AAS 22-604

NOMINAL AND EMERGENCY ROCKET LANDING GUIDANCE USING QUADRATIC PROGRAMMING

Hubert Ménou[*], Eric Bourgeois[†], and Nicolas Petit[‡]

This paper presents nominal and emergency Powered Descent Guidance methods for rocket landing, in the presence of atmosphere. Nominal guidance is seen as a minimum-effort correction problem to a reference trajectory, in free final-time, with non-trivial aerodynamic effects and mixed state-control constraints. It is solved using Quadratic Programming. Emergency guidance – dealing with infeasible nominal scenarios – is described as a hierarchical negotiation of landing problem parameters and is solved using Linear Programming. Nominal and emergency guidance are merged into a single unifying algorithm. The overall computational complexity remains low and compatible with on-board usage. Numerical simulations are provided.

INTRODUCTION

Reusable launcher design is currently a salient topic in aerospace. Landing guidance - or more precisely, Powered Descent Guidance (PDG) - aims at providing the launcher with a reference trajectory for landing, given initial positions and velocities while enforcing multiple constraints. The subject has been extensively studied for planetary landing,[1] when there are no or negligible atmospheric effects. In the presence of an atmosphere such as Earth's, it is still a challenging task. The aerodynamic effects induce non-linearities that rule out analytical guidance methods.[2]

Several numerical methods are described in the literature. Successive Convexification have been extensively discussed,[3–5] is often used along with Lossless Convexification[6,7] and has been field tested.[8] Also, pseudospectral methods have been explored[9] as well as other numerical methods.[10–12] See Reference 13 for a recent survey. All these methods have in common that PDG is formulated as a finite-dimensional optimization problem that must be solved on-board. The inputs of the guidance problem are the states of the rocket, and known disturbances (such as the wind velocity for instance).

Emergency guidance - sometimes known as abort guidance[14,15] - aims at providing a reference trajectory when the nominal guidance procedure failed. For example, if the rocket starts its descent with a horizontal position too far from the desired landing site, the latter is not reachable anymore. However, providing no trajectory for the PDG is not acceptable, and if some trade-off yields a solution, then it must be considered. Mathematically, this means that the guidance optimization problem must be relaxed to partially recover feasibility, by sacrificing landing targets or by weakening designated safety constraints for instance. Emergency guidance can be formulated in many different ways

[*]PhD Candidate, hubert.menou@mines-paristech.fr, Centre Automatique et Systèmes (CAS), MINES Paris, PSL University, 75006 Paris, France.
[†]eric.bourgeois@cnes.fr, CNES, Direction des lanceurs, 52, rue Jacques Hillairet, 75612 Paris Cedex, France.
[‡]Pr., nicolas.petit@mines-paristech.fr, Centre Automatique et Systèmes (CAS), MINES Paris, PSL University, 75006 Paris, France.

and is deeply related to the nominal guidance method, the latter being the one declaring infeasibility in the first place.

In this paper, a PDG method defined in the vincinity (ideally large) of a reference trajectory and computed using only Quadratic Programming (QP) and Linear Programming (LP) is proposed, for which reliable off-the-shelf solvers exist. Here, QP is used directly for itself, and not as part of a Successive QP algorithm, hence guaranteeing the termination of the proposed procedure. These programs exploit offline-computed matrices, defined using standard properties of the flow of Ordinary Differential Equations (ODEs). Extending previous work of the authors,[12] we propose a way to handle emergency scenarios, by negotiating sub-sets of parameters of the landing problem. For example, the final horizontal position, or the safety incidence bound are parameters that can be sacrificed, to some extent. The proposed method explores a hierarchy of problems constructed from the ranking of constraints. This is the main contribution of the article.

This paper is structured as follows. First, the atmospheric PDG problem is presented in detail for a planar motion. Second, the QP-based guidance method is presented. Then, introducing modifications to the PDG problem and the nominal guidance method, the emergency problem is solved using a sequence of Linear Programs. Finally, numerical assessment and general discussions on this approach are provided.

ATMOSPHERIC POWERED DESCENT GUIDANCE

This section presents the rocket dynamic model, and introduces the PDG problem.

Dynamics model for guidance

First, let us introduce a planar rocket model. A non-rotating flat Earth model is considered, with a constant gravity field of magnitude g, oriented vertically. The rocket description presented here being used for guidance purposes, only its point dynamics is considered. For $h \geq 0$ the altitude, $v_h < 0$ the vertical speed, z the horizontal position, v_z the horizontal speed and $m \geq m_{\text{dry}}$ the total mass of the rocket, the Equations of Motion (EoM) are

$$\dot{h} = v_h \tag{1a}$$

$$\dot{v}_h = -g + \frac{F_L \sin\theta + (T - F_D)\cos\theta}{m} \tag{1b}$$

$$\dot{z} = v_z \tag{1c}$$

$$\dot{v}_z = \frac{F_L \cos\theta - (T - F_D)\sin\theta}{m} \tag{1d}$$

$$\dot{m} = -q \tag{1e}$$

where q is the engine flow, θ the attitude, and

$$\begin{aligned} \text{(Thrust)} \quad T &:= g\,\mathrm{I\small{SP}}\,q - P_a(h)S_{\mathrm{e}}, \\ \text{(Relative Speed)} \quad V_r &:= \sqrt{v_h^2 + (v_z - w(h))^2}, \\ \text{(Drag)} \quad F_D &:= \frac{1}{2}\rho(h)V_r^2 S_{\mathrm{ref}} C_D(M_a, \alpha), \\ \text{(Lift)} \quad F_L &:= \frac{1}{2}\rho(h)V_r^2 S_{\mathrm{ref}} C_L(M_a, \alpha), \end{aligned}$$

$$\text{(Mach)}\quad M_a := V_r / S_{\text{sp}}(h),$$
$$\text{(Incidence)}\quad \alpha := \arctan\left(\frac{v_z - w(h)}{|v_h|}\right) - \theta.$$

In these equations, ISP the engine specific impulse, P_a and ρ the atmospheric pressure and density, C_D and C_L the drag and lift coefficients, S_{sp} the sound speed and S_{ref} and S_{e} characteristic rocket surfaces. The pressure bias in the thrust expression conveys the air flow envelop effect on the rocket when landing. When the rocket descends vertically with no wind, $\alpha = \theta = 0$. The wind is assumed purely horizontal. Its speed is denoted $w(h)$. Classically, the wind speed profile is assumed to be piece-wise affine, null at $h = 0$ and defined by its values w_1 at low altitude and w_2 at high altitude. The engine thrust T is assumed to be positive at all times. This very general model has some specifications detailed below.

The aerodynamic model of a rocket moving in the direction of its thrust flame is notoriously hard to determine. Early work from 1966 started to describe the aerodynamic effect of an air jet pushing in front of a body in a supersonic flow. Later work from the early 2000s from JAXA completed these observations,[16] followed by recent experiments of the DLR.[17] The common observation of References 16 and 17 is that for a sufficient jet flow, how it wraps around the rocket body lowers dramatically the drag along the body axis, though orthogonal effects remain strong for non-zero incidences. Therefore, for this paper, we consider $|C_D| << 1$ and a non-trivial expression of C_L based on lookup tables. Naturally, C_L is taken skew-symmetric with respect to α.

Moreover, the rocket engine thrust is assumed to always generate a greater acceleration than the rocket weight. Initially, and during the whole flight, it implies that the vertical speed is always negative, and that the rocket altitude is *always decreasing*.

The states are gathered in $x := (h, v_h, z, v_z, m)^\top$ of dimension $n = 5$. The control is conveyed by $u := (q, \alpha)^\top$ and is of dimension $m = 2$. Finally, the dynamics parameters are chosen[*] as $\eta := (w_1, w_2)^\top$. The EoM are of the shape $\dot{x} = f(x, u, \eta)$, with initial conditions $x(0) = x^0$ and are defined on the time interval $[0, t_f]$. The landing problem can now be stated.

Landing targets and constraints

The main goal in PDG is to provide a trajectory that steers the rocket to the proper landing site while enforcing several constraints. The landing problem is here defined as a *correction* problem with respect to a reference trajectory. This trajectory, denoted $(\bar{x}, \bar{u}, \bar{t}_f)$, is assumed to be computed offline, to satisfy the EoM, and the problem constraints described below. The idea is to find the *best* correction, i.e. that remains as close as possible to the reference control law.

The control correction is described by a parametric function $t \mapsto u_\mu(t)$, where μ is a parameter. For mathematical convenience, let us impose that $\mu \mapsto u_\mu(t)$ be linear for any t, allowing a wide variety of parametric descriptions such as piecewise affine maps, Cubic Splines or piecewise Hermite Polynomials.[18] By definition, we also require that $u_0(.) \equiv 0$. Thus, there is a matrix-valued function $t \mapsto M(t)$ such that $u_\mu(t) = M(t)\mu$. This matrix solely depends on the chosen parametric description. We can now describe the constraints that we wish to enforce.

First, there are *pure control constraints*. In practice, rockets are equipped with engines that naturally have non-instantaneous responses. It implies that the control law and its derivative must be continuous, and should not vary *too abruptly*. The continuous differentiability can be naturally

[*]The methodology being generic, one can select another set of variables as dynamics parameters if wanted (e.g. ISP).

