Six-Degrees-of-Freedom Rocket Landing Optimization by Augmented Convex-Concave Decomposition

Marco Sagliano *
German Aerospace Center, Bremen, Germany, 28359

Ping Lu†
San Diego State University, San Diego, United States, 92182

David Seelbinder‡
German Aerospace Center, Bremen, Germany, 28359

Stephan Theil§
German Aerospace Center, Bremen, Germany, 28359

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 6-DoF rocket landing problem, subject to multiple nonlinear equality constraints. Compared with known approaches such as standard Sequential Convex Programming, it is shown that the proposed ACCD leads to a more robust methodology for the computation of optimal solutions, and to a much more interpretable behavior of the sequential convex algorithm. 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 space applications, ranging from entry guidance problems [1, 2], pinpointing landing rockets [3–6], to low-thrust interplanetary trajectory computation [7, 8], and rendezvous and docking problems [9]. The application of these techniques came in several flavors, depending on the type of problem under exam, 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 [10, 11], inclusion of non-convex aerodynamic effects [12, 13], state-triggered constraints [14, 15], need to retain some features of the original nonlinear problem [16], and choice of a different independent variable rather than time [17, 18].

One such difficult issue might arise from problems having convex, but nonlinear equality constraints. It is well-known that the only equality constraints compatible with convex optimization problems are linear ones [19]. This limitation suggests that an obvious way to deal with nonlinear equality constraints is the use of linearization techniques applied to the current sub-solution. However, this approach does not always provide a smooth convergence, as shown for the Natural-Motion Circumnavigation Injection problem [20].

To circumvent these difficulties the Convex-Concave Decomposition (CCD) was proposed, leading to excellent results for the aforementioned category of problems [20, 21]. However, for different problems this methodology might suffer from infeasibility issues, especially at the beginning of the sequential convex programming approach. The interaction between dynamics, boundary conditions and nonlinear equality constraints can cause convergence issues of the first subproblems. Although 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 solution is the result of an infeasible problem there is no guarantee that such solution is a meaningful point to

*GNC Senior Researcher, German Aerospace Center, AIAA Senior Member
†Professor and Chair, Department of Aerospace Engineering, Fellow AIAA
‡G&C Group Head, German Aerospace Center
§GNC Department Head, German Aerospace Center

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 less deterministic. Finally, it is highly desired 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 presence of equality constraints obeying mild assumptions. We analyze the application of the original convex-concave decomposition method in Ref. [20] to a different class of problems. We will show that, while the original convex-concave decomposition is effective for the problems in Ref. [20], it may suffer from infeasibility issues under some specific conditions. Similar issues can also plague state-of-the-art successive convex optimization techniques when the class of nonlinear, convex 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 show that the convex-concave decomposition paradigm can effectively be extended to a large number of variables without causing issues to the algorithm. This outcome was envisioned and expected in the original formulation by Lu [20], but not practically demonstrated yet. Second, 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. By building upon the outcome of this analysis we propose a method dubbed Augmented Convex-Concave Decomposition (ACCD) to mitigate any feasibility issues that otherwise may arise in the initialization problems when state-of-the-art sequential convex optimization techniques are applied in presence of nonlinear equality constraints. By leveraging the use of the exact penalty function theory applied to a set of slack variables included in the formulation we show that, with the proposed methodology, the feasibility of the problem is ensured from the beginning of the iterative process, leading in many cases to a smoother convergence of the overall algorithm. Third, we apply the new framework to solve a free-time, multi-constrained, complex problem of optimal landing of a reusable rocket subject to six-degree-of-freedom (6DOF) dynamics. This application also gives the chance of formulating the problem in a modified version that takes advantage of the proposed augmented convex-concave decomposition methodology we are proposing.

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 analyze the corresponding Karush–Kuhn–Tucker conditions to assess the behavior of the proposed ACCD methodology; we further investigate the feasibility domains of Sequential Convex methods in presence of Nonlinear Equality Constraints (NECs), how such feasible space is modifed by the use of the standard Convex-Concave Decomposition, and finally how the feasibility is improved by the use of the proposed ACCD for a classical NEC such as the quaternion norm constraint. We finally combine the use of ACCD with hp sequential pseudospectral convex programming and apply the method to a 6-DOF pinpoint propellant-optimal landing problem of a rocket in atmosphere.

The paper is organized as follows: Section II contains a general description of state-of-the-art sequential convex programming methods, while Sec. III describes the core idea and the limitations of the classic convex-concave-decomposition. In Sec. IV we describe the augmented convex-concave decomposition technique proposed, together with the KKT conditions deriving from its use. The feasibility space analysis of the three methodologies is discussed throug the corresponding sections. Sec. 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 Sec. VI shows some numerical examples of the methodology. Finally, in Sec. VII we draw some conclusions about the work performed and the results we 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], [22], to low-thrust optimization [8], guidance of reusable rockets in the aerodynamic phase [13], and during the powered descent [1, 15]. Although convergence proofs are limited to specific cases [23, 24] the methodology works heuristically well in a large number of situations. 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, 17].

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

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

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

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, whereas the term $\mathbf{G}$ 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 [13, 15]. 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* [23], coming from the use of linearization. The common technique is the adoption of the trust region mechanism, e.g.,

$$\|\mathbf{x} - \mathbf{x}_k\|_2 \preceq \delta \tag{6}$$

where the symbol $\preceq$ can be interpreted as component-wise, applied to the 2-norm of the vector $\mathbf{x} - \mathbf{x}_k$, or even as max value, depending on the specific implementation. The trust-region radius δ can be a constant (e.g., [1, 22]), a dynamic variable penalized and appended to the cost function (e.g., [13, 15]), or a value that is updated between iterations (e.g., [25]). 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

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

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 versions with possible inclusion of slack variables in the equalities to counter artificial infeasibility issues. 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. This type of situation is the subject of 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. [20], 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., [16]). As highlighted by Lu in [20] there are some situations for which performing a linearization of Eq. (8) by using the approach of Eq. (6) leads to convergence problems, called *chattering*. The main idea behind the CCD is to prevent such behavior by replacing the constraint of Eq. (8) with a set of three inequalities [20]

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

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 $\mathbf{x}_k$:

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

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 with a finite penalty coefficient on ϵ, whereas a similar argument does not hold for the standard sequential convex approach described in Sec. 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. [22]. 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 $\mathbf{q}_{t_0}$ to $\mathbf{q}_{t_F}$ as

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

with the condition

$$\mathbf{q}^T(t_i)\mathbf{q}(t_i) - 1 = 0, \qquad \begin{matrix} i = 0, \ldots, N \\ \mathbf{q}(t_0) = \mathbf{q}_{t_0} \\ \mathbf{q}(t_N) = \mathbf{q}_{t_F} \end{matrix} \tag{13}$$

representing our NEC, discretized in $N + 1$ nodes, and with the quaternions having their scalar part as 4^{th} component. To build our example we implemented the classical 6-DoF dynamics of the spacecraft as given in [13], 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. The boundary conditions are defined completely

$$\mathbf{q}(t_0) = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{q}(t_N) = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \tag{14}$$

and we want to perform the slew maneuver in a given time interval, in this case 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

$$\mathbf{q}(t_i) = 2 \begin{bmatrix} 0 \\ \cos(\theta_i) \\ 0 \\ \sin(\theta_i) \end{bmatrix}, \qquad \begin{matrix} i = 0, \dots, N \\ \theta_i = i \frac{\pi}{2N} \end{matrix} \tag{15}$$

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 returns an infeasibility certificate. It should be point 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 [26] 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 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, such space is physically separated from the concave one, 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.

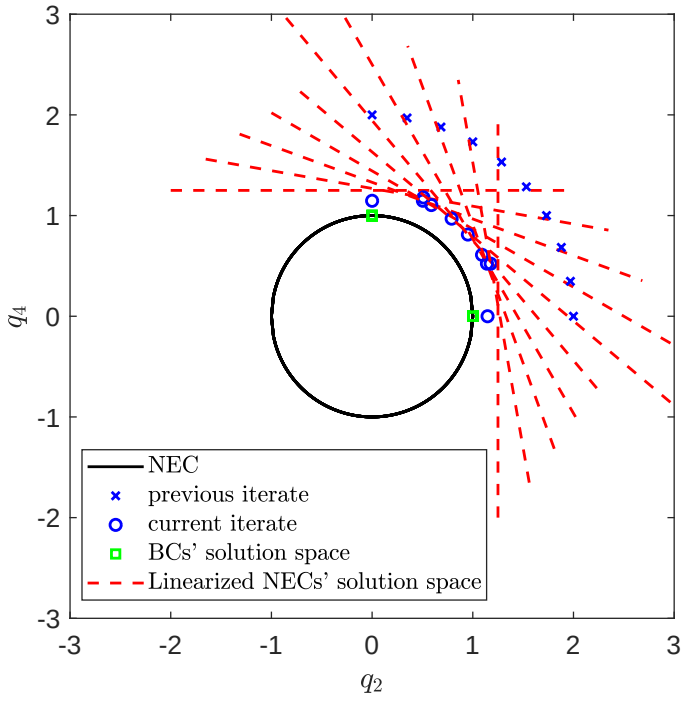


Fig. 1 First iteration with standard sequential convex programming in the presence of a nonlinear equality constraint: the linearized equality constraints do not admit the required boundary conditions.

