JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 59, No. 3, May–June 2022

Convex Optimization of Launch Vehicle Ascent Trajectory with Heat-Flux and Splash-Down Constraints

Boris Benedikter,* Alessandro Zavoli,† and Guido Colasurdo‡
Sapienza University of Rome, 00184 Rome, Italy
and
Simone Pizzurro§ and Enrico Cavallini¶
Italian Space Agency, 00133 Rome, Italy

https://doi.org/10.2514/1.A35194

This paper presents a convex programming approach to the optimization of a multistage launch vehicle ascent trajectory, from the liftoff to the payload injection into the target orbit, taking into account multiple nonconvex constraints, such as the maximum heat flux after fairing jettisoning and the splash-down of the burned-out stages. Lossless and successive convexification methods are employed to convert the problem into a sequence of convex subproblems. Virtual controls and buffer zones are included to ensure the recursive feasibility of the process, and a state-of-the-art method for updating the reference solution is implemented to filter out undesired phenomena that may hinder convergence. A *hp* pseudospectral discretization scheme is used to accurately capture the complex ascent and return dynamics with a limited computational effort. The convergence properties, computational efficiency, and robustness of the algorithm are discussed on the basis of numerical results. The ascent of a VEGA-like launch vehicle toward a polar orbit is used as a case study to discuss the interaction between the heat flux and splash-down constraints. Finally, a sensitivity analysis of the launch vehicle carrying capacity to different splash-down locations is presented.

Nomenclature

A_e	=	nozzle exit area, m^2
$\boldsymbol{D}$	=	aerodynamic drag vector, kN
$\boldsymbol{f}$	=	right-hand side of the equations of motion
J	=	problem cost function
m	=	mass, kg
p	=	atmospheric pressure, kPa
$\dot{Q}$	=	heat flux, W/m^2
$\boldsymbol{q}$	=	virtual control
$\boldsymbol{r}$	=	position vector, km
$\boldsymbol{T}$	=	thrust vector, kN
t	=	time, s
$\boldsymbol{u}$	=	control vector
$\boldsymbol{v}$	=	velocity vector, km/s
w	=	virtual buffer
$\boldsymbol{x}$	=	state vector
η	=	transformed independent variable in $[-1, 1]$
μ	=	Earth gravitational parameter, km^3/s^2
ρ	=	atmospheric density, kg/m^3
τ	=	transformed independent variable in $[0, 1]$
φ	=	latitude, deg
ϕ, ψ	=	elevation and azimuth, deg
$\boldsymbol{\omega}_E$	=	Earth angular velocity vector, rad/s

Subscripts

des	=	desired condition
f	=	final boundary
vac	=	in vacuum
0	=	initial boundary

Superscripts

(i)	=	relative to phase i
$\hat{}$	=	unit vector
$\bar{}$	=	evaluated on the reference solution
$'$	=	derivative with respect to τ
$\star$	=	optimal solution

I. Introduction

IN THE present state-of-the-art, the only propulsion system capable of providing the high thrust required for access to space is the chemical one. However, this system allows for injecting into orbit only a small fraction of the rocket initial mass. Therefore, the ascent trajectory optimization process is of primary interest in order to increase the launcher capacity and reduce the overall mission cost. Besides, the need for a reliable and efficient optimization tool is apparent in the preliminary design phases of a launch vehicle, to evaluate the performance of various configuration concepts, in the advanced preflight analysis, to assess the feasibility of specific mission scenarios, and in the definition of optimization-based real-time guidance algorithms, where computational speed and robustness are primary requirements.

The design of a rocket ascent trajectory is a complex optimal control problem (OCP), greatly sensitive to the optimization variables and characterized by highly nonlinear dynamics and numerous mission requirements. Over the years, various optimization methods have been proposed to solve the ascent problem. Jurovics [1] was one of the first to propose an indirect approach. An indirect procedure also underlies the well-known DUKSUP optimization software [2], which has been extensively employed by NASA in the design of the Atlas, Titan, and Space Shuttle launch systems. More recent work

Received 24 June 2021; revision received 22 October 2021; accepted for publication 26 October 2021; published online 22 December 2021.

*Ph.D. Student, Department of Mechanical and Aerospace Engineering, Via Eudossiana 18; boris.benedikter@uniroma1.it.

†Research Assistant, Department of Mechanical and Aerospace Engineering, Via Eudossiana 18; alessandro.zavoli@uniroma1.it.

‡Full Professor, Department of Mechanical and Aerospace Engineering, Via Eudossiana 18; guido.colasurdo@uniroma1.it.

§Research Fellow, Space Transportation, Space Infrastructures, and In-Orbit Servicing Department, Via del Politecnico snc; simone.pizzurro@est.asi.it.

¶Head of Space Transportation Programs Office, Space Transportation, Space Infrastructures, and In-Orbit Servicing Department, Via del Politecnico snc; enrico.cavallini@asi.it.

based on the indirect method includes both pure trajectory design applications [3–5] and closed-loop guidance algorithms [6–8]. However, when dealing with real-world launch vehicle missions, the indirect method may be unappealing as it requires the derivation of the optimality conditions, which can be a burdensome task, and the solution of the resulting boundary value problem requires a meticulous initialization process to achieve convergence. In addition, if path constraints are included in the formulation, an a priori knowledge of the structure of the constrained arcs is necessary, which, in general, is hard to guess. To mitigate this issue, a hybrid analytic/numerical procedure has been developed by Gath and Calise [9]. Nevertheless, despite all the efforts, direct methods are typically preferred.

A wide spectrum of direct optimization software tools has been developed for solving aerospace problems, including traditional nonlinear programming (NLP) solvers, such as POST [10], OTIS [11], and ASTOS [12], and more recent programs that leverage pseudospectral methods to set up accurate and efficient optimization algorithms, such as GPOPS [13], SPARTAN [14], and DIDO [15]. Direct methods have been used even for solving problems very similar to the one addressed in this paper. Spangelo and Well [16] described a direct formulation of the ascent problem that takes into account the maximum heat flux and constrains the return of a spent stage by bounding the perigee of its osculating orbit at burnout. Instead, later work by Weigel and Well [17] includes a complete simulation of the return of the burned-out stage as an additional phase and, then, solves the resulting OCP via direct multiple shooting. However, these approaches essentially consist in transcribing the continuous-time OCP into a general NLP problem and, despite being easy to set up, this frequently leads to a solution that depends on provided first guess. This is a burdensome drawback for the problem at hand because designing an accurate initialization may be nontrivial due to the aforementioned sensitivity of the problem. Moreover, solving a general NLP problem is a computationally expensive task, with no guarantee on the optimality of the attained solution, as first-order necessary conditions are imposed on the discrete problem only.

Convex optimization techniques are becoming increasingly popular for solving OCPs in the aerospace community [18]. Convex optimization is a special class of mathematical programming that allows for the use of polynomial-time algorithms that provide a theoretically guaranteed optimal solution with a limited computational effort. However, because most aerospace problems are not naturally convex, several *convexification* techniques have been developed to convert a nonconvex problem into a convex one. These methods are grouped into *lossless* and *successive* convexification techniques. The former consist in exploiting either a convenient change of variables or a suitable constraint relaxation to reformulate the problem as convex. For example, Açkmeşe and Blackmore [19] proved that problems with a certain class of nonconvex control constraints can be posed equivalently as relaxed convex problems. Instead, successive convexification offers a way to tackle the nonconvexities that cannot be handled by lossless convexification through linearization around a reference solution that is recursively updated. Differently from lossless convexification, the successive linearization generates a sequence of approximated subproblems. The theoretical proof that also successive convexification leads to a (locally) optimal solution of the originally intended problem is available only under appropriate assumptions [20–22]. Nevertheless, current research offers wide numerical evidence of the effectiveness of successive convexification over a broad spectrum of applications, including spacecraft rendezvous [23], proximity operations [24], formation flying [25], low-thrust transfers [26,27], rocket powered landing [28–31], and atmospheric entry [32–35]. Convex optimization has been proposed also for solving the launch vehicle ascent trajectory problem. However, successful applications are limited to simplified scenarios, where a flat Earth is assumed [36], atmospheric forces are neglected [37,38], or only the upper stage trajectory is optimized [20,39].

In this paper, a realistic dynamic model, which accounts for a Keplerian gravitational model and nonlinear aerodynamic forces, is considered, and the complete ascent problem is solved via convex optimization, building up on the authors' previous work [40,41]. A VEGA-like launch vehicle is taken as case study, but the method can be easily extended to any other rocket. VEGA is a four-stage launcher made up of three solid rocket motors and a small liquid rocket engine that performs the final orbit insertion maneuver [42]. The rocket configuration is such that the third-stage burnout velocity is close to the orbital one, and hence it ends up falling far away from the launch site. In this respect, the impact point of the third stage must be predicted and actively constrained to a safe location. The return of the other stages also requires a careful design, but it would draw in further safety-related requirements, specific of either the launch vehicle or the launch base, that are out of the scope of this paper, and it is thus neglected. The accurate prediction of the splash-down location requires the inclusion of an additional return phase in the formulation that must be cast as a free-time phase to efficiently satisfy the zero-altitude terminal constraint. Moreover, because it features a high-velocity object falling into the atmosphere, the accurate discretization of its dynamics may significantly increase the overall computational burden of the optimization. Thus, a proper discretization scheme must be adopted to limit the problem dimension. In this respect, a hp discretization, based on Radau pseudospectral method [43], is employed to obtain accurate solutions with a limited computational cost [44]. Finally, a constraint related to the maximum heat flux that the payload can undergo once the fairing is jettisoned is included in the formulation. This is another nonconvex constraint to be tackled in the convexification procedure and it presents the further technical difficulty of being coupled with the splash-down constraint because moving the impact point affects the whole trajectory profile and, in particular, the encountered heat flux conditions.

This paper features several original contributions. Compared with the authors' previous works [40,41], a different formulation that makes use of a Cartesian coordinate system is employed to overcome numerical issues arising for near-polar orbits when spherical coordinates are used. Also, the typical lossless convexification strategy originally proposed for powered descent problems [45] is slightly modified by preserving the original state variables and thus preventing accuracy issues related to the large variation between the liftoff and payload mass when a logarithmic scale is used [41]. Finally, additional algorithmic robustness is provided by an original method, named *filtering*, for the recursive update of the reference solution that avoids the use of trust region constraints [46–48], which may limit the search space or modify the objective function and thus require a careful implementation, as well as initialization strategies based on homotopy [7–9], which can significantly increase the computational burden. When using filtering, the update of the reference solution is devised as a weighted sum of the previously found solutions, rather than being based only on the last one. This approach successfully filters out oscillations in the search space and other common undesired phenomena due to the successive linearization, such as artificial unboundedness, without altering the OCP, as no additional constraint or penalty term needs to be added to the formulation. The combination of the novel convexification strategy and the high-accuracy hp pseudospectral discretization sets up a highly efficient and robust optimization process that allows for a successful solution of the ascent problem. Both these aspects of novelty of the technique are discussed in the paper, with reference to the case study. In particular, the filtering technique is compared with hard and soft trust region techniques in terms of robustness and efficiency of the devised optimization technique.

This paper is organized as follows. Section II outlines the multistage launch vehicle ascent problem, describing the phase structure, the system dynamics, and the mission requirements. In Sec. III, the original OCP is transcribed into a convex optimization problem via a combination of lossless and successive convexification methods. The initialization strategy and the recursive update of the reference solution via filtering are detailed in Sec. IV. Numerical results are presented in Sec. V to show the effectiveness of the proposed method. The computational efficiency and robustness to the initial guess are studied via an extensive Monte Carlo analysis, highlighting also the merits of the novel filtering technique. Finally, an analysis of the sensitivity of the achievable payload with respect to different splash-down locations is carried out, showing the interaction between the heat flux and splash-down constraints.

II. Original Problem Formulation

In this section, the ascent trajectory is first divided into multiple phases to account for different guidance programs, coasting phases, and mass discontinuities. Second, the equations of the 3-DoF motion of the launch vehicle are derived. Finally, the constraints and objective function of the addressed OCP are outlined.

A. Flight Strategy and Phase Sequence

A launch vehicle is a system that, from liftoff to payload release, flies through variable conditions and thus requires different guidance programs to meet all mission requirements. Moreover, the ascent of a multistage rocket consists of a sequence of propelled and coasting arcs, and features the separation of inert masses at each stage burnout. To effectively tackle these specificities in the optimization process, the corresponding OCP must be cast as a multiphase problem.

During the first few seconds after liftoff, the rocket has to retain a vertical attitude in order to fly above the launch tower height and safely clear the site. Then, a programmed rotation maneuver, referred to as *pitch-over*, starts steering the vehicle axis off from its vertical attitude and eventually aligns it with the relative-to-atmosphere velocity. In the remainder of the atmospheric flight, the rocket is prescribed to keep heading in the direction of the relative wind to minimize the transverse aerodynamic load. This is called a *zero-lift gravity turn* (ZLGT) maneuver, because it exploits gravity to steer the vehicle while retaining a null angle of attack. Finally, once the rocket reaches the sufficiently rarefied layers of the atmosphere, an optimal guidance program can be followed. This usually corresponds to a Hohmann-like maneuver, meaning that the upper stage performs two burns separated by a long coasting arc.

The considered phase sequence for a VEGA-like launch vehicle is illustrated schematically in Fig. 1 and represents the typical flight strategy of a four-stage configuration. Note that the phases are numbered progressively from 1 to 13 in chronological order, with the relevant exception of the return phase, which, despite being the 13th arc, chronologically starts at the burnout of the third stage, i.e., at the end of phase 8, and takes place concurrently with phases 9–12. Hereinafter, let $t_0^{(i)}$ and $t_f^{(i)}$ denote the initial and final time of the ith phase. For the sake of simplicity, if no phase superscript is specified, then t_0 and t_f denote the liftoff time $t_0^{(1)}$ and the fourth-stage burnout $t_f^{(12)}$, respectively. Likewise, let t_R denote the return time of the spent stage $t_f^{(13)}$.

