Six-Degrees-of-Freedom Rocket Landing Optimization via Augmented Convex-Concave Decomposition[*]

Marco Sagliano [†], David Seelbinder [‡], and Stephan Theil[§]
German Aerospace Center, Bremen, Germany, 28359

Ping Lu[¶]
San Diego State University, San Diego, CA 92182

In this paper an Augmented Convex-Concave Decomposition (ACCD) method for treating nonlinear equality constraints in an otherwise convex problem is proposed. This augmentation improves greatly the feasibility of the problem when compared to the original convex-concave decomposition approach. The effectiveness of the ACCD is demonstrated by solving a fuel-optimal 6-DoF rocket landing problem in atmosphere, subject to multiple nonlinear equality constraints. Compared with known approaches such as Sequential Convex Programming where conventional linearization (with or without slack variables) is employed to treat those nonlinear equality constraints, it is shown that the proposed ACCD leads to more robust convergence of the solution process, and a more interpretable behavior of the sequential convex algorithm. The methodology can also be applied effectively in the case where the determination of discrete controls or decision making variables needs to be made, such as the on-off use of reaction control system thrusters in the rocket landing problem, without the need for a mixed integer solver. Numerical results are shown for a representative, reusable rocket benchmark problem.

I. Introduction

In recent years there has been an increasing use of convex-optimization-based methods in the context of guidance and trajectory optimization for a wide variety of aerospace applications, ranging from entry guidance problems [1, 2], pinpointing landing of rockets [3–6], to low-thrust interplanetary trajectory computation [7, 8], and rendezvous and docking problems [9, 10]. The application of these techniques comes in several flavors (see [11] for a recent and complete survey), depending on the type of problem under examination, and the technical difficulty needed to overcome in a specific case. Examples of these difficulties include, (but are not limited to) free-final time problems [12, 13], inclusion of non-convex aerodynamic effects [14, 15], state-triggered constraints [16, 17], need to retain some features

[*]Presented as Paper 2023-2005 at the AIAA Scitech 2023 Forum, National Harbor, MD, and Virtual, January 23–27, 2023

[†]GNC Senior Researcher, German Aerospace Center, AIAA Senior Member, Corresponding author

[‡]G&C Group Head, German Aerospace Center

[§]GNC Department Head, German Aerospace Center

[¶]Professor and Chair, Department of Aerospace Engineering, Fellow AIAA of the original nonlinear problem [18], and choice of a different independent variable rather than time [19, 20].

One such difficult issue may arise from problems having nonlinear equality constraints. It is well-known that the only equality constraints convex optimization problems allow are linear ones [21]. Therefore the most common way to deal with nonlinear equality constraints is the use of linearization techniques built around the current iterate (sub-solution). However, this approach does not always lead to the convergence of the iterative solution process, as shown for the Natural-Motion Circumnavigation Injection problem [22].

To circumvent the difficulties associated with conventional linearization, the Convex-Concave Decomposition (CCD) was proposed for a class of convex nonlinear equality constraints [22], leading to excellent results for the aforementioned category of problems [23]. However, for different problems this methodology might suffer from infeasibility issues, especially at the beginning of the sequential convex programming (SCP) approach. The interaction between dynamics, boundary conditions and nonlinear equality constraints can cause convergence issues of the first subproblem. Even though most of the time the process converges to the proper solution it is not advisable to rely on a process that might generate infeasible sub-solutions along the iterative process. It is important to stress that each new sub-problem is built upon the current subsolution, i.e., the solution obtained at the previous iteration step. If such a solution is the result of an infeasible problem, there is no guarantee that such solution is a meaningful point to build the next subproblem upon. This becomes even more problematic if we consider that the outcome of an infeasible problem might change depending on the specific convex solver in use, making the solution process in some sense not deterministic, and, although there are algorithmic approaches that take infeasible paths to achieve the solution, it might be worth to generate feasible subproblems from the beginning of the iterative procedure. Finally, it is highly useful that the algorithm remains interpretable at each step to make the analysis of its behavior simpler and more direct.

To overcome these difficulties this paper extends the idea of convex-concave decomposition in the context of convex optimization applied to optimal control problems in the presence of convex nonlinear equality constraints. We analyze the application of the original convex-concave decomposition method in Ref. [22] to a different class of problems. We will show that, while the original convex-concave decomposition is effective for the problems in Ref. [22], it may suffer from infeasibility issues in certain cases. Similar issues can also plague state-of-the-art successive convex optimization techniques when the considered class of nonlinear equality constraints are part of the problem to be solved.

This paper contains three novel contributions with respect to the state of the art. First, we provide an in-depth analysis on the feasibility space for the different optimization problems stemming from state-of-the-art successive convex optimization and the original convex-concave decomposition in presence of nonlinear equality constraints. By leveraging the use of the exact penalty function theory we show that, with the proposed methodology, dubbed Augmented Convex-Concave Decomposition (ACCD), the feasibility of the problem is ensured from the beginning of the iterative process, leading in many cases to a smoother and faster convergence of the overall algorithm. The proposed method is tested to solve a free-time, multi-constrained, complex problem of optimal landing of a reusable rocket subject to six-degree-of-freedom (6DOF) dynamics inside an atmosphere. The last contribution involves the use of the proposed methodology to solve a problem with discrete variables or controls without the need for a dedicated mixed integer solver [24], or applying homotopic paths in conjunction with smooth approximations to the discrete variables [25]. We demonstrate that the ACCD can efficiently handle discrete controls when on-off decision of the reaction control system (RCS) thrusters is part of the controls of the problem. To the best of our knowledge, this is the first effort in a complete 6-DoF rocket landing approach that explicitly includes such a discrete decision-making logic to govern the roll motion in the trajectory generation and optimization process.

An additional benefit of the proposed ACCD method resides in the fact that it leads to a well-understandable and interpretable behavior of the algorithm and the slack variables in each iteration. We assess the behavior of the proposed ACCD methodology and further investigate the feasibility domain of Sequential Convex methods in the presence of Nonlinear Equality Constraints (NECs), how such feasible space is modified by the use of the standard Convex-Concave Decomposition, and finally how the feasibility is improved by the use of the proposed ACCD for common NECs such as the quaternion norm constraint.

The paper is organized as follows: Section II contains a general description of state-of-the-art sequential convex programming methods, while Section III describes the core idea and the limitations of the classic convex-concave-decomposition. In Section IV we describe the augmented convex-concave decomposition technique proposed. In addition, the feasibility space analysis of the three methodologies is discussed through the corresponding sections. Section V describes the 6-DoF Rocket Landing Optimal Control problem formulated with the use of the proposed Augmented Convex-Concave Decomposition, and the corresponding transcription based on hp pseudospectral convex techniques, whereas Section VI shows some numerical examples. Finally, in Section VII we draw some conclusions about the work performed and the results we have obtained.

II. Overview of Sequential Convex Programming

Sequential Convex Programming is a methodology aiming at solving non-convex optimization problems by approximating them with a sequence of convex sub-problems that will iteratively converge to the solution of the original problems. The methodology has found many applications, ranging from atmospheric hypersonic entry [2], [26], to low-thrust optimization [8], guidance of reusable rockets in the aerodynamic phase [15], and during the powered descent [1, 17]. Although convergence proofs are limited to specific cases [27, 28] the methodology works heuristically well in a large number of applications. This property, together with its flexibility and the general high computational speed that characterize convex optimization-based solvers, has made it very popular in recent years.

For a quick review of sequential convex programming, let us define a general optimal control problem in the Bolza form, with non-convex dynamics and path constraints: the cost function is assumed to be convex, e.g., in linear or quadratic form. These are not the only choices, but include a large variety of problems of interest, including atmospheric entry, and powered descent and landing [1, 2, 19].

$$\text{minimize } J = \phi(t_F, \mathbf{x}(t_F)) + \int_{t_0}^{t_F} L(t, \mathbf{x}(t), \mathbf{u}(t))dt \tag{1}$$

Since we are considering a dynamical system, we also have the following constraints,

$$\dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}) \tag{2}$$

$$\mathbf{g}(t, \mathbf{x}, \mathbf{u}) \leq 0 \tag{3}$$

$$\mathbf{x}(t_F) \in \chi \tag{4}$$

and where $\mathbf{x} \in \mathbb{R}^{n_s}$ is the n_s-dimensional state, $\mathbf{u} \in \mathbb{R}^{n_c}$ is the n_c-dimensional control, and $\mathbf{g} \in \mathbb{R}^{n_g}$ are some nonlinear path constraints. Typically (although not necessarily) it is assumed that the initial time t_0 is fixed and known (assumed to be the case for the numerical demonstrations in this work), while the final time t_F can be free or fixed, depending on the specific problem. Finally, the final state $\mathbf{x}(t_F)$ has to belong to a given set χ, which can be finite or infinite (i.e., unconstrained) for the different states depending on the specific problem. In a sequential convex programming fashion the problem is reformulated as follows: the cost remains the same given the hypothesis of convexity (retained for convenience: in many problems of interest it is the case, or it can be always reintroduced through the use of a slack variable and an additional nonlinear constraint). When the dynamics are linearized about the previous iterate of $\{\mathbf{x}_k(t), \mathbf{u}_k(t)\}$, the linearized dynamics are then

$$\dot{\mathbf{x}} = \mathbf{A}(t)\mathbf{x} + \mathbf{B}(t)\mathbf{u} + \mathbf{C}\boldsymbol{\nu} + \mathbf{E}(t)t_F + \mathbf{h}(t) \tag{5}$$

where the matrices $\mathbf{A}$ and $\mathbf{B}$ are the Jacobians of $\mathbf{f}(t, \mathbf{x}, \mathbf{u})$ with respect to $\mathbf{x}$ and $\mathbf{u}$, evaluated along $\{\mathbf{x}_k(t), \mathbf{u}_k(t)\}$ (therefore they are explicit functions of time). The matrix $\mathbf{C}$ maps the so-called *virtual controls* $\boldsymbol{\nu}$, onto the corresponding states to address the *artificial infeasibility* phenomenon [27], whereas the term $\mathbf{h}$ includes the rest terms in the linearization dependent solely on $\{\mathbf{x}_k(t), \mathbf{u}_k(t)\}$. Depending on whether the final time is free or fixed, as well as how the time dependency is embedded in the problem the term $\mathbf{E}$ can be a zero matrix of proper size, or can assume different explicit forms [15, 17]. Since the virtual controls are meant to prevent the potential infeasibility caused by the linearization, it is desired that they become negligible in the converged solution, and are therefore penalized with the use of a $\mathcal{L}_s$ norm, with $s = 1, 2$, or ∞, and this penalization term is properly scaled and added to the original cost function of Eq. (1). In a converged solution this term will be effectively negligible, meaning that the obtained states and controls are accurate enough to satisfy the equations of motion, and therefore the solution is dynamically feasible. The other phenomenon to take care when applying this class of methods is the *artificial unboundedness* [27], coming from the use of linearization. The common technique is the adoption of the trust region mechanism, e.g.,

$$\|\mathbf{x} - \mathbf{x}_k\|_s \le \delta \tag{6}$$

which relies on the same concept of $\mathcal{L}_s$ norm previously mentioned, and with the trust-region radius δ can be a constant (e.g., [1, 26]), a dynamic variable penalized and appended to the cost function (e.g., [15, 17]), or a value that is updated between iterations (e.g., [29]). For those path constraints which cannot be represented by a linear, quadratic, or second-order conic form, they are replaced by their respective linearized expressions

$$\nabla_{\mathbf{x}}^T \mathbf{g}(t)\mathbf{x} + \nabla_{\mathbf{u}}^T \mathbf{g}(t)\mathbf{u} \le \nabla_{\mathbf{x}}^T \mathbf{g}(t)\mathbf{x}_k + \nabla_{\mathbf{u}}^T \mathbf{g}(t)\mathbf{u}_k - \mathbf{g}(t_k, \mathbf{x}_k, \mathbf{u}_k) \tag{7}$$

where again the Jacobians $\nabla_{\mathbf{x}}^T \mathbf{g}$ and $\nabla_{\mathbf{u}}^T \mathbf{g}$ are evaluated along $\{\mathbf{x}_k(t), \mathbf{u}_k(t)\}$ thus are explicit functions of t. Finally, if the terminal state constraints are represented by nonlinear equality or nonlinear non-convex inequality constraints, they are also approximated by their linearized or relaxed versions, (although this might lead to the chattering phenomenon [22, 23]). With these treatments the problem in the current iteration is now a convex optimization problem and solved by an appropriate convex optimization method, to obtain the solution $\{\mathbf{x}_{k+1}(t), \mathbf{u}_{k+1}(t)\}$.

This framework, although powerful, can still be insufficient in some situations, as we will see in the next section.

III. Need for Convex-Concave Decomposition

A. Brief Recap on Convex-Concave Decomposition

Let us move from the continuous domain, where optimal control lives, to the discrete space of optimization, and consider the presence of a convex NEC as defined in Ref. [22], in the form of

$$h(\mathbf{x}) = 0 \quad \mathbf{x} \in \mathbb{R}^n \tag{8}$$

Such a constraint could represent a trigonometric identity in the form $u_1^2 + u_2^2 = \cos^2\sigma + \sin^2\sigma = 1$ (e.g., as done in [1]), an invariant property to be satisfied (such as the norm of a quaternion or a unit vector), or could derive from less common transformations, often used in the process of convexifying non-convex problems while retaining their nonlinear properties (e.g., [18]). As highlighted by Lu in [22] there are some situations for which performing linearization of Eq. (8) leads to a *chattering* phenomenon where the sub-solutions merely oscillate back and forth and never converge. A reason for the solution chattering lies with the fact that replacing the NEC with its linearization changes the underlying nature of the problem in the search for the constrained minimum [22]. The main idea behind the CCD is to prevent such behavior by keeping as much as possible the information in the original NEC. Toward this end, the CCD approach replaces the nonlienar constraint of Eq. (8) with a set of three inequalities [22]

$$\begin{aligned} h(\mathbf{x}) &\leq \epsilon \\ -\hat{h}(\mathbf{x}) &\leq 0 \\ -\epsilon &\leq 0 \end{aligned} \tag{9}$$

where the expression $\hat{h}(\mathbf{x})$ is the linearized form of the original constraint in Eq. (8), and the center of linearization is the current iterate $\mathbf{x}_k$:

$$\hat{h}(\mathbf{x}) \triangleq h(\mathbf{x}_k) + \nabla_x h(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) \tag{10}$$

The intuition behind this choice of the slack variable ϵ relies in the non-negativeness of ϵ, which acts as an exact-penalty term when added in the cost function. This allows the algorithm to converge to $\epsilon \longrightarrow 0$ with a finite penalty coefficient on ϵ, whereas a similar argument does not hold for the standard sequential convex approach described in Section II. The first two inequalities represent a decomposition into a convex and a concave relaxation, with the difference that for the first we can retain the nonlinear convex expression, whereas for the concave one we rely upon the linearized expression, represented by the second relationship in Eq. (9) which can be handled effectively as shown in Ref. [26]. This decomposition holds provided that the first inequality, meant as *epigraph* of the function $h(\mathbf{x})$, is convex, in other words that the set

$$\left\{(\mathbf{x}, \epsilon) \in \mathbb{R}^{n+1} | h(\mathbf{x}) - \epsilon \leq 0\right\} \tag{11}$$

is convex. This statement is equivalent to stating the convexity of the function $h(\mathbf{x})$. In this work we want to emphasize the limitations coming from more complex applications of the CCD, and how to improve it to guarantee the feasibility of the corresponding optimization problem.

