Available online at www.sciencedirect.com

ScienceDirect

Advances in Space Research xxx (xxxx) xxx

ADVANCES IN SPACE RESEARCH (a COSPAR publication)

www.elsevier.com/locate/asr

A study on an optimal analytical and an accurate algorithmic landing techniques

José David Gutierrez de Alba [a,*], Eva Tresaco [a], Daniele Mortari [b]

[a] University of Zaragoza, C. de Pedro Cerbuna, 12, Zaragoza 50009, Zaragoza, Spain
[b] Texas A&M University, 745 H.R. Bright Building, College Station 77843, TX, United States

Received 16 October 2024; received in revised form 13 May 2025; accepted 6 July 2025

Abstract

Landing trajectory optimization is a fundamental challenge in aerospace engineering, traditionally approached by optimal control problem formulation. In practice, for Earth landing, the vehicle is subject to strongly non-linear perturbations, such as drag, wind, and gravity. Indirect control methods cannot deal with these problems, leading to large errors, while direct methods require high computation times when perturbations are not continuous. Thus, in this work, we propose a hybrid algorithm to approximate the solution of nonlinear optimization problems, based on the iteration of the indirect control analytical solution of the non-autonomous linear problem. The algorithm's performance has been studied for the problem of landing rockets on the Earth's surface and compared with a standard boundary value problem solver. The results show that the proposed algorithm tolerates strong discontinuous perturbations, with very small landing error and computation time.

Keywords: Indirect optimal control; trajectory optimization; Optimal landing; Strong perturbations; Drag; Gravity

1. Introduction

The literature on trajectory optimization problems is rich, full of different proposed numerical techniques to solve them. Generally, optimal control problems are solved using direct or indirect methods (Rösmann et al., 2013; Chai et al., 2019; Moyer and Pinkham, 1964). Direct methods transform the problem into a Non-Linear Programming (NLP) problem by discretizing the continuous motion states and control constraints, allowing numerical algorithms to find local minima. However, NLP problems are considered computationally challenging, and the time to find the optimal solution is unpredictable (Lavezzi et al., 2023). To address this, recent works focus on reformulating these problems into convex optimization problems, which guarantee global convergence in polynomial time (Acikmese and Ploen, 2007; Blackmore et al., 2010). Convex optimization has shown success in various aerospace applications, such as planetary landing, atmospheric guidance, and low-thrust propulsion. In Liu et al. (2015) a method was introduced to solve nonsmooth convex optimization problems. However, this process is computationally intensive and requires careful tuning to ensure convergence and accuracy. The application of optimal control theory to derive the first-order necessary conditions for optimality yields a Two-Point Boundary Value Problem (TPBVP) that must be solved to find state and costate.

In the frame of the indirect optimization approach, to minimize energy in a landing scenario and considering the need to have fast algorithms to compute accurate optimal landing trajectories, a novel approach was introduced using a method based on functional interpolation, intro-

* Corresponding author.
E-mail addresses: jgutierrez@unizar.es (J.D. Gutierrez de Alba), etresaco@unizar.es (E. Tresaco), mortari@tamu.edu (D. Mortari).

https://doi.org/10.1016/j.asr.2025.07.017
0273-1177/

Please cite this article as: J. D. Gutierrez de Alba, E. Tresaco and D. Mortari, A study on an optimal analytical and an accurate algorithmic landing techniques, Advances in Space Research, https://doi.org/10.1016/j.asr.2025.07.017

duced as the Theory of Functional Connections (TFC) by Mortari (2017). TFC has been applied to obtain fast and accurate least-squares solutions for the TPBVP that arise in energy-optimal control and guidance problems in no-atmosphere scenarios (Furfaro and Mortari, 2020; Johnston et al., 2020; D'Ambrosio et al., 2022). The performance of this method has been compared to traditional solvers like MATLAB's `bvp4c` and `Chebfun` Toolbox. In Ref. Johnston et al. (2020) the problem was then solved for the fuel-minimum powered descent guidance problem on large planetary bodies, still with constant gravity and no atmosphere. The solution obtained was compared to GPOPS-II Patterson and Rao (2014), with computational time gain (more than 30 times faster). Nevertheless, when studying indirect-optimal landing problems with strong perturbations (such as drag and variable gravity), TFC's reliance on the nonlinear least-squares method increases the overall computational burden, becoming computationally challenging. On the other hand, D'Souza (1997) introduced an indirect optimal guidance law for planetary landing using analytical methods, solving the TPBVP with constant external forces. This approach provides the fastest computation of the solution, yet it is not suitable in non-constant/nonlinear perturbed problems. Therefore, in the range of existing algorithms for solving optimal control problems, there is a trade-off between computation time and accurate representation of the perturbations. In this work, by following D'Souza's method, we extend the analytical solution to time-dependent linear perturbations. Then, to overcome the non-linearity of the perturbations in real scenarios, a hybrid trajectory optimization algorithm is proposed, which iterates the indirect control analytical solution to minimize the error in the trajectory.

This article starts by briefly summarizing the optimal indirect landing problem under general nonautonomous, nonlinear perturbations. The problem is then studied from a mathematical point of view by analyzing the solutions in autonomous and non-autonomous linear systems. Afterwards, the iterative algorithm to approach the solution of the general problem is detailed. The effectiveness of the algorithm is then tested against MATLAB's `bvp4c` solver in an Earth landing scenario, considering perturbations such as variable gravity, drag, and discontinuous wind gusts.

2. Analytical optimal landing control

Landing on a celestial body is a two-point trajectory problem subject to the following boundary conditions,

$$\begin{cases} \boldsymbol{r}(0) = \boldsymbol{r}_0 \\ \boldsymbol{r}(\Delta T) = \boldsymbol{r}_f \end{cases} \quad \text{and} \quad \begin{cases} \boldsymbol{v}(0) = \boldsymbol{v}_0 \\ \boldsymbol{v}(\Delta T) = \boldsymbol{v}_f \end{cases} \tag{1}$$

where $\boldsymbol{r}_0$ and $\boldsymbol{v}_0$ are the initial position and velocity and $\boldsymbol{r}_f$ and $\boldsymbol{v}_f$ the final position and velocity that are reached in the ΔT time interval. The dynamic model for the general case is,

$$\begin{cases} \dot{\boldsymbol{r}}(t) = \boldsymbol{v}(t) \\ \dot{\boldsymbol{v}}(t) = \boldsymbol{f}(t, \boldsymbol{r}, \boldsymbol{v}) + \boldsymbol{a}_c(t) \end{cases} \tag{2}$$

where $\boldsymbol{r}, \boldsymbol{v}$ are position and velocity vectors respectively, $\boldsymbol{f}$ is the vector representing all the external accelerations and $\boldsymbol{a}_c$ is the control vector applied to reach the final conditions in specified time. The optimal landing problem consists of finding a control $\boldsymbol{a}_c$ that solves the landing problem while minimizing an objective functional $\mathscr{J}$. For simplification purposes, an upper bound of the control acceleration is not considered in this work. This functional depends on what we consider optimal control. As an example, the optimal landing problem proposed by D'Souza (1997) and adopted in subsequent articles Furfaro and Mortari (2020); Gaudet et al. (2020); Szmuk et al. (2020); Blackmore et al. (2010); Longuski et al. (2014) involves minimizing the optimality index,

$$\mathscr{J}_{\mathrm{TE}}(\Delta T) = \Gamma \Delta T + \frac{1}{2}\int_0^{\Delta T} \boldsymbol{a}_c^T \boldsymbol{a}_c \, \mathrm{d}t \tag{3}$$

where Γ is a user-defined positive constant weighting the landing time over the energy cost. For $\Gamma > 0$, the solution is called the "time-energy optimal". In this study, optimality is defined as simply minimizing the energy cost,

$$\mathscr{J}(\Delta T) = \frac{1}{2}\int_0^{\Delta T} \boldsymbol{a}_c^T \boldsymbol{a}_c \, \mathrm{d}t \tag{4}$$

The reasons are essentially three:

1. The Γ term appearing in the cost index cost makes sense when it multiplies a competing cost term as, for instance, the final velocity in a fuel-bounded anti-sat problem.
2. Another reason to set $\Gamma > 0$ in Eq. (2) is to bound the landing time. In this work, for the sake of simplicity, we disregard the landing time from our optimization problem.
3. Lastly, $\mathscr{J}(\Delta T)$ represents the trajectory *cost* (not an index) with a clear physical meaning.

