



Autonomous parafoil precision landing using convex real-time optimized guidance and control

Antoine Leeman^{1,2} · Valentin Preda¹ · Irene Huertas¹ · Samir Bennani¹

Received: 14 May 2021 / Revised: 12 November 2021 / Accepted: 19 November 2021 / Published online: 18 January 2022
© The Author(s) 2022

Abstract

To overcome the limitations of current parafoil precision landing capabilities, an efficient real-time convex optimized guidance and control strategy is presented. The parafoil guidance problem is non-convex in essence and it can be solved with a sequence of convex problems, each of those converging in polynomial time to a feasible solution of the approximated original problem. Our approach shows reliable and fast numerical convergence through in-flight recalculation of time of flight and a new optimal trajectory to cope with time-varying dynamics. The efficiency of our strategy is demonstrated via a comparative analysis of the existing X-38 in-flight demonstrated guidance and control system. Exhaustive Monte Carlo simulations show performance improvements of about one order of magnitude. The concept proposed is simple, yet general, as it scales to any atmospheric parafoil landing system and allows efficient implementation relying only on the turn rate saturation information for the parafoil model.

Keywords Parafoil · Convex programming · Guidance and control · Landing · Real time · Trajectory optimization · Sequential convex programming

1 Introduction

1.1 Background

The increased availability of onboard computation power and efficient optimization algorithms associated with automated code generators [23] have dramatically boosted the industrial propagation of embedded optimized guidance and control strategies. Pushed by the research of Açıkmese [21], numerous aerospace problems can be solved in real-time. Namely, it was possible to land spaceships accurately [2, 3] using that strategy. Convex optimization is widely used in real-time trajectory optimization problems due to the maturity of the technology [1, 4] and the convergence speed to a global solution. The successive convexification (SCVx) method used in [20], for instance, has been proven

to be superlinearly convergent to a solution that may not be dynamically feasible [22]. The successes of the adoption of embedded onboard optimization rely on the numerical efficiencies of powerful algorithms such as Interior-Point Methods (IPM). Originally, parafoil guidance has been performed using a sequence of open-loop trajectories, namely acquisition phase, energy management phase, and final phase [31]. The various phases of flight are triggered by a trade-off between potential and kinetic energy indicators guiding sequentially the parafoil to its landing site. In the last section, we compare our results with the method from [31] which is based on this concept. While this method is reliable and fast, it does not take into consideration the uncertainties of the parafoil and wind model, leading to poor landing precision performances. This concept is further explored and improved in [34]. When the landing precision is a driving factor, especially in unknown and rugged terrain, such as for scientific payload delivery for exploration, or fairing recovery, the former technologies do not meet the requirements. To overcome these limitations, this paper demonstrates the potentials of using onboard optimized trajectory generation and control to increase drastically performance figures. Similarly to other techniques, the parafoil and wind models uncertainties are not explicitly handled. The superior

✉ Antoine Leeman
aleeman@ethz.ch

¹ GNC, AOCS and Pointing Division Directorate of Technology, Engineering and Quality European Space Agency, Keplerlaan 1, 2201 AZ Noordwijk, Netherlands

² Institute for Dynamic Systems and Control, ETH Zurich, Zurich, Switzerland

performance stems from the ability to recompute a trajectory onboard multiple times.

Driven by the need to improve the landing accuracy, series of authors have approached the problem using onboard optimization [19]. When the landing precision is the primal goal, all the methods for parafoil guidance face conflicting objectives. One needs both an accurate parafoil model and a fast refresh rate to account for the uncertainties. However, an accurate model in the optimization problem naturally comes together with heavy computation cost. In the parafoil problem, and the wind influences strongly the dynamics and computing new trajectories onboard is unavoidable for accurate landing. To start with, the algorithm introduced by Slegers in [30] uses a Model Predictive Control (MPC) on a 2-Degrees Of Freedom (DOF) model for the terminal guidance phase of the landing trajectory. The method can update the reference with a high refresh rate mitigating the effect of the highly unknown wind, which is assumed constant in the optimization process. In [36], the authors also frame the parafoil landing problem using a famous trajectory optimization strategy. They use Particle Swarm Optimisation (PSO) to generate a trajectory online with an update rate of 1Hz. It can generate smooth trajectories due to the penalty and bounds on the heading acceleration; it shows good performances when compared against other real-time methods. However, PSO do not guarantee convergence to a solution.

In terms of collision avoidance, [14] presents a Rapidly-exploring Random Tree (RRT) algorithm to avoid the especially difficult urban environment. Similarly, [28] proposes a collision avoidance algorithm tailored for parafoil, using stochastic guidance.

It is important to take it into account explicitly the wind uncertainty as highlighted by the authors of [7]. They decide to tackle the wind uncertainty problem with a frame called the Wind Fixed Frame (WFF), initially introduced in [31]. The algorithm presented in [16] focuses on the robustness of the trajectory subject uncertainty of the wind.

Most of the recent literature focuses on exploiting the peculiarity of the parafoil problem. The innovation of the guidance proposed in [12] is the use of Euler's elastica functions to get the trajectory that minimizes the maximum curvature, taking advantage of the similarity of the equations of motion with a structure problem. Sacrificing some precision, the algorithm runs in real time. The idea exposed on [5] incorporates the actuator limitation in the optimization problem which allows both a refresh frequency of 1 Hz and a guarantee of non-saturation of actuators. The parafoil model used in [15] takes into account that the horizontal and vertical velocities are influenced by the altitude through the air density, leading to more representative results. The paper brings a method to take into account non-convex and difficult terrain, while it ignores the winch dynamics. Finally, [33] uses a more realistic model compared to the

aforementioned methods. As a result, it takes into account the flared landing. However, the real-time implementation is not specified.

1.2 Contribution

Onboard optimization for parafoil guidance and control was considered by Slegers in [29]. While the linear MPC offers convergence and recursive feasibility guarantees, the low precision of the linear model used leads to a largely sub-optimal solution, and lower landing precision performances. On the other hand, the work of Rademacher in [25] considers a more accurate model, including more physical parameters of the parafoil. It shows better results in terms of landing precision. Besides, it considers a realistic wind profile. However, the performances depend largely on the precision of estimation of the aerodynamic coefficients of the parafoil model. A small error on each parameter adds up to a large landing precision error as shown in the Monte Carlo simulations. Besides, the convergence guarantees are not clear for an arbitrary initial guess. The contribution of this paper is to provide a method with fast convergence with a 4-DOF model, which is intermediately accurate model with respect to the models used in the literature. In [24], the authors use a sequential convex approach as well, but rely on a 6-DOF model. While the resulting trajectory will be easier to track, each iterate is not guaranteed to be dynamically feasible. Since the model used is more richer, one could expect more optimization variables and therefore a longer convergence time. However, in our approach, since we use a 4-DOF model, the aerodynamic coefficients do not have to be known, and the method is fully transparent to parameters knowledge and estimation. Besides, estimating aerodynamics coefficients come often with errors and large engineering efforts. Moreover, each iteration provides a useful trajectory for the parafoil, despite being potentially largely sub-optimal. The two main requirements for the algorithm to perform are knowledge about the profile of air density and the level of saturation of the actuators. Both are considered light. Overall, it enables the use of onboard optimization for a larger set of missions. It assumes that the landing cone is free of any obstacle, ignoring any unevenness on the landing area. The landing cone is defined by the set of position where a dynamically feasible trajectory exist, leading to the landing site. Overall, this paper provides a new interesting trade-off between fidelity of the trajectory and its computation time, allowing anytime control at the same time.