B. Limitations of the Convex-Concave Decomposition

Let us consider a simple problem, where we are interested in performing a slew maneuver of a spacecraft of 90 deg around its y axis. For the sake of discussion we can imagine the maneuver taking place in two-dimensions, although the argument holds in 3-D as well. In this context, by considering a quaternion to represent the attitude of our spacecraft, we can imagine the rotation being described by the transition from fixed initial and final quaternions $\mathbf{q}_{t_0}$ to $\mathbf{q}_{t_F}$ as

$$\mathbf{q}_{t_0} = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \rightarrow \mathbf{q}_{t_F} = \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}^T \tag{12}$$

with the unit-norm condition

$$\mathbf{q}^T(t_i)\mathbf{q}(t_i) - 1 = 0, \quad i = 0, \ldots, N \tag{13}$$

representing our NEC, discretized in $N + 1$ nodes, and with the quaternions having their scalar part as 4th component. To build our example we implemented the classical 6-DoF dynamics of the spacecraft as given in [15], and we adopted a transcription based on simple finite differences, with the use of virtual controls as described in Eq. (5), and the minimization of the 2-norm of the torque vector that the spacecraft can generate to perform the slew maneuver in a given time interval, e.g., 50 s. It is known that sequential convex algorithms require an initial guess. Even though for the quaternions it is trivial to make a reasonably good initial guess in this problem, it may not be so straightforward in other cases. Therefore, we initialize the algorithm with a poor initial guess, clearly violating the corresponding NECs

$$\mathbf{q}(t_i) = 2\begin{bmatrix} 0 & \cos(\theta_i) & 0 & \sin(\theta_i) \end{bmatrix}^T, \qquad \begin{matrix} i = 0, \dots, N \\ \theta_i = i\frac{\pi}{2N} \end{matrix} \tag{14}$$

When the NECs in Eq. (13) are approximated by their linearized form from the above initial guess, the sequential convex programming algorithm returns a certificate of infeasibility. The reason is due to the fact that the solution that satisfies the linearized NEC constraints cannot satisfy the imposed boundary conditions. The situation is visualized in Fig. 1. The green squares represents the locations where the solution must be at so that the boundary conditions are satisfied. However, the linearization of Eq. (13) will always generate a set (the red dashed lines) that does not contain the points satisfying the original nonlinear constraint as well as the boundary conditions which happen to be on the nonlinear constraint, unless the linearization occurs in a point already satisfying Eq. (13). This observation implies that, in such conditions, the sequential convex algorithm cannot satisfy both the boundary conditions and the linearized equality constraints simultaneously, leading to the infeasibility of the subproblem by design. This is what Fig. (1) shows: the blue crosses are the initial guess, while the red dashed lines are the set allowed by the linearized constraints for each of t_i. Since they are imposed as equality constraints, the new solution is required to stay on the red lines. The green squares represent the points satisfying the boundary conditions. Clearly it is not possible to find a solution having $\mathbf{q}(t_0)$, $\mathbf{q}(t_N)$ being both on the corresponding red dashed lines and on the the green squares, (this only happens if the linearization of the NEC is evaluated at a point satisfying the original NEC). Consequently the optimizer cannot find a feasible solution. It should be pointed out that the situation is the same in this problem if an initial profile $\mathbf{q}(t_i)$ inside the unit circle is used.

What the optimizer will deliver in such a case depends on the type of solver in use. For example, ECOS [30] places the intermediate points on the feasible space represented by the linearized NECs, while placing the extreme points in between the boundary conditions' solution space and the NEC's solution space. Such behavior can be detrimental to the effectiveness of the algorithm, since the quality of the process depends on the sequence of sub-solutions generated in conditions of infeasibility.

What happens when we apply the CCD in this case? Starting from the same initial guess, we can observe the

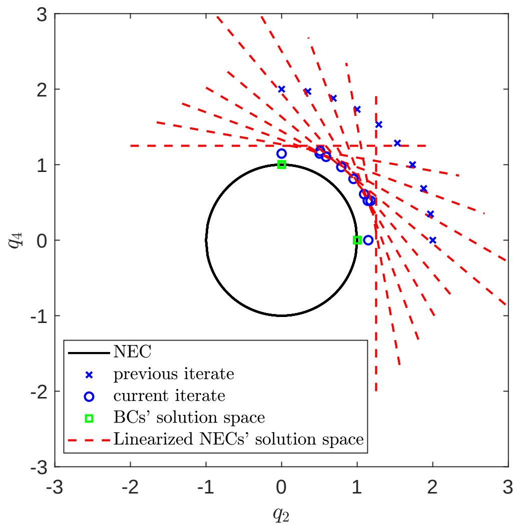
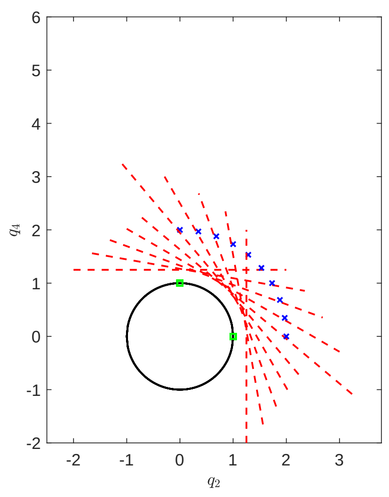


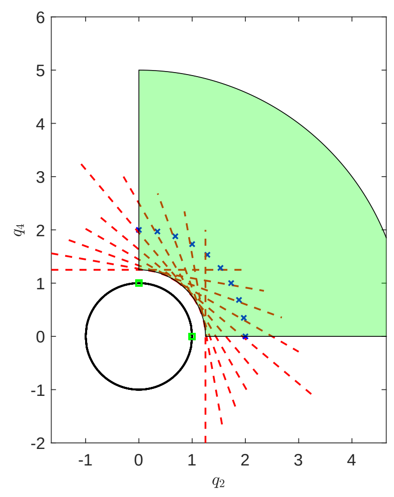
Fig. 1 First iteration with standard sequential convex programming in the presence of a nonlinear equality constraint: the linearized equality constraints do not admit the required boundary conditions.

different components of the CCD construction, and what their feasible space is in Fig. 2. Figure 2(a) shows again the linearized domains as dashed red lines, which identify now the boundary of the half-spaces where the concave inequalities need to be satisfied. This domain is represented explicitly in Fig. 2(b) in green. On the other hand, the convex inequality is represented in red in Fig. 2(c). These feasible spaces are overlapped in Fig. 2(d). The intersection of these domains represents the hyper-volume that must contain the solution. However, it is clear that the boundary conditions are outside of this area. In fact, for such scenario we can see that while the convex one includes now the boundary conditions, the feasibility space associated with the concave inequality built upon the current linearization points, that is, the green area in Fig. 2(b), is physically separated from the green squares representing the boundary conditions, leading to a pretty strong divergence of the algorithm, represented in this case by the blue circles in Fig. 2(d). This behavior is of course not acceptable, and led us to proposing an alternative formulation, which is the subject of the next section.

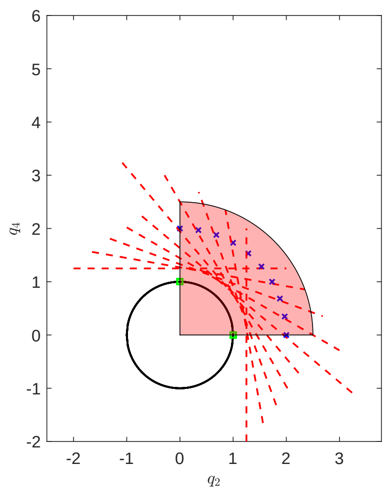
It should be noted that in this example even if the initial guesses are chosen so that the norm of the quaternion $\mathbf{q}(t_i)$ is exactly unity, the same infeasibility issue can still arise. In such a case even though the solution process starts on the NECs, once the solution point moves away from the NECs and outside the unit circle (because of the possibility to move along the linearized constraint), we are back to the situation in Fig. 2(a) and the next solution will be similar to what is in Fig. 2(d). The example illustrated here is therefore meant as magnifying glass to make this type of situations easier to understand.



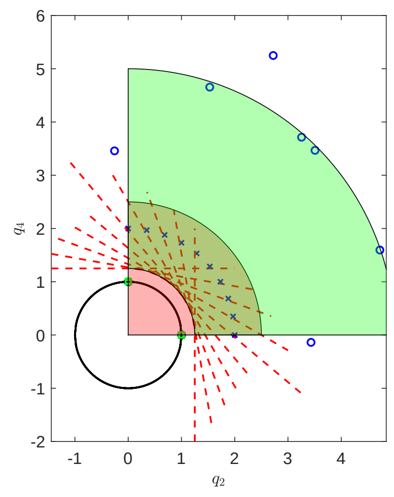
(a) Feasible space associated with boundary conditions



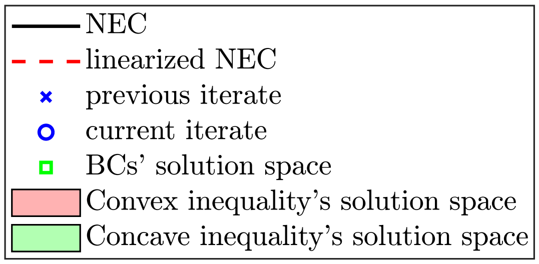
(b) Feasible space associated with concave inequality



(c) Feasible space associated with convex inequality



(d) Intersection of feasible space associated with convex inequality, concave inequality and boundary conditions



NEC
linearized NEC
previous iterate
current iterate
BCs' solution space
Convex inequality's solution space
Concave inequality's solution space

(e) legend

Fig. 2 Analysis of feasible space associated with the use of Convex-Concave Decomposition for the slew maneuver in 2-D: the intersection is empty.

IV. Augmented Convex-Concave Decomposition

A. Formulation of Augmented Convex-Concave Decomposition

To overcome the limitations emphasized in Section III, let us consider to move from the use of a single ϵ to a pair of variables ϵ_1, ϵ_2 for each of the NECs that we have. So, the set of inequalities in [22],

$$\begin{aligned} h(\mathbf{x}) &\leq \epsilon \\ -h(\mathbf{x}_k) - \nabla h(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) &\leq 0 \\ -\epsilon &\leq 0 \end{aligned} \tag{15}$$

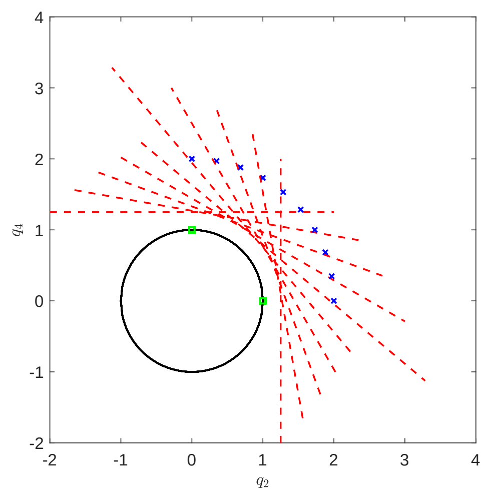
is now expanded to be as follows:

$$\begin{aligned} h(\mathbf{x}) &\leq \epsilon_1 \\ -h(\mathbf{x}_k) - \nabla h(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) &\leq \epsilon_2 \\ -\epsilon_1 &\leq 0 \\ -\epsilon_2 &\leq 0 \end{aligned} \tag{16}$$

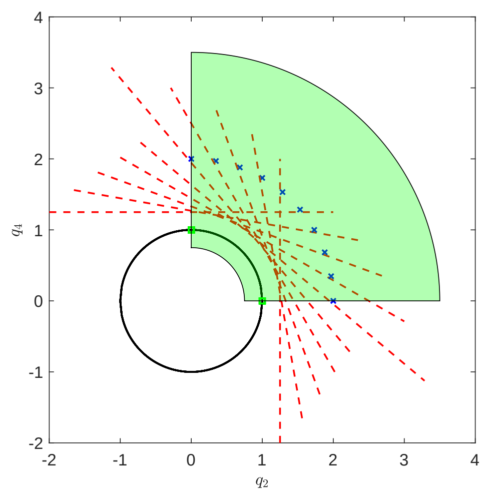
Note that the above formulation is at one collocation point, and an independent pair of $\{\epsilon_1, \epsilon_2\}$ is used at each collocation point. How did the previous example behave in this case? Results are depicted in Fig. 3, where we show how the algorithm behaves in generating the first solution starting from the initial guess provided. The modification to the set of inequalities causes the concave inequality's feasible space to overlap with the convex inequality's feasible space, allowing the algorithm to find a valid solution. In fact, while the linearization construction and the convex feasible set are the same (Fig. 3(a), 3(b)), the use of the new slack variable ϵ_2 allows for an expansion of the concave inequality feasible sets (Fig. 3(c)) such that the intersection of the feasible set of the inequalities and the boundary conditions is always non-empty as depicted in Fig. 3(d).

For the case in Fig. 3 we can see that the algorithm moves several points close to the original NEC, while some others are close to the linearized one. This is the result of the compromise between the dynamics, that pushes for having the points exactly on the NEC, as direct consequence of the quaternion propagation starting from the initial conditions, and the penalty coefficients applied to the concave inequalities, which would instead try to reduce ϵ_2, therefore pushing the solution points on the linearized constraints. This compromise is driven by the penalization weights applied to ϵ_2 and the weights used to penalize the virtual controls, and is automatically resolved along the iteration process.

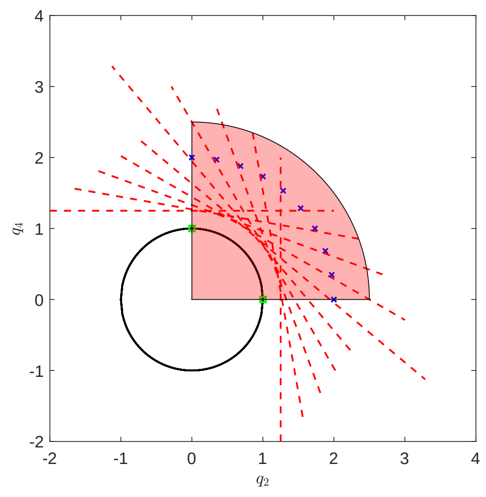
The satisfaction of the boundary constraints is now directly captured by the *stretch* (equal to 1) required in terms of ϵ_2 to find a solution in the points $i = 0$ and $i = N$. ϵ_1 shows non-zero values at the points $i = 1, \ldots, N - 1$ as complementary behavior to ϵ_2, and within few iterations both can be pushed to be close to 0. The two ϵ's have therefore a clear interpretation, leading to a more understandable behavior of the intermediate sub-solutions. A similar example



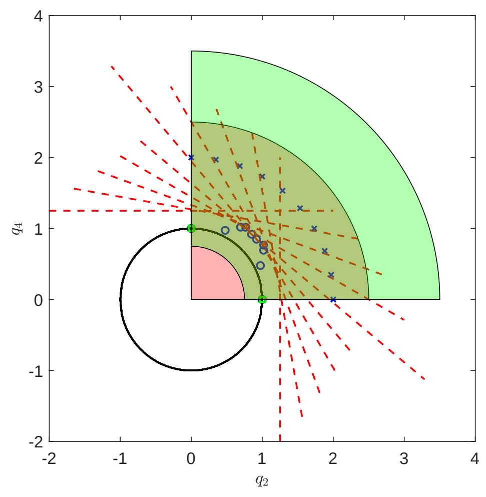
(a) Feasible space associated with boundary conditions



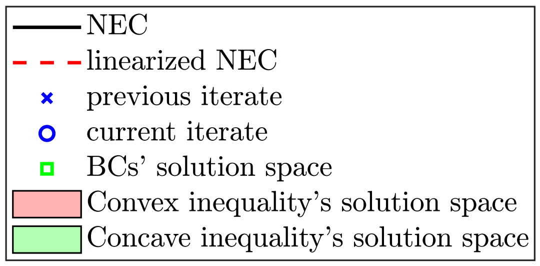
(b) Feasible space associated with concave inequality



(c) Feasible space associated with convex inequality



(d) Intersection of feasible space associated with convex inequality, concave inequality and boundary conditions


NEC
linearized NEC
previous iterate
current iterate
BCs' solution space
Convex inequality's solution space
Concave inequality's solution space

(e) legend

Fig. 3 Analysis of feasible space associated with the use of Augmented Convex-Concave Decomposition for the slew maneuver in 2-D: the intersection is always non-empty.

with an initial guess for the quaternion built within the unit-norm constraint can be found in Section IV of [31].

