Customized Real-Time First-Order Methods for Onboard Dual Quaternion-based 6-DoF Powered-Descent Guidance

Abhinav G. Kamath*, Purnanand Elango*, Taewan Kim*, Skye Mceowen*, Mehran Mesbahi†, & Behçet Açıkmeşe†
University of Washington, Seattle, WA 98195, USA

Yue Yu‡
The University of Texas at Austin, Austin, TX 78712, USA

John M. Carson III§
NASA Johnson Space Center, Houston, TX 77058, USA

The dual quaternion-based 6-DoF powered-descent guidance algorithm (DQG) was selected as the candidate powered-descent guidance algorithm for NASA's Safe and Precise Landing — Integrated Capabilities Evolution (SPLICE) project. DQG is capable of handling state-triggered constraints that are of the utmost importance in terms of enabling technologies such as terrain relative navigation (TRN). In this work, we develop a custom solver for DQG to enable onboard implementation for future rocket landing missions. We describe the design and implementation of a real-time-capable optimization framework, called sequential conic optimization (SeCO), that blends together sequential convex programming and first-order conic optimization to solve difficult nonconvex trajectory optimization problems, such as DQG, in real-time. This framework is entirely devoid of matrix factorizations/inversions, making it suitable for safety-critical applications. Under the hood, the SeCO framework leverages a first-order primal-dual conic optimization solver, based on the proportional-integral projected gradient method (PIPG), that combines the ideas of projected gradient descent and proportional-integral feedback of constraint violation. Unlike other conic optimization solvers, PIPG effectively exploits the sparsity and geometric structure of the constraints, avoids expensive equation solving, and is suitable for both real-time and large-scale applications. We describe the implementation of this solver, and develop customizable first-order methods, including an analytical preconditioning algorithm, to solve the nonconvex DQG optimal control problem in real-time. Strategies such as warm-starting and extrapolation are leveraged to further accelerate convergence. We show that the DQG-customized solver is able to solve the problem significantly faster than other state-of-the-art convex optimization solvers, and thus demonstrate the viability of SeCO for real-time, mission-critical applications onboard computationally constrained flight hardware.

I. Introduction

With robotic and human missions to the Moon and Mars on the horizon, there has been an increased interest in guidance, navigation, and control (GNC) technologies for precision landing [1–6]. Precision landing and hazard avoidance (PL&HA) have been deemed high-priority capabilities by NASA to facilitate missions of exploration to celestial bodies in the solar system [7]. Critical to achieving PL&HA is powered-descent guidance (PDG), which refers to the generation of a feedforward control profile and the corresponding reference state trajectories for powered-descent and landing.

Historically, missions to the Moon, i.e., the Apollo program, and Mars, i.e., the Mars Science Laboratory (MSL) and Mars 2020 missions, made use of polynomial guidance for the powered-descent and landing phase [8–10]. While these missions saw incredible success, the resulting descent and landing trajectories were not propellant optimal. Further, the landing dispersion ellipses for the Apollo missions were prohibitively large (on the order of kilometers) [11] in terms of enabling precision landing, which requires safely landing spacecraft within 100 meters from the target landing site [7].

*Ph.D. Student, William E. Boeing Department of Aeronautics & Astronautics; {agkamath, pelango, twankim, skye95}@uw.edu
†Professor, William E. Boeing Department of Aeronautics & Astronautics; {mesbahi, behcet}@uw.edu
‡Postdoctoral Fellow, Oden Institute for Computational Engineering and Sciences; yueyu@utexas.edu
§Technical Integration Manager – Precision Landing, NASA STMD; john.m.carson@nasa.gov

The advent of convex optimization methods at the turn of the century, along with pivotal results on the lossless convexification (LCvx) of certain nonconvex constraints, led to the invention of a real-time-capable PDG algorithm with strong guarantees [12–14]. This algorithm was successfully flight-tested onboard a terrestrial rocket-powered landing testbed, demonstrating the capability of generating propellant-optimal large divert trajectories in real-time [15, 16]. The problem formulation incorporated a point-mass vehicle model and three degree-of-freedom (3-DoF) translation dynamics. While it was demonstrated that such a guidance algorithm was sufficient to successfully guide and land an inherently 6-DoF vehicle (by using the thrust vector profile as a surrogate for the reference attitude of the vehicle), the algorithm itself was incapable of handling coupled translation and attitude constraints.

Technologies such as terrain relative navigation (TRN) and hazard detection and avoidance (HDA) require constraining the flight envelope in a manner that couples the translation and attitude states. Recent work on 6-DoF powered-descent guidance addresses this problem by formulating state-triggered constraints and encoding them in a continuous optimization framework [17–21]. These are powerful algorithms capable of handling difficult constraints, and have been shown to be amenable to real-time implementation. A real-time implementation of a nonconvex planar landing guidance algorithm has also been proposed [22].

One such state-of-the-art algorithm is the dual quaternion-based powered-descent guidance algorithm (DQG), originally developed by Reynolds, Szmuk, Malyuta, Mesbahi, Açıkmeşe, and Carson III [21], which was chosen as the candidate powered-descent guidance algorithm for NASA's Safe and Precise Landing – Integrated Capabilities Evolution (SPLICE) project, and has been open-loop flight-tested on the Blue Origin New Shepard suborbital rocket onboard the descent and landing computer (DLC) [23–25]. The implementation of the algorithm for these flight tests made use of a customized version of a second-order IPM-based solver called BSOCP, along with a parser interface called CPRS [26, 27]. As noted in [24], that implementation of DQG onboard the DLC takes over 11 seconds to execute (with an optimization horizon length of 20), and close to 6 seconds to execute (with an optimization horizon length of 10), making it prohibitively slow in terms of meeting NASA's guidance update-rate requirements for PL&HA in its current state. Further, the customized solver leads to a very large footprint source code—around 600,000 lines of C code.

In order to tackle the challenges of execution speed and code footprint, first-order optimization algorithms can be used instead of (second-order) IPMs. First-order algorithms typically rely on simple linear algebra operations like matrix-vector multiplications and computation of vector norms at each iteration. Unlike second-order methods, they can avoid factorization of larger matrices and can be warm-started easily. In addition to these benefits, the recently introduced first-order algorithm, the proportional-integral projected gradient (PIPG) method [28, 29], is capable of exploiting the structure of trajectory optimization problems to completely avoid operations on large sparse matrices. As a result, PIPG is readily suitable for onboard, resource-constrained applications.

Recent work on the design and implementation of a customized PIPG solver for the 3-DoF LCvx algorithm demonstrated the capabilities of PIPG in terms of computational efficiency, code footprint, and its viability for easy verification and validation [30]. Moreover, features such as warm-starting and extrapolation [31] for boosting the practical convergence rate have further enhanced PIPG's capabilities as a conic optimization solver for use within sequential convex programming (SCP) algorithms [32]. In fact, SCP algorithms can be specialized to harness the approach taken in PIPG to solve conic optimization problems, as shown in [33], with the sequential conic optimization (SeCO) framework that is entirely devoid of matrix factorizations and inversions.

In this work, we demonstrate a real-time-capable implementation of SeCO for solving the nonconvex optimal control problem in DQG. Sections II, III, IV, and V describe the problem formulation and optimization algorithms, Section VI delves into solver customization, and Section VII provides the numerical results.

Many of the following fundamental mathematical definitions and formalisms can be found in [21, 34], and [33], and are presented here for the sake of completeness. The operations pertaining to quaternion and dual quaternion algebra are documented in the Appendix.

II. Optimal control problem formulation

A. Equations of motion

One of the defining characteristics of DQG is the representation of the 6-DoF equations of motion using unit dual quaternions, which provide for an elegant parameterization of the dynamics by naturally coupling the translational and rotational states and enabling the representation of certain key operational constraints, such as the line-of-sight constraint, as convex constraints (in theory) [21, 35]. Ultimately, however, the use of the dual quaternion parameterization is a design choice, and other parameterizations can be adopted as well, such as Cartesian coordinates for the translational states and unit quaternions for attitude [20]. The interested reader is referred to [21, 36–38] for detailed descriptions of parameterizing rigid body dynamics via unit dual quaternions.

1. States

The state vector is 15-dimensional, and consists of mass, m, the 8-dimensional unit dual quaternion that couples translation and attitude, $\boldsymbol{q}$, and the (reduced-order) 6-dimensional dual velocity, $\boldsymbol{\omega}$, as shown in Equations 1e. $\widetilde{\boldsymbol{\omega}}$ represents the 8-dimensional dual velocity, in which the fourth and eighth terms are zero. q is the attitude (unit) quaternion. In this work, we adopt the scalar-last convention to represent quaternions. See the Appendix for definitions of unit quaternion and unit dual quaternion. The subscripts $\mathcal{I}$ and $\mathcal{B}$ denote that the quantity in question is expressed in the inertial frame or the body frame, respectively. Further, $r \in \mathbb{R}^3$ is the position, $\omega \in \mathbb{R}^3$ is the angular velocity, and $v \in \mathbb{R}^3$ is the velocity.

$$m \in \mathbb{R} \tag{1a}$$

$$\boldsymbol{q} := \begin{pmatrix} q \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}} \\ 0 \end{pmatrix} \otimes q \end{pmatrix} = \begin{pmatrix} q \\ \frac{1}{2} q \otimes \begin{pmatrix} r_{\mathcal{B}} \\ 0 \end{pmatrix} \end{pmatrix} \in \mathbb{R}^8 \tag{1b}$$

$$\widetilde{\boldsymbol{\omega}} := \begin{pmatrix} \begin{pmatrix} \omega_{\mathcal{B}} \\ 0 \end{pmatrix} \\ q^* \otimes \begin{pmatrix} v_{\mathcal{I}} \\ 0 \end{pmatrix} \otimes q \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \omega_{\mathcal{B}} \\ 0 \end{pmatrix} \\ \begin{pmatrix} v_{\mathcal{B}} \\ 0 \end{pmatrix} \end{pmatrix} \in \mathbb{R}^8 \tag{1c}$$

$$\boldsymbol{\omega} := \begin{pmatrix} \omega_{\mathcal{B}} \\ v_{\mathcal{B}} \end{pmatrix} \in \mathbb{R}^6 \tag{1d}$$

$$x := \begin{pmatrix} m \\ \boldsymbol{q} \\ \boldsymbol{\omega} \end{pmatrix} \in \mathbb{R}^{15} \tag{1e}$$

2. Controls

The control input vector is 6-dimensional, with three parameters describing the thrust vector: the thrust magnitude, $T \in \mathbb{R}$, the gimbal deflection angle, $\delta \in \mathbb{R}$, and the gimbal azimuth angle, $\phi \in \mathbb{R}$, and a 3-dimensional body torque vector, $\tau \in \mathbb{R}^3$, as shown in Equations 2. The thrust vector is effected by means of a gimbaled main engine and the torque input is assumed to be effected by means of reaction control system (RCS) thrusters.

We choose to parameterize the thrust vector in terms of spherical coordinates for the following reasons: (1) all the control constraints in DQG become naturally convex, and hence, in combination with the first-order hold (FOH) parameterization (described in Subsection III.B), intersample satisfaction of the control constraints is guaranteed; (2) the control rate constraints can be imposed exactly; and, (3) with a mild assumption, both the magnitude and rate constraints (throttle and gimbaling) can be combined and made projection-friendly, which is beneficial in terms of implementing the solver, as described in Subsection IV.E. Further, with the spherical coordinate parameterization, we note that the thrust magnitude solution possesses a piecewise-affine profile, which will not be the case if the Cartesian coordinate parameterization is adopted.

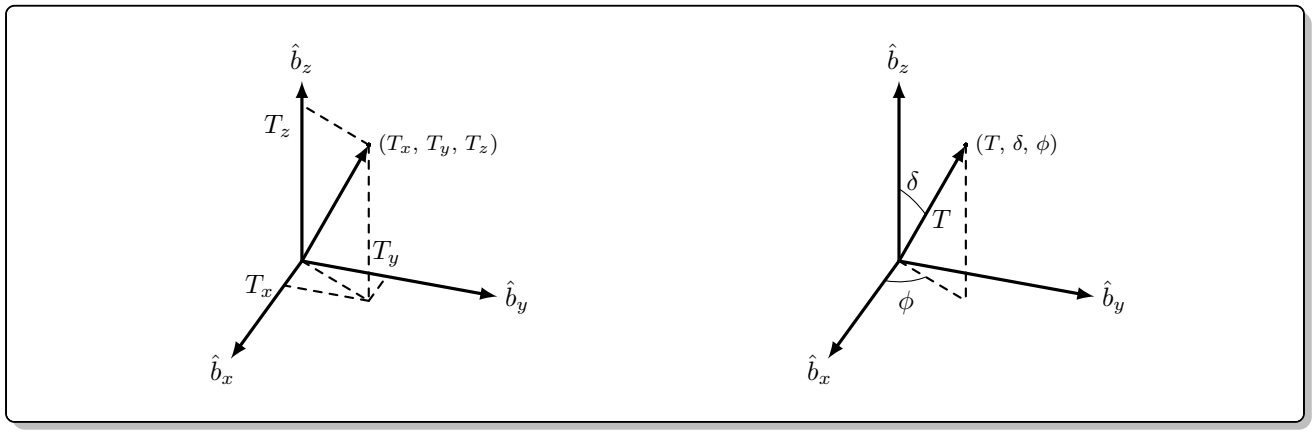

Fig. 1 Parameterization of the thrust vector (expressed in the body frame) in terms of Cartesian coordinates (left) and spherical coordinates (right). In this work, we use the latter.

$$\mathcal{T}_{\mathcal{B}} = \begin{pmatrix} T_x \\ T_y \\ T_z \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} = \begin{pmatrix} T\sin\delta\cos\phi \\ T\sin\delta\sin\phi \\ T\cos\delta \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} \in \mathbb{R}^6 \tag{2a}$$

$$u = \begin{pmatrix} T \\ \delta \\ \phi \\ \tau_x \\ \tau_y \\ \tau_z \end{pmatrix} \in \mathbb{R}^6 \tag{2b}$$

T: thrust magnitude
δ: gimbal deflection angle defined from the body vertical (z) axis
ϕ: gimbal azimuth angle defined from the body x-axis
τ_x, τ_y, τ_z: body torque inputs

Here, $\mathcal{T}_{\mathcal{B}}$ is the wrench vector expressed in the body frame [17], and u is the control input vector. The convexity and simplicity of the resulting control constraints come at the cost of additional trigonometric nonlinearities in the dynamics, as shown in Equation 2a.

3. Mass-depletion

In addition to accounting for thrust due to the main engine, we consider the effect of thrust due to the RCS thrusters on mass-depletion. In order to do so, we assume that two diagonally opposite RCS thrusters fire at any given instant to achieve the desired net torque, such that the thrust due to each thruster is orthogonal to the body vertical (z) axis.

Assuming the mass-center of the vehicle is equidistant from the top-mounted RCS thrusters and the bottom-mounted RCS thrusters/gimbaled main engine, the mass-depletion dynamics can be given as shown in Equation 3. Here, $\alpha_{\text{ME}} \in \mathbb{R}_+$ and $\alpha_{\text{RCS}} \in \mathbb{R}_+$ are the thrust-specific fuel consumption (TSFC) parameters for the main engine and an RCS thruster, respectively, and $l_{\text{CM}} \in \mathbb{R}_{++}$ is the length of the moment-arm of the vehicle, i.e., the distance between the mass-center of the vehicle and the bottom-mounted RCS thrusters/gimbaled main engine.

$$\dot{m} = -\left(\alpha_{\text{ME}}\, T + \alpha_{\text{RCS}} \frac{\|\tau\|_2}{l_{\text{CM}}}\right) \tag{3}$$

4. Kinematics

This dual quaternion kinematic equation requires the use of the 8-dimensional dual velocity vector, where the fourth and eighth terms are zero, and is given by Equation 4.

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \widetilde{\boldsymbol{\omega}} \tag{4}$$

5. Dynamics

For the dynamics, we assume that the moment of inertia is a linear function of the vehicle mass (as opposed to assuming a constant value as in [21]), and thereby account for the effect of mass-depletion on the attitude of the vehicle. The moment of inertia can be assumed to be an affine function of the vehicle mass as well, if required [18]. The dynamics can be given by Equation 5, where $J \in \mathbb{S}^3_{++}$ is the inertia tensor of the vehicle about its mass-center, $l = [0, 0, -l_{\text{CM}}]^\top$ is the body-fixed moment-arm vector, and $g \in \mathbb{R}_+$ is the acceleration due to gravity at the celestial body under consideration.