IV. Augmented Convex-Concave Decomposition

A. Formulation of Augmented Convex-Concave Decomposition

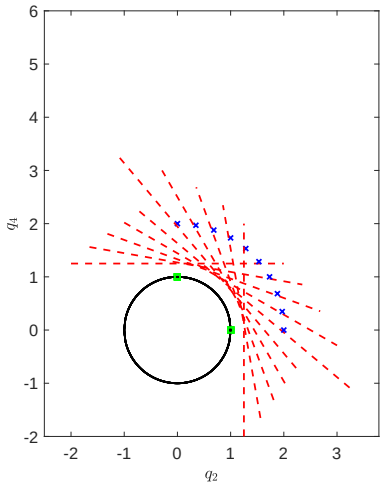
To overcome the limitations emphasized in Sec. III let us consider to move from the use of a single ϵ to a pair of variables ϵ_1, ϵ_2 for each of the NECs that we have. So, the set of inequalities in [20],

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

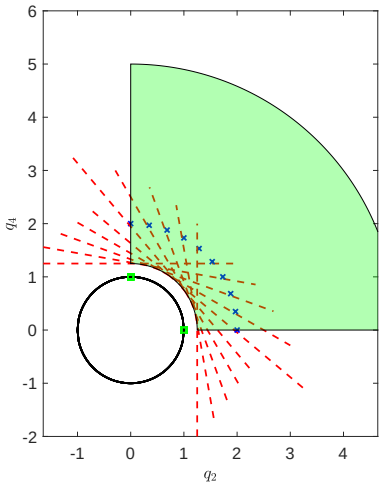
becomes as follows.

$$\begin{aligned} h(\mathbf{x}) &\leq \epsilon_1 \\ -h(\mathbf{x}_k) - \nabla h(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) &\leq \epsilon_2 \\ -\epsilon_1 &\leq 0 \\ -\epsilon_2 &\leq 0 \end{aligned} \tag{17}$$

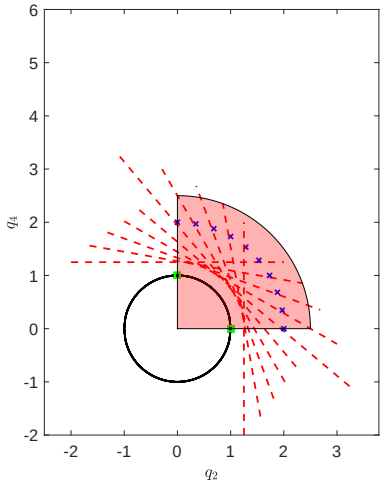
Note that the above formulation is at one collocation point, and an independent pair of $\{\epsilon_1, \epsilon_2\}$ is used at each collocation point. How did the previous example behave in this case? Results are depicted in Figs. 3, 4, where we show how the algorithm behaves in generating the first solution starting from the initial guess provided. Moreover, we also show the behavior for an initial guess generated within the NEC domain (5) as further example (Figs. 5), 6). These two cases are labeled Outside Initial Guess (OIG) and Inside Initial Guess (IIG), respectively. 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 variable ϵ_2 allows for an expansion of the concave inequality feasible set (Fig. 3(c)) such that the intersection of the feasible set of the inequalities and the boundary conditions is always non-empty (Fig. 3(d)).



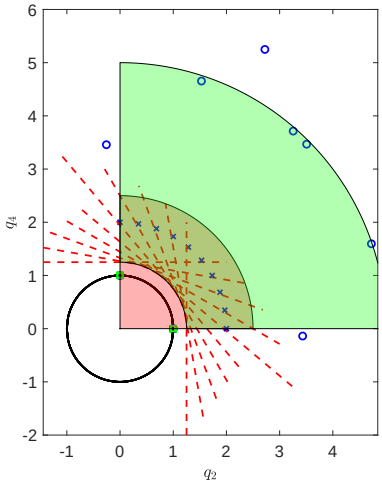
(a) Feasible space associated with boundary conditions



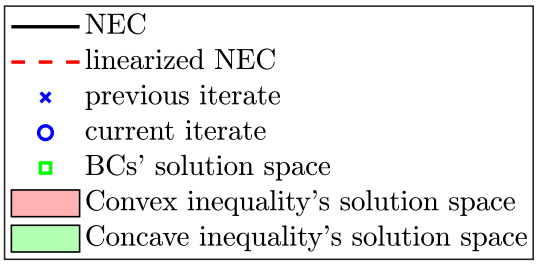
(b) Feasible space associated with concave inequality



(c) Feasible space associated with convex inequality

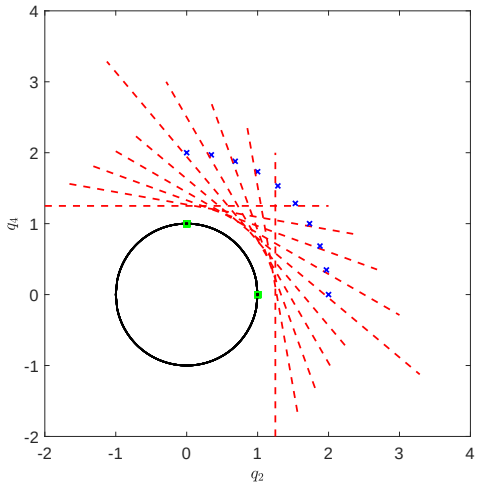


(d) Intersection of feasible space associated with convex inequality, concave inequality and boundary conditions

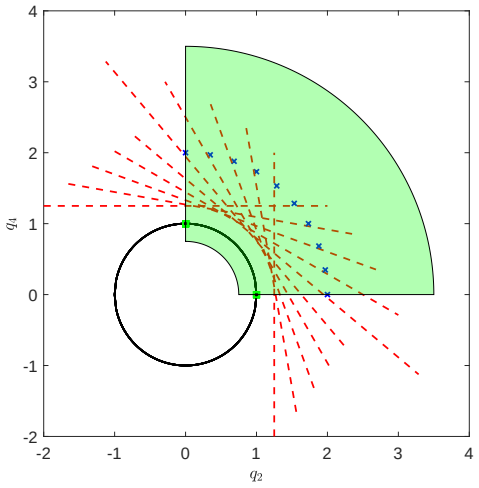


(e) legend

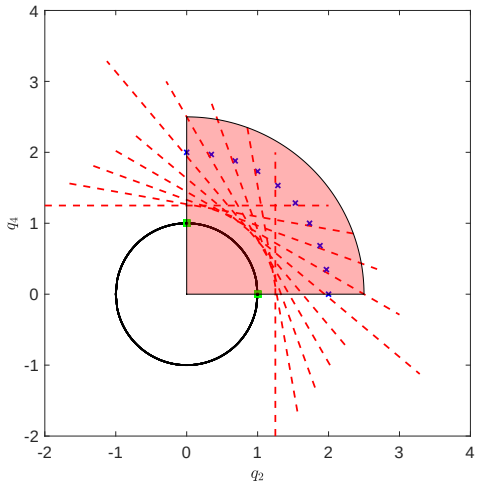
Fig. 2 Analysis of feasible space associated with the use of Convex-Concave Decomposition for the slew maneuver in 2-D: the intersection is empty.



