DOI: 10.13009/EUCASS2019-430

A Convex Approach to Rocket Ascent Trajectory Optimization

Boris Benedikter[⋆†]*, Alessandro Zavoli*[⋆] *and Guido Colasurdo*[⋆]
[⋆]*Department of Mechanical and Aerospace Engineering, Sapienza University of Rome*
Via Eudossiana 18 - 00184, Rome, Italy
boris.benedikter@uniroma1.it · alessandro.zavoli@uniroma1.it · guido.colasurdo@uniroma1.it
[†]Corresponding author

Abstract

A convex programming approach for the ascent trajectory optimization of a multistage launch vehicle is presented. Since the problem cannot be readily solved as a convex optimization problem, a combination of lossless and successive convexification techniques is adopted to generate a sequence of convex subproblems having the same solution as the original one. Convergence towards the optimal solution is guaranteed in a limited, short, time by using highly efficient numerical algorithms. The requirement of free terminal time is explicitly accounted for in the optimization procedure. Numerical results are presented and compared with those provided by a differential evolution algorithm.

1. Introduction

This paper presents a convex programming approach to the optimization of the ascent trajectory of a multistage launch vehicle. The problem under investigation is of crucial importance in the aerospace industry since it aims at defining strategies for a safe and economically convenient access to space. Today, chemical propulsion is the only viable way to provide the large thrust required to achieve the target orbit. However, such propulsion systems only allow for injecting a small fraction of the rocket overall mass into orbit. Trajectory optimization is thus of primary importance for pushing the launcher performance towards its limits.

The optimization of a rocket ascent trajectory is a complex problem, as it is characterized by highly nonlinear dynamics, it is extremely sensitive with respect to the optimization variables, and multiple mission constraints must be satisfied. Both direct[4,10,17] and indirect[5,6,18] methods have been successfully applied to this problem. Despite being powerful and extremely accurate, indirect methods are usually disregarded in case of practical mission scenarios: the need for deriving analytically first-order necessary conditions and solving a complicated multi-point boundary value problem makes them scarcely appealing. Direct methods, which rely on a straightforward transcription of the continuous-time optimal control problem into a general nonlinear programming (NLP) problem where both states and controls are discretized, are commonly preferred as they are easier to setup and to solve. Unfortunately, due to the aforementioned problem high-sensitivity, convergence of these methods strongly depends on the accuracy of the first guess. As a further drawback, solving a general NLP problem takes a significant computational effort and the optimality of the attained solution is usually questionable, that is, only convergence towards a local minimum can be reasonably expected.

Convex optimization has gained increasing popularity in recent years in aerospace applications due to its unique theoretical advantages and availability of state-of-the-art algorithms for convex programming that allow for convergence in a limited, short, time.[13] As aerospace engineering problems do not usually appear to be convex, and most are not, the challenge for researchers in this field is to find out viable techniques to extend the applicability of these powerful optimization algorithms to newer and newer problems.

This paper proposes a methodology based on a combination of lossless and successive convexification techniques, and describes the detailed formulation for transforming the original nonconvex ascent trajectory optimization problem into a sequence of convex problems which converges to the same solution as the original one. In particular, a convenient change of variables is carried out first, in order to obtain a nonlinear control-affine formulation of the system dynamics. Furthermore, an exact constraint relaxation is exploited to transform a nonconvex control constraint into a convex one. Finally, the equations of motion are successively linearized around a previously found solution in

DOI: 10.13009/EUCASS2019-430

order to generate a sequence of convex sub-problems. In order for the successive convexification to work, the convex formulation includes virtual controls and an adaptive trust region.

The paper is organized as follows. Section 2 presents the dynamical model of the multi-stage rocket ascent problem and formulates the corresponding optimal control problem. In section 3 the convexification process of problem is presented along with proper safe-guarding modifications. Finally, preliminary results for a simplified rocket ascent trajectory are presented and compared with the solution provided by a differential evolution algorithm, showing the effectiveness and limits of the suggested approach.

2. Original Problem Formulation

In this section, the ascent problem is described and formulated as a continuous-time optimal control problem made up of a collection of phases.

2.1 Rocket Model

The complete description of the rocket dynamics comprises three translational and three rotational degrees of freedom. However, in the early phases of design, the trajectory of the center of mass of the vehicle is of greater interest than its attitude motion. Furthermore, the true attitude of the launcher affects only the first phases of the trajectory and its effect on the overall trajectory is minimal. Therefore, in this study, rotational dynamics is neglected and the vehicle is modeled as a point mass. The rocket axis is assumed to be aligned with the thrust direction at any time.

This paper investigates a planar (equatorial) two-dimensional problem, so, under these assumptions, the set of state variables is composed of the position vector $\boldsymbol{r}$, the velocity vector $\boldsymbol{v}$ and the launcher mass m. In order to favor an easier statement of the problem, the position vector is expressed in polar Earth-centered inertial (ECI) coordinates while the velocity vector in a Local-Vertical-Local-Horizontal (LVLH) frame. Therefore, the state vector $\boldsymbol{x}$ is:

$$\boldsymbol{x} = \begin{bmatrix} r & \theta & v_r & v_t & m \end{bmatrix} \tag{1}$$

where r is the geocentric distance, θ is the right ascension, i.e., the angular displacement from the launch pad initial position, and v_r and v_t are, respectively, the radial and transverse velocity components.

2.2 Phases

In this paper, a traditional two-stage launch vehicle is considered. The ascent trajectory of such a rocket can be split into several phases. Indeed, the launcher mission is made up of different flight conditions that force the vehicle to adopt diversified guidance laws over the ascent.

After a vertical liftoff, the rocket performs a programmed rotation (*pitch-over*) that allows the vehicle to begin a *zero-lift gravity-turn* (ZLGT) maneuver. During ZLGT, the rocket axis is assumed to be aligned at any instant with the relative-to-atmosphere velocity. The rocket flies with null angle of attack, hence minimizing the transverse aerodynamic load. Once the first stage burns out, it separates from the launcher and a brief coasting takes place before the ignition of the second stage. As the upper stage flies out of the atmosphere, it can jettison the payload fairing and follow an optimal guidance law, that corresponds to a Hohmann-like maneuver, composed of two firings separated by a long coasting phase.

The optimization of such a complex mission is handled by dividing the problem into the following phases:

1. Vertical ascent
2. Pitch-over
3. Zero-lift gravity turn
4. Coasting 1
5. First burn of stage 2
6. Coasting 2
7. Second burn of stage 2

The time-lengths of the first stage phases and of the first coasting are fixed, while the other time-lengths have to be optimized, still retaining the maximum overall burn time of the second stage.

DOI: 10.13009/EUCASS2019-430

2.3 Equations of Motion

The forces acting on the launch vehicle are the gravity $\boldsymbol{g}$, the aerodynamic drag $\boldsymbol{D}$ and the engine thrust $\boldsymbol{T}$. In this study, a basic gravitational model, which assumes a spherical Earth, has been considered. Effects of the gravitational perturbations on the ascent trajectory are deemed minimal, and more sophisticated gravitational models are used only in the advanced phases of trajectory design. The gravitational acceleration is thus expressed as:

$$\boldsymbol{g} = -\frac{\mu}{r^3}\boldsymbol{r} \tag{2}$$

where μ is Earth's gravitational parameter.