The first stage ascent is divided into three phases to properly account for the different guidance programs: vertical ascent (1), pitch-over (2), and gravity turn (3). The gravity turn maneuver continues for the entire second stage burn, so phase 5 lasts for its whole operation. The third stage operates at sufficiently high altitudes and can adopt an optimal guidance program, as aerodynamic loads do not represent a concern anymore. Because also the thermal environment is less critical, during the third-stage flight, the payload fairing is jettisoned. To efficiently handle the related mass discontinuity, the third-stage operation is split into phases 7 and 8 in correspondence of the jettisoning. VEGA's last stage performs a Hohmann-like maneuver and its flight is conveniently split into two burn phases, 10 and 12, separated by a coasting one, phase 11. Note that three other brief coasting arcs (4, 6, and 9) are included at each stage separation. Finally, the return of the third stage is included as phase 13 of the OCP.

For the sake of simplicity, we assume a fixed time schedule of phases 1–9. Therefore, only the time-lengths of phases 10–13 are free to be optimized. Indeed, we assume that the vertical ascent lasts only the few seconds necessary to reach the given clearance altitude over the launchpad and prescribe the pitch-over duration to a value that guarantees that the angle of attack of the launcher is (almost) null at the beginning of the gravity turn. Moreover, the fairing is assumed to be released after an assigned (small) amount of time to guarantee the vehicle attitude controllability and the stage full operative conditions at the jettisoning. The duration of the coasting phases at stage separation are prescribed, with the relevant exception of the Hohmann-like coasting of phase 11. Finally, the time-lengths of the other time-fixed phases are constrained by the (assigned) burn times of each stage. It is worth mentioning that this algorithm could be easily extended to handle coasting arcs of unknown duration, without the need to introduce additional trust regions. However, preliminary results reported only minor differences in the payload mass; thus their value is held fixed in this paper.

B. System Dynamics

The vehicle is modeled as a point mass subject to a 3-DoF translational motion. Under these assumptions, the state vector $\boldsymbol{x}$ is composed of the position vector $\boldsymbol{r}$, the velocity vector $\boldsymbol{v}$, and the launch vehicle mass m: $\boldsymbol{x} = [x, y, z, v_x, v_y, v_z, m]$. Note that the rocket position and velocity are expressed in Cartesian Earth-centered inertial (ECI) coordinates. In particular, the x axis is in the Earth equatorial plane and passes through the meridian of the launch site at the initial time, the z axis is aligned with Earth angular velocity, and the y axis completes the right-hand frame. This set of state variables was preferred over the spherical coordinates used in previous works [23,40,41] because it allows for studying missions toward high inclination orbits without suffering from the numerical issues related to the singularities at the poles. As a downside, when using Cartesian coordinates, the terminal conditions result in nonlinear expressions of the state variables.

The launch vehicle is supposed to be subject only to the gravity acceleration $\boldsymbol{g}$, the aerodynamic drag $\boldsymbol{D}$, and the engine thrust $\boldsymbol{T}$. A Keplerian gravitational model is assumed and the drag force is $\boldsymbol{D} = -1/2 C_D S \rho v_{\text{rel}} \boldsymbol{v}_{\text{rel}}$, where C_D is the drag coefficient, assumed to be constant, S is the reference surface, ρ is the atmospheric density, and $\boldsymbol{v}_{\text{rel}} = \boldsymbol{v} - \boldsymbol{\omega}_E \times \boldsymbol{r}$ is the relative-to-atmosphere velocity, with $\boldsymbol{\omega}_E$ denoting the Earth's angular velocity vector.

As for the propulsive system, each stage is characterized by a vacuum thrust law $T_{\text{vac}}(t)$ and an ejected mass flow rate $\dot{m}_e(t)$. However, note that the actual thrust magnitude acting on the system depends also on the external pressure p as $T = T_{\text{vac}}(t) - pA_e$, where A_e is the nozzle exit area. While the thrust magnitude is prescribed by

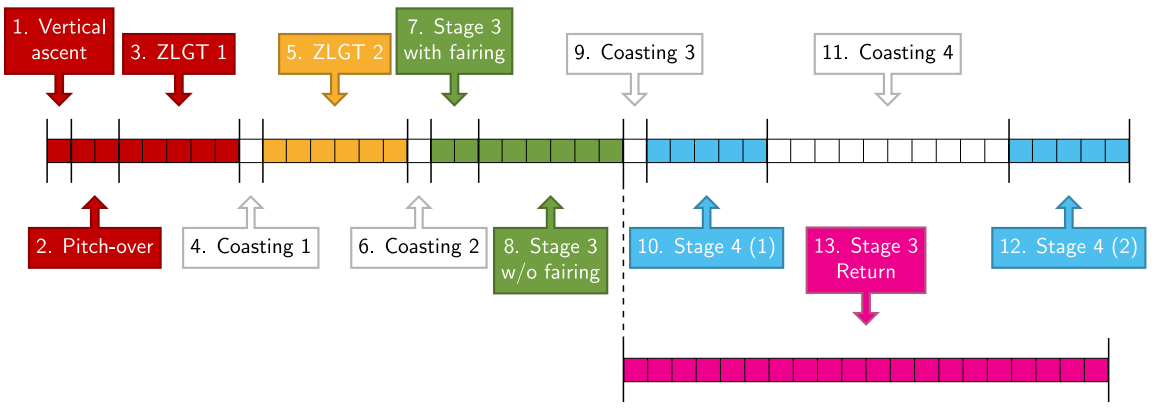


Fig. 1 Phases of the optimal control problem.

the motor characteristics and the atmospheric conditions, the thrust direction vector $\hat{\boldsymbol{T}}$ must be optimized and represents the control $\boldsymbol{u}$. Its elements are expressed in the ECI frame and, since $\hat{\boldsymbol{T}}$ is a unit vector, the following relationship must be satisfied:

$$\hat{T}_x^2 + \hat{T}_y^2 + \hat{T}_z^2 = 1 \tag{1}$$

The resulting equations of motion $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)$ are

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{2}$$

$$\dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} + \frac{T_a}{m}\hat{\boldsymbol{T}} + \frac{T_b - D}{m}\hat{\boldsymbol{v}}_{\text{rel}} + \frac{T_c}{m}\hat{\boldsymbol{r}} \tag{3}$$

$$\dot{m} = -\dot{m}_e \tag{4}$$

where the thrust magnitude T is fictitiously split into three contributions to account for the different guidance programs. Indeed, the thrust direction $\hat{\boldsymbol{T}}$ can be optimized only in some propelled arcs. For instance, during the gravity turn maneuver (phases 3 and 5) the rocket axis (thus, the thrust direction) must be aligned with the relative-to-atmosphere velocity. Likewise, during the liftoff (phase 1) the thrust must be aligned with the local vertical. Thus, T_a represents the optimally controlled thrust contribution, whereas T_b and T_c are multiplied by the relative velocity unit vector and the radial direction vector in Eq. (3). Note that only one of these three terms can be nonzero at a given time:

$$T_a = \begin{cases} T_{\text{vac}} - pA_e & \text{in phases 2, 7, 8, 10, and 12} \\ 0 & \text{in the other phases} \end{cases} \tag{5}$$

$$T_b = \begin{cases} T_{\text{vac}} - pA_e & \text{in phases 3 and 5} \\ 0 & \text{in the other phases} \end{cases} \tag{6}$$

$$T_c = \begin{cases} T_{\text{vac}} - pA_e & \text{in phase 1} \\ 0 & \text{in the other phases} \end{cases} \tag{7}$$

C. Optimal Control Problem

The goal of the optimization is to determine the control law and other mission parameters, such as the duration of free-time arcs, that maximize the payload mass injected into the target orbit. In the present work, the propellant and inert masses of the four stages, denoted by $m_{p,i}$ and $m_{\text{dry},i}$ for $i = 1, \ldots, 4$, are assumed to be assigned. Therefore, we can equivalently decide to maximize the final mass, because it differs from the payload mass by a constant value. Let the OCP be cast as a minimum problem, then the cost function J to minimize is

$$J = -m(t_f) \tag{8}$$

Besides the payload maximization, the optimization must take into account all mission requirements, which are transcribed as differential, boundary, and path constraints. The differential constraints are associated with the equations of motion (2–4). The boundary conditions include the initial, terminal, and linkage constraints. While the initial launcher mass is free to be optimized, the initial position and velocity are completely assigned. In particular, the launcher initial position corresponds to the launch base location at liftoff $\boldsymbol{r}_{\text{LB}}$, and its velocity is equal to the eastward inertial velocity due to Earth rotation:

$$\boldsymbol{r}(t_0) = \boldsymbol{r}_{\text{LB}} \tag{9}$$

$$\boldsymbol{v}(t_0) = \boldsymbol{\omega}_E \times \boldsymbol{r}_{\text{LB}} \tag{10}$$

The terminal conditions ensure that the payload is released in a circular orbit of prescribed radius r_{des} and inclination i_{des} at t_f:

$$x(t_f)^2 + y(t_f)^2 + z(t_f)^2 = r_{\text{des}}^2 \tag{11}$$

$$v_x(t_f)^2 + v_y(t_f)^2 + v_z(t_f)^2 = \mu / r_{\text{des}} \tag{12}$$

$$\boldsymbol{r}(t_f) \cdot \boldsymbol{v}(t_f) = 0 \tag{13}$$

$$x(t_f)v_y(t_f) - y(t_f)v_x(t_f) = h_{z,\text{des}} \tag{14}$$

Equations (11) and (12) constrain the semimajor axis of the final orbit to be r_{des}. Equation (13) guarantees that the radial velocity is zero at payload release; thus, combined with the previous conditions on position and velocity magnitude, it ensures that the final orbit is circular. Finally, Eq. (14) derives from the expression of the inclination in ECI coordinates:

$$i = \cos^{-1}\left(\frac{xv_y - yv_x}{h}\right) \tag{15}$$

Indeed, because the angular momentum h of the target orbit is known and equal to $\sqrt{\mu r_{\text{des}}}$, Eq. (15) can be conveniently expressed as in Eq. (14), with $h_{z,\text{des}} = \cos i_{\text{des}} \sqrt{\mu r_{\text{des}}}$.

Terminal conditions are prescribed also for the return of the burned-out third stage:

$$x(t_R)^2 + y(t_R)^2 + z(t_R)^2 = R_E^2 \tag{16}$$

$$z(t_R) = R_E \sin \varphi_{R,\text{des}} \tag{17}$$

where R_E denotes the Earth radius. Equation (16) constrains the final altitude of the returned stage to be null and Eq. (17) constrains the splash-down location to a given latitude $\varphi_{R,\text{des}}$. Note that for missions toward polar or quasi-polar orbits (e.g., sun-synchronous orbits), constraining the latitude is equivalent to constraining the splash-down distance from the launch base, because the orbital plane of the trajectory is selected during the pitch-over maneuver and remains (almost) constant in the remainder of the ascent. This turns out to be a simple, yet effective, way to impose the splash-down constraint, as it consists in assigning just the final value of the z variable, and it well suits the VEGA target orbits, which are typically high-inclination orbits. Extension to the case of constraining (also) the longitude is straightforward.

Because the problem consists of multiple phases, proper linkage conditions must be enforced at each internal boundary. All state variables are continuous at boundaries, with the relevant exception of mass, which features a discontinuity at each stage separation:

$$m\left(t_0^{(4)}\right) = m\left(t_f^{(3)}\right) - m_{\text{dry},1} \tag{18}$$

$$m\left(t_0^{(6)}\right) = m\left(t_f^{(5)}\right) - m_{\text{dry},2} \tag{19}$$

$$m\left(t_0^{(9)}\right) = m\left(t_f^{(8)}\right) - m_{\text{dry},3} \tag{20}$$

Likewise, at the fairing jettisoning, a mass discontinuity must be accounted for

$$m\left(t_0^{(8)}\right) = m\left(t_f^{(7)}\right) - m_{\text{fairing}} \tag{21}$$

Finally, the return phase initial boundary corresponds to the third-stage burnout, so the following linkage conditions must be enforced:

$$\boldsymbol{r}\left(t_0^{(13)}\right) = \boldsymbol{r}\left(t_f^{(8)}\right) \tag{22}$$

$$\boldsymbol{v}\left(t_0^{(13)}\right) = \boldsymbol{v}\left(t_f^{(8)}\right) \tag{23}$$

$$m\left(t_0^{(13)}\right) = m_{\text{dry},3} \tag{24}$$

Note that Eq. (24) is not properly a linkage condition, because the third-stage dry mass is known a priori.

As mentioned above, the final stage burn is partitioned between two phases (10 and 12). Because we assumed that all the propellant must be consumed, the sum of the time-lengths of the two firings must be equal to the overall stage burn time $t_{b,4}$:

$$\Delta t^{(10)} + \Delta t^{(12)} = t_{b,4} \tag{25}$$

where $\Delta t^{(i)} = t_f^{(i)} - t_0^{(i)}$.