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}\left[\begin{pmatrix} 0_{3\times3} & -\omega_{\mathcal{B}}^{\times} \\ -\omega_{\mathcal{B}}^{\times} & 0_{3\times3} \end{pmatrix} \boldsymbol{J}\boldsymbol{\omega} + \Phi\mathcal{T}_{\mathcal{B}} + m\boldsymbol{g}_{\mathcal{B}}\right] \tag{5}$$

where

$$\boldsymbol{J} := \left(\begin{array}{c|c} 0_{3\times3} & mI_3 \\ \hline mJ & 0_{3\times3} \end{array}\right)_{6\times6} \qquad \Phi := \left(\begin{array}{c|c} I_3 & 0_{3\times3} \\ \hline l^{\times} & I_3 \end{array}\right)_{6\times6} \qquad \boldsymbol{g}_{\mathcal{B}} := \left(\begin{array}{c} g_{\mathcal{B}} \\ \hline 0_{3\times1} \end{array}\right)_{6\times1}$$

$$g_{\mathcal{I}} := [0,\, 0,\, -g]^{\top}$$

$$\begin{pmatrix} g_{\mathcal{B}} \\ 0 \end{pmatrix} = q^{*} \otimes \begin{pmatrix} g_{\mathcal{I}} \\ 0 \end{pmatrix} \otimes q$$

B. Control constraints

All the components of the thrust vector are bounded, as shown in Equations 6, where $T_{\min} \in \mathbb{R}_{++}$ and $T_{\max} \in \mathbb{R}_{++}$ are the lower and upper bounds on the thrust magnitude, respectively, and $\delta_{\max} \in \mathbb{R}_{++}$ is the upper bound on the gimbal deflection angle. Further, they are rate-limited, as shown in Equations 7, where $\dot{T}_{\max} \in \mathbb{R}_{++}$, $\dot{\delta}_{\max} \in \mathbb{R}_{++}$, and $\dot{\phi}_{\max} \in \mathbb{R}_{++}$ are the rate-limits on the thrust magnitude, the gimbal deflection angle, and the gimbal azimuth angle, respectively. An upper bound is levied on the magnitude of the body torque input about each body axis, as shown in 8. We emphasize that every single control constraint is naturally convex, owing to the spherical coordinate parameterization.

1. Thrust vector bounds

$$T_{\min} \leq T(t) \leq T_{\max} \tag{6a}$$
$$0 \leq \delta(t) \leq \delta_{\max} \tag{6b}$$
$$0 \leq \phi(t) \leq 2\pi \tag{6c}$$

2. Thrust vector rate-limits

$$\left|\dot{T}(t)\right| \leq \dot{T}_{\max} \tag{7a}$$
$$\left|\dot{\delta}(t)\right| \leq \dot{\delta}_{\max} \tag{7b}$$
$$\left|\dot{\phi}(t)\right| \leq \dot{\phi}_{\max} \tag{7c}$$

3. Torque bounds

$$\|\tau(t)\|_{\infty} \leq \tau_{\max} \tag{8}$$

C. State constraints

1. Global state constraints

Global state constraints involve constraints that are imposed on the state over the entire time-horizon. These constraints include a maximum tilt constraint, a maximum angular body rate constraint, a maximum speed constraint, and a minimum altitude constraint, as shown in Equations 9, respectively. Here, $t_f \in \mathbb{R}_{+}$ is the time-of-flight, $\theta_{\max} \in \mathbb{R}_{+}$ is the maximum tilt angle from the inertial vertical (z) axis, $\omega_{\max} \in \mathbb{R}_{+}$ is the maximum angular speed about any body axis, $v_{\max} \in \mathbb{R}_{+}$ is the maximum speed, and $h_{\min} \in \mathbb{R}_{+}$ is the minimum altitude. $z_{\mathcal{I}} := [0, 0, 1]^{\top}$ and $\widetilde{z}_{\mathcal{I}} := [z_{\mathcal{I}}^{\top}, 0]^{\top}$ (pure quaternion).

$$\forall t \in [0, t_f)\,,$$
$$\|\boldsymbol{q}^{[1:2]}(t)\|_2 \leq \sin\frac{\theta_{\max}}{2} \tag{9a}$$
$$\|\boldsymbol{\omega}^{[1:3]}(t)\|_{\infty} \leq \omega_{\max} \tag{9b}$$
$$\|\boldsymbol{\omega}^{[4:6]}(t)\|_2 \leq v_{\max} \tag{9c}$$
$$\boldsymbol{q}(t)^{\top} M_g\, \boldsymbol{q}(t) \geq h_{\min} \tag{9d}$$

where

$$M_g := \begin{pmatrix} \mathbf{0}_{4\times4} & [\widetilde{z}_{\mathcal{I}}]_{\otimes}^{\top} \\ [\widetilde{z}_{\mathcal{I}}]_{\otimes} & \mathbf{0}_{4\times4} \end{pmatrix}$$

2. State-triggered constraints

State-triggered constraints (STCs) include the constraints that are to be activated only when the vehicle is within the prescribed trigger window. Here, we consider slant-range-based triggering, and tightly constrain the body tilt angle, angular body rates, maximum speed, and the maximum line-of-sight angle to the target landing site—which is assumed to be at the origin, without loss of generality—as shown in Equations 10, respectively. These constraints are imposed to enable accurate scans of the potential landing site during descent using the hazard detection LiDAR (HDL) [39, 40], for instance, and to initiate diverts if necessary.

Here, $\rho_{\max} \in \mathbb{R}_{++}$ and $\rho_{\min} \in \mathbb{R}_{++}$ are the maximum (activation) and minimum (deactivation) trigger distances from the target landing site, respectively; $\theta_{\mathrm{STC}_{\max}}$, $\omega_{\mathrm{STC}_{\max}}$, and $v_{\mathrm{STC}_{\max}}$ are assumed to be smaller than their counterparts in Equations 9; $\mu_{\mathrm{STC}_{\max}} \in \mathbb{R}_{+}$ is the maximum line-of-sight angle; and $p_{\mathcal{B}} \in \mathbb{R}^3$ is a unit vector in the body frame that represents the body-fixed sensor-pointing direction.

$$\forall t \ni \rho_{\min} \leq \|2\,\boldsymbol{q}^{[5:8]}(t)\|_2 \leq \rho_{\max},$$

$$\|\boldsymbol{q}^{[1:2]}(t)\|_2 \leq \sin\frac{\theta_{\mathrm{STC}_{\max}}}{2} \tag{10a}$$

$$\|\boldsymbol{\omega}^{[1:3]}(t)\|_\infty \leq \omega_{\mathrm{STC}_{\max}} \tag{10b}$$

$$\|\boldsymbol{\omega}^{[4:6]}(t)\|_2 \leq v_{\mathrm{STC}_{\max}} \tag{10c}$$

$$\boldsymbol{q}(t)^\top M_l\,\boldsymbol{q}(t) + \|2\,\boldsymbol{q}^{[5:8]}(t)\|_2 \cos\mu_{\mathrm{STC}_{\max}} \leq 0 \tag{10d}$$

where

$$M_l := \begin{pmatrix} \mathbf{0}_{4\times4} & [\widetilde{p}_{\mathcal{B}}]_{\otimes}^{*\top} \\ [\widetilde{p}_{\mathcal{B}}]_{\otimes}^{*} & \mathbf{0}_{4\times4} \end{pmatrix}$$

Here, $\widetilde{p}_{\mathcal{B}} := [p_{\mathcal{B}}^\top, 0]^\top$ (pure quaternion). We choose not to impose the minimum altitude constraint in the trigger window, with the observation that the simultaneous satisfaction of the maximum tilt and maximum line-of-sight angle constraints implicitly precludes subsurface solutions if $\theta_{\mathrm{STC}_{\max}} \leq \frac{\pi}{2} - \mu_{\mathrm{STC}_{\max}} - \gamma_{\mathrm{boresight}}$, where $\gamma_{\mathrm{boresight}}$ is the angle made by the body-fixed sensor with the body vertical (z) axis, i.e., the (acute) angle between $p_{\mathcal{B}}$ and $\hat{b}_z$.

3. Initial conditions

The initial condition constraints are given by Equations 11, where $m_i \in \mathbb{R}_{++}$ is the initial mass of the vehicle, $q_i \in \mathbb{R}_u^4$ is the initial attitude quaternion, $r_{\mathcal{I}_i} \in \mathbb{R}^3$ is the initial position expressed in the inertial frame, $\omega_{\mathcal{B}_i} \in \mathbb{R}^3$ is the initial angular velocity expressed in the body frame, and $v_{\mathcal{I}_i} \in \mathbb{R}^3$ is the initial velocity expressed in the inertial frame.

$$m(0) = m_i \tag{11a}$$

$$\boldsymbol{q}(0) = \begin{pmatrix} q_i \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \tag{11b}$$

$$\boldsymbol{\omega}(0) = \begin{pmatrix} \omega_{\mathcal{B}_i} \\ q_i^* \otimes \begin{pmatrix} v_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \tag{11c}$$

4. Terminal conditions

The terminal conditions given by Equations 12 subsume the following details. At the final node: (1) the vehicle is upright (zero pitch and yaw); (2) the roll is free; (3) the angular body rates are zero; and, (4) the (inertial) horizontal components of velocity are zero. By infusing these details into the expressions rather than treating them as problem parameters, the terminal condition constraints are rendered convex. Here, $m_f \in \mathbb{R}_{++}$ is the final mass of the vehicle, $r_{\mathcal{I}_f} \in \mathbb{R}^3$ is the final position expressed in the inertial frame, and $v_{z_{\mathcal{I}_f}} \in \mathbb{R}$ is the final velocity along the inertial vertical (z) axis. If desired, the entire final dual quaternion can be fixed as well.

$$m(t_f) \geq m_f \tag{12a}$$

$$\boldsymbol{q}^{[1:2]}(t_f) = \mathbf{0}_{2\times1} \tag{12b}$$

$$\left[\left(\tfrac{1}{2}[r_{\mathcal{I}_f}]_{\otimes}\begin{pmatrix}\mathbf{0}_{2\times2}\\ I_2\end{pmatrix}\right) \quad -I_4\right]_{4\times6} \boldsymbol{q}^{[3:8]}(t_f) = \mathbf{0}_{4\times1} \tag{12c}$$

$$\widetilde{\boldsymbol{\omega}}(t_f) = \begin{pmatrix} \mathbf{0}_{4\times 1} \\ \begin{pmatrix} \mathbf{0}_{2\times 1} \\ q^{[3:4]}(t_f) \end{pmatrix}^* \otimes \begin{pmatrix} \mathbf{0}_{2\times 1} \\ v_{z_{\mathcal{I}_f}} \\ 0 \end{pmatrix} \otimes \begin{pmatrix} \mathbf{0}_{2\times 1} \\ q^{[3:4]}(t_f) \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \mathbf{0}_{4\times 1} \\ \mathbf{0}_{2\times 1} \\ v_{z_{\mathcal{I}_f}} \\ 0 \end{pmatrix} \tag{12d}$$

$$\therefore \boldsymbol{\omega}(t_f) = \begin{pmatrix} \mathbf{0}_{5\times 1} \\ v_{z_{\mathcal{I}_f}} \end{pmatrix} \tag{12e}$$

D. The continuous-time nonconvex optimal control problem

$$\begin{aligned}
&\underset{t_f,\, u(t)}{\text{minimize}} && -m(t_f) \\
&\text{subject to} && \forall t \in [0, t_f) \\
&\boxed{\text{Dynamics}} && \dot{x}(t) = f(t, x(t), u(t)) \\
&\boxed{\text{Control constraints}} && T_{\min} \le T(t) \le T_{\max} \\
& && 0 \le \delta(t) \le \delta_{\max} \\
& && 0 \le \phi(t) \le 2\pi \\
& && \left|\dot{T}(t)\right| \le \dot{T}_{\max} \\
& && \left|\dot{\delta}(t)\right| \le \dot{\delta}_{\max} \\
& && \left|\dot{\phi}(t)\right| \le \dot{\phi}_{\max} \\
& && \|\tau(t)\|_\infty \le \tau_{\max} \\
&\boxed{\text{Global state constraints}} && \|\boldsymbol{q}^{[1:2]}(t)\|_2 \le \sin\frac{\theta_{\max}}{2} \\
& && \|\boldsymbol{\omega}^{[1:3]}(t)\|_\infty \le \omega_{\max} \\
& && \|\boldsymbol{\omega}^{[4:6]}(t)\|_2 \le v_{\max} \\
& && \boldsymbol{q}(t)^\top M_g\, \boldsymbol{q}(t) \ge h_{\min} \\
&\boxed{\text{State-triggered constraints}} && \|\boldsymbol{q}^{[1:2]}(t)\|_2 \le \sin\frac{\theta_{\text{STC}_{\max}}}{2} \\
&\forall t \ni \rho_{\min} \le \|2\,\boldsymbol{q}^{[5:8]}(t)\|_2 \le \rho_{\max} && \|\boldsymbol{\omega}^{[1:3]}(t)\|_\infty \le \omega_{\text{STC}_{\max}} \\
& && \|\boldsymbol{\omega}^{[4:6]}(t)\|_2 \le v_{\text{STC}_{\max}} \\
& && \boldsymbol{q}(t)^\top M_l\, \boldsymbol{q}(t) + \|2\,\boldsymbol{q}^{[5:8]}(t)\|_2 \cos\mu_{\text{STC}_{\max}} \le 0 \\
&\boxed{\text{Initial conditions}} && m(0) = m_i \\
& && \boldsymbol{q}(0) = \begin{pmatrix} q_i \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \\
& && \boldsymbol{\omega}(0) = \begin{pmatrix} \omega_{\mathcal{B}_i} \\ q_i^* \otimes \begin{pmatrix} v_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \\
&\boxed{\text{Terminal conditions}} && m(t_f) \ge m_f \\
& && \boldsymbol{q}^{[1:2]}(t_f) = \mathbf{0}_{2\times 1} \\
& && \left[\left(\tfrac{1}{2}[r_{\mathcal{I}_f}]_\otimes \begin{pmatrix} \mathbf{0}_{2\times 2} \\ I_2 \end{pmatrix}\right) \quad -I_4\right]_{4\times 6} \boldsymbol{q}^{[3:8]}(t_f) = \mathbf{0}_{4\times 1} \\
& && \boldsymbol{\omega}(t_f) = \begin{pmatrix} \mathbf{0}_{5\times 1} \\ v_{z_{\mathcal{I}_f}} \end{pmatrix}
\end{aligned}$$

The continuous-time nonconvex optimal control problem is given by Subsection II.D, where we minimize propellant consumption (by maximizing the final mass of the vehicle).

III. Dynamics

Our approach to treating the dynamics closely follows the methods provided in [20–22], with some key distinctions, such as the inverse-free propagation step, as described later in this section. All of these approaches yield an *exact* discretization of the linear time-varying (LTV) system under consideration—which means that the discrete-time trajectory exactly coincides with the continuous-time trajectory at the temporal nodes—and are analytically equivalent. In practice, however, the approach we propose herein leads to a much simpler implementation, without the need for any matrix factorizations/inversions, thus also making the implementation more numerically stable and reliable. Further, a very similar approach, albeit in a multi-phase guidance setting, is provided in [33].

A. Time-dilation

The original nonlinear dynamics over the entire time-horizon are given by Equation 13. Without loss of generality, we assume that the initial time is zero.

$$\dot{x}(t) = f(t, x(t), u(t)), \;\; t \in [0, t_f) \tag{13}$$

Now, we define an invertible linear map, $\tau : [0, t_f) \to [0, 1)$, as shown in Equation 14.

$$\tau(t) = \frac{t}{t_f^-}, \;\; t \in [0, t_f) \tag{14}$$

This mapping is referred to as *time-dilation*, as it dilates (normalizes) the wall-clock time-horizon to a chosen fixed interval—$[0, 1)$, in our case. Next, we invoke the chain-rule to obtain Equation 15, where $\mathring{\square}$ denotes the derivative operator with respect to the dilated time, τ.

$$\begin{aligned} \mathring{x}(t) = \frac{d}{d\tau}\, x(t) = \frac{dt}{d\tau}\, \frac{d}{dt}\, x(t) = \frac{dt}{d\tau}\, \dot{x}(t) = \underbrace{t_f^-}_{s}\, \dot{x}(t) \\ = s\, f(t, x(t), u(t)) := F(t, x(t), u(t), s) \end{aligned} \tag{15}$$

The multiplier in Equation 15, $s := t_f^- \in \mathbb{R}_+$, termed the *dilation factor*, evaluates to the time-of-flight. Given Equations 14 and 15, t can now be expressed as a function of τ, i.e., $t(\tau) = s\,\tau$, $\tau \in [0, 1)$. Henceforth, we treat τ as the independent variable instead of t, and replace the temporal argument, $t = t(\tau)$, by τ, for notational simplicity.

B. Linearization

We begin by considering nonlinear systems that can be represented as ordinary differential equations (ODEs), as given by Equation 16, where $x(\cdot) \in \mathbb{R}^{n_x}$ is the state vector, $u(\cdot) \in \mathbb{R}^{n_u}$ is the control input vector, $s \in \mathbb{R}$ is the parameter, which in our case, is the time-of-flight, and $F(\cdot) \in \mathbb{R} \times \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \times \mathbb{R} \to \mathbb{R}^{n_x}$ is the nonlinear function representing the time-dilated dynamics, which is assumed to be at least once continuously differentiable.

$$\mathring{x}(\tau) = F(\tau, x(\tau), u(\tau), s) \tag{16}$$

The first-order Taylor series expansion of Equation 16 about an arbitrary reference trajectory $(\overline{x}(\tau), \overline{u}(\tau), \overline{s})$ yields a linear time-varying (LTV) system, given by Equation 17. The equation is approximate as a consequence of linearization via truncation of the higher-order (≥ 2) terms in the Taylor series expansion.

$$\mathring{x}(\tau) \approx A(\tau)x(\tau) + B(\tau)u(\tau) + S(\tau)s + d(\tau) \tag{17}$$

where

$$A(\tau) := \nabla_x F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \tag{18a}$$

$$B(\tau) := \nabla_u F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \tag{18b}$$

$$S(\tau) := \nabla_s F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \tag{18c}$$

$$\begin{aligned} d(\tau) := {} & F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s}) \\ & - A(\tau)\,\overline{x}(\tau) - B(\tau)\,\overline{u}(\tau) - S(\tau)\,\overline{s} \end{aligned} \tag{18d}$$

We adopt a first-order hold (FOH) parameterization of the control input signal, which, in contrast to pseudospectral discretization methods, possesses the following characteristics: (1) inter-sample satisfaction of the convex control constraints is guaranteed (provided they are satisfied at the discrete temporal nodes); and, (2) the resulting conic subproblem has a sparsity pattern that is amenable to real-time implementation [20, 41]. These properties make FOH attractive for optimal control applications.