1.3 Outline

In Sect. 2, the parafoil landing problem is introduced. It contains the parafoil and wind model used and the optimization problem to be solved, and it provides an estimation of the horizontal and vertical speeds and the time of

Fig. 1 Schematic representation of the parafoil along the trajectory. \mathbf{x}_0 and \mathbf{x}_f are, respectively, the initial and final positions. v and r are the horizontal speed and vertical speed. ψ is the heading angle, defined as the angle between the horizontal speed v and the inertial axis \mathbf{x}_{1I} . $\dot{\psi}_m$ is the maximum heading rate

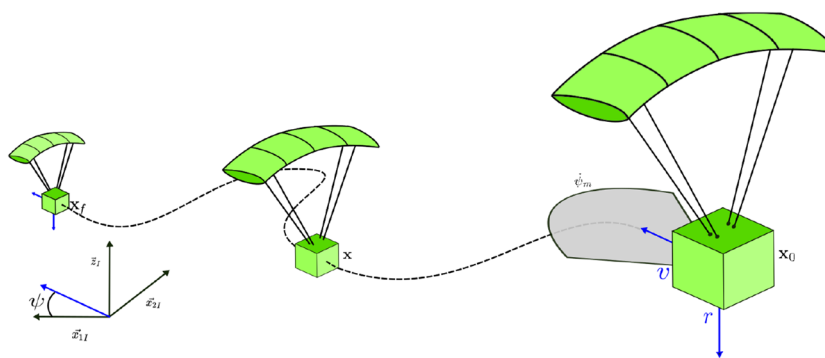


Table 1 Comparison between 4-DOF and 6-DOF parafoil model

Model	4-DOF	6-DOF
Common use	Trajectory optimization	Simulation
Limitations	Roll and pitch angle	Flexible modes
Nature	Kinematics	Dynamics

flight. Then, in Sect. 3, a resolution of the aforementioned parafoil landing problem is presented. The resolution has the possibility to be implemented in real time, since each iterate of the algorithm consists of a convex problem and yields a usable solution to the original non-convex parafoil problem. As the iterative process progresses, the trajectory becomes more and more useful. The possibility to generate a new reference trajectory alleviates the effect of the uncertainties and unmodelled dynamics on the landing precision. Namely, the algorithm presented allows the continuous non-convex problem to be discretized, convexified, and linearized for real-time resolution. A simple controller tracking the generated reference trajectory is finally introduced. In Sect. 4, the performances are assessed with a Monte Carlo simulation, as well as compared to the guidance of the X-38 mission, named PGNC.

1.4 Notation

The scalars are written with the unmodified font v ; the vectors are written in bold \mathbf{x} and the matrices in uppercase bold \mathbf{A} . The lower script numbers \mathbf{x}_k refer to the quantity evaluated at the time step t_k . We have $\mathbf{x}_k = \mathbf{x}(t_k)$ unless it is clear from the context. The lower script \mathbf{x}_0 is referring to the initial time 0 and the lower script \mathbf{x}_f is referring to the final time t_f . The time derivative of a quantity $\mathbf{x}(t)$ is written $\frac{\partial \mathbf{x}(t)}{\partial t} = \dot{\mathbf{x}}(t)$. The overline $\bar{\mathbf{x}}$ refers to the solution of the previous iteration of the sequential Algorithm and \mathbf{x}^* refers to a feasible solution of a non-convex problem. The symbol $\|\mathbf{x}\|$ refers to the 2-norm of a vector \mathbf{x} with $\|\mathbf{x}\| = \sqrt{x_1^2 + x_2^2}$. Moreover, \mathbf{x}^\top is the transpose of \mathbf{x} . The position of the parafoil is defined

with respect to the inertial frame $F_I = \{O_I, \mathbf{x}_{1I}, \mathbf{x}_{2I}, z_I\}$ whose origin O_I lies at the desired landing point, with \mathbf{x}_{1I} aligned with the desired final heading.

2 Problem formulation

2.1 Parafoil and environment modeling

2.1.1 Parafoil

Modeling the dynamical system, the parafoil, constitutes the first step of the problem formulation and there exist several ways to do it. First, we can use the first principles of physics and consider methodically all the forces and torques acting on the payload and parafoil, possibly considering highly nonlinear effects such as aeroelasticity. This classic modeling is accurate, but contains many parameters and takes long to optimize over, which is to be avoided for onboard optimization. It is commonly used as a benchmark model to verify the numerical performances of the guidance offline. Alternatively, we can base our model on the empirical observation that the parafoil rate of fall r and the horizontal speed v are roughly constant for a given air density, leading to a 4-DOF model. The horizontal motion is fixed by the heading angle and the horizontal speed. One can control the rate of heading angle and the last degree of freedom is considered uncontrollable and models the vertical position. This kinematics model is illustrated in Fig. 1, and as in [37], it depends on the wind strength w as an additive term to the speeds. This model is representative of the global parafoil motion and is therefore useful for guidance purposes. However, the 4-DOF model does not allow optimization for flare maneuver and generally ignores ground effect. Those dynamical limitations of the model will result in the suboptimality of the solution of the parafoil landing problem. As it will be seen later, using a simpler model allows to run the optimization online. Both models, the one based on forces and torques (6-DOF) and the overall motion (4-DOF),

are compared in Table 1. The 4-DOF model will be used to derive the optimal parafoil guidance while the 6-DOF from [25] will be used as a benchmark to assess the performance of the Algorithm.

Choosing x_j and w_j as, respectively, the position and the wind speed in the j -direction, ψ as the heading angle, and z as the altitude, the 4-DOF parafoil model used for the optimization is the following:

$$\dot{y}(t) = f_{4\text{DOF}}(y(t)) + B\delta(t) = \begin{bmatrix} \dot{x}_1(t) \\ \dot{x}_2(t) \\ \dot{\psi}(t) \\ \dot{z}(t) \end{bmatrix} \tag{1}$$

$$= \begin{bmatrix} v(z(t)) \cos \psi(t) + w_1(z(t)) \\ v(z(t)) \sin \psi(t) + w_2(z(t)) \\ 0 \\ r(z(t)) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \delta(t).$$

The control input $\delta(t)$ corresponds to an asymmetrical deflection of the parafoil trailing edges. The time of flight t_f , the horizontal speed $v(t)$, and the vertical speed $r(t)$ are derived in the Sect. 2.3.

2.1.2 Environment

A precise model of the environment, such as the wind and the air density, will improve the representativeness of the problem and naturally improve the quality of the solution. In the optimization parafoil model of Eq.(1), the horizontal and vertical velocities depend on the air density modeled as a function of the altitude as

$$\rho(z) = c_h(1 - z \cdot c_z)^{c_e}, \tag{2}$$

where $c_h = 1.225\text{kgm}^{-3}$, $c_z = 2.256e - 5\text{m}^{-1}$ and $c_e = 4.2559$ [37].

As for the wind, it is generated with a Dryden wind turbulence model, similarly to [27]. A Dryden wind turbulence model is a Linear Parameter Varying (LPV) G_w . It can be implemented using a digital filter which that takes a band-limited white-noise with unitary variance n_w as input and outputs the wind speed given the altitude and the ground speed as parameters. The wind generator is made up of three terms; a steady-state profile $G_{w_{ss}}$, a low-frequency/high-amplitude gust $G_{w_{LF}}$, and a high-frequency/low-amplitude gust $G_{w_{HF}}$. They are parametrized with the altitude z and the ground speed v_a . The resulting wind model used is

$$G_w(z, v_a) = \frac{w_{x,y}(z, v_a)}{n_w} = G_{w_{ss}}(z) + G_{w_{LF}}(z, v_a) + G_{w_{HF}}(z, v_a), \tag{3}$$

where we dropped the Laplace variable s for readability. The wind in both horizontal directions is generated with a Dryden wind turbulence model [27]. Figure 2 shows the

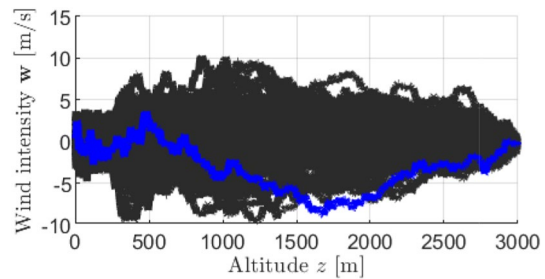


Fig. 2 Set of wind profiles generated by the Dryden wind turbulence model with a single wind outlined

envelope of the wind profile, as a function of the altitude. Those wind profiles are used later for simulation in Sect. 4. We consider the vertical wind speed to be negligible compared to the vertical speed of the parafoil. However, it could be easily integrated inside the time-of-flight estimation.