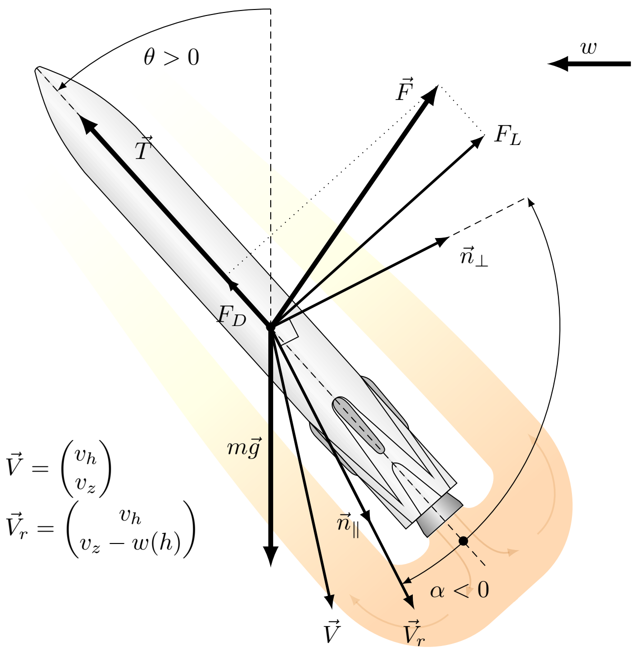

Figure 1. ***Angles and forces.*** **α denotes the incidence, θ the attitude, $\vec{V}_r$ the relative speed, $\vec{F}$ the aerodynamic reaction (lift and drag) and $\vec{T}$ the thrust. The wake of the thrust flame, depicted in orange shades, induces an altitude bias of the thrust.**

ensured by the specific parametric descriptions - such as Cubic Splines for instance.[18] Moreover, for consistency, at $t = 0$ the control is constrained to equal the current value of the control u_{init} (possibly different from $\bar{u}(0)$). This grants continuity of q and α and the initial time. In practice, this constraint is imposed by enforcing

$$u_\mu(0) = \Delta u_{\text{init}} := u_{\text{init}} - \bar{u}(0).$$

Moreover, the engine flow is bounded (mechanical limits of the engine) and the incidence should remain within acceptable limits $\pm\alpha_{\max}$.

Then, *mixed state-control constraints* are considered. A maximum value is imposed on the load factor a_{n}, i.e. the non-gravitational acceleration of the rocket along the axis $\vec{n}_\perp$, the latter being defined as the unit vector perpendicular to the relative velocity, as shown in Figure 1.

Finally, we wish to enforce terminal constraints on the states, conveying the landing site location, and the final speed at landing, such that $h = 0$, $v_h = -\varepsilon^f_{v_h} < 0$, $z = 0$, and $v_z = 0$. For convenience, we write $A^f x(t_f) = b^f$, where A^f is a truncation of the identity matrix of dimension n, and b^f is filled with zeros and the constant $\varepsilon^f_{v_h}$.

The inputs of the PDG problem are Δx^0, $\Delta\eta$ and Δu_{init}, where $\Delta x^0 := x^0 - \bar{x}^0$ and $\Delta\eta = \eta - \bar{\eta}$, as shown in Figure 2. The decision variable - or output - is denoted by $z := (\mu^\top, \Delta t_f)^\top$ and is of dimension N_z. The cost that assesses how close a trajectory is from the reference control is defined as a weighted quadratic map of z, i.e. $J(z) := \frac{1}{2} z^\top P z$, for some positive definite matrix P.

Remark 1. *It should be noted that several articles from the literature consider glide-slope con-*

straints, especially when it comes to planetary landers.[1,5,13] *Considering our high engine thrust assumption, glide-slope constraints would be less relevant in our framework.*

General Powered Descent Guidance problem

Let us introduce the following problem[*], which is an optimization problem with a finite-dimensional variable and an infinite number of constraints

$$\min_{z\in\mathbb{R}^{N_z}} \quad J(z) \tag{2a}$$
$$\text{s.t.:} \quad \dot{x} = f(x, \bar{u} + u_\mu, \bar{\eta} + \Delta\eta) \tag{2b}$$
$$x(0) = \bar{x}^0 + \Delta x^0 \tag{2c}$$
$$A^f x(t_f + \Delta t_f) = b^f \tag{2d}$$
$$u_\mu(0) = \Delta u_{\text{init}} \tag{2e}$$
$$u^- \leq \bar{u}(.) + u_\mu(.) \leq u^+ \tag{2f}$$
$$a_{\min} \leq a_{\text{n}}(x(.), \bar{u}(.) + u_\mu(.), \bar{\eta} + \Delta\eta) \leq a_{\max} \tag{2g}$$

Note that in this paper, all the inequalities must be understood component-wise.

As mentioned above, the reference trajectory is assumed to satisfy the conditions (2b) to (2g) for null inputs. Also, note that due to the expression of J, the reference trajectory is the best control law for null inputs Δx^0 and $\Delta\eta$, i.e. $z = 0$ is the unique global optimum for $\Delta x^0 = 0$ and $\Delta\eta = 0$.

In the constraints (2b) to (2g), u_μ is implicitly defined on $[0, t_f + \Delta t_f]$, which can be ambiguous when differentiating some quantities with respect to Δt_f (see e.g. Section 2.7 in Reference 19). To lift this potential ambiguity, Problem (2) is re-written on a fixed and normalized time-interval. The new time variable is $\tau := t/(t_f + \Delta t_f) \in [0, 1]$, and t_f is considered as an extra state with null dynamics. The augmented state is $\tilde{x} := (x^\top, t_f)^\top \in \mathbb{R}^{n+1}$. The augmented dynamics and the inequality constraints are

$$\tilde{f}(\tilde{x}, u, \eta) := \begin{pmatrix} t_f f(x, u, \eta) \\ 0 \end{pmatrix} \quad \text{and} \quad g(\tilde{x}, u, \eta) := \begin{pmatrix} \bar{u}(\tau) + u_\mu(\tau) - u^+ \\ u^- - \bar{u}(\tau) - u_\mu(\tau) \\ a_{\text{n}}(x, u, \eta) - a_{\max} \\ a_{\min} - a_{\text{n}}(x, u, \eta) \end{pmatrix}.$$

Then, Problem (2) is strictly equivalent to

$$\text{PDG}\left(\Delta x^0, \Delta\eta\right) \quad := \quad \min_{z\in\mathbb{R}^{N_z}} \quad J(z) \tag{3a}$$
$$\text{s.t.:} \quad \dot{\tilde{x}}(\tau) = \tilde{f}(\tilde{x}(\tau), \bar{u}(\tau) + u_\mu(\tau), \bar{\eta} + \Delta\eta), \quad \forall\tau \in [0, 1] \tag{3b}$$
$$\tilde{x}(0) = \begin{pmatrix} \bar{x}^0 + \Delta x^0 \\ \bar{t}_f + \Delta t_f \end{pmatrix} \tag{3c}$$
$$g(\tilde{x}(\tau), \bar{u}(\tau) + u_\mu(\tau), \bar{\eta} + \Delta\eta) \leq 0, \quad \forall\tau \in [0, 1] \tag{3d}$$
$$A^f \tilde{x}(1) = b^f. \tag{3e}$$
$$u_\mu(0) = \Delta u_{\text{init}} \tag{3f}$$

[*]To alleviate the writing, Δu_{init} will not be written in the inputs of PDG, as its role is less critical than the others.

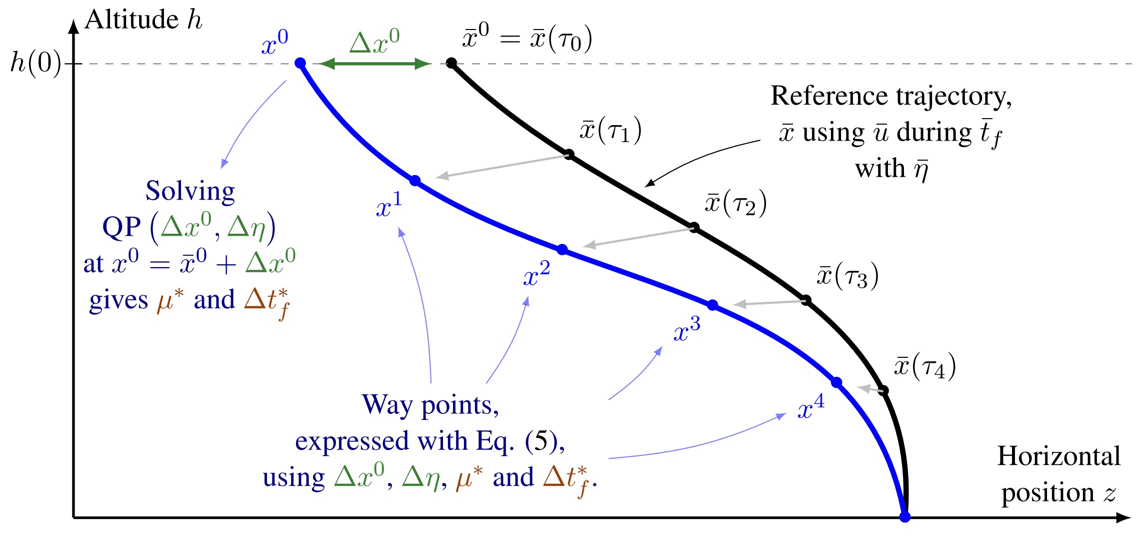
Altitude h
$h(0)$
x^0
Δx^0
$\bar{x}^0 = \bar{x}(\tau_0)$
Reference trajectory, $\bar{x}$ using $\bar{u}$ during $\bar{t}_f$ with $\bar{\eta}$
$\bar{x}(\tau_1)$
$\bar{x}(\tau_2)$
$\bar{x}(\tau_3)$
$\bar{x}(\tau_4)$
Solving QP $(\Delta x^0, \Delta \eta)$ at $x^0 = \bar{x}^0 + \Delta x^0$ gives μ^* and Δt_f^*
x^1
x^2
x^3
x^4
Way points, expressed with Eq. (5), using Δx^0, $\Delta \eta$, μ^* and Δt_f^*.
Horizontal position z

Figure 2. Optimal correction framework. The reference trajectory is chosen depending on the mission's needs. Solving QP $(\Delta x^0, \Delta \eta)$ provides an optimal correction, to stay close to the reference, enforce several constraints and land at the expected spot. See Figure 4 for the application of this principle on an actual model.

