Jet Propulsion Laboratory
California Institute of Technology

POLITECNICO
MILANO 1863

Guidance & Control of a Parafoil-Based Landing on Titan

Master of Science in Mechanical Engineering

Author:
Giacomo Bonaccorsi
883138

Supervisor:
Prof. Francesco Braghin,
Politecnico di Milano

Mentor:
Dr. Marco B. Quadrelli,
Jet Propulsion Laboratory

Academic Year 2018/2019

Acknowledgements

First of all I want to thank my mentor at JPL, Dr. Marco Quadrelli, who never failed to give me valuable advice and guided me throughout my whole internship.

I also want to express my deep gratitude to Prof. Francesco Braghin, for the opportunities he gave me and for his constant support despite the distance.

Immense gratitude goes to my family, who supported me throughout my academic career: all of this would not have been possible without you.

I am grateful for the constant and concrete presence of my girlfriend, who always supported me even from far away.

To my close friends back home and to my amazing lab mates: your companionship has been very important during this journey.

This research was carried out at Jet Propulsion Laboratory, California Institute of Technology, during the internship sponsored by JVSRP (JPL Visiting Student Research Program) and NASA.

Abstract

The aim of this work is to address the problems of and investigate appropriate solutions to the Guidance & Control aspects of autonomous Precision Aerial Delivery Systems (PADS), which in the future could lead to novel space exploration approaches as well as to noticeable cost reduction. The investigated application, in particular, focuses on a Titan landing which would be particularly relevant from a scientific point of view, due to both Titan's peculiar geophysical characteristics and to the fact that PADS have never been used for planetary landing.

PADS dynamics has been investigated, with the adoption of both low-fidelity (three degrees of freedom) and higher-fidelity (six degrees of freedom) dynamic models. The six-DOF model has been implemented in DSENDSEdu, a high-fidelity development and testing environment for spaceflight systems, to efficiently simulate the desired conditions for the whole descent and landing phase. This model was also used to perform some preliminary studies on the feasible glide range, expected wind drifts and average descent speed, as well as to investigate details of the final flare maneuver.

The Guidance & Control aspects are macroscopically divided into three parts. The first part addresses point-to-target control techniques that don't require previous path planning: a Proportional-Derivative (PD) control and a T-approach are investigated. The goal of the second part is to address the computation of optimal trajectories for the terminal part of the descent to overcome the limitations of the previously-investigated techniques. Finally, in the last part, trajectory tracking techniques (both optimal and sub-optimal) are investigated to follow the previously-generated trajectories.

The PD controller and T-guidance are particularly relevant in situations where no offline motion planning is desired and/or computational resources are limited. In particular, for the T-approach an expected wind drift is constantly taken into account to generate virtual waypoints in real-time: these waypoints guide the parafoil to an area close to the Intended Point of Impact, perform eight-patterns for energy management, and then

to perform a final landing maneuver against wind.

Motion planning considers two distinct phases. An initial homing phase is considered first, during which a minimum-time path is followed to reach an area above the target as quickly as possible as to maximize the residual altitude. The optimal trajectory for the final portion of the descent is obtained during this first phase by solving the Euler-Lagrange problem typical of the optimal control theory. If the complete state and control action history is needed, a six-DOF model must be integrated in its entirety. If, instead, only trajectory waypoints are of interest, a three-DOF model can be used to greatly reduce integration time.

In cases where the complete state and control actions history is available, it is possible to perform trajectory tracking by means of a Linear-Quadratic Regulator (LQR) which, through the use of weight matrices, gives the appropriate feedback control action in the neighborhood of the (possibly optimal) nominal trajectory, provided small perturbations are present. If the whole state and input vectors are not available for each time instant (i.e. only waypoints to be followed are given), it is not possible to use a LQR and other waypoint-tracking techniques must be introduced. A Model Predictive Control (MPC) was implemented for this eventuality. Given a sequence of spatial waypoints, the MPC allows to accurately track them by linearizing the system (whose state matrices have been previously computed offline in symbolic form) at every time step and computing the optimal control action, given a desired time horizon which depends on the available computational power.

Finally, a reduced-order Dynamic Programming (DP) implementation which makes use of successive grid-refinement has been investigated for the terminal portion of the descent. This would allow, through offline calculations, to obtain the optimal feedback action for every point close to the landing site.

Contents

List of Tables

Chapter 1

Introduction

The aim of this work is to address the Guidance & Control aspects of a new way to approach space exploration, based on the concept of delivering payloads by means of Autonomous Precision Aerial Delivery Systems (PADS). The investigated application, in particular, focuses on a Titan landing which would be particularly relevant from a scientific point of view, due to both Titan's peculiar geophysical characteristics and to the fact that PADS have never been used for planetary applications. Improvements and advances in this field could lead to drastic reductions in cost as well as to improvements in landing precision.

Landing on Titan poses many unprecedented challenges [1]. To start with, the aerodynamic performance is different than on Earth, and the presence of fast-blowing winds present in the higher atmosphere might initially blow away the parafoil from the Intended Point of Impact (IPI). The only data available concerning atmospheric properties comes form the Cassini-Huygens mission and is not available for every latitude-longitude-altitude combination: the control architecture must then be robust to external disturbances. Many Earth-bound autonomous systems rely on GPS data, which of course is not available and cannot be employed for autonomous navigation on Titan. Additionally, the on-board computer would rely on Terrain Relative Navigation (TRN) to direct the autonomous PADS towards the appropriate landing site, which needs sufficient surface data to be effective. In the case of Titan though the topography comes from radars, with a resolution limited to about 350 meters per pixel. This, combined with the fact that Titan's atmosphere is inherently thick, poses an additional challenge when it comes to vision-based algorithms.

1.1 Autonomous Precision Aerial Delivery Systems

Precision Aerial Delivery Systems (PADS), despite being underactuated, provide a unique capability for accurate airdrops, and substantial investigation is being carried out in many fields where they might be employed. While most PADS used by civilians are human-controlled, extended research is being conducted in the military field in an attempt to improve landing precision and accuracy of autonomous solutions. The U.S. military, for example, is interested in developing autonomous PADS which range from about 100 lbs (or about 45 kg) all the way up to 10,000 lbs (about 4500 kg). NASA started investigating similar problems more than 20 years ago to figure out the landing precision that could be achieved (Fig. 1.1, as an example, shows one of the early tests conducted in 1992). In a similar manner, private companies are also investigating the many advantages of autonomous delivery/recovery. SpaceX, for example, just recently started developing algorithms for recovering rockets fairings, which would lead to additional cost reductions. Parafoils are also used for troop resupply. sensor placement, urban welfare reconnaissance and similar operations [2].

Figure 1.1: NASA Spacewedge subscale model being delivered using a steerable parafoil in 1992 during a flight test.

Despite the economic convenience and ease of deployment, PADS have a few peculiarities that lead to many challenges from a control perspective. Unlikely conventional powered vehicles, for

example, most parafoils don't have the ability to ascend: this, of course, implies that only one attempt can be made to land at a desired position. Control is usually performed by deflecting the rear flaps which, in turn, accelerate the canopy-payload system and have a direct impact on the system attitude. Another complication is the wind profile whose velocities at high altitude exceed, more often than not, the parafoil speed relative to the ground [3]. While wind only accounts for a small percentage of the speed on airplanes, it has a considerable effect on the landing position of a parafoil, which might turn out to be off by several kilometers. Furthermore, landing is usually to be performed upwind to limit payload touchdown speed and avoid (or limit) the risk of payload roll-over. Thanks to the Cassini-Huygens mission we have a basic knowledge of Titan's atmosphere (i.e. density, chemical composition, winds), although many aspects still remain unknown. For this reason it is clear that a control strategy should be as robust as possible in case the actual conditions were different from the expected ones.

Preliminary results indicate that precision control of PADS can be achieved in a dense atmosphere, provided enough control authority and enough knowledge on the atmospheric conditions.

1.2 Landing on Titan

Titan is of high scientific interest and significance mainly due the presence in its atmosphere of liquids and gases very similar to the ones present on Earth. The goal of a mission would also be to investigate the possible presence of underground oceans on water. Furthermore, it is believed that methane and ethane lakes might be connected underneath the surface. Fig. 1.2 shows an overview of Titan's northern lakes and seas. This image was obtained in multiple operating modes with resolutions of $0.3 - 1.5$ km, $2 - 10$ km, and $40 - 200$ km. False coloring is used to distinguish the features: bodies of liquid hydrocarbon are indicated using blue/black, while dry land is brown. Fig. 1.3 highlights the targeted landing area in more detail (with a site close to the Maracaibo Lacus being the goal at the time of writing).

Titan's atmosphere is about five times thicker than Earth's. As a consequence, smaller parafoils can be adopted for landing on its surface and the complete Entry, Descent & Landing (EDL) phase will be relatively slow, meaning that there is enough time for winds to change considerably.

Some specific requirements of the project are still to be defined completely at the time of the writing, while other are covered by the International Traffic in Arms Regulations (ITAR). The following high-level requirements will then be assumed throughout this work:

- Starting altitude (after canopy complete inflation) of 40km above Titan's surface.
- Target altitude and longitude assumed to be zero, although this does not influence the study if the wind model is considered constant.
- Payload mass of 200kg.

