Pseudospectral Convex Optimization for Powered Descent and Landing

Marco Sagliano[1]
Deutsches Zentrum für Luft- und Raumfahrt, Bremen, Germany, 28359

Over the last years two new technologies to solve optimal-control problems were successfully developed, that is pseudospectral optimal control and convex optimization, the former for solving general nonlinear programming problem, and the latter aimed at solving convex problems (e.g., second-order conic problems) in real-time. In this paper a framework for combining them, with a motivational example, are described. The benefits of the new proposed method are demonstrated for the descent phase of the NASA Mars Science Laboratory. Numerical simulations show that the proposed algorithms lead to more accurate results with respect to standard transcription methods.

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

---

[1] GNC Engineer, Navigation and Control Department, AIAA Member

$\mathbf{g}$ = Gravity vector, [m/s^2]

I_n = Identity matrix of dimensions n

i,j,k,m,n = Non-negative, integer indices

J = Cost function

k_t = Physical time - pseudospectral time conversion factor, [s]

$\tilde{L}_n(\tau)$ = Legendre polynomial of degree n

$L_n(\tau)$ = Legendre-Lobatto polynomial of degree n

m = Lander mass, [kg]

$O_{n_1 \times n_2}$ = Zero matrix of dimensions n_1,n_2

$P(\tau)$ = Lagrange polynomial

$R_n(\tau)$ = Legendre-Radau polynomial of degree n

$\mathbf{r}$ = Position vector, [m]

$\mathbf{T_c}$ = 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 system parameter, [s/m]

Γ, σ = slack variables, [N, N/m]

Φ = Mayer term

Ψ = Lagrange term

ρ = Thrust limit, [N]

τ = Pseudospectral time

ω = Radau / Lobatto quadrature weights

Operators and subscripts

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

I. Introduction

Over the last years the space race has seen a dramatic paradigm shift. While in the last 40 years of the 20^{th} century the challenge was played to establish a military supremacy between the western and the eastern blocks, the global imperative is now the economical sustainability of the space missions. SpaceX [1] showed that the reusability is the key for a dramatic reduction of the costs associated with space exploration first and commercial exploitation later on. One of the critical factors for having an efficient descent and landing system is the spacecraft's capability to generate real-time guidance solutions. These include trajectories and commands, which satisfy all the criteria of the mission while properly dealing with the uncertainties acting on the system, (for example the spacecraft has to be able to re-compute its trajectory without violating any constraint, like a given glideslope limit required for proper hazard-avoidance).

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 [2], 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 [3–5], 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 specific criteria (that is, for all those problems which are subject to convex constraints). The method found further aerospace applications (e.g., the atmospheric entry guidance problem [9]), and in general to non-convex problems as well [10]. 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 [11–14]. The method was successfully demonstrated in 2013 with the Masten Space Systems's Xombie flight [15] and in the last successful flights of SpaceX's Falcon 9 [1]. 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 [16–19]. They use non-uniform grids, leading to smoother results, and a small number of nodes required to compute a valid solution [20–22]. 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 [23] or IPOPT [24]. However these methods cannot in general solve the underlying nonlinear programming (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.

In this work we present a novel method based on the hybridization of pseudospectral methods and convex optimization, leading to the proposed pseudospectral convex optimization, potentially able to provide a more accurate class of methods for real-time optimal control. A first step in that sense can be already found in [25]. However, in that case 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. In the present work the properties of pseudospectral methods are deeply combined with the pre-existing convex framework. The idea is improve the accuracy of the current methods without having an excessive worsening of the real-time capability of the convex framework by adopting pseudospectral operators. In fact, their linearity, together with the higher accuracy they provide with respect to standard operators (such as finite differences for differentiation, or the trapezoidal rule for integration) allow to define a new method, which is still real-time capable, and at the same time more accurate than standard convex approaches.

The paper is organized as follows. Sections 2 and 3 provide a brief overview on Pseudospectral methods and Convex optimization, respectively. More specifically, the latter refers to a special form of convex optimization, that is, the Second-Order Conic Programming (SOCP). In Sec. 4 a simple one-dimensional example, motivating the work, is presented, while the problem we focus on, that is, the Mars powered descent problem is presented in Sec. 5. The new pseudospectral convex optimization framework is presented in Sec. 6, while numerical simulations showing the benefits of the proposed techniques are the subject of Sec. 7. Finally, Sec. 8 presents some conclusions about this work.

II. Overview on 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 is gaining popularity, 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:

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

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, 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 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. Moreover, 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:

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

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

Equations (1)-(5) 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 Pseudospectral methods.

B. 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 of the OCP (or *transcription*), 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 [26] and SPARTAN [16, 17, 22, 27, 28]. For 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
- Straightforward implementation
- 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 [29].

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 methods two were considered for this work: the flipped Radau Pseudospectral method (or fRPm) and the Lobatto Pseudospectral method (LPm). It is worth saying that these are not the only possible choices, as other sets of nodes, like Chebyshev [30] or Gauss [21] 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 [21], while LPm is for some problems more accurate and faster in converging than other PS methods. Therefore, it is useful to have a look at these two methods, and at their transcription. This will be the purpose of the next subsection.

C. Flipped Radau Pseudospectral method and Lobatto 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.

$$R_n(\tau) = \widetilde{L}_n(\tau) - \widetilde{L}_{n-1}(\tau) \quad \tau \in [-1, 1] \tag{6}$$

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 over the left-open, right-closed interval $(-1, +1]$ only these polynomials are orthogonal.

Lobatto Pseudospectral method is instead based on a symmetric set of nodes, associated with

the roots of the Legendre-Lobatto polynomial, defined as

$$L_n(\tau) = (1-\tau^2)\dot{\tilde{L}}_{n-1} \quad \tau \in [-1,1] \tag{7}$$

where $\dot{\tilde{L}}_{n-1}$ is the derivative of the Legendre polynomial of order $n-1$. The roots of the Legendre-Lobatto polynomial and the corresponding polynomial of order 10 are represented in Fig. 1(b).

These discrete representations of the domain are useful to reconstruct continuous representations of the functions $x(t)$ as:

$$x(t) \cong \sum_{i=0}^{n} X_i P(t), \qquad P(t) = \prod_{\substack{k=0 \\ k\neq i}}^{n} \frac{t-t_k}{t_i-t_k} \tag{8}$$

in case of fRPm, and

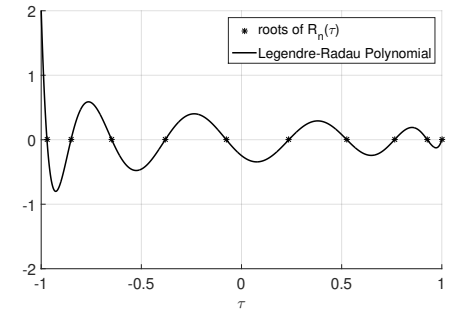
$$x(t) \cong \sum_{i=0}^{n-1} X_i P(t), \qquad P(t) = \prod_{\substack{k=0 \\ k\neq i}}^{n-1} \frac{t-t_k}{t_i-t_k} \tag{9}$$

which holds in case the LPm is adopted. From the inspection of Eqs. (8) and (9) one can see a first difference between the methods. Indeed, given n collocation nodes, fRPm defines $n+1$ discretization nodes, while LPm has n discretization nodes, that is, all the discrete nodes are collocation nodes too. 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 pseudospectral convex method too. This will be further explained in Sec. VI.

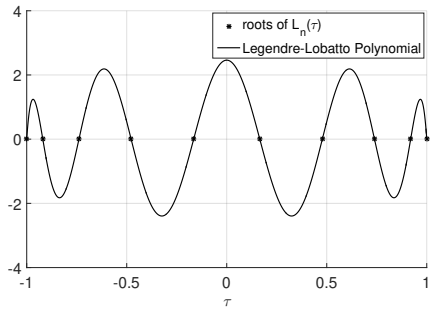
An example of the approximation obtained via Eqs. (8) and (9) is depicted in Figs. 1(c) and 1(d), where the function $1/(1+25\tau^2)$ is reconstructed by using 25 fRPm and 25 LPm nodes, respectively. In both cases the original function is approximated very well with the two sets of discrete nodes.

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

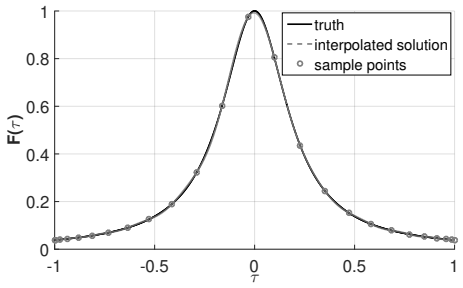
Once the domain has been discretized, and the discrete-to-continuous conversion of states has been defined, the corresponding differential operator needs to be defined. This is required for the



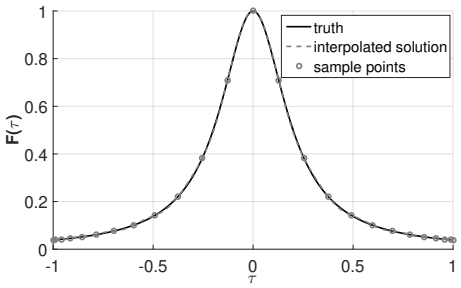
(a) f-Radau discrete domain.