With FOH, the control input variables are only defined at the discrete temporal nodes, and the continuous-time control input signal is obtained by linearly interpolating between the discrete values at successive nodes. Note that the control input signal is restricted to a continuous piecewise-affine function of time, with only a finite number (N) of discrete control variables, as shown in Equation 19.

$$u(\tau) = \sigma_k^-(\tau)\, u_k + \sigma_k^+(\tau)\, u_{k+1}, \quad \forall \tau \in [\tau_k, \tau_{k+1}) \tag{19}$$

where

$$\sigma_k^-(\tau) := \frac{\tau_{k+1} - \tau}{\tau_{k+1} - \tau_k}, \quad \sigma_k^+(\tau) := \frac{\tau - \tau_k}{\tau_{k+1} - \tau_k}, \quad k = 1{:}N{-}1$$

As shown in Equation 20, the LTV dynamics in terms of deviations from the reference can now be written using the piecewise-affine control input parameterization given by Equation 19. $\Delta\square$ denotes the deviation of a quantity from its reference, i.e., $\Delta\square := \square - \overline{\square}$, and $\Delta\mathring{x}(\tau) := \mathring{x}(\tau) - F(\tau, \overline{x}(\tau), \overline{u}(\tau), \overline{s})$. Henceforth, "=" is used in lieu of "≈" for notational simplicity, with the understanding that the LTV system is a first-order approximation of the original nonlinear system.

$$\Delta\mathring{x}(\tau) = A(\tau)\,\Delta x(\tau) + B(\tau)\,\sigma_k^-(\tau)\,\Delta u_k + B(\tau)\,\sigma_k^+(\tau)\,\Delta u_{k+1} + S(\tau)\,\Delta s \tag{20}$$

Equation 20 has a unique solution [32, 42], given by Equation 21, $\forall \tau \in [\tau_k, \tau_{k+1})$.

$$\Delta x(\tau) = \Phi(\tau, \tau_k)\,\Delta x(\tau_k) + \int_{\tau_k}^{\tau} \Phi(\tau, \zeta)\left\{B(\zeta)\,\sigma_k^-(\zeta)\,\Delta u_k + B(\zeta)\,\sigma_k^+(\zeta)\,\Delta u_{k+1} + S(\zeta)\,\Delta s\right\} \mathrm{d}\zeta \tag{21}$$

where $\Phi(\tau, \tau_k)$ is a matrix that satisfies the matrix differential equation given by Equation 22 such that $\Phi(\tau_k, \tau_k) = I_{n_x}$, and is called the *state transition matrix* (STM).

$$\mathring{\Phi}(\tau, \tau_k) = A(\tau)\,\Phi(\tau, \tau_k) \tag{22}$$

C. Discretization

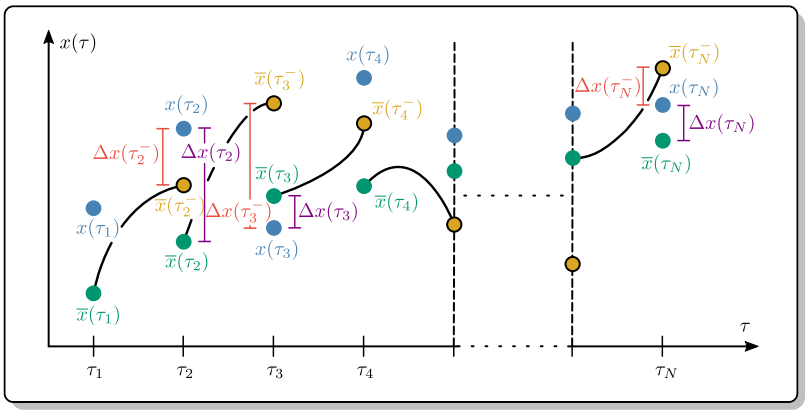

Fig. 2 Propagation of the state in the discretization procedure.

Evaluating Equation 21 at $\tau = \tau_{k+1}^-$, we get Equation 23.

$$\Delta x(\tau_{k+1}^-) = A_k \Delta x(\tau_k) + B_k^- \Delta u_k + B_k^+ \Delta u_{k+1} + S_k \Delta s \tag{23}$$

Note that Equation 24 holds, as is evident from Figure 2. We refer to this equation as the *stitching condition*.

$$\Delta x(\tau_{k+1}^-) + \overline{x}(\tau_{k+1}^-) = \Delta x(\tau_{k+1}) + \overline{x}(\tau_{k+1}) = x(\tau_{k+1}), \quad k = 1{:}N{-}1 \tag{24}$$

The discretized dynamics can now be written as shown in Equation 25, where $\Delta x_k := \Delta x(\tau_k)$, $\Delta x(\tau_1) := 0$, $\Delta x_{k+1} := \Delta x(\tau_{k+1})$, $x_{k+1}^{\text{prop}} := \overline{x}(\tau_{k+1}^-)$, $\overline{x}_{k+1} := \overline{x}(\tau_{k+1})$, and $\overline{u}(\tau) := \sigma_k^-(\tau)\,\overline{u}_k + \sigma_k^+(\tau)\,\overline{u}_{k+1}$, $\forall \tau \in [\tau_k, \tau_{k+1})$.

$$\therefore\ \Delta x(\tau_{k+1}^-) = \Delta x_{k+1} + \overline{x}_{k+1} - x_{k+1}^{\text{prop}} = A_k \Delta x_k + B_k^- \Delta u_k + B_k^+ \Delta u_{k+1} + S_k \Delta s \tag{25}$$

Here,

$$A_k = I_{n_x} + \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} A(\zeta)\,\Psi_A(\zeta)\,\mathrm{d}\zeta \tag{26a}$$

$$B_k^- = \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} \{A(\zeta)\,\Psi_{B^-}(\zeta) + B(\zeta)\,\sigma_k^-(\zeta)\}\,\mathrm{d}\zeta \tag{26b}$$

$$B_k^+ = \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} \{A(\zeta)\,\Psi_{B^+}(\zeta) + B(\zeta)\,\sigma_k^+(\zeta)\}\,\mathrm{d}\zeta \tag{26c}$$

$$S_k = \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} \{A(\zeta)\,\Psi_S(\zeta) + S(\zeta)\}\,\mathrm{d}\zeta \tag{26d}$$

where

$$\mathring{\Psi}_A(\tau) = A(\tau)\,\Psi_A(\tau) \tag{27a}$$

$$\mathring{\Psi}_{B^-}(\tau) = A(\tau)\,\Psi_{B^-}(\tau) + B(\tau)\,\sigma_k^-(\tau) \tag{27b}$$

$$\mathring{\Psi}_{B^+}(\tau) = A(\tau)\,\Psi_{B^+}(\tau) + B(\tau)\,\sigma_k^+(\tau) \tag{27c}$$

$$\mathring{\Psi}_S(\tau) = A(\tau)\,\Psi_S(\tau) + S(\tau) \tag{27d}$$

x_{k+1}^{prop} is evaluated as follows:

$$x_{k+1}^{\text{prop}} = \overline{x}_k + \lim_{y \to \tau_{k+1}^-} \int_{\tau_k}^{y} F(\zeta, \overline{x}(\zeta), \overline{u}(\zeta), \overline{s})\ \mathrm{d}\zeta \tag{28}$$

and $\Psi_A(\tau) := \Phi(\tau, \tau_k)$. As shown in Equations 29, the discretized dynamics in terms of absolute variables can be recovered from Equation 25.

$$x_{k+1} = A_k(x_k - \overline{x}_k) + B_k^-(u_k - \overline{u}_k) + B_k^+(u_{k+1} - \overline{u}_{k+1}) + S_k(s - \overline{s}) + x_{k+1}^{\text{prop}} \tag{29a}$$

$$\therefore\ x_{k+1} = A_k\,x_k + B_k^-\,u_k + B_k^+\,u_{k+1} + S_k\,s\ + d_k \tag{29b}$$

where

$$d_k := x_{k+1}^{\text{prop}} - \left(A_k\,\overline{x}_k + B_k^-\,\overline{u}_k + B_k^+\,\overline{u}_{k+1} + S_k\,\overline{s}\right)$$

IV. Sequential conic optimization (SeCO)

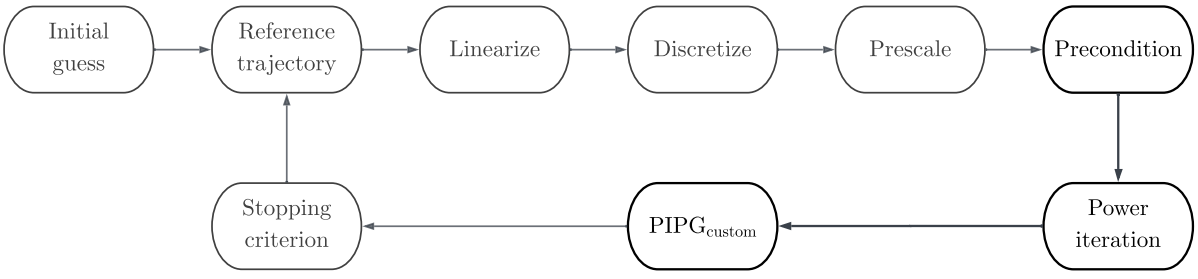

Fig. 3 An overview of the SeCO framework; the blocks in bold constitute the low-level solver.

A. Virtual state

Artificial infeasibility refers to the phenomenon wherein linearization of the nonconvex constraints in a problem can render the convex subproblem infeasible even if there exists a feasible solution to the original nonconvex problem. In order to mitigate this, one approach is to add slack variables to the linearized constraints, thus ensuring that the subproblem is always feasible. These slack variables are usually unconstrained, yet heavily penalized.

The value of these slack variables should go to zero at convergence for a solution to be feasible with respect to the original nonconvex problem. The slack variable is usually referred to as *virtual control* when it is added to the linearized dynamics, and *virtual buffer* when it is added to the linearized constraints [32]. It has been observed that virtual buffer terms are usually not required if virtual control is used [20, 21]. In such cases, however, the

intermediate reference solutions will not be feasible with respect to the LTV dynamics unless the value of the virtual control is zero. If dynamic feasibility of the intermediate reference trajectories is of importance, one approach could be to buffer the constraints and leave the dynamics equation unchanged.

We opt to use a third approach, called *virtual state* [33]. The general idea of this approach is to entirely decouple the dynamics and the control constraints from the state constraints, and to exactly satisfy all the constraints at each iteration, while ensuring that the subproblem is always feasible. In order to achieve this, we introduce a new variable, the virtual state, which acts as a *copy* of the original state variable. If x is the actual state vector, u is the control input vector, and ξ is the virtual state vector, the dynamics constraint is imposed on x, the control constraints are imposed on u, and all the state constraints are imposed on ξ.

In order to ensure that both the dynamics and the path constraints are satisfied at convergence, we minimize the error between x and ξ by including the squared distance between them as a quadratic penalty term in the objective function and heavily penalizing it. The virtual state approach has the benefit of not altering the dynamics manifold, unlike the virtual control approach [20, 21], and preserving the shapes of the conic state constraint sets, unlike the virtual buffer approach [32]. The left superscript, ${}^{\xi}\square$, is used to denote virtual state variables.

B. Trust region

The trust region radius is the distance between the solution to a subproblem and the trajectory about which the system and nonconvex constraints are linearized to create that subproblem. The purpose of a trust region is to: (1) make sure that the solution does not venture too far from the reference trajectory so as to ensure that the linearization is sufficiently accurate, thus preserving its validity; and, (2) mitigate *artificial unboundedness*, which refers to the phenomenon wherein linearization can render the cost unbounded from below even if it is bounded in the original nonconvex problem [32].

There exist both hard and soft trust region methods in the literature [20, 43]. We impose a soft trust region by augmenting the objective function with a quadratic penalty term, and use the penalized trust region (PTR) algorithm, which, as the name suggests, penalizes the trust region radius in lieu of constraining it and adopting an outer-loop update rule [20–22]. This approach has been shown to work very well in practice, and has been employed to successfully solve a wide range of challenging nonconvex optimal control problems in the context of real-time quadrotor path-planning [19, 44, 45], spacecraft rendezvous and docking [46–48], hypersonic entry trajectory optimization [49, 50], and real-time rocket landing guidance [20–22, 33, 51].

C. Initial guess generation

We generate an initial guess trajectory by performing a dual quaternion screw linear interpolation (ScLERP) [52] between the initial dual quaternion and the nominal final dual quaternion. A straight-line interpolation is adopted for the mass profile, and the thrust magnitude profile is derived to counteract the (time-varying) weight of the vehicle—which is then saturated based on the thrust magnitude bounds. The gimbal deflection angle, the gimbal azimuth angle, the angular body rates, and the body torques are set to zeros. The linear velocity profile is obtained by linearly interpolating between the initial and terminal conditions. The time-of-flight parameter can be guessed using an analytical procedure such as the one provided in [53]. The initial guess generation strategy is a design choice and is highly problem dependent; other approaches can be adopted here as well [20, 21, 41].

D. Prescaling

The decision variables are prescaled to ensure that the solutions are roughly on the same order of magnitude (between 0 and 1, in our case). This is an important aspect of numerical optimization algorithms, and can significantly improve solution quality and speed up convergence. The interested reader is referred to [32] for more details on variable scaling.

E. Constraint classification

The template SeCO subproblem is strongly convex, and is given by Equations 30, where all the decision variables are stacked into a single high-dimensional vector, z. Here, Q is a positive definite matrix, and $\mathbb{D}$ is a closed convex set that admits a closed-form projection operation. This vectorized problem possesses a sparsity structure that is amenable to customization, as described in Algorithm 6.

$$\underset{z}{\text{minimize}} \quad \frac{1}{2} z^\top Q z + \langle q, z \rangle \tag{30a}$$

$$\text{subject to} \quad Hz - h = 0 \tag{30b}$$

$$z \in \mathbb{D}$$

We prefer to include as many constraints in the form $z \in \mathbb{D}$ as possible for the following reasons: (1) The constraints $z \in \mathbb{D}$ will be satisfied at every iteration of PIPG. Hence, including more constraints in set $\mathbb{D}$ leads to a smaller search space. In contrast, the constraint $Hz - h = 0$ is only satisfied asymptotically and does not

affect the search space; (2) Transforming many popular constraint sets in optimal control–such as ℓ_2-norm balls, cylinders, boxes–into conic constraints, requires extra auxiliary variables. Imposing these constraints in the form $z \in \mathbb{D}$ eliminates the need for such auxiliary variables and, as a result, decreases the problem size. As shown in the discretized conic subproblem in Section IV.H, every constraint other than the dynamics is classified into set $\mathbb{D}$—this means that these path constraints are exactly satisfied at every solver iteration (up to constraint approximations), and may prove beneficial in the case of premature termination of the solver in an emergency scenario, for instance. The dynamics constraint belongs to the zero cone, and is satisfied asymptotically as the solver converges to an optimum.

F. Constraint reformulations

We choose to *combine* certain intersecting path constraints for the following reasons: (1) to reduce the number of constraints imposed (and in turn, reduce the number of operations that the solver needs to carry out); and, (2) to ensure that all the path constraint sets possess closed-form projection operations. Closed-form expressions only exist for the projection onto the intersection of convex sets in special cases—such as the intersection of a cone and a ball, and the intersection of two halfspaces—but this is not the case in general, even if the individual constraint sets can be projected onto [54]. Although iterative methods such as the alternating direction method of multipliers (ADMM) [55] can be used to compute projections onto the intersection of such convex sets, we opt to reformulate the intersecting constraint sets so as to enable closed-form projections instead.

1. Combined thrust vector constraints

Given the FOH parameterization of the control input, the thrust vector magnitude constraints, in discrete-time, can be expressed as shown in Equations 31, where N is the size of the chosen temporal grid, i.e., the number of discrete temporal nodes in the discretized subproblem.

$$T_{\min} \leq T_k \leq T_{\max}, \quad k = 1\!:\!N \tag{31a}$$

$$0 \leq \delta_k \leq \delta_{\max}, \quad k = 1\!:\!N \tag{31b}$$

$$0 \leq \phi_k \leq 2\pi, \quad k = 1\!:\!N \tag{31c}$$

Further, the thrust vector rate constraints can be expressed as shown in Equations 32, which are exact, owing to the FOH parameterization [50]. s is the dilation factor, and $\frac{s}{N-1}$ is the length of each of the uniformly spaced time-intervals.

$$-\dot{T}_{\max} \leq \frac{T_k - T_{k-1}}{\frac{s}{N-1}} \leq \dot{T}_{\max}, \quad k = 2\!:\!N \tag{32a}$$

$$-\dot{\delta}_{\max} \leq \frac{\delta_k - \delta_{k-1}}{\frac{s}{N-1}} \leq \dot{\delta}_{\max}, \quad k = 2\!:\!N \tag{32b}$$

$$-\dot{\phi}_{\max} \leq \frac{\phi_k - \phi_{k-1}}{\frac{s}{N-1}} \leq \dot{\phi}_{\max}, \quad k = 2\!:\!N \tag{32c}$$

These thrust vector magnitude and rate constraints can be combined by means of an approximation leveraging the reference solution, as shown in Subsection IV.H, such that T_k, δ_k, and ϕ_k, $k = 1\!:\!N$, are the sole decision variables. This form of the constraints enables closed-form projections, ensures that the original thrust vector constraints are never violated, and is exact at convergence.