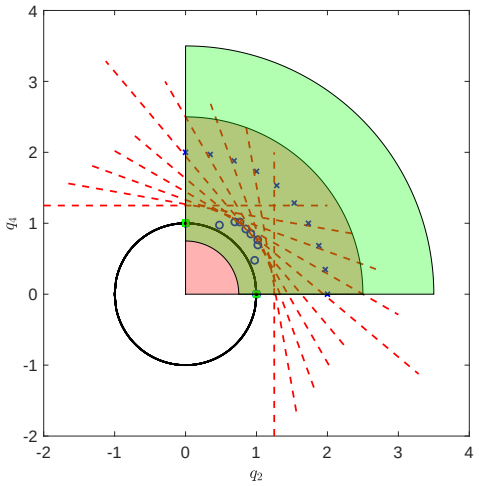
(a) Feasible space associated with boundary conditions



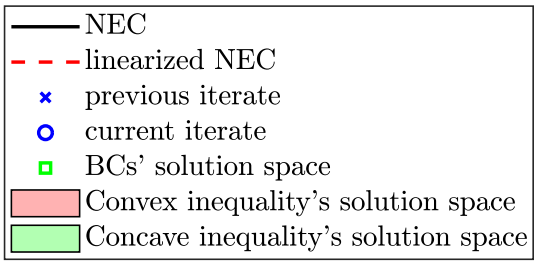
(b) Feasible space associated with concave inequality



(c) Feasible space associated with convex inequality



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



(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.

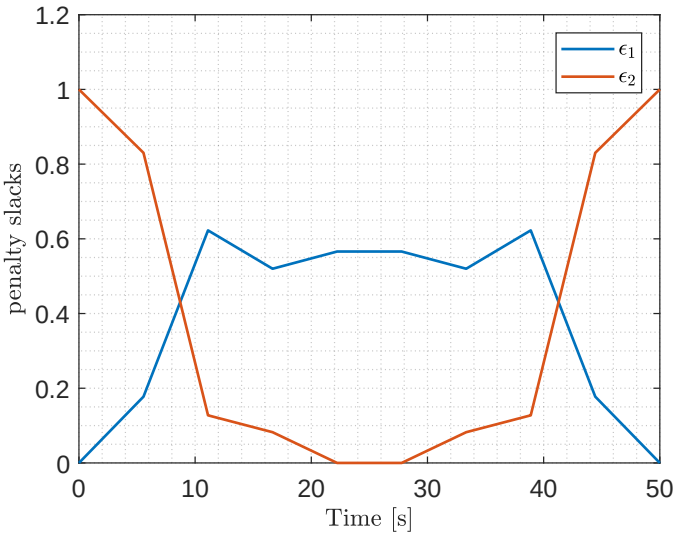


Fig. 4 First iteration with Sequential Convex Programming and ACCD in presence of a Nonlinear Equality Constraint for the OIG case: the intersection between the feasible space for boundary constraints and the NEC is non-empty, allowing the algorithm to find a valid solution.

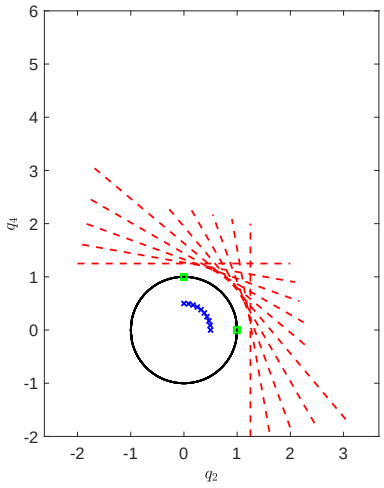
For the case in Fig. 3 we can see that the algorithm moves several points close to the original NEC, while some others are close to the linearized one. This is the result of the compromise between the dynamics, that pushes for having the points exactly on the NEC, as direct consequence of the quaternion 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 the penalty slacks 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 $i = 1, \dots, N-1$ as complementary behavior to ϵ_2, and within few iterations both can be pushed to be close to 0. The two ϵ have therefore a clear interpretation, leading to a more understandable behavior of the intermediate sub-solutions.

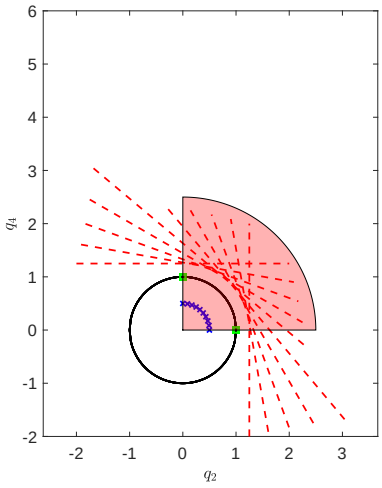
This logic is even more evident when we use another initial guess that is inside the unit circle

$$\mathbf{q}(t_i) = \frac{1}{2}\begin{bmatrix} 0 \\ \cos(\theta_i) \\ 0 \\ \sin(\theta_i) \end{bmatrix}, \qquad \begin{matrix} i = 0, \dots, N \\ \theta_i = i\frac{\pi}{2N} \end{matrix} \tag{18}$$

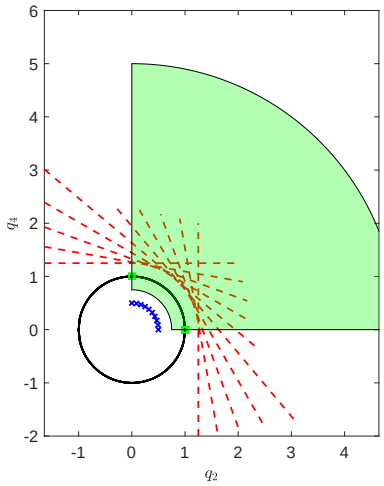
The first iterate starting from this initial guess under the ACCD is illustrated in Figs. 5 and 6. The corresponding feasible spaces are the same as in the previous case (Figs. 5(a)-5(c)). In this case the algorithm places the points almost immediately on the original NEC (Fig. 5(d)), and, in doing so, it pays a price in terms of ϵ_2, which is equal to $\cong 0.25$ in all the points (Fig. 6). This is exactly the gap occurring between the linearized NEC built around the IIG and the actual NEC. Since the points are basically satisfying the NEC, no work from ϵ_1 is in this case required, leading to the corresponding ϵ_1 profile constant and equal to $\cong 0$. The pair ϵ_1, ϵ_2 can therefore robustify the behavior of Sequential Convex Programming methods by 1) guaranteeing the feasibility of each iteration, (and consequently the computation of a meaningful solution needed for the construction of the subsequent cycle), and 2) improving the readability of the algorithm itself, since the ϵ_1, ϵ_2 variables directly act as a measure of the constraints' violations.



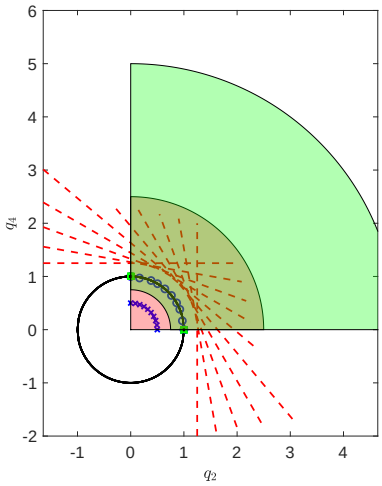
(a) Feasible space associated with boundary conditions



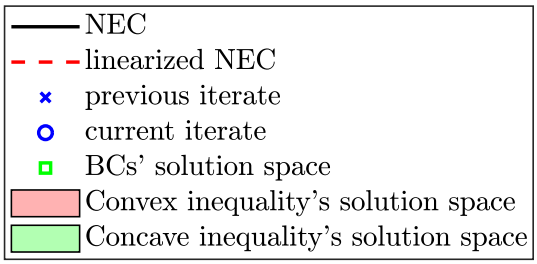
(b) Feasible space associated with convex inequality



(c) Feasible space associated with concave inequality



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



(e) legend

Fig. 5 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.

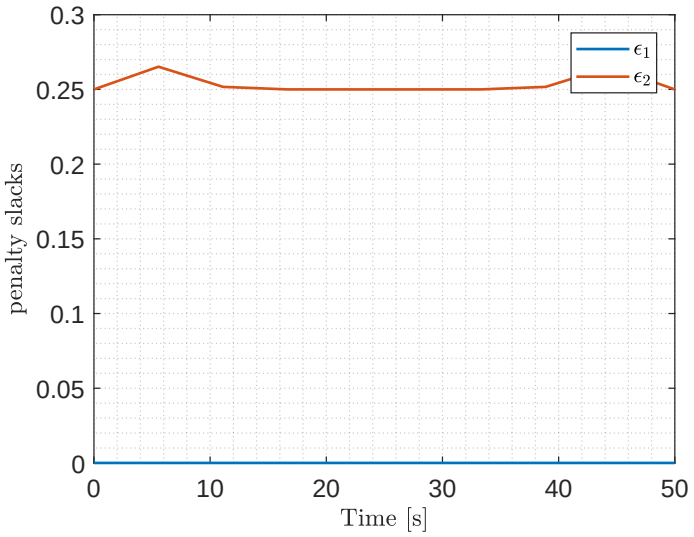


Fig. 6 First iteration with Sequential Convex Programming and ACCD in presence of a Nonlinear Equality Constraint for the IIG case: the intersection between the feasible space for boundary constraints and the NEC is non-empty, allowing the algorithm to find a valid solution.