2.1. Linear autonomous problem

In this section, the optimal landing problem for the case where the external acceleration vector is constant is solved. Under this assumption, Eq. (2) becomes the system

$$\begin{cases} \dot{\boldsymbol{r}}(t) = \boldsymbol{v}(t) \\ \dot{\boldsymbol{v}}(t) = \boldsymbol{f} + \boldsymbol{a}_c(t) \end{cases} \tag{5}$$

which is linear and autonomous. The optimal control problem is solved by applying the Pontryagin Minimum Principle (Pontryagin (1987)). As in Johnston et al. (2020), the Hamiltonian of the system is,

$$\mathscr{H} = \frac{1}{2}\boldsymbol{a}_c^T \boldsymbol{a}_c + \lambda_r^T \boldsymbol{v} + \lambda_v^T (\boldsymbol{f} + \boldsymbol{a}_c), \tag{6}$$

where λ_p and λ_v are the unknown position and velocity costates vectors. The optimal control is obtained by (first variation conditions),

$$\frac{\partial \mathscr{H}}{\partial \boldsymbol{a}_c} = \boldsymbol{a}_c + \lambda_v = \boldsymbol{0} \qquad \rightarrow \qquad \boldsymbol{a}_c = -\lambda_v \tag{7}$$

and the optimal problem becomes solving the two-point boundary values problem of the following systems of state and costate equations,

$$\begin{cases} \dot{\boldsymbol{r}} = \frac{\partial \mathscr{H}}{\partial \lambda_r} = \boldsymbol{v} \\ \dot{\boldsymbol{v}} = \frac{\partial \mathscr{H}}{\partial \lambda_v} = \boldsymbol{f} + \boldsymbol{a}_c \end{cases} \quad \text{and} \quad \begin{cases} \dot{\lambda}_r = -\frac{\partial \mathscr{H}}{\partial \boldsymbol{r}} = \boldsymbol{0} \\ \dot{\lambda}_v = -\frac{\partial \mathscr{H}}{\partial \boldsymbol{v}} = -\lambda_r. \end{cases} \tag{8}$$

Using $\boldsymbol{a}_c = -\lambda_v$ and indicating $\lambda = \lambda_v$, the four equations given in Eq. (8) can be written as,

$$\begin{cases} \ddot{\boldsymbol{r}} + \lambda = \boldsymbol{f} \\ \ddot{\lambda} = \boldsymbol{0}. \end{cases} \tag{9}$$

Now, integrating the second equation, we obtain $\lambda = \boldsymbol{a}_0 + \boldsymbol{a}_1 t$, where $\boldsymbol{a}_0, \boldsymbol{a}_1 \in \mathbb{R}^3$. Substituting into the first equation gives $\ddot{\boldsymbol{r}}$ as a linear polynomial in time. Therefore,

$$\begin{cases} \lambda(t) = \boldsymbol{a}_0 + \boldsymbol{a}_1 t \\ \boldsymbol{r}(t) = \boldsymbol{b}_0 + \boldsymbol{b}_1 t + \boldsymbol{b}_2 t^2 + \boldsymbol{b}_3 t^3 \\ \boldsymbol{v}(t) = \boldsymbol{b}_1 + 2\boldsymbol{b}_2 t + 3\boldsymbol{b}_3 t^2 \end{cases} \tag{10}$$

The initial conditions, $\boldsymbol{r}(0) = \boldsymbol{r}_0$ and $\dot{\boldsymbol{r}}(0) = \boldsymbol{v}_0$, gives,

$$\boldsymbol{b}_0 = \boldsymbol{r}_0 \qquad \text{and} \qquad \boldsymbol{b}_1 = \boldsymbol{v}_0. \tag{11}$$

Substituting in $\boldsymbol{r}(t)$ and imposing the final conditions, $\boldsymbol{r}(\Delta T) = \boldsymbol{r}_f$ and $\dot{\boldsymbol{r}}(\Delta T) = \boldsymbol{v}_f$, we obtain the expressions of $\boldsymbol{b}_2$ and $\boldsymbol{b}_3$

$$\begin{cases} \boldsymbol{b}_2 = -\frac{3(\boldsymbol{r}_0 - \boldsymbol{r}_f)}{\Delta T^2} - \frac{2\boldsymbol{v}_0 + \boldsymbol{v}_f}{\Delta T} \\ \boldsymbol{b}_3 = \frac{2(\boldsymbol{r}_0 - \boldsymbol{r}_f)}{\Delta T^3} + \frac{\boldsymbol{v}_0 + \boldsymbol{v}_f}{\Delta T^2} \end{cases} \tag{12}$$

Finally, substituting these expressions in Eq. (9), we obtain,

$$2\boldsymbol{b}_2 + 6\boldsymbol{b}_3 t + \boldsymbol{a}_0 + \boldsymbol{a}_1 t = \boldsymbol{f}, \tag{13}$$

which is an expression allowing us to obtain $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$,

$$\boldsymbol{a}_0 = \boldsymbol{f} - 2\boldsymbol{b}_2 \qquad \text{and} \qquad \boldsymbol{a}_1 = -6\boldsymbol{b}_3. \tag{14}$$

Then, the expression of the control is,

$$\boldsymbol{a}_c = -\lambda = 2\boldsymbol{b}_2 - \boldsymbol{f} + 6\boldsymbol{b}_3 t \tag{15}$$

and, since $\boldsymbol{a}_c = -\lambda$ and $\lambda_r = -\dot{\lambda} = 6\boldsymbol{b}_3$, the Hamiltonian becomes,

$$\mathscr{H}(\Delta T) = -\boldsymbol{a}_1^T \boldsymbol{v} + \lambda^T \boldsymbol{f} - \frac{1}{2}\lambda^T \lambda \tag{16}$$

It is straightforward to prove that all the coefficients of the time-varying terms of the Hamiltonian, given in Eq. (16), are zero, making the Hamiltonian constant. This validates the optimal solution for the selected ΔT. Therefore, the expression of the Hamiltonian becomes,

$$\mathscr{H}(\Delta T) = \boldsymbol{a}_0^T \boldsymbol{f} - \frac{1}{2}\boldsymbol{a}_0^T \boldsymbol{a}_0 - \boldsymbol{a}_1^T \boldsymbol{v}_0 = \text{constant}. \tag{17}$$

Now, the specific value of ΔT that minimizes the cost function $\mathscr{J}(\Delta T)$ can be obtained by finding the value of ΔT that makes the derivative of $\mathscr{J}(\Delta T)$ equal to zero. Obtaining the zeroes of $\mathscr{J}\prime(\Delta T)$ is equivalent to solving the following depressed quartic equation (no cubic term):

$$\Delta T^4 - \bar{a}\Delta T^2 - \bar{b}\Delta T - \bar{c} = 0, \tag{18}$$

where

$$\bar{a} = \frac{4}{\|\boldsymbol{f}\|^2}\left(\|\boldsymbol{v}_0\|^2 + \|\boldsymbol{v}_f\|^2 + \boldsymbol{v}_0^T \boldsymbol{v}_f\right),$$
$$\bar{b} = \frac{24}{\|\boldsymbol{f}\|^2}\left(\boldsymbol{r}_0 - \boldsymbol{r}_f\right)^T\left(\boldsymbol{v}_0 + \boldsymbol{v}_f\right),$$
$$\bar{c} = \frac{36}{\|\boldsymbol{f}\|^2}\|\boldsymbol{r}_0 - \boldsymbol{r}_f\|^2,$$

with $\|\cdot\|$ denoting the Euclidean norm. This equation is guaranteed to have one positive solution but can have more than one. In that case, we only need to check which root gives the minimum value of $\mathscr{J}$.