Because of the high relative velocity during the atmospheric flight, the rocket undergoes severe thermal conditions. So, the payload must be protected by the fairing during the initial phases of the ascent. Nevertheless, once the atmospheric density has decreased enough, the fairing is jettisoned in order to reduce the inert mass as soon as possible. As a consequence, the payload is directly exposed to the heat flux, which must not exceed a given value. In line with VEGA's user manual, the heat flux is evaluated according to a simple model involving a free molecular flow acting on a plane surface perpendicular to the relative velocity [42]. Thus, the following path constraint is included in the formulation for phases 8–12:

$$\dot{Q} = \frac{1}{2}\rho v_{\text{rel}}^3 \leq \dot{Q}_{\max} \tag{26}$$

III. Convex Transcription

In this section, the OCP is formulated as a second-order cone programming (SOCP) problem. SOCP is a special class of convex programming that is characterized by a linear objective, linear equality constraints, and second-order cone constraints. SOCP allows for representing quite complex constraints and can be solved with a small computational effort by means of highly efficient interior point methods [49]. Because the original problem is not convex, it is converted into an SOCP problem via several convexification methods. First, a convenient change of variables, which produces control-affine dynamics, is proposed. Second, a control constraint is relaxed into a second-order cone constraint. The remaining nonconvexities are then tackled via successive linearization. Virtual controls and buffer zones are introduced to prevent possible artificial infeasibility due to the linearization. Finally, the continuous-time problem is discretized via a hp pseudospectral method.

A. Change of Variables

The equations of motion (2–4) are highly nonlinear in both state and control variables, and thus represent a source of nonconvexity. A successive linearization of these equations would produce linear constraints, but, due to the coupling of states and controls, high-frequency jitters would show up in the solution process, hindering its convergence [50]. To prevent this undesired behavior, a change of variables is exploited to obtain a control-affine dynamic system, following the same approach originally proposed by Acikmese and Ploen in Ref. [45]. The new control is introduced

$$\boldsymbol{u} = \frac{T_a}{m}\hat{\boldsymbol{T}} \tag{27}$$

Note that $\boldsymbol{u}$ includes both the thrust-to-mass ratio T_a/m and the thrust direction $\hat{\boldsymbol{T}}$. Replacing the new control in Eqs. (2–4) directly produces control-affine equations:

$$\dot{\boldsymbol{r}} = \boldsymbol{v} \tag{28}$$

$$\dot{\boldsymbol{v}} = -\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{u} + \frac{T_b - D}{m}\hat{\boldsymbol{v}}_{\text{rel}} + \frac{T_c}{m}\hat{\boldsymbol{r}} \tag{29}$$

$$\dot{m} = -\dot{m}_e \tag{30}$$

So, state and control variables are decoupled and the dynamics can be expressed as $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) = \tilde{\boldsymbol{f}}(\boldsymbol{x}, t) + \tilde{B}\boldsymbol{u}$, where $\tilde{B} = [\boldsymbol{0}_{3\times3}\boldsymbol{I}_{3\times3}\boldsymbol{0}_{1\times3}]^T$. $\boldsymbol{0}_{m\times n}$ and $\boldsymbol{I}_{m\times n}$ denote the null and identity matrix of size $m \times n$.

The new control variables must satisfy Eq. (1), which is reformulated as

$$u_x^2 + u_y^2 + u_z^2 = u_N^2 \tag{31}$$

where the additional variable u_N was introduced:

$$u_N = \frac{T_a}{m} \tag{32}$$

B. Constraint Relaxation

The path constraint (31) is a nonlinear equality constraint that requires to be convexified in order to be included in the SOCP formulation. Let us consider its relaxation attained by substituting the equality sign with the inequality sign:

$$u_x^2 + u_y^2 + u_z^2 \leq u_N^2 \tag{33}$$

Equation (33) is a convex constraint, in particular a second-order cone constraint. The inequality sign allows the control variables to be located inside a sphere of radius u_N, rather than being constrained on its surface. Therefore, the convex relaxation defines a larger feasible set than the original one. Nevertheless, the following proposition ensures that, under mild assumptions, the resulting OCP shares the same solution as the original problem. Note that the return phase can be temporarily removed from the OCP, as, being an uncontrolled phase, it is not affected by the control constraint relaxation.

Assumption 1: Constraint (26) is assumed to be inactive a.e.** in $[t_0, t_f]$.

Remark 1: Assumption 1 states that the heat flux constraint is not active over finite intervals of the solution. This assumption holds almost always for the ascent problem, because typically the heat flux constraint is active only at isolated points in time, e.g., at the fairing jettisoning.

Proposition 1: Let $\mathcal{P}_A$ be the launch vehicle ascent OCP:

$$\mathcal{P}_A: \min_{\boldsymbol{x},\boldsymbol{u},t_f} (8)$$
$$\text{s.t. } (9)-(14),(18)-(21),(25),(26),(28)-(30),(31),(32) \tag{34}$$

Let $\mathcal{P}_R$ be the relaxed version of $\mathcal{P}_A$ obtained by substituting Eq. (31) with Eq. (33), that is,

$$\mathcal{P}_R: \min_{\boldsymbol{x},\boldsymbol{u},t_f} (8)$$
$$\text{s.t. } (9)-(14),(18)-(21),(25),(26),(28)-(30),(32),(33) \tag{35}$$

The solution of the relaxed problem $\mathcal{P}_R$ is the same as the solution of $\mathcal{P}_A$. That is, if $\{\boldsymbol{x}^\star; \boldsymbol{u}^\star; t_f^\star\}$ is a solution of $\mathcal{P}_R$, then it is also a solution of $\mathcal{P}_A$ and $u_x^\star(t)^2 + u_y^\star(t)^2 + u_z^\star(t)^2 = u_N^\star(t)^2$ a.e. in $[t_0, t_f^\star]$.

Proof 1: See Appendix. □

The proof of Proposition 1 is provided in the Appendix and follows the same reasoning as in Refs. [41,51]. The intuition that motivates the relaxation is the same as in the seminal work by Acikmese and Ploen [45], later extended to a broader class of problems [19]. When Eq. (33) is strictly satisfied the engine does not provide the maximum attainable acceleration to the rocket and, because the goal of the optimization is to maximize the mass injected into a target orbit, it is apparent that such a behavior is suboptimal, and thus will be automatically discarded by the solution procedure. Finally, note that this relaxation improves the convergence properties of the successive convexification algorithm compared with a linearization of the

**A condition satisfied almost everywhere (a.e.) means that it can be violated only at a finite number of points (a set of measure zero).

constraint (31), because it introduces no approximation and fully preserves the nonlinearity of the original problem. The benefits of this approach have also been recently investigated and compared with direct linearization by Yang and Liu [52].

C. Successive Linearization

Successive linearization is employed to tackle the remaining nonconvexities, which cannot be tackled via lossless convexification. In particular, the nonconvex constraints are replaced with the first-order Taylor series expansion around a reference solution that is recursively updated.

1. *Equations of Motion*

The equations of motion (28–30) are control-affine but still nonlinear in the state variables, and thus they must be linearized. To account for free-time phases in the optimization procedure, we replace time t with τ, a new independent variable defined, for each phase, over a fixed domain [0, 1], as commonly done in traditional direct methods [53]. Thanks to the unitary domain, the time dilation σ between t and τ is equal to the actual arc time-length:

$$\sigma = \frac{dt}{d\tau} = t_f - t_0 \tag{36}$$

σ is included as an additional optimization parameter for each phase. The equations of motion are then expressed in terms of τ and approximated via the first-order Taylor series expansion around a reference solution $\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\sigma}\}$:

$$\boldsymbol{x}' \coloneqq \frac{d\boldsymbol{x}}{d\tau} = \sigma \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \tau) \approx A\boldsymbol{x} + B\boldsymbol{u} + \Sigma\sigma + \boldsymbol{c} \tag{37}$$

where the following matrices were introduced:

$$A = \bar{\sigma}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \tau) \tag{38}$$

$$B = \bar{\sigma}\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \tau) \tag{39}$$

$$\Sigma = \boldsymbol{f}(\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \tau) \tag{40}$$

$$\boldsymbol{c} = -(A\bar{\boldsymbol{x}} + B\bar{\boldsymbol{u}}) \tag{41}$$

Thanks to the change of variables previously carried out, $\boldsymbol{f}$ is linear in the control variables; thus the A and B matrices do not depend on the reference solution control $\bar{\boldsymbol{u}}$, and $B = \bar{\sigma}\tilde{B}$. This provides enhanced robustness to the successive linearization sequence as intermediate controls can change significantly among the first iterations [23]. However, the linearized dynamic equations are still function of the reference controls, but note that, when $\sigma = \bar{\sigma}$, Eq. (37) reduces to

$$\boldsymbol{x}' = A\boldsymbol{x} + B\boldsymbol{u} + \tilde{\boldsymbol{c}} \tag{42}$$

where

$$\tilde{\boldsymbol{c}} = \bar{\sigma}\tilde{\boldsymbol{f}}(\bar{\boldsymbol{x}}, \tau) - A\bar{\boldsymbol{x}} \tag{43}$$

For arcs of known duration, Eq. (42) automatically replaces Eq. (37), but the other arcs may suffer from instability issues when σ diverges excessively from the reference value, and some expedient may be necessary to ensure convergence. In the present application, the return phase does not exhibit any unstable behavior related to σ, but the other free-time phases need further safeguarding constraints on their duration. In particular, a trust region constraint is imposed on the duration of phases 11 and 12:[††]

$$|\sigma^{(i)} - \bar{\sigma}^{(i)}| \le \delta^{(i)} \qquad i = 11, 12 \tag{44}$$

The trust radii $\delta^{(i)}$ are additional optimization variables that are constrained in the interval $[0, \delta^{(i)}_{\max}]$. In the authors' experience, a suitable choice of the upper bound is usually somewhere between 1 and 10% of $\bar{\sigma}^{(i)}$. Moreover, to further incentivize $\sigma \approx \bar{\sigma}$, the trust radii are included in the cost function as (slightly) penalized terms by introducing the penalty terms:

$$J^{(i)}_{\delta} = \lambda^{(i)}_{\delta}\delta^{(i)} \qquad i = 11, 12 \tag{45}$$

where λ_{δ} are the penalty weights, which should be as small as possible in order not to shadow the originally intended objective and let the optimization autonomously determine the optimal arc time-lengths.

Finally, because the linearization can cause artificial infeasibility [21], a virtual control $\boldsymbol{q}$ is included in the dynamics to prevent this undesired phenomenon:

$$\boldsymbol{x}' = A\boldsymbol{x} + B\boldsymbol{u} + \Sigma\sigma + \boldsymbol{c} + \boldsymbol{q} \tag{46}$$

The virtual control vector is an unbounded variable that enables to reach any point in the state space in finite time, thus solving the infeasibility issue. To ensure that its use is limited to otherwise infeasible instances, an additional penalty term is defined:

$$J_q = \lambda_q P(\boldsymbol{q}) \tag{47}$$

where λ_q is the (high) penalty weight and $P(\boldsymbol{q})$ a penalty function that we will define upon discretization.

2. *Boundary Constraints*

All terminal conditions at payload release (11–14) are nonlinear in the state variables and must be linearized as

$$\bar{\boldsymbol{r}}(t_f)\cdot\bar{\boldsymbol{r}}(t_f) + 2\bar{\boldsymbol{r}}(t_f)\cdot(\boldsymbol{r}(t_f) - \bar{\boldsymbol{r}}(t_f)) = r^2_{\text{des}} \tag{48}$$

$$\bar{\boldsymbol{v}}(t_f)\cdot\bar{\boldsymbol{v}}(t_f) + 2\bar{\boldsymbol{v}}(t_f)\cdot(\boldsymbol{v}(t_f) - \bar{\boldsymbol{v}}(t_f)) = \mu/r_{\text{des}} \tag{49}$$

$$\bar{\boldsymbol{r}}(t_f)\cdot\bar{\boldsymbol{v}}(t_f) + \bar{\boldsymbol{v}}(t_f)\cdot(\boldsymbol{r}(t_f) - \bar{\boldsymbol{r}}(t_f)) + \bar{\boldsymbol{r}}(t_f)\cdot(\boldsymbol{v}(t_f) - \bar{\boldsymbol{v}}(t_f)) = 0 \tag{50}$$

$$\bar{v}_y(t_f)(x(t_f) - \bar{x}(t_f)) - \bar{v}_x(t_f)(y(t_f) - \bar{y}(t_f)) - \bar{y}(t_f)v_x(t_f) + \bar{x}(t_f)v_y(t_f) = h_{z,\text{des}} \tag{51}$$

Likewise, also the condition on the return final radius (16) is linearized as

$$\bar{\boldsymbol{r}}(t_R)\cdot\bar{\boldsymbol{r}}(t_R) + 2\bar{\boldsymbol{r}}(t_R)\cdot(\boldsymbol{r}(t_R) - \bar{\boldsymbol{r}}(t_R)) = R^2_E \tag{52}$$

Because also the linearization of the terminal constraints may generate artificial infeasibility, virtual buffer zones are introduced. In particular, Eqs. (48–51) are grouped into a constraint vector $\boldsymbol{\chi} = \boldsymbol{0}$ and then relaxed as $\boldsymbol{\chi} = \boldsymbol{w}$, where $\boldsymbol{w}$ are free variables, referred to as virtual buffers. Like virtual control, the virtual buffers should be used only when necessary, so a penalty term is defined:

$$J_w = \lambda_w \|\boldsymbol{w}\|_1 \tag{53}$$

where λ_w is the (high) penalty weight.

The augmented cost function that includes the trust radii, the virtual control, and the virtual buffer zone penalties is

$$J = -m(t_f) + J^{(11)}_{\delta} + J^{(12)}_{\delta} + J_q + J_w \tag{54}$$

[††]Note that phase 10 does not require a trust region as its duration is implicitly related to the one of phase 12 via Eq. (25).