B. Karush–Kuhn–Tucker Conditions for the Augmented Convex-Concave Decomposition

Let us consider the augmented cost function coming from the use of the two penalty slack variables we introduced in Eq. (17), in combination with a generic cost function $J_0(\mathbf{x})$. To simplify the discussion and get the central point across, consider the problem only at one collocation point (at a fixed time), and neglect the linear system of equations representing the dynamics. The augmented cost function $J_1(\mathbf{x}, \epsilon_1, \epsilon_2)$ reads

$$J_1(\mathbf{x}, \epsilon_1, \epsilon_2) = J_0(\mathbf{x}) + p_1\epsilon_1 + p_2\epsilon_2 \tag{19}$$

where $p_1 > 0$ and $p_2 > 0$ are penalty coefficients. The Lagrangian is

$$\mathcal{L} = J_0(\mathbf{x}) + p_1\epsilon_1 + p_2\epsilon_2 + \mu_1\left[h(\mathbf{x}) - \epsilon_1\right] + \mu_2\left[-h(\mathbf{x}_k) - \nabla h(\mathbf{x}_k)(\mathbf{x} - \mathbf{x}_k) - \epsilon_2\right] - \mu_3\epsilon_1 - \mu_4\epsilon_2 \tag{20}$$

The KKT conditions give

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \epsilon_1} &= p_1 - \mu_1 - \mu_3 = 0 \\ \frac{\partial \mathcal{L}}{\partial \epsilon_2} &= p_2 - \mu_2 - \mu_4 = 0 \end{aligned} \tag{21}$$

where all $\mu_i \geq 0$. Qualitatively, Eq. (21) (which is equivalent to $\mu_1 + \mu_3 = p_1$ and $\mu_2 + \mu_4 = p_2$) suggests that choices of larger $p_1 > 0$ and $p_2 > 0$ are likely to cause μ_i's to be become nonzero (positive). Consequently the KKT complementary conditions imply that larger $p_1 > 0$ and $p_2 > 0$ will tend to drive the solution to a point where all the inequality constraints in Eq. (17) to become active, where $\epsilon_1 = 0$ and $\epsilon_2 = 0$ and the NEC is satisfied.

Depending on the values of ϵ_1 and ϵ_2 we can identify 4 different cases:
Case 1: $\epsilon_1 = 0$, $\epsilon_2 = 0$. In this case the 2 inequality constraints in Eq. (17) can only both be active. This implies that $h(\mathbf{x}) = 0$. So the solution is exact.
Case 2: $\epsilon_1 = 0$, $\epsilon_2 > 0$. In this case $\mu_4 = 0$. The second condition in Eq. (21) gives $\mu_2 = p_2 > 0$. Therefore the linearized constraint in Eq. (17) is active. At the solution $h(\mathbf{x}) \leq 0$ which allows the possibility of $h(\mathbf{x}) = 0$. Even if $h(\mathbf{x}) = 0$ at the solution, an $\epsilon_2 > 0$ may still be needed for this $\mathbf{x}$ to satisfy the linearized constraint when the linearized form built at the previous iteration still has a gap with respect to the original constraint. This is the case for instance that we see in Fig. 5.
Case 3: $\epsilon_1 > 0$, $\epsilon_2 = 0$. In this case $\mu_3 = 0$, and the first equation in Eq. (21) leads to $\mu_1 = p_1 > 0$. This indicates that $h(\mathbf{x}) - \epsilon_1 = 0$. Hence ϵ_1 will directly measure the violation of the original NEC. Indeed, this is what we see in the inner points $i = 1, \ldots, N-1$ in Fig. 3.

Case 4: $\epsilon_1 > 0$, $\epsilon_2 > 0$. This case is possible as well, and cannot be excluded. This is what is observed for instance in the points $i = 1$, $i = N - 1$ depicted in Fig. 3. Nevertheless, Cases 3 and 4 are acceptable as the result of an intermediate iteration. If the problem has a solution where the NEC is satisfied, the exact penalty function theory ensures that sufficiently large but finite values of p_1 and p_2 will result in sufficiently small ϵ_1 and ϵ_2 to consider the original NEC satisfied within a certain tolerance, consistently with what shown in [20] in presence of a single ϵ.

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. 6-DoF Optimal Rocket Landing

Let us consider the 6-DoF Rocket Landing Problem in atmosphere. We are interested in minimum-time/minimum-fuel formulations. Both setups can be framed in the following cost function to be minimized.

$$\text{minimize } J = (1 - c_F)t_F + c_F \int_{t_0}^{t_F} \|\mathbf{T}(t)\|_2 \, dt \tag{22}$$

where t_0 is the initial time (assumed known, and equal to 0), t_F is the final time, $\mathbf{T}(t) \in \mathbb{R}^3$ is the thrust vector, and $c_F \in \{0, 1\}$ is an integer, equal to 0 for the minimum-time formulation, and 1 for the fuel-optimal problem. The equations of motion are formulated as

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

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), $\mathbf{q}^B_{UEN}(t)$ embeds 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 $\boldsymbol{\Omega}_B$ is the skew matrix built on the angular rate vector ω_B, defined as [27].

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

The angular acceleration is characterized by the Inertia matrix $\mathbf{J}$, and by the total external moments acting on the system, that is, the Reaction-Control-System (RCS), and the Thrust-Vector-Control (TVC) Moments $\mathbf{M}_{RCS}$ and $\mathbf{M}_{TVC}$, with the former equal to 0 N m, and the latter given 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 is constant (i.e., no shift due to the fuel consumption). All these simplifications can be removed with some further modeling efforts (e.g., [17]), and do not change the scope of the demonstration here.

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

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

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. [10], 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 glide-slope constraint as γ,

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

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

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

Finally, we want to explicitly impose the unit-norm constraint on the quaternions for all $t \in [t_0, t_F]$, that is,

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

While the evolution of the quaternions governed by the continuous differential equations in Eq. (23) preserves the unit norm if $\|\mathbf{q}^B_{UEN}(t_0)\| = 1$, 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 the next section 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. Transcription through ACCD-based Sequential Pseudospectral Convex Programming

With the ACCD defined, we can now transcribe our Optimal Control Problem. The transcription will partially leverage the Sequential Pseudospectral Convex method proposed in [13] 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 [28]. The domain construction

follows the scheme of [13] 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. The domain is represented in Fig. 7. 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

$$t^{i,j} = t_0 + \frac{t_F - t_0}{n}(i - \frac{1}{2}) + \frac{t_F - t_0}{2n}\tau^{i,j}, \qquad \begin{matrix} i = 1, \dots, n \\ j = 1, \dots, p_i \end{matrix} \tag{34}$$

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

$$(\mathbf{u}^{i,j})^T(\mathbf{u}^{i,j}) - 1 = 0, \qquad \begin{matrix} i = 1, \dots, n \\ j = 1, \dots, p_i \end{matrix} \tag{35}$$

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

$$\begin{matrix} (\mathbf{u}^{i,j})^T(\mathbf{u}^{i,j}) - 1 - \epsilon_1 \leq 0 \\ -2(\mathbf{u}_k^{i,j})^T(\mathbf{u}^{i,j}) + (\mathbf{u}_k^{i,j})^T(\mathbf{u}_k^{i,j}) + 1 - \epsilon_2 \leq 0 \\ -\epsilon_1 \leq 0 \\ -\epsilon_2 \leq 0 \end{matrix}, \qquad \begin{matrix} i = 1, \dots, n \\ j = 1, \dots, p_i \end{matrix} \tag{36}$$

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

$$T_{min} \leq T_{mag}^{i,j} \leq T_{max}, \qquad \begin{matrix} i = 1, \dots, n \\ j = 1, \dots, p_i \end{matrix} \tag{37}$$

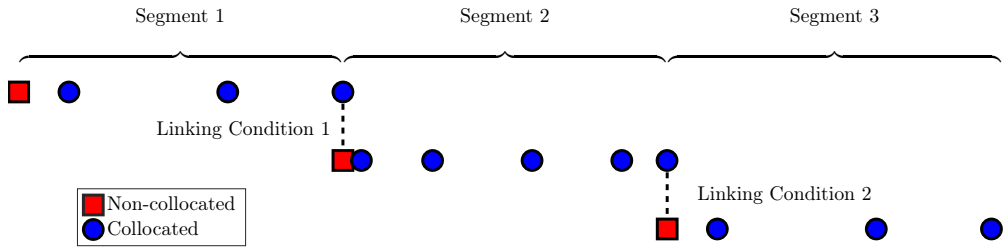


Fig. 7 Example of hp pseudospectral domain with three segments having a different number of collocation points per segment.