2. Combined state constraints

We propose a new approach to modeling state-triggered constraints (STCs) [18, 20, 56, 57] that allows for the combination of the global state constraints and the STCs, thus avoiding intersecting constraint sets on the state variables and, in turn, enabling closed-form projection operations onto the constraint sets. For the dual quaternion variable specifically, we reformulate the constraints so as to enable (closed-form) projections onto the intersection of halfspaces.

With the assumption that the global state bound on a variable, $\square_{\max}$, is greater than its STC counterpart, $\square_{\text{STC}_{\max}}$, i.e., $\square_{\max} > \square_{\text{STC}_{\max}}$, we observe that the bounds can be expressed in a single expression, given by Equation 33, where $g(\cdot)$ is the constraint function under consideration. The trigger function, $\psi(t)$, given by Equation 34, takes the value -1 if the vehicle is outside the trigger window, $+1$ if the vehicle is inside the trigger window, and 0, if the vehicle is at either of the triggers. With this formulation, the RHS of Equation 33 can automatically switch between the global bound and the STC bound based on the value that the trigger function assumes.

$$g(\square) \leq \max\{-\psi(t)\,\square_{\max},\, \square_{\text{STC}_{\max}}\} \tag{33}$$

where

$$\psi(t) := \operatorname{sgn}\left\{\left(\rho_{\max} - \|2\,\boldsymbol{q}^{[5:8]}(t)\|_2\right)\left(\|2\,\boldsymbol{q}^{[5:8]}(t)\|_2 - \rho_{\min}\right)\right\} \ni \psi : [0, t_f) \to \{-1,\, 0,\, +1\} \tag{34}$$

Further, an approximation, similar to the one made with the thrust vector constraints, is made to the trigger function, and the global state constraints and the STCs are combined, as shown in Subsection IV.H. Note that the combined state constraints are exact at convergence.

G. Projections

As shown in the discretized conic subproblem in Subsection IV.H, which is a second-order cone program (SOCP), every single path constraint admits a closed-form projection operation. The constraint sets listed in green possess direct closed-form projection operations. The ones listed in blue and yellow involve closed-form projections onto the intersection of halfspaces; the maximum tilt constraint is linearized to enable that. The interested reader is referred to [58] for a description of these closed-form projection operations.

H. The discretized conic subproblem

$$\underset{s,\, u_{[1:N]}}{\text{minimize}} \quad -w_m\, m_N + \frac{1}{2}\left(w_{\text{tr}} \sum_{k=1}^{N} \left(\|x_k - \overline{x}_k\|_2^2 + \|u_k - \overline{u}_k\|_2^2 \right) + w_{\text{tr}_s} \|s - \overline{s}\|_2^2 + w_{\text{vse}} \sum_{k=1}^{N} \|x_k - \xi_k\|_2^2 \right)$$

subject to

Dynamics

zero cone $\quad x_{k+1} = A_k\, x_k + B_k^-\, u_k + B_k^+\, u_{k+1} + S_k\, s \; + d_k \qquad (\mathbb{K}_0) \quad k = 1\!:\!N-1$

Combined control constraints

box $\quad T_{\min} \le T_1 \le T_{\max} \qquad (\mathbb{D})$

box $\quad 0 \le \delta_1 \le \delta_{\max} \qquad (\mathbb{D})$

box $\quad 0 \le \phi_1 \le 2\pi \qquad (\mathbb{D})$

box $\quad \max\!\left(T_{\min},\, -\dot{T}_{\max} \dfrac{\overline{s}}{N-1} + \overline{T}_{k-1}\right) \le T_k \le \min\!\left(T_{\max},\, \dot{T}_{\max} \dfrac{\overline{s}}{N-1} + \overline{T}_{k-1}\right) \qquad (\mathbb{D}) \quad k = 2\!:\!N$

box $\quad \max\!\left(0,\, -\dot{\delta}_{\max} \dfrac{\overline{s}}{N-1} + \overline{\delta}_{k-1}\right) \le \delta_k \le \min\!\left(\delta_{\max},\, \dot{\delta}_{\max} \dfrac{\overline{s}}{N-1} + \overline{\delta}_{k-1}\right) \qquad (\mathbb{D}) \quad k = 2\!:\!N$

box $\quad \max\!\left(0,\, -\dot{\phi}_{\max} \dfrac{\overline{s}}{N-1} + \overline{\phi}_{k-1}\right) \le \phi_k \le \min\!\left(2\pi,\, \dot{\phi}_{\max} \dfrac{\overline{s}}{N-1} + \overline{\phi}_{k-1}\right) \qquad (\mathbb{D}) \quad k = 2\!:\!N$

box $\quad \|\tau_k\|_\infty \le \tau_{\max} \qquad (\mathbb{D}) \quad k = 1\!:\!N$

Combined state constraints

box $\quad \|{}^{\xi}\boldsymbol{\omega}_k^{[1:3]}\|_\infty \le \max\!\left(-\overline{\psi}_k\, \omega_{\max},\, \omega_{\text{STC}_{\max}}\right) \qquad (\mathbb{D}) \quad k = 2\!:\!N-1$

ball $\quad \|{}^{\xi}\boldsymbol{\omega}_k^{[4:6]}\|_2 \le \max\!\left(-\overline{\psi}_k\, v_{\max},\, v_{\text{STC}_{\max}}\right) \qquad (\mathbb{D}) \quad k = 2\!:\!N-1$

halfspace $\quad \overline{\boldsymbol{q}}_k^{[1:2]\top}\, {}^{\xi}\boldsymbol{q}_k^{[1:2]} \le \|\overline{\boldsymbol{q}}_k^{[1:2]}\|_2 \sin\frac{\theta_{\text{STC}_{\max}}}{2} \qquad (\mathbb{D}) \quad k \ni \overline{\psi}_k \ge 0$

halfspace $\quad$ Linearize Equation 10d (maximum line-of-sight angle) $\qquad (\mathbb{D}) \quad k \ni \overline{\psi}_k \ge 0$

halfspace $\quad \overline{\boldsymbol{q}}_k^{[1:2]\top}\, {}^{\xi}\boldsymbol{q}_k^{[1:2]} \le \|\overline{\boldsymbol{q}}_k^{[1:2]}\|_2 \sin\frac{\theta_{\max}}{2} \qquad (\mathbb{D}) \quad k \ni \overline{\psi}_k < 0$

halfspace $\quad$ Linearize Equation 9d (minimum altitude) $\qquad (\mathbb{D}) \quad k \ni \overline{\psi}_k < 0$

Boundary conditions

singleton $\quad {}^{\xi}m_1 = m_i \qquad (\mathbb{D})$

singleton $\quad {}^{\xi}\boldsymbol{q}_1 = \begin{pmatrix} q_i \\ \frac{1}{2}\begin{pmatrix} r_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \qquad (\mathbb{D})$

singleton $\quad {}^{\xi}\boldsymbol{\omega}_1 = \begin{pmatrix} \omega_{\mathcal{B}_i} \\ q_i^* \otimes \begin{pmatrix} v_{\mathcal{I}_i} \\ 0 \end{pmatrix} \otimes q_i \end{pmatrix} \qquad (\mathbb{D})$

halfspace $\quad {}^{\xi}m_N \ge m_f \qquad (\mathbb{D})$

singleton $\quad {}^{\xi}\boldsymbol{q}_N^{[1:2]} = 0 \qquad (\mathbb{D})$

subspace $\quad \begin{bmatrix} \left(\frac{1}{2}[r_{\mathcal{I}_f}]_\otimes \begin{pmatrix} \mathbf{0}_{2\times 2} \\ I_2 \end{pmatrix} \right) & -I_4 \end{bmatrix} {}^{\xi}\boldsymbol{q}_N^{[3:8]} = 0 \qquad (\mathbb{D})$

singleton $\quad {}^{\xi}\boldsymbol{\omega}_N = \begin{pmatrix} \mathbf{0}_{5\times 1} \\ v_{z_{\mathcal{I}_f}} \end{pmatrix} \qquad (\mathbb{D})$

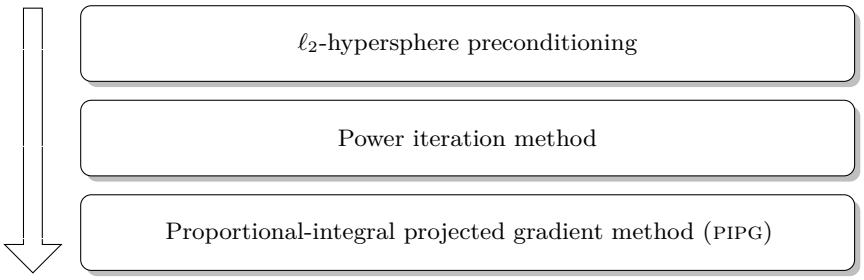

Fig. 4 The SeCO subproblem solver.

A. Preconditioning

First-order methods are sensitive to problem conditioning and typically perform poorly on ill-conditioned problems [59]. As SCP (SeCO) implementations typically impose a heavy weight on the virtual term penalty relative to the original cost and trust region penalties [32], the objective function of the resulting convex (conic) subproblem is inherently ill-conditioned.

Preconditioning is a heuristic that seeks to reduce the condition number of a parameter matrix in an optimization problem via a coordinate transformation. There exist both exact and heuristic methods for preconditioning, both with their own limitations. The exact optimal diagonal preconditioner for a matrix, i.e., the diagonal preconditioner that minimizes the condition number of the resulting matrix, can be found by solving a semidefinite program (SDP) [59, 60]. However, such an approach is not suitable for real-time applications for the following reasons: (1) solving such a problem is typically more computationally expensive than solving the original problem itself [59]; and, (2) SDPs are generally more difficult to solve than second-order cone programs (SOCPs), which usually form the most general class of convex optimization problems that are suitable for safety-critical applications [57].

Matrix equilibration is another approach that has been widely used in the literature for preconditioning in the context of optimization [61–65]. This approach typically involves finding a diagonal preconditioner to scale the matrix under consideration such that its rows have equal norms and its columns have equal norms [66]. Equilibration-based preconditioning techniques either rely on convex optimization or iterative methods, some more reliable than others [59]. Although many of these methods take both the objective function and constraints into account and have been found to work well in practice [59, 67], they either require the solution of another convex optimization problem or rely on heuristic iterative procedures that are not guaranteed to reduce the condition number of the matrix being equilibrated even if they converge. Further, preconditioners based on incomplete matrix factorizations are also popular in practice [68].

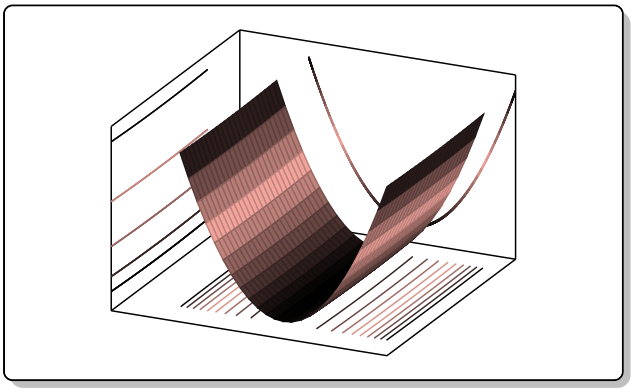

Fig. 5 An ill-conditioned bivariate quadratic function (prior to preconditioning, for instance).

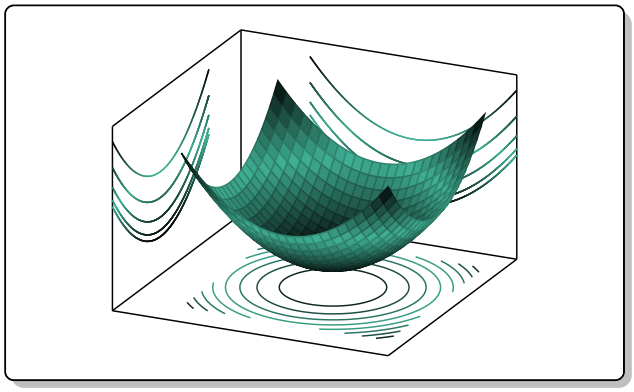

Fig. 6 A well-conditioned bivariate quadratic function (after preconditioning, for instance).

In this work, we propose a preconditioner for the objective function matrix, shown in Algorithm 1, for problems that fit the template of SeCO. This procedure uses an exact Cholesky factorization, while effectively exploiting the structure of the conic subproblem in SeCO with minimal computational overhead—in fact, this preconditioner can be computed analytically, without the need for any explicit matrix factorization/inversion operations or iterative heuristics; i.e., the structure of the SeCO subproblem enables the computation of the Cholesky factor of the objective function matrix in closed-form, as shown in the Algorithm 4. Further, the transformed dynamics matrix still possesses a sparsity structure that is amenable to customization, as shown in Algorithm 6. The preconditioner transforms ill-conditioned quadratic functions to well-conditioned quadratic functions, the level sets of which are ℓ_2-norm hyperspheres; hence the name. The effect of preconditioning on an ill-conditioned quadratic objective

function, such as the one in Figure 5, is depicted in Figure 6. In our initial testing, we observed a reduction in the number of solver iterations to convergence of roughly 5 to 10 times with this preconditioner.

Algorithm 1 ℓ_2-hypersphere preconditioning

Inputs: Q, q, H, $\mathbb{D}$, λ

Require: $Q \succ 0$, $Q = Q^\top$

$L^\top L \leftarrow \operatorname{chol} Q$ ▷ Cholesky decomposition
$L_{\text{inv}} \leftarrow L^{-1}$
$\hat{\mathbb{D}} \leftarrow L\,\mathbb{D}$
$\hat{H} \leftarrow H L_{\text{inv}}$
$\hat{q} \leftarrow \lambda\, L_{\text{inv}}^\top q$

Ensure: $\square \in \hat{\mathbb{D}} \Leftrightarrow L^{-1}\square \in \mathbb{D}$

Return: $\hat{q}$, $\hat{H}$, $\hat{\mathbb{D}}$, L, L_{inv}, σ

As shown in Algorithm 1, we scale the objective function with a positive scalar, λ, that factors into the step-sizes of PIPG and hence serves as a tuning parameter. Scaling the objective function appropriately can also help keep the magnitude of the dual variables in check [59]. The preconditioned conic subproblem is shown in Problem 35b.

$$\underset{\hat{z}}{\text{minimize}} \quad \frac{\lambda}{2}\,\hat{z}^\top \hat{z} + \langle \hat{q}, \hat{z} \rangle \tag{35a}$$

$$\text{subject to} \quad \hat{H}\hat{z} - h = 0 \tag{35b}$$

$$\hat{z} \in \hat{\mathbb{D}}$$

The objective function matrix of the preconditioned problem is of the form λI, with a condition number of 1, which is the minimum attainable condition number (and hence, the preconditioner is optimal). The transformed projection $\hat{z} \in \hat{\mathbb{D}}$ needs to preserve membership of the original primal variable, z, to set $\mathbb{D}$, i.e., $\hat{z} \in \hat{\mathbb{D}} \Leftrightarrow z \in \mathbb{D}$ [67]. The structure of the conic subproblem in SeCO naturally preserves this, as is apparent from Algorithm 4. Post-solution, the original primal variable can be recovered as follows: $z = L_{\text{inv}}\hat{z}$.

B. Power iteration method

The power iteration method described in Algorithm 2 computes the maximum eigenvalue value and the corresponding eigenvector of a given diagonalizable matrix [68]. We can exploit the structure of matrix $\hat{H}$ in trajectory optimization problems to customize Algorithm 2 to Algorithm 5 so as to avoid sparse linear algebra operations with large dimensional matrices and vectors.

Algorithm 2 Power iteration method

Inputs: $\hat{H}$, z, ϵ_{abs}, ϵ_{rel}, ϵ_{buff}, $j_{\max}$

Require: $\|z\|_2 > 0$

$\sigma \leftarrow \|z\|_2$ ▷ initialization
for $j \leftarrow 1 : j_{\max}$ **do**
  $w \leftarrow \frac{1}{\sigma}\hat{H}z$
  $z \leftarrow \hat{H}^\top w$
  $\sigma^\star \leftarrow \|z\|_2$
  if $|\sigma^\star - \sigma| \le \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\{\sigma^\star, \sigma\}$ **then** ▷ stopping criterion
    break
  else if $j < j_{\max}$ **then**
    $\sigma \leftarrow \sigma^\star$
  end if
end for
$\sigma \leftarrow (1 + \epsilon_{\text{buff}})\,\sigma^\star$ ▷ buffer the estimated maximum singular value

Return: σ ▷ $\approx \max \operatorname{spec} \hat{H}^\top \hat{H} = \sigma_{\max}(\hat{H}^\top \hat{H}) = \|\hat{H}\|_2^2$

C. PIPG

The proportional-integral projected gradient method, PIPG, is a first-order primal-dual algorithm for conic optimization [29]. It allows matrix-factorization/inverse-free and easily-verifiable solver implementations for real-time and embedded applications [69]. PIPG achieves the optimal global convergence rates (worst-case) in theory, and performs much faster in practice. It exploits the sparsity structure of conic constraints via parallel matrix operations and the geometric structure of constraint sets via efficient closed-form projections [28]. Unlike most off-the-shelf methods, PIPG allows for warm-starting and enjoys a light computational overhead, as it avoids the cumbersome canonical transformation procedure that standard conic programs are subject to. It is also compatible with extrapolation, which has been shown to accelerate convergence [31]. PIPG not only enables versatile convex optimization, but also has the ability to boost the performance of sequential convex programming (SCP) methods for nonconvex optimization. Moreover, the SeCO framework specializes SCP algorithms to exploit features of PIPG to solve nonconvex optimal control problems in real-time [33].

