ESA GNC & ICATT Conference 2026
28 Sep. – 2 Oct. 2026, Sevilla, Spain
https://atpi.eventsair.com/gncicatt/

A Comparison of First-Order Methodologies for Powered Descent Guidance Applications

Filippo Maria Cataldo[1,*] **and Marco Sagliano**[1]

[1]*Alma Mater Studiorum - Università di Bologna, Via Luciano Montaspro, 97, Forlì, 47121, Italy*
[*]*Corresponding author: filippomaria.cataldo@unibo.it*

ABSTRACT

Methods for Powered Descent Guidance (PDG) are receiving a lot of interest because of their importance in the context of rocket reusability and mission safety. Since the descent phase's duration is in the order of tens of seconds, one of the major limitations of many methods is their high runtime that might inhibit real-time applications. Standard solvers require the computation of second-order information of the cost function and constraints. While this often ensures accurate results, this step represents the major process computational bottleneck. Conversely, first-order algorithms work by definition only with first-order information, thus increasing numerical efficiency at the cost of limited accuracy. Two of the most prominent examples are the Alternating Direction Method of Multipliers and the Proportional-Integral Projected Gradient method. In this work we employ them to solve a 3-DoF and 6-DoF PDG problem: we perform a qualitative and quantitative comparison to understand the benefits and drawbacks of each methodology, aiming at verifying and quantifying the competitiveness of these algorithms compared with more standard approaches. We show that the proposed methods can converge to the same solutions obtained with a state-of-the-art solver and have the potential to become a future efficient alternative to standard approaches.

1 Introduction

Powered Descent Guidance (PDG) has become a critical field of study among the engineering community over the last decade because of its importance in achieving a successful mission, with a vehicle that is effectively reusable, thus dramatically lowering the space access cost threshold. This is one of the key reasons explaining why, in fact, there are nowadays many companies working on pinpoint landing technologies.

Typically, we formulate the PDG problem as an Optimal Control Problem (OCP) aiming at minimizing (or maximizing) an objective function. To numerically solve the problem, we first *transcribe* it, that is, we transform it into a parameter optimization problem which can be solved with dedicated state-of-the-art algorithms. If the problem's equality constraints are linear, while the inequalities outline a convex feasibility region, the problem is defined as convex and enjoys many convergence properties and can be solved very efficiently with specific convex solvers [1]. In the context of rocket landing, Açıkmeşe et Ploen marked a turning point with the formulation of a lossless (with respect to optimality) convexification of the problem [2]. While we can apply a similar transformation in many cases, this is not always possible: in these cases an effective alternative technique consists in linearizing the problem around a reference solution to make it convex, and solving it with the corresponding convex solver. We then feed back the obtained sub-solution into the linearization process and we repeat the steps until convergence. This approach is called Sequential Convex Programming (SCP) [3].

For complex problems, even with the aforementioned methods, obtaining a solution might require high runtime and, since the duration of the landing phase is in the order of tens of seconds, real-time requirements are often difficult to achieve. State-of-the-art solvers usually rely on second-order information of the problem: in fact they often employ Newton's method to satisfy the KKT conditions, and thus involve the computation and inversion

© *2026 Filippo Maria Cataldo and Marco Sagliano*

of the Hessian matrix associated with the cost function and constraints. This ensures accurate results, but the per-iteration computational burden is the main reason for the slowness of such methods. Recently, first-order algorithms are gaining interest because of their low required computational effort since, as the name suggests, they only rely on first-order information. This advantage comes at the cost of lower accuracy or slow convergence, but in many applications, it could be more important to obtain a lower-accuracy solution quickly rather than a highly accurate one in a longer time. Chronologically first examples of algorithms based on first-order information date back to the 1960s, with methods such as and Dual Ascent and Dual Decomposition [4], [5], [6], which update dual variables in the direction of the gradient of the dual function and, under some (rather strict) assumptions, converge to the optimal solution of a convex problem. To overcome the limitations related to assumptions such as strict convexity of the objective function, the Augmented Lagrangian and the Method of Multipliers were introduced [7], which consist in adding a penalty term to the Lagrangian function of the problem and applying Dual Ascent to the modified problem. Finally, these formulations evolved into the well-consolidated Alternating Direction Method of Multipliers (ADMM), that combines the convergence properties of the Augmented Lagrangians with the distributed framework of Dual Decomposition [8], [9]. While this method was originally proposed in the 1970s, only recently the scientific community started to actively exploiting it, because of its efficiency in distributed problems. ADMM splits the primal decision variables into two parts that are alternately updated. With a smart choice of variable splitting, the overall formulation becomes highly parallelizable. In more recent times Yu et Al. introduced the Proportional Integral Projected Gradient (PIPG) method [10], an algorithm that updates the dual variables with a proportional-integral feedback of constraint violation and that is also amenable to parallelization. Both methods can carry out an efficient optimization of convex problems, with well-defined convergence behavior for both the primal-dual gap and the feasibility error; furthermore, they have been both successfully applied to solve 3-DoF and 6-DoF PDG problems [11], [12], [13]. However, the use of first-order methods in PDG contexts still needs further exploration and study, with proper comparisons between them and other state-of-the-art solvers.

In this work we take a step forward in this direction, comparing ADMM and PIPG in both a 3-DoF and 6-DoF PDG problem, with the aim of assessing more in detail their performance. We address their advantages and limitations with respect to traditional methods and we identify some margins for future improvements. The paper is organized as follows: in Sec. 2 we describe the two methods, in Sec. 3 we formulate the two PDG problems, in Sec. 4 we present the numerical results, in Sec. 5 we draw the final conclusions.

1.1 Notation

We define some of the mathematical notation that we will use in the remainder of this paper. We indicate vector components with a numerical subscript: for example a_1 is the first component of vector a. We also employ subscripts to indicate the value of a variable at a specified time node. Superscripts, instead, usually keep track of algorithm's iterations. Later we will introduce the concepts of global variables and reference variables: the former will be denoted with a hat (e.g. $\hat{a}$) and the latter with a bar (e.g. $\bar{a}$). Local variables, instead, will have also a superscript indicating the actual time node that they are physically related to. In general, when quantities require multiple subscripts or superscripts, we separate them with a comma, with their meaning inferable from the context. Lastly, to make notation lighter, we will usually not highlight dependence on time as it is also inferable from the context, while we denote a derivative with respect to physical time with a dot (e.g. $\dot{a}$).

2 First-Order Algorithms

In this section we quickly describe the ADMM and PIPG algorithms and their convergence properties.

2.1 Alternating Direction Method of Multipliers

ADMM is an operator splitting method that, as the name suggests, splits the primal variables into two sets: $x \in \mathbb{R}^{n_x}$ and $z \in \mathbb{R}^{n_z}$. It formulates and solves the following problem:

$$\begin{aligned} \min_{x,z} \quad & \zeta_x(x) + \zeta_z(z) \\ \text{s.t.} \quad & H_x x + H_z z = h \end{aligned} \tag{1}$$

© *2026 Filippo Maria Cataldo and Marco Sagliano*

where $H_x \in \mathbb{R}^{n_h \times n_x}$, $H_z \in \mathbb{R}^{n_h \times n_z}$, $h \in \mathbb{R}^{n_h}$, and $\zeta_x : \mathbb{R}^{n_x} \to \mathbb{R}$ and $\zeta_z : \mathbb{R}^{n_z} \to \mathbb{R}$ are the cost functions, which require no particular properties, apart from convexity. In fact, as in our problem formulations, we can express them as the following non-differentiable indicator function $I_{\mathbb{A}}(a) : \mathbb{R}^{n_a} \to \{0, \infty\}$:

$$I_{\mathbb{A}}(a) = \begin{cases} 0 & \text{if} \quad a \in \mathbb{A} \\ \infty & \text{otherwise} \end{cases} \tag{2}$$

with generic set $\mathbb{A}$ and generic variable $a \in \mathbb{R}^{n_a}$. By including it in the cost function we can effectively impose set constraints [1]. We present the ADMM mehthodology in Algorithm 1, where, besides the primal variables x and z, we introduce the dual variable $y \in \mathbb{R}^{n_y}$; moreover the penalty parameter ρ weighs the equality constraints violation in the augmented Lagrangian of the problem.