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

$$[\mathbf{q}_{UEN}^{B,i,j}]^T \cdot \mathbf{q}_{UEN}^{B,i,j} - 1 = 0, \qquad \begin{matrix} i = 1, \dots, n \\ j = 0, \dots, p_i \end{matrix} \tag{38}$$

Following the ACCD method Eq. (38) is represented in the algorithm by

$$\begin{matrix} (\mathbf{q}_{UEN}^{B,i,j})^T(\mathbf{q}_{UEN}^{B,i,j}) - 1 - \epsilon_3 \leq 0 \\ -2(\mathbf{q}_{UEN,k}^{B,i,j})^T(\mathbf{q}_{UEN}^{B,i,j}) + (\mathbf{q}_{UEN,k}^{B,i,j})^T(\mathbf{q}_{UEN,k}^{B,i,j}) + 1 - \epsilon_4 \leq 0 \\ -\epsilon_3 \leq 0 \\ -\epsilon_4 \leq 0 \end{matrix}, \qquad \begin{matrix} i = 1, \dots, n \\ j = 1, \dots, p_i \end{matrix} \tag{39}$$

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

$$\dot{\mathbf{x}} = \frac{t_F - t_0}{2n} \left[\mathbf{f}(t, \mathbf{x}, \mathbf{u}) + \mathbf{C}\nu \right] \triangleq \tilde{\mathbf{f}}(t, \mathbf{x}, \mathbf{u}, \nu, t_F) \tag{40}$$

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,

$$\mathbf{C} = \begin{bmatrix} \mathbf{I}_6 \\ \mathbf{O}_{8\times 6} \end{bmatrix} \tag{41}$$

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

$$\left\| \begin{matrix} \nu_{i,j,1} \\ \vdots \\ \nu_{i,j,n_\nu} \end{matrix} \right\|_2 \leq \mu^{i,j}, \qquad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, p_i \end{matrix} \tag{42}$$

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

$$\mathbf{A} \triangleq \nabla_{\mathbf{x}}^T \tilde{\mathbf{f}}_k, \quad \mathbf{B} \triangleq \nabla_{\mathbf{u}}^T \tilde{\mathbf{f}}_k, \quad \mathbf{E} \triangleq \nabla_{t_F}^T \tilde{\mathbf{f}}_k, \quad \mathbf{G} \triangleq \tilde{\mathbf{f}}_k - \mathbf{A}\mathbf{x}_k - \mathbf{B}\mathbf{u}_k - \mathbf{C}\nu_k - \mathbf{E}t_{F,k} \tag{43}$$

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 [13], we get a final system of equality constraints in the form

$$\mathbf{D}\mathbf{x} - \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{u} - \mathbf{C}\nu - \mathbf{E}t_F = \mathbf{G} \tag{44}$$

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

$$\mathbf{x}^{1,0} = \mathbf{x}_{t_0}, \quad \mathbf{x}^{n,p_n} = \mathbf{x}_{t_F} \tag{45}$$

where Eq. (45) 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

$$J_a = J + w_{fLGR}^T \left[w_\mu \mu + w_p \left(\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 \right) \right] \tag{46}$$

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

$$w_{fLGR} = \left[\left(w_{fLGR}^{1,p_1} \right)^T, \ \ldots \left(w_{fLGR}^{n,p_n} \right)^T \right]^T \tag{47}$$

The weights w_μ and w_p measure the relative importance of the virtual slack variable μ and the exact penalty functions $\epsilon_1, \ldots, \epsilon_4$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 transcribed problem consists in iteratively minimizing Eq. (46) while satisfying Eqs. (36), (37), (39), (42), (44), and (45), 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.

$$\begin{aligned}
&\max_{i,j} \left\| \mathbf{r}_{k+1}^{i,j} - \mathbf{r}_{k}^{i,j} \right\|_2 \leq \delta_{\mathbf{r}} \;\&\\
&\max_{i,j} \left\| \mathbf{v}_{k+1}^{i,j} - \mathbf{v}_{k}^{i,j} \right\|_2 \leq \delta_{\mathbf{v}} \;\&\\
&\max_{i,j} \left| m_{k+1}^{i,j} - m_{k}^{i,j} \right| \leq \delta_{m} \;\& \quad \Longrightarrow \texttt{convergence achieved}\\
&\max_{i,j} \left\| \mathbf{q}_{k+1}^{i,j} - \mathbf{q}_{k}^{i,j} \right\|_2 \leq \delta_{\mathbf{q}} \;\&\\
&\max_{i,j} \left\| \omega_{k+1}^{i,j} - \omega_{k}^{i,j} \right\|_2 \leq \delta_{\omega}
\end{aligned} \tag{48}$$

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.

VI. Numerical Results

A. Rocket Benchmark

The rocket benchmark used in this work is fundamentally taken from the work of Szmuk et Al. [10]. 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., $n = 5$, $p_1 = \cdots = p_n = 10$).

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]
upper Bound on Thrust T_{max}	5 [MU·LU/[TU^2]	maximum gimbal angle δ_{max}	20 [DEG]

We consider two sets of initial conditions: a first, two-dimensional set, is described in Table 2. To test the method

Table 2 Boundary Conditions

Parameter	Value [Unit]
Initial position $\mathbf{r}_0$	$[4,\ 4,\ 0]^T$ [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]

with a second example we modify the third component of the initial position $r_N = 0.5$ to induce a three-dimensional maneuver. Results are depicted in the next section.

B. 2-D Minimum-time trajectory

In this case we impose $c_F = 0$ in Eq. (22). This choice is made in line with the numerical demonstrations found in [10]. The corresponding results are given in Figs. 8 through 12. Fig. 8 shows the body axes of the rocket (in Red-Green-Blue convention, representing the X, Y, and Z axes, respectively), together with the corresponding trajectory of the rocket. The rocket is initially flying towards West, and the algorithm performs a maneuver of 90 deg to ensure a correct pinpoint landing. The 2-D projections of the trajectory are depicted in Fig. 9, where we can see that the maneuver occurs entirely in the East-Up plane since there is no need to have out-of-plane motion.

The NECs imposed in the problem, that is, the norm of the thrust direction unit vector, and the quaternion norm are visible in Figs. 10(a) and 10(b). The NEC related to the thrust direction is satisfied with an accuracy ranging from 10^{-10} to 10^{-5} in all the collocated points. The only exception is the initial one, which is a non-collocated point, and comes purely from the extrapolation of the control history to the initial one. This is a direct consequence of using the fLGR, and is not a consequence of the algorithm itself. The lower part of Fig. 10(a) also shows that the boundaries on thrust are satisfied. Moreover, the *min-max-min-max* control structure, allowed by 6-DoF formulations ([14]) is retrieved, and resembles quite well the same structure obtained in [10].

In Fig. 10(b) we see in the upper part the quaternion profile, embedding the 90 deg maneuver needed to land vertically. In the lower part the corresponding NEC, represented by the norm of the quaternion, is depicted. This NEC is satisfied with an accuracy between 10^{-10} and 10^{-6}. It is possible to easily spot where the less accurate section of the constraint lies, that is, between 1.4 and 2 TU. This is due to the fact that this is the most dynamic region of the trajectory, with large part of the rotation maneuver performed here (as confirmed also by the bottom-right plot of Fig. 11), and a simultaneous change of thrust from *max* to *min* at 1.4 TU, and from *min* to *max* at about 2 TU. Still, even this part of the constraint is well satisfied from a numerical perspective.

Finally, we can see the four constraints of Eqs. (28)-(31) in Fig. 11. The tilt angle profile (top-left) is in line with the original reference, and with the physics of the scenario: the rocket initially keeps its horizontal attitude and uses the thrust to decelerate as fast as possible. Afterwards, the tilt maneuver begins, triggered by the TVC deflection (bottom-left plot). The most of the rotation maneuver, that is, between 1.4 TU and 2.0 TU, is also reflected in the corresponding angular rate profile (bottom-right), having its maximum right after 2.0 TU, to decrease afterwards until 0 DEG/TU at the end of the trajectory. Finally, the glideslope constraint (top-right figure), even if included, is never really active, since the trajectory is entirely contained within the cone having 20 deg of semi-aperture. In the corresponding plot we can see what seems a violation at the end, but this is only due to the fact that both the numerator and the denominator of Eq. (29) tend to 0 when approaching the landing pad. In this figure we also show the upper limit, but this is mainly for clarity of results, since the glideslope constraint is mathematically bounded from above by $\pi/2$.

