{bmatrix} 0 \\ 0 \\ \tau_i \end{bmatrix} \tag{77}$$

$$f_i = c_f \rho_a \omega_i^2 d^4, \ \tau_i = sign(\omega_i) c_\tau \rho_a \omega_i^2 d^5 \tag{78}$$

Here, $\boldsymbol{r}_i$ is the moment arm of the rotor to the center of gravity of the quadrotor, c_f is the thrust coefficient, c_τ is the torque coefficient, ρ_a is the air density, d is the rotor diameter. The state vector of quadrotor model is 12-dimensional, composed by $\boldsymbol{X}_E$, $\boldsymbol{\Psi}$, $\boldsymbol{V}_E$, and $\boldsymbol{\omega}$, the control vector is $\boldsymbol{u} = \left[\frac{\omega_1}{\omega_\mathrm{m}}, \frac{\omega_2}{\omega_\mathrm{m}}, \frac{\omega_3}{\omega_\mathrm{m}}, \frac{\omega_4}{\omega_\mathrm{m}}\right]^T$, which $\omega_\mathrm{m} = 10000$ rev/min is the maximum rotor rotational speed.

The designated flight task for the quadrotor is to reach the destination $\boldsymbol{X}_E = [10\ \mathrm{m}, 0, 10\ \mathrm{m}]^T$ from the origin $\boldsymbol{X}_E = [0, 0, 10\ \mathrm{m}]^T$. The quadrotor is expected to commence and conclude its journey with

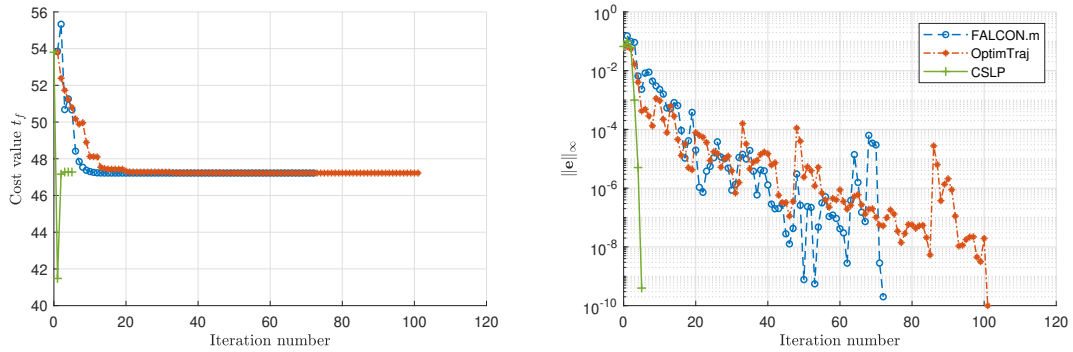

Fig. 7 Comparison of costs and collocation errors in the iterative solutions of aircraft trajectory

zero velocities and angular rates. Each rotor's rotational speed is limited between zero and ω_{m}. A sphere obstacle with a radius of 1 is centered at $[5, 0, 10\ \mathrm{m}]^T$, and the quadrotor must clear it. With this mission and constraints, we seek a flight trajectory that minimizes the total control effort defined by:

$$J = \int_0^{t_f} \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u} \tag{79}$$

To enable the formulation of a linear subproblem, a new state Q is introduced as in Eq. (6):

$$\frac{dQ}{dt} = \frac{1}{2}\boldsymbol{u}^T\boldsymbol{u},\ Q\left(t=0\right) = 0 \tag{80}$$

The cost in Eq. (79) is equivalent to $J = Q\left(t = t_f\right) = Q\left(\tau = 1\right)$, which is a linear function of the extended state vector.

To generate the trajectory using the CSLP algorithm, we adopted $N = 21$ collocation nodes, set $\gamma_\Delta = 0.05$, and applied other parameters as outlined in their default settings in Table 1. Reference trajectories were obtained using *FALCON.m* and *OptimTraj*. All trajectories are compared in Fig. 9, and their time histories of states and controls are recorded in Fig. 10. Given the quadrotor's high degree of freedom, these trajectories exhibit some variation. In all scenarios, the quadrotor successfully cleared the spherical obstacle; the trajectories from *FALCON.m* and *OptimTraj* closely approached the sphere's surface, whereas the CSLP trajectory attained a higher altitude and demonstrated more frequent lateral and directional motions.