One of the major applications of PIPG is in solving trajectory optimization problems, given that all of the sparse linear algebra operations in Equations 3 can be devectorized [30, Algorithm 2], as shown in Algorithm 6, and posed as simple, small-dimensional matrix-vector manipulations that are suitable for real-time performance onboard resource-constrained embedded hardware.

PIPG converges to an optimal solution when the difference between two consecutive iterates converges to zero [31, Theorem 1]. Hence, in practice, we terminate the solver when the difference between two consecutive iterates is sufficiently small. In particular, given a relative accuracy tolerance ϵ_{rel} and an absolute accuracy tolerance ϵ_{abs}, we terminate PIPG when the following conditions are met:

$$\|\hat{z}^{j+1} - \hat{z}^{j}\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\|\hat{z}^{j+1}\|_\infty,\, \|\hat{z}^{j}\|_\infty\right\},$$
$$\|\hat{w}^{j+1} - \hat{w}^{j}\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\|\hat{w}^{j+1}\|_\infty,\, \|\hat{w}^{j}\|_\infty\right\}$$

We note that such a combination of absolute and relative accuracy tolerances is popular among first-order solvers [59, 70].

Algorithm 3 PIPG

Inputs: $\hat{q}$, $\hat{H}$, h, $\hat{\mathbb{D}}$, L, L_{inv}, λ, σ, ρ, ϵ_{abs}, ϵ_{rel}, j_{check}, $j_{\max}$,
$z^\star$, $\hat{w}^\star$ ▷ warm start

$\hat{z}^\star \leftarrow L\, z^\star$ ▷ transform previous primal solution
$\zeta^1 \leftarrow \hat{z}^\star$ ▷ initialize transformed primal variable
$\eta^1 \leftarrow \hat{w}^\star$ ▷ initialize transformed dual variable
$\alpha \leftarrow \frac{2}{\lambda + \sqrt{\lambda^2 + 4\sigma}}$ ▷ step-size
for $j \leftarrow 1 : j_{\max}$ **do**
  $\hat{z}^{j+1} = \pi_{\mathbb{D}}[\zeta^j - \alpha\,(\lambda\,\zeta^j + \hat{q} + \hat{H}^\top \eta^j)]$ ▷ projected gradient step
  $\hat{w}^{j+1} = \eta^j + \alpha\,(\hat{H}(2\,\hat{z}^{j+1} - \zeta^j) - h)$ ▷ PI feedback of affine equality constraint violation
  $\zeta^{j+1} = (1-\rho)\,\zeta^j + \rho\,\hat{z}^{j+1}$ ▷ extrapolate transformed primal variable
  $\eta^{j+1} = (1-\rho)\,\eta^j + \rho\,\hat{w}^{j+1}$ ▷ extrapolate transformed dual variable
  if $j \bmod j_{\text{check}} = 0$ **then** ▷ check stopping criterion every j_{check} iterations
    if $\|\hat{z}^{j+1} - \hat{z}^{j}\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\|\hat{z}^{j+1}\|_\infty,\, \|\hat{z}^{j}\|_\infty\right\}$ **and**
     $\|\hat{w}^{j+1} - \hat{w}^{j}\|_\infty \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{\|\hat{w}^{j+1}\|_\infty,\, \|\hat{w}^{j}\|_\infty\right\}$ **then** ▷ stopping criterion
      break
    end if
  end if
end for
$z^\star \leftarrow L_{\text{inv}}\, \hat{z}^{j+1}$ ▷ recover original primal variable
$\hat{w}^\star \leftarrow \hat{w}^{j+1}$ ▷ retain transformed dual variable

Return: $z^\star$, $\hat{w}^\star$

VI. Solver customization

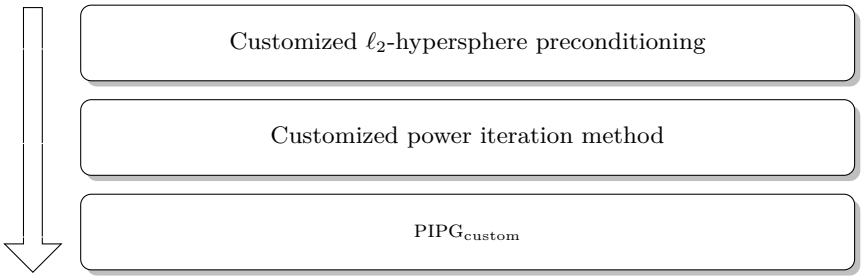

Fig. 7 The customized SeCO subproblem solver.

We define customization as the exploitation of the sparsity pattern of the optimal control problem at hand, so as to enable low-dimensional matrix-vector multiplications and other dense linear algebra operations with devectorized variables, and thus avoid sparse linear algebra operations. In contrast to dense matrix-vector multiplications, sparse matrix-vector multiplications (SpMV) typically suffer from: (1) additional computational overhead in terms of instructions and storage; and, (2) memory access patterns that are indirect and irregular; and are hence inefficient [71]. Customized algorithms preclude these inefficient operations, and are especially effective with optimization problem sizes that are characteristic of onboard guidance [27, 30].

A. Customized preconditioning

Consider the vectorized conic optimization problem given by Equations 30, where

$$z = (x_1, \ldots, x_N, \xi_1, \ldots, \xi_N, u_1, \ldots, u_N, s) \in \mathbb{R}^{n_z}$$
$$q = (q_x, q_\xi, q_u, q_s) \in \mathbb{R}^{n_z}$$
$$Q = \begin{pmatrix} Q_{\text{state}} & & & \\ & Q_u & & \\ & & Q_s & \end{pmatrix} \in \mathbb{S}^{n_z}_{++}$$

$Q_{\text{state}} := W_{\text{state}} \otimes I_{n_x N}$, $Q_u := w_{\text{tr}}\, I_{n_u N}$, $Q_s := w_{\text{tr}_s}$, $W_{\text{state}} := \begin{pmatrix} w_{\text{tr}} + w_{\text{vse}} & -w_{\text{vse}} \\ -w_{\text{vse}} & w_{\text{vse}} \end{pmatrix}$, and n_z is the length of z.

Q is symmetric positive definite (SPD) and therefore has a unique Cholesky decomposition [72, Corollary 7.2.9]. Further, since Q is block diagonal, the blocks are effectively decoupled, and the Cholesky decomposition can be applied directly to the individual blocks, as shown in Equation 36.

$$\text{chol}\, Q = \begin{pmatrix} \text{chol}\, Q_{\text{state}} & & & \\ & \text{chol}\, Q_u & & \\ & & \text{chol}\, Q_s & \end{pmatrix} \tag{36}$$

Let L denote the Cholesky factor of Q, such that $Q = L^\top L$, $L_\square$ denote the Cholesky factor of a block partition of Q, such that $Q_\square = L_\square^\top L_\square$, and R denote the Cholesky factor of W_{state}, such that $W_{\text{state}} = R^\top R$.

Given two SPD matrices A and B, it can be shown that $\text{chol}(A \otimes B) = \text{chol}\, A \otimes \text{chol}\, B$ [73].

$$\therefore \text{chol}\, Q_{\text{state}} = \text{chol}\, W_{\text{state}} \otimes \text{chol}\, I_{n_x N} \tag{37}$$
$$= R^\top R \otimes I_{n_x N} \tag{38}$$
$$= L_{\text{state}}^\top L_{\text{state}} \tag{39}$$

where

$$R = \frac{1}{\sqrt{w_{\text{tr}} + w_{\text{vse}}}} \begin{pmatrix} w_{\text{tr}} + w_{\text{vse}} & -w_{\text{vse}} \\ 0 & \sqrt{w_{\text{tr}} w_{\text{vse}}} \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{40}$$
$$\text{and } L_{\text{state}} = R \otimes I_{n_x N} \tag{41}$$

Since Q_u is a diagonal matrix, $\text{chol}\, Q_u = L_u^\top L_u$, where $L_u = \sqrt{w_{\text{tr}}}\, I_{n_u N}$. Since Q_s is a scalar, $\text{chol}\, Q_s = L_s^2$, where $L_s = \sqrt{w_{\text{tr}_s}}$.

Finally, the Cholesky factor of Q can be computed as follows: $L = \text{blkdiag}\{L_{\text{state}}, L_u, L_s\}$. Since L is block diagonal, its inverse can be written in terms of the inverses of the individual block partitions L_{state}, L_u, and L_s,

as shown in Equation 42.

$$L^{-1} = \text{blkdiag}\{L_{\text{state}}^{-1}, L_u^{-1}, L_s^{-1}\} \tag{42}$$

Given two nonsingular SPD matrices A and B, it can be shown that $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$ [74, Corollary 10].

$$\therefore L_{\text{state}}^{-1} = (R \otimes I_{n_x N})^{-1} = R^{-1} \otimes I_{n_x N} \tag{43}$$

where

$$R^{-1} = \frac{1}{\sqrt{w_{\text{tr}} + w_{\text{vse}}}} \begin{pmatrix} 1 & w \\ 0 & w + \frac{1}{w} \end{pmatrix} \in \mathbb{R}^{2\times 2} \tag{44}$$

$$w = \sqrt{\frac{w_{\text{vse}}}{w_{\text{tr}}}} \tag{45}$$

and

$$L_u^{-1} = \frac{1}{\sqrt{w_{\text{tr}}}} I_{n_u N} \text{ and } L_s^{-1} = \frac{1}{\sqrt{w_{\text{tr}_s}}} \tag{46}$$

For embedded applications, if the problem data does not need to change, i.e., if Q is fixed, the preconditioning parameters can be computed offline and stored onboard. However, even in cases where the problem data may change, note that effectively, the only matrix factorization/inversion operations required in the proposed preconditioning procedure are one Cholesky decomposition of a 2×2 matrix (W_{state}) and one inversion of a 2×2 upper triangular matrix (R), both of which have closed-form expressions. An efficient implementation of the customized ℓ_2-hypersphere preconditioning procedure with no explicit matrix factorizations/inversions is documented in Algorithm 4. Note that all the transformations in the algorithm only involve scaling the problem data by scalars (with the exception of one vector addition operation). Further, these scaling factors only depend on the objective function weights, which are independent of the problem size, thus making the algorithm suitable for large-scale problems as well.

Algorithm 4 Customized ℓ_2-hypersphere preconditioning

Inputs: w_{vse}, w_{tr}, w_{tr_s}, q_x, q_ξ, q_u, q_s, $A_{[1:N-1]}$, $B^-_{[1:N-1]}$, $B^+_{[1:N-1]}$, $S_{[1:N-1]}$, $\mathbb{D}_\xi$, $\mathbb{D}_u$, $\mathbb{D}_s$, λ

Require: w_{vse}, w_{tr}, $w_{\text{tr}_s} > 0$ — R: Equation 40; R^{-1}: Equation 44

$l_{x_1} \leftarrow \sqrt{w_{\text{tr}} + w_{\text{vse}}}$ ▷ $R_{\{1,1\}}$
$l_{x_2} \leftarrow \frac{-w_{\text{vse}}}{l_{x_1}}$ ▷ $R_{\{1,2\}}$
$l_\xi \leftarrow \frac{\sqrt{w_{\text{tr}}\, w_{\text{vse}}}}{l_{x_1}}$ ▷ $R_{\{2,2\}}$
$l_{x_{1_{\text{inv}}}} \leftarrow \frac{1}{l_{x_1}}$ ▷ $R^{-1}_{\{1,1\}} = \frac{1}{R_{\{1,1\}}}$
$l_{x_{2_{\text{inv}}}} \leftarrow \frac{-l_{x_2}}{l_{x_1} l_\xi}$ ▷ $R^{-1}_{\{1,2\}} = \frac{-R_{\{1,2\}}}{R_{\{1,1\}} R_{\{2,2\}}}$
$l_{\xi_{\text{inv}}} \leftarrow \frac{1}{l_\xi}$ ▷ $R^{-1}_{\{2,2\}} = \frac{1}{R_{\{2,2\}}}$
$l_u \leftarrow \sqrt{w_{\text{tr}}}$
$l_{u_{\text{inv}}} \leftarrow \frac{1}{l_u}$
$l_s \leftarrow \sqrt{w_{\text{tr}_s}}$
$l_{s_{\text{inv}}} \leftarrow \frac{1}{l_s}$
$\hat{\mathbb{D}}_\xi \leftarrow l_\xi\, \mathbb{D}_\xi$
$\hat{\mathbb{D}}_u \leftarrow l_u\, \mathbb{D}_u$
$\hat{\mathbb{D}}_s \leftarrow l_s\, \mathbb{D}_s$
$\hat{A}^1_{[1:N-1]} \leftarrow l_{x_{1_{\text{inv}}}} A_{[1:N-1]}$
$\hat{A}^2_{[1:N-1]} \leftarrow l_{x_{2_{\text{inv}}}} A_{[1:N-1]}$
$\hat{B}^-_{[1:N-1]} \leftarrow l_{u_{\text{inv}}} B^-_{[1:N-1]}$
$\hat{B}^+_{[1:N-1]} \leftarrow l_{u_{\text{inv}}} B^+_{[1:N-1]}$
$\hat{S}_{[1:N-1]} \leftarrow l_{s_{\text{inv}}} S_{[1:N-1]}$
$\hat{q}_x \leftarrow \lambda\, l_{x_{1_{\text{inv}}}} q_x$
$\hat{q}_\xi \leftarrow \lambda\, (l_{x_{2_{\text{inv}}}} q_x + l_{\xi_{\text{inv}}} q_\xi)$
$\hat{q}_u \leftarrow \lambda\, l_{u_{\text{inv}}} q_u$
$\hat{q}_s \leftarrow \lambda\, l_{s_{\text{inv}}} q_s$

Return: $\hat{q}_x$, $\hat{q}_\xi$, $\hat{q}_u$, $\hat{q}_s$, $\hat{A}^1_{[1:N-1]}$, $\hat{A}^2_{[1:N-1]}$, $\hat{B}^-_{[1:N-1]}$, $\hat{B}^+_{[1:N-1]}$, $\hat{S}_{[1:N-1]}$, $\hat{\mathbb{D}}_\xi$, $\hat{\mathbb{D}}_u$, $\hat{\mathbb{D}}_s$, l_{x_1}, l_{x_2}, l_ξ, l_u, l_s, $l_{x_{1_{\text{inv}}}}$, $l_{x_{2_{\text{inv}}}}$, $l_{\xi_{\text{inv}}}$, $l_{u_{\text{inv}}}$, $l_{s_{\text{inv}}}$, σ

B. Customized power iteration method

Algorithm 5 Customized power iteration method

Inputs: $\hat{A}^1_{[1:N-1]}$, $\hat{A}^2_{[1:N-1]}$, $\hat{B}^-_{[1:N-1]}$, $\hat{B}^+_{[1:N-1]}$, $\hat{S}_{[1:N-1]}$, $l_{x_{1_{\text{inv}}}}$, $l_{x_{2_{\text{inv}}}}$, $x_{[1:N]}$, $\xi_{[1:N]}$, $u_{[1:N]}$, s, ϵ_{abs}, ϵ_{rel}, ϵ_{buff}, $j_{\max}$

Require: $\|x_{[1:N]}\|_2 > 0,\ \|\xi_{[1:N]}\|_2 > 0,\ \|u_{[1:N]}\|_2 > 0,\ s > 0$