Algorithm 1 ADMM algorithm.

1: Initialize counter $k = 0$.
2: Set initial value z^0 and y^0.
3: **while** $r^{k+1}_{primal} > \varepsilon^{k+1}_{primal} \vee r^{k+1}_{dual} > \varepsilon^{k+1}_{dual}$ **do**
4: $\quad x^{k+1} = \underset{x}{\text{argmin}}(\zeta_x(x) + \rho/2||H_x x + H_z z^k - h + y^k||^2)$
5: $\quad z^{k+1} = \underset{z}{\text{argmin}}(\zeta_z(z) + \rho/2||H_x x^{k+1} + H_z z - h + y^k||^2)$
6: $\quad y^{k+1} = y^k + H_x x^{k+1} + H_z z^{k+1} - h$
7: $\quad$ Update feasibility error $r^{k+1}_{primal}, r^{k+1}_{dual}$ and tolerances $\varepsilon^{k+1}_{primal}, \varepsilon^{k+1}_{dual}$
8: $\quad k \leftarrow k + 1$
9: **end while**

Note that, as shown in [9], we can express the primal and dual feasibility errors as

$$\begin{aligned} r^{k+1}_{primal} &= ||H_x x^{k+1} + H_z z^{k+1} - h|| \\ r^{k+1}_{dual} &= ||\rho H_x^T H_z (z^{k+1} - z^k)|| \end{aligned} \tag{3}$$

Furthermore, we define the tolerance as the composition of both absolute and relative tolerances:

$$\begin{aligned} \varepsilon^k_{primal} &= \sqrt{n_x}\varepsilon_{abs} + \varepsilon_{rel} \max\{||H_x x^k||, ||H_z z^k||, ||h||\} \\ \varepsilon^k_{dual} &= \sqrt{n_y}\varepsilon_{abs} + \varepsilon_{rel}||H_x^T y^k|| \end{aligned} \tag{4}$$

While the variable splitting that can be performed is general, we can leverage some techniques for efficient optimization. See Sec. 3 for our implementation.

2.2 Proportional Integral Projected Gradient Method

Here we describe and employ the PIPG variant with extrapolation [14] which solves conic optimization problems of the form

$$\begin{aligned} \min_x \quad & \frac{1}{2} x^T F x + f^T x \\ \text{s.t.} \quad & Hx - h \in \mathbb{K}, \quad x \in \mathbb{H} \end{aligned} \tag{5}$$

where $\mathbb{K}$ is a closed convex cone and $\mathbb{H}$ is a closed convex set, while $F \in \mathbb{R}^{n_x \times n_x}$ is a symmetric positive semidefinite matrix; moreover $f \in \mathbb{R}^{n_x}$, $H \in \mathbb{R}^{n_h \times n_x}$, $h \in \mathbb{R}^{n_h}$. We detail the method's steps in Algorithm 2 where y is the dual variable, $\rho \in (0, 2]$ is the interpolation/extrapolation constant and $\mathbb{K}^\circ$ is the polar cone of $\mathbb{K}$. We denote with $\pi_{\mathbb{A}}$ the Euclidean projection onto set $\mathbb{A}$. Apart from projection onto sets, the dual update is a proportional-integral term of $Hx^k - h$, while the primal update, if $H = 0$, resembles a projected gradient method that minimizes the cost function over $\mathbb{H}$, hence the name of the algorithm.

Moreover, α, β are the step sizes that, for convergence, must satisfy the following condition:

© *2026 Filippo Maria Cataldo and Marco Sagliano*

$$0 < \alpha < \frac{2}{\sqrt{||F||^2 + 4\chi||H||^2} + ||F||}, \quad \beta = \chi\alpha \tag{6}$$

with $\chi > 0$ and where $||\cdot||$, when applied to a matrix, denotes its largest singular value. Similarly as for ADMM, we can define primal and dual feasibility residuals as

$$\begin{aligned} r^{k+1}_{primal} &= ||1/\beta(y^{k+1/2} - y^k) - H(x^{k+1/2} - x^k)|| \\ r^{k+1}_{dual} &= ||1/\alpha(x^{k+1/2} - x^k) - F(x^{k+1/2} - x^k) - H^T(y^{k+1/2} - y^k)|| \end{aligned} \tag{7}$$

and tolerances as

$$\begin{aligned} \varepsilon^k_{primal} &= \sqrt{n_x}\varepsilon_{abs} + \varepsilon_{rel}||Hx^k|| \\ \varepsilon^k_{dual} &= \sqrt{n_y}\varepsilon_{abs} + \varepsilon_{rel}||H^T y^k|| \end{aligned} \tag{8}$$

Algorithm 2 PIPG algorithm.

Compute step sizes α, β.
Initialize counter $k = 0$.
Set initial value x^0 and y^0.
while $r^{k+1}_{primal} > \varepsilon^{k+1}_{primal} \vee r^{k+1}_{dual} > \varepsilon^{k+1}_{dual}$ **do**
  $x^{k+1/2} = \pi_{\mathbb{H}}[x^k - \alpha(Fx^k + f + H^T y^k)]$
  $y^{k+1/2} = \pi_{\mathbb{K}^\circ}[y^k + \beta(H(2x^{k+1/2} - x^k) - h)]$
  $x^{k+1} = (1-\rho)x^k + \rho x^{k+1/2}$
  $y^{k+1} = (1-\rho)y^k + \rho y^{k+1/2}$
  Update feasibility error $r^{k+1}_{primal}, r^{k+1}_{dual}$ and tolerances $\varepsilon^{k+1}_{primal}, \varepsilon^{k+1}_{dual}$
  $k \leftarrow k + 1$
end while

3 Powered Descent Guidance

In this section we present the two OCPs that we will solve using ADMM and PIPG. The first one is a 3-DoF convex problem, while the second one is a 6-DoF non-convex problem. In Fig. 2 we provide a scheme of the rocket and the two reference frames involved in both problems.

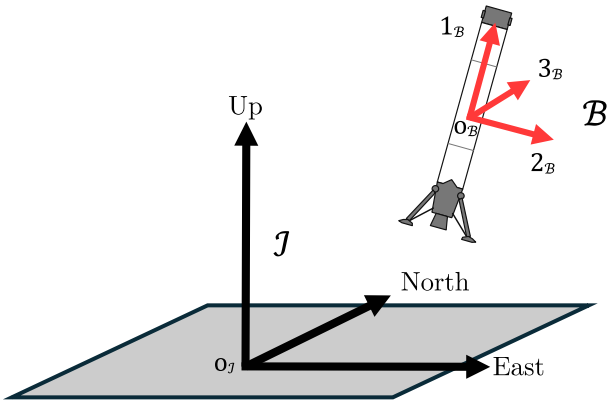

Figure 1 *Scheme of the PDG problem with the rocket, the inertial frame $\mathcal{J}$ centered in $o_{\mathcal{J}}$ and the body frame $\mathcal{B}$ centered in $o_{\mathcal{B}}$.*