(b) Lobatto discrete domain.



(c) f-Radau continuous approximation of a function.



(d) Lobatto continuous approximation of a function.

Fig. 1 Transcription steps: domain discretization with fRPm (a), and LPm (b) and continuous reconstruction of functions with fRPm (c) and LPm (d).

proper representation of the left-hand side of Eq. (2). The differential operator will be in the form

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

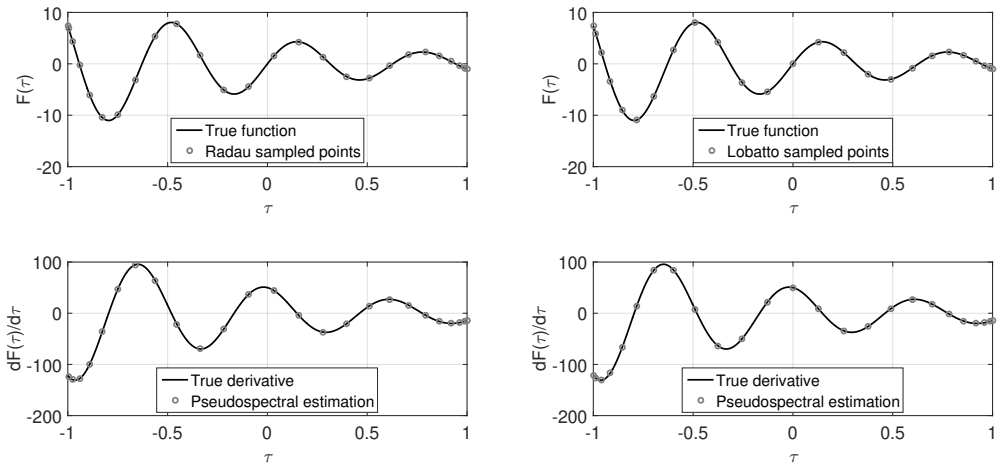
and the dynamics defined in Eq. (2) will be replaced by

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

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,

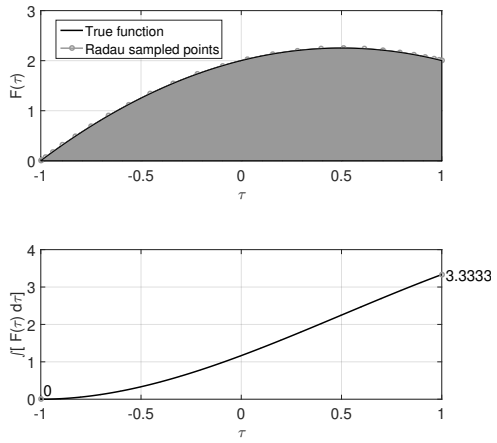
$$t = \frac{t_f - t_0}{2} \tau + \frac{t_f + t_0}{2} \tag{12}$$

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

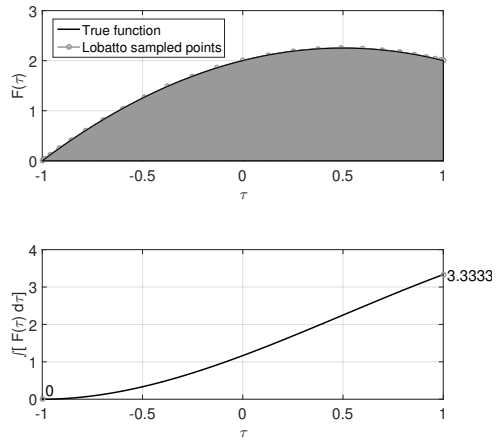


(a) Radau differential operator example.

(b) Lobatto differential operator example.



(c) Radau integral operator example.



(d) Lobatto integral operator example.

Fig. 2 Transcription steps: definition of differential operators with fRPm (a) and LPm (b), and integral operators with fRPm (c) and LPm (d).

which hold for both fRPm and LPm. The difference between the methods is in the matrix $\mathbf{D}$. In the case of the fRPm it has dimensions $[n \times (n+1)]$. 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 in the fRPm, and it is thus assumed to be known. In the LPm instead the matrix $\mathbf{D}$ has dimensions equal to $[n \times n]$. The initial state can be determined by the optimization process. However, since it is generally known, further constraints need to be imposed

to make sure that the solution found by the optimizer satisfies the condition $\mathbf{x}(t_0) = \mathbf{x}_0$. If we look at fRPm (specifically at Eq. (8)), and we take the derivative w.r.t. time, we get

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

as the nodal points are time-independent. When we consider the LPm instead (Eq. (9)) we have

$$\dot{\mathbf{x}}(t) \cong \frac{d}{dt}\sum_{i=0}^{n-1} \mathbf{X_i} P(t) = \sum_{i=0}^{n-1} \mathbf{X_i} \frac{d}{dt} P_i(t) \tag{15}$$

These two sets of derivatives can be efficiently computed with the Barycentric Lagrange Interpolation [31]. An example of the differential operator for the two methods is depicted in Figs. 2(a) and 2(b), where $\mathbf{D}$ is used to approximate the derivative of the continuous test function $F(\tau) = Ae^{-\tau}\sin(\omega\tau)$, $(A = 5, \omega = 10)$ sampled in 25 collocation nodes. It can be seen that the polynomial approximations fit the derivatives very well.

In addition to the differential operator, we need an integral operator, used to discretize the Lagrange term defined in Eq. (1). In that case the Gauss quadrature formula is used [32]. For the fRPm the approach consists of replacing the continuous integral with the discrete sum given by:

$$\int_{t_0}^{t_f} \Psi\left[t, \mathbf{x}(t), \mathbf{u}(t)\right] dt = \frac{t_f - t_0}{2} \sum_{i=1}^{n} w_i \Psi\left[\mathbf{X}_i, \mathbf{U}_i\right] \tag{16}$$

while for the LPm it becomes

$$\int_{t_0}^{t_f} \Psi\left[t, \mathbf{x}(t), \mathbf{u}(t)\right] dt = \frac{t_f - t_0}{2} \sum_{i=0}^{n-1} w_i \Psi\left[\mathbf{X}_i, \mathbf{U}_i\right] \tag{17}$$

Since both methods have the same number of n collocation nodes, both sums use n nodes to represent the integral operators. It can be shown that Eqs. (16) and (17) yield exact results for polynomials of order at most equal to $2n-2$ and $2n-3$ for fRPm and LPm, respectively [21]. 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). For the fRPm the weights w_i can be computed as

$$w = flip(\tilde{w}) \tag{18}$$

$$\tilde{w}_j = \begin{cases} \dfrac{2}{n^2}, & j = 1 \\ \dfrac{(1-\tau_j)}{n^2 \tilde{L}_n(\tau_j)^2}, & j = [2, ...n] \end{cases} \tag{19}$$

where the operator *flip* simply multiplies the input by a factor equal to -1, and sorts the results in increasing order. For the LPm the formula is

$$w_j = \begin{cases} \dfrac{2}{(n-1)n}, & j = 0 \\ \dfrac{2}{(n-1)nL_{n-1}(\tau_j)^2}, & j = [1, \ldots, n-1] \end{cases} \tag{20}$$

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 (Figs. 2(c),2(d)) applied using the same nodes. Numerically, we get exactly the analytical result, that is 3.3333 for both the pseudospectral methods, while the application of the trapezoidal rule gives 3.3298 and 3.3296, respectively, confirming the validity of the quadrature formula applied to the f-RPm and LPm points. Note that when n uniformly distributed nodes are used the trapezoidal rule gives better results (3.3310), but still inferior to the pseudospectral ones.

Once that the differential and integral operators have been described, we are ready to summarize the general NLP transcriptions, which approximates the original OCP as follows.

Flipped Radau Pseudospectral method

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

$$J = \Phi\left[\mathbf{X}_f\right] + \frac{t_f - t_0}{2} \sum_{i=1}^{n} w_i \Psi\left[\mathbf{X}_i, \mathbf{U}_i\right] \tag{21}$$

subject to the nonlinear algebraic constraints

$$\mathbf{F} = \mathbf{D} \cdot \mathbf{X} - \frac{t_f - t_0}{2}\mathbf{f}(t, \mathbf{X}, \mathbf{U}) = \mathbf{0} \tag{22}$$

and to the path constraints

$$\mathbf{g}_L \leq \mathbf{G}\left(\mathbf{X}_i, \mathbf{U}_i\right) \leq \mathbf{g}_U \tag{23}$$

The discrete states and the controls are bounded, as in the continuous formulation.

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

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

Lobatto Pseudospectral method

Minimize (or maximize) the cost function J, for n nodes, $i = 0, \ldots, n-1$,

$$J = \Phi\left[\mathbf{X}_f\right] + \frac{t_f - t_0}{2}\sum_{i=0}^{n-1} w_i \Psi\left[\mathbf{X}_i, \mathbf{U}_i\right] \tag{26}$$

subject to Eqs. (22)-(25).