3. Path Constraints

The auxiliary control variable u_N must be equal to the thrust-to-mass ratio at every time, and thus Eq. (32) represents a nonlinear path constraint to be linearized as

$$u_N = \frac{T_{\text{vac}} - p(\bar{\boldsymbol{r}})A_e}{\bar{m}}\left(1 - \frac{m-\bar{m}}{\bar{m}}\right) - \frac{A_e}{\bar{m}}\frac{dp(\bar{\boldsymbol{r}})}{d\boldsymbol{r}} \cdot (\boldsymbol{r} - \bar{\boldsymbol{r}}) \tag{55}$$

In the same fashion, the linearized heat flux constraint (26) is

$$\dot{Q}(\bar{\boldsymbol{r}}, \bar{\boldsymbol{v}}) + \frac{\partial \dot{Q}}{\partial \boldsymbol{r}}(\bar{\boldsymbol{r}}, \bar{\boldsymbol{v}}) \cdot (\boldsymbol{r} - \bar{\boldsymbol{r}}) + \frac{\partial \dot{Q}}{\partial \boldsymbol{v}}(\bar{\boldsymbol{r}}, \bar{\boldsymbol{v}}) \cdot (\boldsymbol{v} - \bar{\boldsymbol{v}}) \leq \dot{Q}_{\max} \tag{56}$$

where the partial derivatives of the thermal flux with respect to position and velocity are

$$\frac{\partial \dot{Q}}{\partial \boldsymbol{r}} = \frac{1}{2}\frac{d\rho}{d\boldsymbol{r}} v_{\text{rel}}^3 + \frac{3}{2}\rho v_{\text{rel}} \boldsymbol{\omega}_E \times \boldsymbol{v}_{\text{rel}} \tag{57}$$

$$\frac{\partial \dot{Q}}{\partial \boldsymbol{v}} = \frac{3}{2}\rho v_{\text{rel}} \boldsymbol{v}_{\text{rel}} \tag{58}$$

D. Discretization

As a final step, the continuous-time problem must be transcribed into a finite set of variables and constraints to enable the use of numerical algorithms. In this respect, we employ a hp pseudospectral method. The hp discretization combines the advantages of h and p schemes, because it exploits the exponential convergence rate of pseudospectral methods in regions where the solution is smooth and introduces mesh nodes near potential discontinuities [44]. Furthermore, compared with p methods, the hp transcription generates sparser problem instances, i.e., with quasi-diagonal matrices, allowing for the use of more efficient numerical routines.

The discretization splits the time domain into multiple subintervals and imposes the differential constraints in each segment via local orthogonal collocation. In the present paper, we locally employ the Radau pseudospectral method (RPM) [54] because it is one of the most accurate and performing pseudospectral methods [43]. The RPM is also a particularly convenient scheme to embed in a hp discretization, as it avoids redundant control variables at the segment interfaces and provides the optimal control at each mesh point (except for the final node of the final subinterval). Indeed, the RPM is based on the Legendre–Gauss–Radau (LGR) abscissas, which include the initial boundary but not the final one. Locally, this design does not provide the terminal control in each segment, but globally, the ambiguity drops because the final node of a segment corresponds to the initial boundary of the next one, for which, instead, the control is available.

Because details on the implementation of a hp Radau pseudospectral method can be found in the literature [13–15], this paper outlines only the major steps of the discretization scheme. First, the hp method splits the independent variable domain $\tau \in [0, 1]$ of each phase into h segments by defining a grid $\mathcal{H}$ of $h+1$ nodes $0 = \tau_1 < \ldots < \tau_{h+1} = 1$. Then, each segment $[\tau_s, \tau_{s+1}]$ is discretized as a grid $\mathcal{N}_s$ of $p_s + 1$ nodes $-1 = \eta_1 < \ldots < \eta_{p_s+1} = 1$, where p_s is the discretization order of the segment and η is a new independent variable defined in the interval $[-1, 1]$, which can be mapped to the original domain by the following transformation:

$$\tau = \frac{\tau_{s+1} - \tau_s}{2}\eta + \frac{\tau_{s+1} + \tau_s}{2} \tag{59}$$

Because we employ the RPM, the first p_s nodes of each segment correspond to the set of p_s LGR roots and constitute the collocation points $\mathcal{K}_s$. Note that $\mathcal{K}_s$ is a subset of $\mathcal{N}_s$ because it does not incorporate the terminal boundary $\eta = 1$.

Once the grid is set up, the state and control are discretized over it, and a finite set of variables $(\boldsymbol{x}_j^s, \boldsymbol{u}_j^s)$ is obtained. The superscript s denotes the sth segment, and the subscript j refers to the jth node of the segment. In particular, in each segment, the state is discretized over the set $\mathcal{N}_s$ and approximated using a basis of Lagrange polynomials. Note that because the state is continuous among the segments of a phase, in the algorithm implementation the same variable is used for both $\boldsymbol{x}_{p_s+1}^s$ and $\boldsymbol{x}_1^{s+1}$. Instead, the control is discretized only at the collocation points $\mathcal{K}_s$, so Lagrange polynomials of degree $p_s - 1$ are used for the approximation. The final control of the final segment is not included in the discrete problem and it is simply extrapolated from the polynomial approximation of the control signal.

Path constraints are converted into a finite set of algebraic constraints by imposing them at every node, whereas boundary conditions are imposed only at the initial or final point of $\mathcal{H}$. To take into account the system dynamics (46), the time derivative of the state interpolating polynomial is constrained to be equal to the equations of motion at the collocation points of each segment $s = 1, \ldots, h$:

$$\sum_{j=1}^{p_s+1} D_{ij}^s \boldsymbol{x}_j^s = \frac{\tau_{s+1} - \tau_s}{2}\left(A_i^s \boldsymbol{x}_i^s + B_i^s \boldsymbol{u}_i^s + \Sigma_i^s \sigma + \boldsymbol{c}_i^s + \boldsymbol{q}_i^s\right) \qquad i = 1, \ldots, p_s \tag{60}$$

where the same notation used for discrete-time variables was used for the linearization matrices (38–41). In Eq. (60), D^s denotes the LGR differentiation matrix [43], which can be efficiently computed via barycentric Lagrange interpolation [55]. Finally, similarly to the other continuous-time variables, also the virtual control is discretized over the mesh as $\boldsymbol{q}_j^s$. The resulting set of variables is grouped into a vector $\tilde{\boldsymbol{q}}$ and the penalty term introduced in Eq. (47) can be transcribed as

$$J_q = \lambda_q \|\tilde{\boldsymbol{q}}\|_1 \tag{61}$$

As a final remark, in this paper, no automatic mesh refinement is implemented. So, a sufficiently dense grid must be devised a priori according to the desired discretization accuracy.

IV. Reference Solution

The convexification of the original problem nonlinear dynamics and constraints exploits successive linearization, which replaces the original expressions with a first-order Taylor series expansion around a reference solution $\{\bar{\boldsymbol{x}}, \bar{\boldsymbol{u}}, \bar{\sigma}\}$. This section focuses on the reference solution. First, we outline a simple procedure to design a starting trajectory that allows for convergence. Then, an improved method for updating the reference solution based on multiple previous iterations is proposed.

A. Initialization

Sensitivity to the initialization is a major downside of traditional optimization methods. For instance, indirect methods can achieve convergence only if an accurate first guess is provided. This is a cumbersome drawback as an initialization is not required only for the trajectory but also for the costate and the structure of the constrained arcs, which can be difficult to supply. On the other hand, direct methods exhibit greater robustness to the initialization, but the discretization of highly sensitive nonconvex OCPs, such as the one at hand, produces an NLP problem whose solution depends significantly on the first guess. These limitations motivate the upstream effort put into the careful convexification process. Indeed, a greater robustness is observed in the devised algorithm compared with traditional direct optimization methods. Moreover, also compared with our previous work on convex optimization of the ascent problem [40,41], the present algorithm shows an enhanced robustness. The reason for this improvement is the hp pseudospectral discretization, which accurately describes the dynamics and yet retains a sparse problem structure.

The standard way of dealing with the problem at hand is, first, solving the ascent problem without the splash-down constraint, then

simulate the return of the spent stages and, if necessary, constrain the splash-down to a safe location. In fact, the concern on the splash-down of the spent stages exists only if the simulation of the return trajectory corresponds to an unsafe impact location. So, phase 13 and the related constraints can be omitted at first and focus can be placed on designing a reference solution for phases 1–12 only.

The present algorithm does not require an accurate initialization, but, rather, in the authors' experience, any starting trajectory with an altitude profile always above sea level is sufficient to achieve convergence. Such trajectories can be easily generated via numerical integration of the original rocket equations of motion (2–4). To set up the forward propagation, the unknown control history, the duration of free-time arcs, and the initial mass must be prescribed. In general, designing the control laws may be a complex task, but if the atmosphere is removed from the dynamics and by choosing a small value of $m(t_0)$, i.e., which corresponds to a small payload mass, even trivial control laws can produce acceptable trajectories.

During the pitch-over, the elevation ϕ, i.e., the angle between the thrust direction and the local horizontal, is prescribed to vary linearly from 90° to a final value, commonly referred to as *kick* angle, here denoted as $\phi(t_f^{(2)})$. Phase 2 is also assumed to take place in an fixed inertial plane; therefore, the thrust azimuth ψ, i.e., the angle measured clockwise from the north direction to the thrust vector, is kept equal to a constant value $\psi^{(2)}$. While the kick angle must be guessed, a systematic way of choosing $\psi^{(2)}$ is selecting the value that, under the nonrotating Earth assumption, allows for reaching the target orbit plane without further out-of-plane maneuvers:

$$\psi^{(2)} = \sin^{-1}\left(\frac{\cos(i_{\text{des}})}{\cos(\varphi_{LB})}\right) \tag{62}$$

where φ_{LB} is the latitude of the launch base. Unfortunately, when $i_{\text{des}} < \varphi_{LB}$ the previous formula does not hold anymore and an *ad hoc* value must be provided for $\psi^{(2)}$. For stages 3 and 4, even simpler control laws can be devised. Indeed, the tentative solutions are designed such that the orbital plane is kept constant after the second stage burnout; so, the thrust vector is constrained in the orbital plane. The only control to prescribe is the elevation angle ϕ, which is kept null for the entire operation of both final stages. Finally, the subdivision of the fourth-stage burn and the duration of the intermediate coasting must be chosen.

To sum up, the only variables necessary for generating a tentative solution are i) the initial mass $m(t_0)$ (or, equivalently, the payload mass), ii) the kick angle $\phi(t_f^{(2)})$, and iii) the time-lengths of phases 10–12. These values should be set on the basis of the specific launch vehicle and target orbit. Nevertheless, their choice does not represent an arduous task, because a wide range of values can generate acceptable trajectories.

B. Recursive Update

At every iteration of the successive convexification algorithm, the reference solution must be updated. Traditional successive linearization algorithms solve the ith SOCP problem by linearizing the constraints around the $(i-1)$th solution. Instead, we employ an improved method, named *filtering* [23], which consists in computing the reference solution for the ith SOCP problem as a weighted sum of the K previous solutions:

$$\bar{x}^{(i)} = \sum_{k=1}^{K} \alpha_k x^{\max\{0,(i-k)\}} \tag{63}$$

where α_k are constant weights and $x^{(i)}$ denotes the solution to the ith subproblem. Note that if $i < K$ then the initial reference solution $x^{(0)}$ appears multiple times in the sum.

The proposed technique adds another layer of algorithmic robustness to the successive convexification procedure. In fact, it has been observed that sequences solved with $K = 1$ suffered from instability issues, mainly related to artificial unboundedness. The common approach to unboundedness is adding a trust region constraint that limits the search space to the neighborhood of the reference solution [21]. However, if the reference solution is far from the optimal one, constraining the search space may cause convergence toward a suboptimal solution. Instead, filtering efficiently solves the unboundedness issue and it does not affect the optimality of the attained solution, as no additional constraint or penalty term is included in the SOCP formulation.

The same parameters used in a different application [23] revealed to be effective also for the problem at hand. Specifically, the three last solutions are used ($K = 3$) and the corresponding weights are $\alpha_1 = 6/11, \alpha_2 = 3/11$, and $\alpha_3 = 2/11$. Different values of K and of the weights can also achieve convergence in a wide range of missions and are investigated in Sec. V.B.

It is worth noting that the filtering technique is applied to the update of all optimization variables: states, controls, and time-lengths of the free-time phases. As mentioned in Sec. III.C, some time-lengths are also subject to a trust region constraint. In fact, filtering does not exclude the use of a trust region, and, because the problem under investigation is particularly sensitive to some optimization variables, the use of a trust region on these variable was found to improve convergence.

Eventually, the sequential algorithm terminates when all the following criteria are met: i) the difference between the computed solution and the reference one converges below an assigned tolerance $\|\boldsymbol{x} - \bar{\boldsymbol{x}}\|_\infty < \epsilon_{\text{tol}}$; ii) the computed solution adheres to the nonlinear dynamics within a tolerance ϵ_f:

$$\left\| \sum_{j=1}^{p_s+1} D_{ij}^s \boldsymbol{x}_j^s - \frac{t_{s+1} - t_s}{2} \boldsymbol{f}(\boldsymbol{x}_i^s, \boldsymbol{u}_i^s, t_i^s) \right\|_\infty < \epsilon_f \tag{64}$$

in each phase, for $i = 1, \ldots, p$, and $s = 1, \ldots, h$; and iii) the virtual buffers of the computed solution are below the dynamics tolerance $\|\boldsymbol{w}\|_\infty < \epsilon_f$.

V. Numerical Results

In this section, numerical results are presented to show the effectiveness of the proposed approach. The described algorithm has been implemented in C++ using Gurobi [56] as SOCP solver. The values of the penalty weights are $\lambda_\delta^{(11)} = \lambda_\delta^{(12)} = 10^{-4}$ and $\lambda_q = \lambda_w = 10^4$. The convergence and dynamics tolerances were set equal to $\epsilon_{\text{tol}} = 10^4$ and $\epsilon_f = 10^{-6}$. Also, because scaling is key to the effectiveness of any numerical algorithm, we take the Earth radius, the corresponding circular orbit velocity, and a reference mass of 10,000 kg as normalization factors.