© *2026 Filippo Maria Cataldo and Marco Sagliano*

3.1 Three-Degrees-of-Freedom Descent

The OCP that we present here is a 3-DoF rocket landing problem, reformulated with lossless convexification [2]. We define a target-centered inertial reference frame $\mathcal{I}$ with basis directions following the Up-East-North (UEN) sequence. We denote the position and velocity of the rocket's center of mass, in the inertial frame, with $r \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$ respectively, while its mass with $m \in \mathbb{R}$. We consider the control as the three inertial components of the acceleration vector $a \in \mathbb{R}^3$ and we neglect the rocket's translational kinematics and dynamics. The time derivative of the state is as follows:

$$\dot{r} = v \tag{9}$$
$$\dot{v} = g + a \tag{10}$$
$$\dot{m} = -\frac{||a||m}{v_e} \tag{11}$$

where g is the constant gravity acceleration in inertial frame and v_e is the constant engine's exhaust velocity. The rocket's must reach the desired target at final time, while satisfying the following path constraints:

$$\frac{T_{min}}{m} \leq ||a|| \leq \frac{T_{max}}{m} \tag{12}$$
$$||r_H|| \leq r_1 \tan\gamma \tag{13}$$
$$||v|| \leq V_{max} \tag{14}$$
$$||a|| \cos\theta_{max} \leq a_1 \tag{15}$$
$$m(t_f) \geq m_{dry} \tag{16}$$

where T_{min} and T_{max} are the lower and upper bounds on thrust magnitude, $\gamma \in [0°, 90°)$ is the glideslope angle, $r_H \in \mathbb{R}^2$ groups the horizontal position coordinates, V_{max} is the maximum allowable velocity, $\theta_{max} \in [0°, 90°]$ is the maximum thrust pointing angle, t_f is the final time, m_{dry} is the dry mass of the vehicle. All the dynamics and path constraints, except for Eqs. (11) and (12), are convex, when parameters are in the defined ranges. We now employ some reformulations to convexify non-convex constraints. First we introduce the slack control $\sigma \in \mathbb{R}$ that represents the acceleration norm, and we impose the following, convex, inequality constraint:

$$||a|| \leq \sigma \tag{17}$$

It is proven that, when a minimum-fuel OCP is solved, the optimal solution satisfies the previous inequality as equality, thus preserving the physical meaning of the variables [2]. Furthermore, we replace mass with its logarithm $m_{log} = \log m$, such that Eq. (11) and Eq. (12) become

$$\dot{m}_{log} = -\frac{\sigma}{v_e} \tag{18}$$
$$T_{min} e^{-m_{log}} \leq \sigma \leq T_{max} e^{-m_{log}} \tag{19}$$

Lastly, we linearize the bounds in Eq. (19) around a reference mass trajectory, linearly varying with time $t \in \mathbb{R}$, which we define as follows:

$$\bar{m}_{log} = \log\left(m_{wet} - \frac{T_{max}}{v_e} t\right) \tag{20}$$

where m_{wet} is the wet mass of the vehicle. The linearized inequalities then become

$$T_{min} e^{-\bar{m}_{log}} [1 - (m_{log} - \bar{m}_{log})] \leq \sigma \leq T_{max} e^{-\bar{m}_{log}} [1 - (m_{log} - \bar{m}_{log})] \tag{21}$$

with negligible error with respect to the original nonlinear formulation [2]. The thrust pointing angle constraint instead can be expressed as follows.

$$\sigma \cos\theta_{max} \leq a_1 \tag{22}$$

After grouping the state in $s = [r, v, m_{log}]^T$ and the control in $u = [a, \sigma]^T$, we can state the OCP, which is a minimum-fuel problem with fixed initial and final time, $t_0 = 0$ and t_f.

© *2026 Filippo Maria Cataldo and Marco Sagliano*

3-DoF Optimal Control Problem

$$\min_{s,u} \quad \int_{t_0}^{t_f} \sigma dt$$

$$\text{s.t.} \quad \begin{cases} s(t_0) = \begin{bmatrix} r_{t_0} & v_{t_0} & m_{wet} \end{bmatrix}^T \\ \begin{bmatrix} r(t_f) & v(t_f) \end{bmatrix}^T = \begin{bmatrix} r_{t_f} & v_{t_f} \end{bmatrix}^T \\ \text{Dynamics in Eqs. (9, 10, 18)} \\ \text{Path constraints in Eqs. (13, 14, 16, 17, 21, 22)} \end{cases} \tag{23}$$

Note that we define initial and final position with r_{t_0} and r_{t_f}, and initial and final velocity with v_{t_0} and v_{t_f}. Before passing the problem to the solvers, we must first transcribe it into a parameter optimization problem. First, we divide the time domain into $N+1$ uniformly-spaced time nodes, such that we can define the value of the state and control at each of them $\{s_i, u_i\}\ \forall i \in \{0, \dots, N\}$. Second, we propagate the states between time nodes by analytically integrating the dynamics equations and using Zero-Order-Hold (ZOH) for the control (i.e. control input is constant between time nodes). The following discrete system is obtained:

$$s_{i+1} = As_i + Bu_i + c \quad \forall i \in \{0, \dots, N-1\} \tag{24}$$

3-DoF ADMM problem formulation For the ADMM solver, we employ the so-called Consensus Form [9], which allows us to decouple state variables and enable parallelizability of the algorithm. In particular, we make a distinction between *local* and *global* variables: at the i-th time node we define the local state s_i^i at the time node, its single-step propagation s_i^{i+1}, and the global state at the same time node $\hat{s}_i$. Local and global states must reach a *consensus*, expressed as the following set of equality constraints:

$$\begin{aligned} s_i^i &= \hat{s}_i \quad \forall i \in \{0, \dots, N-1\} \\ s_i^{i+1} &= \hat{s}_{i+1} \quad \forall i \in \{0, \dots, N-1\} \end{aligned} \tag{25}$$

In this way, we can impose the dynamics constraints only on the local variables, and in such a way that each subset of local variables (at each time node) are independent one from the other. Moreover, we define some auxiliary variables and constraints that will be useful to ensure satisfaction of inequality constraints. In particular, we create copies of the local position and velocity ($w_r \in \mathbb{R}^3$ and $w_v \in \mathbb{R}^3$ respectively), of the control ($w_u \in \mathbb{R}^3$), and an additional variable ($l \in \mathbb{R}^3$) associated with thrust bounds and thrust pointing angle constraints:

$$w_{r,i} = r_i^{i+1} \quad \forall i \in \{0, \dots, N-2\} \tag{26}$$

$$w_{v,i} = v_i^{i+1} \quad \forall i \in \{0, \dots, N-2\} \tag{27}$$

$$w_{u,i} = u_i \quad \forall i \in \{0, \dots, N\} \tag{28}$$

$$l_i = P_{m,i} m_{log,i}^i + P_u u_i + p_i \quad \forall i \in \{0, \dots, N\} \tag{29}$$

where

$$P_{m,i} = \begin{bmatrix} 0 \\ T_{min} e^{-\bar{m}_{log,i}} \\ -T_{max} e^{-\bar{m}_{log,i}} \end{bmatrix}, \quad P_u = \begin{bmatrix} 1 & 0 & 0 & -\cos\theta_{max} \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \quad p_i = \begin{bmatrix} 0 \\ -T_{min} e^{-\bar{m}_{log,i}} (1 - \bar{m}_{log,i}) \\ T_{max} e^{-\bar{m}_{log,i}} (1 - \bar{m}_{log,i}) \end{bmatrix} \quad \forall i \in \{0, \dots, N\}$$

(30)

© *2026 Filippo Maria Cataldo and Marco Sagliano*

Note that we only create $N-1$ copies of the position and velocity because the associated constraints at boundary points are already satisfied through boundary conditions; we implicitly exploit such considerations throughout the remainder of the paper. Lastly, we introduce the following sets:

$$\mathbb{W}_r = \{r \mid ||r_H|| \leq r_1 \tan\gamma\} \tag{31}$$
$$\mathbb{W}_v = \{v \mid ||v|| \leq V_{max}\} \tag{32}$$
$$\mathbb{W}_u = \{u \mid ||a|| \leq \sigma\} \tag{33}$$
$$\mathbb{L} = \{l \mid l \geq 0\} \tag{34}$$
$$\mathbb{M} = \{m_{log} \mid m_{log} \geq \log m_{dry}\} \tag{35}$$
$$\mathbb{D} = \{(s, s^+, u) \mid s^+ = As + Bu + c\} \tag{36}$$