Figure 11 displays the cost values and collocation errors throughout the iterative process. Similarly to the fixed-wing aircraft example, the CSLP algorithm solution exhibits a marginally sub-optimal minimum control cost ($Q(t_f) = 3.058$) compared to the reference solutions ($Q(t_f) = 3.00$). Yet, it achieves convergence to a feasible solution in 10 iterations, compared to the 77 and 272 iterations necessitated by *FALCON.m* and *OptimTraj*, respectively. The progression of the CSLP algorithm, depicted in Figure 8, reveals notable convergence characteristics. In the initial two iterations, J_{total} dropped significantly, but ρ was below 0.95 (0.586 and 0.312 respectively), prompting a reduction in the maximum trust region size $\Delta_{\max}$. In the third iteration, J_{total} increased from 1733 to 2245, resulting in a ρ value of -0.2961, hence the trust region radius was further reduced. During the sixth and seventh iterations, the ρ values ranged between 0.95 and 1, indicative of minimal linearization error; however, the maximum trust region size $\Delta_{\max}$ remained unchanged as the step size $\|dz\|_\infty$ did not surpass the trust region radius. As in the previous example, although the maximum allowable trust region remained relatively stable in the latter iterations, the trust regions applied in the actual subproblems demonstrated a progressive decrease to 10^{-5}.

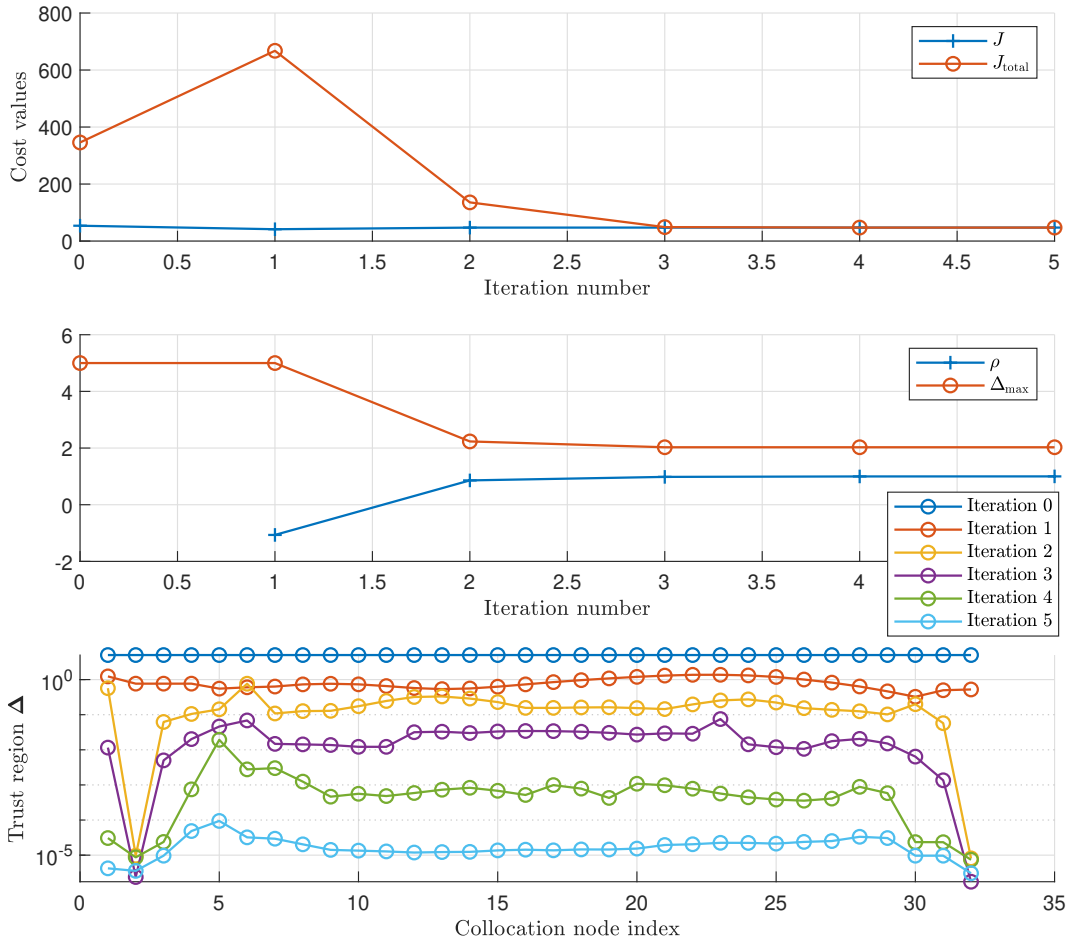

Fig. 8 Costs, ρ, $\Delta_{\max}$, and Δ in the iterative solutions of aircraft trajectory