Drag is the only aerodynamic force included in the model. Indeed, rocket trajectories are designed to avoid high lateral loads. As a result, the lift is much smaller than the drag and it can be neglected without loss of accuracy. The drag force is computed as:

$$D = \frac{1}{2}C_D S \rho v_{rel}^2 \tag{3}$$

where C_D is the drag coefficient, S is the reference surface, ρ is the atmospheric density, and v_{rel} is the relative velocity with respect to the atmosphere. The drag coefficient is here assumed to be constant. An isothermal, exponential atmosphere model has been used for computing air density ρ and pressure p, with a scale height H equal to 8.4 km:

$$\rho = \rho_0 \exp\left(-\frac{r - r_0}{H}\right) \tag{4}$$

$$p = p_0 \exp\left(-\frac{r - r_0}{H}\right) \tag{5}$$

where p_0 and ρ_0 are the reference values of pressure and density at sea level ($r = r_0$). Finally, the atmosphere is assumed to be rotating along with the Earth, so the relative velocity is given by:

$$\boldsymbol{v}_{rel} = \boldsymbol{v} - \boldsymbol{\omega_E} \times \boldsymbol{r} \tag{6}$$

where $\boldsymbol{\omega_E}$ is Earth's angular velocity.

The only control on the system is the thrust, that can be expressed as the product of the thrust magnitude T multiplied by its direction $\hat{\boldsymbol{T}}$. The thrust magnitude depends on the engine and on the altitude. In particular, the engine is characterized by a vacuum thrust T_{vac}, which is a function of time, and is unique for each engine. The actual thrust on the system depends on the external pressure p as:

$$T = T_{vac} - pA_e = T_{vac} - T_p \tag{7}$$

where A_e is the nozzle exit area and $T_p = pA_e$.

The rocket propulsion is characterized by a continuous propellant mass ejection. The mass flow rate is related to the engine vacuum thrust via the effective exhaust velocity $c = g_0 I_{sp}$, where g_0 is the gravity acceleration at sea level and I_{sp} is the specific impulse in vacuum:

$$\dot{m} = -\frac{T_{vac}}{c} \tag{8}$$

In this paper, the vacuum thrust magnitude is pre-assigned for each stage, so the control $\boldsymbol{u}$ to be optimized is the thrust direction. In a two-dimensional model, the thrust direction can be expressed as:

$$\boldsymbol{u} = \hat{\boldsymbol{T}} = \begin{bmatrix} \hat{T}_r & \hat{T}_t \end{bmatrix} \tag{9}$$

where $\hat{T}_r$ and $\hat{T}_t$ are the radial and transverse components. Since $\hat{\boldsymbol{T}}$ is a unit vector, the following path constraint must be ensured at any point in time:

$$\hat{T}_r^2 + \hat{T}_t^2 = 1 \tag{10}$$

During the gravity turn phase of the ascent trajectory the thrust direction is forced to be parallel to the relative velocity. In order maintain the same equations of motion across all phases, the thrust magnitude T is fictitiously split into two contributes, T_a and T_b. T_a represents the optimally controlled thrust contribution, while T_b is always parallel to the relative velocity. It can be noticed that T_a and T_b are alternatively null: during the zero-lift arcs T_a is zero, while T_b is equal to the real thrust magnitude; conversely, $T_a = T$ and $T_b = 0$ during the other propelled arcs.

DOI: 10.13009/EUCASS2019-430

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

$$\dot{r} = v_r \tag{11}$$

$$\dot{\theta} = \frac{v_t}{r} \tag{12}$$

$$\dot{v_r} = \frac{v_t^2}{r} - \frac{\mu}{r^2} + \frac{T_a}{m}\hat{T}_r + \frac{T_b - D}{m}\frac{v_r}{v_{rel}} \tag{13}$$

$$\dot{v_t} = -\frac{v_r v_t}{r} + \frac{T_a}{m}\hat{T}_t + \frac{T_b - D}{m}\frac{v_t - \omega_E r}{v_{rel}} \tag{14}$$

$$\dot{m} = -\frac{T_{vac}}{c} \tag{15}$$

2.4 Objective

The main goal of the optimization is to determine the control law that maximizes the payload injected into orbit. In the present paper the propellant and inert masses, m_p and m_{dry}, are assigned. Hence, assuming that all the propellant is being used, one can equivalently maximize either the initial or final mass since they differ from the payload mass by a constant value. Hereafter, we will consider the latter case, so the objective function J to be minimized is:

$$J = -m(t_f) \tag{16}$$

where t_f denotes the (free) final time.

2.5 Constraints

In addition to the differential constraints associated with the equations of motion, boundary and path constraints need to be taken into account in the optimization procedure. The initial position and velocity are assigned, while the initial mass has to be optimized as it is strictly related to the objective function. The launcher lifts off at the equator from sea level at $\theta = 0°$, with null velocity with respect to the Earth. However, in the inertial frame the launcher has an initial transverse velocity equal to the Earth angular velocity contribution. At the final time the rocket must place the payload into a target equatorial circular orbit of assigned semi-major axis. Hence the final radius and velocity vector are assigned, while the angular position is unconstrained.

Since the problem is divided into several arcs, some linkage conditions must be enforced. All the state variables are continuous across phase boundaries, with the relevant exception of the launch vehicle mass at separation, when both the payload fairing and the first stage inert mass are detached. The constraint on the mass at the boundary 3-4 is:

$$m_+ = m_- - (m_{dry,1} + m_{fairing}) \tag{17}$$

where m_+ and m_- denote the system mass respectively after and before separation.

Finally, the upper stage burn is partitioned between two phases (5 and 7). In order to consume all the propellant, the total duration of the two firings must be equal to the second stage engine burn time $t_{b,2}$, hence the following constraint has to be imposed:

$$\Delta t_5 + \Delta t_7 = t_{b,2} \tag{18}$$

3. Convex Transcription

In this section, the convex optimization problem is formulated. First, the continuous-time problem stated in section 2 is transcribed into a finite-dimensional problem. Second, where possible, lossless convexification is performed. Finally, the remaining nonconvexities are handled by means of successive linearization.

3.1 Discretization

The optimal control problem stated in section 2 is infinite-dimensional since state and control variables are continuous-time functions. However, numerical methods for solving optimization problems require a *finite* set of variables and constraints. In order to convert the optimal control problem into a finite-dimensional problem, a direct transcription method is used.

DOI: 10.13009/EUCASS2019-430

The independent variable, i.e., time, is discretized by dividing the k-th phase into $M^{(k)} - 1$ intervals. So, omitting from now on the superscript k, one obtains M points:

$$t_0 = t_1 < \cdots < t_M = t_f \tag{19}$$

Each point t_j is referred to as *node* of the *mesh*. Note that, the nodes do not have to be equally spaced. In fact, in order not to generate a large discrete problem, the mesh should be dense only in the intervals where a small number of nodes would produce an inaccurate discrete approximation.