To alleviate the writing, we keep the same notation for A^f, since the matrix A^f from (3e) only has an additional null column compared to the one from (2d). Note that b^f on the other hand does not change.

With this new writing, the variable Δt_f becomes a freely chosen initial condition. From now on, the state and control variables will always be expressed on the time interval $[0, 1]$. Providing a solution to PDG $(\Delta x^0, \Delta \eta)$, even for relatively large values of Δx^0 and $\Delta \eta$, is the goal of the next sections.

Remark 2. *Note that in practice, to improve the numerical conditioning of the following subproblems, all the variables are normalized, such that they have the same order of magnitude.*[20]

RESOLUTION OF NOMINAL GUIDANCE VIA QUADRATIC PROGRAMMING

This section describes the method used to solve PDG $(\Delta x^0, \Delta \eta)$, and extends previous work of the authors.[12] First, Problem (3) is converted to a Non-Linear Program (NLP). Then, using sensitivity analysis, a Quadratic Program (QP) is introduced to compute its solutions.

Conversion to Nonlinear Programming

First, dynamic equation (3b) is represented using the flow $\Phi_{\tilde{f}}$ of $\tilde{f}$, whose definition and first order expansion are recalled in the Appendix. For any time $\tau \in [0, 1]$, the augmented state at τ is defined such that

$$\tilde{x}(\tau, z) := \Phi_{\tilde{f}}\left(\tau, \begin{pmatrix} \bar{x}^0 + \Delta x^0 \\ t_f + \Delta t_f \end{pmatrix}, \bar{\eta} + \Delta \eta; \bar{u} + u_\mu\right) \in \mathbb{R}^{n+1}. \tag{4}$$

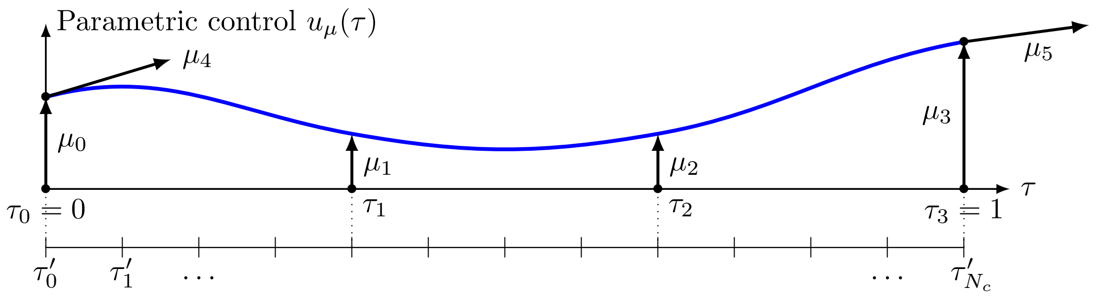

Figure 3. Control discretization and constraint time instances. The correction is described by a parametric function $\tau \mapsto u_\mu(\tau)$. Here is represented a Cubic Spline, described by its values $\mu_0, \dots, \mu_3$ at several time instances, and by its slopes μ_4 and μ_5 at the starting and end points. The inequality constraints are enforced on the subdivision $\tau'_0, \dots, \tau'_{N_c}$.

To alleviate the writing, note that $\tilde{x}(\tau, z)$ depends on Δx^0 and $\Delta\eta$ but this dependency is *not* written explicitly. The term $\tilde{x}(\tau, z)$ has a first order expansion with respect to all the previous problem variables such that

$$\tilde{x}(\tau, z) \approx \tilde{x}[\tau] + \frac{\partial \tilde{x}}{\partial x^0}[\tau].\Delta x^0 + \frac{\partial \tilde{x}}{\partial \eta}[\tau].\Delta\eta + \frac{\partial \tilde{x}}{\partial \mu}[\tau].\mu + \frac{\partial \tilde{x}}{\partial x^0_{n+1}}[\tau].\Delta t_f \tag{5}$$

where $[\tau]$ means that the associated term is evaluated at τ for null values of Δx^0, $\Delta\eta$, μ and Δt_f.

Second, let us enforce the constraints (2f) and (2g) on a finite number of points. In all of the above-mentioned examples of parametric descriptions, the map $\mu \mapsto u_\mu(t)$ is defined in practice with respect to a time subdivision $(\tau_i)_i$ such that $0 = \tau_0 < \dots < \tau_{N_u} = 1$. For example, for continuous piecewise affine functions, μ simply conveys to the values of u_μ at each prescribed time instance τ_i. Using this fact, we choose to enforce (2f) and (2g) on the subdivision $0 = \tau'_0 < \dots < \tau'_{N_c} = 1$, which is a uniformly over-sampled version of the subdivision $(\tau_i)_i$, as pictured in Figure 3.

By introducing the maps

$$c_{\text{in}}(z, \Delta x^0, \Delta\eta) := \begin{pmatrix} g(\tilde{x}[\tau'_0, z], \bar{u}(\tau'_0) + u_\mu(\tau'_0), \bar{\eta} + \Delta\eta) \\ \vdots \\ g(\tilde{x}[\tau'_{N_c}, z], \bar{u}(\tau'_{N_c}) + u_\mu(\tau'_{N_c}), \bar{\eta} + \Delta\eta) \end{pmatrix}$$
$$c_{\text{eq}}(z, \Delta x^0, \Delta\eta) := \begin{pmatrix} A^f \tilde{x}[\tau'_0, z] - b^f \\ u_\mu(0) - \Delta u_{\text{init}} \end{pmatrix},$$

one can approximate Problem (2) into a Nonlinear Problem (NLP) fitting the following standard form

$$\text{NLP}\left(\Delta x^0, \Delta\eta\right) \quad := \quad \min_{z} \quad J(z) \tag{6a}$$
$$\text{s.t.:} \quad c_{\text{in}}(z, \Delta x^0, \Delta\eta) \leq 0, \tag{6b}$$
$$c_{\text{eq}}(z, \Delta x^0, \Delta\eta) = 0. \tag{6c}$$

Resolution via Quadratic Programming

The solution of NLP $\left(\Delta x^0, \Delta\eta\right)$ can be approximated using perturbation methods, also know as sensitivity analysis. For $\Delta x^0 = 0$ and $\Delta\eta = 0$, the solution is known and is $z = 0$. As it is well established in Reference 21, for small values of Δx^0 and $\Delta\eta$, if local preservation of the set of active constraints can be guaranteed, then the solution of NLP $\left(\Delta x^0, \Delta\eta\right)$ is continuously differentiable. However, such a property is often guaranteed by assuming *Strict Complementarity Slackness*, which is unsufficient in the above-defined framework. We need to be able to handle local changes in the set of active constraints. Thus, we overcome this difficulty by using more subtle results from the literature.[22,23] Below are presented the key aspects of the method. For a more detailed description, see Reference 12.

The goal is to provide a way to compute a first order expansion of the optimal solution map of NLP $\left(\Delta x^0, \Delta\eta\right)$

$$\Delta x^0, \Delta\eta \mapsto z^*(\Delta x^0, \Delta\eta).$$

Under mild conditions,[12] this map exists and is directionally differentiable (in the sens of the Dini derivatives), in the vincinity of $(\Delta x^0, \Delta\eta) = (0, 0)$. To state this result in more details, let us first introduce the following QP written in standard form*

$$\begin{aligned} \text{QP}\left(\Delta x^0, \Delta\eta\right) \quad := \quad \min_{z} \quad & \frac{1}{2} z^\top P z \\ \text{s.t.} \quad & Gz \leq h, \\ & Az = b. \end{aligned}$$

where the vectors h and b bear the dependency on the inputs Δx^0 and $\Delta\eta$ such that

$$h := h_0 + H_x \Delta x^0 + H_\eta \Delta\eta$$
$$b := b_0 + B_x \Delta x^0 + B_\eta \Delta\eta$$

and where the remaining constant matrices are

$$\begin{aligned} G &:= \frac{\partial c_{\text{in}}}{\partial z}[0], & h_0 &:= -c_{\text{in}}[0], & H_x &:= -\frac{\partial c_{\text{in}}}{\partial \Delta x^0}[0], & H_\eta &:= -\frac{\partial c_{\text{in}}}{\partial \eta}[0], \\ A &:= \frac{\partial c_{\text{eq}}}{\partial z}[0], & b_0 &:= -c_{\text{eq}}[0], & B_x &:= -\frac{\partial c_{\text{eq}}}{\partial \Delta x^0}[0], & B_\eta &:= -\frac{\partial c_{\text{eq}}}{\partial \eta}[0]. \end{aligned}$$

where $[0]$ means that the associated terms are evaluated at $(0, 0, 0)$.