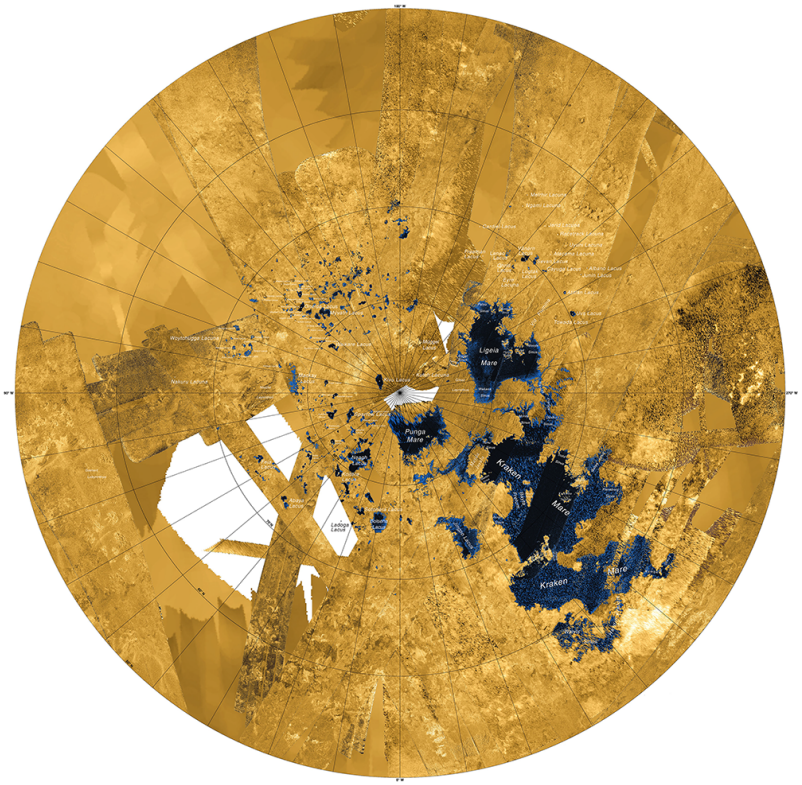


Figure 1.2: Titan's North polar lakes and seas (data from the Cassini Titan RADAR Mapper).

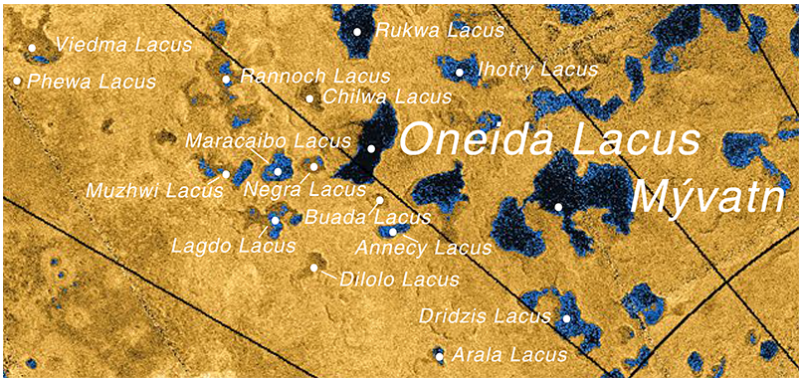


Figure 1.3: Detail of the targeted landing area.

Chapter 2

Titan properties and definitions

The aim of this Chapter is to present the main properties of Titan and provide the necessary definitions of the angles and frames of reference that will be referred to throughout the whole work. Table 2.1 shows Titan's most relevant properties. Note, in particular, how much the density changes when moving from a 40 km altitude to ground level.

Property	**Value**	**Unit**
Angular velocity	$4.056 \cdot 10^{-6}$	rad/s
Density at 40 km	0.7	kg/m^3
Density on surface	5.43	kg/m^3
Gravity acceleration	1.352	m/s^2
Equatorial radius	2575	km
Mass	$1.345 \cdot 10^{23}$	kg

Table 2.1: Titan properties.

2.1 Atmospheric models

Only a few publications have analyzed in detail the atmospheric properties of Titan in the past. A thorough study about the atmospheric structure can be found in [4], which considers and compiles data from the Cassini-Huygens mission to generate an empirical model of mass densities and temperatures, as well as in [5] which, on the other hand, focuses on visibility and downlink Direct-to-Earth (DTE) by analyzing seasonal and latitudinal variations in wind. The present work considers a wind and density profile obtained by Jet Propulsion Laboratory for the site of interest.

2.1.1 Density profile

The model used to compute the density (expressed in kg/m^3) at a generic altitude h is:

$$\rho = 5.43\exp\left(\frac{-0.512h}{1000}\right) \tag{2.1}$$

where h is expressed in meters. The density profile is shown in Fig. 2.1.

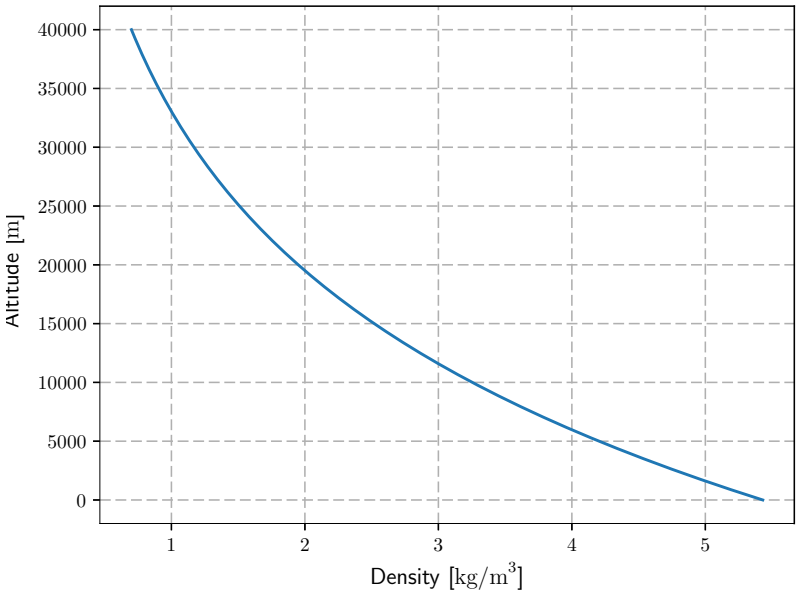


Figure 2.1: Density profile.

2.1.2 Wind profile

Wind is computed with a zonal model which gives the average, high-intensity component through the following equation:

$$W = \frac{W_{300}}{1 + \exp\left(\frac{h_0 - h}{L}\right)} \tag{2.2}$$

Where W_{300} is the wind speed at 300 km, h_0 the reference altitude and L the length scale. The coefficients are as follows, depending on wind conditions:

The nominal wind profile is shown in Fig. 2.2.

Wind profile	W_{300}	h_0	L
Nominal	22	35	8
Maximum	50	38	11
Minimum	-3	0	1

Table 2.2: Wind profile coefficients for different blowing conditions.

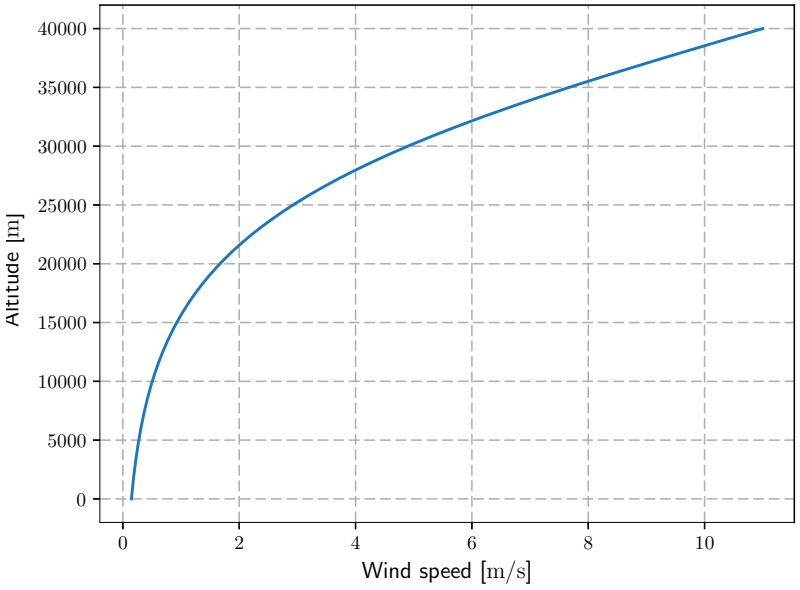


Figure 2.2: Wind profile.

2.2 Frames of reference

Strictly speaking, given a non-null angular rotation of Titan, a planet-centered inertial frame should be defined as the one having:

- Origin at the center of mass of the planet
- z axis along the planetary axis of rotation
- x axis laying on the equatorial plane and pointing towards the vernal equinox
- y axis following the right-hand rule

Analogously, a planet-centered planet-fixed frame could be introduced that has the following properties:

- Origin at the center of mass of the planet

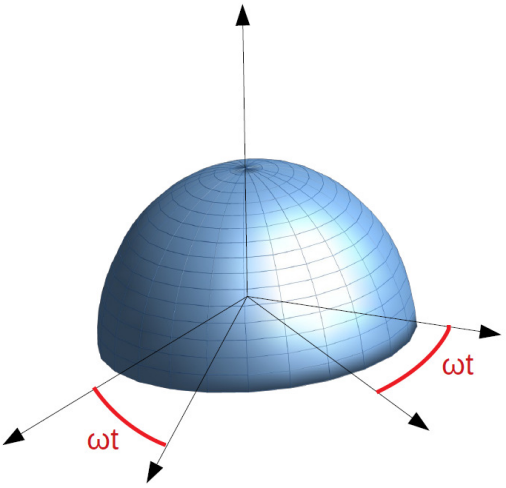


Figure 2.3: Transformation between planet inertial frame and planet-centered planet-fixed frame.