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

III. Overview on Convex Optimization

Over the last thirty years several researchers focused on the development of convex optimization theory [8, 33, 34]. 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. In general a convex optimization problem is defined as follows:

$$\min \; J = f_0(x) \tag{27}$$

subject to

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

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

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

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 [33] and [8].

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 [35]), 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 for 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 Conic Programming (or SOCP). This specific subclass of methods will be briefly described in the next section, whereas more extensive and rigorous descriptions can be found in [33, 34, 36].

A. Second-Order Conic Programming

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

$$\min \; c_0^T x \tag{30}$$

subject to

$$\begin{aligned} A_0 x &= b_0 \\ \|A_i x + b_i\|_2 &\leq c_i^T x + d_i, \; i = 1, \ldots, p \end{aligned} \tag{31}$$

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 $A_0 \in \mathbb{R}^{m \times n}$ 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.

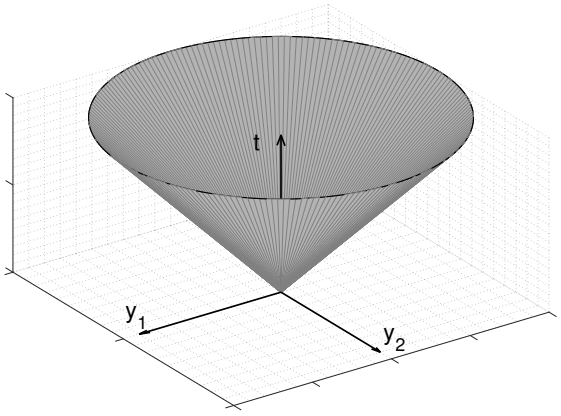


Fig. 3 Example of 3-D cone. The volume of the cone satisfies the condition $\|y\|_2 \leq t$

Among the others, linear programming problems, or quadratically constrained problems can be reformulated as conic programming problems. Moreover, they can efficiently be solved by using primal-dual interior point methods [37], and several solvers, such as SeDuMi [38] and ECOS [39], are available. These aspects make the SOCP technology appealing for several applications, including the one used as example in this work. Further SOCP applications are described in [36].

IV. A motivational example

To motivate the present work we will introduce a very simple optimization problem, which can be formulated as a SOCP problem. We are interested in minimizing the norm of the final state of a first-order linear system.

$$\min J = \|x(t_f)\|_2 \tag{33}$$

The system behavior is described by the following differential equation

$$\dot{x} = ax + bu,\ x, u, a, b \in \mathbb{R} \tag{34}$$

subject to

$$\|u(t)\|_2 \leq u_{max},\ t \in [t_0, t_f] \tag{35}$$

The final time is $t_f = 5$ s. We can discretize the time, the state and the control in n+1 nodes, such that

$$\begin{aligned} t_k &= kdt,\ k = 0, \ldots, n, \\ dt &= (t_f - t_0)/n \end{aligned} \tag{36}$$

If we integrate Eq. (34) by using a trapezoidal scheme we get

$$x_{k+1} = x_k + \frac{1}{2} dt \left[ax_{k+1} + bu_{k+1} + ax_k + bu_k\right],\ k = 0, \ldots, n-1 \tag{37}$$

it is clear that we can formulate the problem as SOCP problem. Let us define the discrete state vector as

$$X = \begin{bmatrix} x_0 & u_0 & \dots & x_n & u_n & s \end{bmatrix}^T \tag{38}$$

where the elements x_i and u_i, $i = 0, \dots, n$ are the discrete states and controls, respectively, and s is a slack variable. If we impose that

$$\|x_n\|_2 \leq s \tag{39}$$

and

$$\|u_i\|_2 \leq u_{max},\ i = 0, \dots, n \tag{40}$$

which clearly are conic constraints, the cost function becomes

$$c = \begin{bmatrix} O_{1\times 2(n+1)} & 1 \end{bmatrix}^T \tag{41}$$

Finally, the discrete dynamics will provide the matrix A and the vector b such that

$$AX = b \tag{42}$$

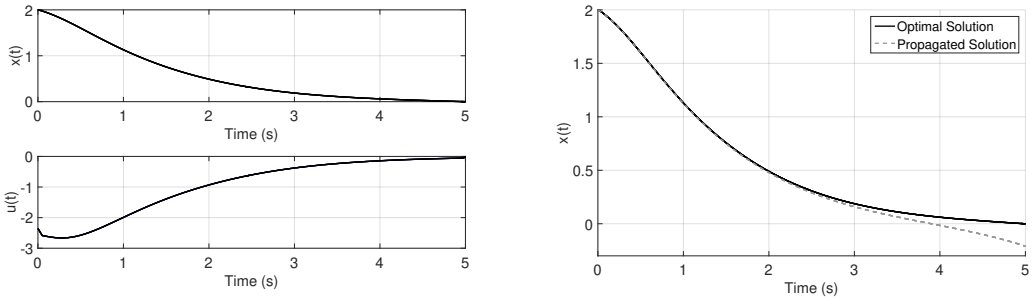
with

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & \dots & \dots & 0 \\ -(1+\frac{dt}{2}a) & -\frac{dt}{2}b & (1-\frac{dt}{2}a) & -\frac{dt}{2}b & \dots & \dots & \dots & 0 \\ 0 & 0 & -(1+\frac{dt}{2}a) & -\frac{dt}{2}b & (1-\frac{dt}{2}a) & -\frac{dt}{2}b & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \dots & -(1+\frac{dt}{2}a) & -\frac{dt}{2}b & (1-\frac{dt}{2}a) & -\frac{dt}{2}b \end{bmatrix} \tag{43}$$

and

$$b = \begin{bmatrix} x_0 & 0 & \dots & 0 \end{bmatrix}^T \tag{44}$$

Results obtained by using this discretization scheme in 100 nodes are represented in Fig. 4(a), where the state and the control are depicted.

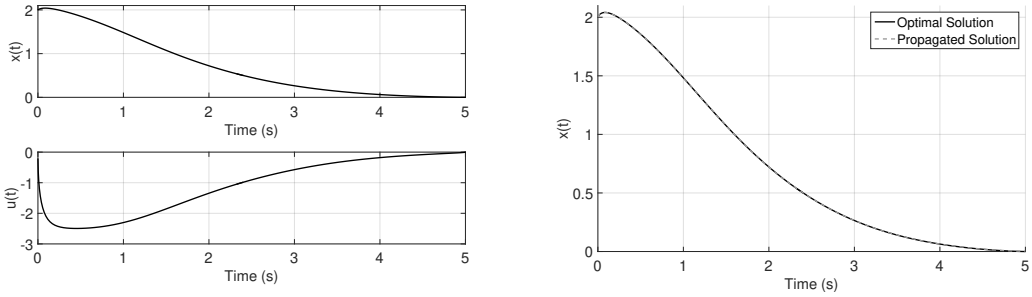


(a) SOCP solution. (b) `ode45` validation.

Fig. 4 One-dimensional problem solved with SOCP discretization - standard approach (a), and validation via Matlab's `ode45` (b).

We can see that the state is correctly driven to 0, as expected. The solution satisfies all the imposed constraints. The linear system representing the dynamics is satisfied with residuals in the order of 10^{-14}. However, a validation of the solution via Matlab's `ode45` shows a much larger error when the obtained controls are used to propagate the initial state (in this case equal to 2). The two solutions are compared in Fig. 4(b). Note that even if this is a simple application, and a relatively large number of nodes was employed, the difference becomes nontrivial. For the case analyzed here the maximum difference between the two solutions in terms of final states is equal to 0.22. We can solve the same problem with the proposed pseudospectral convex approach (the implementation is omitted here for brevity, and fully described in Sec. VI). Results obtained by using the same number of nodes are depicted in Fig. 5(a), where the state and the control are represented, and Fig. 5(b), where the comparison between optimal and propagated solutions can be seen.

In this case the difference between the solutions is reduced to 0.0022, that means 1% of the error obtained with the standard approach. Note that no difference in CPU times were observed between these examples (about 130 ms when standard transcription was employed versus 115 ms when the



(a) Pseudospectral Convex solution. (b) `ode45` validation.

Fig. 5 One-dimensional problem solved with SOCP discretization - pseudospectral convex approach (a), and validation via Matlab's `ode45` (b).

pseudospectral convex approach was used). This significant difference of accuracy motivates to apply the proposed technique to more demanding scenarios.

V. Mars Powered Descent

In 2012 NASA's rover Curiosity successfully landed on the martian surface [40]. One of the most challenging parts of the famous 7 *minutes of terror* [41] was the descent phase, where the retrorockets were used to counteract Martian gravity and ensure the proper conditions for a soft touchdown. This mission is a perfect example of how convex optimization could be applied to face complex and challenging scenarios. An elegant formulation of the Mars descent problem can be found in [11]. Specifically, the optimal-control problem can be stated as follows. We are interested in maximizing the final mass of the lander

$$\max J = m(t_f) \tag{45}$$

subject to the following set of equations:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \frac{\mathbf{T_c}}{m} + \mathbf{g} \\ \dot{m} &= -\alpha \left\| \mathbf{T_c} \right\| \end{aligned} \tag{46}$$

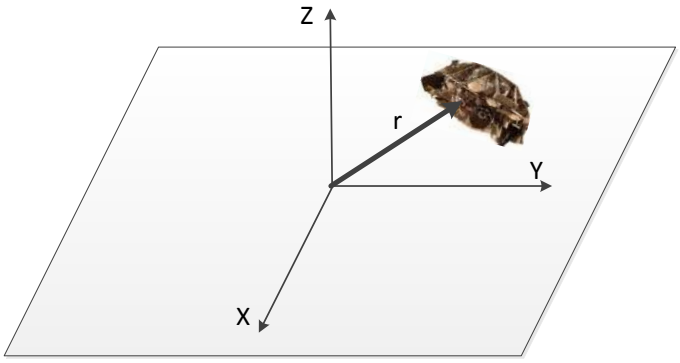


Fig. 6 Surface-fixed reference frame.