2.2. Linear non-autonomous problem

Suppose now that the forcing vector is not constant and the term $\boldsymbol{f}(t)$ has explicit time dependency (non-autonomous linear system),

$$\begin{cases} \dot{\boldsymbol{r}}(t) = \boldsymbol{v}(t) \\ \dot{\boldsymbol{v}}(t) = \boldsymbol{f}(t) + \boldsymbol{a}_c(t) \end{cases} \tag{19}$$

To apply the Pontryagin Minimum Principle we need to transform Eq. (19) into an autonomous system of the form

$$\begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{v} \\ \dot{\boldsymbol{v}} = \boldsymbol{f}(\tau) + \boldsymbol{a}_c \\ \dot{\tau} = 1 \end{cases} \tag{20}$$

where τ is the variable corresponding to time. Now the Hamiltonian becomes

$$\mathscr{H} = \frac{1}{2}\boldsymbol{a}_c^T \boldsymbol{a}_c + \lambda_{\boldsymbol{r}}^T \boldsymbol{v} + \lambda_{\boldsymbol{v}}^T(\boldsymbol{f}(\tau) + \boldsymbol{a}_c) + \lambda_\tau, \tag{21}$$

where $\lambda_{\boldsymbol{r}}, \lambda_{\boldsymbol{v}} \in \mathbb{R}^3$ and $\lambda_\tau \in \mathbb{R}$ are the costates variables. Since the optimal control is also given by $\boldsymbol{a}_c = -\lambda_{\boldsymbol{v}}$, the two-point boundary values problem to solve is,

$$\begin{cases} \dot{\boldsymbol{r}} = \frac{\partial \mathscr{H}}{\partial \lambda_{\boldsymbol{r}}} = \boldsymbol{v} \\ \dot{\boldsymbol{v}} = \frac{\partial \mathscr{H}}{\partial \lambda_{\boldsymbol{v}}} = \boldsymbol{f}(\tau) - \lambda_{\boldsymbol{v}} \\ \dot{\tau} = \frac{\partial \mathscr{H}}{\partial \lambda_\tau} = 1 \end{cases} \quad \text{and} \quad \begin{cases} \dot{\lambda}_{\boldsymbol{r}} = -\frac{\partial \mathscr{H}}{\partial \boldsymbol{r}} = \boldsymbol{0} \\ \dot{\lambda}_{\boldsymbol{v}} = -\frac{\partial \mathscr{H}}{\partial \boldsymbol{v}} = -\lambda_{\boldsymbol{r}} \\ \dot{\lambda}_\tau = -\frac{\partial \mathscr{H}}{\partial \tau} = -\boldsymbol{f}\prime(\tau)^T \lambda_{\boldsymbol{v}} \end{cases} \tag{22}$$

This problem is simplified in the following system of 2nd order ODEs and a quadrature given by

$$\begin{cases} \ddot{\boldsymbol{r}} + \lambda = \boldsymbol{f}(t), \\ \ddot{\lambda} = \boldsymbol{0}, \\ \lambda_\tau = -\int_0^t \boldsymbol{f}'(s)^T \lambda \, \mathrm{d}s, \end{cases} \tag{23}$$

where $\lambda \equiv \lambda_{\boldsymbol{v}}$ and s is a dummy variable. Again, the second equation yields $\lambda = \boldsymbol{a}_0 + \boldsymbol{a}_1 t$. Now we can integrate the first equation and use that $\dot{\boldsymbol{r}}(0) = \boldsymbol{v}_0$ to obtain

$$\dot{\boldsymbol{r}}(t) = \boldsymbol{v}_0 + \int_0^t \boldsymbol{f}(s)\,\mathrm{d}s - \boldsymbol{a}_0 t - \frac{1}{2}\boldsymbol{a}_1 t^2. \tag{24}$$

We introduce the dummy variable u and integrate once more to get the position,

$$\boldsymbol{r}(t) = \boldsymbol{r}_0 + \boldsymbol{v}_0 t + \int_0^t \int_0^s \boldsymbol{f}(u)\, \mathrm{d}u\, \mathrm{d}s - \frac{1}{2}\boldsymbol{a}_0 t^2 - \frac{1}{6}\boldsymbol{a}_1 t^3. \quad (25)$$

We can apply the final conditions, $\boldsymbol{r}(\Delta T) = \boldsymbol{r}_f$ and $\dot{\boldsymbol{r}}(\Delta T) = \boldsymbol{v}_f$, to solve for $\boldsymbol{a}_0$ and $\boldsymbol{a}_1$, obtaining

$$\begin{aligned}\boldsymbol{a}_0 &= \tfrac{1}{\Delta T^2}\left[6\left(\boldsymbol{r}_0 - \boldsymbol{r}_f + \int_0^{\Delta T}\int_0^s \boldsymbol{f}(u)\,\mathrm{d}u\,\mathrm{d}s\right) + 2\Delta T\left(2\boldsymbol{v}_0 + \boldsymbol{v}_f - \int_0^{\Delta T}\boldsymbol{f}(s)\,\mathrm{d}s\right)\right]\\ \boldsymbol{a}_1 &= \tfrac{-6}{\Delta T^3}\left[2\left(\boldsymbol{r}_0 - \boldsymbol{r}_f + \int_0^{\Delta T}\int_0^s \boldsymbol{f}(u)\,\mathrm{d}u\,\mathrm{d}s\right) + \Delta T\left(\boldsymbol{v}_0 + \boldsymbol{v}_f - \int_0^{\Delta T}\boldsymbol{f}(s)\,\mathrm{d}s\right)\right].\end{aligned} \quad (26)$$

Finally, ΔT, which gives optimality to the control, is obtained using the zeros of $\mathscr{J}\prime(\Delta T)$. In this case, $\mathscr{J}\prime$ has the expression:

$$\begin{aligned}\mathscr{J}'(\Delta T) \;=\; & \tfrac{1}{\Delta T^4} 2\left(-\|\boldsymbol{r}_d\|^2 - 6\Delta T \boldsymbol{r}_d^T \boldsymbol{v}_s - \Delta T^2\left(\|\boldsymbol{v}_s\|^2 - \boldsymbol{v}_0^T\boldsymbol{v}_f\right) + \right.\\ & \Delta T \boldsymbol{F}(\Delta T)^T\left(12\boldsymbol{r}_d + 4\Delta T(\boldsymbol{v}_0 + 5\boldsymbol{v}_f T - 4\boldsymbol{F}(\Delta T))\right) -\\ & 6\Phi(\Delta T)^T\left(3\boldsymbol{r}_d + \Delta T(\boldsymbol{v}_s - 2\boldsymbol{F}(\Delta T))\right) - 9\|\Phi(\Delta T)\|^2 -\\ & \left.\Delta T^2 \boldsymbol{f}(\Delta T)^T\left(3\boldsymbol{r}_d + \Delta T(\boldsymbol{v}_0 + 2\boldsymbol{v}_f - 2\boldsymbol{F}(\Delta T)) + 3\Phi(\Delta T)\right)\right),\end{aligned} \quad (27)$$

where $\boldsymbol{r}_d = \boldsymbol{r}_0 - \boldsymbol{r}_f, \boldsymbol{v}_s = \boldsymbol{v}_0 + \boldsymbol{v}_f$, and

$$\begin{aligned}\boldsymbol{F}(\Delta T) &= \textstyle\int_0^{\Delta T} \boldsymbol{f}(s)\,\mathrm{d}s,\\ \Phi(\Delta T) &= \textstyle\int_0^{\Delta T}\int_0^s \boldsymbol{f}(u)\,\mathrm{d}u\,\mathrm{d}s.\end{aligned}$$

3. Adaptive landing algorithm

So far we have established that by restricting our analysis to linear systems, we can derive an analytical solution to the indirect optimal control landing problem. However, trying to extend this analytical approach to the general problem is unviable. Consequently, we propose an algorithm designed to approximate the solution of the general problem by iterating the analytical solutions obtained from linear systems. Specifically, we are interested in computing optimal control $\boldsymbol{a}_c$ that makes the trajectory of the initial value problem

$$\begin{cases}\dot{\boldsymbol{r}} = \boldsymbol{v}\\ \dot{\boldsymbol{v}} = f(t,\boldsymbol{r},\boldsymbol{v}) + \boldsymbol{a}_c,\end{cases} \quad \text{subjectto}: \begin{cases}\boldsymbol{r}(0) = \boldsymbol{r}_0,\\ \boldsymbol{v}(0) = \boldsymbol{v}_0\end{cases} \quad (28)$$

to end, as close as possible, to the desired final conditions $(\boldsymbol{r}_f, \boldsymbol{v}_f)$.

To understand the heuristics of the algorithm, let's suppose that we have an approximate solution $(\tilde{\boldsymbol{r}}(t), \tilde{\boldsymbol{v}}(t))$ of the problem given in Eq. (2). We can linearize this problem by considering $\boldsymbol{f}(t, \tilde{\boldsymbol{r}}(t), \tilde{\boldsymbol{v}}(t))$ as an approximation of external accelerations. The analytical solution to this problem is known, and therefore we can calculate again an approximate trajectory of Eq. (2). This process allows for the iterative refinement of the external accelerations through the solution of non-autonomous linear problems. The final goal is to obtain an iterative process that accurately converges to the solution of the general problem.