- z axis along the planetary axis of rotation
- x axis intersecting the equatorial plane and the reference meridian
- y following the right-hand rule

The transformation between the two systems described above is described in Eq. 2.3 and is shown in Fig. 2.3

$$\boldsymbol{R} = \begin{bmatrix} \cos(\omega t) & \sin(\omega t) & 0 \\ -\sin(\omega t) & \cos(\omega t) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

where ω is the angular velocity of the planet ($4.056 \cdot 10^{-6}$ rad/s) in the case of Titan. Since ω is very low, Titan rotation is usually neglected and the planet-centered planet-fixed frame is considered to be the inertial frame $\{I\}$.

Besides the $\{I\}$ frame presented above (used when dealing with a spherical planet), three main frames of reference will be used throughout this work, presented hereafter.

North-East-Down frame $\{NED\}$

- Origin: arbitrary, but usually set to coincide with the Intended Point of Impact (IPI)
- x axis: positive in the direction of true North
- y axis: positive in the direction of East

- z axis: positive towards the center of the planet

Note that, although the equations of motion will be expressed in this reference frame, a East-North-Up $\{ENU\}$ frame following the opposite convention will be used for the software implementation on DSENDSEdu.

Body-fixed frame $\{B\}$

- Origin: PADS center of gravity
- x_b axis: positive along the longitudinal axis of the PADS on its plane of symmetry
- z_b axis: perpendicular to the x_b axis, in the plane of symmetry of the PADS and positive pointing down
- y_b axis: perpendicular to the x_b-z_b plane, positive determined by the right-hand rule

Again, due to the $\{ENU\}$ convention, the z_b axis will be pointing up in DSENDSEdu, with y_b redefined accordingly.

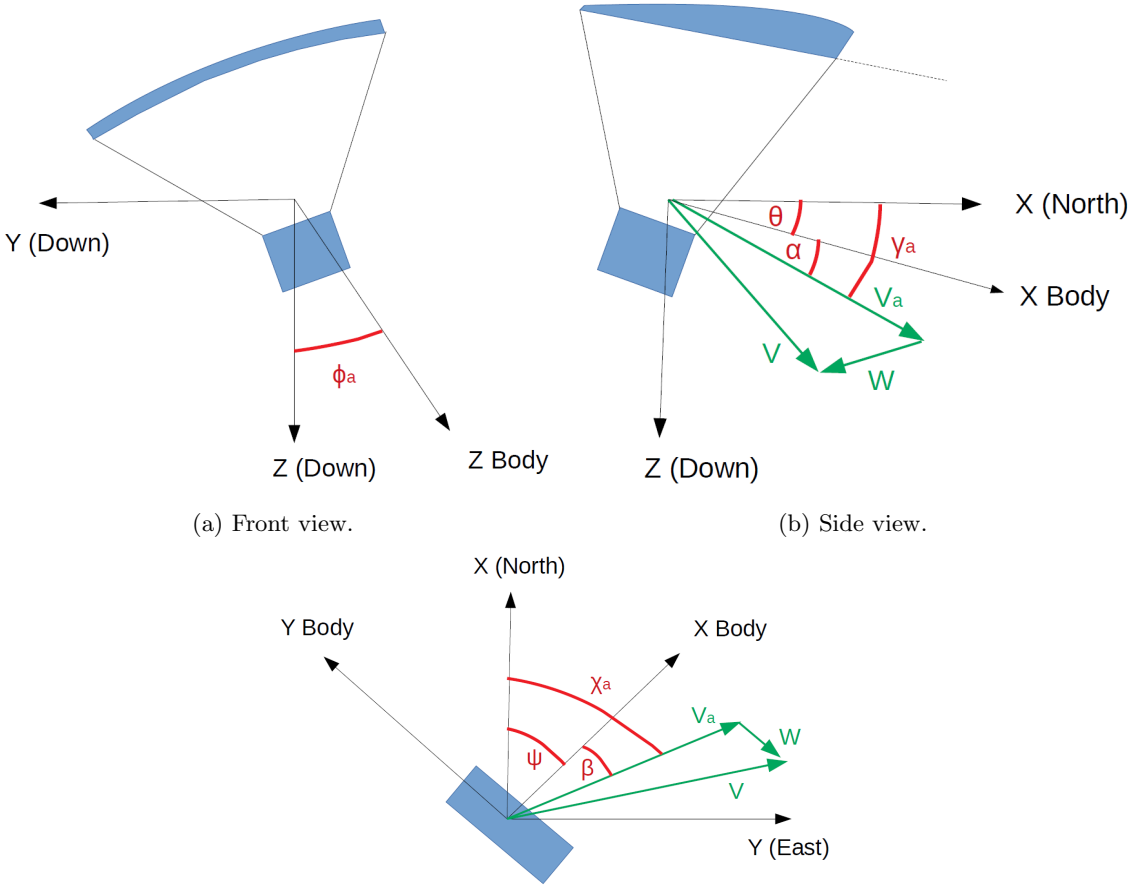
Wind frame $\{W\}$

- Origin: PADS center of gravity
- x_w axis: positive in the direction of the velocity vector relative to the air, airspeed vector $\boldsymbol{V}_a$
- z_w axis: perpendicular to the x_w axis, in the plane of symmetry of the PADS and positive pointing down
- y_w axis: perpendicular to the x_w-z_w plane, positive determined by the right-hand rule

A navigational frame $\{N\}$ parallel to the $\{NED\}$ but with origin at the PADS center of mass is often used when studying parafoils, but it won't be used in the present work since DSENDSEdu allows to avoid using it. Similarly, a geographical frame $\{G\}$ is sometimes used, but it won't be introduced since it has the same formal definition of the $\{NED\}$ frame, with the only difference being the origin set on the surface through the projection at the initial time.

2.3 Angles definitions

The above-mentioned angles are formally defined as follows [3]:



(a) Front view.

(b) Side view.

(c) View from above.

Figure 2.4: Angles definitions.

- Pitch angle θ: angle from a horizontal plane to the longitudinal axis of the body frame x_b
- Yaw angle ψ: angle from North to the longitudinal axis of the body frame x_b
- Roll angle ϕ: completes the transformation from $\{N\}$ to $\{B\}$ (the formal definition refers to a relative transformation expressed in an intermediate plane obtained after a θ and ψ rotation)
- Angle of attack α: angle between the projection of the airspeed vector $\boldsymbol{V}_a$ onto the x_b-z_b plane and the longitudinal axis of the body frame x_b
- Sideslip angle β: angle between the airspeed vector $\boldsymbol{V}_a$ and its projection onto the x_b-z_b plane

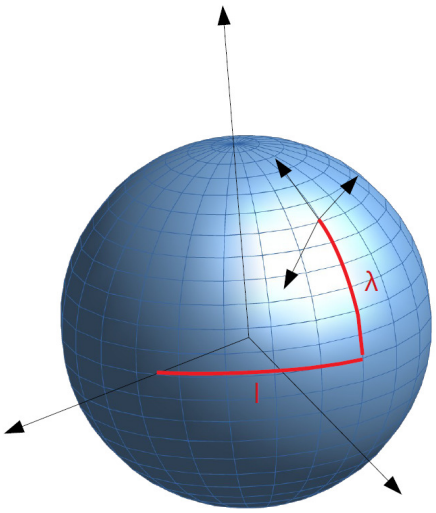


Figure 2.5: Transformation between planet-centered planet-fixed frame and geographical frame.

- Bank angle ϕ_a: rotation of the lift force around the airspeed vector $\boldsymbol{V}_a$ from the vertical plane
- Flight path angle γ_a: angle from a horizontal plane to the airspeed vector $\boldsymbol{V}_a$
- Heading angle χ_a: angle from North to the horizontal component of the airspeed vector $\boldsymbol{V}_a$

2.4 Relative transformations

Having defined the necessary angles, the relative transformations between the different frames are:

- $\{B\}$ to $\{NED\}$ (or $\{N\}$) $\rightarrow$ three Euler angles: roll ϕ, pitch θ, yaw ψ
- $\{B\}$ to $\{W\}$ $\rightarrow$ two Euler angles: angle of attack α, sideslip angle β
- $\{W\}$ to $\{NED\}$ (or $\{N\}$) $\rightarrow$ three Euler angles: bank angle ϕ_a, flight path angle γ_a, heading angle χ_a

Fig. 2.5 shows the transformation of coordinates needed to move from $\{I\}$ to $\{NED\}$ and Eq. 2.4 describes it.

$$\boldsymbol{R}_{GI} = \begin{bmatrix} -\sin(\lambda)\cos(l) & -\sin(\lambda)\cos(l) & \cos(\lambda) \\ -\sin(l) & \cos(l) & 0 \\ -\cos(\lambda)\cos(l) & -cos(\lambda)\sin(l) & -\sin(\lambda) \end{bmatrix} \tag{2.4}$$

where l represents the longitude and λ the latitude.