$\sigma \leftarrow s^2$

for $k \leftarrow 1\!:\!N$ **do** ▷ Algorithm 2, Line 1

  $\sigma \leftarrow \sigma + \|x_k\|_2^2 + \|\xi_k\|_2^2 + \|u_k\|_2^2$

end for

$\sigma \leftarrow \sqrt{\sigma}$

for $j \leftarrow 1\!:\!j_{\max}$ **do**

  for $k \leftarrow 1\!:\!N\!-\!1$ **do** ▷ Algorithm 2, Line 3

    $w_k \leftarrow \frac{1}{\sigma}\left(\hat{A}^1_k\, x_k + \hat{A}^2_k\, \xi_k + \hat{B}^-_k u_k + \hat{B}^+_k u_{k+1} + \hat{S}_k\, s - l_{x_{1_{\text{inv}}}} x_{k+1} - l_{x_{2_{\text{inv}}}} \xi_{k+1}\right)$

  end for

  $x_1 \leftarrow \hat{A}^{1\top}_1 w_1$

  $\xi_1 \leftarrow \hat{A}^{2\top}_1 w_1$

  $u_1 \leftarrow \hat{B}^{-\top}_1 w_1$

  $s \leftarrow \hat{S}^\top_1 w_1$

  for $k \leftarrow 2\!:\!N\!-\!1$ **do** ▷ Algorithm 2, Line 4

    $x_k \leftarrow \hat{A}^{1\top}_k w_k - l_{x_{1_{\text{inv}}}} w_{k-1}$

    $\xi_k \leftarrow \hat{A}^{2\top}_k w_k - l_{x_{2_{\text{inv}}}} w_{k-1}$

    $u_k \leftarrow \hat{B}^{-\top}_k w_k + \hat{B}^{+\top}_{k-1} w_{k-1}$

    $s \leftarrow s + \hat{S}^\top_k w_k$

  end for

  $x_N \leftarrow -l_{x_{1_{\text{inv}}}} w_{N-1}$

  $\xi_N \leftarrow -l_{x_{2_{\text{inv}}}} w_{N-1}$

  $u_N \leftarrow \hat{B}^{+\top}_{N-1} w_{N-1}$

  $\sigma^\star \leftarrow s^2$

  for $k \leftarrow 1\!:\!N$ **do** ▷ Algorithm 2, Line 5

    $\sigma^\star \leftarrow \sigma^\star + \|x_k\|_2^2 + \|\xi_k\|_2^2 + \|u_k\|_2^2$

  end for

  $\sigma^\star \leftarrow \sqrt{\sigma^\star}$

  if $|\sigma^\star - \sigma| \le \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\{\sigma^\star, \sigma\}$ **then** ▷ stopping criterion

    break

  else if $j < j_{\max}$ **then**

    $\sigma \leftarrow \sigma^\star$

  end if

end for

$\sigma \leftarrow (1 + \epsilon_{\text{buff}})\, \sigma^\star$ ▷ buffer the estimated maximum singular value

Return: σ ▷ $\approx \max \operatorname{spec} \hat{H}^\top \hat{H} = \sigma_{\max}(\hat{H}^\top \hat{H}) = \|\hat{H}\|_2^2$

C. Customized PIPG

Algorithm 6 $\text{PIPG}_{\text{custom}}$

Inputs: $\hat{q}_{x_{[1:N]}}$, $\hat{q}_{\xi_{[1:N]}}$, $\hat{q}_{u_{[1:N]}}$, $\hat{q}_s$, $\hat{A}^1_{[1:N-1]}$, $\hat{A}^2_{[1:N-1]}$, $\hat{B}^-_{[1:N-1]}$, $\hat{B}^+_{[1:N-1]}$, $\hat{S}_{[1:N-1]}$, $d_{[1:N-1]}$,
$\hat{\mathbb{D}}_{\xi_{[1:N]}}$, $\hat{\mathbb{D}}_{u_{[1:N]}}$, $\hat{\mathbb{D}}_s$, l_{x_1}, l_{x_2}, l_ξ, l_u, l_s, $l_{x_{1_{\text{inv}}}}$, $l_{x_{2_{\text{inv}}}}$, $l_{\xi_{\text{inv}}}$, $l_{u_{\text{inv}}}$, $l_{s_{\text{inv}}}$, λ, σ,
ρ, ϵ_{abs}, ϵ_{rel}, j_{check}, $j_{\max}$,
$x^\star_{[1:N]}$, $\xi^\star_{[1:N]}$, $u^\star_{[1:N]}$, $s^\star$, $\hat{w}^\star_{[1:N-1]}$ ▷ warm start

$\hat{x}^\star_{[1:N]} \leftarrow l_{x_1} x^\star_{[1:N]} + l_{x_2} \xi^\star_{[1:N]}$ ▷ transform previous primal solution
$\hat{\xi}^\star_{[1:N]} \leftarrow l_\xi \, \xi^\star_{[1:N]}$
$\hat{u}^\star_{[1:N]} \leftarrow l_u u^\star_{[1:N]}$
$\hat{s}^\star \leftarrow l_s \, s^\star$
$x^1_{\zeta_{[1:N]}} \leftarrow \hat{x}^\star_{[1:N]}$ ▷ initialize transformed primal variables
$\xi^1_{\zeta_{[1:N]}} \leftarrow \hat{\xi}^\star_{[1:N]}$
$u^1_{\zeta_{[1:N]}} \leftarrow \hat{u}^\star_{[1:N]}$
$s^1_\zeta \leftarrow \hat{s}^\star$
$\eta^1_{[1:N-1]} \leftarrow \hat{w}^\star_{[1:N-1]}$ ▷ initialize transformed dual variable
$\alpha \leftarrow \frac{2}{\lambda + \sqrt{\lambda^2 + 4\sigma}}$ ▷ step-size
for $j \leftarrow 1 : j_{\max}$ **do**
  $\hat{x}^j_1 \leftarrow x^j_{\zeta_1} - \alpha \, (\lambda \, x^j_{\zeta_1} + \hat{q}_{x_1} + \hat{A}^{1\top}_1 \eta^j_1)$
  $\hat{\xi}^j_1 \leftarrow \pi_{\mathbb{D}_{\xi_1}} [\xi^j_{\zeta_1} - \alpha \, (\lambda \, \xi^j_{\zeta_1} + \hat{q}_{\xi_1} + \hat{A}^{2\top}_1 \eta^j_1)]$
  $\hat{u}^j_1 \leftarrow \pi_{\mathbb{D}_{u_1}} [u^j_{\zeta_1} - \alpha \, (\lambda \, u^j_{\zeta_1} + \hat{q}_{u_1} + \hat{B}^{-\top}_1 \eta^j_1)]$
  $\mathcal{S} \leftarrow \hat{S}^\top_1 \eta^j_1$
  for $k \leftarrow 2 : N-1$ **do** ▷ projected gradient step
    $\hat{x}^{j+1}_k \leftarrow x^j_{\zeta_k} - \alpha \, (\lambda \, x^j_{\zeta_k} + \hat{q}_{x_k} + \hat{A}^{1\top}_k \eta^j_k - l_{x_{1_{\text{inv}}}} \eta^j_{k-1})$
    $\hat{\xi}^{j+1}_k \leftarrow \pi_{\mathbb{D}_{\xi_k}} [\xi^j_{\zeta_k} - \alpha \, (\lambda \, \xi^j_{\zeta_k} + \hat{q}_{\xi_k} + \hat{A}^{2\top}_k \eta^j_k - l_{x_{2_{\text{inv}}}} \eta^j_{k-1})]$
    $\hat{u}^{j+1}_k \leftarrow \pi_{\mathbb{D}_{u_k}} [u^j_{\zeta_k} - \alpha \, (\lambda \, u^j_{\zeta_k} + \hat{q}_{u_k} + \hat{B}^{-\top}_k \eta^j_k + \hat{B}^{+\top}_{k-1} \eta^j_{k-1})]$
    $\mathcal{S} \leftarrow \mathcal{S} + \hat{S}^\top_k \eta^j_k$
  end for
  $\hat{x}^{j+1}_N \leftarrow x^j_{\zeta_N} - \alpha \, (\lambda \, x^j_{\zeta_N} + \hat{q}_{x_N} - l_{x_{1_{\text{inv}}}} \eta^j_{N-1})$
  $\hat{\xi}^{j+1}_N \leftarrow \pi_{\mathbb{D}_{\xi_N}} [\xi^j_{\zeta_N} - \alpha \, (\lambda \, \xi^j_{\zeta_N} + \hat{q}_{\xi_N} - l_{x_{2_{\text{inv}}}} \eta^j_{N-1})]$
  $\hat{u}^{j+1}_N \leftarrow \pi_{\mathbb{D}_{u_N}} [u^j_N - \alpha \, (\lambda \, u^j_{\zeta_N} + \hat{q}_{u_N} + \hat{B}^{+\top}_{N-1} \eta^j_{N-1})]$
  $\hat{s}^{j+1} \leftarrow \pi_{\mathbb{D}_s} [s^j_\zeta - \alpha \, (\lambda \, s^j_\zeta + \hat{q}_s + \mathcal{S})]$
  for $k \leftarrow 1 : N-1$ **do** ▷ PI feedback of affine equality constraint violation
    $\hat{w}^{j+1}_k \leftarrow \eta^j_k + \alpha \, (\hat{A}^1_k \, (2 \, \hat{x}^{j+1}_k - x^j_{\zeta_k}) + \hat{A}^2_k \, (2 \, \hat{\xi}^{j+1}_k - \xi^j_{\zeta_k}) + \hat{B}^-_k \, (2 \, \hat{u}^{j+1}_k - u^j_{\zeta_k}) + \hat{B}^+_k \, (2 \, \hat{u}^{j+1}_{k+1} - u^j_{\zeta_{k+1}})$
    $+ \hat{S}_k \, (2 \, \hat{s}^{j+1} - s^j_\zeta) - l_{x_{1_{\text{inv}}}} \, (2 \, \hat{x}^{j+1}_{k+1} - x^j_{\zeta_{k+1}}) - l_{x_{2_{\text{inv}}}} \, (2 \, \hat{\xi}^{j+1}_{k+1} - \xi^j_{\zeta_{k+1}}) + d_k)$
  end for
  $x^{j+1}_{\zeta_{[1:N]}} \leftarrow (1-\rho) \, x^j_{\zeta_{[1:N]}} + \rho \, \hat{x}^{j+1}_{[1:N]}$ ▷ extrapolate transformed primal variables
  $\xi^{j+1}_{\zeta_{[1:N]}} \leftarrow (1-\rho) \, \xi^j_{\zeta_{[1:N]}} + \rho \, \hat{\xi}^{j+1}_{[1:N]}$
  $u^{j+1}_{\zeta_{[1:N]}} \leftarrow (1-\rho) \, u^j_{\zeta_{[1:N]}} + \rho \, \hat{u}^{j+1}_{[1:N]}$
  $s^{j+1}_\zeta \leftarrow (1-\rho) \, s^j_\zeta + \rho \, \hat{s}^{j+1}$
  $\eta^{j+1}_{[1:N-1]} \leftarrow (1-\rho) \, \eta^j_{[1:N-1]} + \rho \, \hat{w}^{j+1}_{[1:N-1]}$ ▷ extrapolate transformed dual variable
  if $j \bmod j_{\text{check}} = 0$ **then** ▷ check stopping criterion every j_{check} iterations
    TERMINATE $\leftarrow$ $\text{STOPPING}_{\text{custom}}(\hat{x}^{j+1}_{[1:N]}, \hat{\xi}^{j+1}_{[1:N]}, \hat{u}^{j+1}_{[1:N]}, \hat{s}^{j+1}, \hat{w}^{j+1}_{[1:N-1]}, \hat{x}^j_{[1:N]}, \hat{\xi}^j_{[1:N]}, \hat{u}^j_{[1:N]}, \hat{s}^j, \hat{w}^j_{[1:N-1]}, \epsilon_{\text{abs}}, \epsilon_{\text{rel}})$
    if TERMINATE = TRUE **then** ▷ stopping criterion
      break
    end if
  end if
end for
$x^\star_{[1:N]} \leftarrow l_{x_{1_{\text{inv}}}} \hat{x}^{j+1} + l_{x_{2_{\text{inv}}}} \hat{\xi}^{j+1}$ ▷ recover original primal variables
$\xi^\star_{[1:N]} \leftarrow l_{\xi_{\text{inv}}} \hat{\xi}^{j+1}_{[1:N]}$
$u^\star_{[1:N]} \leftarrow l_{u_{\text{inv}}} \hat{u}^{j+1}_{[1:N]}$
$s^\star \leftarrow l_{s_{\text{inv}}} \hat{s}^{j+1}$
$\hat{w}^\star_{[1:N-1]} \leftarrow \hat{w}^{j+1}_{[1:N-1]}$ ▷ retain transformed dual variable

Return: $x^\star_{[1:N]}$, $\xi^\star_{[1:N]}$, $u^\star_{[1:N]}$, $s^\star$, $\hat{w}^\star_{[1:N-1]}$

Algorithm 7 Customized stopping criterion evaluation:

$$\text{STOPPING}_{\text{custom}}\Big(\hat{x}^{j+1}_{[1:N]},\, \hat{\xi}^{j+1}_{[1:N]},\, \hat{u}^{j+1}_{[1:N]},\, \hat{s}^{j+1},\, \hat{w}^{j+1}_{[1:N-1]},\, \hat{x}^{j}_{[1:N]},\, \hat{\xi}^{j}_{[1:N]},\, \hat{u}^{j}_{[1:N]},\, \hat{s}^{j},\, \hat{w}^{j}_{[1:N-1]},\, \epsilon_{\text{abs}},\, \epsilon_{\text{rel}}\Big)$$

Inputs: $\hat{x}^{j+1}_{[1:N]}$, $\hat{\xi}^{j+1}_{[1:N]}$, $\hat{u}^{j+1}_{[1:N]}$, $\hat{s}^{j+1}$, $\hat{w}^{j+1}_{[1:N-1]}$, $\hat{x}^{j}_{[1:N]}$, $\hat{\xi}^{j}_{[1:N]}$, $\hat{u}^{j}_{[1:N]}$, $\hat{s}^{j}$, $\hat{w}^{j}_{[1:N-1]}$, ϵ_{abs}, ϵ_{rel}

$\hat{z}^{j+1}_{\infty} \leftarrow s^{j+1}$
$\hat{z}^{j}_{\infty} \leftarrow s^{j}$
$\hat{z}^{\Delta j}_{\infty} \leftarrow \left| s^{j+1} - s^{j} \right|$
for $k \leftarrow 1{:}N$ **do**
    $\hat{z}^{j+1}_{\infty} \leftarrow \max\left\{ \hat{z}^{j+1}_{\infty},\, \|\hat{x}^{j+1}_{k}\|_{\infty},\, \|\hat{\xi}^{j+1}_{k}\|_{\infty},\, \|\hat{u}^{j+1}_{k}\|_{\infty} \right\}$
    $\hat{z}^{j}_{\infty} \leftarrow \max\left\{ \hat{z}^{j}_{\infty},\, \|\hat{x}^{j}_{k}\|_{\infty},\, \|\hat{\xi}^{j}_{k}\|_{\infty},\, \|\hat{u}^{j}_{k}\|_{\infty} \right\}$
    $\hat{z}^{\Delta j}_{\infty} \leftarrow \max\left\{ \hat{z}^{\Delta j}_{\infty},\, \|\hat{x}^{j+1}_{k} - \hat{x}^{j}_{k}\|_{\infty},\, \|\hat{\xi}^{j+1}_{k} - \hat{\xi}^{j}_{k}\|_{\infty},\, \|\hat{u}^{j+1}_{k} - \hat{u}^{j}_{k}\|_{\infty} \right\}$
end for
$\hat{w}^{j+1}_{\infty} \leftarrow 0$
$\hat{w}^{j}_{\infty} \leftarrow 0$
$\hat{w}^{\Delta j}_{\infty} \leftarrow 0$
for $k \leftarrow 1{:}N-1$ **do**
    $\hat{w}^{j+1}_{\infty} \leftarrow \max\left\{ \hat{w}^{j+1}_{\infty},\, \|w^{j+1}_{k}\|_{\infty} \right\}$
    $\hat{w}^{j}_{\infty} \leftarrow \max\left\{ \hat{w}^{j}_{\infty},\, \|w^{j}_{k}\|_{\infty} \right\}$
    $\hat{w}^{\Delta j}_{\infty} \leftarrow \max\left\{ \hat{w}^{\Delta j}_{\infty},\, \|w^{j+1}_{k} - w^{j}_{k}\|_{\infty} \right\}$
end for
if $\hat{z}^{\Delta j}_{\infty} \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{ \hat{z}^{j+1}_{\infty},\, \hat{z}^{j}_{\infty} \right\}$ **and** $\hat{w}^{\Delta j}_{\infty} \leq \epsilon_{\text{abs}} + \epsilon_{\text{rel}} \max\left\{ \hat{w}^{j+1}_{\infty},\, \hat{w}^{j}_{\infty} \right\}$ **then**
    TERMINATE $\leftarrow$ TRUE
else
    TERMINATE $\leftarrow$ FALSE
end if

Return: TERMINATE

VII. Results

We benchmark $\text{PIPG}_{\text{custom}}$ against three state-of-the-art convex optimization solvers: ECOS, MOSEK, and GUROBI [75–77], by means of a lunar approach-phase test case, with a fixed final attitude quaternion, q_f. The $\text{PIPG}_{\text{custom}}$ solver is implemented via C code, generated using the MATLAB Coder [78, 79]. The YALMIP convex optimization modeling tool in MATLAB is used to parse the problem and interface with the off-the-shelf solvers [80]. All trials are run on a 2018 MacBook Pro with a 2.6 GHz 6-core Intel Core i7 processor and 16 GB of RAM.

For consistency, the DQG problem instance is set up such that each benchmarked solver solves the problem to a predetermined open-loop accuracy in exactly 5 SeCO iterations. The entire DQG problem is solved 100 times and the mean total (across all SeCO iterations) discretization-, parse-, and solve-times are reported, as shown in Figures 11. The same procedure is carried out across 4 different problem sizes, representative of onboard guidance: $N \in \{10, 15, 20, 25\}$, where N is the number of discrete temporal nodes. The terminal position and velocity error tolerances (between the computed solution and the open-loop single-shot integrated trajectory) are set to 10 m and 0.25 m/s, respectively—similar to the tolerances chosen in [21]. The DQG parameter values chosen for the benchmark test are given in Table 1. The 3-dimensional landing trajectory and the line-of-sight angle as a function of time (corresponding to $N = 15$), obtained via $\text{PIPG}_{\text{custom}}$, are shown in Figures 8 and 9, respectively.