The data used to model the VEGA-like launch vehicle are summarized in Table 1. The main assumption concerns the thrust and mass flow rate history of the stages, which are approximated as linear functions of time. Nevertheless, the total impulse is retained and the other quantities are quite accurate, so the overall model is representative of the real system performance. Other design values include the fairing mass m_{fairing} (535.3 kg), the drag coefficient C_D (0.381), and the reference surface S (9.079 m^2). Although a realistic aerodynamic model would be needed to accurately predict the splash-down location, for this work, in a simplified manner, the same coefficients are used also for the return phase. Notwithstanding, the algorithm can consider more realistic aerodynamic characterizations of the launch vehicle and stage return. Finally, the U.S. Standard Atmosphere 1976 model is used to evaluate the air density and pressure as functions of the altitude [57].

As for the discretization, Table 2 reports h and p for every phase. The values have been devised in a heuristic way in order to meet the desired discretization accuracy. In particular, because phases 1–12 are relatively brief and do not feature rapidly changing dynamics, no internal subdivision is necessary and h is simply set to 1. Instead, because a high number of nodes are required to capture the reentry dynamics and high-order approximating polynomials suffer from numerical issues, the return phase is split into 10 equally spaced segments. In each segment, the same discretization order p is used.

Table 1 VEGA-like rocket data

Quantity	Stage 1	Stage 2	Stage 3	Stage 4	Unit
m_p	87,898	23,926	10,006	397	kg
m_{dry}	8,417	2,563	1,326	813	kg
t_b	102.0	75.0	110.0	502.1	s
$T_{\text{vac}}(0)$	2,827	1,075	299	2.45	kN
$T_{\text{vac}}(t_b)$	1,885	717	222	2.45	kN
$\dot{m}_e(0)$	1,034	383	105	0.79	kg/s
$\dot{m}_e(t_b)$	689	255	77	0.79	kg/s
A_e	3.09	1.70	1.18	0.07	m^2

The considered case study is a mission toward a 700 km circular polar Earth orbit ($i_{\text{des}} = 90°$). The vehicle is assumed to take off from the equator in correspondence of the Guiana Space Center meridian. The time-lengths of the arcs of fixed duration are reported in Table 3. The threshold on the bearable heat flux is set to 900 W/m^2. First, the optimal ascent trajectory is found, neglecting the splash-down location of the spent stages. Then, the return phase is included in the OCP and an analysis of the sensitivity of the system performance to different impact points is presented.

A. Unconstrained Return

To set up the optimization, a starting reference solution must be provided. This is generated as described in the previous section and the used parameters are reported in Table 4. Note that a very small payload mass m_{pl} was picked (less than 10% of the expected optimum) and that the duration of phase 10 is omitted as it can be automatically derived from Eq. (25).

The convergence behavior is illustrated in Fig. 2. Starting from the initial guess (dashed black line), the intermediate solutions, whose color transitions from red to green, gradually converge to the final trajectory. Thus, despite that the initial reference solution is far from the solution of the OCP, the termination criteria are eventually met in 22 iterations. Note that the virtual buffer zones introduced to relax the terminal constraints are actively exploited in the first 10 iterations. Indeed, the intermediate subproblems would otherwise be infeasible (even with virtual controls); thus virtual buffers are essential to ensure the recursive feasibility of the sequential process. Without any specific code optimization, the overall computational time is 12.8 s, so each iteration requires 0.58 s on average.[‡‡] By using a custom SOCP solver and running on dedicated hardware, a further speed-up can be expected, thus enabling potential suitability for real-time guidance. Moreover, in real-time applications a much more accurate initialization is used, as a nominal trajectory is already available, so fewer iterations are needed, greatly reducing the computational burden.

Table 2 Discretization segments, order, and nodes in each phase

Grid Param.	Phase												
	1	2	3	4	5	6	7	8	9	10	11	12	13
h	1	1	1	1	1	1	1	1	1	1	1	1	10
p	5	5	17	5	19	14	5	19	9	19	19	19	10
Nodes	6	6	18	6	20	15	6	20	10	20	20	20	101

Table 3 Time-lengths of time-fixed arcs

Phase	Δt, s
1	4.1
2	6.6
3	91.3
4	6.6
5	75.0
6	37.3
7	5.4
8	104.6
9	15.4

Table 4 Values used for the generation of the first guess trajectory

Quantity	Value	Unit
m_{pl}	100.0	kg
ϕ_k	80.0	deg
$\Delta t^{(11)}$	2500.0	s
$\Delta t^{(12)}$	200.0	s

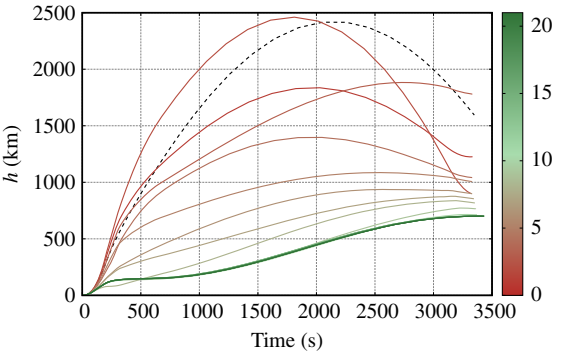


Fig. 2 Iteration sequence starting from the initial reference solution (dashed black line) and transitioning from red (`iter = 1`) to green (`iter = 22`) until convergence.

The unconstrained trajectory is illustrated in Fig. 3. The figure also reports the simulation of the return phase, which provides the optimal splash-down latitude ($\varphi_R^\star = 65.79°$). Figure 4 shows the optimal control laws of every controlled phase in terms of the elevation angle ϕ, defined as in Sec. IV.A. As for the pitch-over, one should constrain the initial elevation to 90 deg to ensure the control continuity with the liftoff phase. However, using Cartesian coordinates this would require adding a nonlinear constraint to the formulation, so, for the sake of simplicity, the initial pitch-over control direction is unconstrained in the present study. All the control laws are regular and very smooth, further proving the optimality of the attained solution.

The accuracy of the converged solution is verified by forward propagation of the original equations of motion (2–4) using the optimal control laws. In particular, the discrepancies in the terminal conditions are inspected. The largest inaccuracy concerns the semimajor axis, but the error is less than 50 m, corresponding to a relative error equal to 0.0008%, which is in agreement with the finite precision of the SOCP solver. A deeper analysis on the solution accuracy is presented in Sec. V.C.

To validate the quality of attained results, the same problem was solved also using EOS [58], a direct shooting algorithm based on differential evolution that was already successfully employed to solve a similar instance of the problem at hand [59]. The comparison between the two solutions is reported in Table 5. The payload mass difference is approximately 5 kg and is due to the different sets of time-lengths found. Indeed, the problem features many local optima with different times but similar costs, so the optimization can converge unpredictably toward one of these. Nevertheless, note that the difference in cost is minimal, so both solutions are acceptable for any practical purpose. Compared with the convex approach, the main drawback of EOS is the large computational effort required (approximately 20 minutes on the same hardware).

[‡‡]The algorithm was tested on a computer equipped with Intel Core i7-7700HQ CPU @ 2.80 GHz.

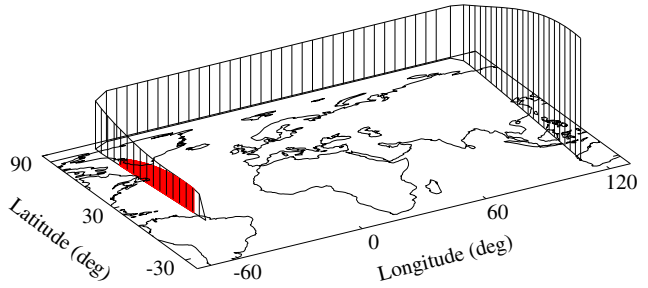


Fig. 3 Visualization of the unconstrained trajectory, with the third-stage return flight colored in red.

As a final remark, the assumption of fixed duration of the coasting arcs is not required for the success of the convex optimization method. By freeing the time-length of phases 4, 6, and 9, with the only requirement of guaranteeing a minimum stage separation time (6.6 s) for safety reasons, a slightly improved solution was found, as reported in Table 6. The initial guess was designed by using the same values as in Table 4 and the nominal coasting times from Table 3. The optimal coasting times are slightly different from the nominal values, with the relevant exception of the duration of the first coasting (phase 4) that is the same as in the fixed-time case and corresponds to the lower bound. The change in the payload mass is quite small (<1 kg). The solution found by the convex approach when freeing the duration of phases 4, 6, and 9 is in good agreement with the one found by EOS when freeing the same variables, validating the quality of the attained solution (see Table 6). Also, a further analysis showed that payload variations corresponding to changes in the third coasting time $\Delta t^{(9)}$ are almost negligible and of the same order as the accuracy of the solver. Eventually, in what follows, all coasting times, except for phase 11, are prescribed to the nominal values of Table 3 for the sake of simplicity.

B. Analysis of Filtering Technique

A Monte Carlo analysis is presented to assess the effectiveness of the filtering technique for updating the reference solution. A systematic analysis of the computational effort (measured in terms of the average run time) and the robustness to a randomly sampled initial guess trajectory is carried out considering, in addition to the *custom* set of weights ($\boldsymbol{\alpha} = [6/11, 3/11, 2/11]$) that was presented in Sec. IV.B and devised in a heuristic way, several other distributions of weights inspired by well-known sequences in mathematics. Also,

Table 5 Unconstrained solution compared with the EOS solution

Quantity	Convex	EOS	Unit	Variation, %
m_{pl}	1400.73	1396.74	kg	0.2855
$\Delta t^{(10)}$	359.71	357.60	s	0.5892
$\Delta t^{(11)}$	2583.50	2660.70	s	2.9016
$\Delta t^{(12)}$	142.39	144.50	s	1.4581

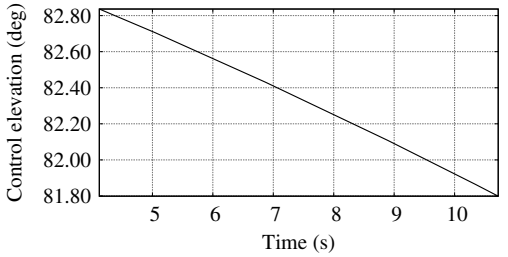
Table 6 Unconstrained solution compared with the EOS solution with free coasting arcs

Quantity	Convex	EOS	Unit	Variation, %
m_{pl}	1400.84	1402.36	kg	0.1084
$\Delta t^{(4)}$	6.6	6.6	s	0
$\Delta t^{(6)}$	38.4	40.70	s	5.6511
$\Delta t^{(9)}$	17.43	6.6	s	164.0909
$\Delta t^{(10)}$	359.90	358.10	s	0.5027
$\Delta t^{(11)}$	2582.24	2595.40	s	0.5071
$\Delta t^{(12)}$	142.20	144.00	s	0.0125

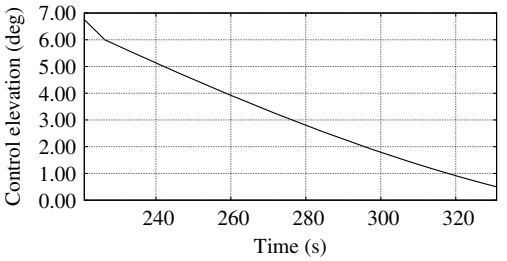
the effect of taking into account fewer or more terms in the weighted sum is investigated by varying K, i.e., the number of previously found solutions used in the recursive update.

For each devised set of weights, we considered 2000 random initial trajectories generated as described in Sec. IV.A by sampling the initialization parameters in the ranges reported in Table 7. To better highlight the effects of the filtering method on the convergence rate, values of the first guess payload mass larger than the one used in Sec. V.A were considered, as lower values (e.g., below 250 kg) were found to significantly reduce the convergence rate, thus flattening out the results of the analysis, regardless of the adopted set of weights.

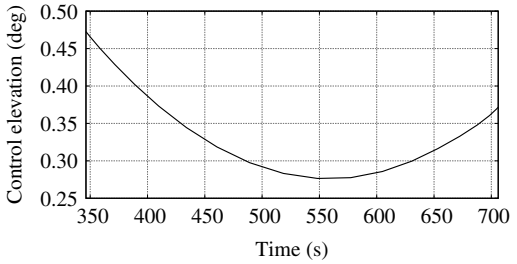
In the present analysis, three families of sets of weights were considered, in addition to the custom set previously introduced. The first family is generated by using a linear sequence of K terms taken in reverse order and normalized by their sum (e.g., $\boldsymbol{\alpha} = [3, 2, 1]/\alpha_N$, with $\alpha_N = 6$ for $K = 3$). The second family of weights uses a geometric sequence of terms: the weights are the first K powers of two, taken in reverse order and normalized by their sum (e.g., $\boldsymbol{\alpha} = [4, 2, 1]/\alpha_N$, with $\alpha_N = 7$ for $K = 3$). The last family of weights corresponds to a



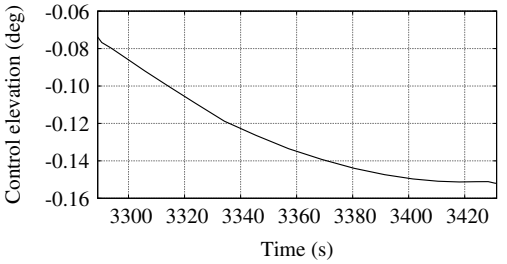
a) Pitch-Over (i.e., phase 2)



b) Stage 3 (i.e., phases 7-8)



c) First burn of stage 4 (i.e., phase 10)



d) Second burn of stage 4 (i.e., phase 12)

Fig. 4 Optimal control laws for the unconstrained scenario.