Now we have all the definition to state in Eq. (37) the optimization problem that we want to solve with ADMM.

$$\min_{x,z} \quad I_{\mathbb{M}}(\hat{m}_{log,N}) + \sum_{i=0}^{N} \sigma_i + I_{\mathbb{W}_u}(w_{u,i}) + I_{\mathbb{L}}(l_i) + \sum_{i=0}^{N-1} I_{\mathbb{D}}(s_i^i, s_i^{i+1}, u_i) + \sum_{i=0}^{N-2} I_{\mathbb{W}_r}(w_{r,i}) + I_{\mathbb{W}_v}(w_{v,i})$$

$$\text{s.t.} \quad \begin{cases} s_0^0 = \begin{bmatrix} r_{t_0} & v_{t_0} & m_{wet} \end{bmatrix}^T \\ \begin{bmatrix} r_{N-1}^N & v_{N-1}^N \end{bmatrix}^T = \begin{bmatrix} r_{t_f} & v_{t_f} \end{bmatrix}^T \\ \text{Consensus constraints in Eqs. (25) to (29)} \end{cases} \tag{37}$$

We performed the variable splitting such that x includes $\{s_i^i, s_i^{i+1}\}_{i=0}^{N-1}$ and $\{u_i\}_{i=0}^{N}$, and z includes the global states and all the other auxiliary variables. The presence of the indicator functions in the cost ensures, at optimality, the satisfaction of the inequality constraints of the original OCP. With the choices we made, the x-update step in Algorithm 1 is the solution to an equality-constrained minimization problem that can be found by solving the associated KKT linear system; the update of global states and the other auxiliary variables can be obtained in closed form, with the latter update being the projection of the augmented Lagrangian residual onto the sets to which auxiliary variables must belong to.

3-DoF PIPG problem formulation The PIPG problem formulation is more straightforward since there is no variable splitting involved. Therefore we can directly state the PIPG problem in Eq. (38), without distinguishing between local and global states, and borrowing some sets definitions from the ADMM formulation.

$$\min_{x} \quad \sum_{i=0}^{N} \sigma_i$$

$$\text{s.t.} \quad \begin{cases} s_0 = \begin{bmatrix} r_{t_0} & v_{t_0} & m_{wet} \end{bmatrix}^T \\ \begin{bmatrix} r_N & v_N \end{bmatrix}^T = \begin{bmatrix} r_{t_f} & v_{t_f} \end{bmatrix}^T \\ \text{Dynamics constraints in Eq. (24)} \\ l_i = P_{m,i} m_{log,i} + P_u u_i + p_i \quad \forall i \in \{0, \dots, N\} \\ m_{log,N} \in \mathbb{M} \\ r_i \in \mathbb{W}_r, \quad v_i \in \mathbb{W}_v \quad \forall i \in \{1, \dots, N-1\} \\ u_i \in \mathbb{W}_u, \quad l_i \in \mathbb{L} \quad \forall i \in \{0, \dots, N\} \end{cases} \tag{38}$$

© 2026 Filippo Maria Cataldo and Marco Sagliano

Note that, with respect to the general PIPG problem in Eq. (5), we chose, as $\mathbb{K}$, $\mathbb{R}^n$ (which has corresponding dual cone $\mathbb{K}^\circ = 0^n$) and, as $\mathbb{H}$, all the other sets listed in the constraints in Eq. (38). However, other choices can be made, with possible differences in algorithm efficiency. It must also be noted that this type of formulation already has a certain degree of parallelizability, as shown in [15].

3.2 Six-Degrees-of-Freedom Descent

We now present a non-convex 6-DoF landing problem and thus, differently from the previous case, we now introduce also the rocket's rotational dynamics. We borrowed the main formulation from [16], with the introduction of some modifications in the setup of the SCP framework. As in the 3-DoF case, we have a target-centered inertial reference frame $\mathcal{I}$, but now we also define a body-fixed reference frame $\mathcal{B}$, with origin in the rocket's center of mass (considered fixed and rigidly moving with the rocket's body), with first basis direction $1_\mathcal{B}$ following the rocket's longitudinal axis, towards the tip, and with the other two directions $2_\mathcal{B}$ and $3_\mathcal{B}$ belonging to the plane perpendicular to $1_\mathcal{B}$ and completing the right-handed system. Note that we consider the system symmetric around the rocket's longitudinal axis, so $2_\mathcal{B}$ and $3_\mathcal{B}$ do not need further definition. Variables r and v denote the position and velocity of the rocket's center of mass in the inertial frame, while m denotes rocket's mass. Moreover, we introduce the unit quaternions $q \in \mathbb{R}^4$ that indicate the attitude of frame $\mathcal{I}$ with respect to frame $\mathcal{B}$, with scalar-first, and the angular velocity $\omega \in \mathbb{R}^3$ of frame $\mathcal{I}$ with respect to frame $\mathcal{B}$, expressed in body frame. Lastly, we denote with $T \in \mathbb{R}^3$ the thrust vector in frame $\mathcal{B}$. We can now define the 6-DoF equations of motion:

$$\dot{r} = v \tag{39}$$

$$\dot{v} = g + C_{\mathcal{I}\leftarrow\mathcal{B}}(q)\frac{T}{m} \tag{40}$$

$$\dot{m} = -\frac{||T||}{v_e} \tag{41}$$

$$\dot{q} = \frac{1}{2}\Omega(\omega)q \tag{42}$$

$$\dot{\omega} = J^{-1}\left([d\times]T - [\omega\times]J\omega\right) \tag{43}$$

where $C_{\mathcal{I}\leftarrow\mathcal{B}}(q)$ is the direction cosine matrix that transforms vectors in body frame into vectors in inertial frame, $\Omega(\omega)$ is the skew-symmetric matrix associated with quaternions' time derivative, and $[a\times]$ is the skew-symmetric matrix used to compute the cross product associated with the generic three-dimensional vector a. See the following equations for their expressions:

$$\begin{aligned} C_{\mathcal{I}\leftarrow\mathcal{B}}(q) &= \begin{bmatrix} 1-2(q_3^2+q_4^2) & 2(q_2q_3-q_1q_4) & 2(q_2q_4+q_1q_3) \\ 2(q_2q_3+q_1q_4) & 1-2(q_2^2+q_4^2) & 2(q_3q_4-q_1q_2) \\ 2(q_2q_4-q_1q_3) & 2(q_3q_4+q_1q_2) & 1-2(q_2^2+q_3^2) \end{bmatrix}, \\ \Omega(\omega) &= \begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix}, \quad [a\times] = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \end{aligned} \tag{44}$$

Moreover, d is the lever arm vector from the center of mass to the engine's gimbal point, expressed in $\mathcal{B}$, and J is the rocket's inertia tensor in body frame (here assumed constant); all the other symbols have the same definition as in Sec. 3.1. We can now define the state $s = [r, v, m, q, \omega]^T$ and the control $u = T$. Therefore, the state derivative can be expressed in a more compact formula:

$$\dot{s} = \psi(s, u) \tag{45}$$

However, since we are addressing a free-final time problem, we want to explicit its presence in the equations of motion. In order to achieve this, we define a normalized time $\tau \in [0, 1]$ such that its dependence on the original time t is linear. It is straightforward to prove that, in this case, $dt/d\tau = t_f$. We can now rewrite the equations of motion in the new time domain:

