Generalized hp Pseudospectral Convex Programming for Powered Descent and Landing

Marco Sagliano *

German Aerospace Center, Bremen, Germany, 28359

In this paper a generalized hp method is combined with pseudospectral methods and convex optimization to provide a high-accuracy, real-time capable method able to compute optimal trajectories for planetary powered descent and landing phases of a spacecraft. The benefits of the method are demonstrated for the Mars Science Laboratory study case. Numerical simulations show that the method provides a suitable alternative to the standard convex approaches, and represents a good trade-off between accuracy and computational speed.

Nomenclature

Roman

$\mathbf{A_c}$	=	Continuous LTI dynamic matrix
$\mathbf{A_d}$	=	Discrete LTI dynamic matrix
$\mathbf{B_c}$	=	Continuous LTI control matrix
$\mathbf{B_d}$	=	Discrete LTI control matrix
$\mathbf{D}$	=	Discrete differentiation matrix
$\mathbf{F}$	=	Discrete function vector
$\mathbf{f}, \mathbf{g}$	=	Generic functions
$\mathbf{g}$	=	Gravity vector, [m/s^2]
I_n	=	Identity matrix of dimensions n
i,j,k,m,n,p	=	Non-negative, integer indices
J	=	Cost function
k_t	=	Time conversion factor, [s]
$\tilde{L}_n(\tau)$	=	Legendre polynomial of degree n
m	=	Lander mass, [kg]
$O_{n_1 \times n_2}$	=	Zero matrix of dimensions n_1,n_2
$P_i(\tau)$	=	i$^{\text{th}}$ Lagrange polynomial
$R_n(\tau)$	=	Legendre-Radau polynomial of degree n
$\mathbf{r}$	=	Position vector, [m]
$\mathbf{T}$	=	Thrust vector, [N]
t	=	Generic time, s
$\mathbf{v}$	=	Velocity vector, [m/s]
$\mathbf{x_c}(t)$, $\mathbf{u_c}(t)$	=	Generic continuous state and control
$\mathbf{X}$, $\mathbf{U}$	=	Discrete state and control vector
$\mathbf{x}(t)$, $\mathbf{u}(t)$	=	Generic state and control
z	=	Logarithm of lander's mass

Greek

α	=	Thruster parameter, [s/m]
Γ, σ	=	slack variables, [N, m/s^2]
Φ	=	Mayer term
Ψ	=	Lagrange term
ρ	=	Thrust limit, [N]
τ	=	Pseudospectral time
ω	=	Radau quadrature weights

Operators and subscripts

$\dot{(\cdot)}$	=	First time derivative, [$\cdot$ /s]
$(\cdot)_0$	=	first element of variable $(\cdot)$, [$\cdot$]
$(\cdot)_f$	=	last element of variable $(\cdot)$, [$\cdot$]
$(\cdot)_l$	=	Lower limit, [$\cdot$]
$(\cdot)_{max}$	=	Maximum limit, [$\cdot$]
$(\cdot)(t_0)$	=	variable $(\cdot)$ evaluated at t_0, [$\cdot$]
$(\cdot)(t_f)$	=	variable $(\cdot)$ evaluated at t_f, [$\cdot$]
$(\cdot)_u$	=	Upper limit, [$\cdot$]
$(\cdot)_x$	=	x component of vector $(\cdot)$, [$\cdot$]
$(\cdot)_y$	=	y component of vector $(\cdot)$, [$\cdot$]
$(\cdot)_z$	=	z component of vector $(\cdot)$, [$\cdot$]

I. Introduction

In the last years SpaceX[1] and Blue Origin's New Shepard[2] showed that the reusability is the key for a dramatic reduction of the costs associated with space exploration first and commercial exploitation later on. All the missions dealing with EDL phases have in common the need for a real-time capability to react to off-nominal conditions and uncertainties. For example, the spacecraft has to be able to re-compute its trajectory

*Marco.Sagliano@dlr.de, Research Engineer, GNC Department, Robert Hooke Str. 7, AIAA Member

without violating any constraint, like a given glideslope limit required for proper hazard-avoidance, or to reschedule the landing spot in case a crater or a boulder, not previously visible, is detected.

Several methods were developed over the years. The first family of methods is a heritage of the Apollo era, and is consequently named Apollo guidance,[3] originally used for the Moon landing. In this case an acceleration profile was computed according to the initial and final (desired) position and velocity. This method solves for the desired terminal conditions, but it is not optimal in terms of propellant consumption, nor allows for including further constraints. An alternative algorithm is the gravity turn,[4,5] characterized by having the thrust direction parallel and opposite to the velocity vector during the powered descent phase. A drawback of this approach can be the high final velocity achieved by the spacecraft.[6] This risk can be mitigated by starting the maneuver earlier. However, the correct execution of the algorithm (and therefore the achievement of the desired final conditions) depends on the initial states, and therefore, requires further modifications to be used. This was the case for the Viking missions.[7] The powered descent algorithm was in this case based on the combination of the gravity-turn technique with two altitude-velocity profiles, employed to generate an interpolated solution for any initial and final conditions experienced during the descent.

A paradigm shift was experienced with the development of convex optimization,[8] a class of methods which allow to obtain in real-time optimal solutions for all those problems satisfying some criteria (that is, for all those problems which are subject to convex constraints). In the field of Entry, Descent, and Landing (EDL) applications a breakthrough was represented by the development of the lossless convexification for the Mars powered descent.[9] The algorithm optimizes the consumption of propellant mass, and allows for the inclusion of further constraints, such as the avoidance of non-physical sub-surface trajectories and glideslope limits during the descent.

An alternative approach has arisen with the development of pseudospectral optimal control, a class of methods particularly efficient for a wide range of non-convex problems, including the powered descent guidance problem.[10] They use non-uniform grids, leading to smoother results, and a small number of nodes required to compute a valid solution.[11,12] The resulting discretized nonlinear programming (NLP) problem can be therefore solved with one of the well-known off-the-shelf NLP packages, such as SNOPT[13] or IPOPT.[14] However these methods cannot in general solve the NLP problem in polynomial time, making harder their direct use in real-time. Moreover, these algorithms compute only local optima, and for complex problems they might require a good initial guess.

A first step towards the hybridization of pseudospectral methods and convex optimization can be already found in the pioneering work of Acikmese et Al.[15] However, Chebyshev polynomials were only used for interpolating the controls. This implies that neither the properties associated with the use of non-uniform distributions of nodes, nor the dedicated differential and integral operators were exploited. A full pseudospectral-convex hybridization formulation, based on flipped Radau pseudospectral method and Lobatto pseudospectral method has been proposed by Sagliano.[16] In this case the accuracy of the solutions, based on a full exploitation of the properties of orthogonal polynomials led to very accurate results, but for larger number of nodes the CPU time might be significantly larger than standard methods. This is due to the structure of the underlying matrices, which are less sparse, leading therefore to larger computation times. In this paper we propose a strategy to mitigate this effect by generalizing the previous pseudospectral-convex optimization in the frame of the broader family of hp schemes. This technique, very popular in the Finite-Element method community,[17] and already implemented in other optimization packages[18] allows for a quasi block-diagonal matrix representing the underlying linear system of equations, and therefore to faster results with limited effects on the accuracy of the solution.

The remainder of this paper is organized as follows. Sections II and III provide a brief overview on hp pseudospectral methods and convex optimization, respectively. More specifically, the latter refers to a special form of convex optimization, that is, the Second-Order Cone Programming (SOCP). In Sec. IV the problem we focus on, that is, the Mars powered descent problem is described. The new pseudospectral convex optimization framework is presented in Sec. V, while numerical simulations showing the benefits of the proposed techniques are the subject of Sec. VI. Finally, Sec. VII presents some conclusions about this work.

II. Pseudospectral Methods

A. Optimal Control Problem

There are several approaches for the generation of reference trajectories. Some methods exploit the structure of the specific problem we deal with. Often, they require simplifications to make the problem mathematically tractable, and therefore generate solutions valid under given hypotheses. A different approach, which has become a standard method, and benefits from the development of the computational capabilities of modern CPUs, is the representation of the trajectory generation problem as an optimal-control problem. This means that we are looking for solutions minimizing (or maximizing) a given criterion, and satisfying at the same time several constraints, which can be differential (i.e., the equations of motion of a spacecraft) and / or algebraic (e.g., the maximum heat-flux that a vehicle can tolerate during the atmospheric entry). The standard form for representing optimal-control problems is the so-called Bolza problem. Given a state vector $\mathbf{x}(t) \in \mathbb{R}^{n_s}$, a control vector $\mathbf{u}(t) \in \mathbb{R}^{n_c}$, the scalar functions $\Phi(t, \mathbf{x}, \mathbf{u})$ and $\Psi(t, \mathbf{x}, \mathbf{u})$, and the vector $\mathbf{g}(t, \mathbf{x}, \mathbf{u}) \in \mathbb{R}^{n_g}$ we can formulate the problem as follows:

$$\text{minimize} \quad J = \Phi\left[t_f, \mathbf{x}\left(t_f\right), \mathbf{u}\left(t_f\right)\right] + \int_{t_0}^{t_f} \Psi\left[\mathbf{x}(t), \mathbf{u}(t)\right] dt \tag{1}$$

subject to the differential equations

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

and to the path constraints

$$\mathbf{g}_L \leq \mathbf{g}\left(t, \mathbf{x}, \mathbf{u}\right) \leq \mathbf{g}_U \tag{3}$$

The first term in the cost function of Eq. (1) takes the name of *Mayer* term (or terminal cost), and represents punctual constraints (e.g., the minimization of a distance according to a given metric), while the argument of the integral is called the *Lagrange* term (also known as running cost) and is used to maximize or minimize variables over the entire mission (e.g., the heat load obtained by integrating the heat-flux over time). The inequalities in Eq. (3) are meant as component-wise. Note that although not specifically expressed, we always refer to autonomous systems of differential equations. Therefore the time dependency in Eq. (2) is never explicit. Since we deal with physical systems, the problem has usually bounded states and controls, that is, $\mathbf{x}(t)$ and $\mathbf{u}(t)$ are compact in $\mathbb{R}^{n_s}$ and $\mathbb{R}^{n_c}$, respectively:

$$\mathbf{x}_L \leq \mathbf{x}(t) \leq \mathbf{x}_U \tag{4}$$

$$\mathbf{u}_L \leq \mathbf{u}(t) \leq \mathbf{u}_U \tag{5}$$

Moreover, initial and final conditions might be constrained as well.

$$\mathbf{x}_0 = \mathbf{x}(t_0) \tag{6}$$

$$\mathbf{x}_f = \mathbf{x}(t_f) \tag{7}$$

Equations (1)-(7) represent a generic continuous optimal control problem. In the next section we will see how this type of Optimal-Control Problem (OCP) can be transcribed by using hp pseudospectral methods.