$\mathbf{r} \in \mathbb{R}^3$ 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. 6. The Martian gravity vector is defined as $\mathbf{g} = [0\ 0\ -3.7114]$ m/s^2. 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. $\mathbf{T_c} \in \mathbb{R}^3$ 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. (46) includes parameters of the thrusters' system, and is computed as

$$\alpha = \frac{1}{I_{sp} g_e \cos\phi} \tag{47}$$

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 lander 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 $T_{c,i}$ is

$$T_{c,i} = T_{max} n T_i \cos\phi,\ i = x, y, z \tag{48}$$

with T_{max} equal to 3.1 kN. Note that T_i obeys the following constraint:

$$T_l \leq T_i \leq T_u,\ i = 1, \ldots, 3 \tag{49}$$

with $T_l = 0.3$ and $T_u = 0.8$. Initial and final positions and velocities are:

$$\begin{aligned} \mathbf{r}(t_0) &= \begin{bmatrix} 2000 \\ 0 \\ 1500 \end{bmatrix} \text{ m}, \quad \mathbf{r}(t_f) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ m} \\ \mathbf{v}(t_0) &= \begin{bmatrix} 100 \\ 0 \\ -75 \end{bmatrix} \text{ m/s}, \mathbf{v}(t_f) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \text{ m/s} \end{aligned} \tag{50}$$

A further condition to be imposed is the so-called glideslope constraint:

$$\tan^{-1}\left[\frac{r_z(t)}{\sqrt{r_x^2(t) + r_y^2(t)}}\right] \geq \widetilde{\theta}_{alt} = 4 \text{ deg} \tag{51}$$

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 [11] showed that this non-convex optimal problem can be transformed into an equivalent convex one. Let us define the following variables:

$$\begin{aligned} \mathbf{u} &= \frac{\mathbf{T_c}}{m} \\ \sigma &= \frac{\Gamma}{m} \\ z &= \log(m) \end{aligned} \tag{52}$$

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

$$\min J = \int_{t_0}^{t_f} \sigma(t) dt \tag{53}$$

subject to:

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ \dot{\mathbf{v}} &= \mathbf{u} + \mathbf{g} \\ \dot{z} &= -\alpha\sigma \end{aligned} \tag{54}$$

The lossless convexification ensures the following inequality remains tight:

$$\|\mathbf{u}(t)\| \leq \sigma(t) \tag{55}$$

The change of variables of Eq. (52) implies that the following constraint acting on z has to be satisfied:

$$\rho_l e^{-z(t)} \leq \sigma(t) \leq \rho_u e^{-z(t)} \tag{56}$$

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

$$\rho_l e^{-z_l} \left[1 - (z - z_l) + \frac{1}{2}(z - z_l)^2\right] \leq \sigma(t) \leq \rho_u e^{-z_u} \left[1 - (z - z_u)\right] \tag{57}$$

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

$$\begin{aligned} z_{l,i} &= log(m_0 - \alpha\rho_l t_i),\ i = 0, \ldots, n \\ z_{u,i} &= log(m_0 - \alpha\rho_u t_i),\ i = 0, \ldots, n \end{aligned} \tag{58}$$

and the terms ρ_l and ρ_u are equal to the minimum and the maximum values of T_c. Moreover, Eq. (51) needs to be satisfied too. This constraint, together with Eq. (54) 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 [11], while further enhancements are covered in [42], [43]. In the next section we will apply the pseudospectral convex optimization algorithm to the original formulation of the problem.

VI. Pseudospectral Convex Optimization

In this section we present the pseudospectral convex framework for generating real-time capable optimal solutions for the Mars descent phase. We use the flipped Radau method and the Lobatto method, and we emphasize the differences with respect to the standard transcription methods.

A. Flipped Radau Pseudospectral Convex method

The first step is the determination of the discrete timesteps, and the state vector representing the solution. For n collocation nodes we can compute the corresponding n roots of the Radau-Legendre polynomials as defined in Eq. (6). The roots correspond to the discrete set of pseudospectral times τ_i, $i = 0, \ldots, n$, which can be converted into physical time by using the first of the affine transformations defined by Eq. (12), leading to

$$t_i = \frac{t_f - t_0}{2}\tau_i + \frac{t_f + t_0}{2},\ i = 0, \ldots, n \tag{59}$$

The discrete time vector is non-uniform, in difference to the standard transcription. For the states and the controls we propose to use the following vector:

$$X = \left[\mathbf{r}_1 \ \ \mathbf{v}_1 \ \ z_1 \ \ \mathbf{u}_1 \ \ \sigma_1 \ \ \ldots \ \ \mathbf{r}_n \ \ \mathbf{v}_n \ \ z_n \ \ \mathbf{u}_n \ \ \sigma_n \right]^T \tag{60}$$

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 X, consistently with the fact that the initial node of the fRPm is *not collocated*.

Cost function

The vector c representing the cost function will be a vector having dimensions $n(n_s + n_c) \times 1$. Of

these, only n elements, corresponding to the σ_i values, are different from zero. Therefore we have

$$c_i = \begin{cases} \dfrac{t_f - t_0}{2} w_j, \ i = j(n_s + n_c), \ k = 1, \ldots, n \\ 0 \quad \text{otherwise} \end{cases} \tag{61}$$

where w_j are the Radau quadrature weight defined in Eqs. (18),(19), and t_0 and t_f are the initial and final times, assumed known. Note that the weights were simply assumed equal to $dt = (t_f - t_0)/(n+1)$ in the standard transcription.

Dynamics

If we define the continuous state vector as

$$\mathbf{x_c} = [\mathbf{r} \quad \mathbf{v} \quad z]^T \tag{62}$$

and the control as

$$\mathbf{u_c} = [\mathbf{u} \quad \sigma]^T \tag{63}$$

the dynamics of Eq. (54) has the following state-space representation:

$$\mathbf{A_c} = \begin{bmatrix} O_{3\times3} & I_3 & 0 \\ O_{3\times3} & O_{3\times3} & O_{3\times1} \\ O_{1\times3} & O_{1\times3} & 0 \end{bmatrix}, \quad \mathbf{B_c} = \begin{bmatrix} O_{3\times3} & 0 \\ O_{3\times3} & 0 \\ O_{1\times3} & -\alpha \end{bmatrix} \tag{64}$$

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 in 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

$$\mathbf{x}(k+1) = \mathbf{A_d}\mathbf{x}(k) + \mathbf{B_d}\mathbf{u}(k) + \mathbf{B_d}\mathbf{g} \tag{65}$$

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

$$\dot{\mathbf{x}}(t) = \mathbf{A_c}\mathbf{x_c}(t) + \mathbf{B_c}\mathbf{u_c}(t) + \mathbf{B_c}\mathbf{g} \tag{66}$$

since $\dot{\mathbf{x}} \cong \mathbf{D}\mathbf{x}$, and keeping in mind Eq. (22) we can write

$$(\mathbf{D} - k_t\mathbf{A_c})\mathbf{x_c}(t) - k_t\mathbf{B_c}\mathbf{u_c}(t) = k_t\mathbf{B_c}\mathbf{g} \tag{67}$$

which, evaluated in the n nodes leads to the following definitions

$$\mathbf{A_{0,dyn}} = \begin{bmatrix} D_{1,1}I_{n_s} - k_tA_c & -k_tB_c & \dots & \dots & D_{1,n}I_{n_s} & O_{n_s\times n_c} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ D_{n,1}I_{n_s} & O_{n_s\times n_c} & \dots & \dots & D_{n,n}I_{n_s} - k_tA_c & -k_tB_c \end{bmatrix} \tag{68}$$

$$\mathbf{b_{0,dyn}} = \begin{bmatrix} -D_{1,0}x_0 + k_tB_cg \\ \vdots \\ \vdots \\ -D_{n,0}x_0 + k_tB_cg \end{bmatrix} \tag{69}$$

The term k_t is defined as $(t_f - t_0)/2$. Note that the knowledge of the initial conditions is exploited to construct the vector $\mathbf{b_{0,dyn}}$ through the first column of the matrix $\mathbf{D}$, representing the discrete, non-collocated point corresponding to x_0.

Final Conditions

Arbitrary final conditions can be met by imposing further terms in the system of linear equations.