© *2026 Filippo Maria Cataldo and Marco Sagliano*

$$\frac{d}{d\tau}s = \frac{dt}{d\tau}\dot{s} = t_f\psi(s,u) \tag{46}$$

In addition to the constraints in Eqs. (16) and (13), we impose the following path constraints:

$$||\omega|| \leq \omega_{max} \tag{47}$$
$$\cos\phi_{max} \leq 1 - 2(q_3^2 + q_4^2) \tag{48}$$
$$||T||\cos\delta_{max} \leq T_1 \tag{49}$$
$$T_{min} \leq ||T|| \leq T_{max} \tag{50}$$

where ω_{max} is the maximum angular velocity, $\phi_{max} \in [0°, 180°]$ is the maximum tilt angle between $1_{\mathcal{B}}$ and the Up direction, $\delta_{max} \in [0°, 90°]$ is the maximum gimbal angle between the thrust vector and $-1_{\mathcal{B}}$. Furthermore, in Eq. (50) we bound the thrust magnitude, but with a different expression with respect to the 3-DoF case. Note that, for problem parameters inside the defined ranges, the non-convex constraints are the dynamics in Eq. (46) and the thrust lower bound in Eq. (50). In Eq. (51) we state the minimum-time 6-DoF non-convex OCP.

6-DoF Optimal Control Problem

$$\min_{s,u,t_f} \quad t_f$$

$$\text{s.t.} \quad \begin{cases} \begin{bmatrix} r(t_0) & v(t_0) & m(t_0) & \omega(t_0) \end{bmatrix}^T = \begin{bmatrix} r_{t_0} & v_{t_0} & m_{wet} & \omega_{t_0} \end{bmatrix}^T \\ \begin{bmatrix} r(t_f) & v(t_f) & q(t_f) & \omega(t_f) \end{bmatrix}^T = \begin{bmatrix} r_{t_f} & v_{t_f} & q_{t_f} & \omega_{t_f} \end{bmatrix}^T \\ \text{Dynamics in Eq. (46)} \\ \text{Path constraints in Eqs. (13, 16, 47, 48, 49, 50)} \end{cases} \tag{51}$$

In addition to initial and final position and velocity, we also define final quaternions q_{t_f}, and initial and final angular velocity ω_{t_0} and ω_{t_f}. To address the non-convexities, we employ sequential convexification [3]: first we linearize the non-convex constraints around a reference trajectory; then we discretize the linearized OCP and we modify it to address linearization approximations; lastly we solve the approximate convex problem. We then repeat the process until convergence, using as new reference trajectory the solution to the last convex problem.

We define the general reference state, control and final time as $\bar{s}$, $\bar{u}$ and $\bar{t}_f$. We approximate the nonlinear state dynamics with its first-order Taylor expansion:

$$\frac{d}{d\tau}s \cong \bar{t}_f \left.\frac{\partial}{\partial s}\psi(s,u)\right|_{\bar{s},\bar{u}} (s-\bar{s}) + \bar{t}_f \left.\frac{\partial}{\partial u}\psi(s,u)\right|_{\bar{s},\bar{u}} (u-\bar{u}) + \psi(\bar{s},\bar{u})t_f \tag{52}$$

Similarly, after approximating the thrust lower bound, we obtain the following inequalities:

$$\begin{aligned} \frac{\bar{T}}{||\bar{T}||}\cdot T &\geq T_{min} \\ ||T|| &\leq T_{max} \end{aligned} \tag{53}$$

We discretize the normalized time domain into $N+1$ uniformly spaced nodes, with time step $\Delta\tau$; moreover we define both the reference trajectory and the trajectory under optimization at each time node: $\{\bar{s}_i, \bar{u}_i, s_i, u_i\}\ \forall i \in \{0, \dots, N\}$. Using the trapezoidal scheme and ZOH control, we obtain the following discrete-time system:

$$A_i^+ s_{i+1} + A_i^- s_i + B_i u_i + D_i t_f + c_i = 0 \quad \forall i \in \{0, \dots, N-1\} \tag{54}$$

Lastly, since we are now dealing with a linearized problem that could present infeasibility even when the original problem is feasible (the so-called *artificial* infeasibility), we need to relax the problem to ensure its

© *2026 Filippo Maria Cataldo and Marco Sagliano*

feasibility. Traditional approaches include virtual control terms in the dynamics or virtual buffer terms in the other path constraints [17]. As an alternative approach, we decided to introduce the virtual state [13], defined as $\xi \in \mathbb{R}^{14}$. It carries the same physical meaning of the original state, but we constrain it differently: the dynamics is imposed only on the original state, while all the other path constraints and boundary conditions are imposed on the virtual state; to ensure that at optimality the two copies coincide, we penalize their difference in the cost function. Since in the proposed first-order algorithms we must often introduce auxiliary variables to perform projections onto different sets, with this methodology we have the advantage of handling both projections and artificial infeasibility with less auxiliary variables.

6-DoF ADMM problem formulation As in the 3-DoF case, we perform the splitting of variables in a Consensus fashion, using local states to decouple the dynamics between time nodes, and imposing the consensus constraints in Eq. (25) between local and global states. Moreover, we define auxiliary variables $w_r \in \mathbb{R}^3$, $w_m \in \mathbb{R}$, $w_q \in \mathbb{R}^2$, $w_\omega \in \mathbb{R}^3$, $w_u \in \mathbb{R}^3$ and $l \in \mathbb{R}^3$ associated with, respectively, glideslope, final mass, tilt angle, angular velocity, gimbal angle and thrust bounds constraints:

$$w_{r,i} = \xi_{r,i+1} \quad \forall i \in \{0, \dots, N-2\} \tag{55}$$
$$w_m = \xi_{m,N} \tag{56}$$
$$w_{q,i} = \begin{bmatrix} \xi_{q_3,i} & \xi_{q_4,i} \end{bmatrix}^T \quad \forall i \in \{0, \dots, N-1\} \tag{57}$$
$$w_{\omega,i} = \xi_{\omega,i+1} \quad \forall i \in \{0, \dots, N-2\} \tag{58}$$
$$w_{u,i} = u_i \quad \forall i \in \{0, \dots, N\} \tag{59}$$
$$l_i = u_i \quad \forall i \in \{0, \dots, N\} \tag{60}$$

where subscripts of ξ indicate also the physical variable they are related to. We define the following sets:

$$\mathbb{W}_r = \{r \mid ||r_H|| \leq r_1 \tan\gamma\} \tag{61}$$
$$\mathbb{W}_m = \{m \mid m \geq m_{dry}\} \tag{62}$$
$$\mathbb{W}_q = \{(q_3, q_4) \mid \cos\phi_{max} \leq 1 - 2(q_3^2 + q_4^2)\} \tag{63}$$
$$\mathbb{W}_\omega = \{\omega \mid ||\omega|| \leq \omega_{max}\} \tag{64}$$
$$\mathbb{W}_u = \{u \mid ||u|| \cos\delta_{max} \leq u_1\} \tag{65}$$
$$\mathbb{L} = \{l \mid \frac{\bar{T}}{||\bar{T}||} \cdot l \geq T_{min} \wedge ||l|| \leq T_{max}\} \tag{66}$$
$$\mathbb{D}_i = \{(s, s^+, u, t_f) \mid A_i^+ s^+ + A_i^- s + B_i u + D_i t_f + c_i = 0\} \quad \forall i \in \{0, \dots, N-1\} \tag{67}$$

We are now ready to state in Eq. (68) the convex subproblem that we want to solve with ADMM.