For this subsection only, denote respectively by z_{nlp} and z_{qp} the solutions of NLP $\left(\Delta x^0, \Delta\eta\right)$ and QP $\left(\Delta x^0, \Delta\eta\right)$. As discussed earlier, $z = 0$ globally minimizes J and satisfies the constraints for null inputs, the $z_{\text{nlp}}(0, 0) = z_{\text{qp}}(0, 0) = 0$. Then, for $\varepsilon \geq 0$, sensitivity analysis results[12,22,23] allow us to write the expansion

$$z_{\text{nlp}}(\varepsilon\Delta x^0, \varepsilon\Delta\eta) = z_{\text{qp}}(\varepsilon\Delta x^0, \varepsilon\Delta\eta) + o(\varepsilon). \tag{7}$$

The main advantages of approximating z_{nlp} by solving QP $\left(\Delta x^0, \Delta\eta\right)$ is that on one hand it supports local changes in the active set, and on the other hand it provides reasonable approximations of the solutions of NLP $\left(\Delta x^0, \Delta\eta\right)$ even for non-local values of Δx^0 and $\Delta\eta$. For more details, see previous work of the authors[12] and the references therein. Also, note that the constant matrices G, h_0, H_x, H_η, A, b_0, B_x and B_η depend directly on the nominal reference trajectory.

To summarize, our nominal guidance method consist in computing the optimal solution of QP $\left(\Delta x^0, \Delta\eta\right)$, which provides an approximate solution of NLP $\left(\Delta x^0, \Delta\eta\right)$.

*The fact that the costs of QP $\left(\Delta x^0, \Delta\eta\right)$ and NLP $\left(\Delta x^0, \Delta\eta\right)$ coincide comes from the fact that J is already quadratic in z and that $z = 0$ is the solution of the latter problem for null inputs. See Reference 12 for more details.

Remark 3. *Note that this approach cannot be realistically used as an elementary step for online Successive Quadratic Programming,*[3,20] *for the matrices of each QPs would take too much time to compute. Yet, the strength of* QP $\left(\Delta x^0, \Delta\eta\right)$ *is twofold: it provides an excellent solution in a non-trivial neighborhood of the reference trajectory and the values returned for high magnitudes of* Δx^0 *and* $\Delta\eta$ *remain very consistent, as demonstrated by extensive simulations.*

EMERGENCY GUIDANCE

A salient source of concern is what happens when QP $\left(\Delta x^0, \Delta\eta\right)$ is not feasible, due to the values of Δx^0 and $\Delta\eta$ being too large. For instance, this occurs when the initial horizontal position is too far from the landing site, because the incidence is bounded. The landing problem must be modified to some extent to gain enough margin to recover feasibility. The main questions that arise are: what can we negotiate to recover feasibility, and how do we determine it?

Negotiable parameter choices

The parameters that can be negotiated are the ones describing the goals of the landing. The physics-based equations of motion are not negotiable. However, the location of the landing site is negotiable, to some extent. If the landing site is located in a wide and flat area, it is of interest to allow landings in a neighborhood of the ideal landing site*. Moreover, some of the other parameters defining the constraints can be partially loosened. For example, the incidence limit should be seen more as a safety constraint and could be slightly widened if necessary, whereas the engine flow limitations are non-negotiable mechanical constraints.

Most of the available negotiable variables can be added with ease in the original guidance problem, as illustrated below. For instance, to negotiate the final horizontal position, the terminal constraint (3e) can by altered by a parameter Δz^f, such that

$$A^f x(1) = b^f + \begin{pmatrix}0 & 0 & \Delta z^f & 0\end{pmatrix}^\top .$$

Likewise, to negotiate the incidence limit, the control bounds of PDG $\left(\Delta x^0, \Delta\eta\right)$ are shifted by a parameter $\Delta\alpha_{\max}$ such that

$$u^- - (0, \Delta\alpha_{\max})^\top \leq \bar{u}(t) + u_\mu(t) \leq u^+ + (0, \Delta\alpha_{\max})^\top .$$

Following these examples, we will study *negotiable parameters* that have a linear influence on the constraints of PDG $\left(\Delta x^0, \Delta\eta\right)$. For the sake of this article, we illustrate our approach with an arbitrary subset of negotiable parameters, which includes the incidence limits ($\Delta\alpha_{\max}$) and the final horizontal position (Δz^f). In the following, the variable p will be used to denote the chosen negotiable parameters, i.e. $p = (\Delta\alpha_{\max}, \Delta z^f)^\top$ above. Thus, we introduce matrices H_p and B_p such that the constraints of QP $\left(\Delta x^0, \Delta\eta\right)$ become the following *negotiated constraints*:

$$Gz \leq h + H_p p \quad \text{and} \quad Az = b + B_p p. \tag{8}$$

There is hierarchy of importance between the negotiable parameters. Indeed, accepting to have a slightly higher incidence during the flight is often less critical than landing outside the landing

*Note that this would not apply to landings on offshore platforms, for obvious reasons.

site. Thus, the negotiable parameters are partitioned in $R \geq 1$ different sub-parameters $p^{(j)}$ of equal importance, sorted in ascending importance between each other, such that

$$p = \left((p^{(1)})^\top \quad \dots \quad (p^{(R)})^\top\right)^\top \in \mathbb{R}^{n_{\text{neg}}}.$$

The higher the index j, the more critical is $p^{(j)}$.

Finally, it is assumed that all negotiable parameters are negotiable within prescribed limits, i.e. that p is bounded such that: $p_{\text{low}} \leq p \leq p_{\text{up}}$.

For our illustration example with $p = (\Delta\alpha_{\max}, \Delta z^f)$, we consider that it is less critical to sacrifice a few degrees of incidence limit than to land outside the desired landing site. Thus, we have $R = 2$, $p^{(1)} = \Delta\alpha_{\max}$ and $p^{(2)} = \Delta z^f$.

Remark 4. *Recovering feasibility has been tackled using various formalisms. In an important paper by Blackmore et al.,*[24] *the problem of finding the actual landing site that minimizes the distance to the desired landing site is described in details, for Mars landing (i.e. without atmosphere), using Successive Convexification. Using the above-mentioned taxonomy, they had two different negotiable parameter that they negotiated at the same time, which were the two final horizontal positions of their 3D lander model.*

Basic parameters negotiation

Now comes the second question: how does one *negotiate* these parameters? When landing is not feasible anymore, the goal is to find the smallest change in the negotiable parameters that recovers feasibility. Once these best negotiated parameters have been computed, the set of feasible parameters is not empty anymore. The latter set can be transferred to the original guidance problem, in order to re-optimize z and get the optimal trajectory z^*. One of the main challenges is to negotiate p such that the map $p \mapsto z^*(p)$ is continuous, *while* enforcing the parameters hierarchy.

Qualitatively, we propose to successively minimize the magnitude of the negotiable parameters while enforcing the existence of at least one feasible trajectory at each step. This minimization will focus on each sub-parameter, starting by the last one (i.e. $p^{(R)}$), down to the first one (i.e. $p^{(1)}$). This means that the most critical negotiable parameters are minimized first. At each step, only the proper sub-parameter $p^{(j)}$ is minimized, such that the result be $p^{(j)} = 0$ if it is not necessary to use this parameter to recover feasibility. Also, in order to enforce the desired hierarchy, a sort of memory effect will be needed so that each step takes into account the results of the previous steps, by preserving the penalty levels. This will be the role of condition (9e) below. Also, as will appear, Linear Programming will play a key role to implement these negotiation steps.

Quantitatively, let us first introduce the *negotiation problems* LP_j such that

$$\begin{aligned}
\text{LP}_j \quad := \quad \min_{z,p} \quad & \|p^{(j)}\|_1 && \text{(9a)}\\
\text{s.t.:} \quad & Gz \leq h + H_p p && \text{(9b)}\\
& Az = b + B_p p && \text{(9c)}\\
& p_{\text{low}} \leq p \leq p_{\text{up}} && \text{(9d)}\\
& \|p^{(i)}\|_1 = \mathcal{P}_i^*, \quad i = j+1, \dots, R && \text{(9e)}
\end{aligned}$$

where $\mathcal{P}_i^*$ denotes the optimal value of LP_i. To make this problem definition well-posed, note that the constraint (9e) does not exist when $j = R$. Moreover, note that the inputs of each LP_j are Δx^0,

$\Delta\eta$ and $\mathcal{P}_i^*$ for $i = j+1, \dots, R$. To alleviate the writing, these are omitted wherever the context is clear enough. Finally, it is important to highlight the fact that z and p are both optimization variables in LP$_j$, even though only a few coefficients of p are penalized in the cost (9a).

The role of LP$_j$ is to minimize a penalty on the j^{-th} negotiable sub-parameters, while making sure that there are still feasible trajectories z, and that the level of penalty of the previous negotiation problems are satisfied. The reason why constraint (9e) must be satisfied instead of a constraint of the type "$p^{(j)} = p^{(j)*}$" is that LP$_j$ does not necessarily have a unique solution*. Imposing $\|p^{(j)}\|_1 = \mathcal{P}_i^*$ makes sure that the level of negotiation reached at step i remain satisfied in the follow-up negotiations. Descending from $j = R$ down to $j = 1$ and imposing this latter constraint is what makes sure that the hierarchy is enforced.