B. Properties of Pseudospectral Methods

Numerical methods for solving OCPs are divided into two major classes, namely, indirect methods and direct methods. Indirect methods are based on the Pontryagin Maximum Principle, which leads to a multiple-point boundary-value problem. Direct methods, instead, consist in the proper discretization (or *transcription*) of the OCP, having as a result a finite-dimensional NLP problem. Pseudospectral methods represent a particular area of interest in the frame of the wider class of direct methods. Examples of tools implementing pseudospectral methods include DIDO,[19] GPOPS,[20] and SPARTAN.[10] For several pseudospectral methods the following properties are valid:

- "Spectral" (i.e., quasi-exponential) convergence of the NLP solution to the OCP solution when the number of nodes employed is increased (and the problem is smooth)
- Runge phenomenon is avoided

- Sparse structure of the associated NLP problem
- Mapping between the discrete costates of the associated NLP and the continuous costates of the Optimal Control Problem in virtue of the Pseudospectral Covector Mapping Theorem.[21]

The transcription process does not only involve the choice of the discrete nodes, but also determines the discrete differential and integral operators needed to solve the associated OCP. Therefore, *transcription* is a more general process than *discretization*. The minimum fundamental steps of a transcription are the following:

- domain discretization
- discrete to continuous conversion of states and / or controls
- characterization of differential and integral operators

Among the families of pseudospectral (PS) methods a specific one was considered for this work: the hp flipped Radau Pseudospectral method (or fRPm). It is worth saying that this is not the only possible choice, as other sets of nodes, like Gauss,[22] Chebyshev[23] or Lobatto[12] exist. The reason behind this choice is that the fRPm allows for a natural and straightforward definition of the initial conditions of the problem, and shows a smoother convergence of the costates with respect to other methods.[12] Moreover, in its hp form we will see that it provides a compact way to transcribe the problem. Therefore, it is useful to have a look at this method, and at its transcription. This will be the purpose of the two next subsections.

C. Flipped Radau Pseudospectral Method

Flipped Radau Pseudospectral Method is an asymmetric pseudospectral method, whose nodes are the roots of the flipped Legendre-Radau polynomial, defined as the combination of the Legendre polynomial of order n and $n-1$ with coefficient equal to 1 and -1 respectively.

$$R_n(\tau) = \widetilde{L}_n(\tau) - \widetilde{L}_{n-1}(\tau) \quad \tau \in [-1, 1] \tag{8}$$

An example of roots associated with the Legendre-Radau polynomial of order 10 is depicted in Fig. 1(a), together with the corresponding polynomial.

Remark 1 Note that the $R_n(-1)$ is not a root of the underlying polynomial, therefore it is not a collocation point, although it is required for the evaluation of the polynomial. This is due to the fact that only over the *left-open*, *right-closed* interval $(-1, +1]$ these polynomials are orthogonal.

This discrete representation of the domain is useful to reconstruct continuous approximations of the functions $x(t)$ as:

$$x(t) \cong \sum_{i=0}^{n} X_i P_i(t), \qquad P_i(t) = \prod_{\substack{k=0 \\ k \neq i}}^{n} \frac{t - t_k}{t_i - t_k} \tag{9}$$

A property of Radau pseudospectral methods is that only one of the two extremal points is collocated. This difference will affect the differential operators we are going to introduce in the next section, as we will see, and has consequences on the proposed hp pseudospectral convex method too. This aspect will be further explained in Sec. V. An example of the approximation obtained via Eq. (9) is depicted in Fig. 1(b), where the function $1/(1 + 25\tau^2)$ is reconstructed by using 25 nodes. It is possible to see that the original function is approximated very well with this set of discrete nodes.

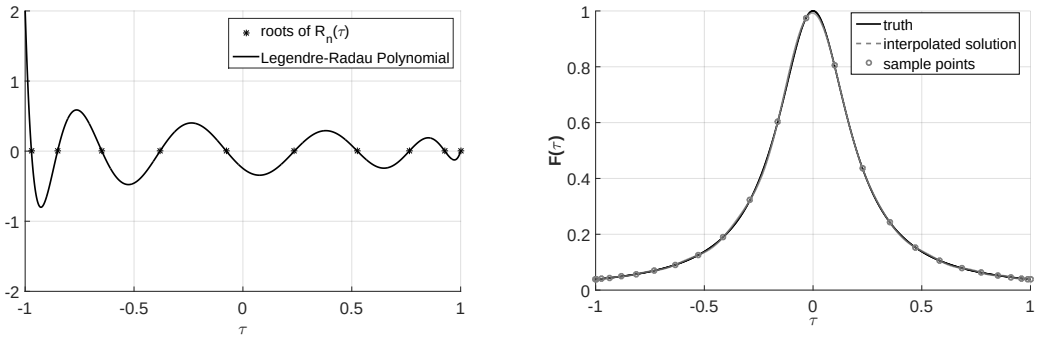
Remark 2 Note that the approximation becomes more accurate when the number of nodes is increased. This is the opposite behavior observed when equi-spaced nodes, which suffer from the aforementioned *Runge Phenomenon*, are employed.

Once that the domain has been discretized, and the discrete-to-continuous conversion of states has been defined, the corresponding differential operator needs to be characterized. This is required for the proper representation of the left-hand side of Eq. (2). The differential operator will be in the form

$$\dot{\mathbf{X}}_i \cong \mathbf{D} \cdot \mathbf{X}, \; i = 1, ...n \tag{10}$$

and the dynamics defined in Eq. (2) will be replaced by

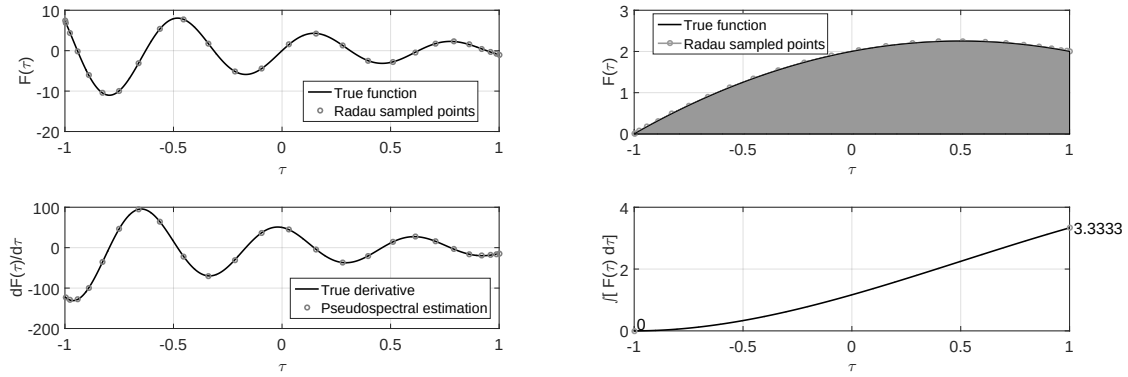
$$\mathbf{D} \cdot \mathbf{X} = \frac{t_f - t_0}{2} \mathbf{f}(t_i, \mathbf{X}_i, \mathbf{U}_i) \tag{11}$$



(a) Discrete domain

(b) Continuous approximation of a function

Figure 1. Transcription steps with fRPm: (a) domain discretization, (b) continuous reconstruction of functions.



(a) Differential operator example

(b) Integral operator example

Figure 2. Transcription steps with fRPm: (a) application of differential operator, (b) application of integral operator.

where t_0 and t_f are the initial and final time, and the term $\frac{t_f - t_0}{2}$ is a scale factor related to the transformation between the physical time domain t, and the pseudospectral time domain $\tau \in [-1, 1]$, given by the following affine transformations:

$$t = \frac{t_f - t_0}{2}\tau + \frac{t_f + t_0}{2} \tag{12}$$

$$\tau = \frac{2}{t_f - t_0}t - \frac{t_f + t_0}{t_f - t_0} \tag{13}$$

The matrix $\mathbf{D}$ has dimensions $[n \times (n+1)]$. Once again, this is due to the fact that the states are defined for $n+1$ discrete points, while the controls $\mathbf{U}$ and the derivatives of the states $\mathbf{f}(t, \mathbf{X}, \mathbf{U})$ are defined in the n collocation points. This means that the initial state $\mathbf{X}_0$ is an input and not an output of the optimization process, and it is thus assumed to be known. If we look at Eq. (9), and we take the derivative w.r.t. time, we get

$$\dot{\mathbf{x}}(t) \cong \frac{d}{dt}\sum_{i=0}^{n} \mathbf{X_i} P_i(t) = \sum_{i=0}^{n} \mathbf{X_i} \frac{d}{dt} P_i(t) \tag{14}$$

as the nodal points are time-independent. These derivatives can be efficiently computed with the Barycentric Lagrange Interpolation.[24] An example of the differential operator for the two methods is depicted in Fig. 2(a), where $\mathbf{D}$ is used to approximate the derivative of the continuous test function $F(\tau) = Ae^{-\tau}\sin(\omega\tau)$,

with $A = 5$, $\omega = 10$, sampled in 25 collocation nodes. It can be seen that the polynomial approximation fits the true derivative of the function very well.

In addition to the differential operator, we need an integral operator. This operator is required as the cost function in Eq. (1) may contain the Lagrange term, which needs a proper discretization. In this case the Gauss quadrature formula is used.[25] For the fRPm the approach consists of replacing the continuous integral with the discrete sum given by:

$$\int_{t_0}^{t_f} \Psi\left[t, \mathbf{x}(t), \mathbf{u}(t)\right] dt = \frac{t_f - t_0}{2} \sum_{i=1}^{n} w_i \Psi\left[\mathbf{X}_i, \mathbf{U}_i\right] \tag{15}$$

It can be shown that Eq. (15) yields exact results for polynomials of order at most equal to $2n - 2$.[12] Once again, the presence of the term $\frac{t_f - t_0}{2}$ is a consequence of the mapping between pseudospectral and physical time domains described in Eq. (12) and (13). The weights w_i can be computed as

$$w = flip(\tilde{w}) \tag{16}$$

$$\tilde{w}_j = \begin{cases} \dfrac{2}{n^2}, & j = 1 \\ \dfrac{(1 - \tau_j)}{n^2 \tilde{L}_n(\tau_j)^2}, & j = [2, ...n] \end{cases} \tag{17}$$

where the operator *flip* simply multiplies its argument by a factor equal to -1, and sorts the results in increasing order. To give a practical example the integral of the test function $F(\tau) = 2\tau + 2 - \tau^2$ has been computed. Results are then compared with the analytical integral, and with the trapezoidal rule (Fig. 2(b)) applied using the same nodes. Numerically, we get exactly the analytical result, that is 3.3333 when fRPm is employed, while the application of the trapezoidal rule gives 3.3298, confirming the validity of the quadrature formula applied to the f-RPm points. Note that when n equi-spaced nodes are used the trapezoidal rule gives better results (3.3310), but still inferior to the pseudospectral ones.