In order to set up the grid, initial and final instants, t_0 and t_f, must be known for each phase. However, if the duration of the phases is variable, these instants are unknown a priori. Therefore, to address the free time problem, each arc must be converted into one on a fixed domain. First, additional variables are added to the set of optimization variables to include the duration of each phase $\Delta t = t_f - t_0$. Second, the following time transformation is considered:

$$t = t_0 + \left(t_f - t_0\right)(\tau - \tau_0) = t_0 + \Delta t\,(\tau - \tau_0) \tag{20}$$

where τ is the new independent variable defined in the fixed domain $\tau_0 \leq \tau \leq \tau_f$, with $\tau_0 = k - 1$ and $\tau_f = k$ for each phase $k = 1, \ldots, N$. As a result, each discretization stepsize becomes:

$$h_j = t_{j+1} - t_j = \left(\tau_{j+1} - \tau_j\right)\Delta t \tag{21}$$

The optimization process will determine the optimal value of Δt and, hence, the actual initial and final time of each phase.

Once the grid is defined, both state and control variables are discretized over it. The differential constraints are replaced by a finite set of algebraic constraints, or *defect* constraints. A simple trapezoidal integration scheme is employed in the current application, and the resulting defect constraint between nodes j and $j + 1$ is:

$$\boldsymbol{x}_{j+1} - \left(\boldsymbol{x}_j + \frac{h_j}{2}\left(\boldsymbol{f}_j + \boldsymbol{f}_{j+1}\right)\right) = \boldsymbol{0} \tag{22}$$

Similarly, path constraints, such as (10), are converted into a finite set of algebraic constraints by imposing them at each mesh node.

3.2 Lossless Convexification

A convex optimization problem is characterized by a convex objective function, linear equality constraints, and inequality constraints that define a convex feasible set. The problem under investigation cannot be readily solved by means of convex programming algorithms. In fact, it has to be converted into a convex problem. In this application, the original problem is converted into a special class of convex programming problems, a Second-Order Cone Programming (SOCP) problem. A SOCP problem has a linear objective, linear equality constraints and second-order cone constraints. This class of programming problems allows for representing quite complex constraints and can be solved by means of highly-efficient interior point methods, even for a large number of variables.[2]

The discretized optimal control problem is still a nonconvex optimization problem. Nonconvexities arise from the path constraint (10) and the nonlinear dynamics (11–15). In this paragraph, a convenient change of variables and a constraint relaxation is performed in order to eliminate the path constraint nonconvexity and to obtain a control-affine dynamical system.

3.2.1 Change of Variables

A change of variables is carried out to replace nonlinear terms in the dynamics by linear terms and obtain a control-affine dynamical system. The new control variables are introduced:

$$u_r = \frac{T_a}{m}\hat{T}_r \tag{23}$$

$$u_t = \frac{T_a}{m}\hat{T}_t \tag{24}$$

$$u_N = \frac{T_a}{m} \tag{25}$$

Notice that these new variables are defined as a function of T_a, that includes both the vacuum thrust $T_{a,vac}$ and the external pressure term $T_{a,p}$. So, the new control vector is:

$$\boldsymbol{u} = \begin{bmatrix} u_r & u_t & u_N \end{bmatrix} \tag{26}$$

DOI: 10.13009/EUCASS2019-430

By introducing u_r and u_t in (13, 14), we obtain the following control-affine equations:

$$\dot{v}_r = \frac{v_t^2}{r} - \frac{\mu}{r^2} + u_r + \frac{(T_b - D)}{m}\frac{v_r}{v_{rel}} \tag{27}$$

$$\dot{v}_t = -\frac{v_r v_t}{r} + u_t + \frac{(T_b - D)}{m}\frac{v_t - \omega_E r}{v_{rel}} \tag{28}$$

However, replacing u_N in (15) would not produce the same effect. A further step is required, and a new state variable is defined:[12]

$$z = \ln m \tag{29}$$

Now, by differentiating (29) and combining it with (15) one obtains:

$$\dot{z} = \frac{\dot{m}}{m} = -\frac{T_{a,vac} + T_{b,vac}}{mc} = -\frac{u_N + T_{a,p}e^{-z}}{c} - \frac{T_{b,vac}}{c}e^{-z} \tag{30}$$

that is an affine function of the control variables, and nonlinear in the state variables.

3.2.2 Constraint Relaxation

The equality path constraint (10) is rewritten in terms of the new control variables, leading to:

$$u_r^2 + u_t^2 = u_N^2 \tag{31}$$

that is a nonlinear, nonconvex, equality constraint.

A common convexification technique consists of relaxing a nonconvex constraint of the original problem into a convex constraint (*constraint relaxation*). In general, the convex relaxation defines a larger feasible set. Nonetheless, in many applications, the optimal solution remains the same as the original problem, thus the relaxation is said to be *exact*.[1]

The control constraint (31) is suitable for an exact relaxation, thus leading to the following inequality constraint:

$$u_r^2 + u_t^2 \leq u_N^2 \tag{32}$$

that corresponds to a second-order cone constraint. Notice that this approach allows to preserve the original problem nonlinearity in the convex formulation. This is particularly advantageous since it has been shown that keeping some nonlinearity can significantly favor the convergence of a successive convexification algorithm.[19]

Finally, the newly defined u_N variable should be bounded by the maximum thrust of the engine.

$$0 \leq u_N \leq \left(T_{vac,max} - T_p\right)e^{-z} \tag{33}$$

However, this constraint is nonconvex and it does not lend itself to a relaxation, so it has to be linearized around a reference solution (k) as:

$$0 \leq u_N \leq \left(T_{vac,max} - T_p^{(k)}\right)e^{-z^{(k)}}\left(1 - \left(z - z^{(k)}\right)\right) + T_p^{(k)}e^{-z^{(k)}}\frac{r - r^{(k)}}{H} \tag{34}$$

The constraints (32) and (34) form altogether a three-dimensional geometric half-cone in the control space, as shown in figure 1.

Furthermore, note that the constraint (34) nominally allows burn phases for including low-thrust or even coasting arcs. This could have been exploited to merge the three second stage phases into a single one. However, this approach does not suit well the problem under investigation since the optimal duration of the second burn results to be much smaller than the ones of the other two phases. So, when setting up the mesh, a particular attention should have been posed on where to place the discretization nodes. Indeed, propelled phases require a much more dense grid than coasting ones. Thus, either a very dense uniform grid should have been used all along the second stage arc, though generating a large number of variables and introducing convergence problems, or an *a priori* increase of the mesh density should have been performed in correspondence of the actual burns. Instead, the division into two firing phases and a coasting one gives more robustness to the sequential solution algorithm, and permits to handily choose an adequate number of nodes for the propelled arcs.