Thus, solving successively LP$_j$ for $j = R, \dots, 1$ (decreasing indices) provides a way to recover feasibility, while hierarchically minimizing what is sacrificed from the original guidance problem, by building the sequence $\mathcal{P}_1^*, \dots, \mathcal{P}_R^*$ from the end. Among others, a noteworthy property of this sequence is that if landing is feasible without any negotiation, then solving LP$_j$ will return $\|p^{(j)}\|_1 = 0$ at each step, implying $p = 0$. Once this negotiation sequence has been computed, there may be many possible values for p, and for z as well. It is thus necessary to pick the best trajectory among these ones, by solving

$$\begin{aligned}
\textsc{Refine} \quad := \quad \min_{z,p} \quad & \frac{1}{2} z^\top P z && \text{(10a)}\\
\text{s.t.:} \quad & Gz \leq h + H_p p && \text{(10b)}\\
& Az = b + B_p p && \text{(10c)}\\
& p_{\text{low}} \leq p \leq p_{\text{up}} && \text{(10d)}\\
& \|p^{(i)}\|_1 = \mathcal{P}_i^*, \quad i = 1, \dots, R && \text{(10e)}
\end{aligned}$$

Note that like LP$_j$, the problem REFINE takes as inputs Δx^0, $\Delta\eta$ and $\mathcal{P}_i^*$ for $i = 1, \dots, R$.

Remark 5. *Using standard techniques from the litterature (see Example 1.13 of Reference 20),* LP$_j$ *can be written as a Linear Program using slack variables* $p^+ \geq 0$ *and* $p^- \geq 0$*, such that* $p = p^+ - p^-$*. Thus:* $\|p\|_1 = \mathbf{1}^\top(p^+ + p^-)$*, where* $\mathbf{1}^\top = (1, \dots, 1)$ *is of dimension* n_{neg}.

Remark 6. *It is important to note that any other choice of cost function in* (9a) *would deeply change the way Problems* (9) *and* (10) *are solved. Also, it is worth mentioning that if it is required to favor negotiations in some directions more specifically within a sub-parameter* $p^{(j)}$*, one can easily consider a weighted variant of the* 1*-norm instead.*

Remark 7. *From a very general mathematical programming point of view, recovering feasibility in Linear Programs has been discussed extensively in the literature, see Chinneck[25] for instance. Problem 9 builds upon right-hand side constraint "alteration" methods, by exploiting the available levers conveyed through the parameter* p*, the matrices* H_p *and* B_p*, and the need to enforce the parameter hierarchy.*

Error anticipation using margins

The above-mentioned approach would be sufficient to correctly handle emergency problems if QP $(\Delta x^0, \Delta\eta)$ was an exact representation of NLP $(\Delta x^0, \Delta\eta)$, which is not true in practice for large values of Δx^0 and $\Delta\eta$ which cause the first order constraint approximations to be erroneous.

*This is a common with Linear Programs. The solution set is often non-trivial and forms a convex set.

	NLP (.) is feasible	NLP (.) is **not** feasible
QP (.) is feasible	Matching behavior	Dangerous
QP (.) is **not** feasible	Conservative	Matching behavior

Table 1. List of possible outcomes. The riskiest situation is when QP $\left(\Delta x^0, \Delta\eta\right)$ declares guidance feasible but NLP $\left(\Delta x^0, \Delta\eta\right)$ would state the opposite.

There are two independent issues to deal with. First, the approximation performed in Equation (7) is not exact, and some errors build up when using the linearized constraints for high values of Δx^0 and $\Delta\eta$. Second, the linear nature of the constraint approximation implies that QP $\left(\Delta x^0, \Delta\eta\right)$ may declare the landing infeasible when it is actually feasible, and vice versa, as summarized in Table 1.

Thus, let us describe a simple way to cope with these issues, and improve the guidance algorithm robustness. Denote by $\mathcal{F}_{eas}\left(\Delta x^0, \Delta\eta, p\right)$ the set of feasible vectors z for the negotiated constraints from Equation (8). We want to restrict this set in order to reduce the "*false positive*" scenarios of Table 1, i.e. when NLP $\left(\Delta x^0, \Delta\eta\right)$ is not feasible but QP $\left(\Delta x^0, \Delta\eta\right)$ is. The idea is to impose that $\mathcal{F}_{eas}$ be non-empty for p, and also that $\mathcal{F}_{eas}$ be non-empty for all $p+\Delta p$ where Δp belongs to some prescribed set $\mathcal{M}$. This set $\mathcal{M}$ represents all the margins that one wishes to enforce. Therefore, Equation (8) is changed into the more abstract conditions

$$z \in \mathcal{F}_{eas}\left(\Delta x^0, \Delta\eta, p\right), \tag{11a}$$

$$\mathcal{F}_{eas}\left(\Delta x^0, \Delta\eta, p+\Delta p\right) \neq \emptyset, \quad \forall \Delta p \in \mathcal{M}. \tag{11b}$$

The former is equivalent to (8), and the latter represents the new margin constraints - i.e. an infinite number of conditions. Due to the linear nature of our constraints, and with mild assumptions on $\mathcal{M}$, it is possible to reduce it to a finite number of constraints. Let us assume that $\mathcal{M}$ is a convex set with a finite number K of extreme points[*], that are denoted $\Delta p^1, \ldots, \Delta p^K$. In other words, it means that $\mathcal{M} = \text{ConvexHull}\left(\Delta p^1, \ldots, \Delta p^K\right)$. Moreover, if we assume that $\mathcal{M}$ contains the null vector, then there exist non-negative scalars σ_k such that

$$\sum_{k=1}^{K} \sigma_k = 1 \quad \text{and} \quad \sum_{k=1}^{K} \sigma_k \Delta p^k = 0.$$

These two last assumptions are handy to transform the abstract condition (11b) into the equivalent condition

$$\mathcal{F}_{eas}\left(\Delta x^0, \Delta\eta, p+\Delta p^k\right) \neq \emptyset, \quad \forall k = 1, \ldots, K.$$

Due to the linearity of the underlying conditions, the latter expression can be directly plugged into LP$_j$ and REFINE.

Guidance method summary

The Powered Descent Guidance approach described above provides a way to compute a guidance control law, as well as guidance way-points.

The inputs of the guidance algorithm are simply the difference of the current rocket states and dynamic parameter with respect to mission-specific reference values. To compute the guidance

[*] $x \in \mathcal{M}$ is an extreme point of $\mathcal{M} \Leftrightarrow \nexists\ a, b \in \mathcal{M}$ and $\lambda \in (0,1)$ such that $a \neq b$ and $x = (1-\lambda)a + \lambda b$.

trajectory, conveyed by the variable $z = (\mu^\top, \Delta t_f)^\top$, it is then necessary to minimize the negotiable parameters and optimize z, as summarized by Algorithm 1, using a sequence of Linear and Quadratic Programs.

Algorithm 1 Nominal and emergency guidance methods, merged as a single algorithm.

```
Require: Difference with respect to the reference: $\Delta x^0$, $\Delta\eta$.
  for $j = R, \dots, 1$ (decreasing indices) do
    $\mathcal{P}_j^* \leftarrow \min\ \mathrm{LP}_j\left(\Delta x^0, \Delta\eta, \mathcal{P}_{j+1}^*, \dots, \mathcal{P}_R^*\right)$
  end for
  $z^* \leftarrow \mathrm{argmin}\ \textsc{Refine}\left(\Delta x^0, \Delta\eta, \mathcal{P}_1^*, \dots, \mathcal{P}_R^*\right)$
  return $z^*$
```

For the *argmin* operation, the value of p is voluntarily ignored, since it is not needed nor unique. However, z is unique.

It is important to observe that for any scenario for which landing is naturally feasible - i.e. near $\Delta x^0 = 0$ and $\Delta\eta = 0$ for example - Algorithm 1 ends up doing exactly the same thing as $\mathrm{QP}\left(\Delta x^0, \Delta\eta\right)$. Therefore, Algorithm 1 naturally provides nominal and emergency guidance in a unifying framework.

The optimal values μ^* and Δt_f^* returned by Algorithm 1 via z^* enable us to describe the guidance control law as a continuous time function. Indeed, interpreting $\bar{u}$ and u_μ as functions defined on $[0,1]$, the guidance law becomes

$$u^*(t) = \bar{u}\left(\frac{t}{\bar{t}_f + \Delta t_f^*}\right) + u_{\mu^*}\left(\frac{t}{\bar{t}_f + \Delta t_f^*}\right), \qquad \forall t \in [0, \bar{t}_f + \Delta t_f^*]. \tag{12}$$

Finally, the guidance trajectory can be expressed in terms of way-points x^k using the approximation of Equation (5). Indeed, if we want the way points to be expressed at some time instances (τ_k) - distributed on the normalized time interval $[0,1]$ - we can define the *augmented* state way-points $\tilde{x}^k$ such that

$$\tilde{x}^k := \tilde{x}[\tau_k] + \frac{\partial\tilde{x}}{\partial x^0}[\tau_k].\Delta x^0 + \frac{\partial\tilde{x}}{\partial\eta}[\tau_k].\Delta\eta + \frac{\partial\tilde{x}}{\partial\mu}[\tau_k].\mu^* + \frac{\partial\tilde{x}}{\partial x_{n+1}^0}[\tau_k].\Delta t_f^* \tag{13}$$

where we recall that $\tilde{x}$ embeds the state and the time-of-flight, i.e. $\tilde{x}^k = \left((x^k)^\top, \bar{t}_f + \Delta t_f^*\right)^\top$ where the *regular* state way-points are the points x^k sought for.