Table 7 Scattering range of the seeding parameters for the Monte Carlo analysis

Parameter	Lower bound	Upper bound	Unit
m_{pl}	250	500	kg
ϕ_k	75	80	deg
Δt_{11}	2000	2800	s
Δt_{12}	100	300	s

monotonic subset of the Fibonacci sequence, starting from 1, with terms taken in reverse order and normalized by their sum (e.g., $\boldsymbol{\alpha} = [5, 3, 2, 1]/\alpha_N$, with $\alpha_N = 11$ for $K = 4$).

Table 8 presents the results of the Monte Carlo analysis in terms of success rate and run time. A run is considered successful if it converges to a solution with a payload mass value that is greater than 95% of the optimal one reported in Table 5. The latter is deemed very close to the global optimum of the problem instance because, despite the high number of runs, none provided a better performing solution. Unsuccessful runs are those for which the algorithm either diverges (i.e., exceeds the maximum number of iterations $i_{\max} = 100$) or it converges to a suboptimal solution (i.e., a feasible solution, but deemed unacceptable because the payload mass is lower than 95% of the optimal value). Reported run time statistics are evaluated by considering successful solutions only.

The results show that the effectiveness of the algorithm strongly depends on K. Indeed, when $K = 1$ the method was never able to achieve convergence, suggesting that if filtering is not employed, then another strategy should be pursued to prevent artificial unboundedness, such as a trust region [41,46,60,61]. By setting $K = 2$, the algorithm can successfully converge to the optimal solution, but only in a limited number of cases. Instead, if $K \geq 3$, the algorithm appears to be much more reliable and it finds the optimal solution in the majority of cases. The best success rates are observed for $K = 5$, but the most efficient setups are the ones that are based on three solutions. This is because filters that use more solutions are generally more conservative, as newer solutions are assigned smaller weights and, despite showing increased ability to compensate for diverging solutions that may appear across the successive iterations, may require more iterations to converge, thus a longer computational time. On the other hand, if fewer solutions are taken into account in the update, the convergence is generally achieved in fewer iterations, as the search space is explored more rapidly, but the algorithm is more sensitive to artificial unboundedness and other diverging phenomena. As for the choice of the weight distribution, the homogeneity of the results for the each K indicates that all the considered families of sets are valuable options. Indeed, only slight differences are observed, suggesting that the geometric set is associated with the shortest mean run times, but the most robust families are the linear one, for $K < 5$, and the Fibonacci one, for $K \geq 5$.

Table 8 Results of the Monte Carlo analysis for several values of K and different sets of filtering weights

			Run time, s		
K	Distribution of weights	Success rate, %	Min	Mean	Max
1	Linear[a]	0.0	—	—	—
2	Linear[a]	16.7	4.6	30.3	76.0
3	Linear[b]	55.9	4.4	25.8	75.2
	Geometric	51.7	4.4	25.6	60.4
	Custom	54.4	4.5	27.4	82.0
5	Linear	59.9	6.0	32.2	90.1
	Geometric	58.0	5.2	26.9	78.5
	Fibonacci	61.4	5.9	28.1	74.7
7	Linear	54.4	8.6	32.9	100.2
	Geometric	58.5	5.5	25.2	74.2
	Fibonacci	60.4	6.0	27.6	67.7

[a]For $K = 1$ and 2 the linear set is identical to the geometric and Fibonacci sets.
[b]For $K = 3$ the linear set is identical to the Fibonacci set.

For the sake of comparison, the effectiveness of the use of a trust region in place of the filtering technique was investigated. In this respect, a further Monte Carlo campaign was carried out using the same 2000 randomly generated initial reference trajectories employed for evaluating the filtering performance, but considering alternatively (i) a dynamic hard trust region, such as the one used in Refs. [35,41,46], or (ii) a soft quadratic trust region, such as in Refs. [47,48]. Both trust regions were applied only on the control variables, because this was observed to be the most effective strategy for the problem under investigation. The initial trust radius of the adaptive hard trust region and the penalty weight of the soft one were selected according to a parametric study, using the most effective values. In particular, a starting trust radius of 10 m/s^2 was used to initialize the hard trust region and a nondimensional penalty weight equal to 10^{-3} was used for the soft trust region.

Table 9 shows that solving the multiphase ascent problem can be quite challenging if using a trust region and starting from an initial guess generated as described in Sec. IV.A. Indeed, the success rates are zero or close to zero if no trust region is included or if either a soft or a hard one are used. When relying on a trust region, the success rate can be greatly improved if a three-step continuation strategy is adopted [41], leading to more meaningful results. Indeed, solving intermediate problems, which either neglect the atmosphere or fix the time-lengths of the launch vehicle ascent phases, greatly improves the robustness of the algorithm to the initial guess. However, even with this improvement, both the hard and soft trust region are much less robust than the filtering approach on this problem, featuring success rates below 20%. This is because, in general, trust regions work better when the initial reference trajectory is closer to the optimal solution, while in this application an accurate guess is not available and is quite hard to design. Even though advanced trust region implementation demonstrated to be highly effective without particularly good initial guesses in diverse applications, including atmospheric entry [35,60] and rocket landing [28,48], a proper tuning of the involved parameters (such as initial radius of the hard trust region or penalty weights of the soft trust region) may require a significant effort. Instead, the filtering approach is much easier to set up, as the method sensitivity to the values of the weights is low; indeed, high success rates are achieved by every considered family as long as $K \geq 3$. In this regard, filtering is an effective and easy-to-set-up method to improve the convergence of successive convexification algorithms.

Note that the circumstances under which the Monte Carlo campaigns were carried out are particularly challenging. In fact, to assess whether the proposed algorithm is able to solve the optimal ascent trajectory problem even starting from a rough initial guess, the starting trajectories were randomly generated by considering wide ranges of the seeding parameters (i.e., kick angle, payload mass, and coasting times), as reported in Table 7. Greater success rates can be achieved if a more accurate initialization guess is used. A typical scenario is real-time guidance, where a nominal trajectory is always available and the vehicle path may only slightly deviate from it. In that case, the algorithm converges in fewer iterations and with success rates very close to 100% regardless of using either filtering or a trust region [62].

Table 9 Results of the Monte Carlo analysis using common trust region algorithms

			Run time, s		
Steps[a]	Trust region	Success rate, %	Min	Mean	Max
	None	0.0	—	—	—
1	Hard	0.2	28.9	61.7	131.8
	Soft	0.0	—	—	—
	None	0.1	13.0	13.0	13.0
3	Hard	4.3	11.5	67.9	309.2
	Soft	19.6	7.8	21.7	56.5

[a]Number of continuation steps in the solution strategy (see Ref. [41]).

C. Discretization Analysis

A further analysis was carried out to investigate the efficiency and accuracy of several hp grids. In addition to the "default" mesh introduced in Table 2, we considered other three grids and varied h (i.e., the number of segments each phase is divided into). Table 10 reports the number of discretization nodes in each phase for each mesh. Grid B is obtained from the default one by picking for each phase the number of intervals equal to the closest power of 2. This allows to set h equal to 1, 2, or 4, and still use local polynomials of the same order p in each phase segment because the number of time intervals is a multiple of h. Grid A is obtained by starting from grid B and halving the number of intervals (and thus the local polynomials order p) in each phase, but keeping the minimum number of nodes above 5. Conversely, grid C is designed by doubling the intervals (and correspondingly the order) of each phase of grid B. When increasing h, it is ensured that the number of segments in a phase is less than the number of intervals in order to use local polynomials of, at least, order $p = 2$. Finally, note that for the sake of simplicity we did not include the return phase in this analysis.

Table 11 reports the results of the Monte Carlo analysis. The same 2000 initial reference trajectories as in the previous section were considered for this campaign. Furthermore, the custom set of filtering weights was used. Results are presented in terms of success rate, run time, and errors on the final orbital elements (semimajor axis a, eccentricity e, and orbit inclination i). The errors on the orbital elements are evaluated as differences between the desired values and the elements corresponding to the propagated final state, which is computed by numerical integration of the original ODEs (2–4) from liftoff to the burnout of the last stage using the optimal control laws. The run time and orbital elements statistics are relative only to the successful runs, defined in the same way as in the previous section.

It is apparent that the computational time increases as the total number of nodes increases; so, for instance, solving the problem with grid C is more demanding than solving it over grid A. Nevertheless, for a given grid, the solution process can be more efficient if the phases are split into smaller segments, i.e., increasing h. However, increasing h on a given grid implies lower-order polynomials, which may be less accurate. Indeed, the average errors increase as h increases. This is particularly emphasized for grid A, the one with fewest nodes, which leads to great inaccuracies if $h > 1$. Indeed, as a limiting case, when grid A is used with $h = 4$, only unphysical solutions (with a payload mass 10% greater than the attainable one) are obtained; hence no run is deemed successful. Instead, the success rates of all the other grids are comparable. In conclusion, the default mesh appears as a reasonable tradeoff between accuracy and efficiency, as the mean run time is quite small and the errors on the final orbital elements are acceptable for any practical purpose, but also some of the other grids can be used if greater accuracy or a faster solution times are required.

Table 10 Discretization grids considered in the Monte Carlo analysis

	Nodes per phase												Total
Grid	1	2	3	4	5	6	7	8	9	10	11	12	nodes
Default	6	6	18	6	20	15	6	20	10	20	20	20	167
A	5	5	9	5	9	9	5	9	5	9	9	9	88
B	5	5	17	5	17	17	5	17	9	17	17	17	148
C	9	9	33	9	33	33	9	33	17	33	33	33	284

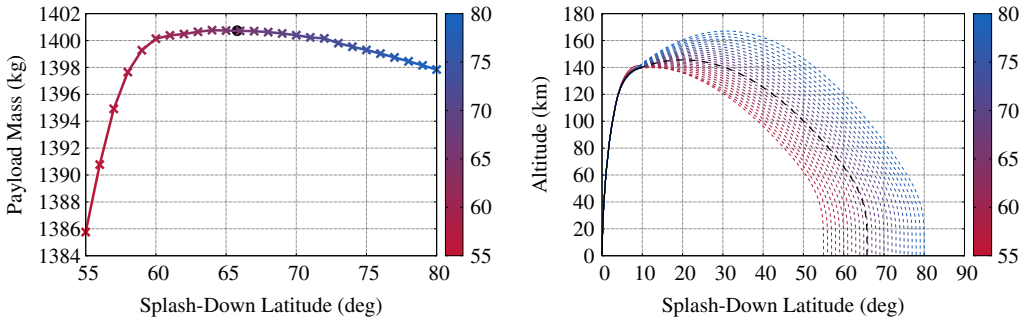
D. Parametric Analysis of the Splash-Down Constraint

Once the optimal solution is obtained, we can investigate the effect of the splash-down constraint. Therefore, the return phase is added to the OCP along with the corresponding constraints and the impact point is gradually moved from its optimal location to different latitudes. So, a series of problems with different $\varphi_{R,\mathrm{des}}$ is solved. Each OCP uses the converged solution of the previous one as initialization. Because this initialization is quite accurate, on average only eight iterations are required to meet the convergence criterion. However, the inclusion of the return phase significantly increases the problem dimension, so each iteration is computationally more demanding (0.95 s on average). Nevertheless, the overall process requires a mean computational time of 7.75 s. The "default" grid detailed in Table 2 was used to discretize the problem. Numerical integration forward in time of the equations of motion showed that the error on the splash-down point is less than 20 km. This error may appear large compared with the one on the semimajor axis, which is below 100 m, but the return dynamics is much more sensitive due to the highly nonlinear atmospheric forces that act on the spent stage until the splash-down. Indeed, if the atmospheric drag is removed from the equations of motion, the error on the splash-down location drops to 500 m, which is in agreement with the errors on the payload orbital elements.

The optimal payload mass is plotted in Fig. 5a as a function of the splash-down latitude. While moving the spent stage return point significantly changes the trajectory, as shown in Figs. 5b and 5c, it does not necessarily hinder the performance. Indeed, the payload curve is essentially flat in the interval $\varphi_R \in [60°, 72°]$ and variations only below 1 kg are observed. When the splash-down location is moved beyond 72°, the decrease in performance is more evident, but it still does not represent a concern as a shift of 15° causes a loss of only 3 kg. Instead, moving the splash-down point closer than 60° appears more critical, as a greater performance drop is observed. Nevertheless, even constraining the third stage to fall 10° closer than $\varphi_R^\star$ results in a payload reduction by only 1% of its optimal value. It is worth studying how the heat flux and splash-down constraints interact with each other. Figure 5d shows the heat flux history that the payload undergoes from the fairing jettisoning until the end of the first firing of stage 4. Red curves correspond to splash-down locations at lower latitudes, i.e., closer to the launch site, whereas blue ones are associated with high-latitude returns. In all trajectories for which $\varphi_R \geq 57°$, the heat flux constraint is active only at the boundaries of phase 8, so Assumption 1 holds in all these cases. Instead, the heat flux constraint is active over intervals of finite duration when the splash-down is moved closer than 57°. In particular, the heat flux peak is delayed

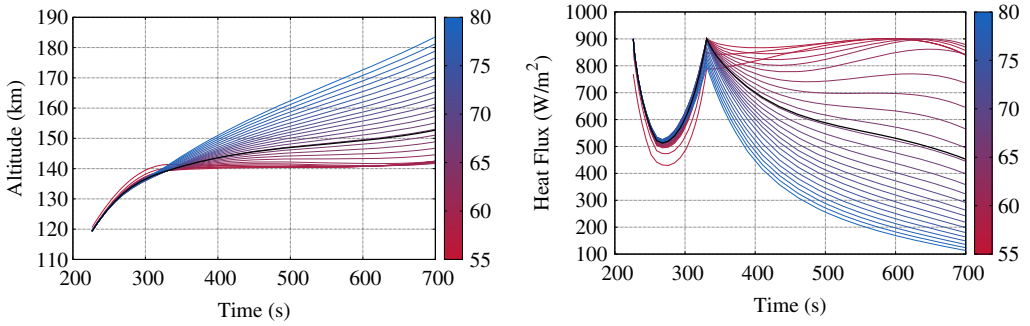
Table 11 Results of the Monte Carlo analysis on different grids

			Run time, s			Average error		
Grid	h	Success rate, %	Min	Mean	Max	a, km	e	i, deg
Default	1	55.4	3.4	16.9	71.3	6.6e–02	1.3e–05	1.5e–04
	1	55.6	1.3	7.3	35.8	7.0e–02	8.4e–05	2.5e–04
A	2	56.4	1.3	6.4	40.0	6.1e+00	1.7e–03	3.3e–04
	4	0.0	—	—	—	—	—	—
	1	55.1	2.9	18.3	55.5	4.1e–02	9.0e–06	5.1e–04
B	2	56.1	3.3	15.6	51.9	2.6e–02	2.2e–05	7.2e–04
	4	54.9	2.3	15.5	47.6	1.4e–01	7.0e–05	4.8e–03
	1	53.9	15.9	65.1	164.3	2.5e–02	4.1e–06	3.7e–04
C	2	54.7	5.0	29.2	95.7	2.0e–02	1.4e–05	1.3e–03
	4	54.9	5.0	26.8	77.8	5.6e–02	2.5e–05	3.2e–03



a) Payload mass vs. splash-down latitude

b) Stage 3 trajectories from liftoff to splash-down



c) Altitude profiles from fairing jettisoning to $t_f^{(10)}$

d) Heat flux profiles from fairing jettisoning to $t_f^{(10)}$

Fig. 5 Performance, trajectories, and thermal conditions corresponding to different splash-down latitudes compared with the unconstrained solution (black curves).