D. Hp Pseudospectral Methods

The method introduced in the previous section uses an implicitly-defined unique domain. In other words, the original time domain of our problem $[t_0,\ t_f]$ is mapped against the so-called *pseudospectral time* $[-1,\ 1]$. To compute a more accurate solution the number of nodes collocated in this interval is increased. This means that we are increasing the degree of the polynomials used to approximate the continuous variables of the original problem, for instance, by using p nodes. Since p is the only parameter we are controlling, we can state that the global flipped Pseudospectral method is a *p-method*. However, one can think to break the time domain in sub-domains, and to *locally* collocate the equations of motion in each of these segments. In this case there will be two parameters to define, that is, the number of segments h, and the number of nodes per segment p. This is the idea behind the so-called *hp methods*, which are very popular in finite-element methods[17] as well as in computational fluid dynamics.[26] Hp methods were successfully applied to optimal control, by using adaptive schemes, which allowed for automatic mesh refinements aimed at improving the accuracy of the solutions.[20,27] Note that p can be a vector $p \in \mathbb{R}^h$, where each element $p_1,\ p_2,\ \ldots,\ p_h$ identifies the degree of the polynomial used for the segment 1, the segment 2, ..., and the segment h. However, throughout this paper since no automatic mesh-refinement is involved, we will use a constant value p for all the segments.

A double notation is now introduced to differentiate the segments and the nodes within a segment. Subscripts i refer to the i^{th} node in a given segment, while superscripts j define the j^{th} segment. Therefore,

$$\begin{matrix} \mathbf{X}_i^j & \mathbf{U}_i^j & \mathbf{G}_i^j, \end{matrix} \quad \begin{matrix} j \in [1, \ldots, n] \\ i \in [1, \ldots, p] \end{matrix} \tag{18}$$

identify the generic state, control and constraint at the i^{th} node of the j^{th} segment. Accordingly, we will have n time segments, defined as

$$[t_0^j,\ t_f^j], \quad j = [1, \ldots, n] \tag{19}$$

and the generic element of the overall time vector is t_i^j. With this notation we can define the transcription according to the hp flipped Radau pseudospectral method:

Minimize (or maximize) the cost function J, for $i = 1, \ldots, p$, and $j = 1, \ldots, n$.

$$J = \Phi\left[\mathbf{X}_p^n, \mathbf{U}_p^n\right] + \frac{t_p^n - t_0^1}{2n} \sum_{j=1}^{n} \sum_{i=1}^{p} w_i \Psi\left[\mathbf{X}_i^j, \mathbf{U}_i^j\right] \tag{20}$$

subject to the nonlinear algebraic constraints

$$\mathbf{F}_i^j = \mathbf{D}^j \cdot \left[\begin{array}{cccc} \mathbf{X}_0^j & \mathbf{X}_1^j & \ldots & \mathbf{X}_p^j \end{array}\right] - \frac{t_p^n - t_0^1}{2n} \mathbf{f}\left(\mathbf{X}_i^j, \mathbf{U}_i^j\right) = \mathbf{0} \tag{21}$$

and to the path constraints

$$\mathbf{g}_L \leq \mathbf{G}\left(\mathbf{X}_i^j, \mathbf{U}_i^j\right) \leq \mathbf{g}_U \tag{22}$$

Note that since we chose to have the same number of nodes p in each segment, the matrix $\mathbf{D}$ is the same for all the segments; therefore

$$\mathbf{D}^j = \mathbf{D} \tag{23}$$

holds, and the index j for can be dropped. The discrete states and the controls are bounded, as in the continuous formulation.

$$\mathbf{x}_L \leq \mathbf{X}_i^j \leq \mathbf{x}_U \tag{24}$$

$$\mathbf{u}_L \leq \mathbf{U}_i^j \leq \mathbf{u}_U \tag{25}$$

Moreover, the so-called *linking conditions* are required, that is

$$\begin{array}{c} t_p^{j-1} = t_0^j \\ \mathbf{X}_p^{j-1} = \mathbf{X}_0^j \end{array}, \quad j = [2, \ldots, n] \tag{26}$$

These equations provide the pseudospectral hp algorithm, which will be combined with convex optimization, briefly summarized in the next section.

III. Convex Optimization

Over the last thirty years several researchers focused on the development of convex optimization theory.[8,28] They demonstrated that for a large class of problems the key-property is not the linearity of the system, but the convexity. In this case, the problem can be solved in real-time, and if the problem is feasible, the computed solution is the global optimum.

A. Convex Programming

In general a convex optimization problem is defined as follows:

$$\text{minimize} \quad J = f_0(x) \tag{27}$$

subject to

$$f_i(x) \leq a_i, \qquad i = 1, \ldots, m \tag{28}$$

where $x \in \mathbb{R}^n$ represents the vector of variables to be determined. The functions f_i, with $i = 0, \ldots, m$, are convex functions, which means that they satisfy the following relationship.

$$f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y), \qquad i = 0, \ldots, m, \quad \forall \alpha, \beta \geq 0 : \alpha + \beta = 1 \tag{29}$$

The previous expression suggests one of the properties of convex problems, that is, they generalize the notion of linearity of a function, leading to the notion of convexity, which has the equality as special case instead of the inequality in Eq. (29). Further details and exhaustive explanations can be found in the works of Boyd, Vandenberghe, Ben Tal and Nemirovski.[8,28]

The following properties characterize convex optimization:

- A large number of problems can be reformulated in convex form
- There are efficient methods to solve convex problems (e.g., primal-dual interior point methods), such that it can be considered more and more a mature technology
- This class of methods does not require an initial guess (a problem which affects many problems when NLP solvers are employed)
- If a solution to the problem exists, it is the global optimum.

While the category of convex optimization is still quite large, and includes several subfields (e.g., Semidefinite programming, Quadratically constrained quadratic programming, and so on), we will instead focus on a specific form of convex optimization, that is, the so-called Second-order Cone Programming (or SOCP). This specific subclass of methods will be briefly described in the next section, whereas more extensive and rigorous descriptions of the theoretical properties and the potential applications of this technology can be found in literature.[8,28,29]

B. Second-Order Cone Programming

An interesting subcategory of convex optimization is represented by Second-Order Cone Programming. This definition encloses all the problems which can be formulated as follows:

$$\text{minimize} \quad c_0^T x \tag{30}$$

subject to

$$\begin{gathered} G_i x \leq h_i,\ i = 1, \ldots, l \\ A_0 x = b_0 \\ \|A_i x + b_i\|_2 \leq c_i^T x + d_i,\ i = 1, \ldots, p \end{gathered} \tag{31}$$

with $x \in \mathbb{R}^{n \times 1}$ representing the variables to determine, $c_0 \in \mathbb{R}^{n \times 1}$ is the vector defining the cost function, whereas $G_i \in \mathbb{R}^{l_i \times n}$ and $h_l \in \mathbb{R}^{l_i \times 1}$ represent (with l_i that might vary for every i) a set of component-wise inequalities. $A_0 \in \mathbb{R}^{m \times n}$ and $b_0 \in \mathbb{R}^{m \times 1}$ describe the linear system of m equations that the solution has to satisfy. The terms $A_i \in \mathbb{R}^{m_i \times n}$, $b_i \in \mathbb{R}^{m_i \times 1}$, $c_i \in \mathbb{R}^{n \times 1}$ and $d_i \in \mathbb{R}$ describe a conic constraint of order $m_i + 1$. These constraints imply that, given the affine transformations

$$\begin{aligned} t &= c_i^T x + d_i \\ y &= A_i x + b_i,\ i = 1, \ldots, p \end{aligned} \tag{32}$$

the solution will always be contained within the volume of each of the p m_i-dimensional cones. An example for $m_i = 2$ is depicted in Fig. 3.

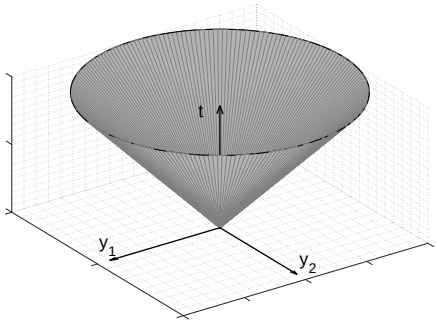


Figure 3. Example of 3-D cone. The volume of the cone satisfies the condition $\|y\|_2 \leq t$.

Among the others, linear programming problems, or quadratically constrained problems can be reformulated as cone programming problems. Moreover, they can efficiently be solved by using primal-dual interior point methods,[30] and several solvers, such as ECOS,[31] are available. These aspects make the SOCP technology appealing for several applications, including the one used as example in this work.

IV. Mars Powered Descent

In 2012 NASA's rover Curiosity successfully landed on the martian surface.[32] One of the most challenging parts of the famous 7 *minutes of terror*[33] was the descent phase, where the retrorockets were used to counteract Martian gravity and ensure the proper conditions for a soft touchdown. An elegant formulation of this problem was proposed by Acimese and Ploen.[9] Specifically, the optimal-control problem can be stated as follows. We are interested in maximizing the final mass of the lander

$$\text{maximize} \quad J = m(t_f) \tag{33}$$

subject to the following set of equations:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \frac{\mathbf{T}}{m} + \mathbf{g} \\ \dot{m} &= -\alpha \left\| \mathbf{T} \right\| \end{aligned} \tag{34}$$

$\mathbf{r} \in \mathbb{R}^3$ is the position vector, and $\mathbf{v} \in \mathbb{R}^3$ represents the velocity vector, both expressed in a surface-fixed reference frame, depicted in Fig. 4. The Martian gravity vector is defined as $\mathbf{g} = [0,\ 0,\ -3.7114]$ m/s^2.

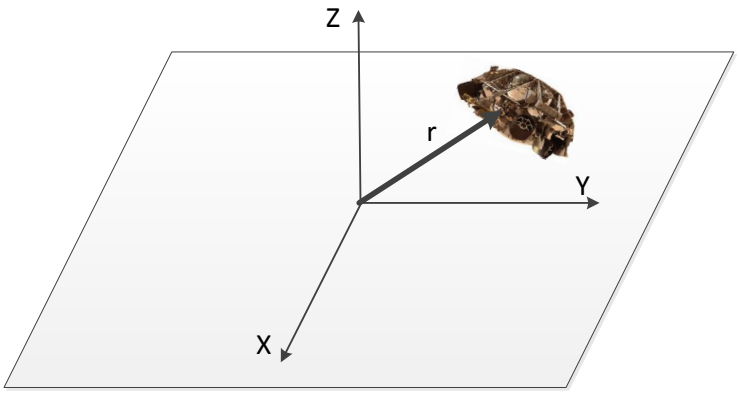


Figure 4. Surface-fixed reference frame.