To describe the algorithm in detail, we start by defining two auxiliary functions, L_a and L_n, which are the solutions to the autonomous and non-autonomous linear problems respectively:

- $L_a(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}, \Delta T) := (\boldsymbol{r}(t), \boldsymbol{v}(t), \lambda(t))$, where $\boldsymbol{r}(t), \boldsymbol{v}(t)$ and $\lambda(t)$ are the solution to

$$\begin{cases}\dot{\boldsymbol{r}} = \boldsymbol{v}\\ \dot{\boldsymbol{v}} = \boldsymbol{f} - \lambda\\ \ddot{\lambda} = 0\end{cases} \quad \text{Subjectto}: \begin{cases}\boldsymbol{r}(0) = \boldsymbol{r}_0 & \boldsymbol{v}(0) = \boldsymbol{v}_0\\ \boldsymbol{r}(\Delta T) = \boldsymbol{r}_f & \boldsymbol{v}(\Delta T) = \boldsymbol{v}_f\end{cases} \quad (29)$$

  Recall that in this case $\boldsymbol{f}$ is a constant vector.
- $L_n(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}(t), \Delta T) := (\boldsymbol{r}(t), \boldsymbol{v}(t), \lambda(t))$, where $\boldsymbol{r}(t), \boldsymbol{v}(t)$ and $\lambda(t)$ are the solution to

$$\begin{cases}\dot{\boldsymbol{r}} = \boldsymbol{v}\\ \dot{\boldsymbol{v}} = \boldsymbol{f}(t) - \lambda\\ \ddot{\lambda} = 0\end{cases} \quad \text{Subjectto} \quad \begin{cases}\boldsymbol{r}(0) = \boldsymbol{r}_0 & \boldsymbol{v}(0) = \boldsymbol{v}_0\\ \boldsymbol{r}(\Delta T) = \boldsymbol{r}_f & \boldsymbol{v}(\Delta T) = \boldsymbol{v}_f\end{cases} \quad (30)$$

Both auxiliary functions are given by the analytic solutions studied in the previous section. Since we are interested in optimal landing, two other auxiliary functions that calculate the optimal time ΔT that minimize the cost $\mathscr{J}$ defined in Eq. (4) are established. First, we define the function that provides the optimal time in the autonomous case,

$$\mathscr{T}_a(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}) := \Delta T, \quad (31)$$

where ΔT is the positive root of Eq. (18) that minimizes $\mathscr{J}$. In the autonomous case, $\mathscr{J}\prime(\Delta T)$ is a degree 4 polynomial; therefore, its roots are finite and perfectly determined. This is not the case for the optimal time in the non-autonomous problem. Since the external acceleration is a time-dependent general function $\boldsymbol{f}(t)$, we cannot define a function that returns the optimal time among all the roots of $\mathscr{J}\prime(\Delta T)$. Hence, we define the function

$$\mathscr{T}_n(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}(t), \Delta T_0) := \Delta T, \quad (32)$$

which returns a root of $\mathscr{J}\prime(\Delta T)$ in the nonautonomous case close to the point ΔT_0. Finally, we present in Alg. 1 an algorithm to approximate the solution of Eq. (2). This algorithm takes as input

- $\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f$, as two-point boundary conditions,
- $\boldsymbol{f}(t, \boldsymbol{r}, \boldsymbol{v})$, function that evaluate the general external accelerations, and
- N, number of iterations of the linear problem.

Then returns the last approximation of the optimal control $\boldsymbol{a}_c$ given by a final time ΔT.

Algorithm 1. Iterative algorithm for optimal landing

Require: $\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}(t, \boldsymbol{r}, \boldsymbol{v}), N$
Ensure: $\boldsymbol{a}_c, \Delta T$
$f_0 \leftarrow f(0, r_0, v_0)$ ▷Get constant external acceleration.
$\Delta T \leftarrow \mathscr{T}_a(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}_0)$ ▷Initial optimal time.
$(\boldsymbol{r}_1(t), \boldsymbol{v}_1(t), \lambda_1(t)) \leftarrow L_a(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}_0, \Delta T)$
▷First approx. by autonomous problem.
$n \leftarrow 1$
while $n < N$ **do**
$n \leftarrow n + 1$
$f_n(t) \leftarrow f(t, r_{n-1}(t), v_{n-1}(t))$▷Approx. external acceleration by the previous trajectory.
$\Delta T \leftarrow \mathscr{T}_n(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}_n(t), \Delta T)$ ▷Optimal time in non-autonomous case.
$(\boldsymbol{r}_n(t), \boldsymbol{v}_n(t), \lambda_n(t)) \leftarrow L_n(\boldsymbol{r}_0, \boldsymbol{v}_0, \boldsymbol{r}_f, \boldsymbol{v}_f, \boldsymbol{f}_n(t), \Delta T)$ ▷New approx. trajectory.
end while
$a_c \leftarrow -\lambda_N$

To measure the accuracy of this algorithm, we define the quantities

$$\begin{aligned} \varepsilon_r &:= \left|\hat{\boldsymbol{r}}(\Delta T) - \boldsymbol{r}_f\right| \\ \varepsilon_v &:= \left|\hat{\boldsymbol{v}}(\Delta T) - \boldsymbol{v}_f\right| \end{aligned} \tag{33}$$

where $(\hat{\boldsymbol{r}}, \hat{\boldsymbol{v}})$ is the solution to the initial value problem of Eq. (28) with the control and final time returned by the algorithm.

The following sections show the performance of the algorithm proposed (Alg. 1) in an Earth landing scenario with a complex environment of perturbative forces.

4. Perturbation models for Earth landing

This section briefly describes the mathematical models adopted in the iterative algorithm for Earth landing. Specifically, the simulation of the rocket's trajectory during Earth landing needs a realistic model to accurately capture the spacecraft's trajectory. Three factors are considered in the equations of motion:

- drag;
- discontinuous winds;
- variable gravity.

These factors make the differential equations governing the rocket's motion a highly non-linear, non-autonomous, and non-continuous system of the form:

$$\begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{v} \\ \dot{\boldsymbol{v}} = \boldsymbol{a}_d + \boldsymbol{a}_w + \boldsymbol{a}_g + \boldsymbol{a}_c, \end{cases} \tag{34}$$

where $\boldsymbol{a}_d, \boldsymbol{a}_w, \boldsymbol{a}_g$, and $\boldsymbol{a}_c$ represent the accelerations from drag, wind, gravity, and control, respectively. The mathematical models for all these accelerations are described in the following subsections.

4.1. Drag model

The International Standard Atmosphere Cavcar (2000) provides the following equation for the air density at the altitude z,

$$\rho(z) = \rho_0 \left(1 + \frac{\lambda z}{T_0}\right)^{-1-g/(R\lambda)} \tag{35}$$

where $\lambda = -0.0065\ ^\circ\text{K/m}$, $R = 287.1$ J/(kg $^\circ$K), $g = 9.08$ m/s^2, and $\rho_0 = 1.225$ kg/m^3 and $T_0 = 288.15\ ^\circ$K is the air density and the temperature at sea level ($z = 0$) in standard conditions. Note that this approximation of the atmospheric density is accurate for Earth's altitudes lower than 11 km. Therefore, the acceleration due to drag is described by

$$\boldsymbol{a}_d = -\frac{1}{2m} C_D A\, \rho(z) \sqrt{\boldsymbol{v}^T \boldsymbol{v}}\, \boldsymbol{v}(t) \tag{36}$$

where $\rho(z)$ is the air density (kg/m^3) at altitude z, $\boldsymbol{v}(t)$ is the velocity (m/s), A is the transverse surface area (m^2), and m is the rocket's mass (kg).

4.2. Wind model

In this work, we will consider that the landing rocket is subject to discontinuous transverse wind gusts. This means that the wind velocity will be given by a vector

$$\boldsymbol{w} = \begin{Bmatrix} w_x \\ w_y \\ 0 \end{Bmatrix} \tag{37}$$

where w_x and w_y are piecewise constant functions. An example of these transverse wind components is illustrated in Fig. 1.

The horizontal wind acceleration is modeled similar to drag:

$$\boldsymbol{a}_w = \begin{Bmatrix} \frac{\rho(z) A_x C_D}{2m} |w_x(t)| w_x(t) \\ \frac{\rho(z) A_y C_D}{2m} |w_y(t)| w_y(t) \\ 0 \end{Bmatrix} \tag{38}$$

where A_x and A_y are the lateral sections of the landing vehicle.