DOI: 10.13009/EUCASS2019-430

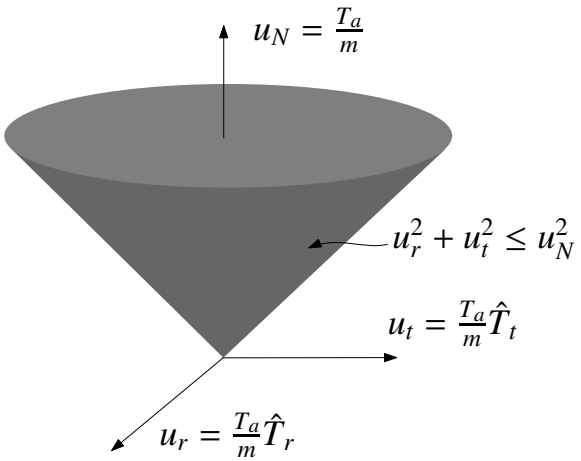


Figure 1: Relaxed control constraint results in a second-order cone constraint.

3.3 Linearization

3.3.1 Time-Fixed Phases

At this point, the only nonconvexity left is due to the nonlinear dynamics. A popular method is to simply use successive linearization around a reference solution (k):

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \approx \boldsymbol{f}\left(\boldsymbol{x}^{(k)}, \boldsymbol{u}^{(k)}, t\right) + \boldsymbol{f}_{\boldsymbol{x}}\left(\boldsymbol{x}^{(k)}, \boldsymbol{u}^{(k)}, t\right)\left(\boldsymbol{x} - \boldsymbol{x}^{(k)}\right) + \boldsymbol{f}_{\boldsymbol{u}}\left(\boldsymbol{x}^{(k)}, \boldsymbol{u}^{(k)}, t\right)\left(\boldsymbol{u} - \boldsymbol{u}^{(k)}\right) \tag{35}$$

However, the change of variables introduced in section 3.2.1 leaded to control-affine dynamics, so the linearization can be reduced to:

$$\dot{\boldsymbol{x}} = \hat{\boldsymbol{f}}(\boldsymbol{x}, t) + B(\boldsymbol{x}, t)\,\boldsymbol{u} \approx \hat{\boldsymbol{f}}\left(\boldsymbol{x}^{(k)}, t\right) + \hat{\boldsymbol{f}}_{\boldsymbol{x}}\left(\boldsymbol{x}^{(k)}, t\right)\left(\boldsymbol{x} - \boldsymbol{x}^{(k)}\right) + B\left(\boldsymbol{x}^{(k)}, t\right)\boldsymbol{u} \tag{36}$$

where $\hat{\boldsymbol{f}} = \boldsymbol{f}(\boldsymbol{u} = \boldsymbol{0})$ and B is the $(n_x \times n_u)$ control coefficients matrix. Equation (36) can be written as:

$$\dot{\boldsymbol{x}} \approx A\left(\boldsymbol{x}^{(k)}, t\right)\boldsymbol{x} + B\left(\boldsymbol{x}^{(k)}, t\right)\boldsymbol{u} + \boldsymbol{c}\left(\boldsymbol{x}^{(k)}, t\right) \tag{37}$$

where $A = \hat{\boldsymbol{f}}_{\boldsymbol{x}}\left(\boldsymbol{x}^{(k)}, t\right)$ is a $(n_x \times n_x)$ matrix and $\boldsymbol{c}$ is a n_x vector:

$$\boldsymbol{c} = \hat{\boldsymbol{f}}(\boldsymbol{x}^{(k)}, t) - A\boldsymbol{x}^{(k)} \tag{38}$$

Equation (37) represents the linearized dynamics. Notice that the coefficients of the dynamics depend only on the reference state $\boldsymbol{x}^{(k)}$ and not on the controls $\boldsymbol{u}^{(k)}$. This provides robustness to a sequential solution method, since intermediate controls, which can come with high-frequency jitters,[14] do not affect the dynamics in the next iteration.

3.3.2 Free-Time Phases

In order to address the free-time problem, a further step has to be taken. To optimize the time duration Δt of the free-time phases the independent variable has to be changed into τ, as defined in equation (20), and the equations of motion become:[16]

$$\boldsymbol{x}' = \frac{d\boldsymbol{x}}{d\tau} = \frac{d\boldsymbol{x}}{dt}\frac{dt}{d\tau} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t) \cdot \Delta t \tag{39}$$

By linearizing equation (39) we obtain:

$$\boldsymbol{x}' \approx \left(A\left(\boldsymbol{x}^{(k)}, t\right)\boldsymbol{x} + B\left(\boldsymbol{x}^{(k)}, t\right)\boldsymbol{u} + \boldsymbol{c}\left(\boldsymbol{x}^{(k)}, t\right)\right)\Delta t^{(k)} + \boldsymbol{f}\left(\boldsymbol{x}^{(k)}, \boldsymbol{u}^{(k)}, t\right)\left(\Delta t - \Delta t^{(k)}\right) \tag{40}$$

These are the linearized dynamics to be considered in the free-time phases. Contrarily to fixed-time phases, the dynamics of free-time phases are dependent upon the controls of the previous iterations, thus we expect the optimization of such arcs to be less robust than the others. Nonetheless, this can be partially mitigated by penalizing the Δt variation across successive iterations (see section 3.4.2). Indeed, if $\Delta t^{(k+1)} \approx \Delta t^{(k)}$, the linearized equations (40) tend to be the same as the time-constrained ones.

DOI: 10.13009/EUCASS2019-430

3.4 Successive Convexification

While the constraint relaxation discussed in section 3.2.2 leads to a problem with the same solution as the original one, the linearization introduced in section 3.3 produces a convex problem that, in general, is an approximation of the original problem. In order to converge towards the solution of the original optimal control problem, a sequence of convex subproblems is generated by updating iteratively the linearization reference solution. However, for the accomplishment of the successive convexification, two safe-guarding modifications should be added: *virtual controls* and a *trust region*.

3.4.1 Virtual Controls

The linearization of a nonconvex problem may result in an infeasible convex subproblem. This phenomenon is referred to as *artificial infeasibility*,[15] and it usually takes place in the first iterations of the sequence. In order to remove this infeasibility, an additional unconstrained term $\boldsymbol{v}$ is added to the linearized dynamics:

$$\boldsymbol{x}' \approx \left(A\left(\boldsymbol{x}^{(k)}, t\right)\boldsymbol{x} + B\left(\boldsymbol{x}^{(k)}, t\right)\boldsymbol{u} + \boldsymbol{c}\left(\boldsymbol{x}^{(k)}, t\right)\right)\Delta t^{(k)} + \boldsymbol{f}\left(\boldsymbol{x}^{(k)}, \boldsymbol{u}^{(k)}, t\right)\left(\Delta t - \Delta t^{(k)}\right) + E\boldsymbol{v} \tag{41}$$

where E is a $(n_x \times n_x)$ matrix, selected in such a way that any state $\boldsymbol{x} \in \mathbb{R}^{n_x}$ is reachable in finite time.