2.2 Optimization problem

2.2.1 Description

The parafoil landing problem consists in finding an optimal time-varying reference control input $\delta(t)$ for the rate of change of the heading angle, leading the parafoil towards the landing site with the desired heading angle while minimizing the squared control cost in the objective function. Compared to other objective functions, such as 1-norm for instance, it will naturally yield higher control margins. Without loss of generality, the desired final position will be taken as zero and the desired final heading angle is defined as zero, aligned with the inertial frame vector x_{1I} . The objective of the problem is captured by the cost function (4a) that we seek to minimize. Formally, the solution of the problem is the functions of time $y^*(t)$ and $\delta^*(t)$ which are required to respect a list of constraints¹. We will see in the following that $\delta^*(t)$ is used as a feed-forward, or guidance, while $y^*(t)$ is used for the feedback control. As such, the constraint (4b) enforces the states $y(t)$ and input variable $\delta(t)$ to respect the 4-DOF dynamics of the parafoil, leading the solution to be dynamically feasible. The actuation faces a physical saturation, and has to be bounded between $-\dot{\psi}_m$ and $\dot{\psi}_m$ as depicted by the constraint (4c). The constraint (4d) shows that the initial states' values y_0 are imposed from the navigation sensors. An accurate estimation of the final time t_f is derived in Sect. 2.3.2, decoupling the horizontal and vertical channels of the problem. The parafoil landing problem writes

¹ The uppercase $*$ is used to disambiguate the solution of the problem from its variables.

$$\begin{aligned} & \underset{\mathbf{y}(t), \delta(t)}{\text{minimize}} \alpha_1 \|\mathbf{x}_f\| + \alpha_2 |\psi_f| + \int_0^{t_f} \delta(t)^2 dt && \text{cost function (4a)} \\ & \text{subject to } \dot{\mathbf{y}}(t) = f_{4\text{DOF}}(\mathbf{y}(t)) + \mathbf{B}\delta(t) && \text{dynamics (4b)} \\ & \quad -\dot{\psi}_m \leq \delta(t) \leq \dot{\psi}_m && \text{control input saturation (4c)} \\ & \quad \mathbf{y}_0 \text{ known} && \text{initial conditions, (4d)} \end{aligned}$$

where \mathbf{y}_0 is the state at the initial time $\mathbf{y}(0)$, $\mathbf{x}_f = [x_1(t_f), x_2(t_f)]^T$, and $\psi_f = \psi(t_f)$. As we seek to minimize both the control cost, the final position \mathbf{x}_f , and final heading angle ψ_f , the parafoil landing problem is multi-objective. The parameters α are used to add penalty on the terms in the cost function with

$$\alpha_1 \gg \alpha_2 \gg 1. \tag{5}$$

2.2.2 Feasibility

A feasible solution to an optimization problem is a solution that satisfies all the equality and inequality constraints. We define an ϵ -feasible solution as a solution that violates at most one of the constraints by ϵ , with ϵ being a small numerical value. Although strictly speaking, an ϵ -feasible solution is infeasible, it is still useful in practice. Namely, the modeling errors are reasonably assumed greater than ϵ , which justifies why a violation of ϵ in the constraints is negligible.

The parafoil Problem (4) has always a trivial feasible solution, since there are no state constraints. One can take, for instance, $\delta(t) = 0 \forall t \in [0, t_f]$ and the corresponding state. However, a feasible solution that is also optimal is non-trivial to achieve. The non-convexity of the constraint (4b) prevents the use of a convex solver, known for providing a global solution reliably. The procedure to convexify those constraints is outlined in Sect. 3.1.1. This work focuses on finding a local optimum solution to Problem (4) with a good level of optimality.

2.3 Parameters

2.3.1 Speed

The two most important parameters of the model (1) are the horizontal v and vertical speed r . Because the density of the air decreases with the altitude, both v and r will be altitude dependent and considering this variation will improve the fidelity of the model and subsequently the landing precision. To determine their functions of the altitude, respectively, $v(z)$ and $r(z)$, we assume the lift force constant and equal to the weight along the trajectory. The quantity ρv_a^2 will remain unchanged [14] where $v_a = \sqrt{r^2 + v^2}$. Based on the measure of the speeds and density at the initial altitude z_0 ,

respectively, $v(z_0)$, $r(z_0)$, and $\rho(z_0)$, we have the value of the horizontal and vertical speed at any altitude z

$$v(z) = v(z_0) \sqrt{\frac{\rho(z_0)}{\rho(z)}} \quad \text{and} \quad r(z) = r(z_0) \sqrt{\frac{\rho(z_0)}{\rho(z)}}. \tag{6}$$

This equation provides a way to estimate the speed at any altitude z , given the speed at the altitude z_0 . The speeds $v(z_0)$ and $r(z_0)$ can be measured onboard using a subset of the following sensors: a variometer, an Inertial Measurement Unit (IMU), Global Positioning System (GPS) receiver, Radio Detection and Ranging (RADAR), Laser imaging, Detection, and Ranging (LiDAR), or barometer combined, for instance, with a Kalman filter [6].

2.3.2 Altitude evolution

From Eq.(6), it follows that knowing the speed at any altitude z , we can estimate the speed at any time t . Indeed, we have

$$\begin{aligned} t(z) &= \int_{z_0}^z \frac{1}{r(\eta)} d\eta = 0 + \frac{\int_{z_0}^z \sqrt{\rho(\eta)} d\eta}{r(z_0) \sqrt{\rho(z_0)}} \\ &= \frac{\sqrt{c_\rho}}{r(z_0) \sqrt{\rho(z_0)}} \left[\frac{(1 - z_0 c_z)^{c_f}}{(c_f) c_h} - \frac{(1 - z c_z)^{c_f}}{(c_f) c_z} \right] \end{aligned} \tag{7}$$

with $c_f = c_e/2 + 1$. The relation can also be inverted to recover the corresponding altitude as a function of time. The idea to exchange the time and altitude variable is also used in [36]

$$z(t) = \frac{1}{c_z} \left(1 - \sqrt[cf]{\left[\frac{(1 - z_0 c_z)^{c_f}}{c_f c_z} - t \left(\frac{\sqrt{c_h}}{r(z_0) \sqrt{\rho(z_0)}} \right)^{-1} \right] c_f c_z} \right). \tag{8}$$

Using these relations, we will be able to transform the control $\delta(t)$ as a function of the altitude $\delta(z)$. This is an asset, since the wind is considered known as a function of the altitude and it will contribute heavily to the dynamics of the system. In practice, during the terminal phase, the density does not change dramatically over the altitude covered by the parafoil and the influence of the altitude evolution may become negligible. Taking the density of the air into the computation becomes more relevant at high altitudes. These considerations will allow swapping the time and altitude dependency in the notation. More importantly, we can evaluate this function for the final altitude z_f to get an estimation of the fixed final time of the trajectory

$$t_f = t(z_f) = \frac{\sqrt{c_h}}{r(z_0)\sqrt{\rho(z_0)}} \left[\frac{(1 - z_0 c_z)^{c_f}}{c_f c_z} - \frac{(1 - z_f c_z)^{c_f}}{c_f c_z} \right]. \tag{9}$$

The final time depends only on the current air density and vertical speed, which can both be measured in-flight. An accurate estimation of the time of flight will drive the landing accuracy. Having a variable final time would constitute another non-convexity to get rid of, as shown in [35].

3 Resolution of the non-convex parafoil problem

3.1 Approach

When a problem is convex, it has a clear global minimum and many numerical solvers exist to retrieve this optimal solution efficiently. Indeed, in a convex optimization problem, a local minimum is guaranteed to be global. When possible, it is always judicious to convert a non-convex problem into an equivalent convex counterpart. The parafoil landing problem as introduced in the Problem (4) is non-convex because of the trigonometric functions in the dynamics. Intuitively, a trigonometric function does not exhibit a clear global minimum and should be avoided in numerical optimization. The general idea of the resolution will be to substitute the highly non-convex trigonometric functions by constrained linear functions with. As such, to exploit the convergence speed of the resolution of convex problems, we will solve the problem via a Sequential Convex Programming (SCP) Algorithm similarly as in [26]. Although the solution of the SCP is not guaranteed to be globally optimal, SCP is a heuristics used to find a solution with a good level of optimality in a quick time, when compared to non-convex solvers. For every iteration, we solve a convexified version of the problem, linearized around the previous solution. We initialize with a dynamically feasible zero-input trajectory and the problem is then sequentially solved until convergence.

At each iteration, we have a convex problem, which can be solved in polynomial time. The relative value of the cost function between two consecutive iterations is chosen as a stopping criterion for the iterative Algorithm. However, the parafoil landing problem is continuous and its solution has an infinite dimension. It is finally shown how to discretize the problem. By constraining the input function $u(t)$ to be piece-wise linear, we limit the number of parameters needed to describe the input and state functions.