4.3. Gravity model

In many rocket landing control models, gravity is considered a function of altitude, and the curvature of the Earth is often ignored, as the changes it produces in gravity are relatively small for certain applications, especially at low altitudes. Then, we consider variable gravity for the distance from the Earth's center z,

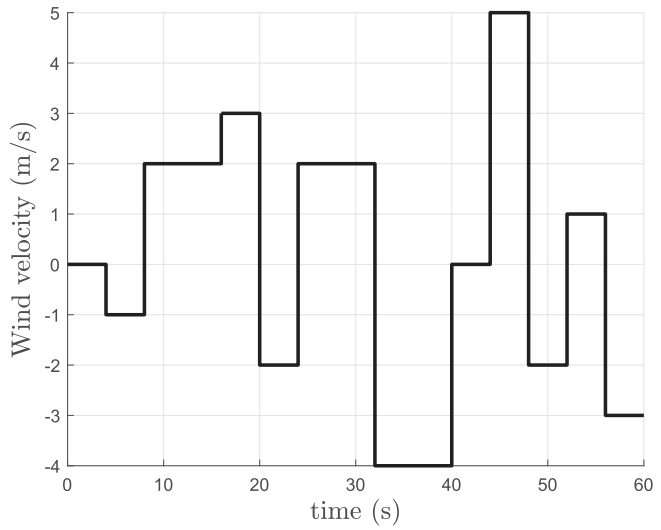

Fig. 1. Example of random wind intensity profile for one horizontal variable.

$$\boldsymbol{a}_g = -\frac{GM_E}{\|R_E + z\|^3}\begin{Bmatrix} 0 \\ 0 \\ z \end{Bmatrix}, \tag{39}$$

where G is the gravitational constant, and M_E, R_E are the mass and radius of Earth, respectively.

5. Numerical results

The purpose of this section is to perform a series of numerical experiments to validate the proposed algorithm and compare its effectiveness with MATLAB's built-in boundary value problem solver, `bvp4c`.

5.1. Problem formulation

The efficiency of the iterative Alg. 1 is tested in an Earth landing scenario of a rocket. For reusable rockets capable of landing, the optimal landing problem implies calculating the trajectory that brings the rocket to Earth's surface, minimizing fuel consumption and error in final position and velocity. In general, this process is a highly difficult task, not as simple as solving a trajectory boundary value problem. However, the equations of motion are used to capture most of the perturbations that the rocket is subject to. As stated in Eq. (2), the dynamics are described by the following equations:

$$\begin{cases} \dot{\boldsymbol{r}} = \boldsymbol{v}, \\ \dot{\boldsymbol{v}} = \boldsymbol{f}(t, \boldsymbol{r}, \boldsymbol{v}) + \boldsymbol{a}_c, \end{cases}$$

where $\boldsymbol{r}(t)$ represents the position of the rocket at time t, $\boldsymbol{v}(t)$ its velocity, $\boldsymbol{f}(t, \boldsymbol{r}, \boldsymbol{v})$ the external accelerations (or perturbations) acting on the rocket, and $\boldsymbol{a}_c$ is the control acceleration applied by the engines. The external accelerations decompose into,

$$\boldsymbol{f}(t, \boldsymbol{r}, \boldsymbol{v}) = \boldsymbol{a}_d + \boldsymbol{a}_w + \boldsymbol{a}_g, \tag{40}$$

where $\boldsymbol{a}_d, \boldsymbol{a}_w$, and $\boldsymbol{a}_g$ are the perturbations of drag, wind, and gravity, respectively, described in Section 4. The objective is to find the optimal control acceleration $\boldsymbol{a}_c(t)$ that minimizes the error in both the final position and the velocity. For our numerical experiments, we modeled the rocket using the physical characteristics data of the Falcon 9 from SpaceX, whose parameters are summarized in Table 1. Based on these values, we can estimate the system parameters as follows.

- The rocket is modeled as a long cylinder with cross-sectional areas $A_x = A_y = 110.16$ m^2 and $A_z = 10.52$ m^2.
- The drag coefficient is set at $C_D = 0.82$.
- At the start of the landing phase, the rocket has 50 % of its total propellant mass, giving a total mass of $m = 1.39 \times 10^5$ kg.

The final position $\boldsymbol{r}_f$ is always set to the origin. On the other hand, to avoid singularities in MATLAB's `bvp4c` due to the Jacobian of the drag acceleration, the final velocity is set to $\boldsymbol{v}_f = (0, 0, -0.1)$ m/s. In the following numerical simulations, the initial guess of solution for `bvp4c` is chosen as the analytical solution to the unperturbed problem given in Eq. (8).

5.2. Algorithm convergence

When implementing the iterative algorithm, it is necessary to take a discretization in the time interval to estimate the integrals of the analytical solution of Eq. (26). Thus, the accuracy of the algorithm, measured by the final position and velocity errors, ε_r and ε_v, depends on both the number of iterations (N) and the number of discretizations in time (M). To understand how the error behaves as N and M increase, we analyze the average error in solving Eq. (2) with random initial conditions based on a uniform distribution in the intervals

$$\begin{cases} r_{0_x}, r_{0_y} \in [-1,000, 1,000]\ (m), \\ r_{0_z} \in [8,000, 10,000]\ (m), \\ v_{0_x}, v_{0_y} \in [-100, 100]\ (m/s), \\ v_{0_z} \in [-500, -300]\ (m/s). \end{cases} \tag{41}$$

Furthermore, in each simulation, the piecewise constant functions that determine the wind speed have been ran-

Table 1
Characteristics of Falcon 9's stage 1 with a Merlin 1C engine (SpaceX, 2017).

Diameter (m)	3.66 m
Length (m)	30.1 m
Empty mass (tonnes)	19.24 t
Propellant Mass (tonnes)	239.3 t
Burn Time (sec)	180 s

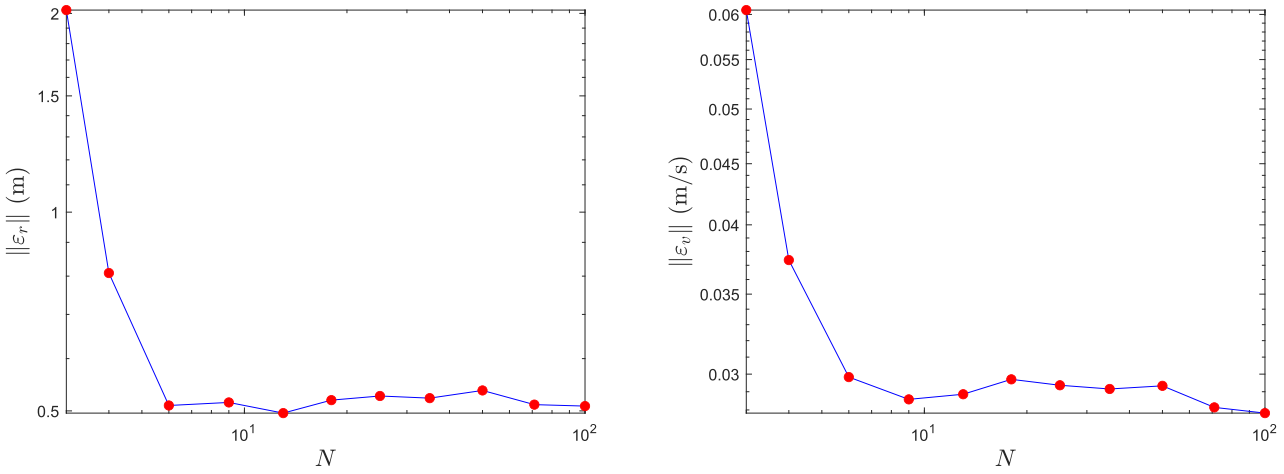

Fig. 2. Mean error of the iterative algorithm with respect to N for a fixed number of discretization points $M = 100$. The results show that convergence is already obtained with $N = 10$ iterations..