B. Discrete Decision-Making through ACCD

A further application of the ACCD within the sequential convex programming paradigm is the possibility to conveniently solve a problem where discrete controls or decision-making variables are present. Such a problem typically requires either an algorithm for mixed integer optimization [24], or some work-around such as homotopic paths combined with smooth approximations as done in [25]. Moreover, state-triggered constraints were recently proposed as another tool to model discrete events in continuous fashion[17]. On the other hand, inspired by an observation made in Eq. (7) in [22], we propose and demonstrate an approach in this paper that is particularly suited for the ACCD technique.

Suppose that we have a discrete variable x that can only take discrete values:

$$x \in \{a,\ b\} \tag{17}$$

where a and b are real numbers. To avoid the need for a mixed-integer algorithm, we will treat x as a continuous variable, but subject to following nonlinear equality constraint

$$(x-a)(x-b)=0, \quad x \in \mathbb{R} \tag{18}$$

This quadratic equality constraint is convex when relaxed into an inequality in epigraphic form[22], therefore the ACCD technique applies. Now x can be regarded as one of the continuous variables in the optimization problem and solved together. If the rest of the problem is convex, the NEC of Eq. (18) is then decomposed by

$$\begin{aligned} -\epsilon_1 &\leq 0 \\ -\epsilon_2 &\leq 0 \\ x^2-(a+b)x+ab-\epsilon_1 &\leq 0 \\ (a+b-2x_k)x+x_k^2-ab-\epsilon_2 &\leq 0 \end{aligned} \tag{19}$$

where x_k is the value found at the previous iteration of the sequential convex programming loop. We will use the model of Eq. (19) in Section VI.C where x determines the on and off control of an RCS thruster. In other cases the discrete variable x may represent decision-making action such as the selection of the optimal landing site [32].

We note that it is possible to extend the ACCD to other scenarios where a discrete variable can take multiple values (more than 2). For instance, consider a discrete variable x that can be 0, c, and $2c$ where c is a real number. Define two continuous real variables x_1 and x_2 and the constraints on them

$$\begin{aligned} x_1(x_1 - c) &= 0 \\ x_2(x_2 - c) &= 0 \end{aligned} \tag{20}$$

Let x be defined by

$$x = x_1 + x_2 \tag{21}$$

Then it is clear that the NECs in Eq. (20) are convex, and constraint (21) is linear. Now the problem include x_1, x_2, and x as continuous variables, and the ACCD applies. At the convergence of the solution process x can have only one of the 3 values: 0, c, or $2c$. Similarly, this approach can be extended to any $n+1$ discrete values $0, c, \ldots, nc$, where $n > 2$.

V. Augmented Convex-Concave Decomposition-Based 6-DoF Rocket Landing

In this section we describe the 6-DoF Optimal Rocket Landing Problem and how it can be solved with the use of the ACCD combined with a transcription based on the hp Sequential Pseudospectral Convex Programming.

A. Six-DoF Optimal Rocket Landing Problem

Let us consider the 6-DoF minimum-fuel rocket landing problem in atmosphere with the following cost function to be minimized,

$$\text{minimize } J = \int_{t_0}^{t_F} \|\mathbf{T}(t)\|_2 \, dt \tag{22}$$

where t_0 is the initial time (assumed known, and equal to 0), t_F is the final time, and $\mathbf{T}(t) \in \mathbb{R}^3$ is the thrust vector. The 6-DoF equations of motion are formulated as

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{g} + \frac{\mathbf{T}}{m} + \frac{\mathbf{D}}{m} \\ \dot{\mathbf{q}}^B_{UEN} &= \frac{1}{2}\mathbf{\Omega}_B \cdot \mathbf{q}^B_{UEN} \\ \dot{\omega}_B &= \mathbf{J}^{-1}\left[\mathbf{M}_{TVC} + \mathbf{M}_{RCS} - \omega_B \times (\mathbf{J} \cdot \omega_B)\right] \\ \dot{m} &= -\frac{\|\mathbf{T}\|_2}{I_{sp} g_0} \end{aligned} \tag{23}$$

where $\mathbf{r}(t) \in \mathbb{R}^3$ and $\mathbf{v}(t) \in \mathbb{R}^3$ are the position and the velocity of the center of mass of the rocket expressed with respect to a target-centered Up-East-North reference frame (UEN), $\mathbf{q}^B_{UEN}(t)$ is the quaternion representing the orientation of the body axes with respect to UEN, $\omega_B(t) = \left[\omega_x(t),\ \omega_y(t),\ \omega_z(t)\right]$ is the angular rate vector expressed in body reference frame, and $m(t)$ is the mass of the rocket. Terms dominating the right-hand side of the equations contain the gravity acceleration $\mathbf{g} = [-g,\ 0,\ 0]^T$, whereas $\mathbf{D}$ is the aerodynamic drag force, computed as follows.

$$\mathbf{D} = -\frac{1}{2}\rho\, \|\mathbf{v}\|\, \mathbf{v} S C_D \tag{24}$$

The matrix $\mathbf{\Omega}_B$ is the skew matrix built on the angular rate vector ω_B, defined as in [33].

$$\mathbf{\Omega}_B = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & \omega_y & \omega_z & 0 \end{bmatrix} \tag{25}$$

The angular acceleration is characterized by the Inertia matrix $\mathbf{J}$, and by the total external moments acting on the system, that is, the RCS moment $\mathbf{M}_{RCS}$, and the Thrust-Vector-Control (TVC) moment $\mathbf{M}_{TVC}$. A discussion and modeling on $\mathbf{M}_{RCS}$ are given in the next subsection. The TVC moment $\mathbf{M}_{TVC}$ is modeled by

$$\mathbf{M}_{TVC} = \mathbf{l}_{arm} \times \mathbf{T} \tag{26}$$

with $\mathbf{l}_{arm}$ representing the lever arm between the point of application of the thrust (i.e., the nozzle), and the body center of mass. The description of the dynamics is completed by the specific impulse I_{sp}, the sea-level gravity g_0, the atmospheric density ρ, the surface of the rocket S, and its drag coefficient C_D. In our simplified model we are neglecting the depletion term in the mass rate equation, (in other words, the distinction between vacuum and atmospheric thrust). Moreover, we are assuming for demonstration purposes i) there is no dependency of ρ and C_D on any variable (in other words, they are simply kept constant), ii) the lift generated by the rocket is negligible, iii) the lever arm and the inertia tensor are constant (i.e., no shift due to the fuel consumption). All these simplifications can be removed with some further modeling efforts (e.g., [19]), and do not change the scope of the demonstration here.

We have some boundary conditions to be satisfied, e.g.,

$$\begin{aligned} \mathbf{r}(t_0) &= \mathbf{r}_0 & \mathbf{r}(t_F) &= \mathbf{r}_F \\ \mathbf{v}(t_0) &= \mathbf{v}_0 & \mathbf{v}(t_F) &= \mathbf{v}_F \\ \mathbf{q}_{UEN}^B(t_0) &= \text{free} & \mathbf{q}_{UEN}^B(t_F) &= \mathbf{q}_F \\ \omega(t_0) &= \omega_0 & \omega(t_F) &= \omega_F \\ m(t_0) &= m_0 & m(t_F) &= \text{free} \end{aligned} \tag{27}$$

with the final time t_F to be determined as well. Finally, there are some constraints to be included in the formulation. By retrieving the work of Szmuk et Al. [12], we consider the tilt angle constraint ϕ, defined as the angle between the x-axis of the rocket and the Up direction,

$$\phi = \cos^{-1}\left[2(q_2^2(t) + q_3^2(t)) - 1\right] \leq \phi_{max} \tag{28}$$

the constraint on the glide-slope angle γ along the trajectory of the center of mass of the rocket,

$$\gamma = \tan^{-1}\left[\frac{r_U}{\sqrt{r_E^2 + r_N^2}}\right] \geq \gamma_{min} \tag{29}$$

the maximum TVC deflection δ

$$\delta = \cos^{-1}\left[\frac{T_x}{\|\mathbf{T}\|_2}\right] \leq \delta_{max} \tag{30}$$

and the maximum angular rate constraint.

$$\|\omega(t)\|_2 \leq \omega_{max} \tag{31}$$

These four constraints are formulated or can be recast as second-order cone constraints, and therefore pose no problems within our framework. Moreover, there are bounds on thrust to be considered, including the well-known non-convex lower bound [3], that is

$$T_{min} \leq \|\mathbf{T}(t)\|_2 \leq T_{max} \tag{32}$$

All the (upper and/or lower) bounds in the constraints in Eqs. (28)–(32) are given constants. Finally, we want to explicitly impose the unit-norm constraint on the quaternions for all $t \in [t_0, t_F]$, that is,

$$[\mathbf{q}_{UEN}^B(t)]^T \cdot \mathbf{q}_{UEN}^B(t) - 1 = 0 \tag{33}$$

While the evolution of the quaternions governed by the continuous differential equations in Eq. (23) preserves the unit norm if $\|\mathbf{q}_{UEN}^B(t_0)\| = 1$, the explicit enforcement of Eq. (33) at the collocation points will serve to ensure that the feasibility of the solution is not compromised by discretization errors once the problem is transcribed into a finite-dimensional form.

With these definitions our continuous Optimal Control Problem is complete: we want to minimize Eq. (22) subject to Eqs. (23)-(33). In Sec. V.B we will move from the continuous OCP perspective to the finite-dimensional one by introducing the corresponding transcription, based on the combination of the ACCD and the hp Sequential Pseudospectral Convex Programming.

B. Inclusion of RCS for Full 6-DoF Guidance

There is a caveat in the rocket benchmark as formulated in [12]: the RCS moment not included (or $\mathbf{M}_{RCS} \equiv 0$). Therefore there is no roll control/stabilization loop. This is due to the physics of the problem, since the roll cannot be controlled by the TVC. In Ref. [12] the neutrally stable roll channel is assumed to remain at zero roll rate and angle throughout the entire trajectory, rendering the problem de-facto 5-DoF guidance. If the initial roll angle and roll rate are not the same as the desired final values, as most likely the case in practice, it is necessary to include RCS control in the problem for roll stabilization and control, as we will do in the following.

Given the discrete (on-off) nature of the RCS thrusters their control assumes values equal to 0 (closed valve) or 1 (open valve). We include in the transcription two RCS ($RCS1$ and $RCS2$), oriented in opposite direction. The term $\mathbf{M}_{RCS}$ in Eq. (23) will be modified as follows.

$$\mathbf{M}_{RCS} = \mathbf{l}_{arm}^{RCS} \times \mathbf{T}_{max}^{RCS} thr^{RCS1} - \mathbf{l}_{arm}^{RCS} \times \mathbf{T}_{max}^{RCS} thr^{RCS2} \tag{34}$$

The terms $\mathbf{l}_{arm}^{RCS}$ and $\mathbf{T}_{max}^{RCS}$ represent the distance of the RCS thrusters'direction from the longitudinal axis of the rocket and the maximum thrust they can exert, whereas thr^{RCS1} and thr^{RCS2} are the throttle opening, which can only be equal to 0 or 1. They are opposite in sign, since we want to be able to produce roll torque in both direction. Equivalently, they will imply a certain consumption of fuel to be taken into account, so the last relationship in Eq. (23) becomes

$$\dot{m} = -\frac{\|\mathbf{T}\|_2}{I_{sp} g_0} - \frac{\mathbf{T}_{max}^{RCS}\left(thr^{RCS1} + thr^{RCS2}\right)}{I_{sp}^{RCS} g_0} \tag{35}$$

Note that in this model, if the RCS is on, only one of $RCS1$ and $RCS2$ will work at any given time, since their combined effect would not generate any torque if both are on. This is a modeling simplification. In reality roll torques with a given sign are always generated by pairs of thrusters. Therefore, each of the signals $RCS1$ and $RCS2$ would in practice be associated with a given pair, so that the RCS only generates roll torque but not non-zero total side force.

Our new variables to be modeled through ACCD are the throttle values thr^{RCS1} and thr^{RCS2}. Their relaxation is simply given by Eq. (19), with $x = thr^{RCS1}, thr^{RCS2}$, $a = 0$, $b = 1$.

C. Transcription through ACCD-based Sequential Pseudospectral Convex Programming

We can now transcribe our Optimal Control Problem. The transcription will partially leverage the Sequential Pseudospectral Convex method proposed in [15] with some differences that will be highlighted wherever needed. The reasons behind the use of hp Sequential Pseudospectral methods reside in the good compromise between efficiency and accuracy, which can easily be modulated with a proper choice of the elements h and p. Although not done here, the strategy can be further improved to include mesh-refinement schemes (e.g., [8]). As first step, we introduce the pseudospectral discrete domain, made by $i = 1, \ldots, n$ segments, each with p_i collocated nodes and $p_i + 1$ discretized nodes, since the flipped Legendre-Gauss-Radau transcription is adopted [34]. The domain construction follows the scheme of [15] with the main difference that each segment can have its independent number of collocated nodes, meaning that p_i can change for each of the segments. In total we have $\sum_{i=1}^{n} p_i$ collocated nodes. The domain is represented in Fig. 4. Each red square represents a discrete, non-collocated node. The segments containing the collocation are then linked together by ensuring state continuity through the *Linking conditions*, which are state equality constraints simply enforcing that the values at the end of a segment must be equal to the ones at the beginning of the following segment. The conversion of each discrete pseudospectral timestep at the node j of the segment i $\tau^{i,j} \in [-1,\ 1]$ to its corresponding physical time value $t^{i,j} \in [t_0,\ t_F]$ can be performed by using the following expression

$$t^{i,j} = t_0 + \frac{t_F - t_0}{n}(i - \frac{1}{2}) + \frac{t_F - t_0}{2n}\tau^{i,j}, \qquad \begin{array}{c} i = 1, \ldots, n \\ j = 1, \ldots, p_i \end{array} \tag{36}$$

Before transcribing the dynamics we redefine our controls from the set $\mathbf{T} = \begin{bmatrix} T_x, & T_y, & T_z \end{bmatrix}^T \in \mathbb{R}^3$ to the 4-dimensional set $\begin{bmatrix} \mathbf{u}, & T_{mag} \end{bmatrix}^T \in \mathbb{R}^4$, with the vector $\mathbf{u} = \begin{bmatrix} u_x, & u_y, & u_z \end{bmatrix}^T$ representing the thrust direction unit vector, therefore obeying our first NECs:

$$(\mathbf{u}^{i,j})^T(\mathbf{u}^{i,j}) - 1 = 0, \qquad \begin{array}{c} i = 1, \ldots, n \\ j = 1, \ldots, p_i \end{array} \tag{37}$$

By the ACCD approach, these NECs are cast in the algorithm by

$$\begin{array}{c} (\mathbf{u}^{i,j})^T(\mathbf{u}^{i,j}) - 1 - \epsilon_1 \leq 0 \\ -2(\mathbf{u}_k^{i,j})^T(\mathbf{u}^{i,j}) + (\mathbf{u}_k^{i,j})^T(\mathbf{u}_k^{i,j}) + 1 - \epsilon_2 \leq 0 \\ -\epsilon_1 \leq 0 \\ -\epsilon_2 \leq 0 \end{array}, \qquad \begin{array}{c} i = 1, \ldots, n \\ j = 1, \ldots, p_i \end{array} \tag{38}$$

Consequently, the bounds (including the lower non-convex one) on the thrust vector reduce to a set of 2 linear inequalities per each node.

$$T_{min} \leq T_{mag}^{i,j} \leq T_{max}, \qquad \begin{array}{c} i = 1, \ldots, n \\ j = 1, \ldots, p_i \end{array} \tag{39}$$

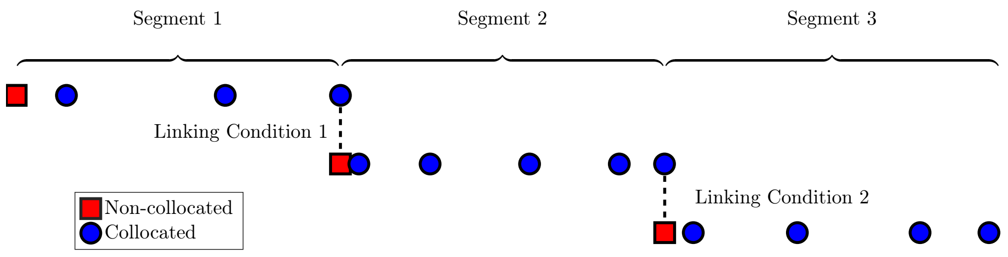


Fig. 4 Example of hp pseudospectral domain with three segments having a different number of collocation points per segment.