By having a look at the cost function in Fig. 12 we can see that, starting from iteration 7 no sensitive changes in the cost, (that is, the time of flight) are observed. Moreover, the augmented cost function collapses to the original one, meaning that both the virtual controls and the penalty slacks play no longer a role in the solution, since basically both sets of variables become negligible.

C. 3-D Minimum-fuel trajectory

Next, we apply the same logic to solve the fuel-optimal problem, that is, the case with $c_F = 1$ in Eq. (22). We impose a three-dimensional initial position to enforce a non-zero out-of-plane component of position and velocity. The corresponding results are given in Figs. 13 through 17.

The type of trajectory we obtain for the 3-D fuel optimal problem is similar to the previous one. However, as visible in Figs. 13 and 14 there is now 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. For the NEC constraint in Fig. 15(a) the same reasoning as before can be made. We have an accurate enforcement of the NEC, with the largest violation in this case in the order of $3 \cdot 10^{-5}$. The thrust profile follows the classical *max-min-max* structure, and also in this case we obtain a clean solution with the two switches well defined. The quaternion profile in Fig. 15(b) shows in the top plot on one side again the 90-deg maneuver captured by the elements q_2 and q_3. However, since we are now dealing with a 3-D maneuver, we have non-zero q_0 and q_1 elements, differently from the previous example. The quaternion norm is well satisfied in this case too, 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, despite the presence of out-of-plane components. Finally, all the constraints (Fig. 16) are satisfied too. Since no sharp maneuvers as in the previous case 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 previous example. The generally less dynamic maneuver is also a factor helping the higher

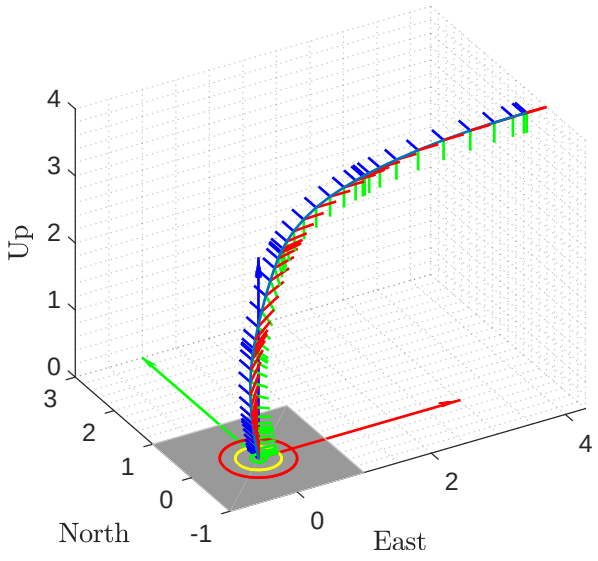


Fig. 8 2-D Minimum-time Problem: body axes and resulting trajectory.

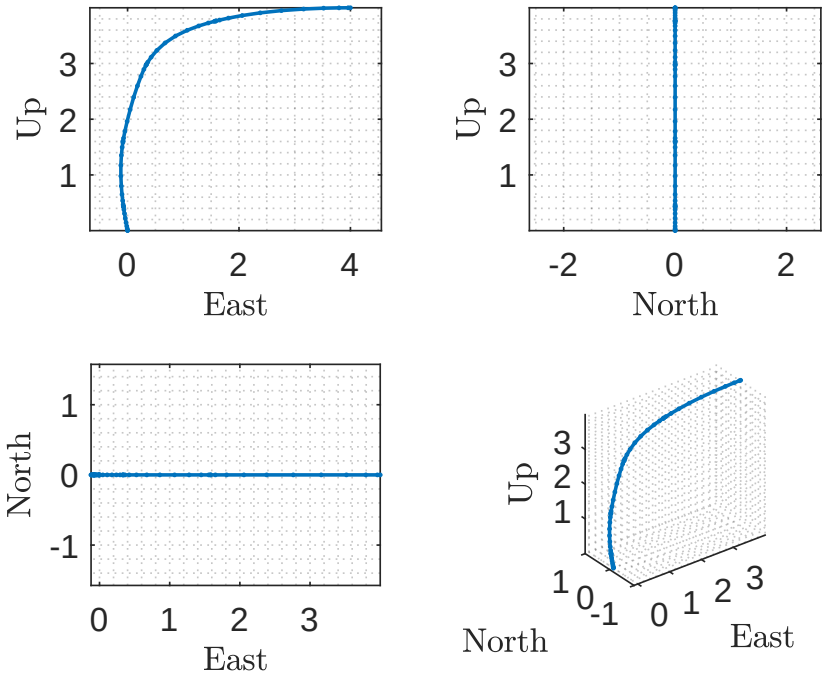
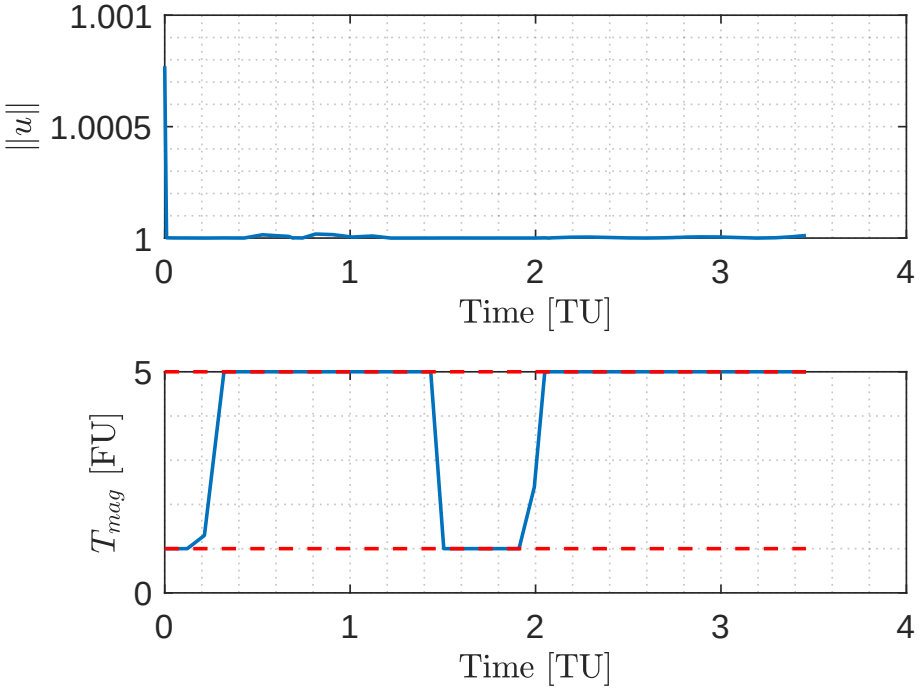
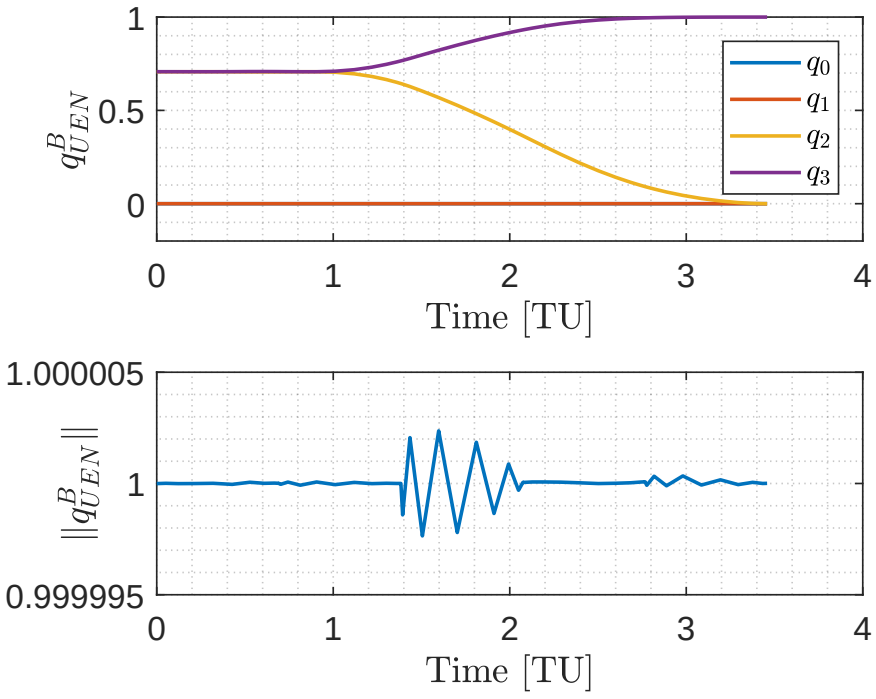


Fig. 9 2-D Minimum-time Problem: trajectory in-plane projections.

accuracy of the quaternion norm achieved in this case. In the last plot in Fig. 17 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), and 2) also in this case the augmented cost function converges to the original cost of the problem. 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.