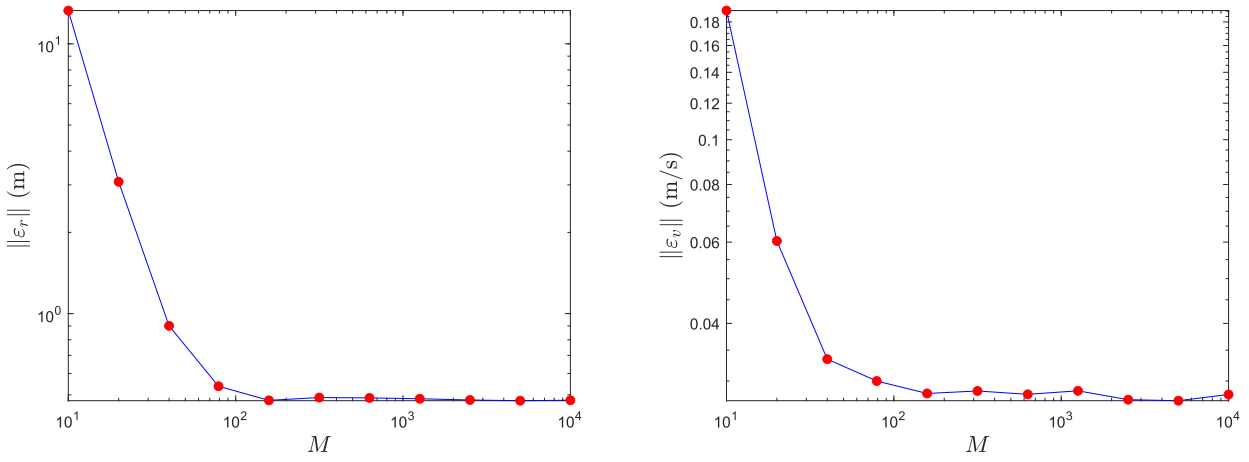

Fig. 3. Mean error of the iterative algorithm with respect to M for a fixed number of iterations $N = 100$.

domly generated, so that the maximum speed is 10 m/s and the average duration of each gust, or constant piece, is 5 s (see Fig. 4). After 1,000 simulations, the errors as a function of N and M are represented in Figs. 2 and 3, respectively. It can be observed that above a threshold in N and M, the decrease in the error stabilizes, leaving an oscillation due to the randomness of the initial conditions. Therefore, a value of $N = 50$ and $M = 1,000$ will be taken in subsequent simulations to ensure that the error is minimal.

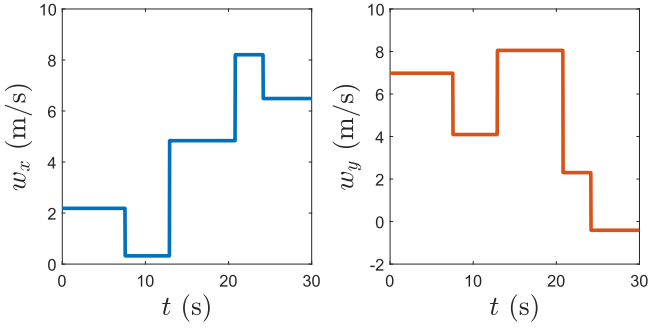

Fig. 4. Example of the wind profile with piecewise constant functions.

5.3. Wind-free scenario

As seen in the previous section, wind perturbation significantly affects the accuracy of the proposed algorithm, often leading to substantial errors. To evaluate performance without this source of error, this experiment compares the proposed algorithm with the MATLAB solver in a wind-free landing scenario. We generated 1,000 random initial conditions within the same intervals specified in Eq. (41). These initial conditions were used to solve the boundary value problem of Eq. (2) with $\boldsymbol{f} = \boldsymbol{a}_d + \boldsymbol{a}_g$ using both Alg. 1 and the MATLAB solver.

Fig. 5 and Fig. 6 present the distribution of the error norm in both position and velocity across these simulations. As summarized in Table 2, the mean error values for each component of position and velocity are similar between the two algorithms, demonstrating comparable accuracy. However, Alg. 1 solves the problem in approxi-

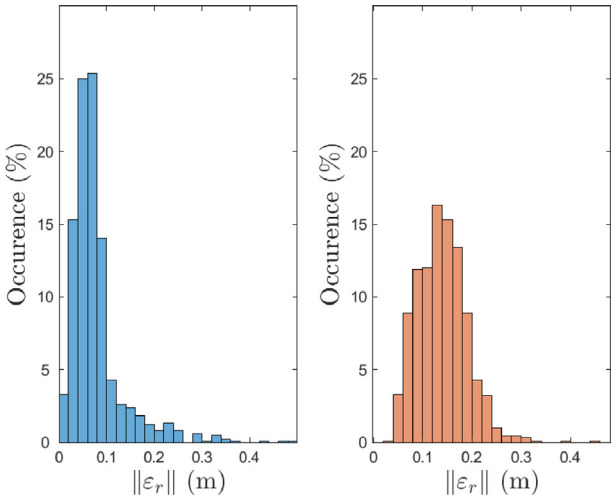

Fig. 5. Distribution of $\|\varepsilon_r\|$ in the iterative algorithm (blue) and `bvp4c` (orange) in the wind-free scenario.

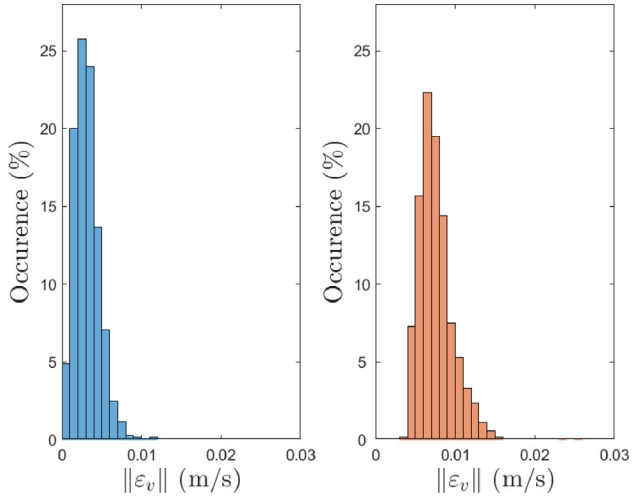

Fig. 6. Distribution of $\|\varepsilon_v\|$ in the iterative algorithm (blue) and `bvp4c` (orange) in the wind-free scenario.

mately half the computation time of the MATLAB solver. The average optimal maneuver time in this experiment was $\Delta T_{avg} = 44.69$ s.

5.4. Unidirectional wind scenario

Next, we evaluate the effectiveness of the proposed algorithm under gusty wind conditions from a fixed direction. Due to the radial symmetry of Eqs. (34), we can consider any wind direction to study this case. Thus, the wind component in the y-direction (w_y) is set to zero, while the wind component in the x-direction (w_x) is modeled as a random piecewise constant function in each simulation. The wind speed is uniformly distributed from 0 to 10 m/s, and the average duration of each gust is 5 s. After conducting simulations $1,000$, the error distributions for both the proposed algorithm and `bvp4c` are shown in Fig. 7 and Fig. 8. The results show that the error distribution is significantly narrower for Alg. 1, indicating greater robustness and stability under varying gust conditions. Furthermore, the proposed algorithm achieves this improved performance with a much shorter computation time (see Table 3). In Fig. 9, we can see the projection of the landing points in the X-Y plane. As expected, the error in the landing position is biased towards the direction of the wind. The average optimal maneuver time in this case is similar to the previous experiment, with $\Delta T_{avg} = 44.40$ s.

5.5. Random wind scenario

Finally, we study the performance of the proposed algorithm in scenarios where wind gusts can blow from any direction, creating a more complex perturbation environment. In this case, both wind components (w_x and w_y) are generated independently as piecewise constant functions, simulating gusts with random magnitudes and directions (see Fig. 4). The error distributions and performance metrics for Alg. 1 and `bvp4c` are presented in Fig. 10, Fig. 11 and Table 4. The results indicate that the introduction of gusts with varying directions significantly increases the overall error in both algorithms, with the effects being more pronounced compared to unidirectional gusts. However, the proposed algorithm demonstrates superior performance, maintaining lower error values, and showing better response to changing wind directions. In contrast, the MATLAB solver exhibits much higher errors, suggesting that it struggles to converge properly under these condi-

Table 2
Error summary in windless simulations for the iterative algorithm and `bvp4c`.