$$\begin{aligned} \min_{x,z} \quad & t_f + \nu_{t_f}|t_f - \bar{t}_f|^2 + \nu_{su}||s_0^0 - \bar{s}_0||^2 + \nu_{vs}||s_0^0 - \xi_0||^2 + I_{\mathbb{W}_m}(w_m) + \\ & + \textstyle\sum_{i=0}^{N} I_{\mathbb{W}_u}(w_{u,i}) + I_{\mathbb{L}}(l_i) + \nu_{su}||u_i - \bar{u}_i||^2 + \\ & + \textstyle\sum_{i=0}^{N-1} I_{\mathbb{W}_q}(w_{q,i}) + I_{\mathbb{D}_i}(s_i^i, s_i^{i+1}, u_i, t_f) + \nu_{su}||s_i^{i+1} - \bar{s}_{i+1}||^2 + \nu_{vs}||s_i^{i+1} - \xi_{i+1}||^2 + \\ & + \textstyle\sum_{i=0}^{N-2} I_{\mathbb{W}_r}(w_{r,i}) + I_{\mathbb{W}_\omega}(w_{\omega,i}) \end{aligned} \tag{68}$$
$$\text{s.t.} \quad \begin{cases} \begin{bmatrix} \xi_{r,0} & \xi_{v,0} & \xi_{m,0} & \xi_{\omega,0} \end{bmatrix}^T = \begin{bmatrix} r_{t_0} & v_{t_0} & m_{wet} & \omega_{t_0} \end{bmatrix}^T \\ \begin{bmatrix} \xi_{r,N} & \xi_{v,N} & \xi_{q,N} & \xi_{\omega,N} \end{bmatrix}^T = \begin{bmatrix} r_{t_f} & v_{t_f} & q_{t_f} & \omega_{t_f} \end{bmatrix}^T \\ \text{Consensus constraints in Eq. (25) and Eqs. (55) to (60)} \end{cases}$$

It can be noticed that in the cost function we penalize the squared norm of the difference between states and virtual states with weight ν_{vs}, as well as deviations of the state and control from the reference with weight ν_{su} and final time from its reference with weight ν_{tf}. We introduce the second penalization to avoid large deviations from

© *2026 Filippo Maria Cataldo and Marco Sagliano*

the reference that would undermine linearization accuracy. In this case we performed the variable splitting such that x includes $\{s_i^i, s_i^{i+1}\}_{i=0}^{N-1}, \{\xi_i, u_i\}_{i=0}^{N}$ and t_f, and z includes the global states and all the other auxiliary variables. Considerations similar to the ones in the 3-DoF case can be made on the x and z-updates in the ADMM algorithm.

6-DoF PIPG problem formulation In Eq. (69) we directly state the PIPG optimization problem.

$$
\begin{aligned}
\min_{x,z} \quad & t_f + \nu_{t_f}|t_f - \bar{t}_f|^2 + \\
& + \textstyle\sum_{i=0}^{N} \nu_{su}||s_i - \bar{s}_i||^2 + \nu_{su}||u_i - \bar{u}_i||^2 + \nu_{vs}||s_i - \xi_i||^2 \\
\text{s.t.} \quad & \begin{cases}
\begin{bmatrix} \xi_{r,0} & \xi_{v,0} & \xi_{m,0} & \xi_{\omega,0} \end{bmatrix}^T = \begin{bmatrix} r_{t_0} & v_{t_0} & m_{wet} & \omega_{t_0} \end{bmatrix}^T \\
\begin{bmatrix} \xi_{r,N} & \xi_{v,N} & \xi_{q,N} & \xi_{\omega,N} \end{bmatrix}^T = \begin{bmatrix} r_{t_f} & v_{t_f} & q_{t_f} & \omega_{t_f} \end{bmatrix}^T \\
\text{Dynamics constraints in Eq. (54)} \\
l_i = u_i, \quad l_i \in \mathbb{L} \quad , u_i \in \mathbb{W}_u \quad \forall i \in \{0, \dots, N\} \\
\xi_{m,N} \in \mathbb{W}_m \\
\xi_{r,i} \in \mathbb{W}_r, \quad \xi_{\omega,i} \in \mathbb{W}_\omega \quad \forall i \in \{1, \dots, N-1\} \\
\xi_{q,i} \in \mathbb{W}_q \quad \forall i \in \{0, \dots, N-1\}
\end{cases}
\end{aligned}
\tag{69}
$$

As in the 3-DoF case, with respect to the general PIPG problem in Eq. (5), we chose $\mathbb{R}^n$ as $\mathbb{H}$ and all the other sets listed in the constraints in Eq. (69) as $\mathbb{H}$.

SCP algorithm In Algorithm 3 we present the SCP procedure. We exit the loop only when the virtual states' error and the difference between new and old reference values (trust region) are below a certain tolerance.

Algorithm 3 6-DoF SCP algorithm.

1: Set initial reference $\{\bar{s}_i, \bar{u}_i\}_{i=0}^{N}$ and $\bar{t}_f$.
2: **while** $r_{su} > \varepsilon_{su} \vee r_{vs} > \varepsilon_{vs}$ **do**
3: Compute matrices of Eq. (54) by linearizing around the reference.
4: Solve Eq. (68) with ADMM or Eq. (69) with PIPG, and extract optimal solution $\{s_i^*, u_i^*\}_{i=0}^{N}$ and t_f^*
5: Update trust region error $r_{su} = \sqrt{\sum_{i=0}^{N} ||s_i^* - \bar{s}_i||^2 + ||u_i^* - \bar{u}_i||^2}$
6: Update virtual states error $r_{vs} = \sqrt{\sum_{i=0}^{N} ||s_i^* - \xi_i||^2}$
7: $\{\bar{s}_i, \bar{u}_i\}_{i=0}^{N} \leftarrow \{s_i^*, u_i^*\}_{i=0}^{N}$
8: $\bar{t}_f \leftarrow t_f^*$
9: **end while**

4 Numerical Simulations

In this section we show the results coming from the numerical solution of the problems presented in the previous section. For benchmarking purposes, we also solved the problems using MOSEK solver [18] with YALMIP parser [19], which allowed us to use problems formulations very similar to the PIPG ones. We carried out all the experiments on a machine with 3.20 GHz AMD Ryzen 7 7735HS CPU and 16 GB RAM, using MATLAB. First, we list all problems' parameters in Tables 1 and 2, noting that all physical parameters have been made dimensionless and that diag(a) indicates a zero matrix with the elements of a on its diagonal. In the 3-DoF case, we want the

rocket to reach the target at the origin of the reference frame with zero velocity, with $r_{t_0} = [0.56, -0.12, -1.5]^T$ and $v_{t_0} = [-0.12, -0.04, -0.04]^T$. Likewise, in the 6-DoF case, the final condition is the same, but $r_{t_0} = [4, 4, 0]^T$ and $v_{t_0} = [0, -4, 2]^T$; moreover, the rocket must land vertically and the angular velocity must be null both at initial and final time. Regarding the SCP algorithm, we used weights $\nu_{t_f} = 1$, $\nu_{su} = 10^{-3}$, $\nu_{vs} = 10^3$ and exit tolerances $\varepsilon_{su} = 10^{-1}$, $\varepsilon_{vs} = 10^{-1}$. We selected those values based on the convergence results of the benchmark solver: the ones associated with the trust region of the state, control and final time are similar to the ones in [16], while the virtual states' error carries a different physical meaning to the traditional virtual control or buffer ones, and we found no references to start from. To build the initial guess on the reference trajectory, we linearly interpolated between boundary conditions, when fully defined, and between wet and dry mass, while we chose a constant vertical attitude; we set the control opposite to the weight force resulting from the guess on the mass profile and we chose reference final time $\bar{t}_f = 4$.

Lastly, we discretized both problems using $N + 1 = 50$ time nodes. For both the proposed methods, we chose as exit tolerances $\varepsilon_{abs} = 10^{-6}$ and $\varepsilon_{rel} = 10^{-6}$. In ADMM we set $\rho = 10$, while in PIPG we used $\rho = 1.5$ and $\chi = 10$ for the 3-DoF problem, and $\rho = 1.9$ and $\chi = 10^4$ for the 6-DoF one, choosing α almost equal, with a relatively small margin, to its upper bound in Eq. (6). Note that, prior to solving the 6-DoF problem with our methods, we preconditioned the problem to improve its conditioning, given the high penalty weights in the cost function.