(a) Thrust unit vector norm and magnitude



(b) Quaternion components and norm

Fig. 10 2-D Minimum-time Problem: thrust and quaternion profiles.

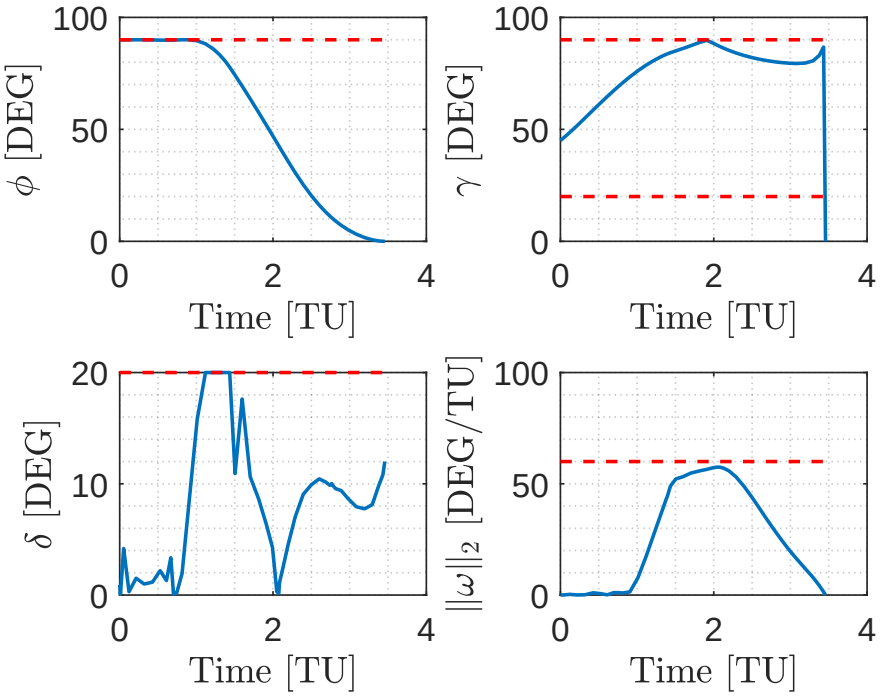


Fig. 11 2-D Minimum-time Problem: constraints.

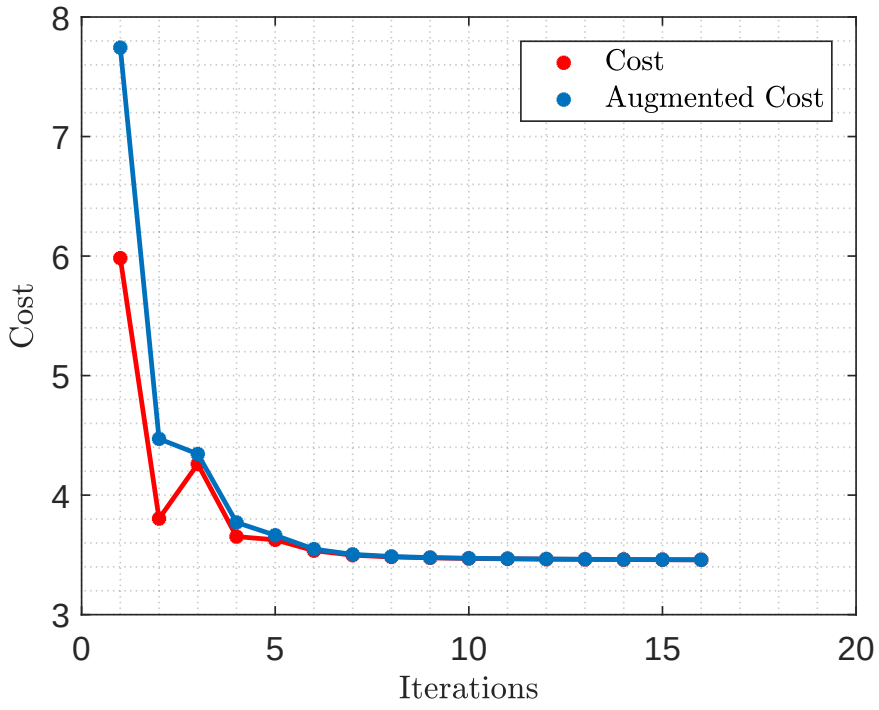


Fig. 12 2-D Minimum-time Problem: cost functions.

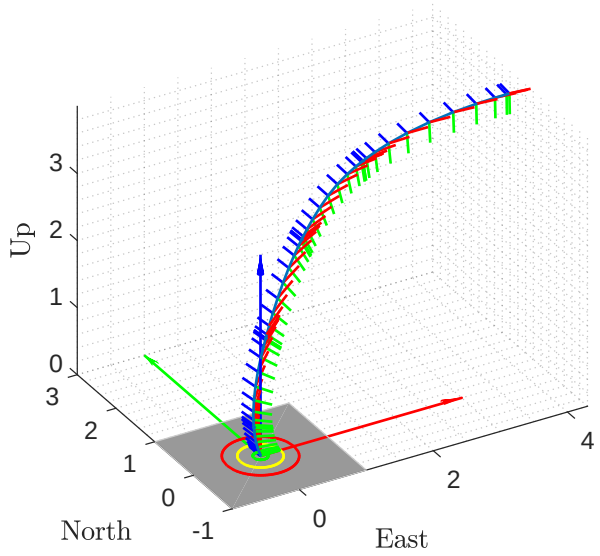


Fig. 13 3-D Minimum-Fuel Problem: body axes and resulting trajectory.