Note that assuming a constant, vertical gravity vector is a valid assumption given the altitude of the lander at this stage of the mission. Moreover, the velocities are much smaller than the ones experienced during the entry and initial descent phase, and therefore the aerodynamic accelerations can be neglected in this context. $\mathbf{T} \in \mathbb{R}^3$ is the net thrust vector in Newton, and is the control of the system. m is the mass of the lander, initially equal to 1905 kg. The time of flight is assigned and equal to 81 s. The coefficient α in the last of Eq. (34) includes parameters of the thrusters' system, and is computed as

$$\alpha = \frac{1}{I_{sp} g_e \cos\phi} \tag{35}$$

where $I_{sp} = 225$ s is the specific impulse of the thrusters, and $g_e = 9.807$ m/s^2 is the Earth's gravitational constant. The rover is equipped with $n = 6$ thrusters, having a cant angle $\phi = 27$ degrees and able to provide a thrust T_i along each of the axes. The relationship between T_i and the thrust level $\widehat{T}_i$ is

$$T_i = T_{max} n \widehat{T}_i \cos\phi,\ i = x, y, z \tag{36}$$

with T_{max} equal to 3.1 kN. Note that $\widehat{T}_i$ obeys the following constraint:

$$\widehat{T}_l \leq \widehat{T}_i \leq \widehat{T}_u,\ i = 1, \ldots, 3 \tag{37}$$

with $\widehat{T}_l = 0.3$ and $\widehat{T}_u = 0.8$. Initial and final positions and velocities are:

$$\begin{aligned} \mathbf{r}(t_0) &= \begin{bmatrix} 2000 \\ 0 \\ 1500 \end{bmatrix} \text{ m}, \quad \mathbf{r}(t_f) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ m} \\ \mathbf{v}(t_0) &= \begin{bmatrix} 100 \\ 0 \\ -75 \end{bmatrix} \text{ m/s}, \mathbf{v}(t_f) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ m/s} \end{aligned} \tag{38}$$

There are two constraints that need to be included in the formulation of the problem. The first is the sub-surface constraint,

$$r_z(t) > 0,\ \forall t \in [t_0, t_f) \tag{39}$$

which simply means that the lander's trajectory lies above the surface of the planet. This constraint is often replaced by a stricter one, that is, the glideslope constraint:

$$\tan^{-1}\left[\frac{r_z(t)}{\sqrt{r_x^2(t) + r_y^2(t)}}\right] \geq \widetilde{\theta}_{alt} = 4 \text{ deg} \tag{40}$$

This constraint ensures that during its descent the lander moves within a cone having a semi-angle equal to $90 - \widetilde{\theta}_{alt}$ degrees, and therefore does not reduce the altitude below a given threshold while reaching the target position. Acikmese and Ploen[9] showed that this non-convex optimal problem can be transformed into an equivalent convex one. Let us define the following variables:

$$\begin{aligned} \mathbf{u} &= \frac{\mathbf{T}}{m} \\ \sigma &= \frac{\Gamma}{m} \\ z &= \log(m) \end{aligned} \tag{41}$$

The scalar variables Γ and σ are introduced to overcome the nonconvexity of the original control set. With these definitions, the problem becomes:

$$\text{minimize} \quad J = \int_{t_0}^{t_f} \sigma(t) dt \tag{42}$$

subject to:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{u} + \mathbf{g} \\ \dot{z} &= -\alpha\sigma \end{aligned} \tag{43}$$

The lossless convexification implies that for the controls the following condition holds:

$$\|\mathbf{u}(t)\| \leq \sigma(t) \tag{44}$$

The change of variables of Eq. (41) implies that the following constraint acting on z has to be satisfied:

$$\rho_l e^{-z(t)} \leq \sigma(t) \leq \rho_u e^{-z(t)} \tag{45}$$

and these limits are approximated with the following second-order Taylor expansion and first-order Taylor expansion for the lower and the upper boundaries:

$$\rho_l e^{-z_l}\left[1 - (z - z_l) + \frac{1}{2}(z - z_l)^2\right] \leq \sigma(t) \leq \rho_u e^{-z_u}\left[1 - (z - z_u)\right] \tag{46}$$

The centers of expansion z_l and z_u can be computed according to

$$\begin{aligned} z_l &= log(m_0 - \alpha\rho_l t) \\ z_u &= log(m_0 - \alpha\rho_u t) \end{aligned} \tag{47}$$

and the terms ρ_l and ρ_u are equal to the minimum and the maximum values of T. Moreover, Eqs. (39) / (40) need to be satisfied too. One of these two last conditions, together with Eq. (43) define the entire convex problem to be solved, characterized by having $n_s = 7$ states, and $n_c = 4$ controls. Full technical details on the lossless convexification can be found in the work by Acikmese et Al.,[9] while further enhancements are covered in Blackmore et Al.,[34] and Szmuk et Al.[35] In the next section we will apply the hp pseudospectral convex optimization algorithm to the original formulation of the problem.

V. Hp Pseudospectral Convex Optimization

In this section we present the hp pseudospectral convex framework for generating real-time capable optimal solutions for the Mars powered descent and landing phase.

A. Hp Flipped Radau Pseudospectral Convex Method

The first step is the determination of the discrete timesteps, and the state vector representing the solution. For each of the n segments we can compute the p roots of the flipped Radau-Legendre polynomials as defined in Eq. (8). Note that in each of the n segments we have $p+1$ discrete nodes. However, the first point is not collocated. Moreover, we have $n-1$ link conditions defined in Eq. (26). Therefore, we want to determine the solution only in np points. These points correspond to the discrete set of pseudospectral times $\tau_i \in [-1,\ 1]$, which can be converted into physical timesteps by using the first of the affine transformations defined by Eq. (12), adapted to the j^{th} segment,

$$t_i^j = t_f^{j-1} + \frac{t_f^j - t_0^j}{2}\tau_i + \frac{t_f^j + t_0^j}{2},\ i = 0, \ldots, p,\ j = 1, \ldots, n, \quad t_f^0 \triangleq 0 \tag{48}$$

The discrete time vector is of course non-uniform in virtue of the nature of the hp pseudospectral transcription. For the states and the controls we propose to use the following vector:

$$\mathbf{X} = \left[\begin{array}{ccccccccccc} \mathbf{r}_1^1 & \mathbf{v}_1^1 & z_1^1 & \mathbf{u}_1^1 & \sigma_1^1 & \ldots & \mathbf{r}_p^n & \mathbf{v}_p^n & z_p^n & \mathbf{u}_p^n & \sigma_p^n \end{array} \right]^T \tag{49}$$

Note that the initial conditions ($\mathbf{r}_0$, $\mathbf{v}_0$, and $\mathbf{z}_0$) and the initial controls ($\mathbf{u}_0$ and σ_0) are excluded from the definition of $\mathbf{X}$, consistently with the fact that the initial node of the fRPm is *not collocated*. The vector $\mathbf{X}$ will have dimensions $[(n_s + n_c)np \times 1]$

Cost function
The vector $\mathbf{c}$ representing the cost function will be a vector having dimensions $np(n_s + n_c) \times 1$. Of these, only np elements, corresponding to the σ_i^j values, are different from zero. Therefore we have

$$c_k = \begin{cases} \dfrac{t_f^n - t_0^1}{2n} w_i, & \begin{array}{l} i = 1, \ldots, p \\ j = 1, \ldots, n \\ k = ij(n_s + n_c) \end{array} \\ 0, & \text{otherwise} \end{cases} \tag{50}$$

where w_i are the Radau quadrature weight defined in Eqs. (16),(17), and t_0^1 and t_f^n are the initial and final times of the entire mission profile, assumed to be known. Note that for each of the segments the first node is non-collocated. However, the same node is the last collocated node of the previous segment, as depicted in Fig. 5. Therefore, its weight will simply correspond to w_p.

Dynamics
If we define the continuous state vector as

$$\mathbf{x_c} = \left[\mathbf{r} \quad \mathbf{v} \quad z \right]^T \tag{51}$$

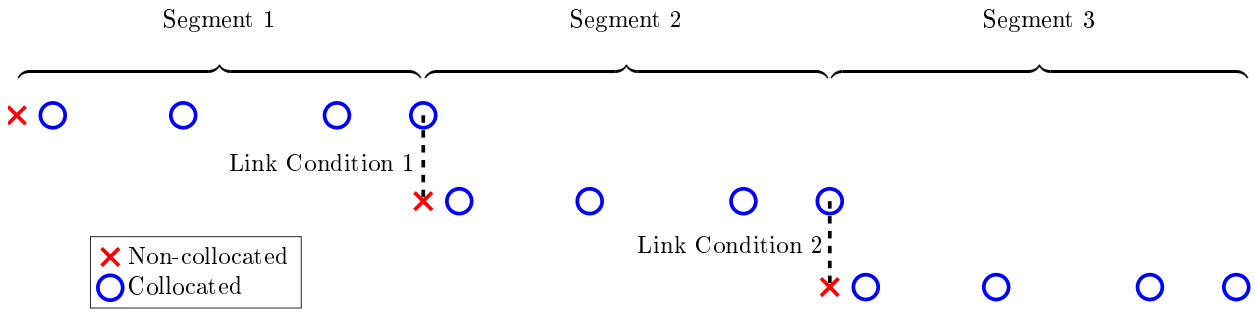


Figure 5. Linking conditions between segments in the hp collocation scheme.

and the control as

$$\mathbf{u_c} = \begin{bmatrix} \mathbf{u} & \sigma \end{bmatrix}^T \tag{52}$$

the dynamics of Eq. (43) has the following state-space representation:

$$\mathbf{A_c} = \begin{bmatrix} O_{3\times3} & I_3 & 0 \\ O_{3\times3} & O_{3\times3} & O_{3\times1} \\ O_{1\times3} & O_{1\times3} & 0 \end{bmatrix}, \qquad \mathbf{B_c} = \begin{bmatrix} O_{3\times3} & 0 \\ O_{3\times3} & 0 \\ O_{1\times3} & -\alpha \end{bmatrix} \tag{53}$$

where $O_{n_1\times n_2}$ and I_{n_3} are the zero matrix of dimensions n_1 and n_2 and the identity matrix of dimensions n_3, respectively. In the standard transcription the matrices $\mathbf{A_c}$ and $\mathbf{B_c}$ were converted into their discrete counterparts $\mathbf{A_d}$ and $\mathbf{B_d}$. These matrices were then used in the discrete scheme for building the linear system defined in Eq. (31). Instead, with pseudospectral convex framework we can skip this transformation, and directly use $\mathbf{A_c}$ and $\mathbf{B_c}$. The reason is the different construction of the linear system of equations. In the standard transcription the system is constructed by exploiting the equation

$$\mathbf{x}(k+1) = \mathbf{A_d}\mathbf{x}(k) + \mathbf{B_d}\mathbf{u}(k) + \mathbf{B_d}\mathbf{g} \tag{54}$$