Evidently, these controls should be used only when necessary by the SOCP solver. In order to ensure a correct behavior, an augmented objective function L should be used:

$$L = J + \lambda_v \sum_{i=1}^{N} \sum_{j=1}^{M^{(i)}} \sum_{k=1}^{n_x^{(i)}} \left| \sum_{m=1}^{n_x^{(i)}} E_{km} v_m(t_j) \right| \tag{42}$$

where λ_v is the penalty weight. The choice of λ_v is a trade-off between the need to pick a sufficiently high coefficient, in order to discourage the use of virtual controls, and the will of choosing a value small enough not to shadow the original problem objective J.

As a remark, since the convex subproblems are obtained also by linearizing the nonlinear path constraints, such as the maximum thrust constraint (34), artificial infeasibility may arise from such approximation too. In this case other variables (referred to as *virtual buffers*) should be added to the model and included in the augmented cost function similarly as the virtual controls. However, such infeasibility did not show up for the rocket ascent trajectory optimization, therefore the virtual buffers have not been included in the convex formulation.

3.4.2 Trust Region

Another undesired effect of the linearization is the *artificial unboundedness*. This phenomenon takes place due to the fact that the linearized constraints are a valid approximation of the original ones only in the neighborhood of the reference solution. In order to mitigate the risk of artificial unboundedness the search space must be limited to a so-called trust region.

Two main types of trust region constraints exists: the *hard* trust region and the *soft* trust region. A hard trust region constraint limits the search space of an optimization variable to an *a priori* fixed radius δ:

$$|x_i - x_i^{(k)}| \leq \delta \tag{43}$$

Instead, a soft trust region constraint, that has the same expression as (43), does not require *a priori* the δ radius value, as it is automatically determined by the optimization process. This is carried out by including an additional term in the augmented cost function:

$$L = J + \lambda_\delta \delta \tag{44}$$

where λ_δ is the penalty weight assigned to the radius.

Each of the two approaches has its own pros and cons. The hard trust region allows to solve the problem to full optimality if the global solution belongs to the trust region. However, in order to prevent artificial unboundedness, generally δ has to be sufficiently small, thus the reference solution must be close enough to the real solution. On the other hand, the soft region method is easier to setup but the penalty weight of δ has to be sufficiently small in order not to prevail over the original objective function and thus converge towards a suboptimal solution.

Due to the launcher problem complexity a more sophisticated trust region has to be implemented. Indeed, the equations of motion include highly nonlinear terms, such as gravity and aerodynamic forces, that require particular attention. Furthermore, the free-time phases introduce additional instability to the optimization procedure. The adaptive trust region algorithm proposed by Mao et al.[15] revealed to be very effective and it is exploited in the present work.

DOI: 10.13009/EUCASS2019-430

A hard trust region constraint is enforced on all control variables, thus limiting the variation of the controls between two successive iterations. The trust radius value is updated at every iteration according to the quality of the linear approximation. Specifically, when the linearization error is big, the trust region is shrunk and the new solution may be even rejected, thus the iteration is repeated with the new trust radius. Otherwise, if the convex subproblem is believed to well approximate the original problem, the reference solution is overridden and the trust radius is retained or increased.

In order to estimate the linearization error at each iteration the following ratio $\rho^{(k)}$ is evaluated:

$$\rho^{(k)} = \frac{\Delta\hat{J}^{(k)}}{\Delta L^{(k)}} = \frac{\hat{J}^{(k-1)} - \hat{J}^{(k)}}{L^{(k-1)} - L^{(k)}} \tag{45}$$

where L is the augmented cost function of the convex subproblem and $\hat{J}$ is an augmented objective function that includes a measure of the linearized dynamics error. In this application, the following $\hat{J}$ is considered:

$$\hat{J} = J + \lambda_f \sum_{i=1}^{N} \sum_{j=1}^{M^{(i)}-1} \sum_{k=1}^{n_x^{(i)}} \left| x_k(t_{j+1}) - \left(x_k(t_j) + \frac{t_{j+1} - t_j}{2} \left(f_k(t_j) + f_k(t_{j+1}) \right) \right) \right| \tag{46}$$

In equation (46) λ_f is a penalty weight and $\boldsymbol{f}$ is the right hand side of the original nonlinear equations of motion. The ratio $\rho^{(k)}$ is used to determine the quality of the convex subproblem approximation since it compares the nonlinear cost reduction $\Delta\hat{J}^{(k)}$ with the subproblem cost reduction $\Delta L^{(k)}$. When this ratio is close to unity, then the approximation is assumed to be accurate. Notice that if $\rho^{(k)}$ is greater than unity the linear cost reduction under-predicts the actual $\Delta\hat{J}$, and is thus conservative, hence the trust radius can be increased. Only when $\rho^{(k)}$ is significantly smaller than one the trust region should be shrunk. The successive convexification algorithm is terminated when the nonlinear cost reduction $\Delta\hat{J}$ goes below a tolerance ϵ_J.

Finally, in order to address the free-time problem, the Δt change has been included in the augmented convex subproblem cost function L. Indeed, since the equations of motions are formulated as in (40), the solver may exploit the Δt variation to artificially violate the dynamics. So, as already done for the virtual controls, the following augmented objective function is defined for each subproblem:

$$L = J + \lambda_v \sum_{i=1}^{N} \sum_{j=1}^{M^{(i)}} \sum_{k=1}^{n_x^{(i)}} \left| \sum_{m=1}^{n_x^{(i)}} E_{km} v_m(t_j) \right| + \lambda_{\Delta t} \sum_{i=1}^{N} \left| \Delta t_i - \Delta t_i^{(k)} \right| \tag{47}$$

The penalty weight $\lambda_{\Delta t}$ has to be chosen carefully in order, on the one hand, not to let the time duration vary excessively among successive iterations, thus making the problem unbounded, and, on the other hand, not to freeze the time-lengths and allow the algorithm for determining the optimal values.

3.4.3 Initialization

By definition, the local minimum of a convex problem is also its global minimum. Hence, contrarily to general nonlinear problems, the first guess choice does not affect the converged solution quality. This is a unique feature that indirect and general NLP optimization methods do not share. However, the successive convexification algorithm requires an initial reference solution. For the rocket ascent problem a reasonable starting solution is needed to achieve convergence. Nonetheless, by adopting a sequential initialization procedure, an easy first guess, far from the optimal solution, can be provided.

Initially the successive convexification algorithm is applied to a simplified version of the rocket ascent problem without the Earth's atmosphere and with time-fixed phases. The starting reference solution for this problem is obtained through numerical integration of some guessed initial conditions, and easy control laws. A few remarks should be done. First, the numerical integration is carried out using the original problem dynamics (11–15). Second, since the first guess has been obtained by an accurate numerical integration method, the corresponding dynamical error is below tolerance. This may cause some problems in the successive convexification algorithm, since better solutions in terms of J may be rejected due to a worse violation of the dynamics, and hence a worse augmented index $\hat{J}$. So, a slight modification has been made on the original Mao et al.[15] algorithm: the first convex subproblem solution cannot be rejected. Finally, this problem is solved as time-fixed, on the one hand, to provide a greater robustness to the optimization and, on the other one, to prevent the optimizer from finding "falling-back" trajectories, i.e., trajectories that during the second stage coasting phase descend towards low altitudes, exploiting the absence of the atmosphere. Indeed, the latter trajectories are not suitable initializations for the successive (atmospheric) convexification.