5.3 Result Discussion

5.3.1 Computation Efforts

The two case studies presented in this section demonstrate the effectiveness of the CSLP trajectory generation algorithm developed in this work. This algorithm can generate flight trajectories that comply with nonlinear flight dynamics and path constraints by iteratively solving a limited number of linear programming subproblems. Although the trajectories generated by the CSLP algorithm exhibit slight sub-optimality in the optimization objectives—approximately 0.1% and 1.9% sub-optimality in the respective applications—they require significantly fewer iterations compared to two state-of-the-art toolboxes that also use the trapezoidal collocation discretization. Furthermore, the computational efficiency of the three approaches was evaluated on a laptop equipped with an AMD Ryzen 5 3500U CPU and 8GB RAM. The results, detailed in Table 2, indicate that the CSLP algorithm reduces the number of iterations required and significantly decreases the computation time. This efficiency is further enhanced when employing a commercial linear programming solver such as *Gurobi* [30]. The CSLP algorithm was implemented in MATLAB, leveraging its user-friendly interface for code development and testing. For future work, we plan to translate the code into a general-purpose programming language to optimize execution speed and resource management.

5.3.2 Effect of Trust Region Penalty Weight

When constructing the linear subproblem in this paper, the sizes of the distributed trust regions are integrated into the cost function via the weight γ_Δ. This penalty term aims to reduce the deviation of the subproblem's solution from the reference solution, thereby controlling the linearization error. It must be