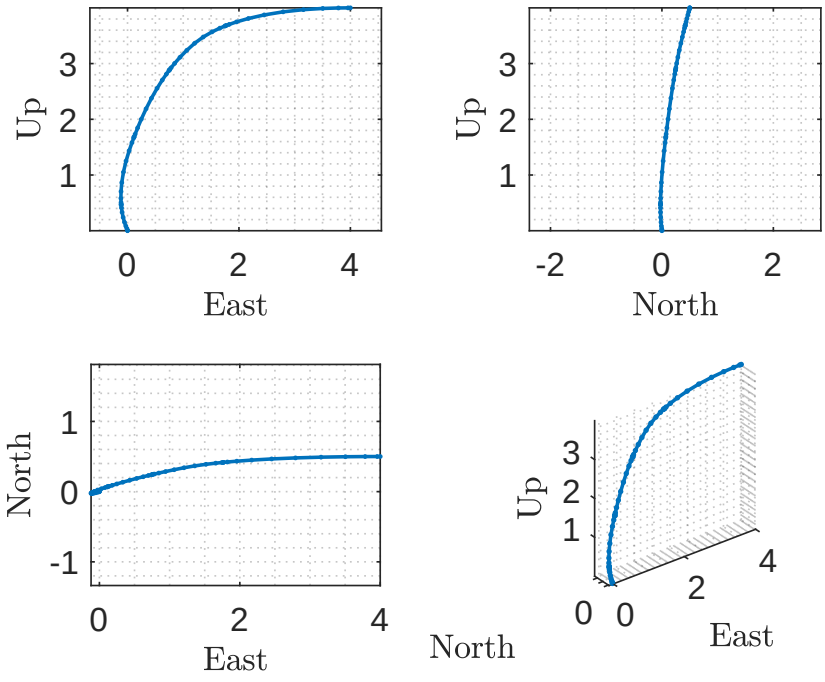
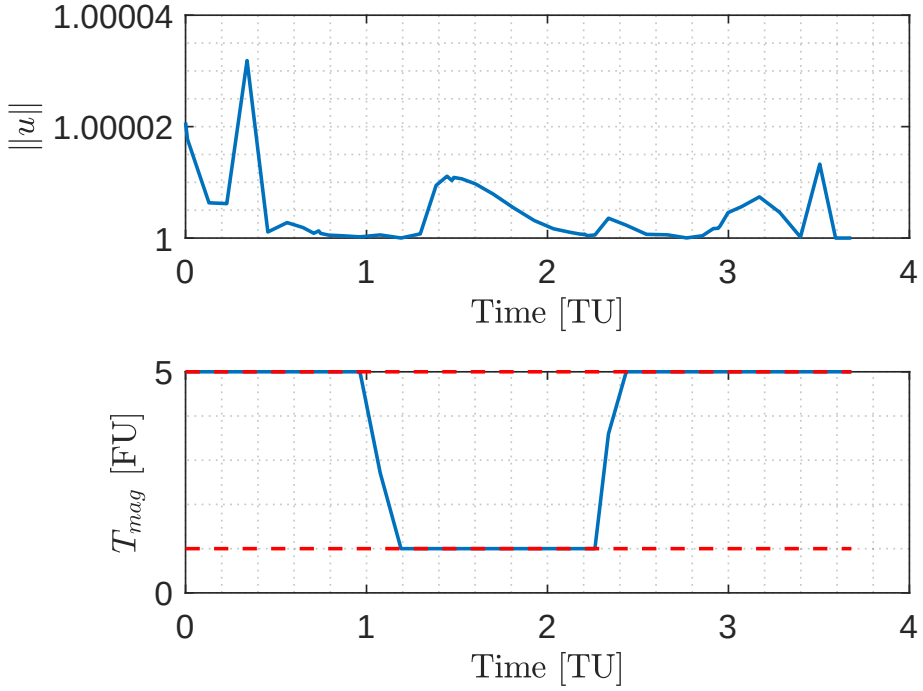
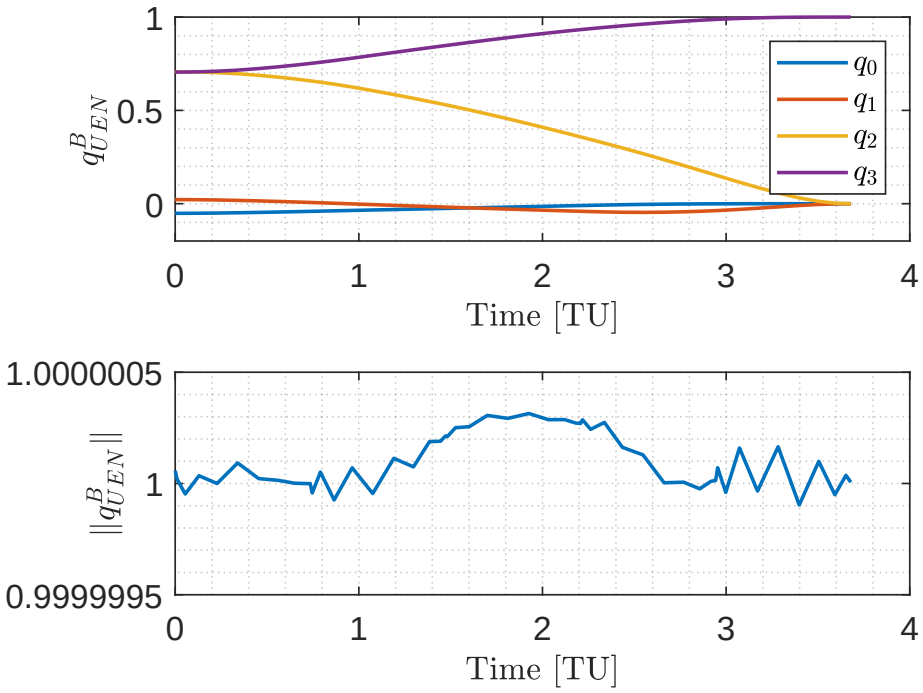


Fig. 14 3-D Minimum-Fuel Problem: in-plane projections.



(a) Thrust unit vector norm and magnitude



(b) Trajectory In-plane projections

Fig. 15 3-D Minimum-Fuel Problem: thrust and quaternion constraints and structure.

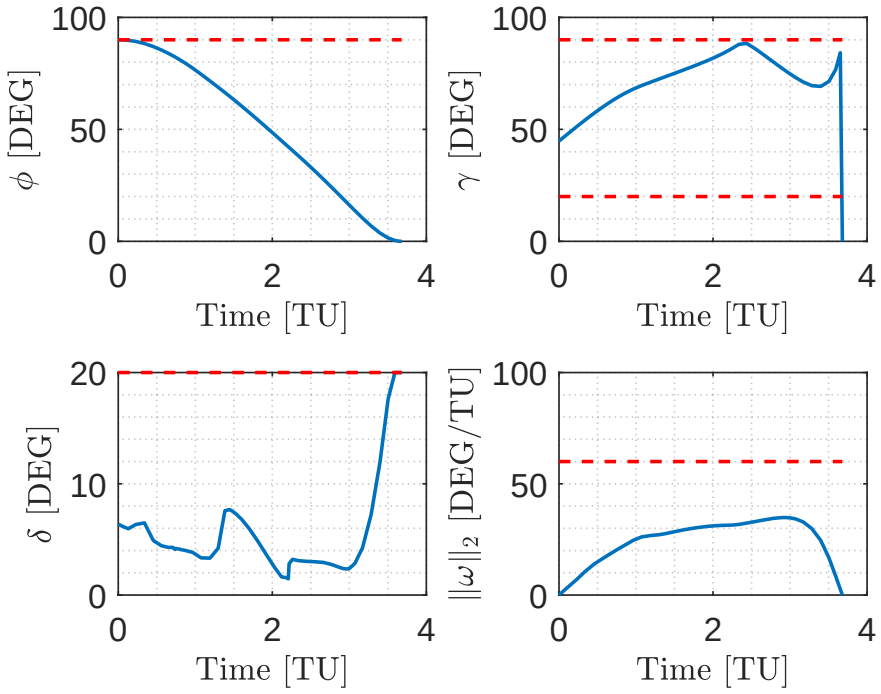


Fig. 16 3-D Minimum-Fuel Problem: constraints.

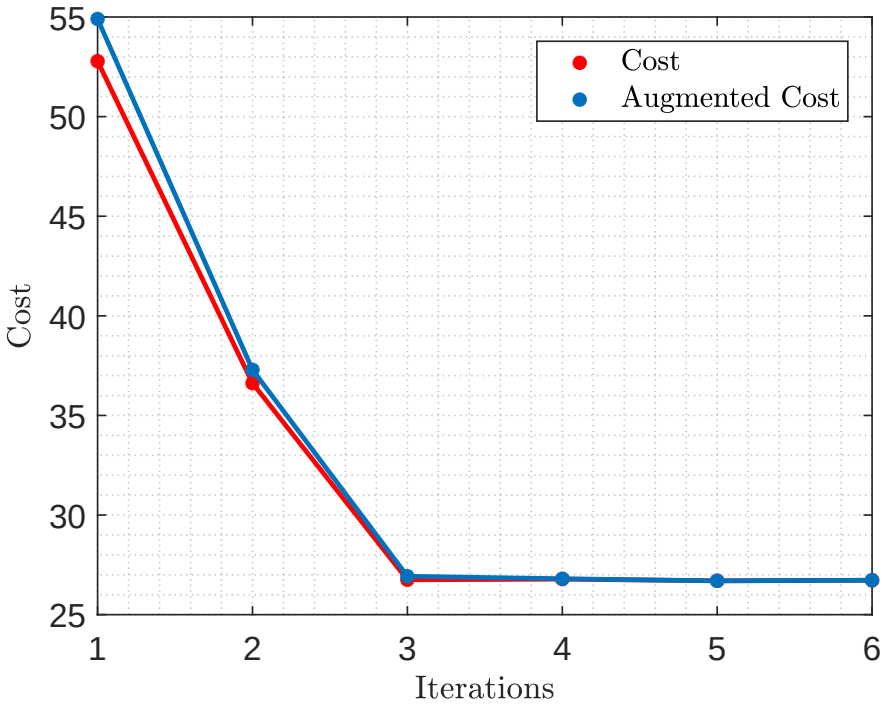


Fig. 17 3-D Minimum-Fuel Problem: cost functions.

VII. Conclusions

In this work we proposed an augmentation of the original Convex-Concave Decomposition and extended its use to continuous variables rather than applying it to a single nonlinear equality constraint. By analyzing the feasibility space of state-of-the-art methods and the proposed Augmented Convex-Concave Decomposition we showed that the infeasibility issues, potentially able to damage the convergence of sequential convex programming methods, are drastically improved, leading to a more robust and interpretable algorithm. The use of the method has been shown for a challenging problem, namely, the Six-Degrees-of-Freedom Powered Landing Problem, both in its minimum-time and minimum-fuel formulations. Results confirmed the validity of the proposed approach, leading to a new tool which can be incorporated and combined in the plethora of existing Sequential Convex programming methods.

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] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] 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.