3.1.1 Input substitution

To form the convexified problem, we introduce a change of variables and a quadratic constraint to remove the inherently

non-convex trigonometric functions. A similar method was used in [7]. Instead of optimizing over $\delta(t)$ and $\psi(t)$, we introduce a new variable $u(t)$ and will adapt the constraints and cost function accordingly. We define

$$u(t) = \begin{bmatrix} v(t) \cos \psi(t) \\ v(t) \sin \psi(t) \end{bmatrix} = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} \quad \text{and} \quad \|u(t)\| = v(t), \tag{10}$$

where we simplify the notation $v(t) = v(z(t))$. Conversely, we have the following relation for the heading angle $\psi(t) = \text{atan2}(u_1(t), u_2(t))$, and we have

$$\delta(t) = \dot{\psi}(t) = \frac{u_2(t)\dot{u}_1(t) - u_1(t)\dot{u}_2(t)}{\|u(t)\|^2}. \tag{11}$$

It follows that $|\delta(t)| = \frac{\|\dot{u}(t)\|}{\|u(t)\|^2}$. Likewise, we define the state vector $x(t) = [x_1(t), x_2(t)]^T$. The parafoil landing Problem (4) is equivalent to

$$\underset{x(t), u(t)}{\text{minimize}} \alpha^T \begin{bmatrix} \|x\|_f \\ u_1(t_f) \end{bmatrix} + \int_0^{t_f} \frac{\|\dot{u}(t)\|^2}{\|u(t)\|^2} dt \quad \text{cost function} \tag{12a}$$

$$\text{subject to } \dot{x}(t) = u(t) + w(z) \quad \text{dynamics} \tag{12b}$$

$$\|u(t)\| - v(t) = 0 \quad \text{input constraint} \tag{12c}$$

$$\frac{\|\dot{u}(t)\|}{\|u(t)\|} \leq \dot{\psi}_m \quad \text{input saturation} \tag{12d}$$

$$x_0 \text{ and } u_0 \text{ fixed} \quad \text{initial conditions,} \tag{12e}$$

where the parameters x_0 and u_0 are inferred from Problem (4). Namely, $u_0 = [v_0 \cos \psi_0, v_0 \sin \psi_0]^T$. The solution of Problem (4) $\{\delta^*(t), y^*(t)\}$ can be recovered from the solution of the Problem (12) $\{u^*(t), x^*(t)\}$ using the following transformation:

$$\delta^*(t) = \frac{\partial}{\partial t} \text{atan2}(u^*(t)) \quad y^*(t) = \begin{bmatrix} x^*(t) \\ \text{atan2}(u^*(t)) \\ r(z(t)) \end{bmatrix}. \tag{13}$$

The estimation of the wind profile $w(z)$ goes beyond the scope of this article. As emphasized in [7], it is important to consider the wind proactively rather than reactively to achieve better landing precision performances.

3.1.2 Discretization

Standard numerical tools such as [9] require the variables of the optimization problem to be real numbers. To convert the continuous problem into a discrete counterpart, we define a temporal mesh and express the cost function, the dynamical equations, and the constraints as a function of the state and

input variables evaluated on the temporal nodes. Motivated by [18], we choose to constrain the input function to the following format:

$$\mathbf{u}(t) = \lambda_k^-(t)\mathbf{u}(t_k) + \lambda_k^+(t)\mathbf{u}(t_{k+1}) \quad \forall t \in [t_k, t_{k+1}), \quad (14)$$

which interpolates linearly the value between $\mathbf{u}(t_k)$ and $\mathbf{u}(t_{k+1})$ with

$$\lambda_k^-(t) = \frac{t_{k+1} - t}{\Delta t} \quad \lambda_k^+(t) = \frac{t - t_k}{\Delta t}. \quad (15)$$

In doing so, we restrict the search space to the set of piece-wise-affine functions. Although Δt was selected constant, the algorithm could easily be modified to account for a variable step size, potentially denser closer to the landing site. The drastic reduction of the dimension of the problem introduced by this discretization results in an approximated solution. Given the nature of the problem, its solution is smooth and well fitted by a piece-wise linear function provided that the mesh is dense enough. In practice, increasing iteratively the number of nodes, the quality of the solution did not improve further beyond 40 nodes, indicating the convergence to the continuous problem's solution for a fixed final time. Based on this observation, $N = 31$ was chosen as the number of nodes for the discretization.

In the following, to simplify the notation, the subscript will refer to the corresponding time value, such as, $\mathbf{x}_k = \mathbf{x}(t_k)$ and $\mathbf{u}_k = \mathbf{u}(t_k)$. Using this input format and skipping the algebraic details, we can derive an approximation of the integral in the objective function (12a) as

$$\int_0^{t_f} \frac{\|\dot{\mathbf{u}}(t)\|^2}{\|\mathbf{u}(t)\|^2} dt \approx \sum_{k=0}^{N-2} \frac{\|\mathbf{u}_{k+1} - \mathbf{u}_k\|^2}{v_k^2 \Delta t}, \quad (16)$$

where we assume $\|\mathbf{u}(t)\| = v_k \quad \forall t \in [t_k, t_{k+1})$ and $\dot{\mathbf{u}}(t) \approx \frac{\mathbf{u}_{k+1} - \mathbf{u}_k}{\Delta t} \quad \forall t \in [t_k, t_{k+1})$. Likewise, we have an approximation of the rate of change of the heading angle as a quadratic function using the substituted input variables. In discrete time, the constraints (12d) is approximated with

$$\frac{\|\mathbf{u}_{k+1} - \mathbf{u}_k\|}{v_k \Delta t} \leq \dot{\psi}_m. \quad (17)$$

We discretize the dynamics without loss of precision using the state-transition matrix [26]. In the time-invariant case, we can recover its explicit formulation. With piece-wise affine input from (14), the dynamics constraint (12b) can be expressed as

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}^-\mathbf{u}_k + \mathbf{B}^+\mathbf{u}_{k+1} + \mathbf{w}_k, \quad (18)$$

where \mathbf{w}_k represents the integrated influence of the wind between t_k and t_{k+1} , and the state-transition matrices are

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{B}^- = \frac{\Delta t}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \mathbf{B}^+ = \frac{\Delta t}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (19)$$

3.1.3 Linearization of constraints

For the problem to be convex, the last step is to linearize the following non-convex equality constraint:

$$\|\mathbf{u}(t)\| - v(z) = 0. \quad (20)$$

This equality is replaced by two inequalities

$$-\varepsilon \leq \|\mathbf{u}(t)\| - v(z) \leq \varepsilon, \quad (21)$$

where the first one is non-convex and the second one is convex. This approximation will also introduce a ε level of infeasibility in the solution of the original problem. Relaxing an equality constraint by two inequalities is a common method to avoid numerical infeasibilities. The first inequality is linearized using a first-order truncated Taylor series expansion around a known previous iteration $\bar{\mathbf{u}}(t)$, and it gives the following approximated linear constraint which is valid for $\bar{\mathbf{u}}(t) \approx \mathbf{u}(t)$:

$$\|\bar{\mathbf{u}}(t)\| - v(z) + \frac{\bar{\mathbf{u}}(t)^\top}{\|\bar{\mathbf{u}}(t)\|}(\mathbf{u}(t) - \bar{\mathbf{u}}(t)) \geq -\varepsilon \quad (22)$$

Knowing that $\bar{\mathbf{u}}(t)^\top \bar{\mathbf{u}}(t) = \|\bar{\mathbf{u}}(t)\|^2$, we can reformulate as

$$\frac{\bar{\mathbf{u}}(t)^\top \mathbf{u}(t)}{\|\bar{\mathbf{u}}(t)\|} - v(z) \geq -\varepsilon. \quad (23)$$

Finally, we have two convex inequalities

$$\frac{\bar{\mathbf{u}}(t)^\top \mathbf{u}(t)}{\|\bar{\mathbf{u}}(t)\|} - v(z) \geq -\varepsilon \quad (24a)$$

$$\|\mathbf{u}(t)\| - v(z) \leq \varepsilon, \quad (24b)$$

where ε is close to 0. Figure 3 shows how the non-convex constraint (20) is replaced by two convex constraints, which automatically ensures $\bar{\mathbf{u}}(t) \approx \mathbf{u}(t)$.

Remark Using this method, we allow for a small constraint violation, which is a common heuristic. In general, one needs to add a linearization tubes to ensure that the linearized constraint approximates well the original nonconvex constraint. However, in this specific case, the linearization validity is automatically ensured geometrically. It explains why we can guarantee a small level of infeasibility and why this heuristics works especially well for the parafoil problem considered.

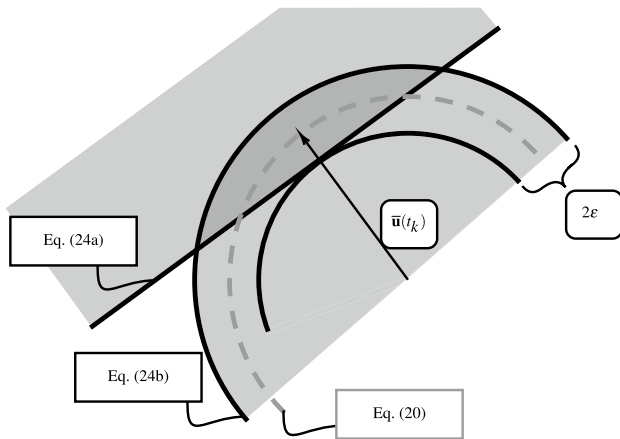


Fig. 3 The intersection of the convex regions, (24a) and (24b) (grey areas), shows an approximation of the non-convex constraints (20) (dashed grey line) (valid close to $\bar{u}(t)$ for $t = t_k$).