The transformation between the NED frame and body is described by Eq. 2.5.

$$\boldsymbol{R}_{B,NED} = \begin{bmatrix} c(\psi)c(\theta) & s(\psi)c(\theta) & -s(\theta) \\ c(\psi)s(\theta)s(\phi) - s(\psi)c(\phi) & s(\psi)s(\theta)s(\phi) + c(\psi)c(\phi) & c(\theta)s(\phi) \\ c(\psi)s(\theta)c(\phi) + s(\psi)s(\phi) & s(\psi)s(\theta)c(\phi) - c(\psi)s(\phi) & c(\theta)c(\phi) \end{bmatrix} \tag{2.5}$$

The rotation matrix between body-fixed and wind frame is:

$$\boldsymbol{R}_{WB} = \begin{bmatrix} \cos(\alpha)\cos(\beta) & -\sin(\beta) & \sin(\alpha)\cos(\beta) \\ \cos(\alpha)\sin(\beta) & \cos(\beta) & \sin(\alpha)\sin(\beta) \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{bmatrix} \tag{2.6}$$

where α is the angle of attack and β is the angle of sideslip.

In order to move directly from wind to NED frame, the matrix $\boldsymbol{R}_{WG}$ has to be defined as ([6]):

$$\boldsymbol{R}_{W,NED} = \begin{bmatrix} c(\chi_a)c(\gamma_a) & s(\chi_a)c(\gamma_a) & -s(\gamma_a) \\ c(\chi_a)s(\gamma_a)s(\phi_a) - s(\chi_a)c(\phi_a) & s(\chi_a)s(\gamma_a)s(\phi_a) + c(\chi_a)c(\phi_a) & c(\gamma_a)s(\phi_a) \\ c(\chi_a)s(\gamma_a)c(\phi_a) + s(\chi_a)s(\phi_a) & s(\chi_a)s(\gamma_a)c(\phi_a) - c(\chi_a)s(\phi_a) & c(\gamma_a)c(\phi_a) \end{bmatrix} \tag{2.7}$$

Chapter 3

Parafoil dynamics

The dynamics of a Precision Aerial Delivery System (PADS) is described by various dynamic models characterized by different levels of complexity, depending on the scope of the model. In most cases it is unnecessary to model every physical degree of freedom of the canopy and payload, unless the need to study some specific details of payload oscillations arises. Depending on the desired accuracy, there are models that range from three DOF in which the PADS is considered to be a point mass, all the way up to nine DOF in which canopy and payload are modeled as two separate bodies connected by an appropriate constraint (usually spherical). Simpler one-DOF models are also sometimes, though seldom, used when only the vertical descent dynamics needs to be analyzed.

Not only are simpler models adopted during the early stages of a project, but due to their reliability in describing the system accurately (provided reliable coefficients and parameters are available), they are often used in more advanced stages of the projects to guarantee faster convergence of the GNC algorithms (i.e. for applications such as real-time motion planning or trajectory optimization).

The present chapter adopts the typical aerospace convention of using a North-East-Down (NED) frame of reference when writing the equations of motion, even though the opposite convention (East-North-Up, ENU) will be used for DSENDSEdu simulations later on for convenience, mainly for being consistent with the software's default frames of reference.

Also, while Euler angles are used in the present chapter for a more clear physical description

of the problem, quaternions have been used for the actual software implementation due to the computational saving given by avoiding matrix operations.

3.1 Parafoil properties

Symbol	Property	Value	Units
AR	Aspect ratio	3.0	-
S	Canopy area	3.14	m^2
t	Canopy thickness	0.075	m
R	Lines length	1.84	m
a	Canopy height	0.164	m
x_c	Center of mass to aerodynamic center distance	0.26	m
σ	Final sigma (contractor's value)	0.45	kg/m^2
z_{rp}	Payload distance	1.5	m
l_{diam}	Lines diameter	0.005	m
l_{ha}	Harness length	-	m
b	Canopy span	-	m
c	Canopy chord	-	m
m_p	Canopy mass	0.002	kg
S_p	Drag surface	-	m^2

Table 3.1: Canopy properties.

Symbol	Property	Value	Units
a	Depth	0.5	m
b	Width	0.5	m
c	Height	0.5	m
m_l	Mass	200	kg
S_l	Drag surface	0.25	m^2

Table 3.2: Payload properties.

3.2 Wind effect

Calling $\boldsymbol{W}$ the wind vector expressed in the geodesic frame

$$\boldsymbol{W} = [w_x, w_y, w_z]^T \tag{3.1}$$

and $\boldsymbol{V}$ the ground speed vector

$$\boldsymbol{V} = [V_x, V_y, V_z]^T \tag{3.2}$$

the velocity relative to the air (airspeed vector) $\boldsymbol{V}_a$ is defined as follows

$$\boldsymbol{V}_a = \boldsymbol{V} - \boldsymbol{W}. \tag{3.3}$$

$$\boldsymbol{V}_a = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix} - \boldsymbol{R}_{BN}\boldsymbol{W} \tag{3.4}$$

The actual planet-referenced flight path angle γ can be defined as

$$\gamma = -\arcsin\left(\frac{V_d}{V}\right) \tag{3.5}$$

with the real descent rate of the parafoil is

$$V_d = V_z = V_v + w_z \tag{3.6}$$

The flight path angle γ_a is usually computed through the vertical component of the airspeed vector V_v (neglecting w_x) so that

$$\gamma_a = \arcsin\left(\frac{V_v}{V_a}\right) \qquad V_a = |\boldsymbol{V}_a| \tag{3.7}$$

It will then be assumed that $\gamma \approx \gamma_a$ holds.

3.3 Reduced three-DOF model

[3] and [7] describe a three-DOF dynamic model written in wind reference frame. The state vector $\boldsymbol{x} = [V_a, \gamma_a, \chi_a]^T$ contains respectively the velocity vector, the flight path angle and the heading angle. Its dynamics is expressed by Eq. 3.8.

$$\begin{cases} \dot{V_a} = -\dfrac{1}{m}\left(D + mg\sin\left(\gamma_a\right)\right) \\ \dot{\gamma_a} = \dfrac{1}{mV_a}\left(L\cos\left(\phi_a\right) - mg\cos\left(\gamma_a\right)\right) \\ \dot{\chi_a} = \dfrac{L\sin\left(\phi_a\right)}{mV_a\cos\left(\gamma_a\right)} \end{cases} \tag{3.8}$$

where $L = 0.5\rho V_a^2 S C_L$ and $D = 0.5\rho V_a^2 S C_D$ indicate lift and drag, m is the total mass of the PADS, g is the gravitational acceleration and ϕ_a indicates the banking angle, which in this case coincides with the control variable.

Given the rotation matrix needed to go from the wind to the inertial frame of reference $\boldsymbol{R_{WN}}$ and the geodetical wind components, kinematic equation 3.9 allows the extract the inertial coordinates x, y and z.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \boldsymbol{R}_{WN}^T \begin{bmatrix} V_a \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \\ w_z \end{bmatrix} \tag{3.9}$$

The above equations assume the planet to be flat and are used in situations where the parafoil altitude is negligible with respect to the planet radius, typically for terminal descent. They also rely on the strong hypothesis that the angle of attack does not influence lift and drag coefficients, a condition that can be compensated by adequately varying C_L and C_D.

3.4 Reduced three-DOF model for a spherical planet

While the flat-planet three-DOF model is reliable whenever path planning is performed at relatively low altitudes (a few kilometers), a somewhat more accurate dynamics must be considered whenever the initial altitude is high when compared to a planet's radius. An improvement over the previous model is presented in both [7] and [8]. Although the PADS is still modeled as a point mass, the sphericity of the planet is taken into account and the state vector is expanded by adding latitude, longitude and radius. The buoyancy effect is also considered.

The full system of equations (comprising the kinematic) is presented in Eq. 3.10

$$\begin{cases} \dot{V_a} = \left(\frac{B}{m} - g\right)\sin(\gamma_a) - \frac{D}{m} \\ \dot{\gamma_a} = \frac{V_a}{R_P + h}\cos(\gamma_a) + \left(\frac{B}{m} - g\right)\frac{\cos(\gamma_a)}{V_a} + \frac{L\cos(\phi_a) + Y\sin(\phi_a)}{mV_a} \\ \dot{\chi_a} = -\frac{V_a}{R_P + h}\cos(\gamma_a)\cos(\chi_a)\tan(\lambda) + \frac{L\sin(\phi_a) - Y\sin(\phi_a)}{mV_a\cos(\gamma_a)} \\ \\ \dot{r} = \dot{h} = V_a\sin(\gamma_a) \\ \dot{\Lambda} = \frac{V_a\cos(\gamma_a)\cos(\chi_a)}{r\cos(\lambda)} \\ \dot{\lambda} = \frac{V_a\cos(\gamma_a)\sin(\chi_a)}{r} \\ \dot{x} = V_a\sin(\gamma_a) + w_x \\ \dot{y} = V_a\cos(\gamma_a)\cos(\chi_a) + w_y \\ \dot{z} = V_a cos(\gamma_a)\sin(\chi_a) + w_z \end{cases} \tag{3.10}$$

where h is the altitude above the surface, Λ indicates the longitude, λ indicates the latitude and B is the buoyancy force. $r = R_t + h$ is the sum of Titan's radius and current altitude, and coincides with the norm of $[x, y, z]$.