Next, the optimal solution with no atmosphere is used as initial reference solution for solving the atmospheric problem but still with fixed time-lengths. Only when a time-fixed in-atmosphere solution is obtained as reference, the time intervals are freed and a last successive convexification is performed.

DOI: 10.13009/EUCASS2019-430

3.5 Mesh Refinement

All the steps described in 3.4.3 are carried out on the same mesh. However, once the final solution is obtained, the quality of the discrete solution must be formally inspected and eventually a mesh refinement process has to be carried out in order to meet the desired tolerances.

To evaluate the discretization error, the discrete solution must be converted into a continuous-time solution $(\tilde{\boldsymbol{x}}(t), \tilde{\boldsymbol{u}}(t))$ that approximates the real (unknown) solution. The state $\boldsymbol{x}(t)$ is approximated as a vector of cubic splines, with the conditions:

$$\tilde{\boldsymbol{x}}(t_j) = \boldsymbol{x}(t_j) \tag{48}$$

$$\frac{d}{dt}\tilde{\boldsymbol{x}}(t_j) = \boldsymbol{f}\left(\boldsymbol{x}(t_j), \boldsymbol{u}(t_j), t_j\right) \tag{49}$$

Instead, the control is represented as a linear interpolation of the node values. Moreover, the control is assumed to be correct and optimal.

At this point, the error between $\tilde{\boldsymbol{x}}(t)$ and the true solution is computed as:

$$\boldsymbol{\eta}_j = \int_{t_j}^{t_{j+1}} |\boldsymbol{\varepsilon}(t)| dt \tag{50}$$

where:

$$\boldsymbol{\varepsilon}(t) = \dot{\tilde{x}}(t) - \boldsymbol{f}[\tilde{\boldsymbol{x}}(t), \tilde{\boldsymbol{u}}(t)] \tag{51}$$

The integral (50) is computed using a very accurate quadrature method, with tolerance close to machine precision. If the error is above a given tolerance, the grid has to be refined. This is carried out by adding new nodes to the mesh.

It has been observed that a basic refinement approach, such as simply taking twice as much intervals in each phase, caused convergence problems of the successive convexification algorithm. Hence, it is important to add as few points as possible. Many techniques for choosing the new mesh nodes have been proposed over the years. The Betts and Huffman[3] approach has been used. This method selects new grid points by solving an integer programming problem. In particular, new points are selected to minimize the maximum discretization error by subdividing the current grid. This approach guarantees that the grid size among successive refinements does not increase much, and only in the intervals above tolerance.

As a remark, the mesh refinement process is carried out only on the atmospheric, free-time solution. So, the overall process remains computationally efficient. Furthermore, the initial reference solution on each new mesh is automatically obtained by interpolating the previous solution at the adjacent nodes. The only attention has to be addressed towards the first mesh choice that, on the one hand, must be sufficiently dense to approximate accurately the continuous-time problem and, on the other, it has to be coarse enough not to cause convergence problems.[11]

4. Numerical Results

In this section, the numerical results are provided in order to demonstrate the effectiveness of the proposed approach. The presented algorithm was implemented in C++ using Gurobi[9] as SOCP solver. For comparison, the same nonconvex problem was solved via an optimization algorithm based on Differential Evolution (DE), that proved to be effective in other space trajectory optimization problems.[7,8]

The optimization is based on SpaceX Falcon 9 Full Thrust two-stage rocket. The data used for the simulation are based on publicly available data retrievable on the internet and are reported in table 1. In addition, the fairing mass is equal to 1700 kg, the drag coefficient C_D is assumed to be constant and equal to 0.329, while the reference surface S is 10.52 m^2.

Table 1: Rocket data

Stage	m_{dry} [kg]	m_p [kg]	T_{vac} [kN]	t_b [s]	c_{vac} [km/s]	A_e [m^2]
1	22 200	410 900	8227	162	3.244	11.039
2	4000	107 500	934	397	3.449	1.227

The mission under investigation is a two-dimensional rocket ascent in the equatorial plane. Starting from sea level with null relative velocity the launcher must reach a 700 km equatorial circular orbit. The nominal time-lengths of time-fixed arcs are reported in table 2.

DOI: 10.13009/EUCASS2019-430

Table 2: Nominal time-lengths of time-fixed arcs

Phase	Liftoff	Pitch-Over	ZLGT	Coasting 1
Δt [s]	5.00	13.67	143.33	1.00

4.1 First Guess Solution

As already mentioned in section 3.4.3, a first reference solution must be provided to the successive convexification algorithm. Such trajectory is obtained as a numerical integration of a reasonable choice of the initial conditions, the controls and the other mission parameters. While picking the aforementioned variables, the main attention has been posed on finding a trajectory with an altitude profile always above the sea level. Such requirement did not toughen the process. In fact, by using smaller-than-optimal values of the launcher initial mass it is relatively easy to find such trajectories. Evidently, this solution corresponds to a low payload mass (approximately 3500 kg) and it does not reach the target orbit, but this does not represent a problem for the optimization procedure.

The tentative control laws are very simple. The elevation, i.e., the angle between the thrust direction and the local horizontal, at the end of the pitch-over is set to 89.5°. Then, during the first firing of the upper stage it varies linearly from 20° to 0°. Finally, it is fixed to zero during the last firing of the second stage.

The time-lengths of the second stage phases have been set to the typical values of a two-stage launcher. Specifically, since the upper stage is expected to perform a Hohmann-like maneuver, the coasting arc is assumed to be quite long, hence its duration is fixed to 3000 s. Moreover, most of the second stage burn will be executed during the first firing, while only a small fraction of it will be left for the second firing. This is due to the high thrust-to-mass ratio that characterizes the rocket at the end of the first firing. Hence, a 5.5 s duration is assigned to the injection burn while the remaining burn time (391.5 s) is assumed for the first burn.

The resulting reference trajectory is shown in figure 2. Clearly, the optimal solution will be very different from it. Nonetheless, it is a valid initial solution for the successive convexification. In fact, this further proves the robustness of the proposed methodology with respect to the initialization.

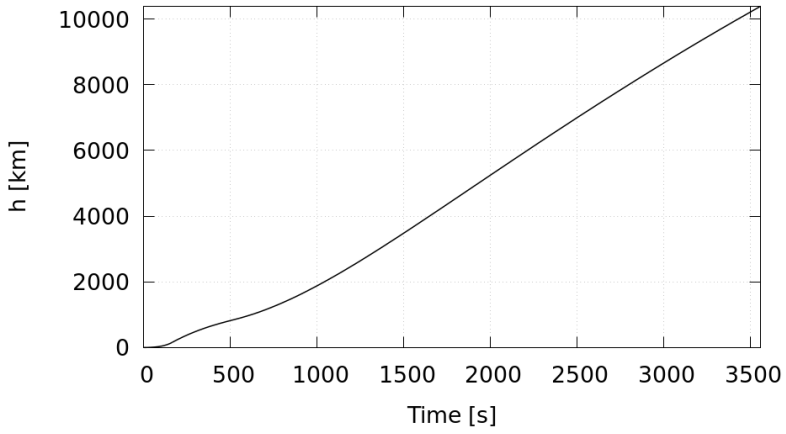


Figure 2: Altitude profile of the first guess solution.