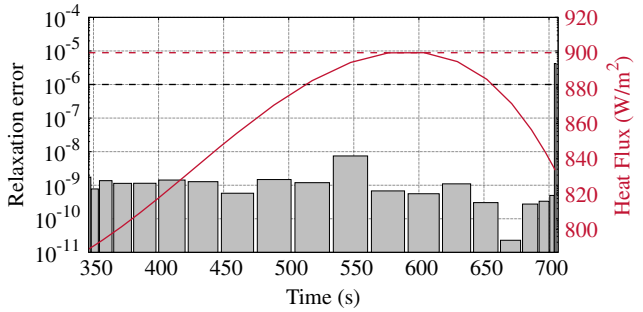


Fig. 6 Relaxation error during phase 10 of the trajectory constrained to $\varphi_R = 55°$.

and occurs during phase 10. Note that the duration and location of the bounded arc are very difficult to predict, but, thanks to the direct discretization method, the optimal switching structure is automatically determined and no a priori guess is required.

Figure 6 reports the relaxation error during phase 10 of the solution corresponding to $\varphi_R = 55°$. The error is always below the solver feasibility threshold (10^{-6}), except for the final node of the phase that, due to the Radau discretization scheme, is not an optimization variable and is extrapolated a posteriori from the approximating polynomial. This solution is particularly interesting as Assumption 1 does not hold anymore in the interval [576.6, 604.2] s. Nevertheless, even though no theoretical proof can be provided, the relaxation is still lossless, as the resulting controls satisfy Eq. (33) with the equality sign within tolerance.

VI. Conclusions

This paper presented a convex approach to the optimization of the ascent trajectory of a multistage launch vehicle. The intrinsic nonconvexities of the problem have been effectively tackled via a thoughtful convexification process. This exploits a convenient change of variables to reduce the coupling of state and control, and it preserves some of the original problem nonlinearity by relaxing, in a lossless fashion, a control constraint. These expedients, combined with successive linearization, are essential to set up a convex formulation of the original problem that does not suffer from numerical issues, such as high-frequency jitters in the control, and other undesired phenomena linked to the linearization that may hinder convergence. In this respect, virtual controls and buffer zones ensure the recursive feasibility of the iterative process, and a simple, yet effective, method to update the reference solution based on multiple previous iterations is implemented to filter out oscillations in the search space and provide further stability to the procedure. Moreover, it was shown that the employed hp discretization scheme can, on the one hand, accurately capture the complex ascent and return dynamics and, yet, produce a computationally efficient and sparse discrete problem.

The present paper investigated a VEGA-like launch vehicle configuration and analyzed its performance sensitivity to the splash-down location of the third stage. Results show that moving the return point of the spent stage can significantly change the mission profile. As a result, the payload undergoes different, hardly predictable, thermal conditions. Nevertheless, the convex optimization approach was proved to effectively handle both the heat flux and splash-down requirements in a systematic way, retaining system performance to acceptable levels in a wide range of scenarios. While the numerical results are relative to this particular case study, the general approach can be easily extended to different missions and vehicle configurations, possibly including further (nonconvex) constraints that account for additional mission requirements (e.g., visibility aspects at stage separation).

The devised algorithm exhibits great robustness to the initialization, as it can achieve convergence even starting from a rough reference trajectory, and high overall computational efficiency, as the sequential process terminates successfully and quickly, after just a few iterations. Thus, it represents a fast and reliable alternative to traditional optimization methods, which, in turn, often manifest high sensitivity to the supplied first guess solution or require a large computational effort to achieve convergence. These beneficial properties make the proposed approach potentially suitable for further studies and applications to optimization-based guidance, as speed-ups can be achieved if better initialization is provided; custom SOCP solvers are used; and the code is executed on dedicated hardware. Naturally, specific validation tests are necessary to rigorously demonstrate the real-time applicability of the algorithm, but convex optimization, combined with pseudospectral

discretization methods, holds the promise to be a disruptive tool in the design of future computational guidance algorithms.

Appendix: Proof of Proposition 1

The proof that the optimal solution of $\mathcal{P}_R$ is the optimal solution also of problem $\mathcal{P}_A$ is here provided. Proposition 1 is proved by contradiction, using the direct adjoint approach [63]. We introduce the Hamiltonian H:

$$H = \boldsymbol{\lambda} \cdot \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) = \boldsymbol{\lambda}_r \cdot \boldsymbol{v} + \boldsymbol{\lambda}_v \cdot \left(-\frac{\mu}{r^3}\boldsymbol{r} + \boldsymbol{u} + \frac{T_b - D}{m}\hat{\boldsymbol{v}}_{\text{rel}} + \frac{T_c}{m}\hat{\boldsymbol{r}} \right) + \lambda_m(-\dot{m}_e) \tag{A1}$$

where $\boldsymbol{\lambda}(t) = [\boldsymbol{\lambda}_r^T \quad \boldsymbol{\lambda}_v^T \quad \lambda_m]^T$ is the costate vector, conveniently split in position, velocity, and mass subvectors. To take into account the path constraints, the Lagrangian L is defined:

$$L = H - \mu_Q(\dot{Q} - \dot{Q}_{\max}) - \mu_u(u_x^2 + u_y^2 + u_z^2 - u_N^2) \tag{A2}$$

where $\mu_u(t)$ and $\mu_Q(t)$ are the Lagrange multipliers associated with the constraints (33) and (26), respectively. The optimal solution must satisfy the complementary slack conditions:

$$\mu_Q \geq 0, \qquad \mu_Q(\dot{Q} - \dot{Q}_{\max}) = 0 \tag{A3}$$

$$\mu_u \geq 0, \qquad \mu_u(u_x^2 + u_y^2 + u_z^2 - u_N^2) = 0 \tag{A4}$$

According to Pontryagin's minimum principle, the optimal control $\boldsymbol{u}^\star$ must be such that

$$\boldsymbol{u}^\star = \arg\min_{\boldsymbol{u} \in \mathcal{U}} H(\boldsymbol{x}^\star, \boldsymbol{u}, \boldsymbol{\lambda}, t) \tag{A5}$$

where $\mathcal{U}$ is the set of controls that satisfies Eq. (33):

$$\mathcal{U}(\boldsymbol{x}, t) = \{(u_x, u_y, u_z) : u_x^2 + u_y^2 + u_z^2 \leq ((T_{\text{vac}} - pA_e)/m)^2\} \tag{A6}$$

where u_N was replaced with Eq. (32). The Karush–Kuhn–Tucker (KKT) condition for minimizing the Hamiltonian over $\boldsymbol{u} \in \mathcal{U}$ is

$$\frac{\partial L}{\partial \boldsymbol{u}} = \boldsymbol{\lambda}_v - 2\mu_u \boldsymbol{u} = 0 \tag{A7}$$

The costate equations are

$$\dot{\boldsymbol{\lambda}} = -\frac{\partial L}{\partial \boldsymbol{x}} = -\left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right]^T \boldsymbol{\lambda} - \mu_Q \frac{\partial \dot{Q}}{\partial \boldsymbol{x}} - 2\mu_u u_N \frac{\partial u_N}{\partial \boldsymbol{x}} \tag{A8}$$

The Jacobian matrix of the dynamics has the following structure:

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \boldsymbol{0}_{3\times3} & I_{3\times3} & \boldsymbol{0}_{3\times1} \\ A_{vr} & A_{vv} & \boldsymbol{a}_{vm} \\ \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times3} & \boldsymbol{0}_{1\times1} \end{bmatrix} \tag{A9}$$

where A_{vr} and A_{vv} are full (i.e., without any identically null elements) 3×3 matrices, and $\boldsymbol{a}_{vm}$ is a full 3×1 vector. The costate equations expand to

$$\dot{\boldsymbol{\lambda}}_r = -A_{vr}^T \boldsymbol{\lambda}_v - \mu_Q \frac{\partial \dot{Q}}{\partial \boldsymbol{r}} - 2\mu_u u_N \frac{\partial u_N}{\partial \boldsymbol{r}} \tag{A10}$$

$$\dot{\boldsymbol{\lambda}}_v = -\boldsymbol{\lambda}_r - A_{vv}^T \boldsymbol{\lambda}_v - \mu_Q \frac{\partial \dot{Q}}{\partial \boldsymbol{v}} \tag{A11}$$

$$\dot{\lambda}_m = -\boldsymbol{a}_{vm} \cdot \boldsymbol{\lambda}_v - 2\mu_u u_N \frac{\partial u_N}{\partial m} \tag{A12}$$

A transversality condition useful for the present demonstration is

$$\lambda_m(t_f) = 1 \tag{A13}$$

Finally, because the terminal time t_f is free, the Lagrangian is null at the final boundary:

$$L(t_f) = 0 \tag{A14}$$

Now we show that the constraint of (33) must be active almost everywhere. In particular, first we assume that Eq. (33) is strictly satisfied; i.e., it is not active, and then argue that it is not possible because it generates a contradiction. When Eq. (33) is not active, the complementary condition (A4) requires that $\mu_u = 0$. Thus, the KKT condition (71) becomes

$$\boldsymbol{\lambda}_v(t) = \boldsymbol{0} \tag{A15}$$

For the purpose of the demonstration, we also suppose that the heat flux constraint is inactive a.e., as stated in Assumption 1. Thus, according to Eq. (A3)

$$\mu_Q = 0 \tag{A16}$$

Replacing (A15), (A16), and $\mu_u = 0$ in the costate equations (A10–A12) leads to

$$\boldsymbol{\lambda}_r(t) = \boldsymbol{0} \tag{A17}$$

$$\dot{\lambda}_m = 0 \tag{A18}$$

Because of Eqs. (A15) and (A17), the Lagrangian condition (A14) requires

$$\lambda_m(t_f) = 0 \tag{A19}$$

However, this violates Eq. (A13), thus generating a contradiction. This contradiction proves that the optimal solution of $\mathcal{P}_R$ must satisfy Eq. (33) with the equality sign. □

Acknowledgments

This work was supported by the Agreement (No. 2019-4–HH.0 CUP F86C17000080005) "Technical Assistance on Launch Vehicles and Propulsion" between the Italian Space Agency and the Department of Mechanical and Aerospace Engineering of Sapienza University of Rome.

References

[1] Jurovics, S., "Optimum Steering Program for the Entry of a Multistage Vehicle Into a Circular Orbit," *ARS Journal*, Vol. 31, No. 4, 1961, pp. 518–523. https://doi.org/10.2514/8.5545

[2] Spurlock, F., and Williams, C. H., "DUKSUP: A Computer Program for High Thrust Launch Vehicle Trajectory Design & Optimization," *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, AIAA Paper 2014-3671, 2014. https://doi.org/10.2514/6.2014-3671

[3] Colasurdo, G., Pastrone, D., and Casalino, L., "Optimization of Rocket Ascent Trajectories Using an Indirect Procedure," *Guidance, Navigation, and Control Conference*, AIAA Paper 1995-3323, 1995. https://doi.org/10.2514/6.1995-3323

[4] Martinon, P., Bonnans, F., Laurent-Varin, J., and Trelat, E., "Numerical Study of Optimal Trajectories with Singular Arcs for an Ariane 5 Launcher," *Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 1, 2009, pp. 51–55. https://doi.org/10.2514/1.37387

[5] Casalino, L., and Pastrone, D., "Optimization of Hybrid Propellant Mars Ascent Vehicle," *50th AIAA/ASME/SAE/ASEE Joint Propulsion*

Conference, AIAA Paper 2014-3953, 2014.
https://doi.org/10.2514/6.2014-3953

[6] Calise, A. J., Melamed, N., and Lee, S., "Design and Evaluation of a Three-Dimensional Optimal Ascent Guidance Algorithm," *Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 6, 1998, pp. 867–875.
https://doi.org/10.2514/2.4350

[7] Lu, P., Sun, H., and Tsai, B., "Closed-Loop Endoatmospheric Ascent Guidance," *Journal of Guidance, Control, and Dynamics*, Vol. 26, No. 2, 2003, pp. 283–294.
https://doi.org/10.2514/2.5045

[8] Bonalli, R., Hérissé, B., and Trélat, E., "Optimal Control of Endoatmospheric Launch Vehicle Systems: Geometric and Computational Issues," *IEEE Transactions on Automatic Control*, Vol. 65, No. 6, 2020, pp. 2418–2433.
https://doi.org/10.1109/TAC.2019.2929099

[9] Gath, P. F., and Calise, A. J., "Optimization of Launch Vehicle Ascent Trajectories with Path Constraints and Coast Arcs," *Journal of Guidance, Control, and Dynamics*, Vol. 24, No. 2, 2001, pp. 296–304.
https://doi.org/10.2514/2.4712

[10] Brauer, G. L., Cornick, D. E., and Stevenson, R., "Capabilities and Applications of the Program to Optimize Simulated Trajectories (POST). Program Summary Document," NASA CR-2770, 1977.