3.2 Algorithm and convex programming

3.2.1 Two stages' description

Based on all the previous transformations, we can transform the Problem (12) into a discrete convex problem solved for each iteration. More specifically, the convex problem can be described as a Second-Order Cone Program (SOCP). Using \bar{u}_k as the solution of the problem at the iteration (n), we solve the following problem at the iteration (n+1):

$$\underset{x_k, u_k}{\text{minimize}} J(x_k, u_k) = \alpha^T \left[\frac{\|x_f\|}{u_1(t_f)} \right] + \sum_{k=0}^{N-2} \frac{\|u_{k+1} - u_k\|^2}{v_k^2 \Delta t} \quad \text{cost function} \quad (25a)$$

$$\text{subject to } x_{k+1} = Ax_k + B^- u_k + B^+ u_{k+1} + w_k \quad \text{dynamics} \quad (25b)$$

$$\frac{\bar{u}_k^T u_k}{\|\bar{u}_k\|} - v_k \geq -\epsilon \quad \text{substitution input} \quad (25c)$$

$$\|u_k\| - v_k \leq \epsilon \quad \text{substitution input} \quad (25d)$$

$$\frac{\|u_{k+1} - u_k\|}{v_k \Delta t} \leq \dot{\psi}_m \quad \forall k \quad \text{heading angle rate} \quad (25e)$$

$$x_0 \text{ and } u_0 \text{ known} \quad \text{initial conditions.} \quad (25f)$$

The cost function $J(x_k, u_k)$ is evaluated between each iteration and the convergence of the SCP is assumed when its variation reaches a chosen threshold ϵ_j . The SCP based on (25) will yield a solution with a small dynamical infeasibility ϵ , because of the approximation introduced in Eq.(21). To remove this small infeasibility, once the SCP has converged

a first time, we consider ϵ to be a decision variable, add a new constraint $\epsilon \geq 0$ and a new term $\alpha_3 \epsilon$ to the cost function which will minimize the small dynamical infeasibility ϵ [8]. As a second stage of the Algorithm, the convex problem to be solved iteratively until convergence is the following:

$$\begin{aligned} &\underset{x_k, u_k, \epsilon}{\text{minimize}} J(x_k, u_k) + \alpha_3 \epsilon && \text{cost function} \quad (26a) \\ &\text{subject to Eq. (25b)} \rightarrow \text{Eq. (25f)} && \text{identical} \quad (26b) \\ &\epsilon \geq 0 && \text{penalty} \quad (26c). \end{aligned}$$

Basically, we solve two different but very similar SCP, where the first one gives an initial guess to the second one, as suggested in [13]. The second step tends to the idea of exact penalty functions [8]. The guidance Algorithm is shown below

Algorithm 1: SCP-based guidance for a parafoil

```

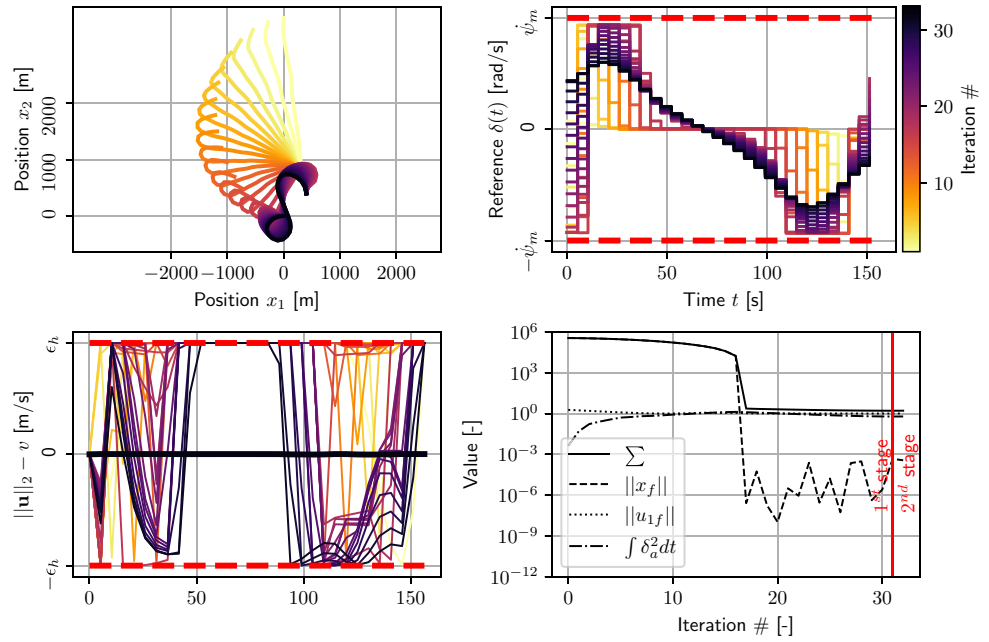
Initialization of  $\epsilon, N, \alpha, c_h, c_p, c_e, A, B^-$  and  $B^+$  % mission independent;
Initialization of  $\psi_m$ ; % mission dependent;
Initialization of  $t_f, t_k, z_k, v_k, w_k, u_0, x_0$  based on the navigation sensors;
Initial guess of the trajectory  $\bar{u}_k, n=0$ ;
while cost has not converged do
    Resolution of the problem (25);
    Update  $\bar{u}_k$  with the new solution at iteration (n);
    n = n+1;
end
while cost has not converged do
    Resolution of the problem (26);
    Update  $\bar{u}_k$  with the new solution at iteration (n);
    n=n+1;
end
Output  $u_k^* = \bar{u}_k$  and  $x_k^*$  as the solution of the Problem (12);
Evaluate Eq.(13) to recover the solution of Problem (4);
Send the solution  $\delta^*(t)$  and  $y^*(t)$  to the controller function;
Result: Feasible reference trajectory for the guidance problem
    
```

3.2.2 ϵ -Feasible sequential convex programming for the parafoil problem

Most sequential convex programming implementations does not provide a feasible solution of the non-convex problem at each iteration (even though at each iteration, we solve a convex problem whose global optimum can be recovered efficiently). Especially, in [17], the use of artificial input is stressed as a mean to avoid numerical infeasibility and dynamic feasibility can only be recovered at convergence, if ever. This artificial input, or slack variable, on the dynamics has no physical meaning and yields a dynamically infeasible trajectory at intermediate iterates. In this contribution, however, we can prove by induction that each iterates yields an ϵ -feasible solution of the non-convex Problem (4).

Lemma 1 *Each iteration of the Algorithm contains a convex problem whose optimum solution is ϵ -feasible for the non-convex Problem (4).*

Fig. 4 Sequential iterations of the Algorithm. (top left) State trajectory \mathbf{x} for each iterate. (top right) Reference $\delta(t)$ constrained between $\dot{\psi}_m$ and $-\dot{\psi}_m$ for each iterate. (bottom left) Level of infeasibility for each iterate, bounded by $-\epsilon$ and ϵ . (bottom right) Value of each term of the cost function across iterates.



Proof First, we can always find an feasible solution for the non-convex Problem (4), at the iteration $n = 0$ using a zero-input trajectory (see Sect. 2.2.2). This feasible solution is by definition also ϵ -feasible. The convex optimization problem of the first iteration will then be feasible by construction.

Second, if we assume that the convex problem at the iteration (n) is feasible, we can also show that the convex problem at the iteration (n+1) is also feasible. For instance, we can always take $\mathbf{u}_k = \bar{\mathbf{u}}_k$ in Problem (25), since $\bar{\mathbf{u}}_k$ is known and we can verify that all constraints are satisfied

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B^-\bar{\mathbf{u}}_k + B^+\bar{\mathbf{u}}_{k+1} + \mathbf{w}_k \tag{27a}$$

$$\frac{\bar{\mathbf{u}}_k^T \bar{\mathbf{u}}_k}{\|\bar{\mathbf{u}}_k\|} - v_k \geq -\epsilon \tag{27b}$$

$$\|\bar{\mathbf{u}}_k\| - v_k \leq \epsilon \tag{27c}$$

$$\frac{\|\bar{\mathbf{u}}_{k+1} - \bar{\mathbf{u}}_k\|}{v_k \Delta t} \leq \dot{\psi}_m \quad \forall k. \tag{27d}$$

Equation (27a) is satisfied, since there is no constraints on \mathbf{x}_k . Equations (27b)–(27d) are satisfied, since the iteration (n) is feasible by assumption. This shows recursive feasibility of the Algorithm by induction.