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 $\mathbf{A_{0,fc}}$, and a further vector $\mathbf{b_{0,fc}}$ as

$$\mathbf{A_{0,fc}} = \left[\begin{array}{cccccc} O_{(n_s-1)\times n_s} & O_{(n_s-1)\times n_c} & \cdots & \cdots & I_{(n_s-1)} & O_{(n_s-1)\times n_c+1} \end{array} \right] \tag{70}$$

$$\mathbf{b_{0,fc}} = \left[\begin{array}{cc} \mathbf{r_f} & \mathbf{v_f} \end{array} \right]^T \tag{71}$$

Remark 3 Note that the number of rows of $\mathbf{A_{0,fc}}$ and elements of $\mathbf{b_{0,fc}}$ are in this case equal to $n_s - 1$ because the final mass is not constrained.

Remark 4 The number of rows of $\mathbf{A_{0,fc}}$ and elements of $\mathbf{b_{0,fc}}$, 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

$$\mathbf{A_0} X = \mathbf{b_0} \tag{72}$$

where

$$\mathbf{A_0} = \left[\begin{array}{cc} \mathbf{A_{0,dyn}} & \mathbf{A_{0,fc}} \end{array} \right]^T, \ \mathbf{b_0} = \left[\begin{array}{cc} \mathbf{b_{0,dyn}} & \mathbf{b_{0,fc}} \end{array} \right]^T \tag{73}$$

Constraints

As first step we need to include the condition described by Eq. (55). This is done by including the following conic constraint:

$$\left\| \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array} \right] \left[\begin{array}{c} u_x \\ u_y \\ u_z \\ \sigma \end{array} \right]_i \right\|_2 \leq \left[\begin{array}{cccc} 0 & 0 & 0 & 1 \end{array} \right] \left[\begin{array}{c} u_x \\ u_y \\ u_z \\ \sigma \end{array} \right]_i, \ i = 1, \ldots, n \tag{74}$$

The glideslope constraint will be represented by the following conic inequality:

$$\left\| \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}_i \right\|_2 \leq \begin{bmatrix} 0 & 0 & \frac{1}{\tan\theta_{alt}} \end{bmatrix} \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix}_i, \; i = 1, \ldots, n \tag{75}$$

The discrete version of the left-hand side of Eq. (57) can be modeled as a conic constraint too. Let us define the following matrices and vectors:

$$A_\rho = \begin{bmatrix} \rho_l e^{-z_l} \frac{\sqrt{2}}{2} & 0 \end{bmatrix}_i, \; b_\rho = -\begin{bmatrix} \rho_l e^{-z_l} + z_l & 1 \end{bmatrix}_i, \; c_\rho = \left[\rho_l e^{-z_l}\left(1 + z_l + \tfrac{1}{2}z_l^2\right)\right]_i \tag{76}$$

and

$$A_c = \begin{bmatrix} \frac{b_\rho}{2} \\ A_\rho \end{bmatrix}, \; b_c = \begin{bmatrix} \frac{c_\rho}{2} + \frac{1}{2} \\ 0 \end{bmatrix}, \; c_c = -\tfrac{b_\rho}{2}, \; d_c = \tfrac{1}{2} - \tfrac{c_\rho}{2} \tag{77}$$

With these definitions, it is possible to impose the first part of Eq. (57) as

$$\|A_c \tilde{z} + b_c\|_2 \leq c_c^T \tilde{z} + d_c, \; i = 1, \ldots, n \tag{78}$$

where

$$\tilde{z} = \begin{bmatrix} z \\ \sigma \end{bmatrix}_i \tag{79}$$

Finally, the right-hand side of Eq. (57) is a linear constraint, and using the definition of Eq. (79), is discretized as

$$\begin{bmatrix} \rho_u e^{-z_u} & 1 \end{bmatrix} \tilde{z} \leq \rho_u e^{-z_u}\left(1 + z_u\right)\tilde{z}, \; i = 1, \ldots, n \tag{80}$$

The entire problem is therefore expressed as

$$\min J = c_0 X \tag{81}$$

subject to the linear system of Eq. (72), together with the constraints of Eqs. (74)-(80), and represents the transcription of the Mars powered descent problem according to the flipped Radau Pseudospectral Convex method (or 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.

B. Lobatto Pseudospectral Convex method

Let us define now the Lobatto Pseudospectral Convex method (or LPCm). For n nodes, the following vector is used:

$$X = \left[\mathbf{r}_0 \;\; \mathbf{v}_0 \;\; z_0 \;\; \mathbf{u}_0 \;\; \sigma_0 \;\; \dots \;\; \mathbf{r}_{n-1} \;\; \mathbf{v}_{n-1} \;\; z_{n-1} \;\; \mathbf{u}_{n-1} \;\; \sigma_{n-1} \right]^T \tag{82}$$

In this case all the discrete nodes are collocated. This implies that we have to include the initial conditions in the optimization process, and that we have to constrain them to be equal to the assigned initial conditions of our problem.

Cost function

The cost function is formally identical to the one of Eq. (61), with the only difference that the weights are computed according to the definition given in Eq. (20).

Dynamics

The matrix $\mathbf{A_{0,dyn}}$ and the vector $\mathbf{b_{0,dyn}}$ have the same dimensions of the previous case, but they are slightly different. Now the differentiation matrix $\mathbf{D}$ is squared of dimensions $n \times n$. Since there

are no discrete, non-collocated nodes, the known vector $\mathbf{b_{0,dyn}}$ does not contain the initial states:

$$\mathbf{A_{0,dyn}} = \begin{bmatrix} D_{1,0}I_{n_s} - k_t A_c & -k_t B_c & \dots & \dots & D_{1,n-1}I_{n_s} & O_{n_s \times n_c} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ D_{n,0}I_{n_s} & O_{n_s \times n_c} & \dots & \dots & D_{n,n-1}I_{n_s} - k_t A_c & -k_t B_c \end{bmatrix} \tag{83}$$

$$\mathbf{b_{0,dyn}} = \begin{bmatrix} k_t B_c g \\ \vdots \\ \vdots \\ k_t B_c g \end{bmatrix} \tag{84}$$

Final Conditions

As for the fRPCm we impose the final conditions in the system of linear equations. The matrix $\mathbf{A_{0,fc}}$ and the vector $\mathbf{b_{0,fc}}$ remain unchanged with respect to Eqs. (70) and (71).

Initial Conditions

Since the initial states are now variables, we have to impose that they are equal to the known initial conditions. In a similar fashion to what was done in Eqs. (70) and (71) these constraints are ensured by defining a matrix $\mathbf{A_{0,ic}}$ and a vector $\mathbf{b_{0,ic}}$ as

$$\mathbf{A_{0,ic}} = \begin{bmatrix} I_{n_s \times n_s} & O_{n_s \times n_c} & \dots & \dots & O_{n_s \times n_s} & O_{n_s \times n_c} \end{bmatrix} \tag{85}$$

$$\mathbf{b_{0,ic}} = \begin{bmatrix} \mathbf{r_0} & \mathbf{v_0} & z_0 \end{bmatrix}^T \tag{86}$$

Remark 5 The number of rows in this case is equal to n_s, as all of the initial conditions are assigned.

The linear system representing the dynamics, the final and the initial conditions is therefore

given by the following condition

$$\mathbf{A_0} X = \mathbf{b_0} \tag{87}$$

with

$$\mathbf{A_0} = \begin{bmatrix} \mathbf{A_{0,ic}} & \mathbf{A_{0,dyn}} & \mathbf{A_{0,fc}} \end{bmatrix}^T, \ \mathbf{b_0} = \begin{bmatrix} \mathbf{b_{0,ic}} & \mathbf{b_{0,dyn}} & \mathbf{b_{0,fc}} \end{bmatrix}^T \tag{88}$$

Constraints

The constraints are assigned exactly in the same way as for the fRPCm. Therefore, Eqs. (74) through (80) hold for the LPCm too, for $i = 0, \ldots, n-1$. The initial states are trivially satisfied, and the initial control are directly computed by the optimizer. The transcription is therefore complete and no further actions are required.

VII. Numerical Performances

A series of simulations to assess the performance of the proposed methods was performed. The solution associated with 50 nodes is depicted in Figs. 7-9. We can see from Fig. 7, (showing position, velocity, acceleration and controls) that the solution is fully consistent with the results of [11]. The glideslope constraint is also fully satisfied (Fig. 8). The final mass consumption is equal to 399.5 kg. Figure 9 shows the control history: the original non-convex control constraints are satisfied too.