The second set of NECs is simply the enforcement of Eq. (33) at each of the discretized nodes:

$$[\mathbf{q}_{UEN}^{B,i,j}]^T \cdot \mathbf{q}_{UEN}^{B,i,j} - 1 = 0, \qquad \begin{array}{c} i = 1, \dots, n \\ j = 0, \dots, p_i \end{array} \tag{40}$$

Following the ACCD method Eq. (40) is represented in the algorithm by

$$\begin{array}{c} (\mathbf{q}_{UEN}^{B,i,j})^T (\mathbf{q}_{UEN}^{B,i,j}) - 1 - \epsilon_3 \leq 0 \\ -2(\mathbf{q}_{UEN,k}^{B,i,j})^T (\mathbf{q}_{UEN}^{B,i,j}) + (\mathbf{q}_{UEN,k}^{B,i,j})^T (\mathbf{q}_{UEN,k}^{B,i,j}) + 1 - \epsilon_4 \leq 0 \\ -\epsilon_3 \leq 0 \\ -\epsilon_4 \leq 0 \end{array}, \qquad \begin{array}{c} i = 1, \dots, n \\ j = 1, \dots, p_i \end{array} \tag{41}$$

For the cases where a non-zero roll rate is considered, we formulate two more ACCDs for each of the discrete points to include the RCS terms through the use of Eq. (19), thus completing the handling of the nonlinear equality constraints and the discrete decision making required for the 6-DoF landing problem.

For the dynamics, we apply the collocation scheme based on flipped Radau pseudospectral method, which implies that the differential equations in (23) can be rewritten as

$$\dot{\mathbf{x}} = \frac{t_F - t_0}{2n} \left[\mathbf{f}(t, \mathbf{x}, \mathbf{u}) + \mathbf{C}\nu \right] \triangleq \tilde{\mathbf{f}}(t, \mathbf{x}, \mathbf{u}, \nu, t_F) \tag{42}$$

where $\mathbf{f}(t, \mathbf{x}, \mathbf{u})$ is the right-hand side of the differential equations of our problem, $\nu \in \mathbb{R}^{n_\nu}$ is the vector of virtual controls, and $\mathbf{C} \in \mathbb{R}^{n_x \times n_\nu}$ an *allocation matrix* which maps the virtual controls onto the differential equations, depending on which entries are equal to 0 or 1. For the case proposed here,

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_6 & \mathbf{O}_{6\times 8} \end{bmatrix}^T \tag{43}$$

meaning that we adopt synthetic translational velocities and accelerations, whereas the mass and the attitude differential equations are not directly affected by virtual controls. However, this is potentially not the only choice. The virtual controls are pointwise bounded by a slack variable $\mu^{i,j}$ as

$$\left\| \begin{matrix} \nu_{i,j,1} \\ \vdots \\ \nu_{i,j,n_\nu} \end{matrix} \right\|_2 \leq \mu^{i,j}, \qquad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p_i \end{matrix} \tag{44}$$

with n_ν representing the number of virtual controls, (in this case equal to 6), and with $\mu^{i,j}$ embedding the norm of the entire virtual control vector in a single scalar. Pushing to 0 such slack variable will conversely shrink the entire set of virtual controls, consequently ensuring the dynamic feasibility of the solution. By defining

$$\mathbf{A} \triangleq \nabla_{\mathbf{x}}^T \tilde{\mathbf{f}}_k, \quad \mathbf{B} \triangleq \nabla_{\mathbf{u}}^T \tilde{\mathbf{f}}_k, \quad \mathbf{E} \triangleq \nabla_{t_F}^T \tilde{\mathbf{f}}_k, \quad \mathbf{G} \triangleq \tilde{\mathbf{f}}_k - \mathbf{A}\mathbf{x}_k - \mathbf{B}\mathbf{u}_k - \mathbf{C}\nu_k - \mathbf{E}t_{F,k} \tag{45}$$

where the set $[\ \mathbf{x}_k, \mathbf{u}_k, \nu_k, t_{F,k}\ ]$ is the solution obtained at the previous iterate and $\tilde{\mathbf{f}}_k \triangleq \tilde{\mathbf{f}}(t_k, \mathbf{x}_k, \mathbf{u}_k, \nu_k, t_{F,k})$. By considering the discrete differential matrix $\mathbf{D}$ in the corresponding augmented representation [15], we get a final system of equality constraints in the form

$$\mathbf{D}\mathbf{x} - \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{u} - \mathbf{C}\nu - \mathbf{E}t_F = \mathbf{G} \tag{46}$$

The boundary conditions are simply imposed as linear equalities in the form

$$\mathbf{x}^{1,0} = \mathbf{x}_{t_0}, \quad \mathbf{x}^{n,p_n} = \mathbf{x}_{t_F} \tag{47}$$

where Eq. (47) hold only for the corresponding elements specified in Eq. (27). Finally, the constraints of Eqs. (28)-(31) are evaluated in each collocated node. For what regards the cost function, we can augment the one of Eq. (22), to get

$$J_a = J + w_{fLGR}^T \left[w_\mu \mu + w_p \left(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 \right) \right] \tag{48}$$

where μ and ϵ_k are here meant with a little abuse of notation as vectors stacking the entire set of collocated variables $\mu^{i,j}$, $\epsilon_k^{i,j}$, $k = 1, \ldots, 4$, respectively. Equivalently, the vector w_{fLGR}^T is an expanded vector containing the quadrature weights associated with the flipped Legendre-Gauss-Radau method of order p, and obtained as

$$w_{fLGR} = \left[\left(w_{fLGR}^{1,p_1} \right)^T, \ \ldots \left(w_{fLGR}^{n,p_n} \right)^T \right]^T \tag{49}$$

The weights w_μ and w_p measure the relative importance of the virtual slack variable μ and the exact penalty functions $\epsilon_1, \dots, \epsilon_4$with respect to the true cost J, meant as multiplied by a unitary weight. The variables $\epsilon_{1,2}$ and $\epsilon_{3,4}$ are associated with the concave and convex parts of the thrust unit vector, and the concave and convex parts of the quaternion norm constraints, respectively. The solution process iteratively minimizes Eq. (48) while satisfying Eqs. (38), (39), (41), (44), (46), and (47), in addition to Eqs. (28)-(31) evaluated at each collocated node. The process is terminated at the iteration $k + 1$ if a given threshold for each group of variables is achieved, completing the transcription.

$$\left.\begin{aligned}
||\Delta \mathbf{r}||_\infty &\triangleq \max_{i,j} \left\| \mathbf{r}_{k+1}^{i,j} - \mathbf{r}_k^{i,j} \right\|_2 \leq \delta_{\mathbf{r}} \ \& \\
||\Delta \mathbf{v}||_\infty &\triangleq \max_{i,j} \left\| \mathbf{v}_{k+1}^{i,j} - \mathbf{v}_k^{i,j} \right\|_2 \leq \delta_{\mathbf{v}} \ \& \\
||\Delta m||_\infty &\triangleq \max_{i,j} \left| m_{k+1}^{i,j} - m_k^{i,j} \right| \leq \delta_m \ \& \\
||\delta \mathbf{q}||_\infty &\triangleq \max_{i,j} \left\| \mathbf{q}_{k+1}^{i,j} - \mathbf{q}_k^{i,j} \right\|_2 \leq \delta_{\mathbf{q}} \ \& \\
||\Delta \omega||_\infty &\triangleq \max_{i,j} \left\| \omega_{k+1}^{i,j} - \omega_k^{i,j} \right\|_2 \leq \delta_\omega
\end{aligned}\right. \implies \texttt{convergence achieved} \tag{50}$$

For the results shown in this paper, we adopted $\delta_{\mathbf{r}} = 0.01$ LU, $\delta_{\mathbf{v}} = 0.01$ LU/TU, $\delta_m = 0.1$ MU, $\delta_{\mathbf{q}} = 0.001$, and $\delta_\omega = 0.25$ DEG/TU, where LU, TU and MU represent the non-dimensional length, time, and mass unit, respectively, consistently with the original benchmark scenario definition [12].

VI. Numerical Results

A. Rocket Benchmark

The data on the rocket benchmark used in this work is essentially taken from the work of Szmuk and Acikmese [12]. The only differences are the values of I_{sp} and g_0, not specified in the original source, and the drag-related terms, that is, ρ, S, and C_D. All the parameters are given in Table 1. In terms of hp all the results have been generated with 5 segments, containing 10 collocated points each, (i.e., $n = 5$, $p_1 = \cdots = p_n = 10$). In this work we focus on fuel-optimal results, while we omitted for space reasons the time-optimal results, which can be found in our previous work ([31]), as well as in the original source ([12]). The set of initial conditions are described in Table 2. The corresponding results are shown in the next sections. The convex solver in use for all the cases is the open-source software ECOS[30].

B. Minimum-Fuel Trajectory with Zero-Roll Rate

The fuel-optimal landing problem is solved and the corresponding results are given in Figs. 5 through 9.

Figure 5 shows the body axes of the rocket (in red-green-blue convention, representing the x_b, y_b, and z_b axes, respectively), together with the 3-dimensional corresponding trajectory of the rocket in the UEN frame. The rocket is initially flying mostly towards West, and the algorithm performs a maneuver of about 90 deg to ensure a correct

Table 1 Rocket Benchmark Parameters

Parameter	Value [Unit]	Parameter	Value [Unit]
gravity acceleration g	1 [LU/TU^2]	specific Impulse I_{sp}	294.2 [TU]
wet mass m_{wet}	2 [MU]	sea-level gravity g_0	1 [LU/TU^2]
dry mass m_{dry}	1 [MU]	inertia matrix $\mathbf{J}$	$0.01 \cdot \mathbf{I}_{3\times3}$ [MU·LU^2]
drag coefficient C_D	0.1	lever arm $\mathbf{l}_{arm}$	$[-0.01,\ 0,\ 0]^T$ [LU]
atmospheric density ρ	1 [MU/LU^3]	minimum glideslope angle γ_{min}	20 [DEG]
reference Surface S	0.5 [LU^2]	maximum tilt angle ϕ_{max}	90 [DEG]
lower Bound on Thrust T_{min}	1 [MU·LU/TU^2]	maximum angular rate ω_{max}	60 [DEG/TU]
upper Bound on Thrust T_{max}	5 [MU·LU/TU^2]	maximum gimbal angle δ_{max}	20 [DEG]

Table 2 Boundary Conditions

Parameter	Value [Unit]
Initial position $\mathbf{r}_0$	$[4,\ 4,\ 0.5]^T$ [LU]
Initial velocity $\mathbf{v}_0$	$[0,\ -4,\ 0]^T$ [LU/TU]
Initial angular rate ω_0	$[0,\ 0,\ 0]^T$ [1/TU]
Initial mass m_0	2 [MU]
Final position $\mathbf{r}_F$	$[0,\ 0,\ 0]^T$ [LU]
Final velocity $\mathbf{v}_F$	$[0,\ 0,\ 0]^T$ [LU/TU]
Final quaternion $\mathbf{q}^B_{UEN,F}$	$[0,\ 0,\ 0,\ 1]^T$
Final angular rate ω_F	$[0,\ 0,\ 0]^T$ [1/TU]

pinpoint landing. Given the initial conditions, and as visible in Fig. 6 there is an out-of-plane component, along the North direction. The algorithm compensates for it along the trajectory, and correctly guides the rocket towards the prescribed target position.

The NECs imposed in the problem is the norm of the thrust direction unit vector, visible in Fig. 7(a), with the largest violation in this case being in the order of $3 \cdot 10^{-5}$. The thrust profile follows the classical *max-min-max* structure, and we obtain a clean solution with the two switches well defined. The quaternion profile in Fig. 7(b) shows in the top plot on one side again the 90-deg maneuver captured by the elements q_2 and q_3. The quaternion norm (Fig. 7(b)) is well satisfied, with a maximum violation in the order of $3 \cdot 10^{-7}$. The higher accuracy is probably due to a simpler control structure than for the time-optimal case of [31], despite the presence of out-of-plane components. Finally, all the constraints (Fig. 8) are satisfied too (with the only exception coming from the last point, associated with zero altitude, and therefore returning a glideslope angle equal to 0). Note that, differently from the time-optimal case of [31] no sharp maneuvers are needed for the fuel-optimal case. It follows that neither the tilt angle, nor the gimbal angle, nor the angular rate are saturated, or at least not as much as in the minimum-time formulation. The generally less dynamic maneuver seems also a factor helping the higher accuracy of the quaternion norm achieved in this case. In the last plot in Fig. 9 we can observe that 1) the (in some sense) easier maneuver allows for a quicker convergence (only 6 iteration against the 16 needed for the time-optimal case of [31]), and 2) also in this case the augmented cost function converges to the original cost of the problem (meaning that the slack variables are essentially zero). Moreover, the initial discrepancy between the two cost profiles is smaller than the time-optimal one, meaning that the algorithm requires a smaller use of virtual controls on one side, and penalty slacks on the other. Finally, we show the convergence of the indivudual errors in Fig. 10, where the evolution of the groups defined in Eq. (50) is plotted on a semi-logarithmic scale along the iterative process. It is well visible as all the states consistently converge.