In our case we build the residuals of the differential equations as

$$\dot{\mathbf{x}}(t) = \mathbf{A_c}\mathbf{x_c}(t) + \mathbf{B_c}\mathbf{u_c}(t) + \mathbf{B_c}\mathbf{g} \tag{55}$$

since $\dot{\mathbf{x}} \cong \mathbf{D}\mathbf{x}$, and keeping in mind Eq. (21) we can write that for each phase j and node i

$$\mathbf{D}\mathbf{x_c}^j(t) - k_t\mathbf{A_c}\mathbf{x_c}_i^j(t) - k_t\mathbf{B_c}\mathbf{u_c}_i^j(t) = k_t\mathbf{B_c}\mathbf{g} \tag{56}$$

holds. This relationship, evaluated in the p nodes within that segment, leads to the following definitions. In each segment ($j = 1, \ldots, n$) we have

$$\mathbf{A_{dyn}^j} = \begin{bmatrix} D_{1,1}I_{n_s} - k_tA_c & -k_tB_c & \ldots & \ldots & D_{1,p}I_{n_s} & O_{n_s\times n_c} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ D_{p,1}I_{n_s} & O_{n_s\times n_c} & \ldots & \ldots & D_{p,p}I_{n_s} - k_tA_c & -k_tB_c \end{bmatrix} \tag{57}$$

moreover, we can define for the first phase $j = 1$ the vector $\mathbf{b_{dyn}^1}$ as

$$\mathbf{b_{dyn}^1} = \begin{bmatrix} -D_{1,0}x_0 + k_tB_cg \\ \vdots \\ \vdots \\ -D_{p,0}x_0 + k_tB_cg \end{bmatrix} \tag{58}$$

while for the other segments (i.e., $j \in [2, \ldots, n]$) we have

$$\mathbf{b^j_{dyn}} = \begin{bmatrix} -D_{1,0}x_p^{j-1} + k_t B_c g \\ \vdots \\ \vdots \\ -D_{p,0}x_p^{j-1} + k_t B_c g \end{bmatrix} \tag{59}$$

The term k_t is computed as $(t_f^n - t_0^1)/2n$, according to the definition of Eq. (20). Note that the knowledge of the initial conditions is exploited to construct the vector $\mathbf{b^1_{dyn}}$ through the first column of the matrix $\mathbf{D}$, representing the discrete, non-collocated point corresponding to x_0, while the linking conditions are implicitly guaranteed by including the elements x_p^{j-1} in the definition of the vectors $\mathbf{b^j_{dyn}}$.

Final Conditions

Arbitrary final conditions can be met by imposing further terms in the system of linear equations we are building. Supposing that all the six components on position and velocity are constrained to some values $\mathbf{r_f}$, $\mathbf{v_f}$, we can impose them by defining a further matrix $\mathbf{A_{fc}}$, and a further vector $\mathbf{b_{fc}}$ as

$$\mathbf{A_{fc}} = \begin{bmatrix} O_{(n_s-1)\times n_s} & O_{(n_s-1)\times n_c} & \ldots & \ldots & I_{(n_s-1)} & O_{(n_s-1)\times n_c+1} \end{bmatrix} \tag{60}$$

$$\mathbf{b_{fc}} = \begin{bmatrix} \mathbf{r_f} & \mathbf{v_f} \end{bmatrix}^T \tag{61}$$

Remark 3 Note that the number of rows of $\mathbf{A_{fc}}$ and elements of $\mathbf{b_{fc}}$ are in this case equal to $n_s - 1$ because the final mass is not constrained.

Remark 4 The number of rows of $\mathbf{A_{fc}}$ and elements of $\mathbf{b_{fc}}$, can be further reduced in case only some of the components of $\mathbf{r_f}$ and $\mathbf{v_f}$ are constrained. In that case it is sufficient to delete the rows and elements corresponding to the non-constrained final values.

The linear system representing the dynamics and the final conditions is therefore given by the following condition

$$\mathbf{AX} = \mathbf{b} \tag{62}$$

where

$$\mathbf{A} = \begin{bmatrix} \mathbf{A^1_{dyn}}^T & \ldots & \mathbf{A^n_{dyn}}^T & \mathbf{A_{fc}}^T \end{bmatrix}^T \tag{63}$$

and

$$\mathbf{b} = \begin{bmatrix} \mathbf{b^1_{dyn}}^T & \ldots & \mathbf{b^n_{dyn}}^T & \mathbf{b_{fc}}^T \end{bmatrix}^T \tag{64}$$

Constraints

As first step we need to include the condition described by Eq. (44). This is done by including the following conic constraint:

$$\left\| \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \\ \sigma \end{bmatrix}_i^j \right\|_2 \leq \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_x \\ u_y \\ u_z \\ \sigma \end{bmatrix}_i^j, \quad \begin{matrix} j = [1, \ldots, n] \\ i = [1, \ldots, p] \end{matrix} \tag{65}$$

The no-subsurface constraint can be imposed by the following inequality

$$\begin{bmatrix} 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}_i^j \leq 0, \quad \begin{matrix} j = [1, \ldots, n] \\ i = [1, \ldots, p] \end{matrix} \tag{66}$$

Note that this inequality represents a positive orthant, which is a special case of cone.

In case we want the glideslope constraint to be active, the previous constraint can be replaced by the following conic constraint:

$$\left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}_i^j \right\|_2 \leq \begin{bmatrix} 0 & 0 & \frac{1}{\tan\theta_{alt}} \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}_i^j, \quad \begin{matrix} j = [1,\ldots,n] \\ i = [1,\ldots,p] \end{matrix} \tag{67}$$

The discrete version of the left-hand side of Eq. (46) can be modeled as a conic constraint too. Let us define the following matrices and vectors:

$$A_\rho = \begin{bmatrix} \frac{\sqrt{2\rho_l e^{-z_l}}}{2} & 0 \end{bmatrix}_i^j, \quad b_\rho = -\begin{bmatrix} \rho_l e^{-z_l}(1+z_l) & 1 \end{bmatrix}_i^j, \quad c_\rho = \rho_l e^{-z_l}\left[1 + z_l + \tfrac{1}{2}z_l^2\right]_i^j \tag{68}$$

and

$$A_c = \begin{bmatrix} \frac{b_\rho}{2} \\ A_\rho \end{bmatrix}_i^j, \quad b_c = \begin{bmatrix} \frac{c_\rho}{2} + \frac{1}{2} \\ 0 \end{bmatrix}_i^j, \quad c_c = -\left.\frac{b_\rho}{2}\right|_i^j, \quad d_c = \left.\frac{1}{2} - \frac{c_\rho}{2}\right|_i^j \tag{69}$$

With these definitions, it is possible to impose the first part of Eq. (46) as

$$\|A_c\tilde{z} + b_c\|_2 \leq c_c^T\tilde{z} + d_c, \quad \begin{matrix} j = [1,\ldots,n] \\ i = [1,\ldots,p] \end{matrix} \tag{70}$$

where

$$\tilde{z} = \begin{bmatrix} z \\ \sigma \end{bmatrix}_i^j \tag{71}$$

Finally, the right-hand side of Eq. (46) is a linear constraint, and using the definition of Eq. (71), is discretized as

$$\begin{bmatrix} \rho_u e^{-z_u} & 1 \end{bmatrix} \tilde{z}\Big|_i^j \leq \left[\rho_u e^{-z_u}(1+z_u)\right]_i^j, \quad \begin{matrix} j = [1,\ldots,n] \\ i = [1,\ldots,p] \end{matrix} \tag{72}$$

The entire problem is therefore expressed as

$$\text{minimize} \quad J = \mathbf{cX} \tag{73}$$

subject to the linear system of Eq. (62), together with the constraints of Eqs. (65)-(72), and represents the transcription of the Mars powered descent problem according to the hp flipped Radau Pseudospectral Convex method (or hp-fRPCm). The initial state is known, and the initial control is extrapolated from the control history once that the problem is solved. The computation of the initial control completes the solution with all the missing information.

VI. Numerical Examples

In this section we describe a series of numerical examples. First, we show the results coming from the nominal scenario. To assess the reliability of the method, we performed a Monte-Carlo campaign with 1000 runs is performed. Finally, an analysis of the accuracy of the solution when the parameters h and p are varied is carried on.

A. Nominal Results

The first analysis is based on the original scenario proposed by Acikmese.[9] In this case we computed the solution for $n = 8$, $p = 7$ (which leads to a total of 50 discrete nodes, including the initial condition). The solution is depicted in Figs. 6-10. We can see from Fig. 6, (showing position, velocity, acceleration and controls) that the solution is fully consistent with the results obtained in the original formulation.[9] The glideslope constraint is also fully satisfied (Fig. 7). Note that with this constraint there is no need to include the no-subsurface condition in the transcription process, since the cone defined by Eq. (40) is stricter than the inequality of Eq. (39), which therefore becomes redundant. The final mass consumption is equal to 399.5 kg, as depicted in Fig. 8. Figures 9-10 show the control history with respect to time and in polar coordinates respectively. The original non-convex control constraints are satisfied, and the equality condition $\Gamma = \|\mathbf{T}\|$ is satisfied, as prescribed by the application of the Pontryagin's minimum principle to this specific problem.