To perform a more systematic analysis of the results the problem has been solved for each of the three transcription methods (i.e., fRPCm, LPCm and standard transcription) with two different SOCP solvers (ECOS [39], and SDPT3 [44]) by varying the number of nodes between 40 and 120. The comparison has been performed in terms of

- Cost function
- Mean and maximum error between optimal and propagated solutions (ode45)
- CPU time

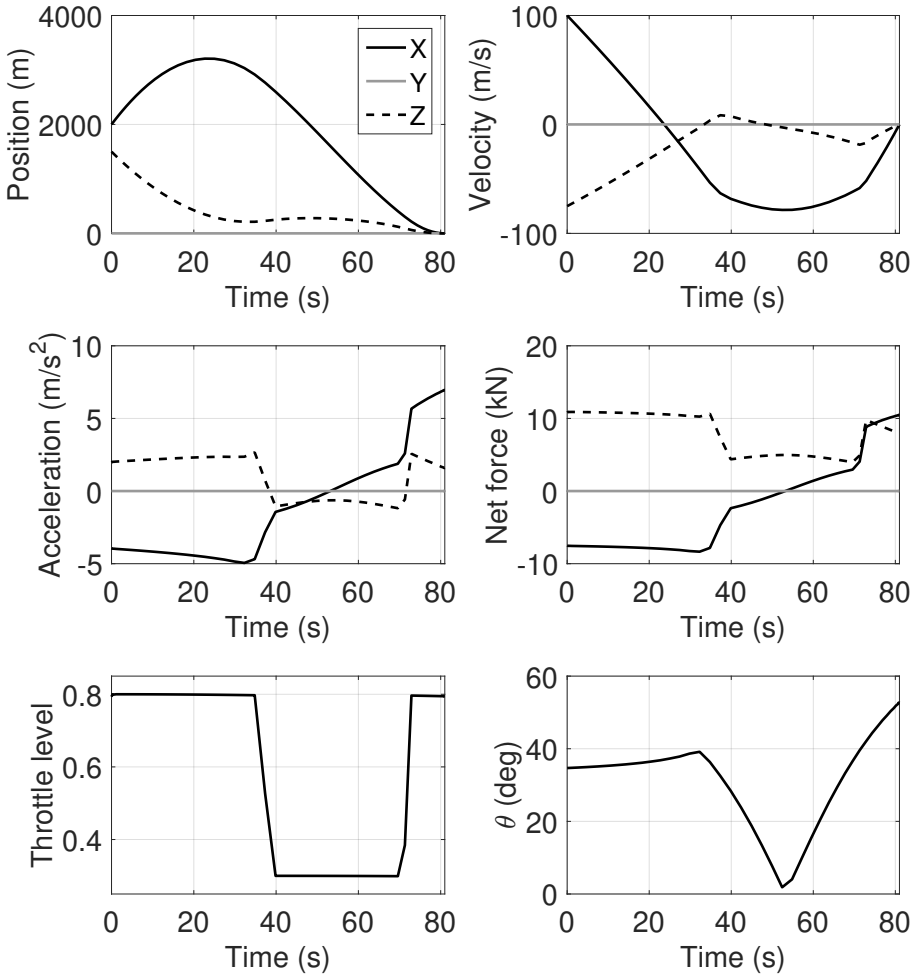


Fig. 7 Solution obtained with flipped Radau pseudospectral convex method - states and controls.

For a better characterization of the CPU times each run has been repeated 10 times. All the cases have been run on a laptop having a i7-368 processor with clock frequency of 2.6 GHz. Results are depicted from Fig. 10(a) through 15(b). In terms of fuel usage (Figs. 10(a) and 10(b)) we can observe that for both ECOS and SDPT3 the pseudospectral convex framework generates better cost indices, even with smaller number of nodes, where the difference can be up to 0.9 kg. Similar differences can be observed for the mean errors on position (Figs. 11(a) and 11(b)) and velocity (Figs. 12(a) and 12(b)), where the proposed methods leads to much better results.

More specifically, in terms of positions (Figs. 11(a) and 11(b)) the mean error ranges from 48 m (for $n = 40$) to 15 m ($n = 120$). When pseudospectral convex optimization is used these

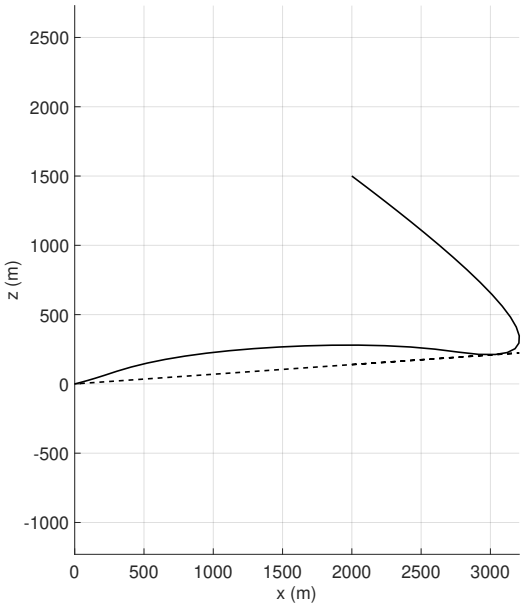


Fig. 8 Solution obtained with flipped Radau pseudospectral convex method - trajectory.

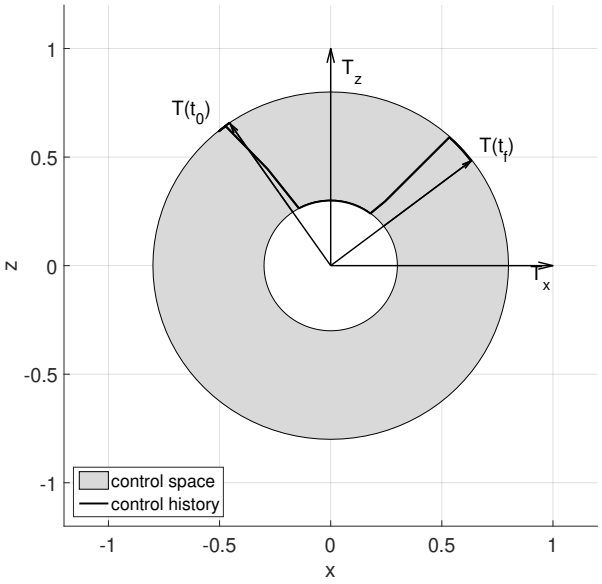


Fig. 9 Solution obtained with flipped Radau pseudospectral convex method - control space.

errors are 3.4 and 0.36 m, respectively. Note also that the error is not only smaller if compared with the standard methods, but also decreases more rapidly, for both fRPCm and LPCm. Similar conclusions can be drawn for the velocity errors (Figs. 12(a) and 12(b)), which are reduced up to 25 times (3.05 m/s for the standard methods against 0.10 m/s when pseudospectral convex methods

are employed) for $n = 40$. The ratio increases up to 83 times (1 m/s versus 0.012 m/s) when n becomes equal to 120. The differences become even larger when the maximum values are taken. For the positions indeed (Fig. 13(a) and 13(b) the error is reduced to 3.8% of the value obtained with standard methods when $n = 40$ nodes are taken, and to 1% when n is equal to 120. This percentage is more or less constant, and equal to 5% when the maximum velocity errors are considered, with reference to Figs. 14(a) and 14(b).

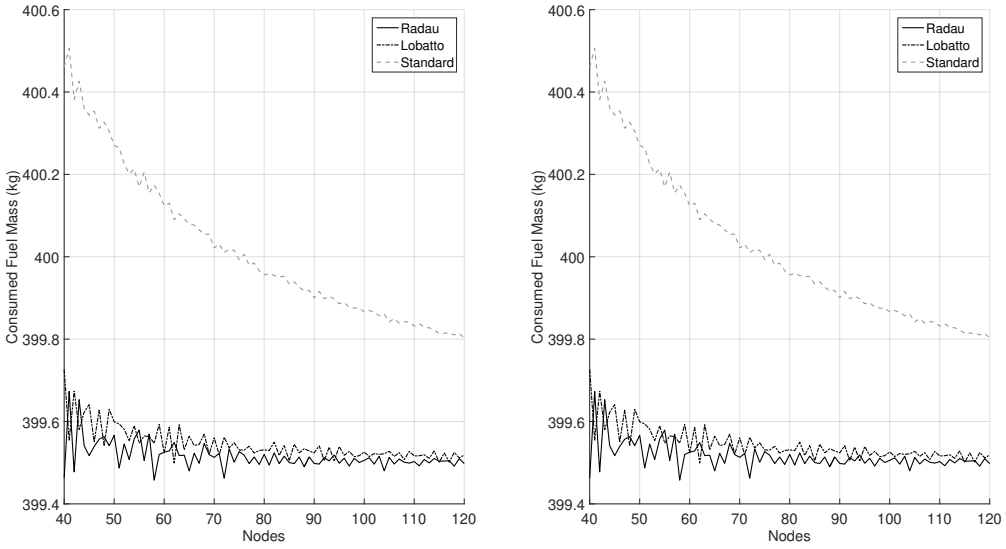
From Fig. 15(a) we can observe that with ECOS the proposed methods are slower than the standard technique, even if quite efficient for small number of nodes. Moreover, the CPU time for the standard method is less sensitive to the increase in the size of the problem until n = 120, with an observed CPU time between 25 and 139 ms. For the two pseudospectral methods the CPU times are between 217 (best case, obtained with LPm) and 5358 ms (worst case, associated with the use of the fRPm). This is mainly due to the largest number of interactions between the several discrete states (note that the differentiation matrix $\mathbf{D}$ creates dependencies among all of them, while the numerical scheme used in the standard methods implies that only two consecutive discrete states are linearly dependent with respect to each other.