	Mean value		Maximum value	
	Algorithm 1	`bvp4c`	Algorithm 1	`bvp4c`
Computing time (s)	0.1661	0.4687	0.6749	1.4088
ε_{r_x} (m)	3.5720×10^{-4}	3.5388×10^{-4}	3.3344×10^{-3}	2.7489×10^{-3}
ε_{r_y} (m)	3.6289×10^{-4}	3.4779×10^{-4}	2.9599×10^{-3}	2.8259×10^{-3}
ε_{r_z} (m)	4.7052×10^{-4}	1.2459×10^{-3}	2.7784×10^{-3}	2.8553×10^{-3}
ε_{v_x} (m/s)	1.2661×10^{-5}	1.8686×10^{-4}	6.6254×10^{-5}	1.6394×10^{-4}
ε_{v_y} (m/s)	1.2667×10^{-5}	1.8756×10^{-4}	5.6494×10^{-5}	1.6334×10^{-4}
ε_{v_z} (m/s)	2.1786×10^{-5}	6.9127×10^{-4}	1.0235×10^{-4}	1.3961×10^{-4}

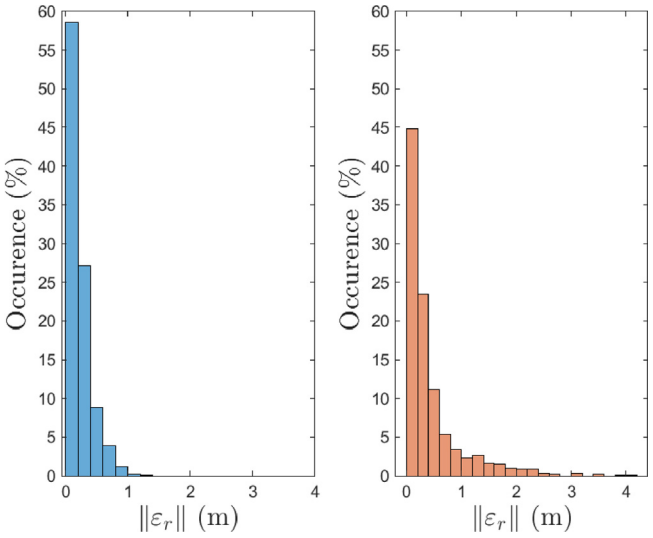

Fig. 7. Distribution of $\|\varepsilon_r\|$ in the iterative algorithm (blue) and `bvp4c` (orange) in the unidirectional wind scenario.

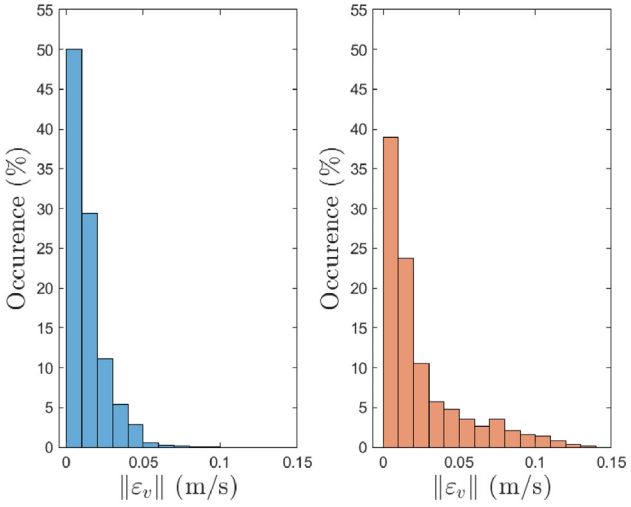

Fig. 8. Distribution of $\|\varepsilon_v\|$ in the iterative algorithm (blue) and `bvp4c` (orange) in the unidirectional wind scenario.

tions. In this case, the average maneuver time in the simulations was $\Delta T_{avg} = 44.46$ s.(see Fig. 12).

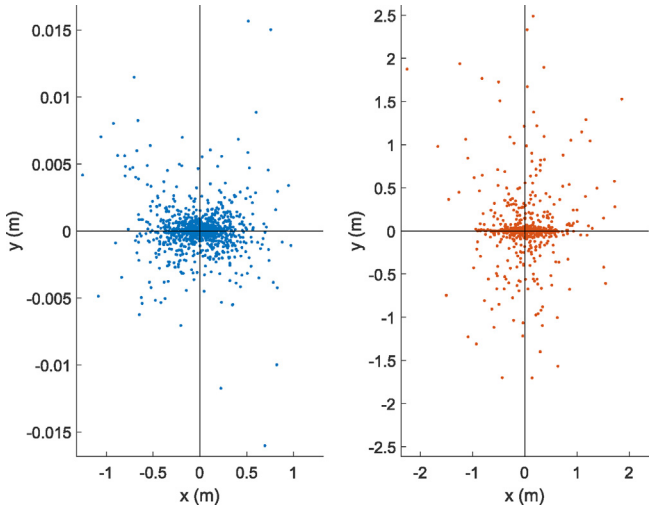

Fig. 9. Projection of the endpoint of the trajectory onto the X-Y plane for the proposed algorithm (blue) and for `bvp4c` (orange). Note that the axes are on different scales in both cases to appreciate better the error bias in the left figure. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

6. Conclusions

In this study, we have explored an indirect method to solve the optimal landing problem, extending the work of D'Souza. We derived an analytical solution for linear problems with time-dependent external accelerations. This solution enabled us to provide a closed-form expression for the control, which allowed us to calculate the total maneuver time to minimize the cost of the landing process. Using this analytical solution as a base, we developed an algorithm capable of approximating trajectories for more complex nonlinear landing scenarios. The algorithm was tested under various conditions simulating a rocket landing, including scenarios with variable drag forces, non-constant gravity, and discontinuous wind gusts. Despite these challenging conditions, the proposed algorithm consistently demonstrated robust performance, yielding errors significantly smaller than MATLAB's `bvp4c` solver, while achieving remarkably low computation times.

Furthermore, this work also reveals the impact of discontinuous perturbations on the accuracy of numerical

Table 3
Error summary in simulations with unidirectional wind for the iterative algorithm and `bvp4c`.

	Mean value		Maximum value	
	Algorithm 1	`bvp4c`	Algorithm 1	`bvp4c`
Computing time (s)	0.1587	0.9227	0.3118	1.6267
ε_{r_x} (m)	2.1447×10^{-1}	2.5813×10^{-1}	1.2537×10^{0}	2.2560×10^{0}
ε_{r_y} (m)	1.0584×10^{-3}	1.3069×10^{-1}	1.6027×10^{-2}	2.4882×10^{0}
ε_{r_z} (m)	1.3764×10^{-3}	2.1588×10^{-1}	1.6073×10^{-2}	3.2856×10^{0}
ε_{v_x} (m/s)	1.3043×10^{-2}	1.4705×10^{-2}	9.6547×10^{-2}	1.0529×10^{-1}
ε_{v_y} (m/s)	7.0018×10^{-5}	4.7704×10^{-3}	1.3201×10^{-3}	8.7488×10^{-2}
ε_{v_z} (m/s)	7.9432×10^{-5}	1.3354×10^{-2}	6.8755×10^{-4}	1.1947×10^{-1}

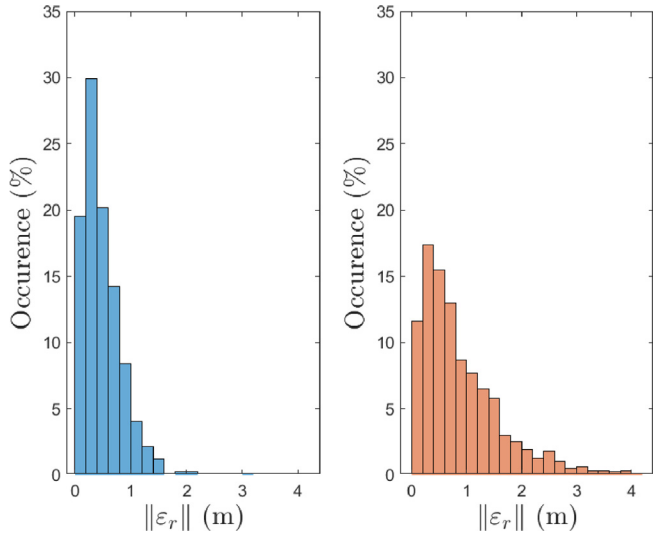

Fig. 10. Distribution of $\|\varepsilon_r\|$ in the iterative algorithm (blue) and `bvp4c` (orange) in the bidirectional wind scenario. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