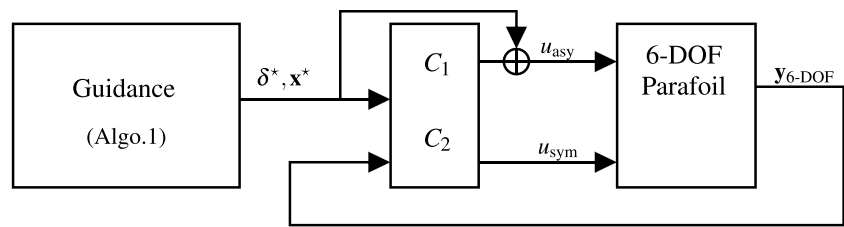
Finally, by construction, a feasible solution to the convex Problem (25) always provides a ϵ -feasible solution of Problem (4). In particular, the optimum solution of the convex Problem (25) is also ϵ -feasible solution of Problem (4). This completes the proof. \square

Furthermore, as shown in Fig. 3, there are several ϵ -feasible solutions at the iteration (n+1) as long as $\epsilon > 0$. This proves that all the iterates will yield a linearly decreasing cost function [10, 20, 22]. In the simulations, each iteration of the Algorithm yields a ϵ -feasible solution, as depicted in Fig. 4. Among all the ϵ -feasible trajectories, the optimizer will yield the one with the lowest cost function. If the mission real-time constraints require guarantees on the refresh rate of the reference trajectory provided by the Algorithm, one can always output the result of the last iteration, which will be sub-optimal but ϵ -feasible and, therefore, useful. Finally, [22] stresses that if the Algorithm converges to a feasible solution, then that solution is also locally optimum.

3.2.3 Numerical parameters

In practice, the selection of ϵ is driven by the following trade-off. When ϵ is small, the convergence of the SCP is slower, but the resulting solution is more accurate. Oppositely, when ϵ is larger, fewer iterations will be needed and the resulting solution will be more dynamically accurate. It determines the quality of the ϵ -feasible solution if we were to stop the Algorithm before it can switch to the second stage. Choosing $\epsilon \approx v_0/200$ for the first stage offers a comfortable trade-off. In practice, the second SCP converges usually much quicker. The number of time step N is discussed in Sect. 3.1.2. If we select a higher N , the discrete-time solution will be closer to the continuous-time solution. However, the resulting convex optimization will take longer to be solved by the numerical solver.

Fig. 5 Architecture of the guidance and control strategy



3.2.4 Implementation

To start the iterative process, we choose a constant initial guess for $\mathbf{u}(t)$, propagating from the initial state, such as $\bar{\mathbf{u}}(t) = [v(t) \cos \psi_0, v(t) \sin \psi_0]^\top$. Details of the implementation can be found on https://github.com/antoineleeman/parafoil_SCP_guidance. To reduce the number of iterations required to converge, further development could focus on a better feasible initial guess, such as Dubin's path. To accelerate the convergence, we can also infer an initial feasible trajectory based on a previously computed optimal trajectory with different initial states. Figure 4 shows the successive iterations of the Algorithm constituted with 2 SCP. The top left shows how the trajectory iteratively bends towards the point $[0, 0]^\top$. The shape of the resulting trajectory is not obvious, since it takes into account the wind perturbation. The top right shows that the reference $\delta(t)$ remains within the input constraints for each iteration, a necessary condition for each trajectory to be dynamically feasible. Because of the choice of discretization, $\delta(t)$ is piece-wise constant. We see that towards the end of the iterative process, the reference $\delta(t)$ converges to a fixed function. The bottom left illustrates that each iteration is ε -feasible by construction and that the last iteration achieves exactly $\|\mathbf{u}(t)\| = v$. Finally, the bottom right shows the value of each term of the sum in Eq.(25a). The switch to the second stage arises around 1–2 iterations before the end and we will see that $\|x_f\|$ is the main contributor of the cost for the first half of the iterates when it is still strictly positive.

3.3 Controller

3.3.1 Benchmark model

The guidance Algorithm is applied to a 6-DOF model from [25]

$$\dot{\mathbf{y}}_{6\text{-DOF}}(t) = f_{6\text{-DOF}}(\mathbf{y}_{6\text{-DOF}}(t), [u_{\text{asy}}(t), u_{\text{sym}}(t)]^\top), \quad (28)$$

where $\mathbf{y}_{6\text{-DOF}}(t)$ is the state of the 6-DOF parafoil and $u_{\text{asy}}(t)$, $u_{\text{sym}}(t)$ represent, respectively, the asymmetrical and symmetrical deflections of the trailing edges. The symmetrical deflection controls the rate of fall, whereas the asymmetrical controls the rate of turn of the parafoil. The longitudinal

control compensates for the error in the estimation of both the time of flight and the rate of fall. The state is assumed to be measured perfectly the navigation sensors of the parafoil.

3.3.2 Guidance and control strategy

The controller is needed to compensate for the modeling approximations, unmodelled perturbations, and noise. The algorithm presented above gives a reference for the input $\delta^*(t)$ and for the position $\mathbf{x}^*(t)$, used as a feed-forward to the controller as illustrated in Fig. 5, where we drop the time dependency.

The control function operates independently on both the lateral and longitudinal channel to ensure that the parafoil has, respectively, a minimum lateral and longitudinal error. Inspired by [25], we have the following asymmetrical deflection for the lateral control:

$$u_{\text{asy}}(t) = \delta^*(t) + C_1(\mathbf{y}_{6\text{-DOF}}(t), \mathbf{x}^*(t)). \quad (29)$$

We have the symmetrical deflection for the longitudinal control

$$u_{\text{sym}}(t) = C_2(\mathbf{y}_{6\text{-DOF}}(t), \mathbf{x}^*(t)). \quad (30)$$

The controller is implemented only for illustrative purposes and further development could focus on an optimally tuned controller. C_1 and C_2 are controllers, respectively, on the cross-track error and the along-track error.

4 Results

The 6-DOF model developed by Rademacher [25] was used as a benchmark of the physical reality to assess the performances of the Algorithm, including the saturations on the actuators. The reference trajectory is applied to the 6-DOF model with 600 different randomly sampled initial conditions \mathbf{x}_0 and ψ_0 from a uniform distribution. The conditions of the simulation are exposed in Table 2.

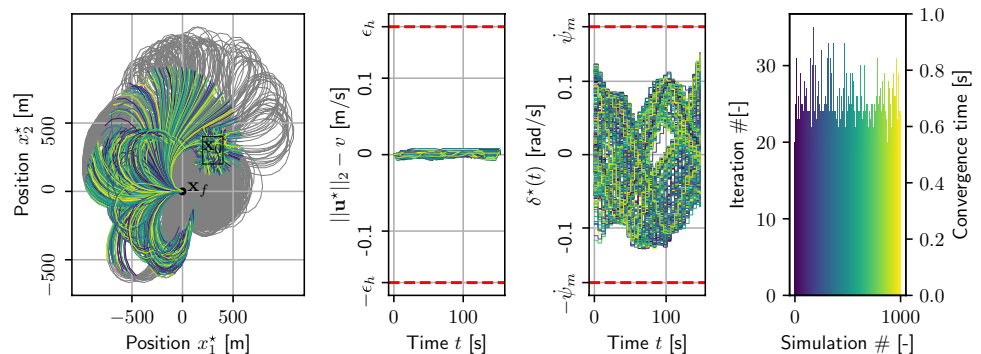
Table 2 Parameters value for the 600 Monte Carlo simulations

Parafoil			Numerical		
Parameter	Value	Unit	Parameter	Value	Unit
v_0	18.5	[m/s]	ϵ_f	0.01	[-]
z_0	1200	[m]	ϵ	0.1	[m/s]
r_0	-7.9	[m/s]	N	31	[-]
ψ_m	0.14	[rad/s]	α_1	100	[-]
\mathbf{x}_0	$[200, 400]^2$	[m]	α_2	10	[-]
ψ_0	$[-\pi, \pi]$	[rad]	α_3	1	[-]

4.1 Convergence

The 600 runs of the Monte Carlo simulation show the consistent convergence of the Algorithm. In Fig. 6, we can see the solution $\mathbf{x}^*(t)$ and $\delta^*(t)$ of the Algorithm for each of the 600 different initial conditions. We can make the following three observations on the four subplots (from left to right). First, each of the 600 realizations of the Algorithm succeeds in finding a feasible trajectory, with a good cost. In this case, since the set of initial conditions \mathbf{x}_0 is relatively close to the desired landing position at $[0, 0]^T$, each trajectory is able to reach that position $\mathbf{x}_f = [0, 0]^T$. The Algorithm converges for any initial conditions, where it always seeks to minimize $\|\mathbf{x}_f\|$. The final angle ψ_f is also close to 0 for each trajectory. Second, as the constraint (20) is non-convex, we approximated with two inequalities (21) and the second stage of the Algorithm aims at minimizing that approximation. The plot shows that the value of the infeasibility on the constraint $\|\mathbf{u}^*\| - v$ is negligible for each of the 600 runs, demonstrating that, using the second stage of the Algorithm, we got rid of the ϵ -infeasibility. Third, we see that the constraint on $\delta(t)$ is satisfied for each of the simulations. Fourth, we see that the number of iterations is rarely above 30, which gives a convergence time to a feasible solution under 1 s for each of the 600 trajectories of the Monte Carlo. The convergence time has been measured on a processor i7 8th Gen, 16Go RAM with the default setting of ECOS [11]. No attempt was

Fig. 6 Monte Carlo simulation with hard constraints in dashed lines. (left) Set of all the trajectories $\mathbf{x}^*(t)$ with a randomly initialized \mathbf{x}_0 inside the box. (Center left) Dynamical infeasibility remaining after convergence for each trajectory. (Center right) Optimal reference $\delta^*(t)$ for each trajectory. (Right) Convergence time for each simulation.



made to improve the numerical code efficiency which can be obtained via specific customization and code adaptation. Eventually, Fig. 6 provides evidence that the Algorithm used provides consistently and rapidly a locally optimal solution to the non-convex Problem (4).