Parameter	Value	Units
g	$[-0.2, 0, 0]^T$	[-]
v_e	25	[-]
m_{wet}	1	[-]
m_{dry}	0.75	[-]
T_{min}	0.1	[-]
T_{max}	0.25	[-]
V_{max}	0.4	[-]
γ	80	[°]
θ_{max}	27	[°]
t_f	11	[-]

Table 1 *3-DoF problem parameters.*

Parameter	Value	Units
g	$[-1, 0, 0]^T$	[-]
v_e	294.201	[-]
m_{wet}	2	[-]
m_{dry}	1	[-]
J	diag([0.01, 0.01, 0.01])	[-]
d	$[-0.01, 0, 0]^T$	[-]
T_{min}	0.3	[-]
T_{max}	5	[-]
γ	70	[°]
ω_{max}	60	[°/-]
ϕ_{max}	90	[°]
δ_{max}	20	[°]

Table 2 *6-DoF problem parameters.*

The numerical results of the 3-DoF problem are presented in Fig. 2, where we compare the optimal trajectories computed with the proposed methods, to the benchmark one. For space purposes, we only show the position, velocity and mass against time, as well as the control profile. All the trajectories are superimposed, confirming the validity of the solution obtained through first-order methods; moreover, the thrust bang-bang profile is in accordance to the optimal control theory for minimum-fuel trajectories [2]. The same considerations can be made on the 6-DoF solution in Fig. 3, where all the trajectories are superimposed and reach the desired target at the same final time (note that we show here angular speed instead of translational speed). Also in this case, the thrust profile shows the bang-bang behavior typical of minimum-time trajectories [16].

For what concerns numerical efficiency, Table 3 shows the computational time required to solve both problems. In the 3-DoF case, the benchmark solver outperforms the proposed methods in terms of solve time. Nevertheless, three considerations must be made: first, MOSEK is a highly optimized, compiled solver, while our implementation of first-order algorithms is still rough and needs and efficient formulation in compiled language; second, it is known that first-order methods are slow at reaching a highly accurate solution, thus increasing the exit tolerance could result in a great improvement of computational time, while still preserving some accuracy; last, we didn't exploit any parallelization technique, even though our problem formulations are almost fully parallelizable (with some minor modifications), leaving a margin for further computational speed-up. The main objective of this work was, however, the assessment of the validity of the proposed methods to solve PDG problems, when compared with state-of-the-art solver. Apart from that, PIPG proved to be faster than ADMM, probably also owing to the

© *2026 Filippo Maria Cataldo and Marco Sagliano*

fact that in the PIPG formulation states were not doubled into both local and global copies. We observe the same behavior also in the 6-DoF case. In any case, PIPG outperforms both the other methods when considering total time. For the benchmark solver the main bottleneck is the parsing time, in fact the problem must be parsed again at each SCP iteration, after linearization. ADMM and PIPG methods on the other hand, only require a relatively short time to define matrices and problem structure, and are thus in that specific regard very efficient.

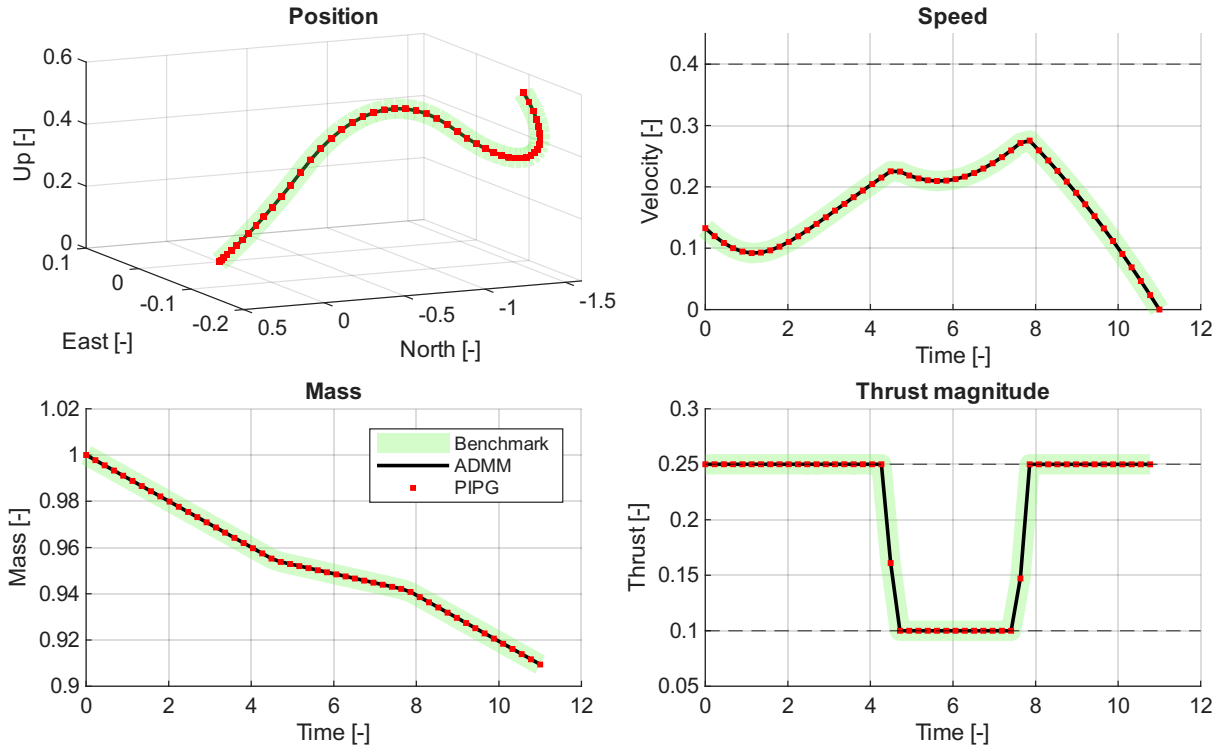

Figure 2 *3-DoF solution. Dashed lines indicate bounds on quantities. All trajectories are superimposed*

	3-DoF			6-DoF		
	Benchmark	**ADMM**	**PIPG**	**Benchmark**	**ADMM**	**PIPG**
Solve time [s]	0.014	2.058	0.888	1.35	270.89	80.80
Total time [s]	0.977	2.103	0.961	114.35	272.91	88.96
SCP iterations	–	–	–	23	22	23

Table 3 *Computational time analysis for the 3-DoF and 6-DoF cases.*

5 Conclusion

In this work we showed the validity of first-order methods in solving PDG optimization problems. After providing a general description of two first-order algorithms, namely ADMM and PIPG, we employed them to solve both a 3-DoF and 6-DoF rocket landing problem. The first one admitted a convex reformulation, and we could solve it directly; the second one presented non-convexities, thus we solved it by exploiting SCP. We compared both methods with the standard solver MOSEK, showing that the proposed approaches are able to converge to the same solution of the benchmark. In future work we will investigate the parallelizability and more advanced stopping criteria, and we will work on optimized versions of these methods to increase their computational efficiency.