4.2 Convergence Behavior

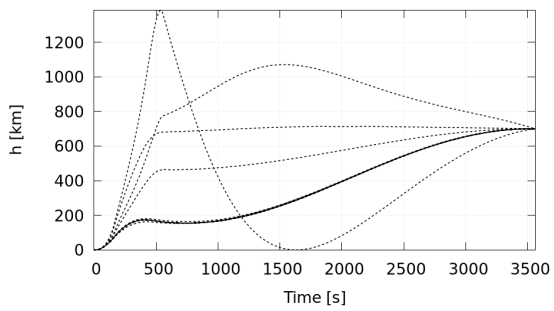
As mentioned in section 3.4.3, in order to make the procedure more robust with respect to the initialization, the successive convexification algorithm is executed three times passing gradually from a non-atmospheric time-fixed problem to an atmospheric free-time problem. In figure 3 the iteration sequences for the three processes are reported. It can be observed that the greatest variations take place in the first sequence (fig. 3a). Indeed, the initial reference solution is a very different trajectory from the optimal one, so the algorithm has to do a significant work in order to move and converge towards the optimal solution. Convergence is achieved in 85 iterations (41 accepted).

Figure 3b shows how the successive convexification algorithm moves from the no-atmosphere optimal solution to the atmospheric optimal trajectory. The variation across the iterations is smaller than in the previous sequence, but still noticeable. In fact, the atmospheric drag and the reduced thrust significantly affect the launcher dynamics. In

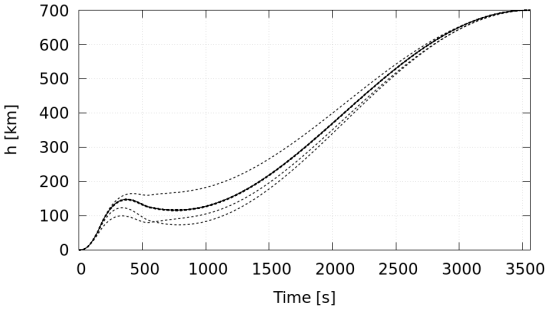
DOI: 10.13009/EUCASS2019-430

particular, the drag is a highly nonlinear term that, added to the equations of motion, makes the problem unbounded. Nonetheless, the adaptive trust region provides stability to the algorithm and permits to find the optimal solution. The successive convexification is terminated after reaching the a priori limit of 100 iterations. However, the best, acceptable solution, i.e., below tolerances, is found after 55 iterations (34 accepted).

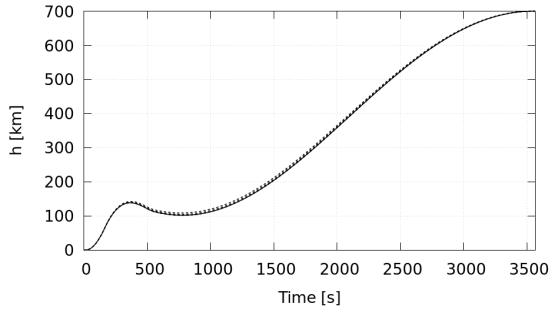
In order to permit the non-atmospheric to atmospheric transition, the time-lengths of the second stage phases had to remain fixed. However, the assigned times are only (good) guesses; the optimal times should be determined by the optimization procedure. This is done in a last step: starting from the time-fixed solution, the arc lengths are freed and the successive convexification is called again. Convergence is attained in 24 iterations (5 accepted). The trajectory changes as reported in figure 3c. The variation is small, due to the fact that also the time-lengths do not vary much. Indeed, the launcher problem presents many sub-optimal solutions with different sets of time-lengths. So, the optimization procedure may encounter difficulties to move from one local minimum to another one. However, as proven also by the differential evolution results, that are characterized by a set of times different from the one obtained, the payload mass is very close to the optimum value. Hence, even though convergence is attained towards a sub-optimal solution, the trajectory quality is still extremely good.



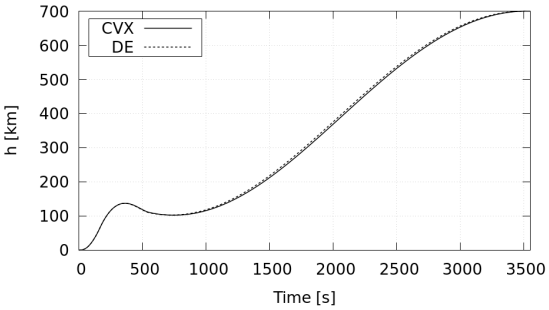
(a) Starting from the first guess trajectory and solving the time-fixed, with no atmosphere problem.



(b) Starting from the no-atmosphere solution and solving the time-fixed atmospheric problem.



(c) Starting from the atmospheric time-fixed solution and solving the free-time problem.



(d) Final refined trajectory versus the solution obtained with the DE algorithm.

Figure 3: Altitude profile across the iteration sequences. The dashed curves denote the intermediate solutions while the solid one represents the converged solution.

The penalty weights assigned to virtual controls and time variations and the initial trust radius on the controls are reported in table 3.

Table 3: Successive convexification algorithm parameters

Parameter	λ_v	λ_f	$\lambda_{\Delta t}$	ϵ_J
Value	10	10	0.01	1e−4

4.3 Final Solution

The three successive convexification steps were performed on the same mesh. Once convergence was attained, a mesh refinement process has been carried out in order to reduce the discretization error below 1e−6. After setting up a

DOI: 10.13009/EUCASS2019-430

new grid, the successive convexification and mesh refinement steps have been repeated until the converged solution discretization error did not exceed tolerances in any interval. The initial and final grid sizes are reported in table 4. Since the original grid was already quite dense, new nodes were added only to some phases and in a limited number of intervals.

The whole solution process required approximately 5 minutes. However, most of the time is spent on the computation of the mesh discretization error. In particular, the computation of the integral (50) with tolerance close to machine precision for each grid interval is the most expensive task. Conversely, each successive convexification requires only 15–30 s to attain convergence. This time, as expected, is very brief, even though no particular code optimization was performed.

Table 4: Mesh nodes of each phase

Phase	Liftoff	Pitch-over	ZLGT	Coasting 1	Stage 2 (1)	Coasting 2	Stage 2 (2)
Original mesh	6	6	41	6	41	81	21
Final mesh	6	6	75	6	57	81	21

The final solution, after the mesh refinement, differs from the one reported in figure 3c and its altitude profile is plotted in figure 3d and compared to the DE solution. Differences concern all the optimization variables: states, controls and time-lengths. In table 5 the results for the final trajectory are reported and compared with those obtained with the DE algorithm.

Table 5: Final solution compared to the DE solution.