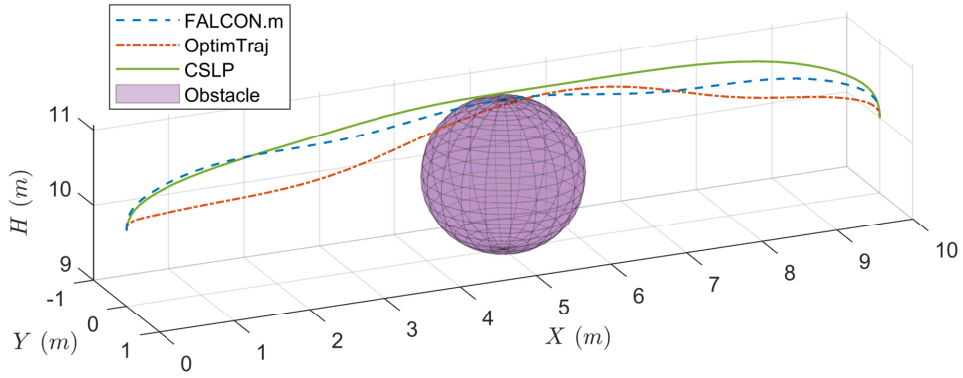

Fig. 9 Minimum control trajectories of the quadrotor

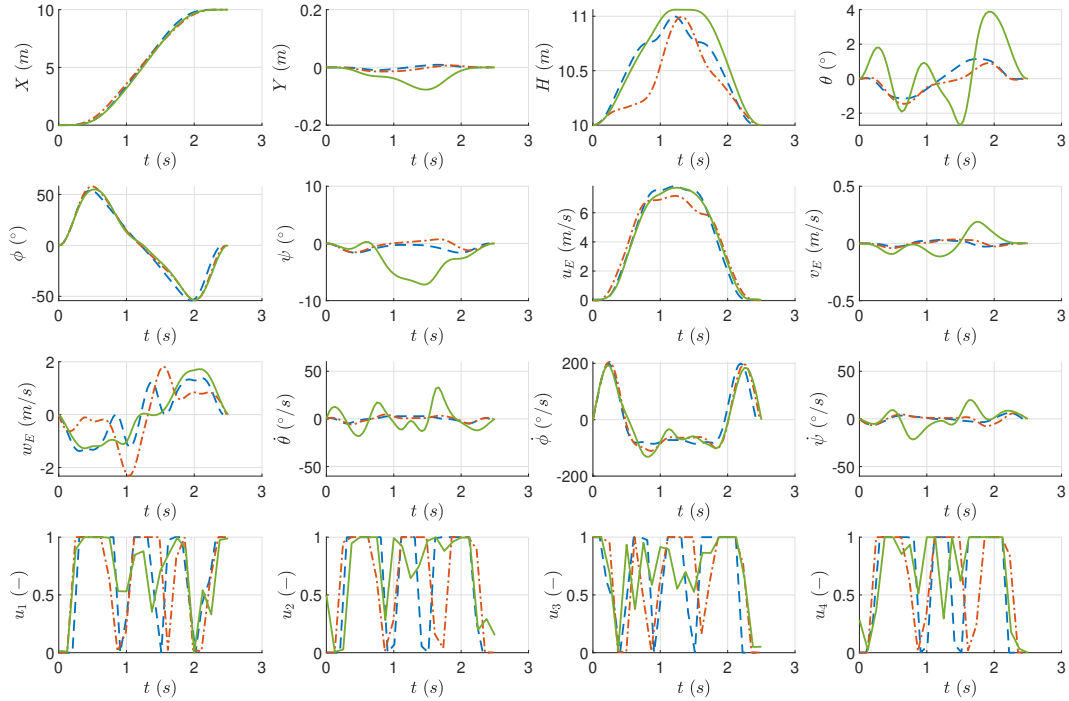

Fig. 10 Time histories of states and controls of the quadrotor

acknowledged, however, that this penalty term modifies the original cost function. This modification can result in sub-optimal solutions compared to the theoretical optimal values, as demonstrated in the two applications discussed in this paper.

To further investigate the influence of the weight γ_Δ on both convergence and sub-optimality, we consider the case of quadrotor trajectory generation. We recorded the iterative solution process under four weight coefficients: 0.01, 0.05, 0.1, and 0.5. The resulting cost values and collocation errors are shown in Fig. 13. Analysis reveals that larger weights γ_Δ expedite the convergence of the iterative process. However, they also tend to increase the sub-optimality of the solution. Consequently, preliminary experiments are recommended to fine-tune the weight when adapting the CSLP algorithm presented in this paper for other flight trajectory generation tasks. In practical applications, a higher weight is advisable as the priority often shifts towards ensuring feasible flight trajectories over achieving absolute performance optimality.