© *2026 Filippo Maria Cataldo and Marco Sagliano*

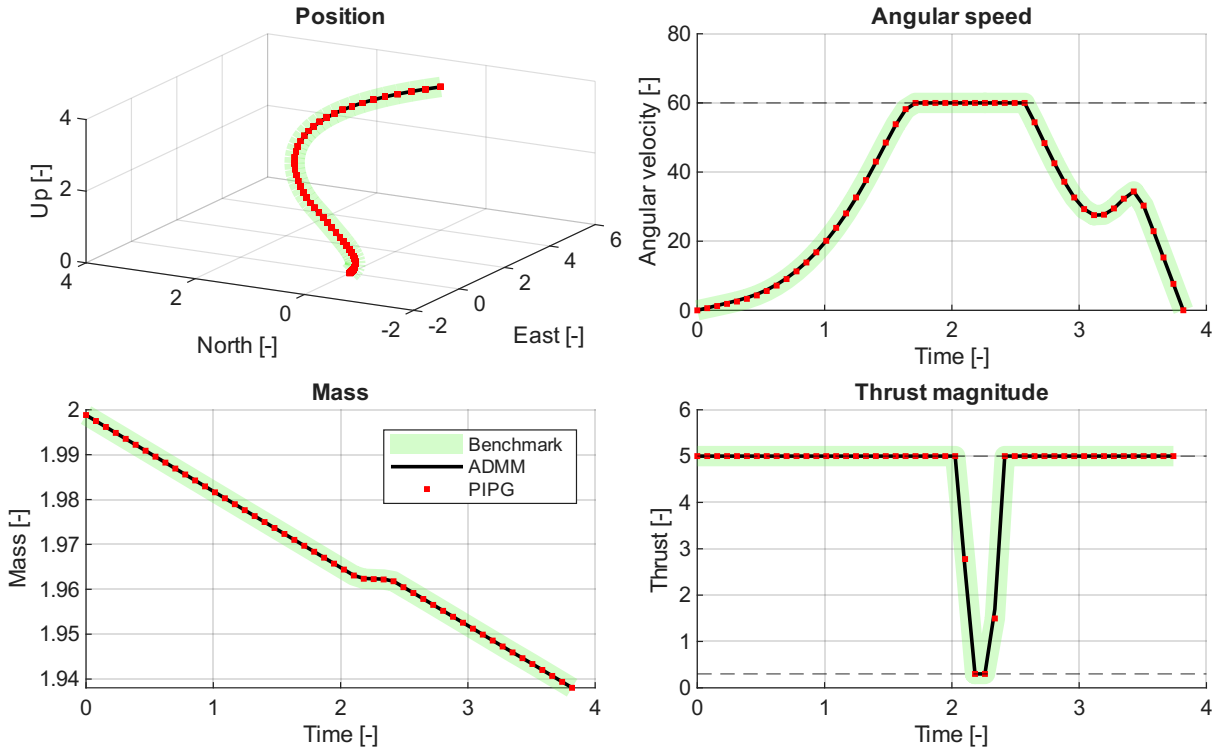

Figure 3 *6-DoF solution. Dashed lines indicate bounds on quantities. All trajectories are superimposed.*

Acknowledgments

Funded by the European Union (ERC, STARGATE, 101169959). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.



References

[1] S. P. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, England: Cambridge University Press, 2004.

© *2026 Filippo Maria Cataldo and Marco Sagliano*

[2] B. Açıkmeşe and S. R. Ploen, "Convex Programming Approach to Powered Descent Guidance for Mars Landing," *Journal of Guidance, Control, and Dynamics*, vol. 30, no. 5, pp. 1353–1366, Sep. 2007. DOI: 10.2514/1.27553

[3] J. Duchi, S. P. Boyd, and J. Mattingley, "Sequential Convex Programming," Stanford University, Stanford, CA, Lecture notes, 2018. [Online]. Available: https://web.stanford.edu/class/ee364b/lectures/seq_notes.pdf

[4] H. Everett III, "Generalized Lagrange Multiplier Method for Solving Problems of Optimum Allocation of Resources," *Operations Research*, vol. 11, no. 3, pp. 399–417, Jun. 1963. DOI: 10.1287/opre.11.3.399

[5] L. S. Lasdon, *Optimization Theory for Large Systems* (Macmillan Series for Operations Research). New York, NY: Macmillan Inc., 1970.

[6] N. Z. Shor, *Minimization Methods for Non-Differentiable Functions* (Springer Series in Computational Mathematics). Berlin, Germany: Springer Berlin Heidelberg, 1985, vol. 3. DOI: 10.1007/978-3-642-82118-9

[7] M. R. Hestenes, "Multiplier and Gradient Methods," *Journal of Optimization Theory and Applications*, vol. 4, no. 5, pp. 303–320, Nov. 1969. DOI: 10.1007/BF00927673

[8] D. Gabay and B. Mercier, "A dual algorithm for the solution of nonlinear variational problems via finite element approximation," *Computers & Mathematics with Applications*, vol. 2, no. 1, pp. 17–40, 1976. DOI: 10.1016/0898-1221(76)90003-1

[9] S. P. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers," *Foundations and Trends® in Machine Learning*, vol. 3, no. 1, pp. 1–122, Jul. 2011. DOI: 10.1561/2200000016

[10] Y. Yu, P. Elango, U. Topcu, and B. Açıkmeşe, "Proportional–integral projected gradient method for conic optimization," *Automatica*, vol. 142, p. 110 359, Aug. 2022. DOI: 10.1016/j.automatica.2022.110359

[11] J. Choi and J.-H. Kim, "A Distributed Method for First-Order Optimization With Expansive Projection for Powered Descent Guidance," in *AIAA SCITECH 2025 Forum*, Orlando, FL: American Institute of Aeronautics and Astronautics, Jan. 2025. DOI: 10.2514/6.2025-1507

[12] P. Elango, A. Kamath, Y. Yu, J. M. Carson, and B. Açıkmeşe, "A Customized First-Order Solver for Real-Time Powered-Descent Guidance," in *AIAA SCITECH 2022 Forum*, San Diego, CA & Virtual: American Institute of Aeronautics and Astronautics, Jan. 2022. DOI: 10.2514/6.2022-0951

[13] A. G. Kamath et al., "Customized Real-Time First-Order Methods for Onboard Dual Quaternion-based 6-DoF Powered-Descent Guidance," in *AIAA SCITECH 2023 Forum*, National Harbor, MD & Virtual: American Institute of Aeronautics and Astronautics, Jan. 2023. DOI: 10.2514/6.2023-2003

[14] Y. Yu, P. Elango, B. Açıkmeşe, and U. Topcu, "Extrapolated Proportional-Integral Projected Gradient Method for Conic Optimization," *IEEE Control Systems Letters*, vol. 7, pp. 73–78, 2023. DOI: 10.1109/LCSYS.2022.3186647

[15] Y. Yu, P. Elango, and B. Açıkmeşe, "Proportional-Integral Projected Gradient Method for Model Predictive Control," *IEEE Control Systems Letters*, vol. 5, no. 6, pp. 2174–2179, Dec. 2021. DOI: 10.1109/LCSYS.2020.3044977

[16] M. Szmuk and B. Açıkmeşe, "Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time," in *2018 AIAA Guidance, Navigation, and Control Conference*, Kissimmee, FL: American Institute of Aeronautics and Astronautics, Jan. 2018. DOI: 10.2514/6.2018-0617

[17] Y. Mao, M. Szmuk, X. Xu, and B. Açıkmeşe, *Successive Convexification: A Superlinearly Convergent Algorithm for Non-convex Optimal Control Problems*, Feb. 2019. DOI: 10.48550/arXiv.1804.06539

[18] M. ApS, "MOSEK Modeling Cookbook, Release 3.4.0," Copenhagen, Denmark, Manual, Nov. 2025. [Online]. Available: https://docs.mosek.com/modeling-cookbook/index.html

[19] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *2004 IEEE International Conference on Robotics and Automation (IEEE Cat. No.04CH37508)*, Taipei, Taiwan: IEEE, 2004, pp. 284–289. DOI: 10.1109/CACSD.2004.1393890

© *2026 Filippo Maria Cataldo and Marco Sagliano*