A different scenario is observed when SDPT3 is adopted (Fig. 15(b)). In this case the time differences significantly decrease, and the CPU times are generally larger than in the previous campaign. When standard transcription is used the CPU times range from 763 to 2156 ms, against a range of [1998, 13570] ms obtained when the fRPCm is employed. It is interesting to observe that these results are much smoother and consistent with the number of nodes w.r.t. the ones obtained by using the LPCm. In fact, this method shows much larger oscillations between 40 and 50 nodes, as well as between 82 and 90, and above 110 nodes. A deeper analysis of these results revealed that, although the solution is always valid and corresponds to the expected one, for those cases the SDPT3 solver has convergence troubles. The same issue affects the standard transcription in the ranges [60 80] and [100 120], while this issue never affects the Radau-based method, suggesting that this one is preferable as more numerically stable, and accurate as the Lobatto one.

All these results are also summarized in Tables 1 and 2, which provide a quick overview of the performance of the proposed methods. Specifically, as previously mentioned, SDPT3 computes the

solution in a generally larger amount of time. This is mainly due to the fact the while ECOS is specialized for SOCP, SDPT3 is conceived for solving semidefinite programming (SDP) problems, which is a larger branch of convex optimization, and therefore it is slightly less optimized in handling SOCP problems. In general what we can observe is that for a small number of nodes the difference of time between standard and proposed methods is reduced (it takes between 1.2 and 13 times more), but the accuracy in position is between 15 and 20 times better, and in terms of velocity the error is reduced by a factor varying between 15 and 20 times. This improvement at a reduced CPU cost makes the proposed algorithms a valid alternative for this range of nodes. **Remark 6** Note that, despite the use of CVX, the CPU times depicted in Fig. 15(b) are the ones obtained by SDPT3 only, i.e., without taking the parsing time into account.

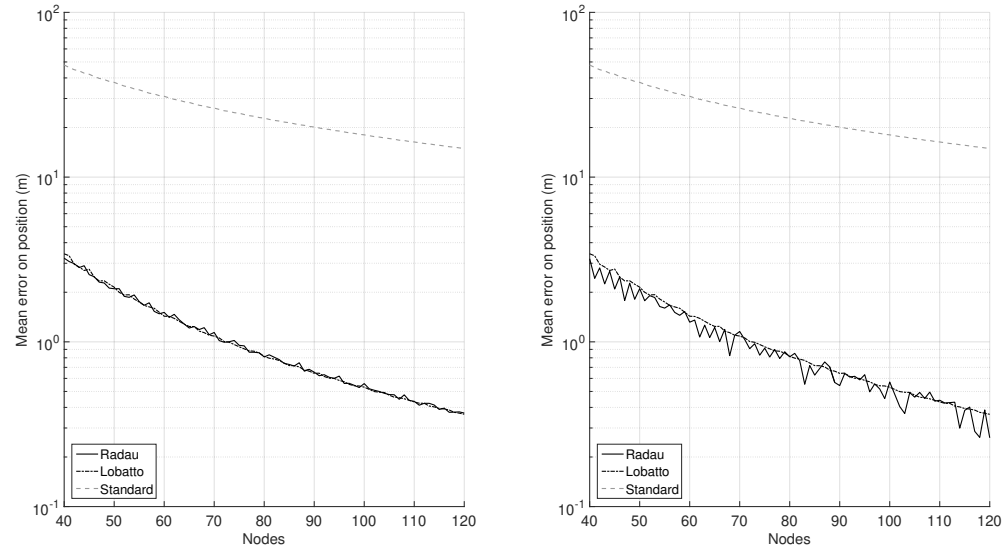
For what regards the differences between the two proposed methods, even if both perform well, the use of the fRPCm is recommended for three reasons: first, it is more accurate than LPCm in integrating the cost function in virtue of the higher order accuracy of its quadrature formula. Second, for the flipped Radau points a direct connection between the discrete Lagrange multipliers and the continuous costates of the continuous OCP holds in virtue of the Covector Mapping Theorem, while this is not true for the Lobatto-based framework [21, 29]. Finally, the fRPCm does not show any numerical issue, which might affect the Lobatto method. All these reasons make the fRPCm more promising for future applications.



(a) Fuel usage (ECOS).

(b) Fuel usage (SDPT3).

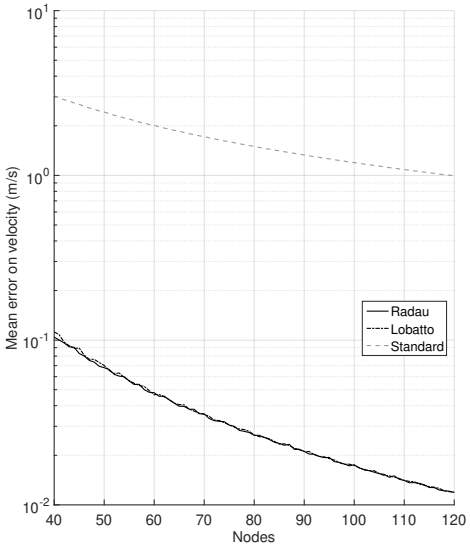
Fig. 10 Performance comparison - Cost function obtained with different solvers.



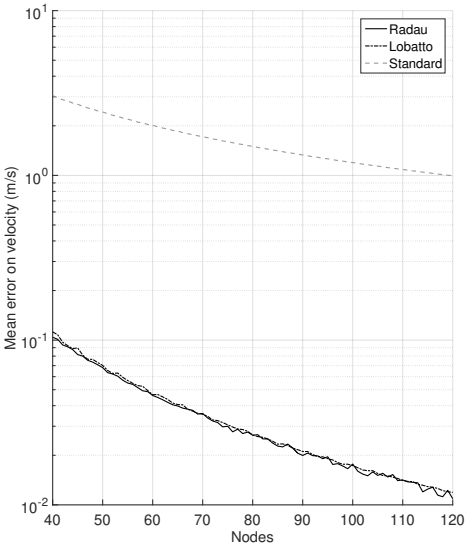
(a) Mean error on position (ECOS).

(b) Mean error on position (SDPT3).

Fig. 11 Performance comparison - Error on position obtained with different solvers.

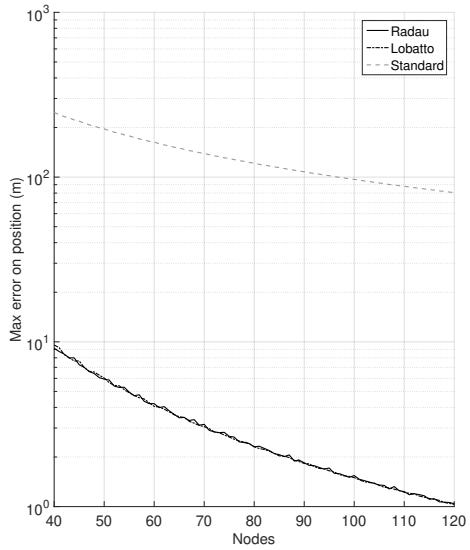


(a) Mean error on velocity (ECOS).

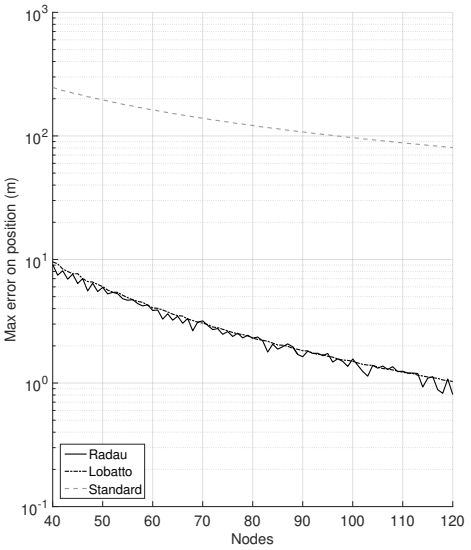


(b) Mean error on velocity (SDPT3).

Fig. 12 Performance comparison - Error on velocity obtained with different solvers.

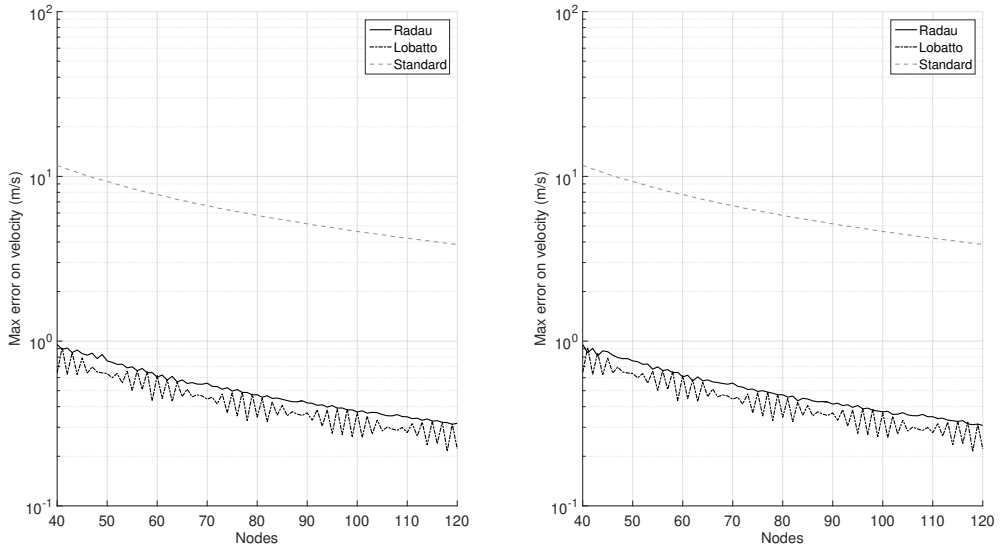


(a) Max error on position (ECOS).