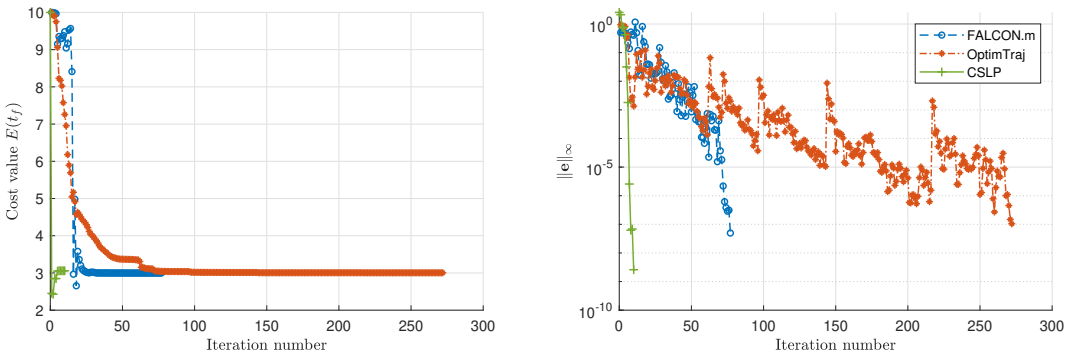

Fig. 11 Costs and collocation errors in the iterative process

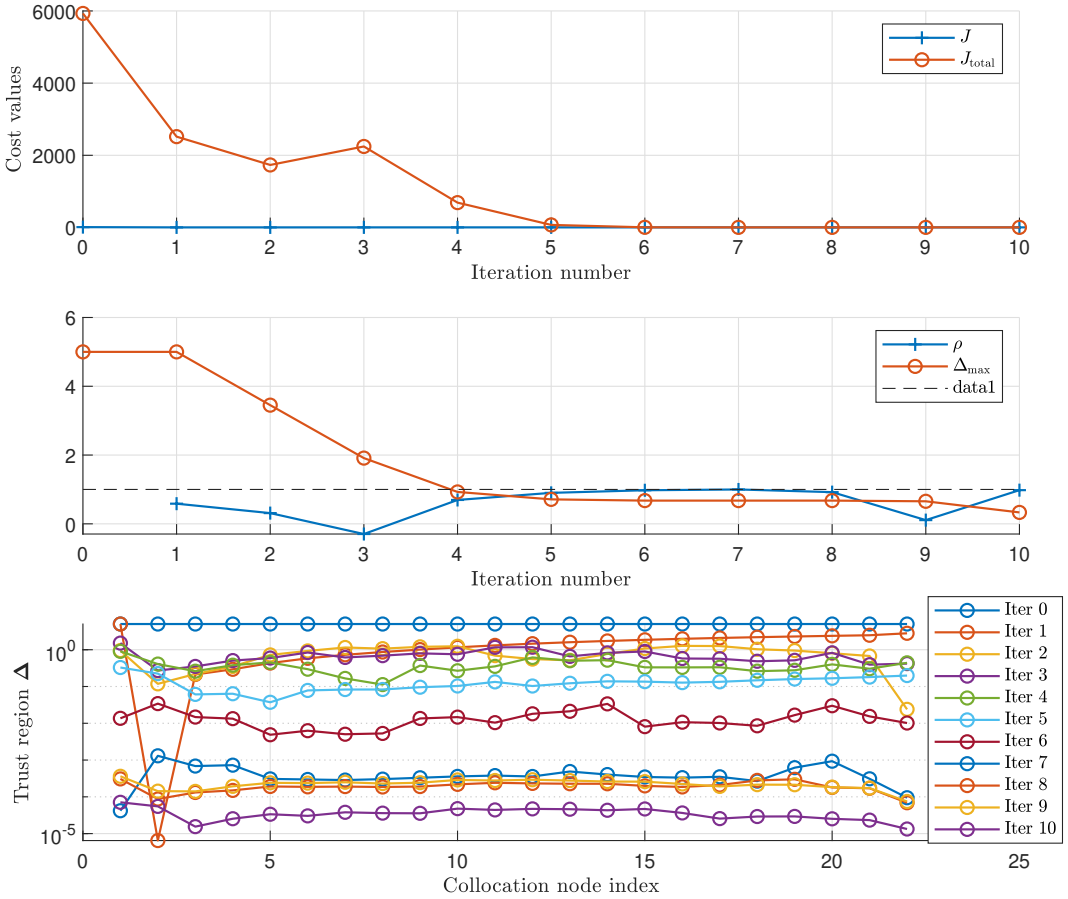

Fig. 12 Costs, ρ, $\Delta_{\max}$, and Δ in the iterative solutions of quadrotor trajectory

6 Conclusion

This study has developed a method integrating successive linear programming into a collocation framework to tackle non-convex trajectory generation challenges. Observing that many trajectory generation tasks involve linear objectives, we strategically linearize nonlinear collocation constraints and path constraints around a reference trajectory, thus creating a linear programming subproblem. This subproblem is resolved iteratively, allowing for continuous refinement of the trajectory. We incorporate linearized constraints into the cost function to address potential infeasibility issues using exact penalty functions. Distributive trust region constraints over parameters and states/controls at all collocation nodes are introduced to ensure the subproblem is unbounded and is a good approximation of the original OCP. These trust regions are implemented as linear inequality constraints in the subproblem. We also penalize the sizes of these trust regions in the cost function via a small weight. Furthermore, a variant of the

Table 2 Computation time required in the two trajectory generation tasks with different solvers

	FALCON.m	OptimTraj	CSLP with matlab *linprog*	CSLP with gorubi *linprog*
Aircraft	5.01 s	14.01 s	0.74 s	0.37 s
Quadrotor	12.23 s	19.22 s	2.86 s	1.35 s