3.5 Six-DOF model

[3] presents a full six-DOF model in which the PADS attitude is considered as well. This is the model used for the simulations throughout this work, mainly because of the accuracy of the results. The parameters of the physical canopy, in fact, are only available for a 6 DOF model at the time of writing. The whole system is considered as a rigid body in which canopy and payload rotate/translate together and, unless someone is specifically interested in investigating the payload oscillation (e.g. to estimate camera movements and/or to verify maximum payload acceleration), this is the main model presented in literature (see, for example, [3] and [9]). The state vector includes the three linear velocity components u, v, w and the three angular rates p, q, r, expressed in body frame. The three spatial coordinates x, y, z as well as the three attitude components ϕ, θ, ψ are obtained through the appropriate kinematic relationships.

3.5.1 Inertias

As a first step, the inertia matrices of both the parafoil canopy and payload must be obtained. These will lay the foundation to compute the full inertia matrices by adding the apparent terms, which will be done in the next section.

The moments of inertia of the load referring to the body axes are:

$$\begin{cases} I_{x,l} = \frac{m_l}{12}\left(b_{load}^2 + c_{load}^2\right) \\ I_{y,l} = \frac{m_l}{12}\left(a_{load}^2 + c_{load}^2\right) \\ I_{z,l} = \frac{m_l}{12}\left(a_{load}^2 + b_{load}^2\right) \end{cases} \tag{3.11}$$

where a_{load}, b_{load} and c_{load} represent length, width and height of the payload, respectively.

The parafoil center of gravity is located in the longitudinal plane of symmetry at $c/4$ and at a distance l_{cgp} from the point where the suspension lines confluence.

$$l_{cgp} = \frac{2}{3}\frac{(l_{lines}+t)^3 - l_{lines}^3}{(l_{lines}+t)^2 - l_{lines}^2}\frac{\sin\epsilon}{\epsilon} \qquad t = \frac{v_{vol}}{S_p} \tag{3.12}$$

with t being the average canopy thickness, S_p being the reference area defined as $S_p = cb$ and v_{vol} approximated as $0.09c^2b$.

Defining h_{mean} as

$$h_{mean} = \frac{v_{vol}}{cb} \tag{3.13}$$

the moments of inertia of the parafoil can be expressed as

$$\begin{cases} I_{x,p} = \frac{v_{vol}\rho + m_p}{12}\left(b^2 + h_{mean}^2\right) \\ I_{y,p} = \frac{v_{vol}\rho + m_p}{12}\left(c^2 + h_{mean}^2\right) \\ I_{z,p} = \frac{v_{vol}\rho + m_p}{12}\left(c^2 + b^2\right) \end{cases} \tag{3.14}$$

The above moments of inertia can be projected onto the body frame as follows:

$$\boldsymbol{I} = \boldsymbol{R}_y \boldsymbol{I}' \boldsymbol{R}_y^T \qquad \text{where} \quad \boldsymbol{R}_y = \begin{bmatrix} \cos(\mu) & 0 & \sin(\mu) \\ 0 & 1 & 0 \\ -\sin(\mu) & 0 & \cos(\mu) \end{bmatrix} \tag{3.15}$$

$$\begin{aligned}\boldsymbol{I} &= \begin{bmatrix} I_{xx} & 0 & I_{xz} \\ 0 & I_{yy} & 0 \\ I_{zx} & 0 & I_{zz} \end{bmatrix} = \\ &= \begin{bmatrix} I_{x,p}\cos(\mu)^2 + I_{z,p}\sin(\mu)^2 & 0 & \frac{1}{2}(I_{z,p} - I_{x,p})\sin(2\mu) \\ 0 & I_{y,p} & 0 \\ \frac{1}{2}(I_{z,p} - I_{x,p})\sin(2\mu) & 0 & I_{x,p}\sin(\mu)^2 + I_{z,p}\cos(\mu)^2 \end{bmatrix}\end{aligned} \tag{3.16}$$

This rotation is only to be performed for the canopy, since it is rotated of the rigging angle μ.

3.5.2 Apparent mass and inertia

When a body moves in air, the air around it is set to move as well. Apparent mass effects depend not only on the acceleration of the center of mass, but also on the local acceleration at a significant distance from the body center of mass. The real effect of the local acceleration can be described by detailed analytical formulations out of the scope of this work. Iterative procedures can also be adopted as an alternative. This motion introduces pressure forces on the body itself, the *apparent mass pressures*. These effects are far from negligible and their magnitude follows an inverse relationship with the mass ratio (i.e. the ratio between the vehicle mass m and the displaced air mass). While for conventional vehicles such as airplanes this ratio is extremely high, it is much smaller for lighter-than-air vehicles.

Apparent mass and inertia contributions are given ([3]) by Eq. 3.17 and Eq. 3.18, both written in the rotating parafoil coordinate frame.

$$\tilde{\boldsymbol{F}}_{a.m.} = -\left(\boldsymbol{I}_{a.m.} \begin{bmatrix} \dot{\tilde{v}}_x \\ \dot{\tilde{v}}_y \\ \dot{\tilde{v}}_z \end{bmatrix} + \boldsymbol{S}(\tilde{\boldsymbol{\omega}}) \boldsymbol{I}_{a.m.} \begin{bmatrix} \tilde{v}_x \\ \tilde{v}_y \\ \tilde{v}_z \end{bmatrix} \right) \tag{3.17}$$

$$\tilde{\boldsymbol{M}}_{a.i.} = -\left(\boldsymbol{I}_{a.i.} \begin{bmatrix} \dot{\tilde{p}} \\ \dot{\tilde{q}} \\ \dot{\tilde{r}} \end{bmatrix} + \boldsymbol{S}(\tilde{\boldsymbol{\omega}}) \boldsymbol{I}_{a.i.} \begin{bmatrix} \tilde{p} \\ \tilde{q} \\ \tilde{r} \end{bmatrix} + \boldsymbol{S}(\tilde{\boldsymbol{V}}_a) \boldsymbol{I}_{a.m.} \begin{bmatrix} \tilde{v}_x \\ \tilde{v}_y \\ \tilde{v}_z \end{bmatrix} \right) \tag{3.18}$$

The $\sim$ symbol is used to denote the components of the canopy airspeed vector $\tilde{\boldsymbol{V}}_a$ and of the canopy

angular velocity vector $\tilde{\boldsymbol{\omega}}$, expressed in the parafoil canopy frame, which is different from the body frame because of the nonzero rigging angle.

The apparent mass force and apparent moment of inertia act at the centroid of the apparent mass of the body, which depends on the body's geometry and for ellipsoidal canopy shapes resides at approximately volumetric canopy centroid [3]. $\boldsymbol{I}_{a.m.}$ and $\boldsymbol{I}_{a.i.}$ have the diagonal form:

$$\boldsymbol{I}_{a.m.} = \begin{bmatrix} A & 0 & 0 \\ 0 & B & 0 \\ 0 & 0 & C \end{bmatrix} \qquad \boldsymbol{I}_{a.i.} = \begin{bmatrix} I_A & 0 & 0 \\ 0 & I_B & 0 \\ 0 & 0 & I_C \end{bmatrix} \tag{3.19}$$

Based on [10], [11] derived more accurate expressions for the apparent masses A, B and C and for the apparent inertias I_A, I_B and I_C, as shown in Eqs. 3.20 and 3.21 (the approach is the same presented by [3] and similar to that of [9]). These expressions assume that the parafoil axes system coincides with the principal axes of the canopy (which is typically the case).

$$\begin{aligned} A &= 0.666\rho \left(1 + \frac{8}{3}a^{*2}\right) t^2 b \\ B &= 0.267\rho \left[1 + 2\frac{a^{*2}}{t^{*2}} AR^2 \left(1 - t^{*2}\right)\right] t^2 c \\ C &= 0.785\rho \sqrt{1 + 2a^{*2}\left(1 - t^{*2}\right)} \frac{AR}{1 + AR} c^2 b \end{aligned} \tag{3.20}$$

$$\begin{aligned} I_A &= 0.055\rho \frac{AR}{1 + AR} c^2 b^3 \\ I_B &= 0.0308\rho \frac{AR}{1 + AR} \left[1 + \frac{\pi}{6}\left(1 + AR\right) AR a^{*2} t^{*2}\right] c^4 b \\ I_C &= 0.0555\rho \left(1 + 8a^{*2}\right) t^2 b^3 \end{aligned} \tag{3.21}$$

where the following quantities have been used:

- aspect ratio

$$AR = \frac{b}{c} \tag{3.22}$$

- arc-to-span-ratio

$$a^* = \frac{a}{b} \tag{3.23}$$

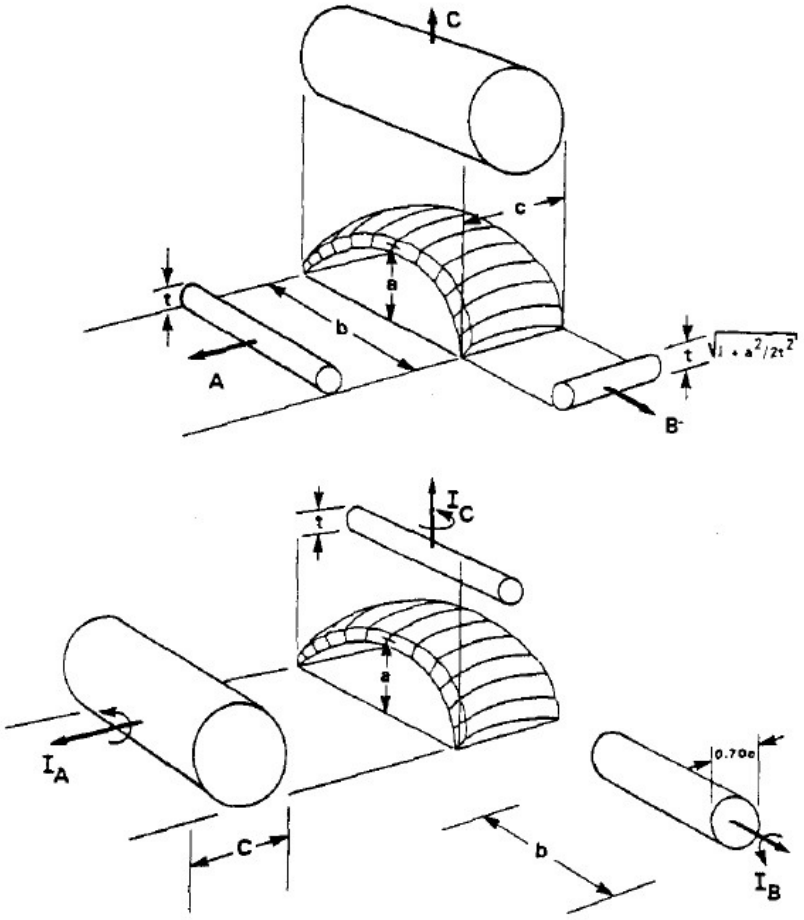


Figure 3.1: Visualization of apparent mass and inertia.

- relative thickness

$$t^* = \frac{t}{c} \tag{3.24}$$

$\boldsymbol{I}_{a.f.}$ and $\boldsymbol{I}_{a.i.}$ are then rotated in body frame through the following equations:

$$\boldsymbol{I}^*_{a.m.} = \boldsymbol{R}^T_{PB} \boldsymbol{I}_{a.m.} \boldsymbol{R}_{PB} \tag{3.25}$$

Under steady flight conditions (which include steady turning maneuvers), this model represents

accurately the apparent mass effects. Consequently, as long as the system is not subject to intense dynamic flight conditions, this model is a good approximation.

Fig. 3.1 offers a visual representation of the contributions presented above.

3.5.3 Parafoil canopy aerodynamics

The aerodynamic forces refer to the center of gravity of the parafoil. They are defined in the axes system determined by the local air velocity $\boldsymbol{V_p}$. The local attack and sideslip angles, respectively α and β, are defined as:

$$\alpha = \arctan\left(\frac{w_p}{u_p}\right) \tag{3.26}$$

$$\beta = \arctan\left(\frac{v_p}{\sqrt{u_p^2 + w_p^2}}\right) \tag{3.27}$$

Once they are computed in the intermediate coordinate systems, the forces and moments can be expressed in the body frame making use of the transformation matrix M_{ba}:

$$\boldsymbol{M}_{BA} = \begin{pmatrix} \cos(\mu+\alpha) & 0 & -\sin(\mu+\alpha) \\ 0 & 1 & 0 \\ \sin(\mu+\alpha) & 0 & \cos(\mu+\alpha) \end{pmatrix} \tag{3.28}$$

3.2 shows the CLARK-Y aerodynamic profile, adopted as a reference for parafoil canopies (the rotation corresponds to the rigging angle μ).

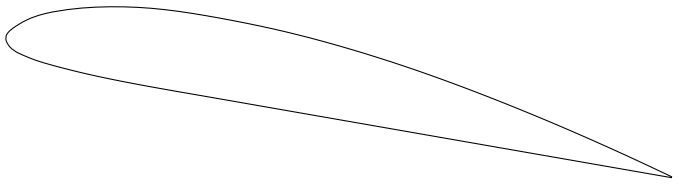


Figure 3.2: CLARK-Y aerodynamic profile.

A downward deflection of the canopy trailing edges is considered in terms of a dimensionless ratio $\delta_{curr}/\delta_{max}$ where δ_{curr} represents the actual deflection and δ_{max} the maximum available deflection. The left and right deflection are referred to as δ_l and δ_r (right and left are defined when looking towards the direction the parafoil is moving in). δ_l and δ_r, being dimensionless, vary between 0 and 1.

In the aerodynamic model used in this work, the symmetric and asymmetric deflections δ_s and δ_a

are used instead, since they have the clear physical meaning of influencing the longitudinal and lateral dynamics, respectively. These are:

$$\delta_s = \frac{1}{2}(\delta_r + \delta_l) \tag{3.29}$$

$$\delta_a = \delta_r - \delta_l \tag{3.30}$$

All the following coefficients refer to a local kinetic pressure $\frac{1}{2}\rho V_p^2$, a reference area S_p and a chord c. The following forces and coefficients all refer to the canopy, even though the subscripts have been omitted to simplify the notation.

Lift force

$$C_L = C_{L,0} + C_{L,\alpha}\alpha + C_{L,\delta_s}\delta_s \tag{3.31}$$

$$F_Z = \frac{1}{2}\rho V_p^2 S_p(-C_L) \tag{3.32}$$

Coefficient	Value	Units
$C_{L,0}$	0.091	-
$C_{L,\alpha}$	0.90	rad^{-1}
C_{L,δ_s}	0.21	-

Table 3.3: Canopy lift coefficients.

Drag force

$$C_D = C_{D,0} + C_{D,\alpha^2}\alpha^2 + C_{D,\delta_s}\delta_s \tag{3.33}$$

$$F_X = \frac{1}{2}\rho V_p^2 S_p(-C_D) \tag{3.34}$$

Side force

$$C_Y = C_{Y,\beta}\beta \tag{3.35}$$

Coefficient	Value	Units
$C_{D,0}$	0.25	-
C_{D,α^2}	0.12	rad^{-2}
C_{D,δ_s}	0.30	-

Table 3.4: Canopy drag coefficients.

$$F_Y = \frac{1}{2}\rho V_p^2 S_p C_Y \tag{3.36}$$

with $C_{Y,\beta} = -0.23\ \text{rad}^{-1}$.

Rolling moment

$$C_l = C_{l,\beta}\beta + C_{l,\delta_a}\delta_a + C_{l,p}\frac{b}{2V_{a,p}}p + C_{l,r}\frac{b}{2V_{a,p}}r \tag{3.37}$$

$$L = \frac{1}{2}\rho V_p^2 S_p c C_l \tag{3.38}$$

Coefficient	Value	Units
$C_{l,\beta}$	-0.036	rad^{-1}
C_{l,δ_a}	-0.0035	-
$C_{l,p}$	-0.84	rad^{-1}
$C_{l,r}$	-0.082	rad^{-1}

Table 3.5: Canopy rolling moment coefficients.

Pitching moment

$$C_m = C_{m,0} + C_{m,\alpha}\alpha + C_{m,q}\frac{c}{V_{a,p}}q \tag{3.39}$$

$$M = \frac{1}{2}\rho V_p^2 S_p c C_m \tag{3.40}$$

Yawing moment

$$C_n = C_{n,\beta}\beta + C_{n,\delta_a}\delta_a + C_{n,p}\frac{b}{2V_{a,p}}p + C_{n,r}\frac{b}{2V_{a,p}}r \tag{3.41}$$

Coefficient	Value	Units
$C_{m,0}$	0.35	-
$C_{m,\alpha}$	-0.72	rad^{-1}
$C_{m,q}$	-1.49	rad^{-1}

Table 3.6: Canopy pitching moment coefficients.

$$N = \frac{1}{2}\rho V_p^2 S_p c C_n \tag{3.42}$$

Coefficient	Value	Units
$C_{n,\beta}$	-0.0015	rad^{-1}
C_{n,δ_a}	0.0155	-
$C_{n,p}$	-0.082	rad^{-1}
$C_{n,r}$	-0.27	rad^{-1}

Table 3.7: Canopy yawing moment coefficients.

3.5.4 Suspension lines aerodynamics

It is assumed that only one force (drag) acts on the suspension lines, applied at half distance between the center of mass of the parafoil and the center of gravity of the parafoil, with a direction parallel to that of the local air velocity. The reference area S_{line} of the suspension lines is obtained starting from the area S_{ref} of a single line, which is multiplied by the number of lines N and scaled to keep into account the cascading part of the lines.

The aerodynamic force is then computed as

$$\frac{1}{2}\rho V_s^2 S_{line}(-C_{line}) \qquad \text{with } C_{line} = 1.0 \tag{3.43}$$

3.5.5 Payload aerodynamics

The aerodynamic model employed for the payload is similar to that of the suspension lines and only accounts for the drag:

$$\frac{1}{2}\rho V_l^2 S_l(-C_{load}) \qquad \text{with } C_{load} = 1.0 \tag{3.44}$$

3.5.6 Equations of motion

Given the terms described in the previous sections, the motion is governed by the following equation:

$$\boldsymbol{A}\dot{\boldsymbol{x}} = \boldsymbol{b} \tag{3.45}$$

with

$$\dot{\boldsymbol{x}} = \begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} \tag{3.46}$$

Before expanding the terms above, a few vectors and rotation matrices must be defined.

Weight force $\boldsymbol{F_g}$ represents the weight force expressed in body-fixed frame and is defined as follows:

$$\boldsymbol{F}_g = mg \begin{bmatrix} -\sin(\theta) \\ \cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \end{bmatrix} \tag{3.47}$$

where m is the mass and g the gravity acceleration.