Remark 8. *As long as the widest negotiation problem is feasible - i.e.* LP_R *- then Algorithm 1 will necessarily return a solution. Indeed, by construction, the solution of each problem* LP_j *defines a feasible point for the next-to-be-solved problem* LP_{j-1}*, and the solution of* LP_1 *is feasible for* Refine*. Moreover, it can be proved that the solution* z^* *is unique (contrary to* p*), using that* $P \succ 0$.

Remark 9. *The map* $(\Delta x^0, \Delta\eta) \mapsto z^*(\Delta x^0, \Delta\eta)$ *defined by Algorithm 1 is Lipschitz-continuous. However, the proof of this result is out of scope for this paper. Basically, it relies on properties of Quadratic Programs such as the Lispchitz-continuity of the feasible set with respect to the constraint right-hand side.*[26] *This will be part of a future publication.*

NUMERICAL RESULTS

Let us now illustrate the behavior of Algorithm 1 from a numerical point of view.

Let us start by describing an arbitrary landing scenario defined by $(\bar{q}, \bar{\alpha})$, as shown in Figure 4-b and c, in plain **black**. On one hand, the engine flow is described by a smooth time law with a high-low-medium thrust structure. On the other hand, the incidence is also described by a smooth time law, that is positive and non-zero only for a third of the descent. This boils down to a trajectory that does one main turn and ends vertically, as shown in Figure 4-a. For the rest of this section, the states are normalized with respect to the reference initial condition $\bar{x}(0)$, such that all the initial states equal ± 1. The reference time-of-flight $\bar{t}_f$ has been arbitrarily set to 30 s. Also, as mentioned earlier, it must be recalled that the choice of negotiable parameters for this article is $p := (\Delta\alpha_{\max}, \Delta z^f)^\top$.

Now, let us check several aspects of Algorithm 1. First, a salient topic focuses on the quality of the outputs of Algorithm 1. This will be represented in terms of u^*, as described in Equation (12), and in terms of way-points x^k.

Then, since QP $(\Delta x^0, \Delta\eta)$, LP$_j$, Refine and Equation (13) are based on linearized data, it is possible that integrating the control law u^* in an open-loop fashion on the non-linear dynamics (1) does not result in a trajectory that lands at the proper place. However, we want to point-out how even a simple feedback law is sufficient to go from a near acceptable control law to a completely acceptable one. To this purpose, let us introduce the following feedback.

The engine flow law q^* is modified into $\hat{q}$ in order to take into account the vertical speed errors "$v_h(t) - \hat{v}_h(t)$", and saturated to enforce the mechanical limits, such that

$$\hat{q}(t) := \mathrm{Sat}_{q_{\min}}^{q_{\max}} \left(q^*(t) - K_{v_h}(v_h(t) - \hat{v}_h(t))\right). \tag{14}$$

where $K_{v_h} > 0$ is a small scalar. The same kind of feedback law is added to the incidence, taking into account deviations in horizontal speed *and* in horizontal position. This new incidence law $\hat{\alpha}$ is saturated at $\pm(\alpha_{\max} + \Delta\alpha^*_{\max})$, where $\Delta\alpha^*_{\max}$ is the optimally negotiated limit. Contrary to the latter, the feedback of Equation (14) only considers vertical *speed* errors and not *altitude* errors, for the speed tracking error accumulation mainly implies (small) time-of-flight changes - since it is implicitly defined by the condition $h = 0$ - which has little consequences in practice. All of this leads to a control law $\hat{u} = (\hat{q}, \hat{\alpha})^\top$, which tracks the states $\hat{x}$. Thus, another way to assess the quality of the results of Algorithm 1 is to integrate the non-linear dynamics (1) with this new control $\hat{u}$, as summarized in Algorithm 2.

Algorithm 2 Simulation procedure

```
Require: $\Delta x^0$ and $\Delta\eta$.
  Compute $z^*$ from Algorithm 1.
  Build $u^*$ using $z^*$ and compute the way-points $x^k$ from Equation (13).
  Build the function $\hat{x} = (\hat{h}, \hat{v}_h, \hat{z}, \hat{v}_z, \hat{m})$ defined on $[0, \bar{t}_f + \Delta t_f^*]$, that interpolates the points $x^k$.
  Build the function $\hat{u}$, using $\hat{x}$, as in Equation (14).
  Integrate $\dot{x} = f(x, \hat{u}, \bar{\eta} + \Delta\eta)$ from $x(0) = \bar{x}(0) + \Delta x^0$, until $h = 0$.
  return Values of $x$ and $\hat{u}$.
```

Variation of a single variable

To illustrate the basic principle of Algorithm 1, let us consider that only the value of the initial horizontal position changes, as shown in Figure 4. It allows us to highlight two important features.

On one hand, it clearly shows how the emergency modes - i.e. when either $\Delta\alpha_{\max}$ or Δz^f are non-zero - are successively triggered, providing continuously varying results with respect to the inputs. On the other hand, observe that the thin lines representing the feedback $(\hat{x}, \hat{u})$ almost completely blends with the control law u^* and the way-points x^k. It demonstrates how little effort is needed to remain on the way-point based trajectories.

As far as computation time is concerned, it was measured that, on average, once the matrices defining the problems LP$_j$ and REFINE have been stored in memory and for the choice of negotiable parameters of this article, executing Algorithm 1 takes several milliseconds using of-the-shelf solvers such as `cvxopt`.[27]

Variation of multiple variables

To illustrate many other behaviors of Algorithm 1, its results are displayed in Figure 5, in which the inputs are dispersed for many different values of the inputs.

DISCUSSIONS

Some side elements regarding the above-mentioned topics have been omitted for the sake of clarity and are commented below.

First, the choice of the reference trajectory has a handy influence on the performance of the trajectories returned by Algorithm 1. For example, computing a reference trajectory that is fuel-optimal will make the optimal correction near fuel-optimal by nature, even though the cost J does not directly relate to fuel-consumption. For further development on this topic, see e.g. Reference 28. Also, note that in general, optimal reference trajectories are likely to be bang-bang.[29–31]

Second, instead of the cost $J(z) = \frac{1}{2} z^\top P z$, it would have been possible to consider more general non-linear costs, such as Mayer-costs[20] for instance. In this case, the linearization needed by QP $\left(\Delta x^0, \Delta\eta\right)$ would have required to expand the Lagrangian of NLP $\left(\Delta x^0, \Delta\eta\right)$ to the second order, and in the remaining of the paper the term $\frac{1}{2} z^\top P z$ would have to be replaced by an expression of the shape

$$\frac{1}{2} z^\top P z + \left(Q_x \Delta x^0 + Q_\eta \Delta\eta\right)^\top z$$

for some matrices Q_x and Q_η depending only on the reference trajectory. For further details on how to do it, see previous work of the authors.[12] Considering this new cost would actually favor the corrections that are closed to the reference trajectory. However, it is important to note that *changing the cost function does not change anything in the above-mentioned feasibility analysis*, nor in the execution of the problems LP$_j$.

Finally, the way points computed using Equation 5 are a rather good approximation in practice, but are not exact. If it is required that the way-point be exactly dynamically feasible, then several options are possible. Among others, a simple way to solve it is to *project* the guidance trajectory that is not exactly feasible onto the set of feasible trajectories by integrating the optimal control law on the non-linear dynamic model (1), while using an extra feedback term tracking the guidance trajectory. This approach will generate a new guidance trajectory that will land *close* to the desired landing site, but most importantly that will be exactly dynamically feasible. For further details on this kind of approach see e.g. Reference 32.