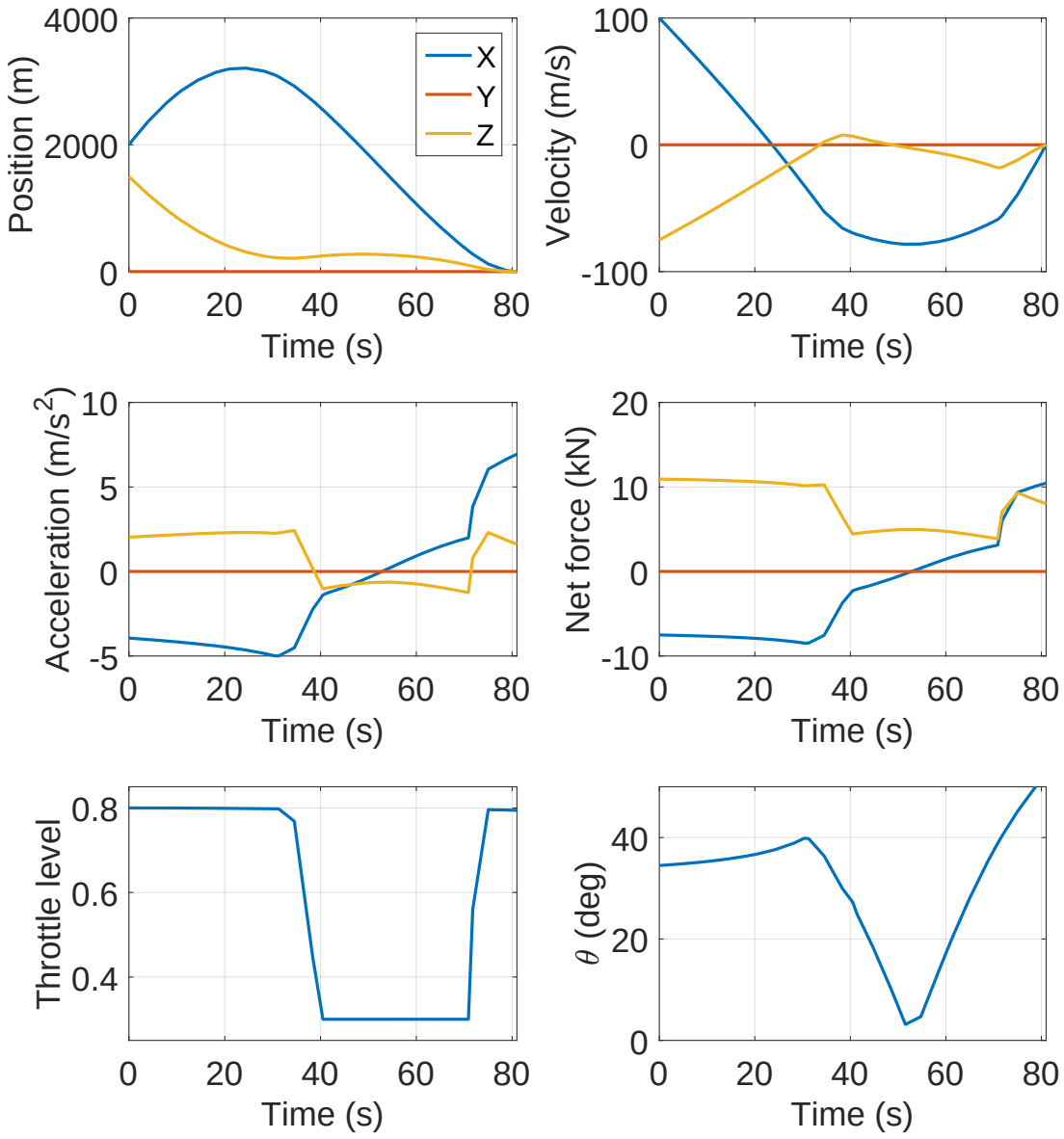


Figure 6. Solution obtained with hp pseudospectral convex method - states and controls.

B. Monte-Carlo Campaign

In this section we show the results associated with a Monte-Carlo campaign (1000 cases) to test the reliability of the method. Perturbations in terms of initial position, velocity, mass, and final time are included. The ranges and the type of uncertainty for each of the variables included in the analysis are listed in Table 1. States and controls are depicted in Fig. 11. The resulting perturbed trajectories are shown in Fig. 12(a), while the mass consumption profiles are plotted in Fig. 12(b).

Remark 5 Note that in the Figs. 11-12 only the first 25 cases are shown for a better visualization of the results.

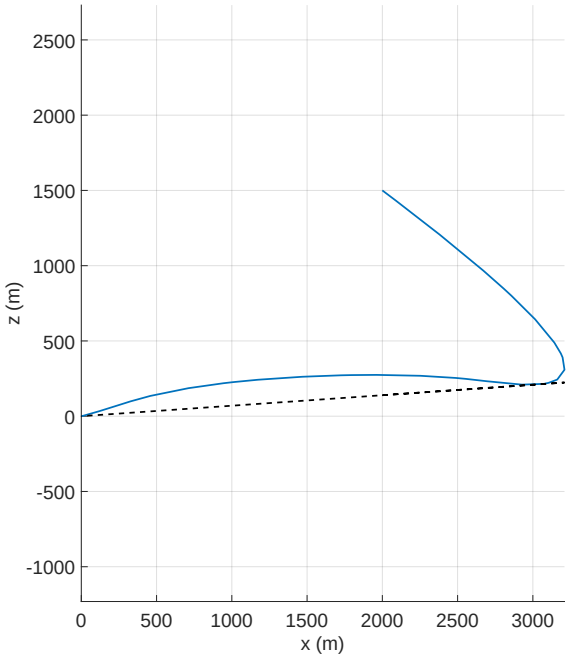


Figure 7. Solution obtained with hp pseudospectral convex method - trajectory.

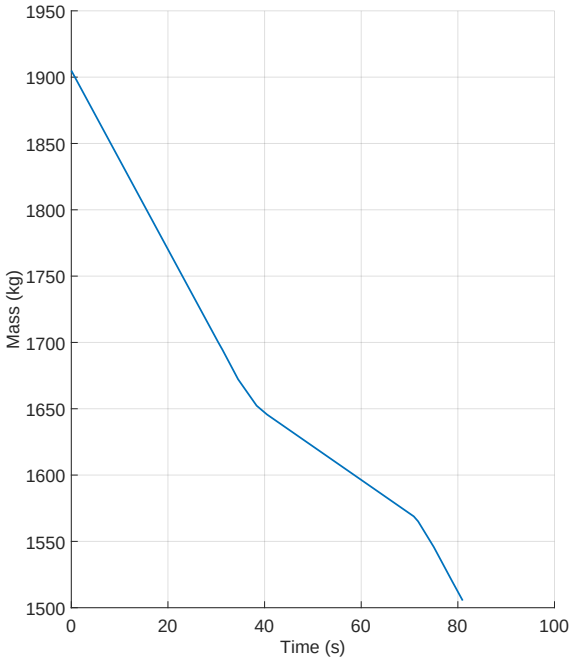


Figure 8. Solution obtained with hp pseudospectral convex method - mass consumption.

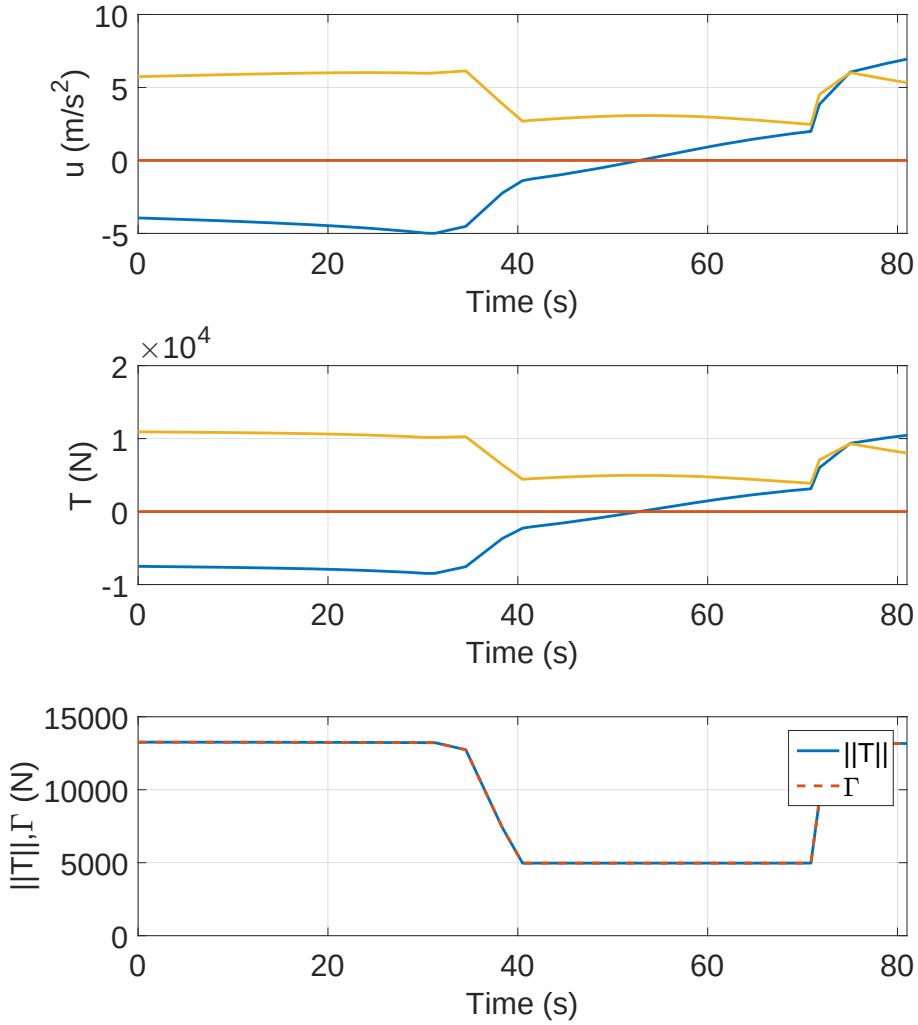


Figure 9. Solution obtained with hp pseudospectral convex method - control components.

From the analysis of Figs. 11 we can observe that all the solutions satisfy the constraints as in the nominal case. Despite the large variations in initial velocities, the scheme is able to steer the lander towards the prescribed point in every case analyzed. Of course, since the final time is not optimal, and significant variations of initial position and velocity are introduced, the mass consumption varies quite strongly. In fact, the optimal value of mass spent during the landing varies between 381.9 and 428.7 kg (Fig. 12(a)). The large variability of initial conditions is visible in the resulting trajectories depicted in Fig. 12(b), where not only two-dimensional cases, but also full three-dimensional cases are considered.

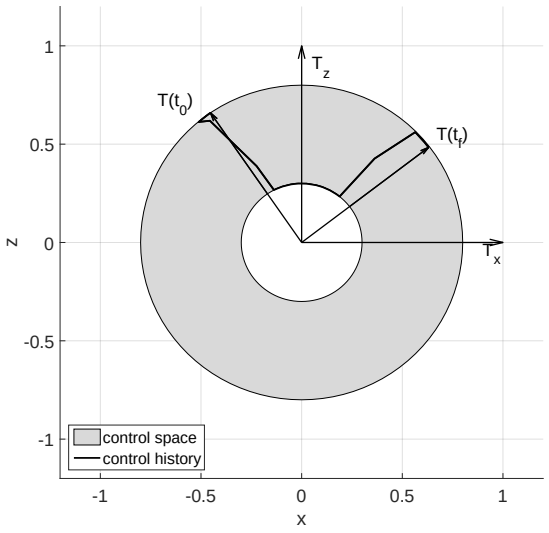


Figure 10. Solution obtained with hp pseudospectral convex method - control space.

Table 1. Monte-Carlo analysis - uncertainties.

Parameter	Dispersion	Distribution	Units
$\|\mathbf{r}(t_0)\|$	± 25	Normal (1σ)	m
$\|\mathbf{v}(t_0)\|$	± 5	Normal (1σ)	m/s
$m(t_0)$	± 1	Uniform	kg
t_f	± 3	Uniform	s

C. Accuracy Comparison

Finally, a comparison of accuracy between the pre-existing techniques and the new method has been carried out. The comparison has been performed by looking at CPU times required to obtain a solution and at the errors accumulated when the solution is propagated via Runge-Kutta scheme. The errors obtained by propagating the optimal solution for the case $n = 8$, $p = 7$ are depicted in Fig. 13. Specifically, the errors in position and velocity obtained by using standard convex methods are shown in Fig. 13(a), whereas Fig. 13(b) depicts the corresponding errors associated with the proposed hp pseudospectral method. It is already well-visible that the proposed scheme is more accurate than standard methods given the number of nodes employed to compute the solutions.