Aerodynamic forces $\boldsymbol{F}_a$ is the vector containing the aerodynamic forces acting along x, y and z components of the body frame.

$$\boldsymbol{F}_a = -QS\boldsymbol{R_{BW}} \begin{bmatrix} C_D \\ C_Y \\ C_L \end{bmatrix} \tag{3.48}$$

where $Q = 0.5\rho V_a^2$ represents the dynamic pressure.

Aerodynamic moments $\boldsymbol{M}_a$ contains the aerodynamic moments expressed in body-fixed frame:

$$\boldsymbol{M}_a = QS \begin{bmatrix} b\,C_l \\ c\,C_m \\ b\,C_n \end{bmatrix} \tag{3.49}$$

where b and c are the span and chord of the parafoil.

$\boldsymbol{S}_\omega$ and $\boldsymbol{S}_{r_{BM}}$ are skew symmetric matrices built from $\boldsymbol{\omega} = [p, q, r]^T$ and $\boldsymbol{r}_{BM} = [x_{BM}, y_{BM}, z_{BM}]$. $\boldsymbol{r}_{BM}$ is the vector that goes from the system center of mass to the apparent center of mass, where apparent forces of the parafoil are applied: here it's modeled as $[0, 0, -R\cos(\epsilon)]$, where R is the line length and ϵ is the angle between line length and vertical direction. They are defined as:

$$\boldsymbol{S_\omega} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix} \tag{3.50}$$

$$\boldsymbol{S_{r_{BM}}} = \begin{bmatrix} 0 & -z_{BM} & y_{BM} \\ z_{BM} & 0 & -x_{BM} \\ -y_{BM} & x_{BM} & 0 \end{bmatrix} \tag{3.51}$$

$$\boldsymbol{I_{a.i.}}^* = \boldsymbol{R_{PB}}^T \boldsymbol{I_{a.I.}} \boldsymbol{R_{PB}} \tag{3.52}$$

$\boldsymbol{R}_{PB}$ is defined as:

$$\boldsymbol{R_{PB}} = \begin{bmatrix} \cos(\mu) & 0 & -\sin(\mu) \\ 0 & 1 & 0 \\ \sin(\mu) & 0 & \cos(\mu) \end{bmatrix} \tag{3.53}$$

where μ is the rigging angle (defined as unique property of the parafoil geometry). Finally some terms which have not been defined yet have to be introduced: $m_e = 0.09\rho bc^2$ is the added mass (mass of air entrapped in the ram air canopy), $\boldsymbol{I_{3x3}}$ is the identity 3x3 matrix. A final point has to be underlined: knowing (u, v, w, ϕ, θ, ψ), bank angle, flight path angle, heading angle and

airspeed velocity can be retrieved in this way:

$$\boldsymbol{R}_{BN}{}^{T}\left(\begin{bmatrix}u\\v\\w\end{bmatrix}-\boldsymbol{R}_{BN}\begin{bmatrix}w_x\\w_y\\w_z\end{bmatrix}\right)=\boldsymbol{R}_{WN}{}^{T}\begin{bmatrix}V_a\\0\\0\end{bmatrix} \tag{3.54}$$

knowing that $\boldsymbol{V}_a$ is calculated as the modulus of:

$$\boldsymbol{V}_a=\begin{bmatrix}u\\v\\w\end{bmatrix}-\boldsymbol{R}_{BN}\begin{bmatrix}w_x\\w_y\\w_z\end{bmatrix} \tag{3.55}$$

Assumptions which have been done to obtain this model are:

- The parafoil is treated as a 3D rigid body: there is not relative motion between the canopy and and the payload.

- The parafoil canopy is considered to be a fixed shape once it has completely inflated.

- Apparent mass and inertia are applied to canopy center of mass (they are rotated with respect to body frame of the angle μ)

Coming back to Eq. 3.45, all the elements to define $\boldsymbol{A}$ and $\boldsymbol{b}$ are now available.

$$\boldsymbol{A}=\begin{bmatrix}(m+m_e)\boldsymbol{I}+\boldsymbol{I_{a.m.}}^{*} & -\boldsymbol{I_{a.m.}}^{*}\boldsymbol{S_{r_{BM}}}\\ \boldsymbol{S_{r_{BM}}}\boldsymbol{I_{a.m.}}^{*} & \boldsymbol{I}+\boldsymbol{I_{a.i.}}^{*}-\boldsymbol{S_{r_{BM}}}\boldsymbol{I_{a.m.}}^{*}\boldsymbol{S_{r_{BM}}}\end{bmatrix} \tag{3.56}$$

$$\boldsymbol{b}=\begin{bmatrix}\boldsymbol{B_1}\\\boldsymbol{B_2}\end{bmatrix} \tag{3.57}$$

where:

$$\boldsymbol{B_1} = \boldsymbol{F_a} + \boldsymbol{F_g} - \boldsymbol{S_\omega}((m+m_e)\boldsymbol{I_{3x3}} + \boldsymbol{I_{a.m.}}^*) \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \\ + \boldsymbol{S_\omega}\boldsymbol{I_{a.m.}}^*\boldsymbol{S_{r_{BM}}} \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \boldsymbol{S_\omega}\boldsymbol{I_{a.m.}}^*\boldsymbol{R_{BN}}\boldsymbol{W} \quad (3.58)$$

$$\boldsymbol{B_2} = \boldsymbol{M_a} - (\boldsymbol{S_\omega}(\boldsymbol{I} + \boldsymbol{I_{a.i.}}^*) - \boldsymbol{S_{r_{BM}}}\boldsymbol{S_\omega}\boldsymbol{I_{a.m.}}^*\boldsymbol{S_{r_{BM}}}) \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \\ - \boldsymbol{S_{r_{BM}}}\boldsymbol{S_\omega}\boldsymbol{I_{a.m.}}^* \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \boldsymbol{S_{r_{BM}}}\boldsymbol{S_\omega}\boldsymbol{I_{a.m.}}^*\boldsymbol{R_{BN}}\boldsymbol{W} \quad (3.59)$$

Translational and rotational kinematics relationships are used to retrieve linear and angular positions. These kinematic equations are as follows:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \boldsymbol{R_{BN}}^T \begin{bmatrix} u \\ v \\ w \end{bmatrix} \quad (3.60)$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\dfrac{\sin(\theta)}{\cos(\theta)} & \cos(\phi)\dfrac{\sin(\theta)}{\cos(\theta)} \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\dfrac{1}{\cos(\theta)} & \cos(\phi)\dfrac{1}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \quad (3.61)$$

3.5.7 Implementation details

While the previous section adopted a North-East Down (NED) convention, DSENDSEdu implementation uses a East-North-Up (ENU) frame of reference. This is mainly due to a more straightforward implementation it and the need of a more flexible framework which might be used in the

future. The following list highlights the most important differences:

- The sign of the rigging angle μ, sideslip angle β and most of the angular velocities will be opposite. Rotation matrices will differ as well.
- In the software implementation, projections are handled through quaternions, which are far more efficient than matrices when it comes to computational efficiency
- [9] aims at obtaining a model for which all the forces are applied to the center of mass of the entire system, mainly for a easier representation and a more convenient numerical approach. For a more convenient implementation in DSENDSEdu, the present work distinguishes between the following groups of forces, applied independently from each other:
  - acting on the center of pressure of the canopy (aerodynamic forces)
  - acting on the body center (apparent masses, buoyancy)
  - acting on the lines of the parafoil (drag)
  - acting on the payload to be deployed (drag and lift)
  - gravity acting on each body separately
- Linear/angular velocities and accelerations are probed directly from the local body when needed, so that no Rivals transportation theorem is not to be implemented directly. This allows to exploit the Spatial Operator Algebra (SOA) properties which will be introduced in the next chapter.
- The model analyzed in [9] brings to equations that need to be integrated numerically, for this reason some terms (usually the ones depending on acceleration) are neglected. In this work no terms will be neglected since DSENDSEdu allows to handle them with relative simplicity.

3.6 Six-DOF model linearization

3.6.1 Linearization

The need for linearized models arises for both stability analysis and controllers necessity. Implementing the linear dynamics results in much faster computation times, where suitable. The 6-DOF

model presented in the previous section can be written as:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} \tag{3.62}$$

For an equilibrium position we have that

$$\boldsymbol{0} = \boldsymbol{f}(\boldsymbol{x}_0) + \boldsymbol{g}(\boldsymbol{x}_0)\boldsymbol{u}_0 \tag{3.63}$$

and the linearization of 3.62 around 3.63 leads to [3]

$$\Delta\dot{\boldsymbol{x}} = \left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} - \boldsymbol{f}\boldsymbol{g}^{-1}\frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\right)\Bigg|_{\boldsymbol{x}=\boldsymbol{x}_0} \Delta\boldsymbol{x} + \boldsymbol{g}_0\Delta\boldsymbol{u} = \boldsymbol{A}\Delta\boldsymbol{x} + \boldsymbol{B}\Delta\boldsymbol{u} \tag{3.64}$$

with $\boldsymbol{A}$ and $\boldsymbol{B}$ being the usual state and input matrices, respectively. If needed, the output is also linearized as:

$$\Delta\boldsymbol{y} = \boldsymbol{C}\Delta\boldsymbol{x} + \boldsymbol{D}\Delta\boldsymbol{u} \tag{3.65}$$

where $\boldsymbol{C}$ and $\boldsymbol{D}$ output and feedthrough matrices. Considering Eq. 3.46, the state vector would be $[\Delta u, \Delta v, \Delta w, \Delta p, \Delta q, \Delta r]^T$ while the control vector is $[\Delta\delta_a, \Delta\delta_s]^T$. In practice however, rotational kinematics is usually included as well, as suggested by [3], which leads to

$$\boldsymbol{x} = [\Delta u, \Delta v, \Delta w, \Delta p, \Delta q, \Delta r, \Delta\phi, \Delta\theta, \Delta\psi]^T \tag{3.66}$$

The longitudinal and lateral-directional motions are usually studied independently, so that the former motion is described by

$$\boldsymbol{x}_{lon} = [u, w, q, \theta]^T \qquad \text{with} \quad u = \delta_s \tag{3.67}$$

and the latter by

$$\boldsymbol{x}_{lat} = [v, p, r, \phi, \psi]^T \qquad \text{with} \quad u = \delta_a \tag{3.68}$$

.