[11] Vlases, W. G., Paris, S. W., Lajoie, R. M., Martens, M. J., and Hargraves, C. R., "Optimal Trajectories by Implicit Simulation," Boeing Aerospace and Electronics TR WRDC-TR-90-3056, Wright–Patterson AFB, OH, 1990.

[12] Wiegand, A., *ASTOS User Manual*, Vol. 17, Astos Solutions GmbH, Unterkirnach, Germany, 2010.

[13] Patterson, M. A., and Rao, A. V., "GPOPS-II: A MATLAB Software for Solving Multiple-Phase Optimal Control Problems Using Hp-Adaptive Gaussian Quadrature Collocation Methods and Sparse Nonlinear Programming," *ACM Transactions on Mathematical Software*, Vol. 41, No. 1, 2014, pp. 1–37.
https://doi.org/10.1145/2558904

[14] Sagliano, M., Theil, S., D'Onofrio, V., and Bergsma, M., "SPARTAN: A Novel Pseudospectral Algorithm for Entry, Descent, and Landing Analysis," *Advances in Aerospace Guidance, Navigation and Control*, edited by B. Dołęga, R. Głębocki, D. Kordos, and M. Żugaj, Springer International Publishing, Cham, Switzerland, 2018, pp. 669–688.
https://doi.org/10.1007/978-3-319-65283-2_36

[15] Ross, I. M., and Fahroo, F., "Pseudospectral Knotting Methods for Solving Nonsmooth Optimal Control Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 27, No. 3, 2004, pp. 397–405.
https://doi.org/10.2514/1.3426

[16] Spangelo, I., and Well, K. H., "Rocket Ascent with Heat-Flux and Splash Down Constraints," *Automatic Control in Aerospace 1994, IFAC Postprint Volume*, Pergamon, Oxford, 1995, pp. 9–15.
https://doi.org/10.1016/B978-0-08-042238-1.50005-7

[17] Weigel, N., and Well, K. H., "Dual Payload Ascent Trajectory Optimization with a Splash-Down Constraint," *Journal of Guidance, Control, and Dynamics*, Vol. 23, No. 1, 2000, pp. 45–52.
https://doi.org/10.2514/2.4485

[18] Liu, X., Lu, P., and Pan, B., "Survey of Convex Optimization for Aerospace Applications," *Astrodynamics*, Vol. 1, No. 1, 2017, pp. 23–40.
https://doi.org/10.1007/s42064-017-0003-8

[19] Açkmeşe, B., and Blackmore, L., "Lossless Convexification of a Class of Optimal Control Problems with Non-Convex Control Constraints," *Automatica*, Vol. 47, No. 2, 2011, pp. 341–347.
https://doi.org/10.1016/j.automatica.2010.10.037

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

[21] Mao, Y., Szmuk, M., and Açkmeşe, B., "Successive Convexification of Non-Convex Optimal Control Problems and Its Convergence Properties," *2016 IEEE 55th Conference on Decision and Control (CDC)*, Inst. of Electrical and Electronics Engineers, New York, 2016, pp. 3636–3641.
https://doi.org/10.1109/CDC.2016.7798816

[22] Bonalli, R., Cauligi, A., Bylard, A., Lew, T., and Pavone, M., "Trajectory Optimization on Manifolds: A Theoretically-Guaranteed Embedded Sequential Convex Programming Approach," *Proceedings of Robotics: Science and Systems*, The Robotics: Science and Systems Foundation, Paper 78, 2019.
https://doi.org/10.15607/RSS.2019.XV.078

[23] Benedikter, B., Zavoli, A., and Colasurdo, G., "A Convex Optimization Approach for Finite-Thrust Time-Constrained Cooperative Rendezvous," *Advances in the Astronautical Sciences*, Vol. 171, Aug. 2019, pp. 1483–1498.

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

[25] Morgan, D., Chung, S.-J., and Hadaegh, F. Y., "Model Predictive Control of Swarms of Spacecraft Using Sequential Convex Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 37, No. 6, 2014, pp. 1725–1740.
https://doi.org/10.2514/1.G000218

[26] Wang, Z., and Grant, M. J., "Optimization of Minimum-Time Low-Thrust Transfers Using Convex Programming," *Journal of Spacecraft and Rockets*, Vol. 55, No. 3, 2018, pp. 586–598.
https://doi.org/10.2514/1.A33995

[27] Wang, Z., and Grant, M. J., "Minimum-Fuel Low-Thrust Transfers for Spacecraft: A Convex Approach," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 54, No. 5, 2018, pp. 2274–2290.
https://doi.org/10.1109/TAES.2018.2812558

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

[29] Sagliano, M., "Pseudospectral Convex Optimization for Powered Descent and Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 41, No. 2, 2018, pp. 320–334.
https://doi.org/10.2514/1.G002818

[30] Sagliano, M., "Generalized hp Pseudospectral-Convex Programming for Powered Descent and Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 7, 2019, pp. 1562–1570.
https://doi.org/10.2514/1.G003731

[31] Liu, X., "Fuel-Optimal Rocket Landing with Aerodynamic Controls," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 1, 2019, pp. 65–77.
https://doi.org/10.2514/1.G003537

[32] Wang, Z., and Grant, M. J., "Constrained Trajectory Optimization for Planetary Entry via Sequential Convex Programming," *AIAA Atmospheric Flight Mechanics Conference*, AIAA Paper 2016-3241, 2016.
https://doi.org/10.2514/6.2016-3241

[33] Wang, Z., and Grant, M. J., "Autonomous Entry Guidance for Hypersonic Vehicles by Convex Optimization," *Journal of Spacecraft and Rockets*, Vol. 55, No. 4, 2018, pp. 993–1006.
https://doi.org/10.2514/1.A34102

[34] Sagliano, M., and Mooij, E., "Optimal Drag-Energy Entry Guidance via Pseudospectral Convex Optimization," AIAA Paper 2018-1315, 2018.
https://doi.org/10.2514/6.2018-1315

[35] Calabuig, G. J. D., and Mooij, E., "Optimal On-Board Abort Guidance Based on Successive Convexification for Atmospheric Re-Entry," AIAA Paper 2021-0860, 2021.
https://doi.org/10.2514/6.2021-0860

[36] Zhang, K., Yang, S., and Xiong, F., "Rapid Ascent Trajectory Optimization for Guided Rockets via Sequential Convex Programming," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 233, No. 13, 2019, pp. 4800–4809.
https://doi.org/10.1177/0954410019830268

[37] Li, Y., Guan, Y., Wei, C., and Hu, R., "Optimal Control of Ascent Trajectory for Launch Vehicles: A Convex Approach," *IEEE Access*, Vol. 7, Dec. 2019, pp. 186,491–186,498.
https://doi.org/10.1109/ACCESS.2019.2960864

[38] Li, Y., Pang, B., Wei, C., Cui, N., and Liu, Y., "Online Trajectory Optimization for Power System Fault of Launch Vehicles via Convex Programming," *Aerospace Science and Technology*, Vol. 98, March 2020, Paper 105682.
https://doi.org/10.1016/j.ast.2020.105682

[39] Cheng, X., Li, H., and Zhang, R., "Efficient Ascent Trajectory Optimization Using Convex Models Based on the Newton-Kantorovich/Pseudospectral Approach," *Aerospace Science and Technology*, Vol. 66, July 2017, pp. 140–151.
https://doi.org/10.1016/j.ast.2017.02.023

[40] Benedikter, B., Zavoli, A., and Colasurdo, G., "A Convex Approach to Rocket Ascent Trajectory Optimization," *8th European Conference for Aeronautics and Space Sciences (EUCASS)*, Paper 430, 2019.
https://doi.org/10.13009/EUCASS2019-430

[41] Benedikter, B., Zavoli, A., Colasurdo, G., Pizzurro, S., and Cavallini, E., "Convex Approach to Three-Dimensional Launch Vehicle Ascent Trajectory Optimization," *Journal of Guidance, Control, and Dynamics*, Vol. 44, No. 6, 2021, pp. 1116–1131.
https://doi.org/10.2514/1.G005376

[42] *Vega User's Manual*, Arianespace, Evry-Courcouronnes Cedex, France, 2014.

[43] Garg, D., Patterson, M., Hager, W. W., Rao, A. V., Benson, D. A., and Huntington, G. T., "A Unified Framework for the Numerical Solution of Optimal Control Problems Using Pseudospectral Methods," *Automatica*, Vol. 46, No. 11, 2010, pp. 1843–1851. https://doi.org/10.1016/j.automatica.2010.06.048

[44] 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. https://doi.org/10.1002/oca.957

[45] Acikmese, B., and Ploen, S. R., "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1353–1366. https://doi.org/10.2514/1.27553

[46] Mao, Y., Szmuk, M., and Acikmese, B., "Successive Convexification: A Superlinearly Convergent Algorithm for Non-Convex Optimal Control Problems," arXiv preprint, arXiv:1804.06539, 2018.

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

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

[49] Alizadeh, F., and Goldfarb, D., "Second-Order Cone Programming," *Mathematical Programming*, Vol. 95, No. 1, 2003, pp. 3–51. https://doi.org/10.1007/s10107-002-0339-5

[50] Liu, X., Shen, Z., and Lu, P., "Entry Trajectory Optimization by Second-Order Cone Programming," *Journal of Guidance, Control, and Dynamics*, Vol. 39, No. 2, 2016, pp. 227–241. https://doi.org/10.2514/1.G001210

[51] Liu, X., Shen, Z., and Lu, P., "Exact Convex Relaxation for Optimal Flight of Aerodynamically Controlled Missiles," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 52, No. 4, 2016, pp. 1881–1892. https://doi.org/10.1109/TAES.2016.150741

[52] Yang, R., and Liu, X., "Comparison of Convex Optimization-Based Approaches to Solve Nonconvex Optimal Control Problems," *AIAA Scitech 2019 Forum*, AIAA Paper 2019-1666, 2019. https://doi.org/10.2514/6.2019-1666

[53] Betts, J. T., *Practical Methods for Optimal Control and Estimation Using Nonlinear Programming*, SIAM, Philadelphia, PA, 2010, Vol. 19, pp. 132–134, Chap. 4. https://doi.org/10.1137/1.9780898718577

[54] Garg, D., "Advances in Global Pseudospectral Methods for Optimal Control," Ph.D. Thesis, Univ. of Florida, Gainesville, FL, 2011.

[55] Berrut, J.-P., and Trefethen, L. N., "Barycentric Lagrange Interpolation," *SIAM Review*, Vol. 46, No. 3, 2004, pp. 501–517. https://doi.org/10.1137/S0036144502417715

[56] *Gurobi Optimizer Reference Manual*, Gurobi Optimization, LLC, Beaverton, Oregon, 2020.

[57] *U.S. Standard Atmosphere*, NOAA, NASA, and USAF, U.S. Government Printing Office, Washington, D.C., 1976.

[58] Federici, L., Benedikter, B., and Zavoli, A., "EOS: A Parallel, Self-Adaptive, Multi-Population Evolutionary Algorithm for Constrained Global Optimization," *2020 IEEE Congress on Evolutionary Computation (CEC)*, IEEE Publ., Piscataway, NJ, 2020, pp. 1–10. https://doi.org/10.1109/CEC48606.2020.9185800

[59] Federici, L., Zavoli, A., Colasurdo, G., Mancini, L., and Neri, A., "Integrated Optimization of Ascent Trajectory and SRM Design of Multistage Launch Vehicles," *Advances in the Astronautical Sciences*, Vol. 168, April 2019, pp. 733–752.

[60] Wang, Z., and Lu, Y., "Improved Sequential Convex Programming Algorithms for Entry Trajectory Optimization," *Journal of Spacecraft and Rockets*, Vol. 57, No. 6, 2020, pp. 1373–1386. https://doi.org/10.2514/1.A34640

[61] Szmuk, M., Reynolds, T. P., and Açkmeşe, B., "Successive Convexification for Real-Time Six-Degree-of-Freedom Powered Descent Guidance with State-Triggered Constraints," *Journal of Guidance, Control, and Dynamics*, Vol. 43, No. 8, 2020, pp. 1399–1413. https://doi.org/10.2514/1.G004549

[62] Benedikter, B., Zavoli, A., Colasurdo, G., Pizzurro, S., and Cavallini, E., "Autonomous Upper Stage Guidance Using Convex Optimization and Model Predictive Control," *AIAA ASCEND*, AIAA Paper 2020-4268, 2020. https://doi.org/10.2514/6.2020-4268

[63] Hartl, R. F., Sethi, S. P., and Vickson, R. G., "A Survey of the Maximum Principles for Optimal Control Problems with State Constraints," *SIAM Review*, Vol. 37, No. 2, 1995, pp. 181–218. https://doi.org/10.1137/1037043

M. A. Ayoubi
Associate Editor