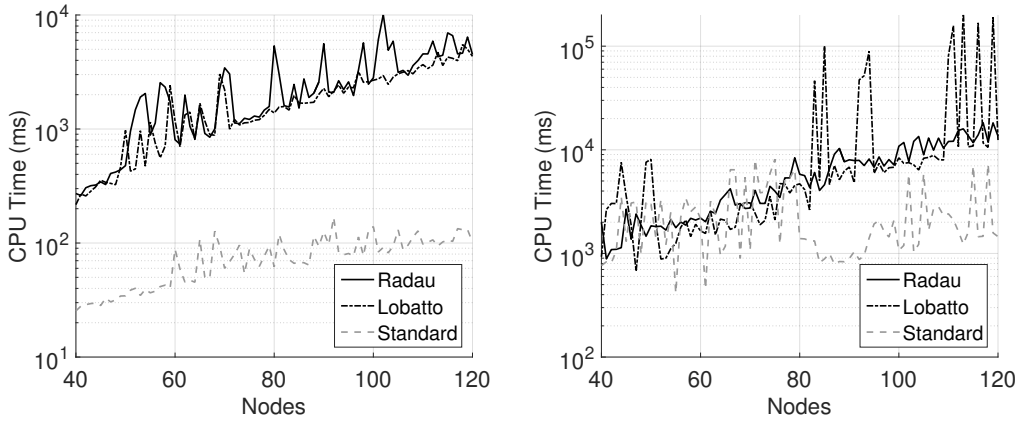
(b) Max error on position (SDPT3).

Fig. 13 Performance comparison - Error on position obtained with different solvers.



(a) Max error on velocity (ECOS). (b) Max error on velocity (SDPT3).

Fig. 14 Performance comparison - Error on velocity obtained with different solvers.



(a) CPU time (ECOS). (b) CPU time (SDPT3).

Fig. 15 Performance comparison - CPU times obtained with different solvers.

Table 1 Performance obtained with ECOS

Nodes	Method	mean CPU time (ms)	mean pos error (m)	mean vel error (m/s)
40	fRPCm	271	3.22	0.105
	LPCm	217	3.42	0.112
	SCm	25	47.91	3.051
60	fRPCm	805	1.51	0.048
	LPCm	1174	1.43	0.047
	SCm	88	30.95	2.014
80	fRPCm	5358	0.81	0.026
	LPCm	1381	0.82	0.027
	SCm	62	22.81	1.501
100	fRPCm	2808	0.56	0.018
	LPCm	2678	0.53	0.017
	SCm	139	18.05	1.196
120	fRPCm	4498	0.37	0.012
	LPCm	4337	0.36	0.012
	SCm	102	14.92	0.994

Table 2 Performance obtained with SDPT3

Nodes	Method	mean CPU time (ms)	mean pos error (m)	mean vel error (m/s)
40	fRPCm	1998	3.19	0.104
	LPCm	928	3.42	0.112
	SCm	763	47.91	3.051
60	fRPCm	2194	1.32	0.046
	LPCm	1652	1.43	0.047
	SCm	2156	30.95	2.014
80	fRPCm	5803	0.81	0.026
	LPCm	4696	0.82	0.027
	SCm	1391	22.81	1.501
100	fRPCm	10830	0.57	0.018
	LPCm	8360	0.53	0.017
	SCm	1070	18.05	1.196
120	fRPCm	13570	0.26	0.011
	LPCm	12420	0.36	0.012
	SCm	1453	14.92	0.994

VIII. Conclusions

In this paper, pseudospectral methods and convex optimization are combined to provide a more accurate real-time oriented framework for optimal control. Two pseudospectral methods are employed for this hybridization, leading to the flipped Radau pseudospectral convex method and to the Lobatto pseudospectral convex method. The proposed approaches are applied to the Mars powered descent scenario and compared with standard convex methods. The comparison is performed in terms of fuel usage, as well as position and velocity errors, and CPU time. Two different solvers (that is, ECOS and SDPT3) to assess the results in a more general context are used.

Both the proposed approaches lead to results that can be up to 100 times more accurate than the ones obtained with standard methods. The difference in the accuracy of the results is also significant for the cases where a small number of nodes (40âĂŞ50) was considered. This range of nodes is characterized by having CPU times on the order of about 220âĂŞ800 ms. This subset of nodes therefore represents the region where the application of pseudospectral convex optimization provides a large improvement of accuracy at the price of a reasonably increased computational time.

Acknowledgments

The author thanks the DLR Technology Marketing department, the Jostarndt AG Patent Attorneys and the German Patent OFfice (DPMA) for the strong support in filing the applicatin for the patent 102017219076.0.

References

[1] Blackmore, L., "Autonomous Precision Landing of Space Rockets," *The Bridge*, Vol. 4, 2016.

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

[3] Connor, M. A., "Gravity turn trajectories through the atmosphere." *Journal of Spacecraft and Rockets*, Vol. 3, No. 8, Aug. 1966, pp. 1308–1311.

[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] 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.

[10] 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, 2015, pp. 227–241.

[11] 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.

[12] Acikmese, B., Carson, J. M., and Blackmore, L., "Lossless Convexification of Nonconvex Control Bound and Pointing Constraints of the Soft Landing Optimal Control Problem," *IEEE Transactions on Control Systems Technology*, Vol. 21, No. 6, nov 2013, pp. 2104–2113.

[13] Dueri, D., Acikmese, B., Scharf, D. P., and Harris, M. W., "Customized Real-Time Interior-Point Methods for Onboard Powered-Descent Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, Feb. 2017, pp. 197–212.

[14] Scharf, D. P., Acikmese, B., Dueri, D., Benito, J., and Casoliva, J., "Implementation and Experimental Demonstration of Onboard Powered-Descent Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 2, Feb. 2017, pp. 213–229.

[15] Acikmese, B., Aung, M., Casoliva, J., Mohan, S., Johnson, A., Scharf, D., Masten, D., Scotkin, J., Wolf, A., and Regehr, M. W., "Flight testing of trajectories computed by G-FOLD: Fuel optimal large divert guidance algorithm for planetary landing," *Advances in the Astronautical Sciences*, Vol. 148, 2013, pp. 1867–1880.

[16] Huneker, L., Sagliano, M., and Arslantas, Y., "SPARTAN: An Improved Global Pseudospectral Algorithm for High-Fidelity Entry-Descent-Landing Guidance Analysis," *30^{th} International Symposium on Space Technology and Science, Kobe, Japan, 2015*, 2015.

[17] Sagliano, M., Theil, S., D'Onofrio, V., and Bergsma, M., "SPARTAN: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis," *4^{th} CEAS Eurognc conference, Warsaw*, 2017.

[18] Sagliano, M., Theil, S., Bergsma, M., D'Onofrio, V., Whittle, L., and G., V., "On the Radau Pseudospectral Method: theoretical and implementation advances," *CEAS SPACE Journal*, Vol. 9, No. 3,

jun 2017, pp. 313–331.

[19] Sagliano, M., *Development of a Novel Algorithm for High Performance Reentry Guidance*, phdthesis, 2016.

[20] 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.

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

[22] Sagliano, M. and Theil, S., "Hybrid Jacobian Computation for Fast Optimal Trajectories Generation," *AIAA Guidance, Navigation, and Control Conference, Boston, USA,*, 2013.

[23] 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.

[24] 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.

[25] 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.

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

[27] Sagliano, M., "Performance analysis of linear and nonlinear techniques for automatic scaling of discretized control problems," *Operations Research Letters, Vol.42 Issue 3, May 2014, pp. 213-216*, 2014.

[28] D'Onofrio, V., Sagliano, M., and Arslantas, Y., "Exact Hybrid Jacobian Computation for Optimal Trajectory Computation via Dual Number Theory," *AIAA Science and Technology Forum and Exposition*, 2016.

[29] 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.

[30] 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.

[31] 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 245Ũ262*, 2003.

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

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

[34] El Ghaoui, L., "Introduction to Convex Optimization," .

[35] Mattingley, J. and S., B., "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, , No. 13, 2012, pp. 1–27.

[36] 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.

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

[38] Storm, J., "Using SeDuMi 1.02, A Matlab Toolbox for Optimization over Symmetric Cones," *Optimization Methods and Software*, 1999.

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

[40] 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.

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

[42] 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.

[43] 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.

[44] Toh, K.-C., Todd, M. J., and Tütüncü, R. H., *On the Implementation and Usage of SDPT3 – A Matlab Software Package for Semidefinite-Quadratic-Linear Programming, Version 4.0*, Springer US, Boston, MA, 2012, pp. 715–754.