	Final Solution	**DE Solution**
Payload mass	26 791.12 kg	26 790.28 kg
Stage 2 (1) Δt	391.34 s	391.35 s
Coasting 2 Δt	2991.23 s	2945.76 s
Stage 2 (2) Δt	5.66 s	5.65 s

The thrust direction elevation angle is reported in figure 4 and compared to the DE results. The time laws appear quite regular: either linear or constant. The small irregularities that can be observed on the plots are due to numerical errors and do not affect the solution quality. These control laws represent a proof of the solution optimality. Compared to the results obtained by the DE algorithm, that parameterizes the control laws as linear, the first upper stage firing control law is practically the same. Instead, the second firing elevation, despite showing a similar trend, is quite different. This is due to the fact that the DE solution is characterized by a shorter coasting phase, as reported in table 5. Hence the second burn of the last stage is performed at different times (shifted to the same time interval in figure 4 only for comparison) and with a different rocket orientation.

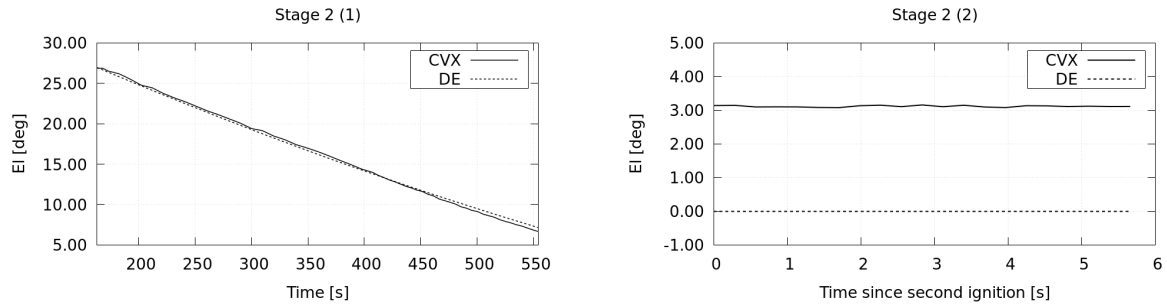


Figure 4: Control laws. The solid lines represent the optimal thrust direction elevation according to the convex algorithm, while the dashed ones correspond to the differential evolution algorithm results.

Finally, it has been verified that the relaxation of the control constraint (31) into (32) is exact. Indeed, the equality constraint (31) is satisfied at each discretization node with a tolerance of 1e−6. Hence, the final solution is characterized by controls (u_r, u_t) that are placed on the cone surface rather than inside the cone.

DOI: 10.13009/EUCASS2019-430

5. Conclusions

This paper presented a convex methodology to solve the optimal control problem of a multi-stage rocket ascent. Since the problem is not promptly suitable for a convex programming approach, the detailed convex transcription process has been described. By using lossless and successive convexification techniques the real, nonlinear, multi-phase dynamics of the problem is maintained. An automatic procedure that gradually transforms the problem from a simplified time-fixed, without atmosphere one into the original one allows to use a very basic initial guess, thus overcoming the optimization sensitivity with respect to the initialization.

One of the principal merits of the proposed method is the convergence towards the optimal solution in a limited, short, time. Indeed, the comparison with the solution obtained with a DE algorithm proved the optimality of the final trajectory. The major drawback is represented by the need for analytical derivation of the convex formulation, that requires some additional effort by the user. Nonetheless, the proposed algorithm implementation, even though not optimized, allows to obtain an optimal solution in a very short time. This opens a series of opportunities for time-critical applications, such as real-time rocket guidance, or robust optimization studies.

Future work will include three-dimensional motion and additional boundary and path constraints in order to study a more realistic scenario. Furthermore, the development of a more robust successive convexification algorithm may reduce the automatic initialization procedure to a single step, fastening even more the solution process.

References

[1] Behçet Açıkmeşe and Lars Blackmore. Lossless convexification of a class of optimal control problems with non-convex control constraints. *Automatica*, 47(2):341–347, 2011.

[2] Farid Alizadeh and Donald Goldfarb. Second-order cone programming. *Mathematical programming*, 95(1):3–51, 2003.

[3] John T Betts and William P Huffman. Mesh refinement in direct transcription methods for optimal control. *Optimal Control Applications and Methods*, 19(1):1–21, 1998.

[4] GL Brauer, DEj Cornick, and R_ Stevenson. Capabilities and applications of the program to optimize simulated trajectories (post). program summary document. 1977.

[5] Lorenzo Casalino and Dario Pastrone. Optimization of hybrid propellant mars ascent vehicle. In *50th AIAA/ASME/SAE/ASEE Joint Propulsion Conference*, page 3953, 2014.

[6] Guido Colasurdo, Dario Pastrone, and Lorenzo Casalino. Optimization of rocket ascent trajectories using an indirect procedure. In *Guidance, Navigation, and Control Conference*, page 3323, 1995.

[7] Lorenzo Federici, Alessandro Zavoli, and Guido Colasurdo. Preliminary capture trajectory design for europa tomography probe. *International Journal of Aerospace Engineering*, 2018, 2018.

[8] Lorenzo Federici, Alessandro Zavoli, Guido Colasurdo, Lucandrea Mancini, and Agostino Neri. Integrated optimization of ascent trajectory and srm design of multistage launch vehicles. In *29th AAS/AIAA Space Flight Mechanics Meeting*, Ka'anapali, Maui, HI, January 2019.

[9] LLC Gurobi Optimization. Gurobi optimizer reference manual, 2019.

[10] Charles R Hargraves and Stephen W Paris. Direct trajectory optimization using nonlinear programming and collocation. *Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[11] Matthew P Kelly. Transcription methods for trajectory optimization. *Tutorial, Cornell University, Ithaca, New York*, 2015.

[12] Xinfu Liu. Fuel-optimal rocket landing with aerodynamic controls. *Journal of Guidance, Control, and Dynamics*, 42(1):65–77, 2018.

[13] Xinfu Liu, Ping Lu, and Binfeng Pan. Survey of convex optimization for aerospace applications. *Astrodynamics*, 1(1):23–40, 2017.

[14] Xinfu Liu, Zuojun Shen, and Ping Lu. Entry trajectory optimization by second-order cone programming. *Journal of Guidance, Control, and Dynamics*, 39(2):227–241, 2015.

DOI: 10.13009/EUCASS2019-430

[15] Yuanqi Mao, Michael Szmuk, and Behcet Acikmese. Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. *arXiv preprint arXiv:1804.06539*, 2018.

[16] Michael Szmuk and Behcet Acikmese. Successive convexification for 6-dof mars rocket powered landing with free-final-time. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 0617, 2018.

[17] Andreas Wiegand et al. Astos user manual. *Unterkirnach, Germany: Astos Solutions GmbH*, 17, 2010.

[18] CH Williams and OF Spurlock. Duksup: A computer program for high thrust launch vehicle trajectory design and optimization. 2014.

[19] Runqiu Yang and Xinfu Liu. Comparison of convex optimization-based approaches to solve nonconvex optimal control problems. In *AIAA Scitech 2019 Forum*, page 1666, 2019.