Finally, a comparison in terms of accuracy and CPU time has been performed. The proposed method has been compared with the standard convex algorithm (here indicated simply as *convex*), as well as with the p-pseudospectral convex method, based on the use of a unique domain (called *p-ps-convex*). Consistently, the proposed method has been labeled *hp-ps-convex*.

Results are compared in terms of errors, which measure the accuracy of the different convex algorithms, and the CPU time required to compute a valid solution. The errors are obtained as difference between the optimal trajectory and the propagated solution generated by propagating the optimal controls. The CPU times are simply the time required by ECOS[31] to solve the corresponding SOCP problem. For a better characterization of the CPU times, every case has been computed 10 times, and the results averaged. The dataset is computed for different values of h and p. The compound number of nodes (that is, 25, 50 and 100) associated with each *hp* case has been used as input to the other two algorithms for a fair comparison of the results, presented in Table 2.

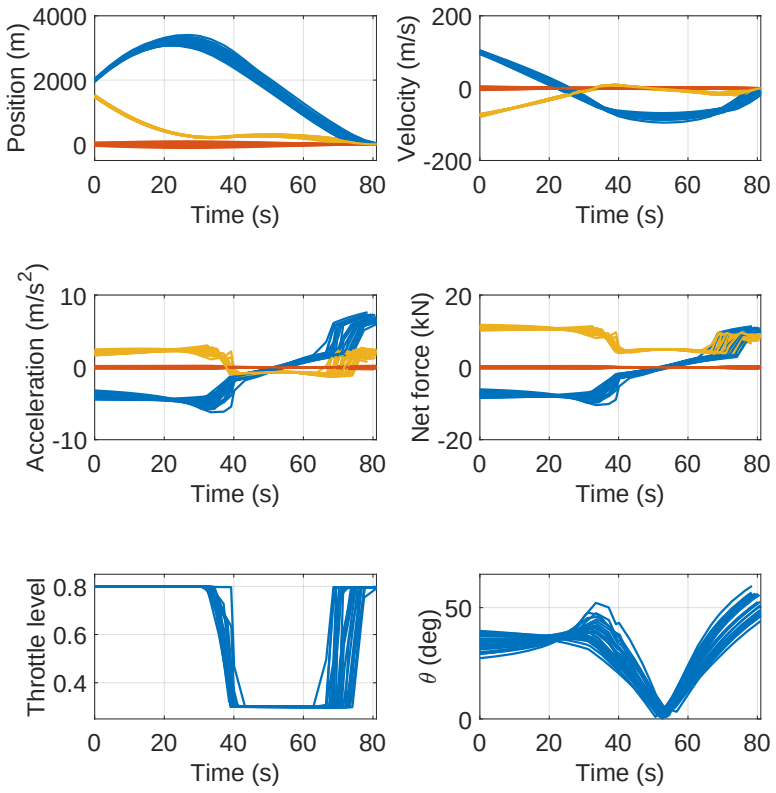
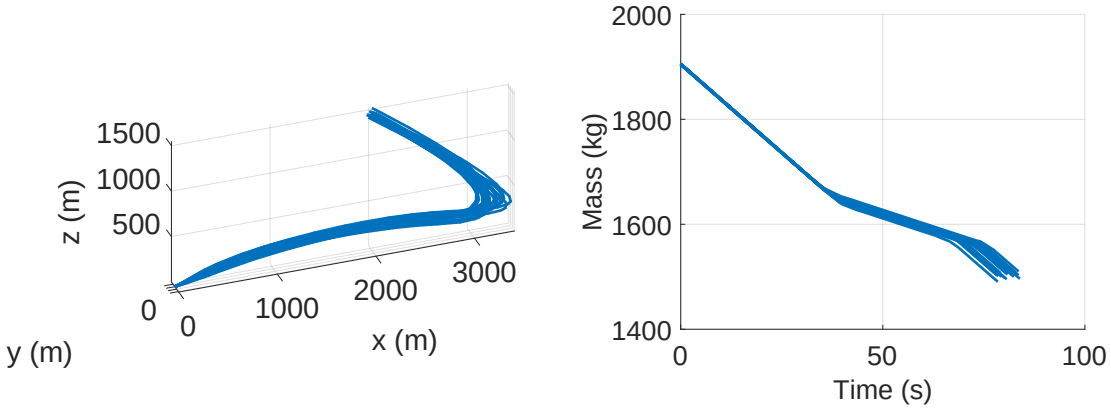


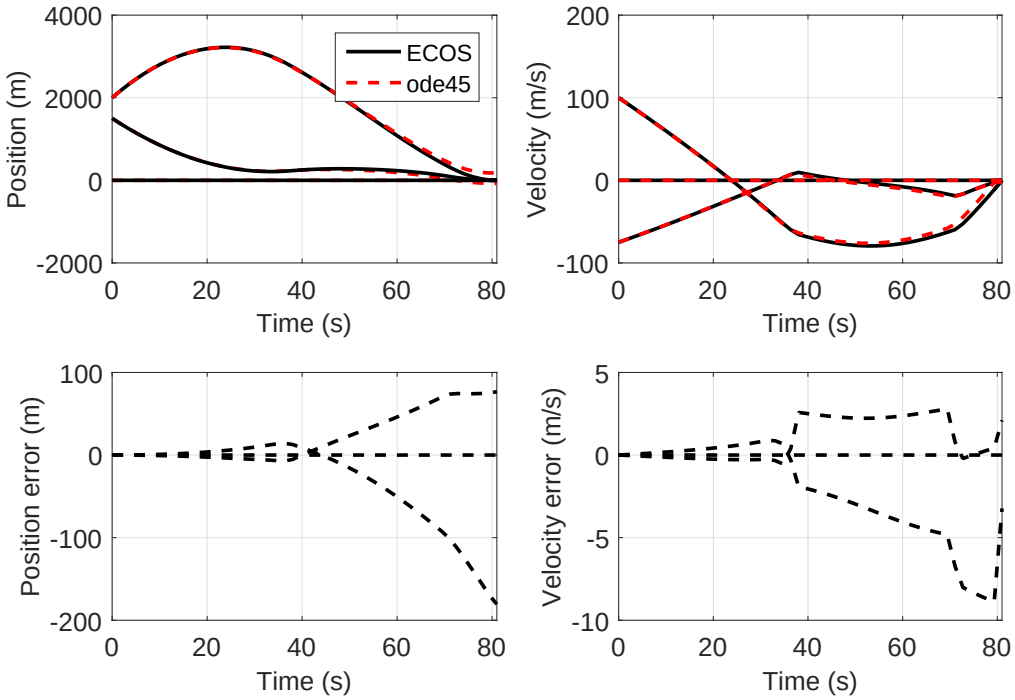
Figure 11. Monte-Carlo analysis of hp pseudospectral convex method - states and controls, x component in blue, y component in red, z component in yellow (1000 cases).



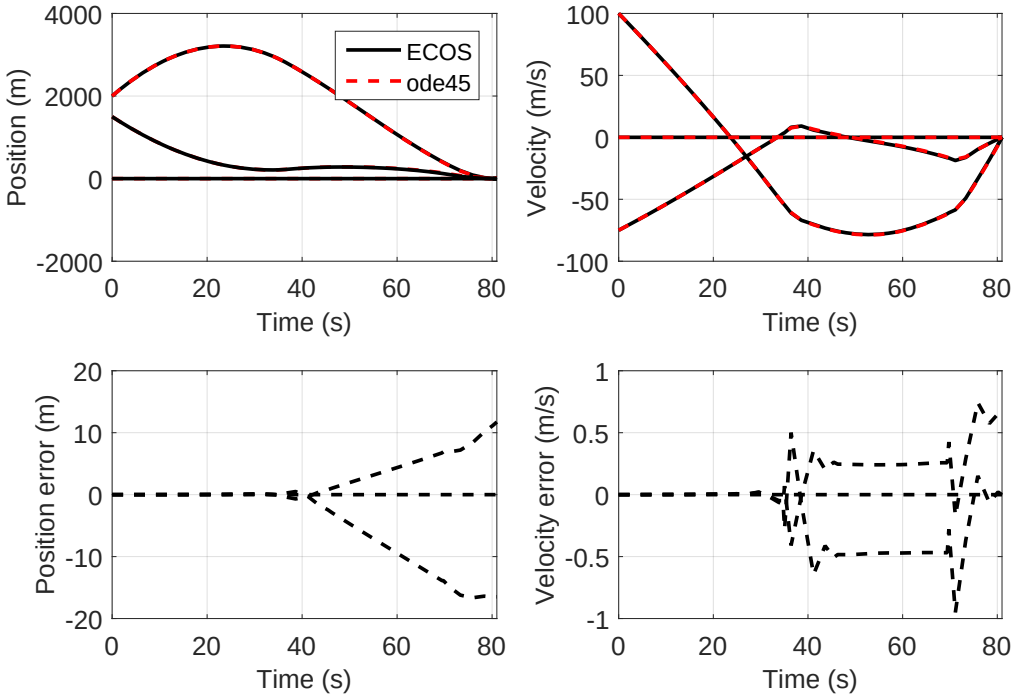
(a) Trajectories

(b) Mass profiles

Figure 12. Monte-Carlo analysis of hp pseudospectral convex method - trajectories and mass consumption (1000 cases).



(a) Errors obtained with standard convex transcription



(b) Errors obtained with hp pseudospectral convex transcription

Figure 13. Accuracy analysis - standard convex method vs hp pseudospectral convex method - 50 nodes.

Table 2. Comparison of accuracy and CPU times.