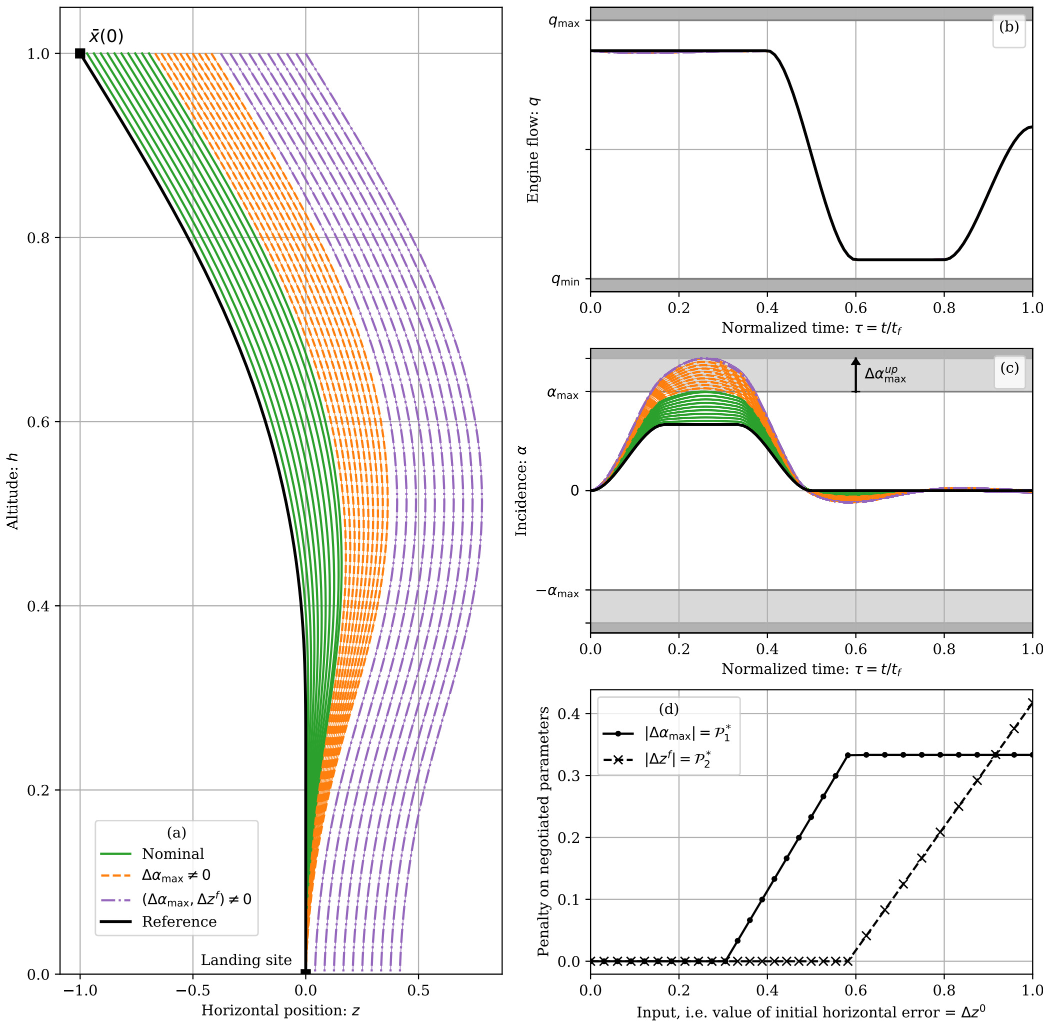

Figure 4. Computing guidance with Algorithm 1, for several values of the initial position error Δz^0. The trajectories in Figure (a) are computed using Equation (5). The control laws for Figures (b) and (c) rely on Equation (12). Finally, Figure (d) illustrates the continuity of the penalties $\mathcal{P}_i^*$ with respect to the inputs.

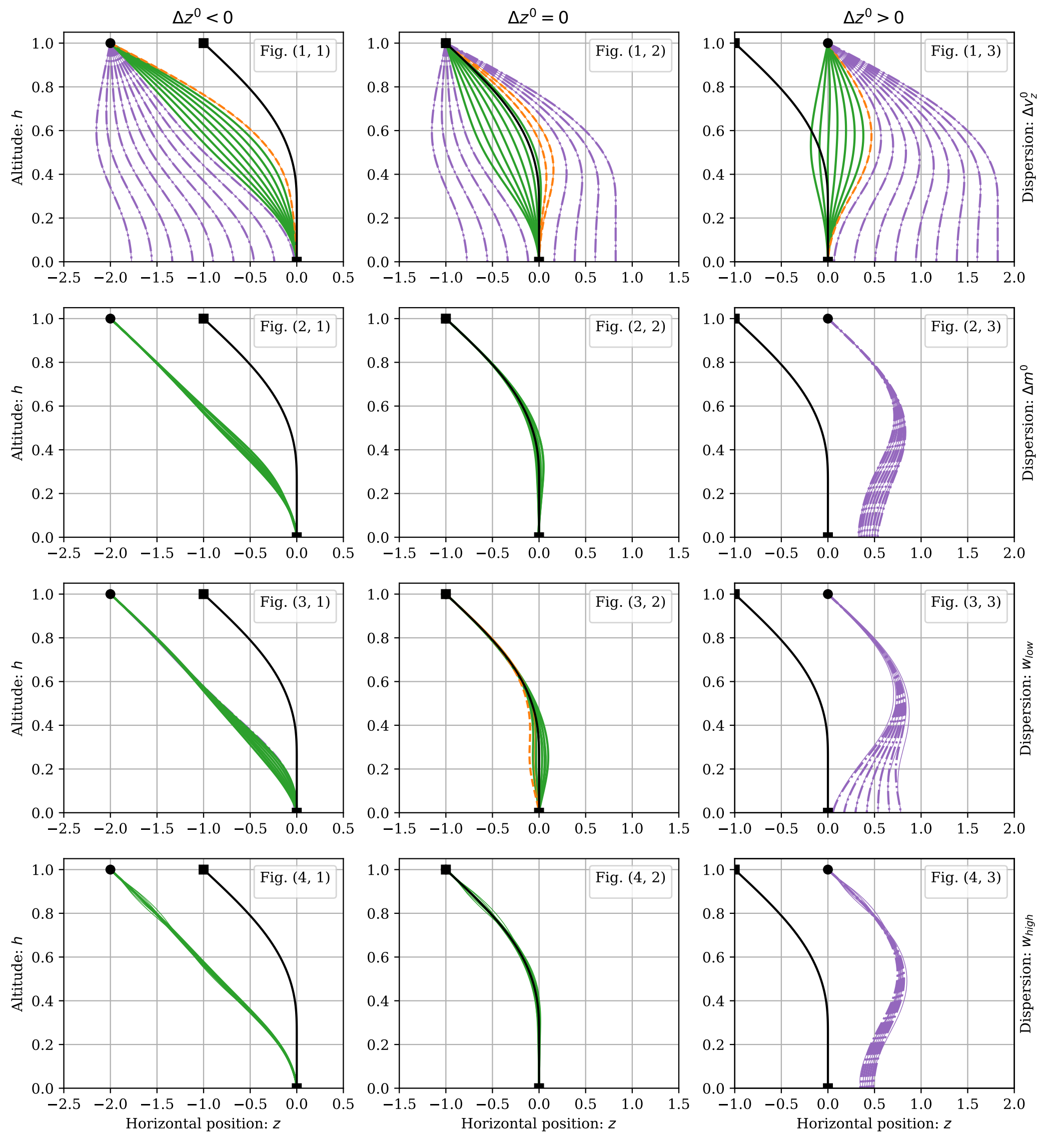

Figure 5. Showing guidance provided by Algorithm 1, for various input values. The legend is the same as in Figure 4-a. As in Figure 4, both way-point based trajectories, and trajectories computed using Algorithm 2 are displayed. Each column corresponds to a different value of Δz^0. Each row corresponds to the dispersion of a certain variable, known to the algorithm, respectively the initial horizontal speed Δv_z^0, the initial mass Δm^0, and the winds w_1 and w_2 at low and high altitudes. The values of these variables have been dispersed over significantly wide intervals in order to trigger emergency modes. It is noteworthy that some input combinations tend to have a naturally *constructive* or *destructive* behavior on the PDG problem. For example in Fig. (1, 3), Δz^0 and Δv_z^0 have a constructive behavior for the left-most trajectories (descent starts horizontaly closer to the landing site but slower), and a destructive behavior for the right-most trajectories (which require emergency guidance).

CONCLUSION

In this paper, we exposed a method to provide nominal and emergency guidance for atmospheric powered descent guidance of a high-thrust reusable launcher. The method boils down to a sequential minimization of the amplitude of negotiable parameters, enforcing a prescribed hierarchy between these parameters. Algorithm 1 gathers nominal and emergency guidance methods together thanks to a unifying framework. Numerical simulations demonstrated a high numerical efficiency.

Future work will focus on extending this approach to more complex scenarios - especially with 3D rocket models, non-instantaneous engine dynamics and a wider list of negotiable parameters - and exposing theoretical guarantees of the behavior of Algorithm 1.

APPENDIX

Flow differentiation

The sensitivity computations used through this paper are standard material in the literature, though different ways of writing it exist. The formalism chosen in this paper, and more precisely the way the control u is considered is specified below. Material can be found in Section 3.2 of Reference 33 for a rigorous definition in Banach spaces, and more applied methods can be found in well-known standard text books, such as Appendix A4 in Reference 19, Section 2.4 in Reference 34 or Equation 4.13 in Reference 35.