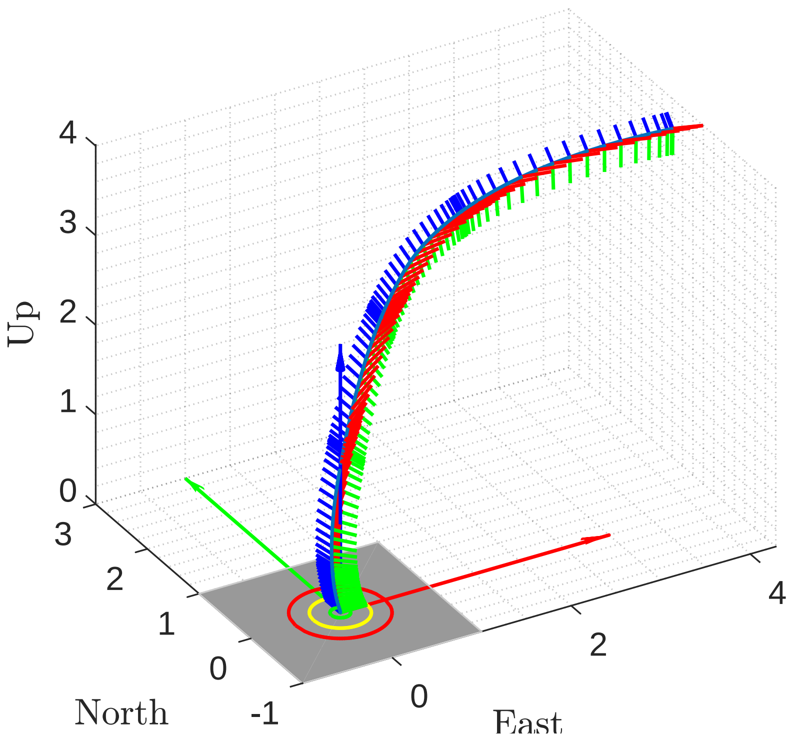


Fig. 5 Minimum-fuel problem: body axes and resulting trajectory.

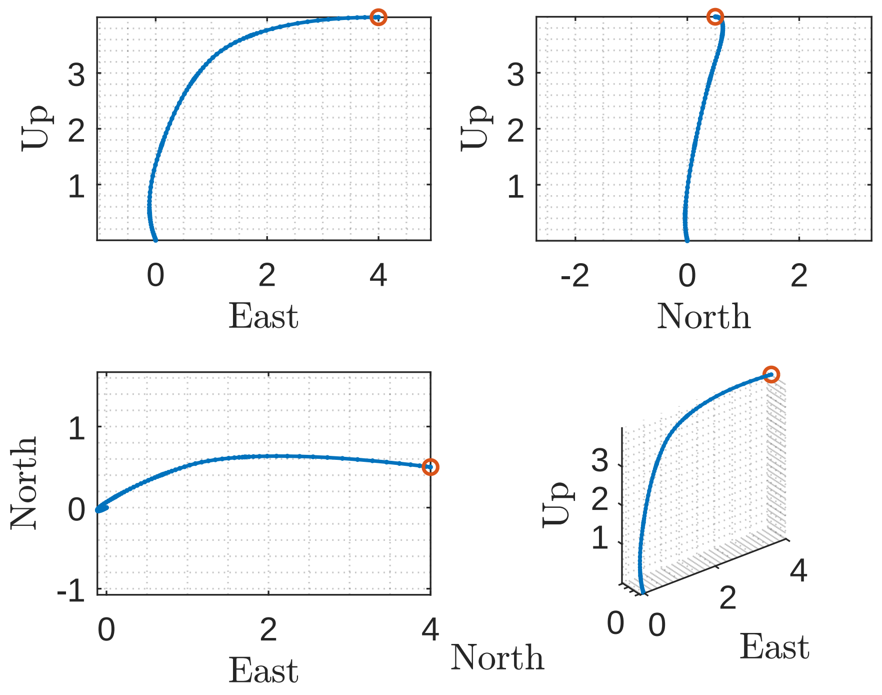
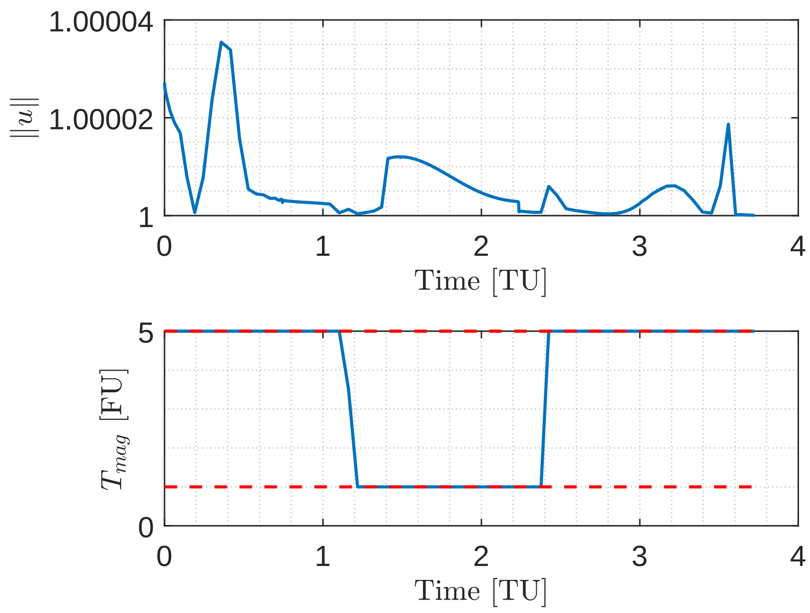
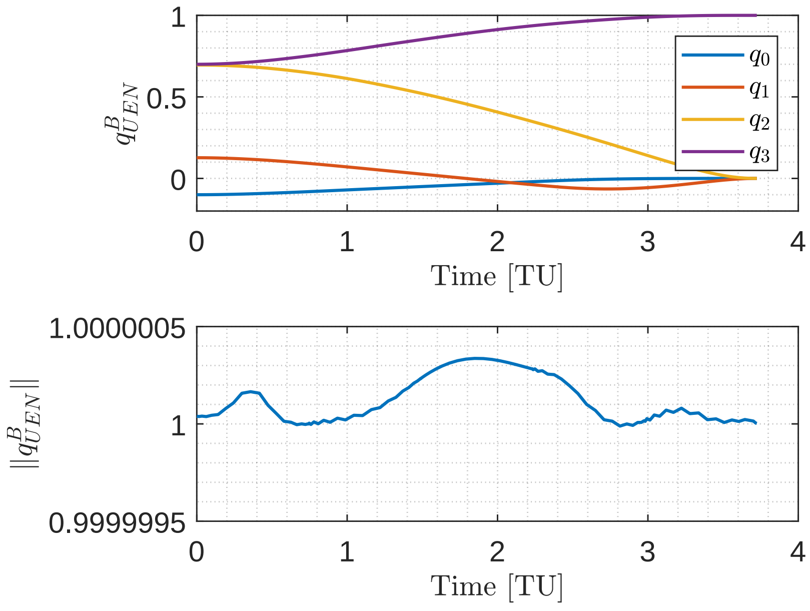


Fig. 6 Minimum-fuel problem: in-plane projections.



(a) Thrust unit vector norm and magnitude



(b) Quaternion components and norm

Fig. 7 Minimum-fuel problem: thrust and quaternion constraints and structure.

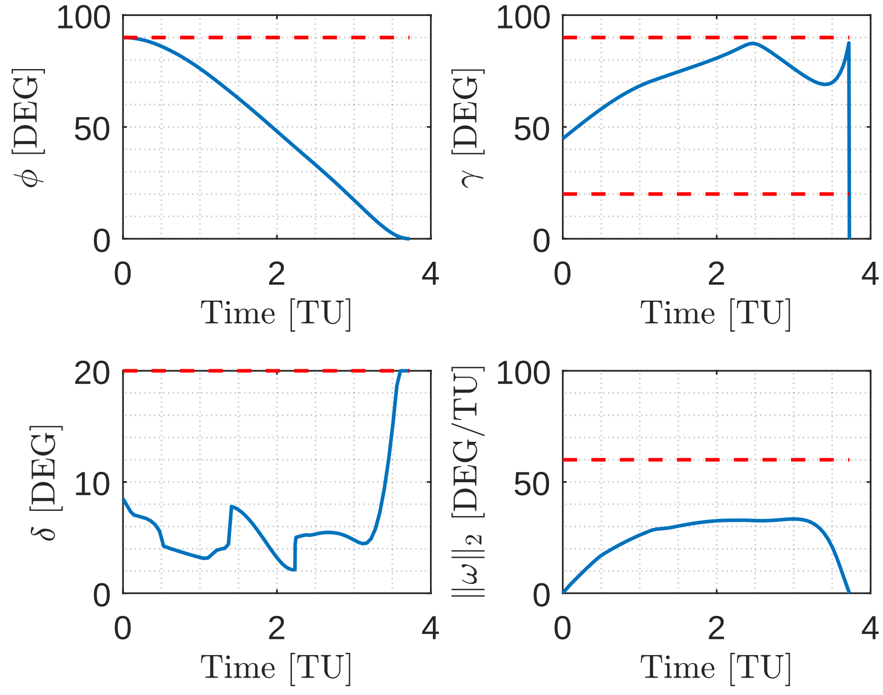


Fig. 8 Minimum-fuel Problem: constraints.

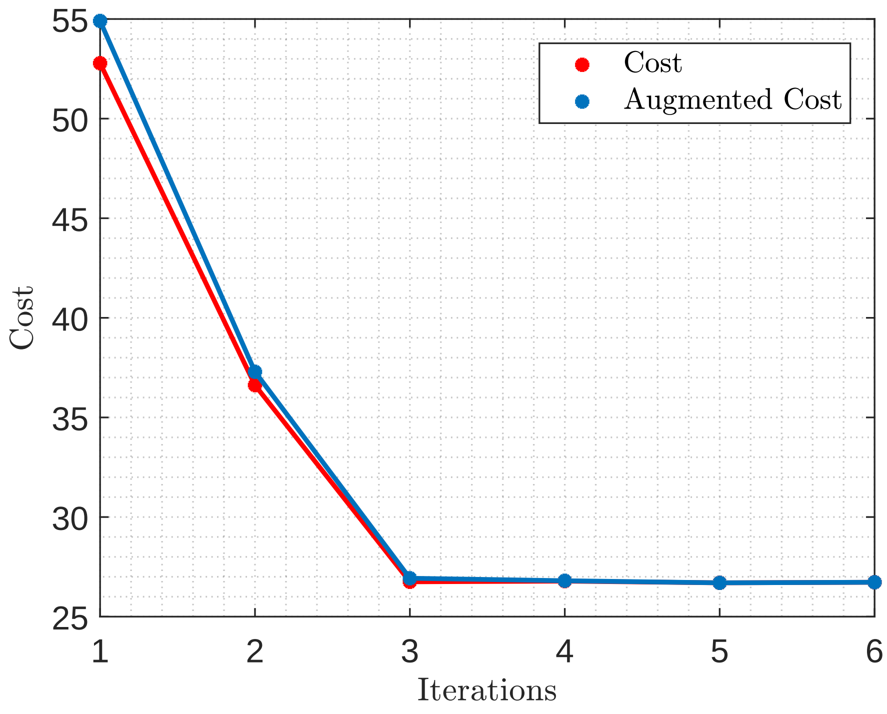


Fig. 9 Minimum-fuel problem: cost functions.

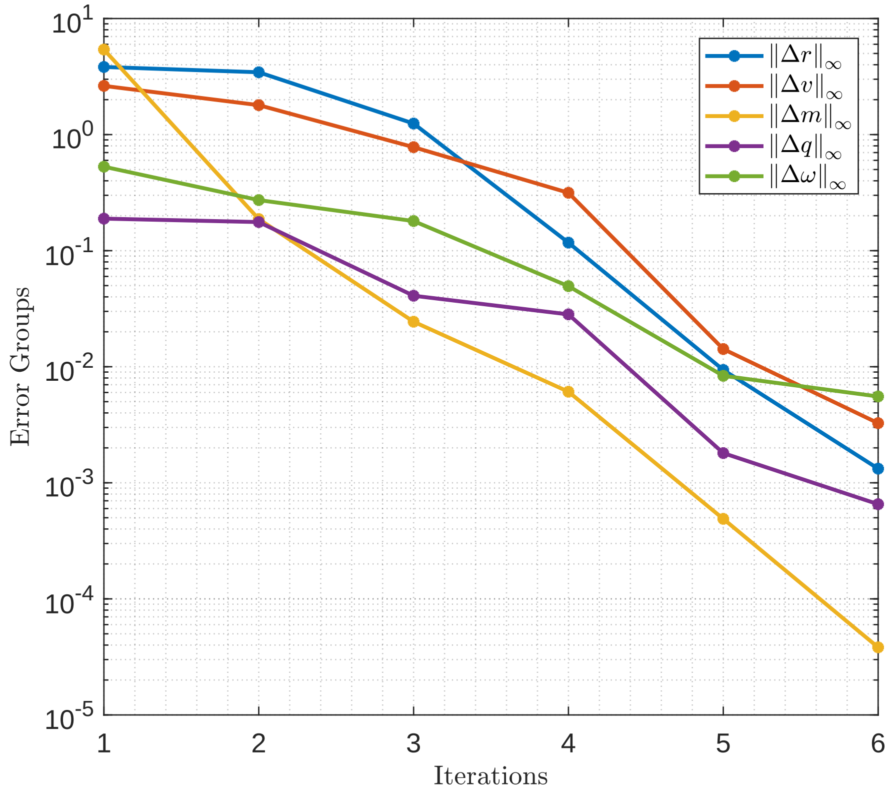


Fig. 10 Minimum-fuel problem: evolution of individual state group errors.

C. Minimum-Fuel Solution with Non-Zero Roll Rate

By considering the same minimum-fuel scenario described by Table 2, the only modification we propose here is given by an initial angular rate equal to $\omega(t_0)$ = 30 DEG/TU. This value is beyond the reasonable bounds coming from safety concerns, but has been selected for demonstration certain purposes. The numerical values adopted for modeling the RCS are $\mathbf{T}_{max}^{RCS}$ = 0.2 MU·LU/TU^2, $\mathbf{l}_{arm}^{RCS}$ = [0, 0.004, 0] LU, and I_{sp}^{RCS} = 200 TU. The obtained results are shown in Figs. 11-14. The trajectory is depicted in Fig. 11. Although the general shape is the same, we can now see from the orientation changes of the body frame that there is effectively a rotation around the x_b-axis of the rocket. The RCS acts to linearly reduce the roll rate from the initial value to 0 DEG/TU. Also in this case the structure of the thrust magnitude is correctly reconstructed, and the NECs previously introduced satisfied (Fig. 12). Moreover, the solution shows that the second RCS thruster (the one able to generate negative roll torque) is activated until the angular rate becomes 0 DEG/TU, as expected. Although not shown here, identical results were obtained with a pseudospectral general-purpose trajectory optimization tool ([35]) by directly enforcing Eq. (18). Finally, no big differences are visible with respect to the convergence process. The augmented cost converges to the original cost, and the process is completed in 7 iterations (Fig. 14). The error groups follow a trend similar to the one depicted in Fig. 10, and have been omitted to avoid repetitions.