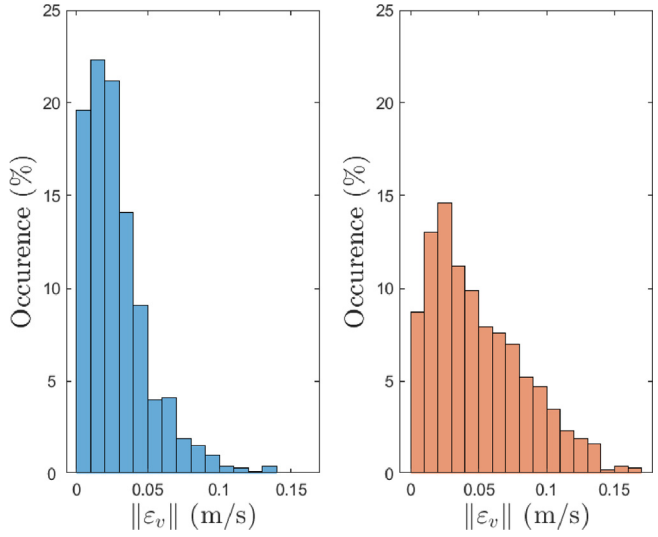

Fig. 11. Distribution of $\|\varepsilon_v\|$ in the iterative algorithm (blue) and `bvp4c` (orange) in the bidirectional wind scenario. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

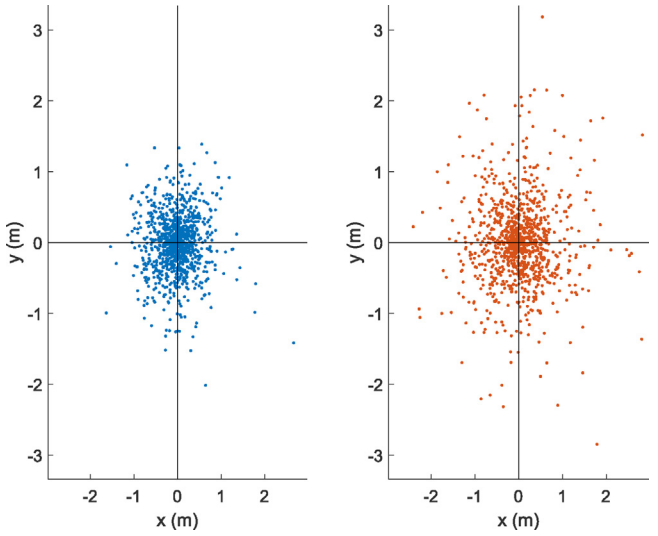

Fig. 12. Projection of the endpoint of the trajectory onto the XY plane for the proposed algorithm (blue) and for `bvp4c` (orange). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

methods for boundary value problems. Extreme discontinuous wind gusts like the ones studied in this paper, although rare in practice, provide a critical upper bound on error margins under real conditions. In addition, the direction of discontinuous perturbations biases the algorithm error on landing, as shown in the unidirectional wind experiment. This offers an advantage over MATLAB's solver, where `bvp4c` distributes the error to directions where there are no discontinuities.

The effectiveness of the algorithm in handling abrupt environmental changes, such as wind gusts and non-uniform drag forces, shows its potential for real-world applications. The fast convergence and accuracy in the trajectory approximation make it particularly useful for situations where GPS corrections are unavailable. In addition, the algorithm can serve as a tool for estimating an initial trajectory in more complex landing models that consider additional factors like variable rocket mass or spacecraft's attitude. By providing a fast and reliable esti-

Table 4
Error summary in simulations with bidirectional wind for the iterative algorithm and `bvp4c`.

	Mean value		Maximum value	
	Algorithm 1	`bvp4c`	Algorithm 1	`bvp4c`
Computing time (s)	0.1291	1.3180	0.1803	2.0972
ε_{r_x} (m)	3.0782×10^{-1}	4.5643×10^{-1}	2.6658×10^{0}	2.8284×10^{0}
ε_{r_y} (m)	3.1102×10^{-1}	4.2765×10^{-1}	2.0163×10^{0}	3.1822×10^{0}
ε_{r_z} (m)	3.1287×10^{-3}	4.4428×10^{-1}	3.1309×10^{-1}	3.0436×10^{0}
ε_{v_x} (m/s)	1.8659×10^{-2}	2.3725×10^{-2}	1.0555×10^{-1}	1.2944×10^{-1}
ε_{v_y} (m/s)	1.8027×10^{-2}	2.2270×10^{-2}	1.3405×10^{-1}	1.4013×10^{-1}
ε_{v_z} (m/s)	1.7834×10^{-4}	2.7075×10^{-2}	1.2337×10^{-2}	1.4269×10^{-1}

mate, it supports the refinement of landing strategies in systems where precise control is critical.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The research was supported by the Ministry of Science and Innovation and the State Research Agency of Spain: grant PID2020-117066 GB-I00 funded by MICIU/AEI/10.13039/501100011033 and grant PRE2021-099561 funded by MICIU/AEI/10.13039/501100011033 and by "ESF+".

References

Acikmese, B., Ploen, S.R., 2007. Convex programming approach to powered descent guidance for mars landing. J. Guid., Control, Dynam. 30 (5), 1353–1366. https://doi.org/10.2514/1.27553.

Blackmore, L., Acikmese, B., Scharf, D.P., 2010. Minimum-landing-error powered-descent guidance for mars landing using convex optimization. J. Guid., Control, Dynam. 33 (4), 1161–1171. https://doi.org/10.2514/1.47192.

Blackmore, L., Açikmeşe, B., Scharf, D.P., 2010. Minimum-landing-error powered-descent guidance for mars landing using convex optimization. J. Guid., Control, Dynam. 33 (4), 1161–1171.

Cavcar, M., 2000. The International Standard Atmosphere (ISA). Anadolu University, Turkey 30 (9), 1–6.

Chai, R., Savvaris, A., Tsourdos, A., et al., 2019. Solving trajectory optimization problems in the presence of probabilistic constraints. IEEE Trans. Cybernet. 50 (10), 4332–4345.

D'Souza, C., 1997. An optimal guidance law for planetary landing. Guidance, Navigat., Control Conf., 3709

D'Ambrosio, A., Schiassi, E., Johnston, H., et al., 2022. Time-energy optimal landing on planetary bodies via theory of functional connections. Adv. Space Res. 69 (12), 4198–4220.

Furfaro, R., Mortari, D., 2020. Least-squares solution of a class of optimal space guidance problems via theory of connections. https://doi.org/10.1016/j.actaastro.2019.05.050.

Gaudet, B., Linares, R., Furfaro, R., 2020. Deep reinforcement learning for six degree-of-freedom planetary landing. Adv. Space Res. 65 (7), 1723–1741.

Johnston, H., Schiassi, E., Furfaro, R., et al., 2020. Fuel-efficient powered descent guidance on large planetary bodies via theory of functional connections. J. Astronaut. Sci. 67 (4), 1521–1552. https://doi.org/10.1007/s40295-020-00228-x.

Lavezzi, G., Guye, K., Cichella, V., et al., 2023. Comparative analysis of nonlinear programming solvers: performance evaluation, benchmarking, and multi-uav optimal path planning. Drones 7 (8), 487. https://doi.org/10.3390/drones7080487.

Liu, F., Hager, W.W., & Rao, A.V. (2015). Adaptive mesh refinement method for optimal control using nonsmoothness detection and mesh size reduction. Journal of the Franklin Institute, 352(10), 4081–4106. URL: https://www.sciencedirect.com/science/article/pii/S0016003215002045. doi: 10.1016/j.jfranklin.2015.05.028.

Longuski, J.M., Guzmán, J.J., Prussing, J.E., 2014. Optimal control with aerospace applications, volume 32. Springer.

Mortari, D., 2017. The theory of connections: connecting points. Mathematics 5 (4), 57-1–57-15. https://doi.org/10.3390/math5040057.

Moyer, H.G., Pinkham, G., 1964. Several trajectory optimization techniques: Part II: application. In Computing Methods in Optimization Problems. Elsevier, pp. 91–105.

Patterson, M.A., Rao, A.V., 2014. GPOPS-II: A MATLAB software for solving multiple-phase optimal control problems using hp-adaptive Gaussian quadrature collocation methods and sparse nonlinear programming. ACM Trans. Math. Software (TOMS) 41 (1), 1–37.

Pontryagin, L.S., 1987. Mathematical theory of optimal processes. Gordon and Breach Science Publishers, Switzerland.

Rösmann, C., Feiten, W., Wösch, T., et al., 2013. Efficient trajectory optimization using a sparse model. In: 2013 European Conference on Mobile Robots. IEEE, pp. 138–143.

SpaceX, 2017. Falcon 9 launch vehicle: Payload user's guide. URL: https://sma.nasa.gov/LaunchVehicle/assets/spacex-falcon-9-data-sheet.pdf.

Szmuk, M., Reynolds, T.P., Açikmeşe, B., 2020. Successive convexification for real-time six-degree-of-freedom powered descent guidance with state-triggered constraints. J. Guid., Control, Dynam. 43 (8), 1399–1413.