We observe that the solution framework (SeCO) itself leads to a speedup, regardless of the solver chosen, when compared with previously used SCP methods and solve-times reported in the literature [21, 22, 24]. Further, PIPG is significantly faster than the solvers it is benchmarked against, as shown in Figure 10, and over an order of magnitude faster than the previously reported mean solve-time for DQG [21] for the same problem size.

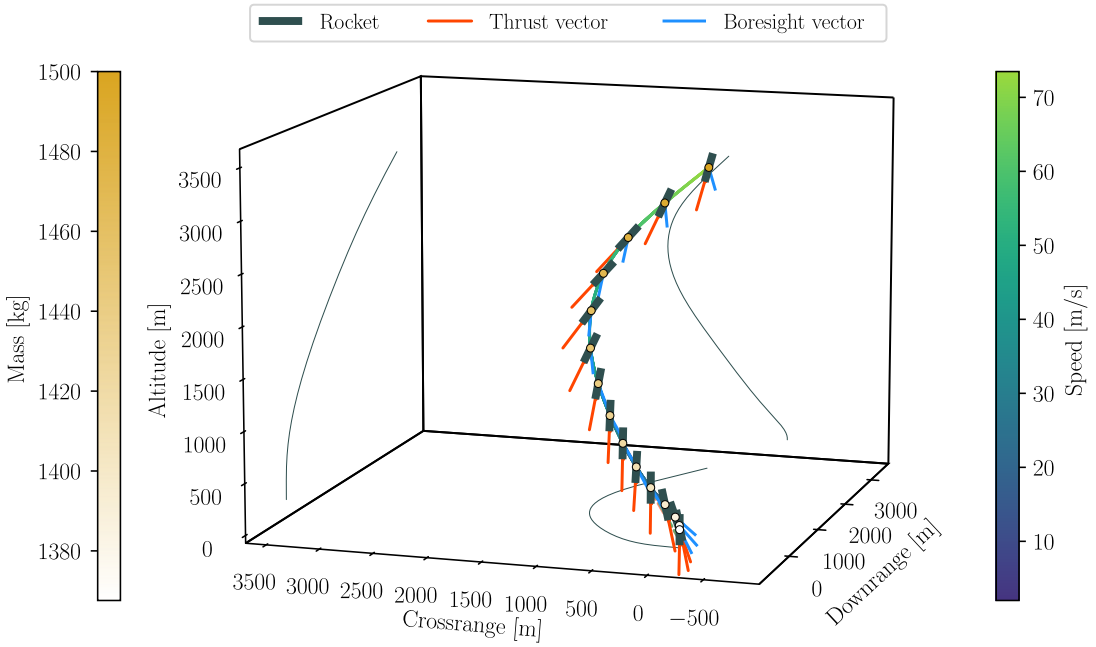

Fig. 8 The 3D landing trajectory obtained via SeCO in real-time ($N = 15$).

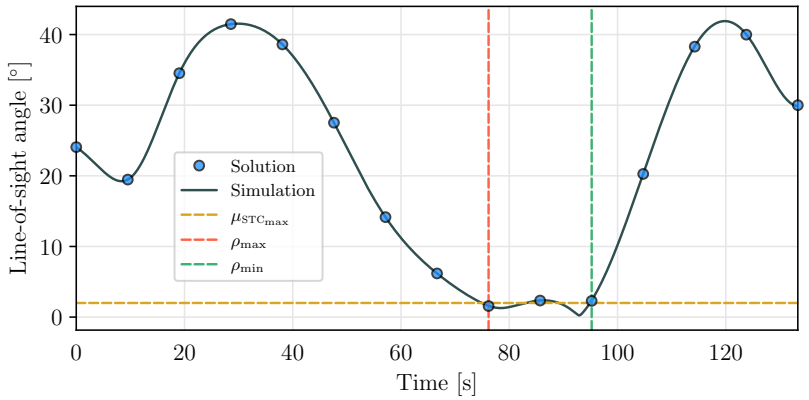

Fig. 9 The line-of-sight angle, which is constrained to be within 2° in the trigger window ($N = 15$).

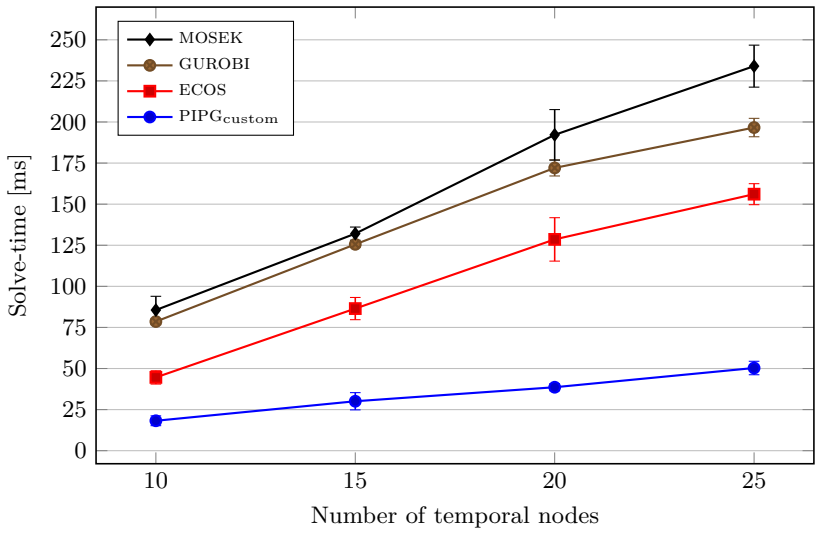

Fig. 10 Solve-time comparison between the DQG-customized version of PIPG and three state-of-the-art convex optimization solvers. The error bars indicate three standard deviations ($\pm 3\sigma$).

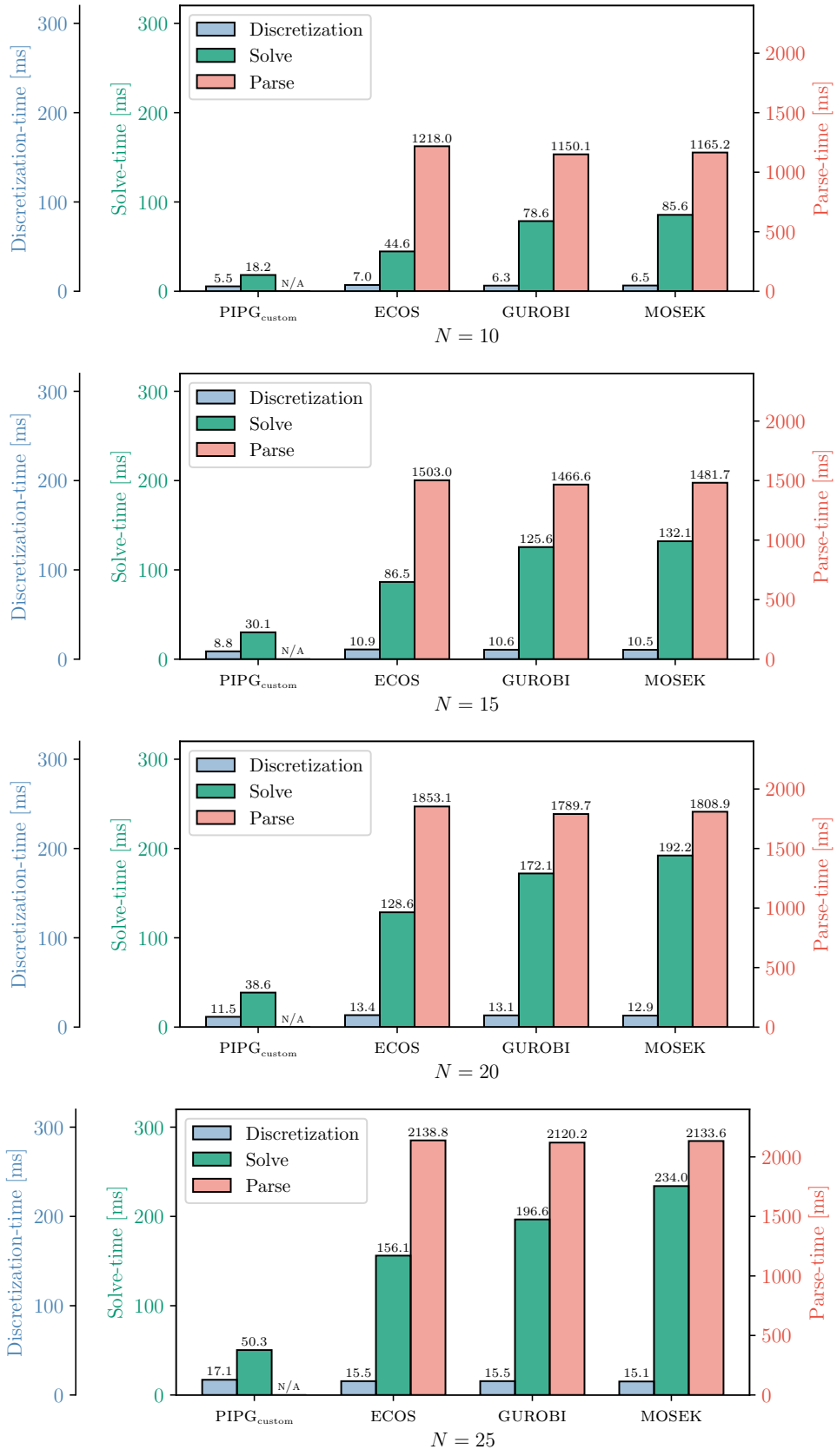

Fig. 11 DQG benchmark test results.

Parameter	Value
g	$1.625\ \mathrm{m\,s^{-2}}$
g_0	$9.81\ \mathrm{m\,s^{-2}}$
$I_{\mathrm{sp}_{\mathrm{ME}}}$	300 s
α_{ME}	$\frac{1}{I_{\mathrm{sp}_{\mathrm{ME}}} g_0}\ \mathrm{s\,m^{-1}}$
$I_{\mathrm{sp}_{\mathrm{RCS}}}$	200 s
α_{RCS}	$\frac{1}{I_{\mathrm{sp}_{\mathrm{RCS}}} g_0}\ \mathrm{s\,m^{-1}}$
$\tau_{\max}$	$50\ \mathrm{kg\,m^2\,s^{-2}}$
$T_{\max}$	$3000\ \mathrm{kg\,m\,s^{-2}}$
$T_{\min}$	$600\ \mathrm{kg\,m\,s^{-2}}$
$\dot{T}_{\max}$	$0.75 \cdot (T_{\max} - T_{\min})\ \mathrm{kg\,m\,s^{-3}}$
$\delta_{\max}$	5°
$\dot{\delta}_{\max}$, $\dot{\phi}_{\max}$	$5^\circ\,\mathrm{s^{-1}}$
l_{CM}	1 m
$p_{\mathcal{B}}$	$\left[0.5,\ 0,\ -\frac{\sqrt{3}}{2}\right]^\top$
m_i	1500 kg
m_f	750 kg
J	$\mathrm{diag}\{4.2,\ 4.2,\ 0.6\}\ \mathrm{m^2}$

Parameter	Value
$\theta_{\max}$	90°
$\omega_{\max}$	$5^\circ\,\mathrm{s^{-1}}$
$v_{\max}$	$90\ \mathrm{m\,s^{-1}}$
$h_{\min}$	100 m
$\rho_{\max}$	1250 m
$\rho_{\min}$	500 m
$\theta_{\mathrm{STC}_{\max}}$	20°
$\omega_{\mathrm{STC}_{\max}}$	$1^\circ\,\mathrm{s^{-1}}$
$v_{\mathrm{STC}_{\max}}$	$30\ \mathrm{m\,s^{-1}}$
$\mu_{\mathrm{STC}_{\max}}$	2°
$r_{\mathcal{I}_i}$	$[3000,\ 600,\ 3000]^\top$ m
$r_{\mathcal{I}_f}$	$[0,\ 0,\ 100]^\top$ m
$v_{\mathcal{I}_i}$	$[-60,\ 30,\ -30]^\top\ \mathrm{m\,s^{-1}}$
$v_{z_{\mathcal{I}_f}}$	$-2\ \mathrm{m\,s^{-1}}$
q_i	$[-0.15,\ 0.3,\ -1,\ 1]^\top$ (normalized)
q_f	$[0,\ 0,\ -1.25,\ 1]^\top$ (normalized)
$\omega_{\mathcal{B}_i}$	$[0,\ 0,\ 0]^\top\ ^\circ\,\mathrm{s^{-1}}$

Table 1 The DQG parameter values chosen for the solver benchmark test.

VIII. Conclusions

Sequential conic optimization (SeCO) combines sequential convex programming (SCP) with first-order conic optimization to solve difficult trajectory optimization problems, such as the dual quaternion-based 6-DoF powered-descent guidance (DQG) problem, in real-time. First-order optimization solvers, such as PIPG, are attractive for: (1) real-time applications (given their execution speed); (2) implementation onboard resource-constrained systems (owing to the small footprint of the resulting codebase); and, (3) verification and validation (since they only rely on simple computations). Recent advances have enabled them to match (and even outdo) solvers based on interior-point methods (IPMs), in terms of performance. Further, PIPG is amenable to warm-starting and performance-efficient customization for trajectory optimization problems.

We formulate the nonconvex DQG problem—with mission-critical constraints—in compliance with the SeCO framework, and solve it using $\mathrm{PIPG}_{\mathrm{custom}}$, a custom first-order conic optimization solver developed for this application. This solver is able to solve the entire nonconvex problem in a matter of milliseconds, and is much faster than other state-of-the-art convex optimization solvers across varying problem sizes.

Planned future work includes running comprehensive benchmark tests onboard the NASA SPLICE descent and landing computer (DLC), and devising algorithms to automatically tune the solver parameter λ.

Acknowledgements

The authors thank the members of the Autonomous Controls Laboratory (ACL) at the University of Washington, especially Dayou Luo and Samet Uzun, for the discussions on solver development and acceleration, and Govind Chari, for the detailed review of the manuscript. We thank Benjamin Chung from Northeastern University for the discussions on sparse linear algebra. We also thank the members of the Flight Mechanics and Trajectory Design branch (EG5) at the NASA Johnson Space Center, especially Gavin Mendeck, Breanna Johnson, Dan Matz, and Ron Sostaric; and Javier Doll from Draper, for their valuable guidance, insight, and the many helpful discussions. The authors give their special thanks to the developers of the original DQG algorithm: Taylor Reynolds, Miki Szmuk, and Danylo Malyuta, for enabling this research and also their ongoing support. This research was supported by NASA grant NNX17AH02A and was partially carried out at the NASA Johnson Space Center; Government sponsorship is acknowledged.

References

[1] Smith, M., Craig, D., Herrmann, N., Mahoney, E., Krezel, J., McIntyre, N., and Goodliff, K., "The Artemis Program: An Overview of NASA's Activities to Return Humans to the Moon," *2020 IEEE Aerospace Conference*, 2020, pp. 1–10. https://doi.org/10.1109/AERO47225.2020.9172323.

[2] Chavers, G., Suzuki, N., Smith, M., Watson-Morgan, L., Clarke, S. W., Engelund, W. C., Aitchison, L., McEniry, S., Means, L., DeKlotz, M., et al., "NASA's Human Lunar Landing Strategy," *70th International Astronautical Congress*, 2019, pp. 1–6.

[3] Chavers, G., Watson-Morgan, L., Smith, M., Suzuki, N., and Polsgrove, T., "NASA's Human Landing System: The Strategy for the 2024 Mission and Future Sustainability," *2020 IEEE Aerospace Conference*, 2020, pp. 1–9. https://doi.org/10.1109/AERO47225.2020.9172599.

[4] Petersen, D., Charvat, J., Somers, J., Pattarini, J., Stenger, M., Van Baalen, M., and Lee, S., "Apollo to Artemis: Mining 50-Year Old Records to Inform Future Human Lunar Landing Systems," *LSAH Newsletter*, Vol. 25, No. 1, 2020, pp. 6–7.

[5] Musk, E., "Making humans a multi-planetary species," *New Space*, Vol. 5, No. 2, 2017, pp. 46–61.

[6] Muirhead, B. K., Nicholas, A. K., Umland, J., Sutherland, O., and Vijendran, S., "Mars Sample Return Campaign Concept Status," *Acta Astronautica*, Vol. 176, 2020, pp. 131–138. https://doi.org/10.1016/j.actaastro.2020.06.026, URL https://doi.org/10.1016/j.actaastro.2020.06.026.

[7] Carson, J. M., Munk, M. M., Sostaric, R. R., Estes, J. N., Amzajerdian, F., Blair, J. B., Rutishauser, D. K., Restrepo, C. I., Dwyer-Cianciolo, A. M., Chen, G., et al., "The SPLICE project: Continuing NASA development of GN&C technologies for safe and precise landing," *AIAA Scitech 2019 Forum*, 2019, p. 0660.

[8] Klumpp, A. R., "Apollo lunar descent guidance," *Automatica*, Vol. 10, No. 2, 1974, pp. 133–146.

[9] San Martin, A. M., Lee, S. W., and Wong, E. C., "The development of the MSL guidance, navigation, and control system for entry, descent, and landing," *AAS*, 2013.

[10] Casoliva, J., Singh, G., Brugarolas, P., and Way, D. W., "Reconstructed Flight Performance of the Powered Descent Guidance and Control System for the Mars 2020 Perseverance Mission," *AAS*, 2021.