4.2 Performance

In the previous section, we showed that the Algorithm could provide consistently a reference trajectory, often called feed-forward. In this section, we explore the quality of that reference trajectory when applied to a 6-DOF model. Using the same initial conditions as in the previous section, Fig. 7 shows the landing accuracy on a 6-DOF parafoil. In this case, the guidance and control architecture is outlined in Sect. 3.3.2. A single reference trajectory has been computed and tracked, hence the "open-loop guidance" naming. In practice, better results can be achieved when a new trajectory is computed as soon as the parafoil is too far from the reference ("closed-loop guidance"). Using an accurate wind model, Fig. 7 shows that most of the simulation runs see the parafoil landing below 30 m away from the target and below 20 deg difference with the target heading angle. Better results can be achieved with the guidance in closed-loop, with an optimized controller, and with a final angle aligned with the wind direction. Closed-loop guidance is made available because of the low convergence time of the Algorithm and should be implemented in practical applications. Furthermore, one could argue that the Monte Carlo does not have enough simulation to give an accurate estimation of the landing precision. Figure 7(right) intends to disprove this. Indeed, as we increase the number of simulations for the estimation of the average landing distance $\hat{\mu}(\|\mathbf{x}_f\|)$ and the average landing angle $\hat{\mu}(\psi_f)$, the reliability of the estimation goes higher, by the statistical law of large numbers. The same goes for the corresponding standard deviation $\hat{\sigma}(\|\mathbf{x}_f\|)$ and $\hat{\sigma}(\psi_f)$. Therefore, the most useful value is the one corresponding to the highest number of simulations, on the right. However, this estimation was already achieved relatively accurately after 200 simulations, indicating that we do not

Fig. 7 Monte Carlo simulation where the reference $\delta^*(t)$ computed by the Algorithm is applied to a 6-DOF parafoil model. (Left) final position and heading angle with open-loop guidance, lateral and longitudinal control, perfect knowledge of the strong wind. (Right) estimation of the mean and variance of the missed distance and missed heading angle. The horizontal axis corresponds to the number of simulations used for the estimation.

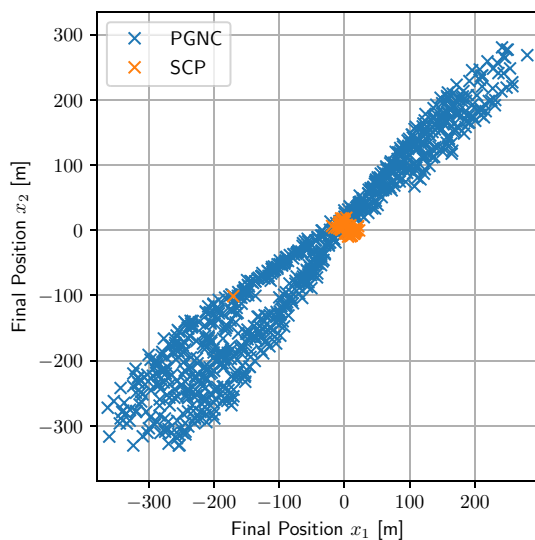
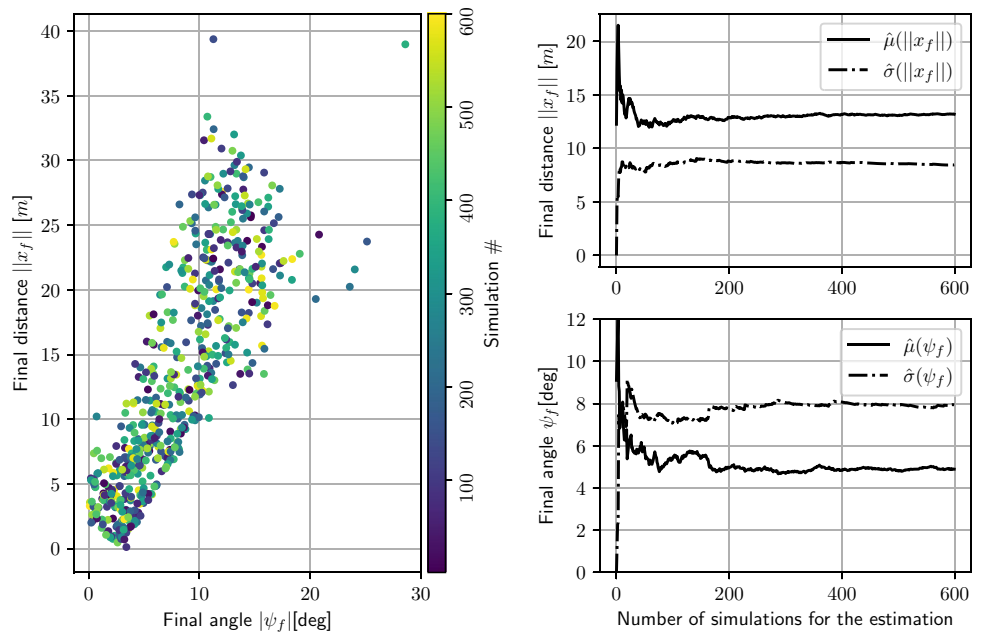


Fig. 8 Comparison of the missed landing distance between the traditional offline guidance 'PGNC' and the online optimization 'SCP' presented in this article, with known wind distribution

need more simulations to capture the true performance of the Algorithm.

4.3 Comparison

The online trajectory generation method is compared against a more traditional method from [31, 32] referenced as Parafoil Guidance, Navigation and Control (PGNC) in the following. The PGNC method is tested for a 4-DOF X-38 parafoil model calibrated with flight data, while the aforementioned Algorithm is used on a comparable 6-DOF

parafoil model, with the same wind model in both cases. With similar initial conditions, we can observe the difference in landing precision in Fig. 8. The final heading angle constraint has been removed for higher consistency and comparability since the EM method does not target a specific heading angle. The dispersion of the landing precision is much smaller for the online optimization method.

5 Conclusion

The parafoil guidance problem has been solved by the application of sequential convex programming. Using a substitution for the trigonometric function and linearization, it is possible to convert the non-convex problem into a sequence of convex problems, suitable for onboard implementation and real-time use. Especially, each iteration yields an ϵ -feasible solution of the physical problem in polynomial time and can be used as a sub-optimal feed-forward trajectory, to comply with the real-time computational requirement. The algorithm takes into account the influence of the altitude-varying density of the air on the aerodynamics and requires only the control saturation level for the parafoil model as a parameter for the model. It also considers the wind distribution and finds a locally optimal solution, minimizing the control energy or the missed landing, or both when possible. The algorithm is tested on a 6-DOF parafoil model. The navigation error is not included in the simulation. Future work could take the measurement noise explicitly into account in the problem formulation. Further tests will include hardware-in-the-loop simulations.

Acknowledgements The authors thank all the TEC-SA division within the European Space Agency for their continuous support, as well as the numerous comments and feedback of Dr. Carlo Alberto Pascucci from Embotech and the precious help from Hans Strauch for the simulation of the guidance PGNC. This work has been carried out during a Young Graduate Trainee Program at the European Space Agency.