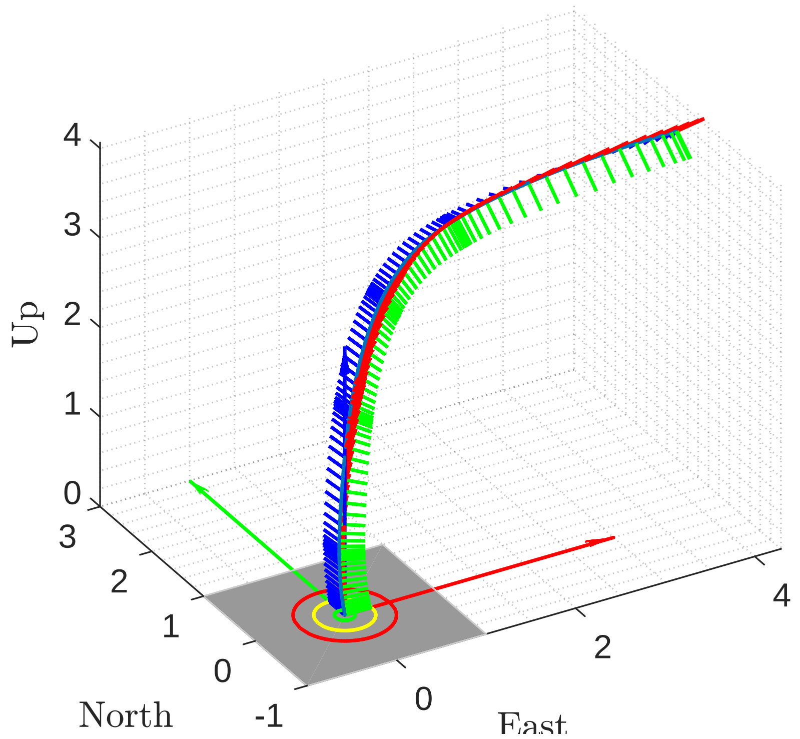
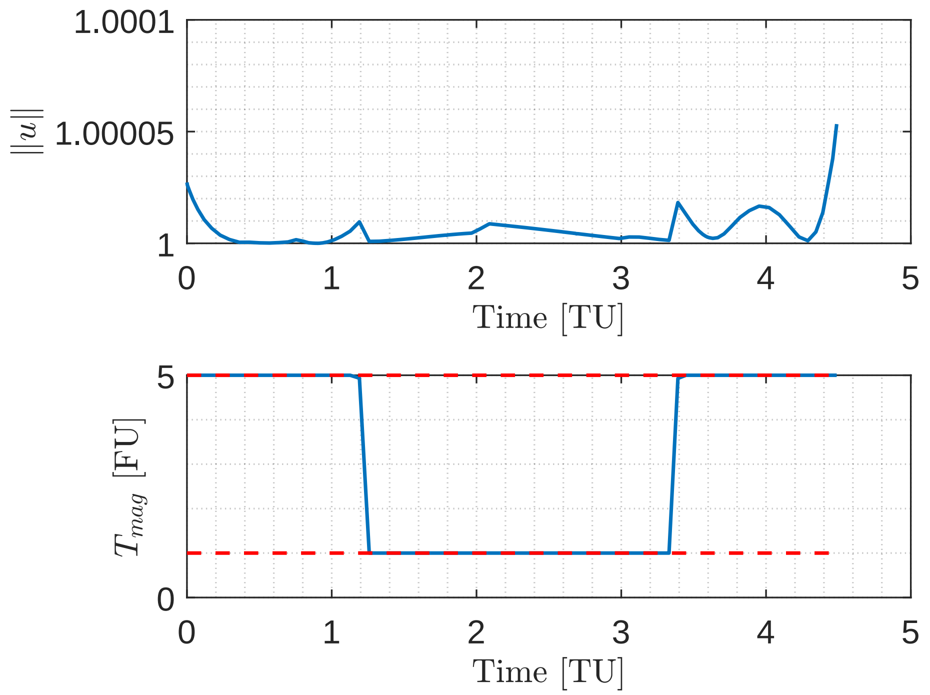
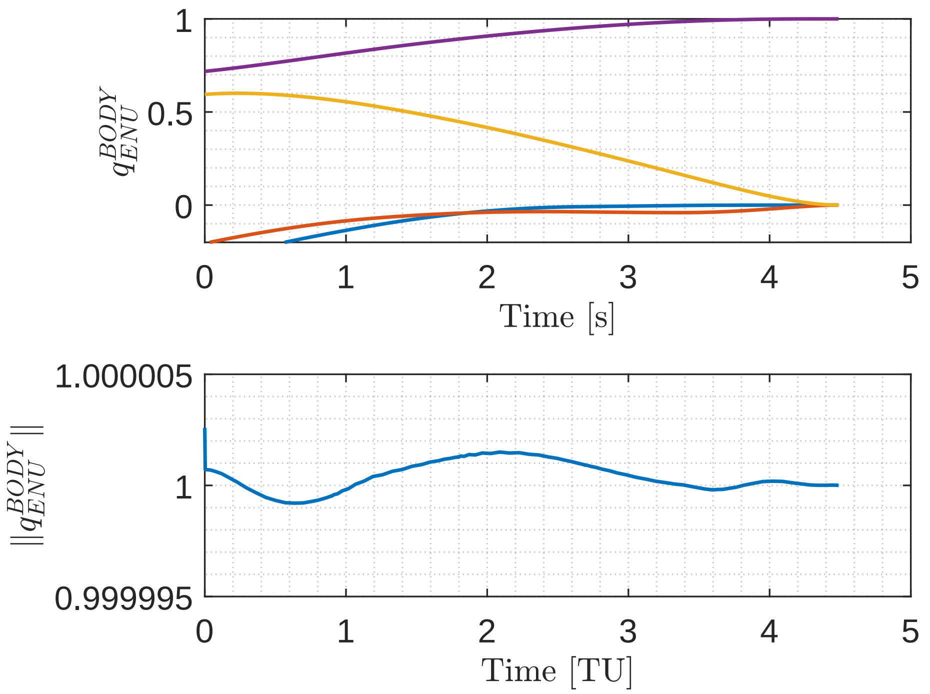


Fig. 11 Minimum-fuel problem in the presence of non-zero roll rate: body axes and resulting trajectory.



(a) Thrust unit vector norm and magnitude



(b) Quaternion components and norms

Fig. 12 Minimum-fuel problem in the presence of non-zero roll rate: thrust and quaternion constraints and structure.

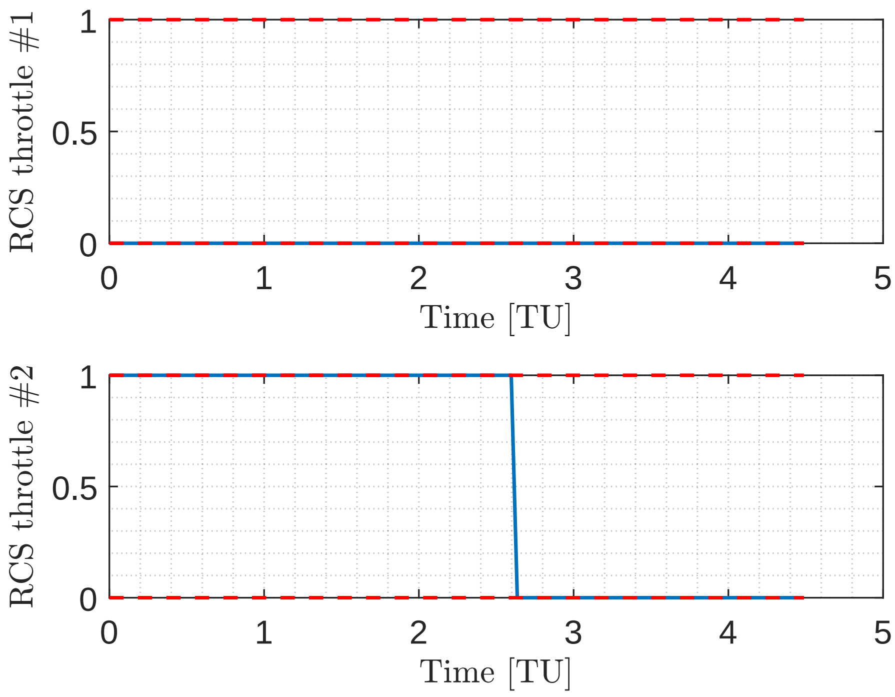


Fig. 13 Minimum-fuel problem in the presence of non-zero roll rate: RCS throttle values.

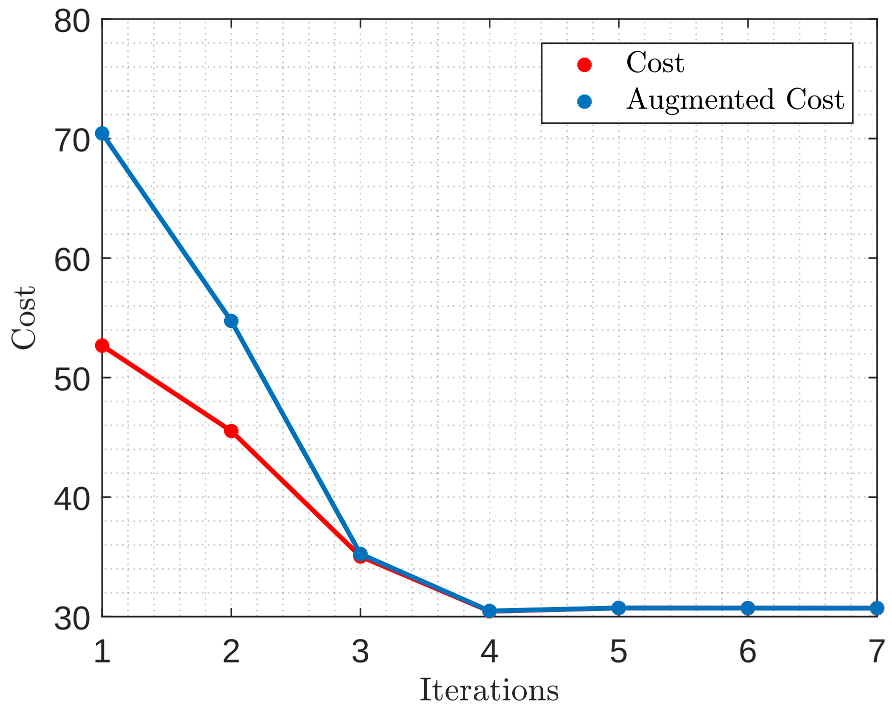


Fig. 14 Minimum-fuel problem in the presence of non-zero roll rate: cost functions.

D. Regularizing the Full 6-DoF Guidance Problem

During the execution of the Monte-Carlo campaign (Sec. VI.G) we noticed that the activity of the RCS is sometimes arbitrarily split into multiple pulses. By inspecting the formulation of the problem it is observed that effectively only the total roll torque to be given to compensate for a certain initial angular rate is determined by the solution, but how this torque is distributed during the time of the mission is arbitrary, as two different distributions of RCS torque pulses can lead to the same fuel consumption provided that the total width of the pulse is the same, with minimal differences on the rest of the trajectory (i.e., spending the RCS fuel earlier or later would imply having a slightly lighter or heavier rocket, but the difference is in practice negligible). However, having more RCS pulses than the minimum number complicates the flight operation unnecessarily. To prevent the solution from using unnecessary number of RCS pulses, we include a regularization term ΔJ in the augmented cost function of Eq. (48), defined as

$$\Delta J = k_{reg} \int_{t_0}^{t_F} |\omega_x(t)| \, dt \tag{51}$$

which, in discrete form becomes

$$\Delta J = k_{reg} w_{fLGR}^T s_{reg} \tag{52}$$

with s_{reg} representing a regularization slack variable subject to the following conic constraint

$$\left|\omega_x^{i,j}\right| \leq s_{reg}^{i,j}, \qquad \begin{array}{l} i = 1, \ldots, n \\ j = 1, \ldots, p_i \end{array} \tag{53}$$

The idea behind Eq. (52), (and Eq. (53)) is to discourage the optimizer from keeping a non-zero roll rate over most of the trajectory. This simple modification strongly helped reduce the number of RCS pulses at no fuel cost (given the practical invariance of the fuel cost with respect to the number and the location of the RCS pulses). An example in that sense is visible in Fig. 15, showing a case extracted from the Monte-Carlo campaign, and for which the basic (i.e., non-regularized) and the regularized cases are compared.

While without regularization the optimizer arbitrarily decides to place a first RCS pulse at the beginning of the trajectory and a second, small one in the middle of the descent, the regularized version coherently commands only one pulse at the beginning of the trajectory, and this is done with no extra-costs in terms of fuel. The overall profiles of roll angle is very similar, with minor differences in the middle part of the trajectory due to the different angular rate. The central plot shows that, consistently with the different RCS throttle profiles, the angular rate is set to 0 DEG/TU earlier in the regularized case compared to the non-regularized one, and with no difference in fuel consumption between the two cases.

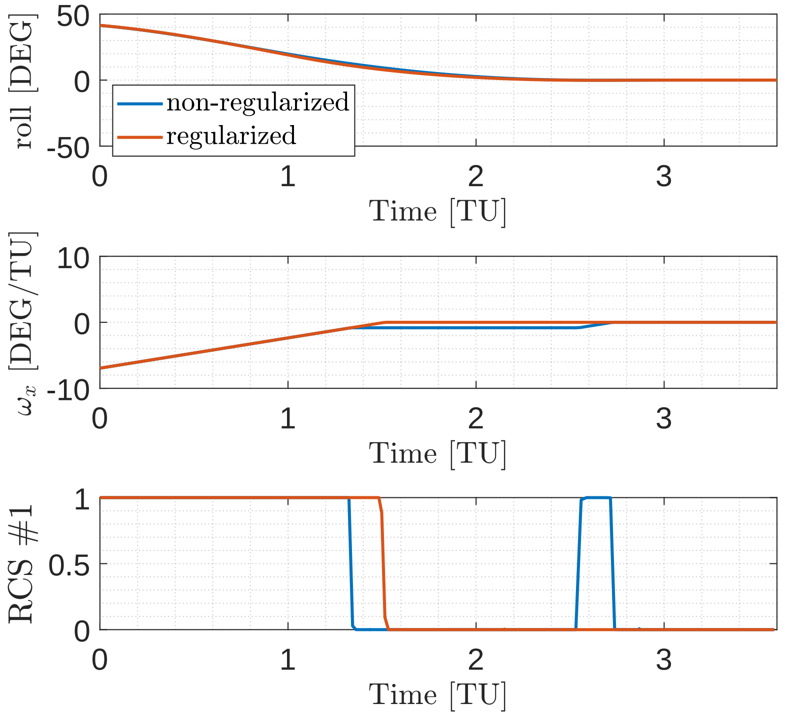


Fig. 15 Minimum-fuel problem in the presence of non-zero roll rate: non-regularized and regularized formulations

E. Convergence and Chattering Behaviors Comparison

A major benefit of using the ACCD-technique over the traditional linearization methods for nonlinear equality constraints is the suppression of the chattering phenomenon found in recent years in multiple applications of sequential convex programming [1, 22, 29]. We have also done an extensive comparison of the ACCD approach with the most commonly used linearization techniques with the 6-DoF rocket landing problem, and re-confirmed the advantages of ACCD. As a limited demonstration of the comparative study we have done, we provide in this subsection the convergence behavior (or, the lack thereof) of three different strategies to handle the quaternion-norm constraint in Eq. (33):

1) SCP with enforcement of Eq. (33) in standard linearized form

$$2(\mathbf{q}^B_{UEN,k})^T(\mathbf{q}^B_{UEN}) - (\mathbf{q}^B_{UEN,k})^T(\mathbf{q}^B_{UEN,k}) - 1 = 0 \tag{54}$$

2) SCP with the linearized constraint relaxed through the use of a slack variable v_q

$$2(\mathbf{q}^B_{UEN,k})^T(\mathbf{q}^B_{UEN}) - (\mathbf{q}^B_{UEN,k})^T(\mathbf{q}^B_{UEN,k}) - 1 + v_q = 0, \quad v_q \in \mathbb{R} \tag{55}$$

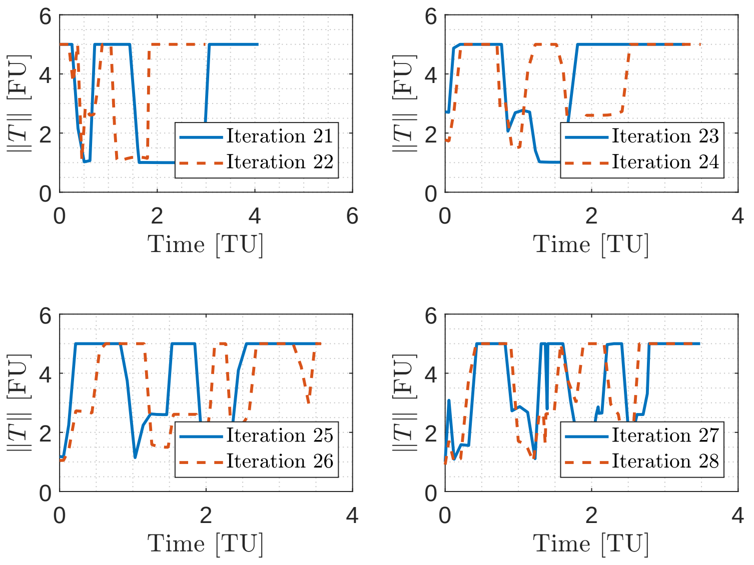
and a penalty term $k_q|\nu_q|$ with $k_q > 0$ is appended in the augmented cost function

3) SCP by using the ACCD strategy to treat Eq. (33). The initial guess is always given by linearly interpolating between the provided initial and final conditions (as visible by the blue straight lines in Figs. 16(b)-18(b).