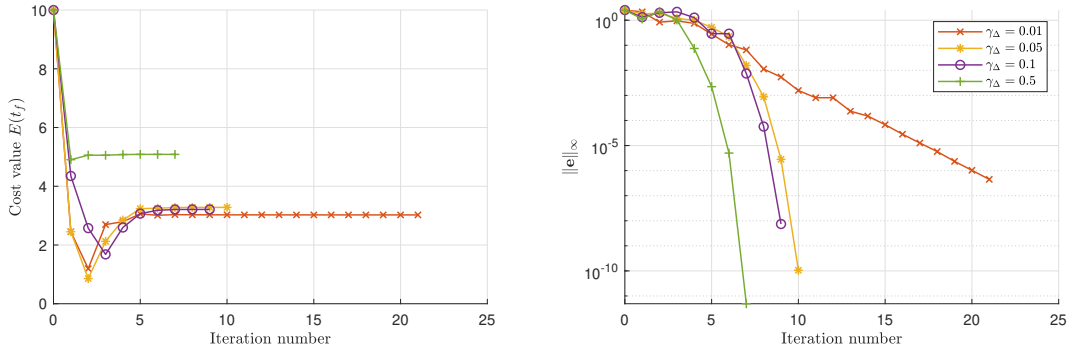

Fig. 13 Costs and collocation errors in quadrotor trajectory iterations under different weight γ_Δ

classical trust-region update rule is applied to adaptively adjust the maximum allowable size of these trust regions in each iteration according to the quantitative measure of linearization error.

Applications of this algorithm to a fixed-wing aircraft and a quadrotor have substantiated its competence. When juxtaposed with state-of-the-art collocation-based toolboxes, our algorithm generates sub-optimal trajectories. Still, it exhibits rapid convergence to feasibility and less computational effort, which is crucial for real-world applications. Tuning trust region penalty weight offers flexibility to balance solution sub-optimality and convergence speed. The code for the CSLP algorithm and two examples presented in this study is available at `https://github.com/lenleo1/Colloc_SLP.git`. Future studies could extend the research to other types of aerial vehicles and to multi-phase trajectory generation tasks.

Acknowledgments

The authors would like to thank Felix Schweighofer for his instruction on the FALCON.m toolbox. Zhidong Lu acknowledges the financial support from the Graduate School of the Technical University of Munich.

References

[1] Jennifer N Wilburn, Mario G Perhinschi, and Brenton K Wilburn. Implementation of composite clothoid paths for continuous curvature trajectory generation for uavs. In *AIAA Guidance, Navigation, and Control (GNC) Conference*, page 5230, 2013. DOI: 10.2514/6.2013-5230.

[2] Mark Owen, Randal W. Beard, and Timothy W. McLain. *Implementing Dubins Airplane Paths on Fixed-Wing UAVs**, pages 1677–1701. Springer Netherlands, Dordrecht, 2015. ISBN: 978-90-481-9707-1. DOI: 10.1007/978-90-481-9707-1_120.

[3] Javier Garcia-Heras, Manuel Soler, and Francisco J Saez. Collocation methods to minimum-fuel trajectory problems with required time of arrival in atm. *Journal of Aerospace Information Systems*, 13(7):243–265, 2016. DOI: 10.2514/1.I010401.

[4] Moritz Diehl, Hans Georg Bock, Holger Diedam, and P-B Wieber. Fast direct multiple shooting algorithms for optimal robot control. *Fast motions in biomechanics and robotics: optimization and feedback control*, pages 65–93, 2006. DOI: 10.1007/978-3-540-36119-0_4.

[5] Behçet Açıkmeşe and Lars Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2):341–347, 2011. DOI: j.automatica.2010.10.037.

[6] Behçet Açıkmeşe, John M Carson, and Lars Blackmore. Lossless convexification of nonconvex control bound and pointing constraints of the soft landing optimal control problem. *IEEE Transactions on Control Systems Technology*, 21(6):2104–2113, 2013. DOI: 10.1109/TCST.2012.2237346.

[7] Xinfu Liu and Ping Lu. Solving nonconvex optimal control problems by convex optimization. *Journal of Guidance, Control, and Dynamics*, 37(3):750–765, 2014. DOI: 10.2514/1.62110.

[8] Yuanqi Mao, Michael Szmuk, and Behçet Açıkmeşe. Successive convexification of non-convex optimal control problems and its convergence properties. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pages 3636–3641. IEEE, 2016. DOI: 10.1109/CDC.2016.7798816.

[9] Zhenbo Wang and Ye Lu. Improved sequential convex programming algorithms for entry trajectory optimization. *Journal of Spacecraft and Rockets*, 57(6):1373–1386, 2020. DOI: 10.2514/1.A34640.

[10] Haichao Hong, Arnab Maity, and Florian Holzapfel. Free final-time constrained sequential quadratic programming–based flight vehicle guidance. *Journal of Guidance, Control, and Dynamics*, 44(1):181–189, 2021. DOI: 10.2514/1.G004874.