[11] Quaide, W., and Oberbeck, V., "Geology of the Apollo landing sites," *Earth-Science Reviews*, Vol. 5, No. 4, 1969, pp. 255–278.

[12] Açıkmeşe, B., and Ploen, S. R., "Convex programming approach to powered descent guidance for Mars landing," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366.

[13] Açıkmeşe, B., and Blackmore, L., "Lossless convexification of a class of optimal control problems with non-convex control constraints," *Automatica*, Vol. 47, No. 2, 2011, pp. 341–347.

[14] Açıkmeşe, B., Carson, J. M., and Blackmore, L., "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, 2013, pp. 2104–2113.

[15] Açıkmeşe, B., Aung, M., Casoliva, J., Mohan, S., Johnson, A., Scharf, D., Masten, D., Scotkin, J., Wolf, A., and Regehr, M. W., "Flight testing of trajectories computed by G-FOLD: Fuel optimal large divert guidance algorithm for planetary landing," *AAS/AIAA spaceflight mechanics meeting*, 2013, pp. 863–870.

[16] Scharf, D. P., Açıkmeşe, B., Dueri, D., Benito, J., and Casoliva, J., "Implementation and experimental demonstration of onboard powered-descent guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 213–229.

[17] Szmuk, M., Reynolds, T., Açıkmeşe, B., Mesbahi, M., and Carson, J. M., "Successive convexification for 6-DoF powered descent guidance with compound state-triggered constraints," *AIAA Scitech 2019 Forum*, 2019, p. 0926.

[18] Reynolds, T., Szmuk, M., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "A state-triggered line of sight constraint for 6-DoF powered descent guidance problems," *AIAA Scitech 2019 Forum*, 2019, p. 0924.

[19] Szmuk, M., Malyuta, D., Reynolds, T. P., Mceowen, M. S., and Açıkmeşe, B., "Real-time quad-rotor path planning using convex optimization and compound state-triggered constraints," *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2019, pp. 7666–7673.

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

[21] Reynolds, T. P., Szmuk, M., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson III, J. M., "Dual quaternion-based powered descent guidance with state-triggered constraints," *Journal of Guidance, Control, and Dynamics*, 2020.

[22] Reynolds, T., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "A real-time algorithm for non-convex powered descent guidance," *AIAA Scitech 2020 Forum*, 2020, p. 0844.

[23] Rutishauser, D., Ramadorai, R., Prothro, J., Fleming, T., and Fidelman, P., "NASA and Blue Origin Collaborative Assessment of Precision Landing Algorithms and Computing," *AIAA Scitech 2021 Forum*, 2021, p. 0377.

[24] Strohl, L., Doll, J., Fritz, M., Berning, A. W., White, S., Bieniawski, S. R., Carson, J. M., and Açıkmeşe, B., "Implementation of a Six Degree of Freedom Precision Lunar Landing Algorithm Using Dual Quaternion Representation,"

AIAA SCITECH 2022 Forum, 2022, p. 1831.

[25] Fritz, M., Doll, J., Ward, K. C., Mendeck, G., Sostaric, R. R., Pedrotty, S., Kuhl, C., Açıkmeşe, B., Bieniawski, S. R., Strohl, L., et al., "Post-Flight Performance Analysis of Navigation and Advanced Guidance Algorithms on a Terrestrial Suborbital Rocket Flight," *AIAA SCITECH 2022 Forum*, 2022, p. 0765.

[26] Dueri, D., Zhang, J., and Açıkmeşe, B., "Automated custom code generation for embedded, real-time second order cone programming," *IFAC Proceedings Volumes*, Vol. 47, No. 3, 2014, pp. 1605–1612.

[27] Dueri, D., Açıkmeşe, B., Scharf, D. P., and Harris, M. W., "Customized real-time interior-point methods for onboard powered-descent guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 197–212.

[28] Yu, Y., Elango, P., and Açıkmeşe, B., "Proportional-integral projected gradient method for model predictive control," *IEEE Control Systems Letters*, Vol. 5, No. 6, 2020, pp. 2174–2179.

[29] Yu, Y., Elango, P., Topcu, U., and Açıkmeşe, B., "Proportional-Integral Projected Gradient Method for Conic Optimization," *arXiv preprint arXiv:2108.10260*, 2021.

[30] Elango, P., Kamath, A. G., Yu, Y., Carson III, J. M., Mesbahi, M., and Açıkmeşe, B., "A Customized First-Order Solver for Real-Time Powered-Descent Guidance," *AIAA SciTech 2022 Forum*, 2022, p. 0951.

[31] Yu, Y., Elango, P., Açıkmeşe, B., and Topcu, U., "Extrapolated Proportional-Integral Projected Gradient Method for Conic Optimization," *arXiv preprint arXiv:2203.04188*, 2022.

[32] Malyuta, D., Reynolds, T. P., Szmuk, M., Lew, T., Bonalli, R., Pavone, M., and Açıkmeşe, B., "Convex Optimization for Trajectory Generation: A Tutorial on Generating Dynamically Feasible Trajectories Reliably and Efficiently," *IEEE Control Systems*, Vol. 42, No. 5, 2022, pp. 40–113. URL https://doi.org/10.1109/mcs.2022.3187542.

[33] Kamath, A. G., Elango, P., Yu, Y., Mceowen, S., Carson III, J. M., and Açıkmeşe, B., "Real-Time Sequential Conic Optimization for Multi-Phase Rocket Landing Guidance," *arXiv preprint arXiv:2212.00375*, 2022.

[34] Reynolds, T. P., *Computational Guidance and Control for Aerospace Systems*, University of Washington, 2020.

[35] Lee, U., and Mesbahi, M., "Optimal power descent guidance with 6-DoF line of sight constraints via unit dual quaternions," *AIAA Guidance, Navigation, and Control Conference*, 2015, p. 0319.

[36] Lee, U., and Mesbahi, M., "Dual quaternions, rigid body mechanics, and powered-descent guidance," *2012 ieee 51st ieee conference on decision and control (cdc)*, IEEE, 2012, pp. 3386–3391.

[37] Lee, U., and Mesbahi, M., "Constrained autonomous precision landing via dual quaternions and model predictive control," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, 2017, pp. 292–308.

[38] Reynolds, T. P., and Mesbahi, M., "Coupled 6-DOF control for distributed aerospace systems," *2018 IEEE Conference on Decision and Control (CDC)*, IEEE, 2018, pp. 5294–5299.

[39] Restrepo, C., Lovelace, R., Sostaric, R., and Carson, J., "NASA SPLICE Project: Developing the Next Generation Hazard Detection System," Tech. rep., NASA, 2019.

[40] Restrepo, C. I., Chen, P.-T., Sostaric, R. R., and Carson, J. M., "Next-generation NASA hazard detection system development," *AIAA Scitech 2020 Forum*, 2020, p. 0368.

[41] Malyuta, D., Reynolds, T., Szmuk, M., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Discretization performance and accuracy analysis for the rocket powered descent guidance problem," *AIAA Scitech 2019 Forum*, 2019, p. 0925.

[42] Antsaklis, P. J., and Michel, A. N., *Linear systems*, Basel, Switzerland: Birkhauser, 2006.

[43] Mao, Y., Szmuk, M., and Açıkmeşe, B., "Successive convexification of non-convex optimal control problems and its convergence properties," *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 3636–3641.

[44] Szmuk, M., Pascucci, C. A., Dueri, D., and Açıkmeşe, B., "Convexification and real-time on-board optimization for agile quad-rotor maneuvering and obstacle avoidance," *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2017, pp. 4862–4868.

[45] Szmuk, M., Pascucci, C. A., and Açıkmeşe, B., "Real-time quad-rotor path planning for mobile obstacle avoidance using convex optimization," *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1–9.

[46] Malyuta, D., Reynolds, T., Szmuk, M., Açıkmeşe, B., and Mesbahi, M., "Fast trajectory optimization via successive convexification for spacecraft rendezvous with integer constraints," *AIAA Scitech 2020 Forum*, 2020, p. 0616.

[47] Malyuta, D., and Açıkmeşe, B., "Fast Homotopy for Spacecraft Rendezvous Trajectory Optimization with Discrete Logic," *arXiv preprint arXiv:2107.07001*, 2021.

[48] Malyuta, D., "Convex Optimization in a Nonconvex World: Applications for Aerospace Systems," Ph.D. thesis, University of Washington, 2021.

[49] Mceowen, S., and Açıkmeşe, B., "Hypersonic Entry Trajectory Optimization via Successive Convexification with Abstracted Control," *AIAA SCITECH 2022 Forum*, 2022, p. 0950.

[50] Mceowen, S., Kamath, A. G., Elango, P., Kim, T., Buckner, S. C., and Açıkmeşe, B., "High-Accuracy 3-DoF Hypersonic Reentry Guidance via Sequential Convex Programming," *AIAA SCITECH 2023 Forum*, 2023.

[51] Kim, T., Elango, P., Malyuta, D., and Açıkmeşe, B., "Guided Policy Search using Sequential Convex Programming

for Initialization of Trajectory Optimization Algorithms," *2022 American Control Conference (ACC)*, IEEE, 2022, pp. 3572–3578.

[52] Kenwright, B., "Dual-quaternions: From classical mechanics to computer graphics and beyond," *Princeton, Citeseer*, 2012.

[53] D'Souza, C., "An optimal guidance law for planetary landing," *Guidance, Navigation, and Control Conference*, 1997, p. 3709.

[54] Bauschke, H. H., Bui, M. N., and Wang, X., "Projecting onto the intersection of a cone and a sphere," *SIAM Journal on Optimization*, Vol. 28, No. 3, 2018, pp. 2158–2188.

[55] Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, Vol. 3, No. 1, 2011, pp. 1–122.

[56] Szmuk, M., Reynolds, T. P., Açıkmeşe, B. A., Mehran, M., and Carson III, J. M., "Successive Convexification for 6-DoF Powered Descent Guidance with Compound State-Triggered Constraints," *AIAA Guidance, Navigation, and Control Conference*, San Diego, 2019, p. 0926.

[57] Malyuta, D., Yu, Y., Elango, P., and Açıkmeşe, B., "Advances in trajectory optimization for space vehicle control," *Annual Reviews in Control*, Vol. 52, 2021, pp. 282–315.

[58] Bauschke, H. H., Combettes, P. L., et al., *Convex analysis and monotone operator theory in Hilbert spaces*, Vol. 408, Springer, 2011.

[59] Stellato, B., Banjac, G., Goulart, P., Bemporad, A., and Boyd, S., "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, Vol. 12, No. 4, 2020, pp. 637–672.

[60] Boyd, S., El Ghaoui, L., Feron, E., and Balakrishnan, V., *Linear matrix inequalities in system and control theory*, SIAM, 1994.

[61] Ruiz, D., "A scaling algorithm to equilibrate both rows and columns norms in matrices," Tech. rep., CM-P00040415, 2001.

[62] Sinkhorn, R., and Knopp, P., "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific Journal of Mathematics*, Vol. 21, No. 2, 1967, pp. 343–348.

[63] Pock, T., and Chambolle, A., "Diagonal preconditioning for first order primal-dual algorithms in convex optimization," *2011 International Conference on Computer Vision*, IEEE, 2011, pp. 1762–1769.

[64] Giselsson, P., and Boyd, S., "Diagonal scaling in Douglas-Rachford splitting and ADMM," *53rd IEEE Conference on Decision and Control*, IEEE, 2014, pp. 5033–5039.

[65] Fougner, C., and Boyd, S., "Parameter selection and preconditioning for a graph form solver," *Emerging Applications of Control and Systems Theory*, Springer, 2018, pp. 41–61.

[66] Diamond, S., and Boyd, S., "Stochastic matrix-free equilibration," *Journal of Optimization Theory and Applications*, Vol. 172, No. 2, 2017, pp. 436–454.

[67] O'Donoghue, B., Chu, E., Parikh, N., and Boyd, S., "Conic optimization via operator splitting and homogeneous self-dual embedding," *Journal of Optimization Theory and Applications*, Vol. 169, No. 3, 2016, pp. 1042–1068.

[68] Trefethen, L. N., and Bau III, D., *Numerical linear algebra*, Vol. 50, Siam, 1997.

[69] Yu, Y., Nagpal, K., Mceowen, S., Açıkmeşe, B., and Topcu, U., "Real-Time Quadrotor Trajectory Optimization with Time-Triggered Corridor Constraints," *arXiv preprint arXiv:2208.07259*, 2022.

[70] O'Donoghue, B., "Operator splitting for a homogeneous embedding of the linear complementarity problem," *SIAM Journal on Optimization*, Vol. 31, No. 3, 2021, pp. 1999–2023.

[71] Williams, S., Oliker, L., Vuduc, R., Shalf, J., Yelick, K., and Demmel, J., "Optimization of sparse matrix-vector multiplication on emerging multicore platforms," *SC'07: Proceedings of the 2007 ACM/IEEE Conference on Supercomputing*, IEEE, 2007, pp. 1–12.

[72] Horn, R. A., and Johnson, C. R., *Matrix analysis*, Cambridge University Press, 2012.

[73] Schacke, K., "On the Kronecker product," *Master's thesis, University of Waterloo*, 2004.

[74] Broxson, B. J., "The Kronecker product," *Master's thesis, University of North Florida*, 2006.

[75] Domahidi, A., Chu, E., and Boyd, S., "ECOS: An SOCP solver for embedded systems," *2013 European Control Conference (ECC)*, IEEE, 2013, pp. 3071–3076.

[76] MOSEK ApS, *The MOSEK Optimization Toolbox for MATLAB Manual. Version 9.0.*, 2019. URL http://docs.mosek.com/9.0/toolbox/index.html.

[77] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," , 2021. URL https://www.gurobi.com.

[78] *MATLAB version 9.11.0.1769968 (R2021a)*, The Mathworks, Inc., Natick, Massachusetts, 2021.

[79] *MATLAB Coder version 5.3*, The Mathworks, Inc., Natick, Massachusetts, 2021.

[80] Lofberg, J., "YALMIP: A toolbox for modeling and optimization in MATLAB," *2004 IEEE international conference on robotics and automation (IEEE Cat. No. 04CH37508)*, IEEE, 2004, pp. 284–289.

Appendix

A. Quaternion algebra

1. Unit quaternions

$$a = \left[a_v^\top,\, a_4\right]^\top,\ b = \left[b_v^\top,\, b_4\right]^\top \in \mathbb{R}_u^4 := \left\{ q \in \mathbb{R}^4 \;\middle|\; q^\top q = 1 \right\}$$

where

$$a_v = [a_1,\, a_2,\, a_3]^\top,\ b_v = [b_1,\, b_2,\, b_3]^\top \in \mathbb{R}^3 \text{ and } a_4, b_4 \in \mathbb{R}$$

Note: All quaternions are in accordance with the scalar-last convention.

2. Conjugation

$$a^* := \left[-a_v^\top,\, a_4\right]^\top$$

3. Skew-symmetric matrix operator

$$a_v^\times := \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}$$

4. $SO(4)$ matrix operators

$$[a]_\otimes := \begin{pmatrix} a_4 I_3 + a_v^\times & a_v \\ -a_v^\top & a_4 \end{pmatrix}$$

$$[b]_\otimes^* := \begin{pmatrix} b_4 I_3 - b_v^\times & b_v \\ -b_v^\top & b_4 \end{pmatrix}$$

5. Multiplication

$$\begin{aligned} a \otimes b &:= \left[a_4 b_v + b_4 a_v + a_v \times b_v,\, a_4 b_4 - a_v^\top b_v\right]^\top \\ &= [a]_\otimes b \\ &= [b]_\otimes^* a \end{aligned}$$

6. Cross product

$$a \oslash b := \left[a_4 b_v + b_4 a_v + a_v \times b_v,\, 0\right]^\top$$

B. Dual quaternion algebra

1. Unit dual quaternions

$$\boldsymbol{a} = \left[a_1^\top,\, a_2^\top\right]^\top,\ \boldsymbol{b} = \left[b_1^\top,\, b_2^\top\right]^\top \in \mathbb{R}_u^8 := \left\{ \boldsymbol{q} = \left[q_1^\top,\, q_2^\top\right]^\top \in \mathbb{R}^8 \;\middle|\; q_1^\top q_1 = 1 \text{ and } q_1^\top q_2 = 0,\ q_1, q_2 \in \mathbb{R}^4 \right\}$$

2. Conjugation

$$\boldsymbol{a}^* := \begin{pmatrix} a_1^* \\ a_2^* \end{pmatrix}$$

3. Multiplication matrix operators

$$[\boldsymbol{a}]_{\otimes} := \begin{pmatrix} [a_1]_{\otimes} & 0_{4\times 4} \\ [a_2]_{\otimes} & [a_1]_{\otimes} \end{pmatrix}$$

$$[\boldsymbol{b}]^{*}_{\otimes} := \begin{pmatrix} [b_1]^{*}_{\otimes} & 0_{4\times 4} \\ [b_2]^{*}_{\otimes} & [b_1]^{*}_{\otimes} \end{pmatrix}$$

4. Multiplication

$$\boldsymbol{a} \otimes \boldsymbol{b} := [\boldsymbol{a}]_{\otimes}\boldsymbol{b} = [\boldsymbol{b}]^{*}_{\otimes}\boldsymbol{a}$$

5. Cross product

$$\boldsymbol{a} \oslash \boldsymbol{b} := [a_1 \oslash b_1,\ a_1 \oslash b_2 + a_2 \oslash b_1]^{\top}$$