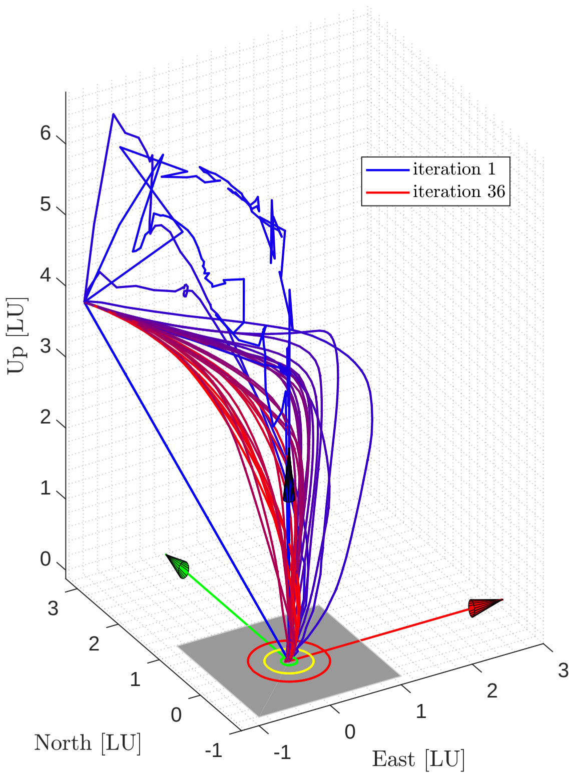
A possible alternative is given by the use of ideal transcription methods, that do not enforce the constraint explicitly, but rather rely on nonlinear exact propagation preserving it [36]. Still, while this technique can alleviate the problem for state NECs, it cannot be applied to control NECs, for which there is no nonlinear dynamics propagation (unless they get transformed into state constraints by augmenting the equations of motion). Moreover, in this context we want to focus on the chattering reduction, so we prefer to explicitly include NECs in the formulation. To make the analysis as simple as possible we formulate the problem with only one nonlinear equality constraint, i.e., the quaternion norm, while the controls are given by the classical thrust vector $\mathbf{T} \in \mathbb{R}^3$ representation. Note that it was observed that the ACCD algorithm behaves well for all the cases analyzed in this work even without the use of trust regions. However, since they are needed to get valid results for standard sequential convex approaches, only for this analysis we adopted static trust regions applied to groups of variables, i.e., at each iteration k the following inequalities must hold

$$\begin{array}{lcll} \left\|\mathbf{r}_k^{i,j} - \mathbf{r}_{k-1}^{i,j}\right\|_2 & \leq & \Delta_r & \\ \left\|\mathbf{v}_k^{i,j} - \mathbf{v}_{k-1}^{i,j}\right\|_2 & \leq & \Delta_v & i = 1, \ldots, n \\ \left\|\mathbf{q}_k^{i,j} - \mathbf{q}_{k-1}^{i,j}\right\|_2 & \leq & \Delta_q & j = 1, \ldots, p_i \\ \left\|\omega_k^{i,j} - \omega_{k-1}^{i,j}\right\|_2 & \leq & \Delta_\omega & \end{array} \tag{56}$$

with $\Delta_r = 5$ LU, $\Delta_v = 2$ LU/TU, $\Delta_q = 1$, $\Delta_\omega = 0.5$RAD/TU. Results show that ACCD is more effective not only in converging, but also in detecting the correct thrust structure. An example of this behavior is visible in Figs. 16-18, associated with initial conditions $\mathbf{r}(t_0) = [4,\ -1.2463,\ 2.8278]$ LU, $\mathbf{v}(t_0) = [-0.5953,\ 1.9983,\ -2.3061]$ LU/TU, $\omega(t_0) = [0,\ 2.9555,\ -3.7812]$ DEG/TU. In Fig. 16 we impose the quaternion norm in pure linear form. Although for some cases (like the nominal one associated with Table 2) the results are correct, this is not always the case. In fact, for this example the algorithm with standard linearization does not detect correctly the thrust profile (Fig. 16(a)). Moreover, the feasibility issues mentioned in Section II appear in the first iterations, as visible in some of the blue trajectories, which exhibit non-realistic oscillations 16(b). When we relax the NEC by using a slack variable we obtain the results in Fig. 17. In this case the solution chattering is in plain sight: each sub-problem k built on the current sub-solution k leads to a subsolution $k+1$. When the new subproblem $k+1$ is built, the subsolution $k+2$ will be very close to the subsolution k, and the cycle will start again (Fig. 17(a)). Consequently, the algorithm does not converge properly, and the maximum number of iterations (in this case 35) is reached (Fig. 17(b)). The final case is the sequence of sub-solutions generated by the use of the proposed ACCD (Fig. 18). In this case the chattering phenomenon does not occur, and the correct thrust profile is already found at around iteration 8 (Fig. 18(a)), leading in each iteration to a better and more refined solution. In fact, the sub-solutions are basically identical to each other starting from iteration 9, as visible in Fig. 18(b). Moreover, despite the larger number of variables, (and the consequent slighter larger CPU time required to get the individual convex sub-solution), having an algorithm whose outer loop converges faster can be desirable from the real-time implementation perspective.

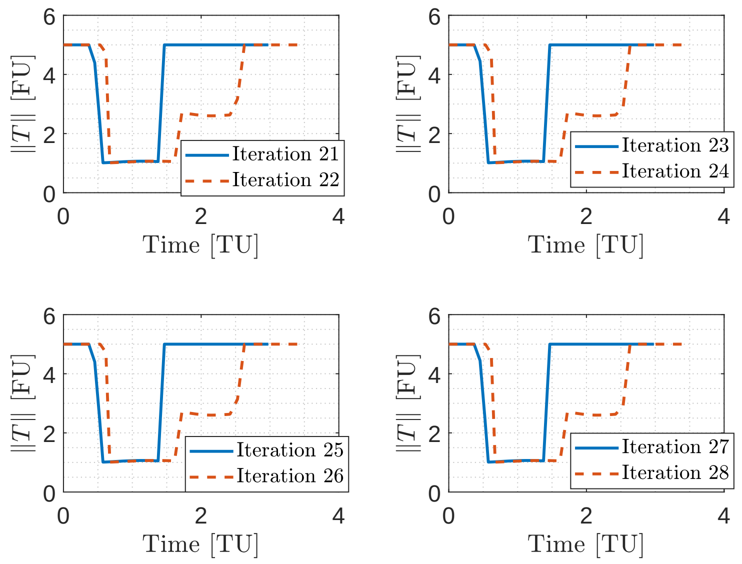


(a) Evolution of thrust magnitude profiles: note the non-convergence

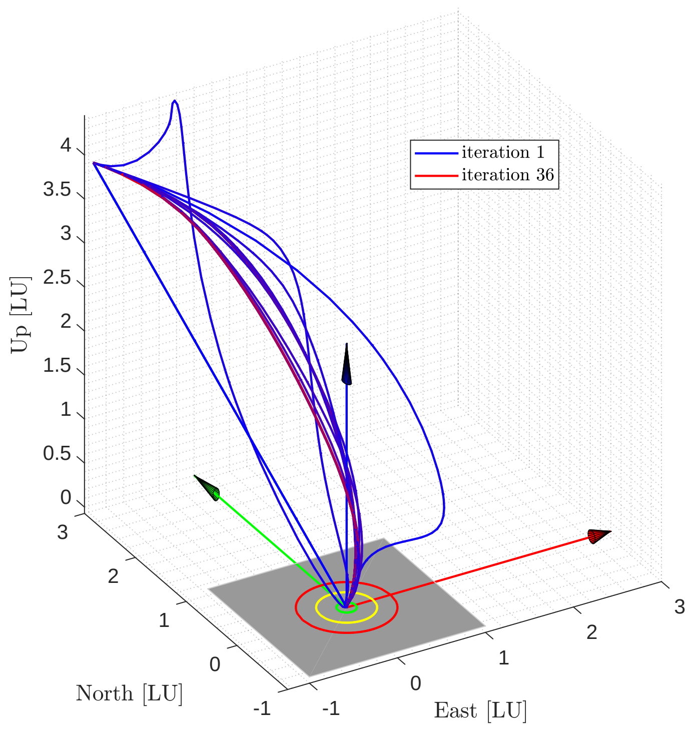


(b) Evolution of trajectories in sub-solutions; note the infeasibilty in the first few iterations

Fig. 16 Convergence of sequential convex programming with standard linearized NEC (from blue to red representing the iteration progress).

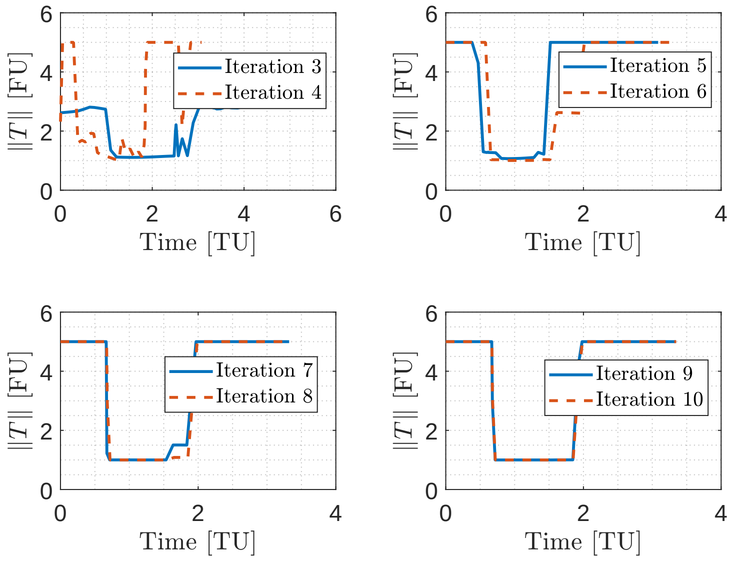


(a) Evolution of thrust magnitude profiles; note the back-and-forth solution chattering

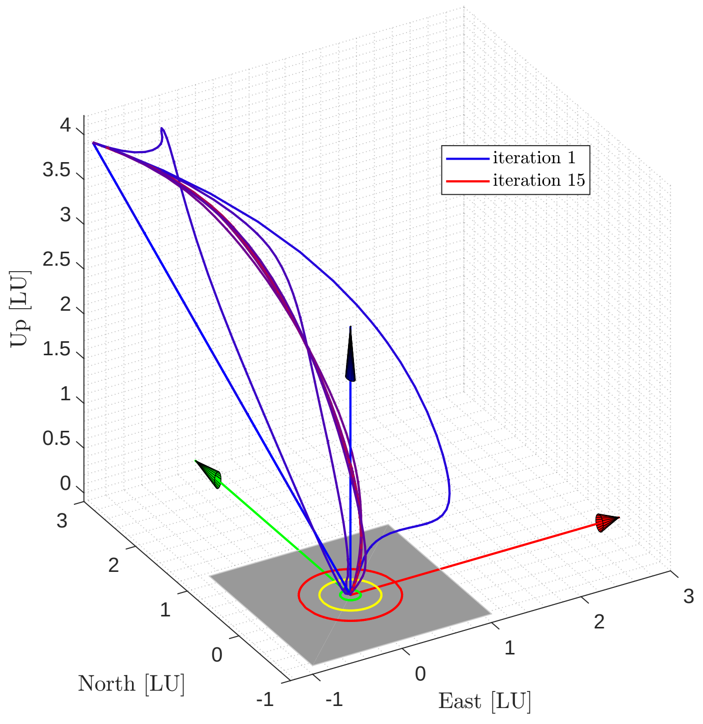


(b) Evolution of trajectories in sub-solutions

Fig. 17 Convergence of sequential convex programming with relaxed linearization of NEC (from blue to red representing the iteration progress).



(a) Evolution of thrust magnitude profiles; note the clear and rapid convergence

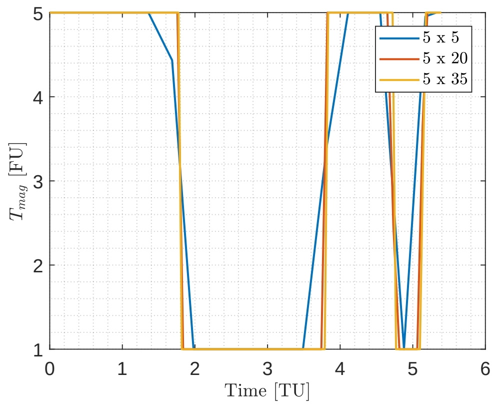


(b) Evolution/convergence of trajectories in sub-solutions

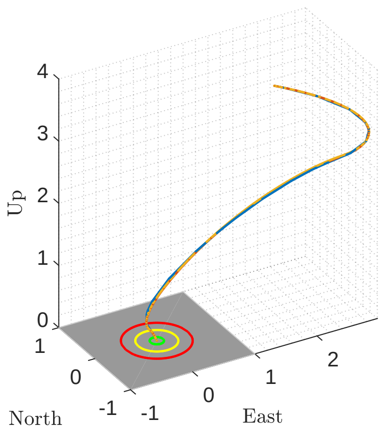
Fig. 18 Convergence of sequential convex programming with the use of ACCD (from blue to red representing the iteration progress).

F. Sensitivity with respect to the Number of Nodes

A small, separated analysis has been performed to see how the results evolve when the discretization mesh is modified. A good indicator of a robust algorithm is a low sensitivity of the solution with respect to the number of nodes, (i.e., the solution does not substantially change, but only gets refined due to the larger number of finite degrees of freedom used to approximate what would be a piece-wise continuous function). To test our algorithm we generated a solution having 4 switches, i.e., a richer structure to be identified. We repeated the trajectory optimization three times, keeping the number of segments constant and equal to 5, and increasing the number of nodes per segment (in the three cases 5, 20 and 35 nodes per segment have been adopted). Results in terms of thrust magnitude and trajectory are depicted in Fig. 19. It is observed that, when the number of nodes is increased, the algorithm finds the same control structure, and is able to make the switches sharper, as we expect (Fig. 19(a)), while no significant changes in the trajectories (Fig. 19(b)) nor in other variables (here omitted for space reasons) have been observed. Specifically, the first case (5 segments, with 5 nodes per segment) shows that the mesh is not rich enough to properly capture the second pair of switches, occurring at around 5 TU. The optimizer places a node at $T_{mag} = 1$, but it cannot do much more. Once the mesh is dense enough, the proper max-min-max-min-max structure can be reconstructed properly in both the 5×20 and 5×35 cases. These two cases are quite similar, confirming that further refinements are not needed.