[11] Haichao Hong, Patrick Piprek, Matthias Gerdts, and Florian Holzapfel. Computationally efficient trajectory generation for smooth aircraft flight level changes. *Journal of Guidance, Control, and Dynamics*, 44(8):1532–1540, 2021. DOI: 10.2514/1.G005529.

[12] Zhidong Lu, Haichao Hong, and Florian Holzapfel. Directional flight envelope prediction based on convex optimization. In *AIAA Scitech 2022 Forum*, page 1398, 2022. DOI: 10.2514/6.2022-1398.

[13] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. DOI: 10.1017/CBO9780511804441.

[14] Riccardo Bonalli, Abhishek Cauligi, Andrew Bylard, and Marco Pavone. Gusto: Guaranteed sequential trajectory optimization via sequential convex programming. In *2019 International conference on robotics and automation (ICRA)*, pages 6741–6747. IEEE, 2019. DOI: 10.1109/ICRA.2019.8794205.

[15] Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999. ISBN: 0-387-30303-0.

[16] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robotics*, 6(56):eabh1221, 2021. DOI: 10.1126/scirobotics.abh1221.

[17] Xiang Fang, Neng Wan, Hamidreza Jafarnejadsani, Donglei Sun, Florian Holzapfel, and Naira Hovakimyan. Emergency landing trajectory optimization for fixed-wing uav under engine failure. In *AIAA Scitech 2019 Forum*, page 0959, 2019. DOI: 10.2514/6.2019-0959.

[18] Nguyen X Vinh. Optimal trajectories in atmospheric flight. *Space mankind's fourth environment*, pages 449–468, 1982.

[19] Zhidong Lu, Haichao Hong, Johannes Diepolder, and Florian Holzapfel. Maneuverability set estimation and trajectory feasibility evaluation for evtol aircraft. *Journal of Guidance, Control, and Dynamics*, 46(6):1184–1196, 2023. DOI: 10.2514/1.G007109.

[20] Christoph Krammer, Felix Schweighofer, Daniel Gierszewski, Simon Scherer, Tuğba Akman, Haichao Hong, and Florian Holzapfel. Requirements-based generation of optimal vertical takeoff and landing trajectories for electric aircraft. In *33rd Congress of the International Council of the Aeronautical Sciences (ICAS). ICAS*, 2022.

[21] Fernando Palacios-Gomez, L Lasdon, and Michael Engquist. Nonlinear optimization by successive linear programming. *Management science*, 28(10):1106–1120, 1982.

[22] Jianzhong Zhang, Nae-Heon Kim, and L Lasdon. An improved successive linear programming algorithm. *Management science*, 31(10):1312–1331, 1985.

[23] Danylo Malyuta, Taylor P Reynolds, Michael Szmuk, Thomas Lew, Riccardo Bonalli, Marco Pavone, and Behçet Açıkmeşe. Convex optimization for trajectory generation: A tutorial on generating dynamically feasible trajectories reliably and efficiently. *IEEE Control Systems Magazine*, 42(5):40–113, 2022. DOI: 10.1109/MCS.2022.3187542.

[24] Jérome MB Walmag and Éric JM Delhez. A note on trust-region radius update. *SIAM Journal on Optimization*, 16(2):548–562, 2005. DOI: 10.1137/030602563.

[25] Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017. DOI: 10.1137/16M1062569.

[26] Matthias Rieck, Matthias Bittner, Benedikt Grüter, Johannes Diepolder, Patrick Piprek, Christoph Göttlicher, Florian Schwaiger, Barzin Hosseini, Felix Schweighofer, Tuğba Akman, and Florian Holzapfel. Falcon.m user guide, version 1.29, 2023. `http://www.falcon-m.com`.

[27] Florian Holzapfel, Felix Schweighofer, and Tuğba Akman. Lecture notes in practical course optimal control, 2022. `https://www.fsd.ed.tum.de/teaching/pos/`.

[28] Matlab optimization toolbox, 2020. The MathWorks, Natick, MA, USA. `https://www.mathworks.com/products/optimization.html`.

[29] Andreas Wächter and Lorenz T Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical programming*, 106:25–57, 2006. DOI: 10.1007/s10107-004-0559-y.

[30] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. `https://www.gurobi.com`.