Funding Open access funding provided by Swiss Federal Institute of Technology Zurich.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Ben-Tal, A., Nemirovski, A.: Lectures on modern convex optimization. Soc. Ind. Appl. Math. (2001). <https://doi.org/10.1137/1.9780898718829>
- Blackmore, L.: Auton. Precis. Land. Space Rockets. Front. Eng. **46**, 15–20 (2016)
- Blackmore, L., Acikmece, B., Scharf, D.P.: Minimum-landing-error powered-descent guidance for mars landing using convex optimization. J. Guid. Control. Dyn. **33**(4), 1161–1171 (2010). <https://doi.org/10.2514/1.47202>
- Boyd, S.P., Vandenberghe, L.: Convex optimization. Cambridge University Press, Cambridge (2004). <https://doi.org/10.1017/cbo9780511804441>
- Carter, D., Singh, L., Wholey, L., Rasmussen, S., Barrows, T., George, S.: Band-limited guidance and control of large parafoils. AIAA Aerodyn. Decelerator Syst. Technol. Conf. Semin. (2009). <https://doi.org/10.2514/6.2009-2981>
- Carter, D.W., George, S., Hattis, P.D., Mcconley, M.W., Rasmussen, S.A., Singh, L., Tavan, S.: Autonomous large parafoil guidance, navigation, and control system design status. AIAA Aerodyn. Decelerator Syst. Technol. Conf. Semin. (2007). <https://doi.org/10.2514/6.2007-2514>
- Chiel, B.S., Dever, C.: Autonomous parafoil guidance in high winds. J. Guid. Control. Dyn. **38**(5), 963–969 (2015)
- Di Pillo, G., Grippo, L.: Exact penalty functions in constrained optimization. SIAM J. Control Optim. **27**(6), 1333–1360 (1989). <https://doi.org/10.1137/0327068>
- Diamond, S., Boyd, S.P.: CVXPY: A python-embedded modeling language for convex optimization. J. Mach. Learn. Res. **17**(1), 2909–2913 (2016)
- Dinh, Q.T., Diehl, M.: Local convergence of sequential convex programming for nonconvex optimization. In: Diehl, M., Glineur, F., Jarlebring, E., Michiels, W. (eds.) Recent advances in optimization and its applications in engineering, pp. 93–102. Springer, Berlin (2010)
- Domahidi, A., Chu, E., Boyd, S.: ECOS: An SOCP solver for embedded systems. Eur. Control Conf. (2013). <https://doi.org/10.23919/ecc.2013.6669541>
- Handley, P.M., Streetman, B., Neave, M.D., Bergeron, K., Noetscher, G.: Euler elastica terminal parafoil guidance. AIAA Aerodyn. Decelerator Syst. Technol. Conf. (2017). <https://doi.org/10.2514/6.2017-3877>
- Kelly, M.: An introduction to trajectory optimization: how to do your own direct collocation. SIAM Rev. **59**(4), 849–904 (2017). <https://doi.org/10.1137/16m1062569>
- Le Floch, B., How, J.P., Stoeckle, M., Breger, L.: Trajectory planning for autonomous parafoils in complex terrain. AIAA Aerodyn. Decelerator Syst. Technol. Conf. (2017). <https://doi.org/10.2514/6.2017-3220>
- Luders, B., Ellertson, A., How, J.P., Sugel, I.: Wind uncertainty modeling and robust trajectory planning for autonomous parafoils. J. Guid. Control Dyn. **39**(7), 1614–1630 (2016). <https://doi.org/10.2514/1.G001043>
- Luders, B.D., Sugel, I., How, J.P.: Robust trajectory planning for autonomous parafoils under wind uncertainty. In: AIAA Infotech@Aerospace (I@A) Conference (2013)
- Malyuta, D., Reynolds, T.P., Szmuk, M., Lew, T., Bonalli, R., Pavone, M., Acikmece, B.: Convex optimization for trajectory generation. Ann. Rev. Control. arXiv preprint [arXiv:2106.09125](https://arxiv.org/abs/2106.09125) (2021). <http://arxiv.org/abs/2106.09125> (Accepted)
- Malyuta, D., Reynolds, T.P., Szmuk, M., Mesbahi, M., Acikmece, B., Carson, J.M.: Discretization performance and accuracy analysis for the rocket powered descent guidance problem. AIAA Scitech (2019). <https://doi.org/10.2514/6.2019-0925>
- Malyuta, D., Yu, Y., Elango, P., Acikmese, B.: Advances in trajectory optimization for space vehicle control. arXiv preprint [arXiv:2108.02335](https://arxiv.org/abs/2108.02335) (2021). <http://arxiv.org/abs/2108.02335> (Submitted)
- Mao, Y., Szmuk, M., Acikmece, B.: Successive convexification of non-convex optimal control problems and its convergence properties. IEEE Conf. Decis. Control (CDC) (2016). <https://doi.org/10.1109/CDC.2016.7798816>
- Mao, Y., Szmuk, M., Acikmece, B.: A tutorial on real-time convex optimization based guidance and control for aerospace applications. Proc. Am. Control Conf. (2018). <https://doi.org/10.23919/ACC.2018.8430984>
- Mao, Y., Szmuk, M., Xu, X., Acikmece, B.: Successive convexification: A superlinearly convergent algorithm for non-convex optimal control problems. arXiv preprint [arXiv:1804.06539](https://arxiv.org/abs/1804.06539) pp. 1–35 (2018). <http://arxiv.org/abs/1804.06539> (Submitted)
- Mattingley, J., Boyd, S.P.: CVXGEN: a code generator for embedded convex optimization. Optim. Eng. **13**(1), 1–27 (2012). <https://doi.org/10.1007/s11081-011-9176-9>
- Mooij, E., Mazouz, R., Quadrelli, M.B.: Convex optimization guidance for precision landing on titan. AIAA Scitech (2021). <https://doi.org/10.2514/6.2021-1345>
- Rademacher, B.J., Lu, P., Strahan, A.L., Cerimele, C.J.: In-flight trajectory planning and guidance for autonomous parafoils. J. Guid. Control Dyn. **32**(6), 1697–1712 (2009). <https://doi.org/10.2514/1.44862>
- Reynolds, T.P., Malyuta, D., Mesbahi, M., Acikmece, B., Carson, J.M.: A real-time algorithm for non-convex powered descent guidance. AIAA Scitech (2020). <https://doi.org/10.2514/6.2020-0844>
- Simplicio, P., Bennani, S., Marcos, A., Roux, C., Lefort, X.: Structured singular-value analysis of the vega launcher in atmospheric flight. J. Guid. Control Dyn. **39**(6), 1342–1355 (2016). <https://doi.org/10.2514/1.G000335>
- Slegers, N., Brown, A., Rogers, J.: Experimental investigation of stochastic parafoil guidance using a graphics processing unit.

- Control Eng. Pract. **36**, 27–38 (2015). <https://doi.org/10.1016/j.conengprac.2014.12.002>
29. Slegers, N., Costello, M.: Model predictive control of a parafoil and payload system. *J. Guid. Control Dyn.* **28**(4), 816–821 (2005). <https://doi.org/10.2514/1.12251>
 30. Slegers, N.J., Yakimenko, O.A.: Optimal control for terminal guidance of autonomous parafoils. In: 20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, p. 2958 (2009)
 31. Soppa, U., Strauch, H., Goerig, L., Belmont, J.P., Cantinaud, O.: GNC concept for automated landing of a large parafoil. *Aerodyn. Decelerator Syst. Technol. Conf.* (1997). <https://doi.org/10.2514/6.1997-1464>
 32. Strahan, A.: Testing of parafoil autonomous guidance, navigation and control for X-38. *AIAA Aerodyn. Decelerator Syst. Technol. Conf. Semin.* (2003). <https://doi.org/10.2514/6.2003-2115>
 33. Sun, H., Luo, S., Sun, Q., Chen, Z., Wu, W., Tao, J.: Trajectory optimization for parafoil delivery system considering complicated dynamic constraints in high-order model. *Aerosp. Sci. Technol.* (2020). <https://doi.org/10.1016/j.ast.2019.105631>
 34. Sun, H., Sun, Q., Luo, S., Chen, Z., Wu, W., Tao, J., He, Y.: In-flight compound homing methodology of parafoil delivery systems under multiple constraints. *Aerosp. Sci. Technol.* **79**, 85–104 (2018). <https://doi.org/10.1016/j.ast.2018.04.037>
 35. Szmuk, M., Acikmece, B.: Successive convexification for 6-DoF mars rocket powered landing with free-final-time. *AIAA Guid. Navig. Control Conf.* **5**, 5 (2018). <https://doi.org/10.2514/6.2018-0617>
 36. Weinstein, M.J., Streetman, B., Neave, M.D., Bergeron, K., Noetscher, G.: Trajectory optimization via particle swarms for robust parafoil guidance. *AIAA Guid. Navig. Control Conf.* (2018). <https://doi.org/10.2514/6.2018-1855>
 37. Yakimenko, O.A.: Precision aerial delivery systems: modeling, dynamics, and control, vol. 248. American Institute of Aeronautics and Astronautics Inc, Reston (2015). <https://doi.org/10.2514/4.101960>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.