(a) Thrust magnitude for different number of nodes



(b) Trajectory for different number of nodes

Fig. 19 Sensitivity analysis: thrust and trajectory obtained for different number of nodes.

G. Robustness Assessment with Dispersed Initial Conditions

Lastly, the ability of the proposed method to effectively and reliably solve the 6-DoF rocket landing problem with different initial conditions is assessed. The initial values of the state variables of dispersed cases are randomly generated (in uniform distribution) in the ranges listed in Table 3. The initial quaternion is left free to be determined by the optimizer, and is therefore not dispersed. Moreover, note that for the roll angular rate a bias of 3 DEG/TU has been added to induce a certain use of the RCS for all the simulations. For demonstration purposes 100 cases have been generated and SCP aided by the ACCD then solves the problem from the dispersed initial condition.

The resulting trajectories are depicted in Fig. 20. All the solutions converge and successfully ensure a correct pinpoint landing maneuver. Moreover, all the constraints are satisfied. From the point of view of the convergence behavior on average 21 iterations were needed to converge, with a standard deviation equal to 13. The explanation to this behavior can be rooted back to the nature of the optimal control solutions for this specific problem: for the cases having a non-small constant thrust phases in relation to the total time of flight the prescribed number of nodes is sufficient to capture well the overall optimal behavior. This translates into a smaller number of iterations, typically in the order of 10.

For those cases characterized by small min or max segments it could happen that the number and the location of the nodes were either not sufficient or not sufficiently accurate to capture the behavior. This is also the reason for which the overall accuracy requirements given by Eq. (50) and defining a successful converged case could be satisfied in 88 out of 100 cases. For the other 12 cases the level of accuracy was not reached yet, despite being the solutions dynamically feasible and close to be optimal. To confirm our conjecture we performed a repetition of some of those cases with a larger number of nodes. The analysis demonstrated in fact that the prescribed convergence could be reached if the mesh is fine enough, and suggests that further improvements could come from adaptive mesh refinement strategies (i.e., [8].

For sake of completeness we performed the same Monte-Carlo campaign with the original CCD, and reported the results in Table 4. We defined a full success a case where the stopping criterion was achieved before reaching the maximum number of iterations, and a partial success when the trajectory was valid, but the accuracy was not the required one. A failure is defined as a sequential convex approach completely diverging. It is possible to see that the ACCD is, at least for the scenario under exam here, superior to the original CCD in all the aspects. Not a single failure showed up, while the CCD diverged 41% of the cases. Moreover, the number of iterations and the CPU times are significantly smaller compared to the CCD (in this case the statistics were computed out of the 59 cases that did not diverge). These results confirm that the ACCD successfully circumvented some problematics that appear when the original formulation is applied to a demanding case such as the one that is the subject of this paper.

Finally, in terms of computational times a complete solution took on average 9 s to be generated. Note however that the computational efficiency was not the focus of the method, that instead wanted to shed a light on the robustness and the quality of the solutions in presence of NECs. Still, these values can be significantly improved with a more efficient code implementation, and the separation of offline (i.e., computations to be done only once before starting the iterative procedure) and online contributions, not taken into account in this context.

Table 3 Monte-Carlo Initial dispersions

Parameter	Value	[Unit]
$\lVert \mathbf{r}_{E,N} \rVert_2$	$[-4,\ 4]$	[LU]
$\mathbf{v}_U$	$[-1,\ 0]$	[LU/TU]
$\lVert \mathbf{v}_{E,N} \rVert_2$	$[-4,\ 4]$	[LU/TU]
$\omega_x(t_0)$	$[-8,\ -3] \cup [3,\ 8]$	[DEG/TU]
$\omega_y(t_0)$	$[-5,\ 5]$	[DEG/TU]
$\omega_z(t_0)$	$[-5,\ 5]$	[DEG/TU]

Table 4 Monte-Carlo comparison of ACCD and CCD

Metric	ACCD	CCD
Full success	88	27
Partial success	12	32
Failures	0	41
Iterations (mean)	21.61	30.46
Iterations (std)	7.85	6.24
CPU time (mean) [s]	9.31	35.75
CPU time (std) [s]	3.46	11.88

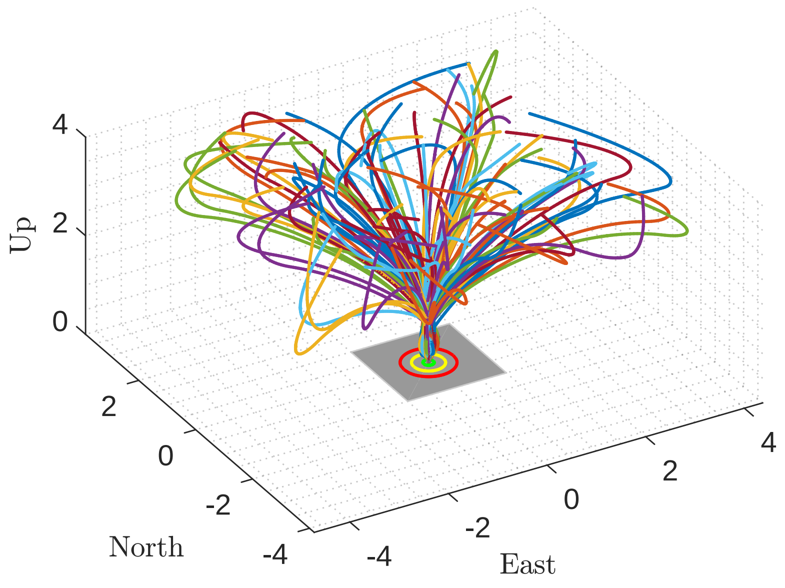


Fig. 20 One hundred dispersed fuel-optimal trajectories of the 6-DoF rocket landing problem.

VII. Conclusions

In this work we propose an augmentation of the original Convex-Concave Decomposition and extend its use to nonlinear equality constraints on the entire trajectory (as opposed to just at a point). By analyzing the feasibility space of state-of-the-art methods and the proposed Augmented Convex-Concave Decomposition (ACCD) we show that the infeasibility issues that may prevent the convergence of sequential convex programming methods are drastically improved, leading to a more robust and interpretable algorithm. In addition, the proposed ACCD technique is shown to be applicable to solve problems with discrete decision-making variables without the need for a mix-integer algorithm. The ACCD method embedded in a sequential convex programming framework is demonstrated by solving the challenging problem of full six-degrees-of-freedom fuel-optimal landing of a rocket in atmosphere, including on-off reaction control system thrusters for roll control. Using the same rocket landing problem, we provide an effective example of how the ACCD can be beneficial for the formulation and the resolution of problems with nonlinear equality constraints as the ones in use here. The use of intuitive slack variables for the proposed decomposition, together with the alleviation of the chattering phenomenon show that this technique is likely the best way to handle the specified class of nonlinear equality constraints when they are required in the formulation. The evidence presented in this paper therefore suggests that the ACCD technique would be a welcomed addition to the plethora of existing sequential convex programming methods to improve the ability to meet the long-standing challenge of handling nonlinear equality constraints effectively.

References

[1] Liu, X., Shen, Z., and Lu, P., "Entry Trajectory Optimization by Second-Order Cone Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 2, 2016, pp. 227–241. https://doi.org/10.2514/1.g001210.

[2] Wang, Z., and Grant, M. J., "Constrained Trajectory Optimization for Planetary Entry via Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 10, 2017, pp. 2603–2615. https://doi.org/10.2514/1.g002150.

[3] Acikmese, 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. https://doi.org/10.2514/1.27553.

[4] Blackmore, L., Açikmeşe, B., and Scharf, D. P., "Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 33, No. 4, 2010, pp. 1161–1171. https://doi.org/10.2514/1.47202.

[5] Sagliano, M., "Pseudospectral Convex Optimization for Powered Descent and Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018, pp. 320–334. https://doi.org/10.2514/1.g002818.

[6] Sagliano, M., "Generalized hp Pseudospectral-Convex Programming for Powered Descent and Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 7, 2019, pp. 1562–1570. https://doi.org/10.2514/1.g003731.

[7] Wang, Z., and Grant, M. J., "Optimization of Minimum-Time Low-Thrust Transfers Using Convex Programming," *Journal of Spacecraft and Rockets*, Vol. 55, No. 3, 2018, pp. 586–598. https://doi.org/10.2514/1.a33995.

[8] Hofmann, C., and Topputo, F., "Rapid Low-Thrust Trajectory Optimization in Deep Space Based on Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 7, 2021, pp. 1379–1388. https://doi.org/10.2514/1.g005839.

[9] Lu, P., and Liu, X., "Autonomous Trajectory Planning for Rendezvous and Proximity Operations by Conic Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 36, No. 2, 2013, pp. 375–389. https://doi.org/10.2514/1.58436.

[10] Liu, X., and Lu, P., "Robust Trajectory Optimization for Highly Constrained Rendezvous and Proximity Operations," *AIAA Guidance, Navigation, and Control (GNC) Conference*, American Institute of Aeronautics and Astronautics, 2013. https://doi.org/10.2514/6.2013-4720.

[11] 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. https://doi.org/10.1109/mcs.2022.3187542.

[12] Szmuk, M., and Acikmese, B., "Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time," *2018 AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, 2018. https://doi.org/10.2514/6.2018-0617.

[13] Yang, R., and Liu, X., "Fuel-optimal powered descent guidance with free final-time and path constraints," *Acta Astronautica*, Vol. 172, 2020, pp. 70–81. https://doi.org/10.1016/j.actaastro.2020.03.025.

[14] Szmuk, M., Acikmese, B., and Berning, A. W., "Successive Convexification for Fuel-Optimal Powered Landing with Aerodynamic Drag and Non-Convex Constraints," *AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, 2016. https://doi.org/10.2514/6.2016-0378.

[15] Sagliano, M., Heidecker, A., Hernández, J. M., Farì, S., Schlotterer, M., Woicke, S., Seelbinder, D., and Dumont, E., "Onboard Guidance for Reusable Rockets: Aerodynamic Descent and Powered Landing," *AIAA Scitech 2021 Forum*, American Institute of Aeronautics and Astronautics, 2021. https://doi.org/10.2514/6.2021-0862.

[16] Reynolds, T. P., Szmuk, M., Malyuta, D., Mesbahi, M., Açıkmeşe, B., and Carson, J. M., "Dual Quaternion-Based Powered Descent Guidance with State-Triggered Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 9, 2020, pp. 1584–1599. https://doi.org/10.2514/1.g004536.

[17] 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. https://doi.org/10.2514/1.g004549.

[18] Liu, X., "Fuel-Optimal Rocket Landing with Aerodynamic Controls," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 1, 2019, pp. 65–77. https://doi.org/10.2514/1.g003537.

[19] Sagliano, M., and Mooij, E., "Optimal drag-energy entry guidance via pseudospectral convex optimization," *Aerospace Science and Technology*, Vol. 117, 2021, p. 106946. https://doi.org/10.1016/j.ast.2021.106946.

[20] Liu, X., Li, S., and Xin, M., "Mars Entry Trajectory Planning with Range Discretization and Successive Convexification," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 4, 2022, pp. 755–763. https://doi.org/10.2514/1.g006237.

[21] Stephen Boyd, L. V., *Convex Optimization*, Cambridge University Press, 2004. URL https://www.ebook.de/de/product/3677442/stephen_boyd_lieven_vandenberghe_convex_optimization.html.

[22] Lu, P., "Convex–Concave Decomposition of Nonlinear Equality Constraints in Optimal Control," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 1, 2021, pp. 4–14. https://doi.org/10.2514/1.g005443.

[23] Lu, P., Lewis, A., Adams, R. J., DeVore, M. D., and Petersen, C. D., "Finite-Thrust Natural-Motion Circumnavigation Injection by Convex Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 45, No. 3, 2022, pp. 453–467. https://doi.org/10.2514/1.g006123.

[24] Floudas, C. A., *Nonlinear and Mixed-Integer Optimization*, Oxford University Press, 1995. https://doi.org/10.1093/oso/9780195100563.001.0001.

[25] Malyuta, D., Reynolds, T., Szmuk, M., Acikmese, B., and Mesbahi, M., "Fast Trajectory Optimization via Successive Convexification for Spacecraft Rendezvous with Integer Constraints," *AIAA Scitech 2020 Forum*, American Institute of Aeronautics and Astronautics, 2020. https://doi.org/10.2514/6.2020-0616.

[26] Liu, X., and Lu, P., "Solving Nonconvex Optimal Control Problems by Convex Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 3, 2014, pp. 750–765. https://doi.org/10.2514/1.62110.

[27] Mao, Y., Szmuk, M., Xu, X., and Acikmese, B., "Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems," , 2018. https://doi.org/10.48550/ARXIV.1804.06539.

[28] Bonalli, R., Cauligi, A., Bylard, A., and Pavone, M., "GuSTO: Guaranteed Sequential Trajectory Optimization via Sequential Convex Programming," , 2019. https://doi.org/10.48550/ARXIV.1903.00155.

[29] Wang, Z., and Lu, Y., "Improved Sequential Convex Programming Algorithms for Entry Trajectory Optimization," *Journal of Spacecraft and Rockets*, Vol. 57, No. 6, 2020, pp. 1373–1386. https://doi.org/10.2514/1.a34640.

[30] Domahidi, A., Chu, E., and Boyd, S., "ECOS: An SOCP solver for embedded systems," *2013 European Control Conference (ECC)*, IEEE, 2013. https://doi.org/10.23919/ecc.2013.6669541.

[31] Sagliano, M., Lu, P., Seelbinder, D., and Theil, S., "Six-Degree-of-Freedom Rocket Landing Optimization by Augmented Convex-Concave Decomposition," *AIAA SCITECH 2023 Forum*, American Institute of Aeronautics and Astronautics, 2023. https://doi.org/10.2514/6.2023-2005.

[32] Pei, C., You, S., Dai, R., and Rea, J. R., "Mixed-Input Learning for Multi-point Landing Guidance with Hazard Avoidance Part I: Offline Mission Planning based on Multi-Stage Optimization," *AIAA SCITECH 2023 Forum*, American Institute of Aeronautics and Astronautics, 2023. https://doi.org/10.2514/6.2023-1445.

[33] Markley, F. L., and Crassidis, J. L., *Fundamentals of Spacecraft Attitude Determination and Control*, Springer New York, 2014. https://doi.org/10.1007/978-1-4939-0802-8.

[34] Garg, D., Patterson, M. A., Francolin, C., Darby, C. L., Huntington, G. T., Hager, W. W., and Rao, A. V., "Direct trajectory optimization and costate estimation of finite-horizon and infinite-horizon optimal control problems using a Radau pseudospectral method," *Computational Optimization and Applications*, Vol. 49, No. 2, 2009, pp. 335–358. https://doi.org/10.1007/s10589-009-9291-0.

[35] Patterson, M. A., and Rao, A. V., "GPOPS-II," *ACM Transactions on Mathematical Software*, Vol. 41, No. 1, 2014, pp. 1–37. https://doi.org/10.1145/2558904.

[36] Malyuta, D., Reynolds, T., Szmuk, M., Mesbahi, M., Acikmese, B., and Carson, J. M., "Discretization Performance and Accuracy Analysis for the Rocket Powered Descent Guidance Problem," *AIAA Scitech 2019 Forum*, American Institute of Aeronautics and Astronautics, 2019. https://doi.org/10.2514/6.2019-0925.