Consider a smooth dynamic function f depending on a state x, a control u and a parameter η. Considering a function $u : [0,1] \mapsto \mathbb{R}^m$ and some $t \in [0,1]$, the flow $\Phi_f\left(t, x^0, \eta; u\right)$ is defined based on the following Initial Value Problem (IVP)

$$\Phi_f\left(t, x^0, \eta; u\right) := x(t) \Leftrightarrow \begin{cases} \dot{x}(s) = f(x(s), u(s), \eta), & \forall s \in [0,t] \\ x(0) = x^0 \end{cases} \tag{15}$$

Consider a parametric control $t \mapsto u_\mu(t)$, continuously differentiable in t and μ. The following expansion holds

$$\begin{aligned}\Phi_f\left(t, x^0 + \Delta x^0, \eta + \Delta\eta; u_\mu\right) = \; & \Phi_f\left(t, x^0, \eta; u_0\right) + A(t)\Delta x^0 + B(t)\Delta\eta + C(t)\mu \\ & + \varepsilon(\Delta x^0, \Delta\eta, \mu)\end{aligned}$$

where ε is a function such that $\varepsilon(\star)/\|\star\|_2$ tends to zero when $\star = (\Delta x^0, \Delta\eta^0, \mu)$ tends to zero, and where A, B and C are matrix valued functions defined by the following IVPs

$$\begin{aligned}
\dot{A}(t) &= \frac{\partial f}{\partial x}(x(t), u_0(t), \eta) \cdot A(t), \qquad A(0) = \mathbb{I}_n, \\
\dot{B}(t) &= \frac{\partial f}{\partial x}(x(t), u_0(t), \eta) \cdot B(t) + \frac{\partial f}{\partial \eta}(x(t), u_0(t), \eta), \qquad B(0) = \mathbb{O}_{n \times n_\eta}, \\
\dot{C}(t) &= \frac{\partial f}{\partial x}(x(t), u_0(t), \eta) \cdot C(t) + \frac{\partial f}{\partial u}(x(t), u_0(t), \eta) \cdot \left.\frac{\partial u_\mu}{\partial \mu}\right|_{\mu=0}(t), \qquad C(0) = \mathbb{O}_{n \times n_\mu}.
\end{aligned}$$

REFERENCES

[1] B. A. Steinfeldt, M. J. Grant, D. A. Matz, R. D. Braun, and G. H. Barton, “Guidance, Navigation, and Control System Performance Trades for Mars Pinpoint Landing,” *Journal of Spacecraft and Rockets*, Vol. 47, Jan. 2010, pp. 188–198, 10.2514/1.45779.

[2] A. R. Klumpp, “Apollo Lunar Descent Guidance,” *Automatica*, 1974, pp. 133–146, 10.1016/0005-1098(74)90019-3.

[3] D. Malyuta, T. P. Reynolds, M. Szmuk, T. Lew, R. Bonalli, M. Pavone, and B. Açikmeşe, “Convex Optimization for Trajectory Generation,” *arXiv:2106.09125 [cs, eess, math]*, June 2021.

[4] B. Açikmeşe and S. R. Ploen, “Convex Programming Approach to Powered Descent Guidance for Mars Landing,” *Journal of Guidance, Control, and Dynamics*, Vol. 30, Sept. 2007, pp. 1353–1366, 10.2514/1.27553.

[5] M. Szmuk and B. Açikmeşe, “Successive Convexification for 6-DoF Mars Rocket Powered Landing with Free-Final-Time,” *2018 AIAA Guidance, Navigation, and Control Conference*, American Institute of Aeronautics and Astronautics, 2018, 10.2514/6.2018-0617.

[6] B. Açıkmeşe and L. Blackmore, “Lossless convexification of a class of optimal control problems with non-convex control constraints,” *Automatica*, 2011, pp. 341–347, 10.1016/j.automatica.2010.10.037.

[7] L. Blackmore, B. Açıkmeşe, and J. M. Carson, “Lossless convexification of control constraints for a class of nonlinear optimal control problems,” *Systems & Control Letters*, Vol. 61, Aug. 2012, pp. 863–870, 10.1016/j.sysconle.2012.04.010.

[8] B. Açikmeşe, J. Casoliva, J. M. Carson, and L. Blackmore, “G-FOLD: A Real-Time Implementable Fuel Optimal Large Divert Guidance Algorithm for Planetary Pinpoint Landing,” 2012.

[9] M. Sagliano, A. Heidecker, J. M. Hernández, S. Farì, M. Schlotterer, S. Woicke, D. Seelbinder, and E. Dumont, “Onboard Guidance for Reusable Rockets: Aerodynamic Descent and Powered Landing,” Jan. 2021, p. 35.

[10] L. Ma, K. Wang, Z. Shao, Z. Song, and L. T. Biegler, “Trajectory Optimization for Planetary Multi-Point Powered Landing,” *IFAC-PapersOnLine*, Vol. 50, July 2017, pp. 8291–8296, 10.1016/j.ifacol.2017.08.1404.

[11] L. Ma, K. Wang, Z. Xu, Z. Shao, Z. Song, and L. T. Biegler, “Multi-point powered descent guidance based on optimal sensitivity,” *Aerospace Science and Technology*, Vol. 86, Mar. 2019, pp. 465–477, 10.1016/j.ast.2019.01.028.

[12] H. Ménou, E. Bourgeois, and N. Petit, “Sensitivity Analysis for Powered Descent Guidance: Overcoming degeneracy,” *2022 European Control Conference*, London, 2022. Accepted for publication.

[13] Z.-y. Song, C. Wang, S. Theil, D. Seelbinder, M. Sagliano, X.-f. Liu, and Z.-j. Shao, “Survey of autonomous guidance methods for powered planetary landing,” *Frontiers of Information Technology & Electronic Engineering*, Vol. 21, May 2020, pp. 652–674, 10.1631/FITEE.1900458.

[14] J. Hanson, D. Coughlin, G. Dukeman, J. Mulqueen, and J. McCarter, “Ascent, transition, entry, and abort guidance algorithm design for the X-33 vehicle,” *Guidance, Navigation, and Control Conference and Exhibit*, Boston,MA,U.S.A., American Institute of Aeronautics and Astronautics, Aug. 1998, 10.2514/6.1998-4409.

[15] P. Lu and S. A. Sandoval, “Abort Guidance during Powered Descent for Crewed Lunar Missions,” *AIAA Scitech 2021 Forum*, Virtual Event, American Institute of Aeronautics and Astronautics, Jan. 2021, 10.2514/6.2021-0505.

[16] S. Nonaka, H. Nishida, H. Kato, H. Ogawa, and Y. Inatani, “Vertical Landing Aerodynamics of Reusable Rocket Vehicle,” *Transactions Of The Japan Society For Aeronautical And Space Sciences, Aerospace Technology Japan*, Vol. 10, No. 0, 2012, pp. 1–4, 10.2322/tastj.10.1.

[17] A. Marwege, J. Riehmer, J. Klevanski, A. Gülhan, T. Ecker, B. Reimann, and E. Dumont, “First Wind Tunnel Data of CALLISTO Reusable VTVL Launcher First Stage Demonstrator,” 2019, p. 15.

[18] D. Kraft, “On Converting Optimal Control Problems into Nonlinear Programming Problems,” *Schittkowski K. (eds) Computational Mathematical Programming*, Vol. 15 of *NATO ASI Series (Series F: Computer and Systems Sciences)*, 1985.

[19] A. E. Bryson and Y.-C. Ho, *Applied optimal control: optimization, estimation and control*. CRC Press, 1975.

[20] J. T. Betts, *Practical methods for optimal control and estimation using nonlinear programming*. Advances in Design and Control, SIAM, 2001.

[21] A. V. Fiacco, *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming*, Vol. 165 of *Mathematics in Science and Engineering*. Elsevier, 1983, 10.1016/S0076-5392(08)X6041-2.

[22] K. Jittorntrum, “Solution point differentiability without strict complementarity in nonlinear programming,” *Sensitivity, Stability and Parametric Analysis*, Vol. 21, pp. 127–138, Berlin, Heidelberg: Springer Berlin Heidelberg, 1984.

[23] J. F. Bonnans and A. Shapiro, *Perturbation Analysis of Optimization Problems*. New York, NY: Springer New York, 2000, 10.1007/978-1-4612-1394-9.

[24] L. Blackmore, B. Açikmeşe, and D. P. Scharf, “Minimum-Landing-Error Powered-Descent Guidance for Mars Landing Using Convex Optimization,” *Journal of Guidance, Control, and Dynamics*, 2010, pp. 1161–1171.

[25] J. W. Chinneck, *Feasibility and Infeasibility in Optimization: Algorithms and Computational Methods*. Springer Science & Business Media, 2007.

[26] O. L. Mangasarian and T.-H. Shiau, “Lipschitz Continuity of Solutions of Linear Inequalities, Programs and Complementarity Problems,” *SIAM Journal on Control and Optimization*, Vol. 25, May 1987, pp. 583–595, 10.1137/0325033.

[27] M. Andersen, J. Dahl, and L. Vandenberghe, “CVXOPT: Convex Optimization,” Aug. 2020.

[28] S. A. Deshpande, D. Bonvin, and B. Chachuat, “Directional Input Adaptation in Parametric Optimal Control Problems,” *SIAM Journal on Control and Optimization*, Vol. 50, Jan. 2012, pp. 1995–2024, 10.1137/110820646.

[29] J. Meditch, “On the problem of optimal thrust programming for a lunar soft landing,” *IEEE Transactions on Automatic Control*, Vol. 9, Oct. 1964, pp. 477–484, 10.1109/TAC.1964.1105758.

[30] H. Ménou, E. Bourgeois, and N. Petit, “Fuel-optimal program for atmospheric vertical powered landing,” *60th Conference on Decision and Control*, 2021.

[31] C. Leparoux, B. Hérissé, and F. Jean, “Structure of optimal control for planetary landing with control and state constraints,” *arXiv:2204.06794 [math]*, Apr. 2022.

[32] I. Notarnicola, F. A. Bayer, G. Notarstefano, and F. Allgower, “Final-State Constrained Optimal Control via a Projection Operator Approach,” *arXiv:1703.08356 [cs]*, Mar. 2017.

[33] J. F. Bonnans, *Course on Optimal Control, Part I: the Pontryagin approach*. SOD311 Ensta Paris Tech, Aug. 2019.

[34] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 2 sub ed., 1995.

[35] E. D. Sontag, *Mathematical Control Theory*, Vol. 6 of *Texts in Applied Mathematics*. New York, NY: Springer New York, 1998, 10.1007/978-1-4612-0577-7.