Method	Nodes	CPU time (μ, ms)	CPU time (1σ, ms)	max error pos (m)	max error vel (m/s)
hp ps-convex	25	46	18.0	89.82	3.30
convex	25	16	0.3	403.00	18.87
p ps-convex	25	74	1.0	24.89	1.35
hp ps-convex	50	73	0.5	20.36	1.21
convex	50	34	0.4	196.24	9.31
p ps-convex	50	450	26.5	6.07	0.83
hp ps-convex	100	208	26.4	15.42	0.65
convex	100	108	47.4	96.86	4.63
p ps-convex	100	2760	354.1	1.51	0.38

The most accurate method is the *p* ps-convex, with a maximum error in terms of position and velocity ($n = 25$) equal to about 24 m and 1.4 m/s, respectively. The errors go up to more than 400 m and 18 m/s when the standard convex algorithm is employed. The proposed *hp* ps-convex reduces this error to about 90 m and 3.3 m/s. In terms of CPU time the standard method is still faster, with a mean value of 16 ms against a value 4.5 times larger when the *p* method is used. The *hp* method represents an ideal compromise since it is faster than the *p* method (46 ms), but the error produced is much closer to the *p* method than to the standard convex scheme. When the number of nodes is increased the differences in time become more larger as the *p* method computation is heavier (about 2.7 s), against 208 ms required by the *hp* method. However, the error obtained by the *hp* method is still much better than the standard method. The difference in the CPU times can be easily understood if one looks at the patterns of the associated matrix representing the underlying system of equations, depicted for the three methods in Figs. 14, 15 and 16. While the standard method is much closer to a diagonal matrix, the *p* method results in a sparse matrix having the elements much more uniformly distributed. The introduction of segments provided by the *hp* method instead allows for a block-diagonal matrix, which translates into a faster method than the *p* method, and at the same time into a more accurate algorithm with respect to the standard method.

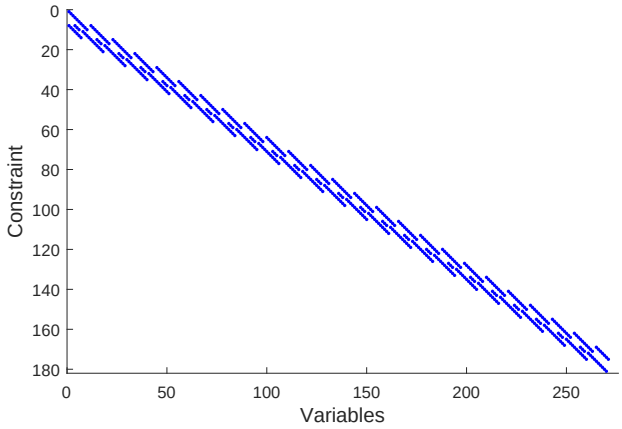


Figure 14. Matrix associated with the system of linear equations - standard convex transcription: density = 1.18%.

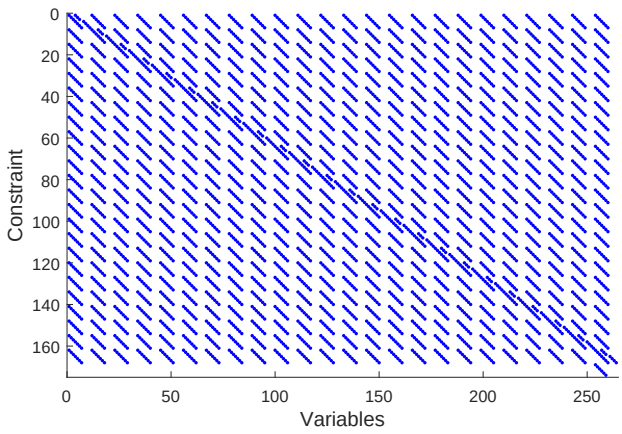


Figure 15. Matrix associated with the system of linear equations - p pseudospectral convex transcription: density = 9.16%.

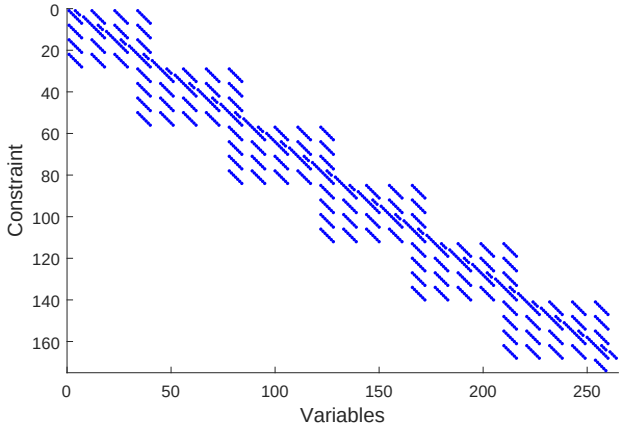


Figure 16. Matrix associated with the system of linear equations - hp pseudospectral convex transcription: density = 2.15%.

VII. Conclusions and Future Outlook

In this work a new hybrid framework consisting of hp-pseudospectral methods and convex optimization has been proposed. The purpose was to reduce the computational gap with respect to standard convex schemes applied to descent and landing scenarios, while increasing the accuracy of the solution computed by the optimization process beneath them. Results confirm that the proposed technique leads to a significant reduction of the CPU times (e.g., up to more than 13 times when 100 nodes are used), with respect to the previous generation of pseudospectral convex methods, while being at the same time significantly more accurate than standard convex optimization. Future work will include different scenarios to extend the analysis to a broader class of problems, and will analyze more in depth the influence of the h and p terms in the accuracy of the scheme, to come up with an optimal selection criterion to be integrated within the method.

References

[1]space.com, "http://www.space.com/31420-spacex-rocket-landing-success.html," 2015.

[2]space.com, "http://www.space.com/31202-blue-origin-historic-private-rocket-landing.html," .

[3]Bennett, F., "Lunar descent and ascent trajectories," *Aerospace Sciences Meetings, American Institute of Aeronautics and Astronautics*, Jan. 1970.

[4]Jungmann, J., "Gravity turn trajectories through planetary atmospheres," Meeting Paper Archive, American Institute of Aeronautics and Astronautics, Aug. 1967.

[5]McInnes, C. R., "Gravity-Turn Descent from Low Circular Orbit Conditions," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 1, Jan. 2003, pp. 183–185.

[6]Sostaric, R. and Rea, J., "Powered Descent Guidance Methods For The Moon and Mars," *Guidance, Navigation, and Control and Co-located Conferences*, American Institute of Aeronautics and Astronautics, Aug. 2005.

[7]Ingoldby, R. N., "Guidance and Control System Design of the Viking Planetary Lander," *Journal of Guidance, Control, and Dynamics*, Vol. 1, No. 3, May 1978, pp. 189–196.

[8]Boyd, S. and Vandenberghe, L., *Convex Optimization*, 2004.

[9]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.

[10]Sagliano, M., Theil, S., Bergsma, M., D'Onofrio, V., Whittle, L., and Viavattene, G., "On the Radau pseudospectral method: theoretical and implementation advances," *CEAS Space Journal*, Vol. 9, No. 3, Sep 2017, pp. 313–331.

[11]Ross, I. M., Sekhavat, P., Fleming, A., and Gong, Q., "Pseudospectral Feedback Control: Foundations, Examples and Experimental Results," *AIAA Guidance, Navigation, and Control Conference, Keystone, USA,*, 2006.

[12]Garg, D., *Advances in Global Pseudospectral Methods for Optimal Control*, Ph.D. thesis, University of Florida, Gainesville, 2011.

[13]Gill, P. E., Murray, W., and Saunders, M. A., *User's Guide for SNOPT Version 7: Software for Large-Scale Nonlinear Programming*, University of California, San Diego, USA, 2008.

[14]Wächter, A. and Biegler, L. T., "On the implementation of an interior-point filter linesearch algorithm for large-scale nonlinear programming," *Math. Program. 106(1) , Springer-Verlag, New York, 2006*, 2006.

[15]Acikmese, A. B. and Ploen, S., "A Powered Descent Guidance Algorithm for Mars Pinpoint Landing," Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug. 2005.

[16]Sagliano, M., "Pseudospectral Convex Optimization for Powered Descent and Landing," *Journal of Guidance, Control and Dynamics, Accepted*, 2017.

[17]Melenk, J. M., *hp-Finite Element Methods for Singular Perturbations*, Springer, 2004.

[18]Rao, A. V., "A Survey of Numerical Methods for Optimal Control," *AAS/AIAA Astrodynamics Specialist Conference, AAS Paper 09-334, Pittsburgh, PA, August 10 - 13*, 2009.

[19]Elissar, "Description of DIDO Optimal Control software," June 2015.

[20]Rao, A. V., Benson, D. A., L., D. C., Patterson, M. A., Francolin, C., I., S., and Huntington, G. T., "GPOPS: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using the Gauss Pseudospectral Method," *ACM Transactions on Mathematical Software*, Vol. 37, No. 2, apr 2010, pp. 1–39.

[21]Gong, Q., Ross, I. M., Kang, W., and Fahroo, F., "Connections Between The Covector Mapping Theorem and Convergence of Pseudospectral Methods for Optimal Control," *Comput Optim Appl, 2008*, 2008.

[22]Huntington, G. T., *Advancement and analysis of a Gauss pseudospectral transcription for optimal control problems*, Ph.D. thesis, Citeseer, 2007.

[23]Fahroo, F. and Ross, I. M., "Direct Trajectory Optimization by a Chebyshev Pseudospectral Method," *Journal of Guidance, Control, and Dynamics*, Vol. 25, No. 1, Jan. 2002, pp. 160–166.

[24]Martins, J. R. R., Sturdza, P., and Alonso, J. J., "The Complex-Step Derivative Approximation," *ACM Transactions on Mathematical Software, Vol. 29, No. 3, September 2003, Pages 245262*, 2003.

[25]Abramovitz, M. and Stegun, I. A., *Handbook of Mathematical Functions*, Dover Publications, 1695.

[26]Karniadakis, G. and Sherwin, S., *Spectral/hp element methods for computational fluid dynamics*, Oxford University Press, 2013.

[27]Darby, C. L., Hager, W. W., and Rao, A. V., "An hp-adaptive pseudospectral method for solving optimal control problems," *Optimal Control Applications and Methods*, Vol. 32, No. 4, 2011, pp. 476–502.

[28]Ben Tal, A. and Nemirovski, A., *Modern Lectures on Convex Optimization*, 2001.

[29]Lobo, M. S., Vandenberghe, L., Boyd, S., and Lebret, H., "Applications of Second-Order Conic Programming," *Linear Algebra and Its Applications*, 1998, pp. 193–228.

[30]Domahidi, A., *Methods and Tools for Embedded Optimization and Control*, Ph.D. thesis, 2013.

[31]Domahidi, A., Chu, E., and Boyd, S., "ECOS: An SOCP Solver for Embedded Systems," *European Control Conference*, 2013.

[32]Martin, M. S., G.Mendeck, Brugarolas, P. B., Singh, G., and Serricchio, F., "In-Flight Experience of the Mars Science Laboratory Guidance, Navigation, and Control System for Entry, Descent, and Landing," *9^{th} International ESA Conference on Guidance, Navigation, and Control Systems*, 2014.

[33]JPL, "7 Minutes of Terror," 2012.

[34]Blackmore, L., Acikmese, 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, July 2010, pp. 1161–1171.

[35]Szmuk, M., Eren, U., and Acikmese, B., "Successive Convexification for Mars 6-DoF Powered Descent Landing Guidance," AIAA SciTech Forum, American Institute of Aeronautics and Astronautics, Jan. 2017.