This is possible since δ_s and δ_a are responsible for controlling the longitudinal and lateral dynamics, respectively.

Linearizing the model around its steady-state condition (free gliding) at an altitude of 20 km leads

to Eqs. 3.69 and 3.70.

$$\begin{bmatrix} \dot{u} \\ \dot{w} \\ \dot{q} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} -0.0504 & -0.00195 & -9 & -1.34 \\ -0.0409 & -0.0374 & 16.9 & 0.0519 \\ 1.7 & -0.861 & -3.99 & -0.101 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \\ q \\ \theta \end{bmatrix} + \begin{bmatrix} -0.364 \\ -0.734 \\ 5.06 \\ 0 \end{bmatrix} \delta_s \tag{3.69}$$

$$\begin{bmatrix} \dot{v} \\ \dot{p} \\ \dot{r} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 0.00588 & 9.16 & -17.2 & 1.34 & 0 \\ -0.0722 & -2.86 & 0.326 & -0.0709 & 0 \\ -0.076 & -3.06 & -3.63 & 0.00458 & 0 \\ 0 & 1 & -0.0427 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} v \\ p \\ r \\ \phi \\ \psi \end{bmatrix} + \begin{bmatrix} 8.35 \cdot 10^{-5} \\ -0.475 \\ 2.14 \\ 0 \\ 0 \end{bmatrix} \delta_a \tag{3.70}$$

3.6.2 Stability analysis

Once the system is reduced to the state-space model, its stability is assessed by analyzing the eigenvalues of matrix A which are all stable. As mentioned above, this analysis is conducted by linearizing around a trimming point at 30 km from the ground during a free gliding condition. A steady-state flight condition is not the same at every altitude due to the change in air density, and poles change slightly during the descent.

The following modes typical of a parafoil can be identified around the trim condition, and they are consistent with the results presented in [12] and [3]:

Mode	**Eigenvalues**	**Period [s]**	**Damping ratio**
Roll subsidence	-4.7	0.3	-
Spiral mode	-3.8	0.2	-
Short period oscillations	$-2.6 \pm 4.7\,i$	1.3	0.49
Phugoid mode	$-0.086 \pm 0.0041\,i$	11	≈ 1
Dutch roll	$-0.072 \pm 0.15\,i$	14	0.44

Table 3.8: PADS modes.

Lowering the altitude, the phugoid mode tends to disappear: the two complex conjugate poles become real.

3.6.3 Controllability

Starting from Eqs. 3.69 and 3.70, it is possible to compute the longitudinal and lateral controllability matrices (which hold around the linearization condition) as:

$$\boldsymbol{K_{C,lon}} = \left[\boldsymbol{B}_{lon}, \boldsymbol{A}_{lon}^2\boldsymbol{B}_{lon}, \boldsymbol{A}_{lon}^3\boldsymbol{B}_{lon}\right] \tag{3.71}$$

$$\boldsymbol{K_{C,lat}} = \left[\boldsymbol{B}_{lat}, \boldsymbol{A}_{lat}^2\boldsymbol{B}_{lat}, \boldsymbol{A}_{lat}^3\boldsymbol{B}_{lat}, \boldsymbol{A}_{lat}^4\boldsymbol{B}_{lat}\right] \tag{3.72}$$

whose determinants are $\det\left(\boldsymbol{K_{C,lon}}\right) \approx -105.6 \neq 0$ and $\det\left(\boldsymbol{K_{C,lat}}\right) \approx -5.976 \neq 0$, assuring controllability around the steady-state gliding condition.

Chapter 4

DSENDSEdu implementation and preliminary results

The Mobility and Robotic Systems Section at Jet Propulsion Laboratory developed the Dynamics And Real-Time Simulation (DARTS), which provides several multi-mission space system simulation tools for the closed-loop development and testing of flight algorithms and software ([13]).

Dshell was developed for simulating real-time spacecraft dynamics, and it uses the DARTS engine for solving the dynamics. On top of Dshell is DSENDS - a Dshell-based simulator for entry, descent and landing (EDL) scenarios. DSENDSEdu is the educational version of DSENDS and is the simulation environment of choice in this work.

The model has mainly been coded through high-level Python wrappers that make use of lower-level C++ libraries. The final implementation consists of different Python stand-alone modules that allow for a higher flexibility, especially because of the Object Oriented Programming (OOP) paradigm: in case a change was needed in the geometry, dynamics, control logic or any other aspect of the project, only a few modules would need to be rewritten.

4.1 Spatial Operator Algebra

DSENDSEdu dynamics solver is based on a novel approach to model linked multi-body systems developed by the authors of the software, referred to as a Spatial Operator Algebra (SOA). SOA

enables a convenient representation of spatial kinematics and dynamics of complex systems. While mathematical complexity is reduced, no information about the system is lost in the Spatial Operator approach, as the spatial operators still contain precise information about the system. However, this information is organized and presented using much fewer symbols.

While in most traditional methods the number of arithmetical operations grows with the cube of the number of degrees of freedom ($\mathcal{O}(n^3)$), the computational complexity of SOA grows linearly with n, $\mathcal{O}(n)$.

Not only does using DSENDSEdu allow to reduce the computational complexity, but it also provides a convenient way to verify the results graphically, if needed.

4.2 Six-DOF model implementation

The first model to be implemented assumes a flat planet (used for data validation and for the following GNC algorithms). This laid the ground for other improvements and future model improvements. The following hierarchical tree structure was used:

```
Root_Body
└── Parafoil_Body
    ├── Canopy_Body
    └── Payload_Body
```

Where `Root_Body` represents the planet surface and is where the inertial frame belongs, `Parafoil_Body` is the physical/graphical model of the parafoil and `Payload_Body` is a cube with realistic mass and inertia properties associated to it.

`Parafoil_Body` and `Root_Body` are attached through a `FULL6DOF_INERTIAL` hinge type, which allows the body the freely move in the 3D space and allows to probe inertial values directly through the usage of `parentHinge().specNode()`.

The final implementation is shown in Fig. 4.1

Once a model is set up on DSENDSEdu, it is possible to extract any kinematic or dynamic property directly at any time instant. while Fig. **??** shows some of the properties that can be probed from the model, all referred to the inertial frame (which, in the case of the flat planet assumption, corresponds to a $\{ENU\}$ frame).

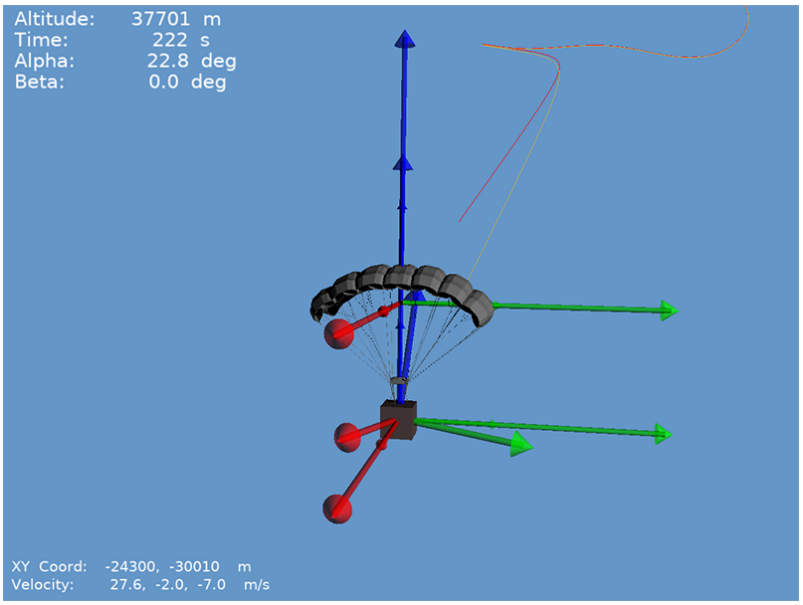


Figure 4.1: DSENDSEdu simulation.

PROPERTY	x	y	z	Unit
Canopy position	-21679.51	-29936.91	36844.78	m
Canopy attitude	-0.2799	3.4676	-3.0072	deg
Canopy velocity	26.8577	0.4590	-7.0351	m/s
	-0.0006	0.0000	0.0004	rad/s
Canopy acceleration	-0.0095	0.0029	0.0018	m/s^2
	0.0000	-0.0000	0.0000	rad/s^2
Payload velocity	26.8577	0.4579	-7.0351	m/s
Alpha (canopy)			22.7830	deg
Beta (canopy)			-0.0091	deg

Figure 4.2: Properties probed directly with DSENDSEdu.

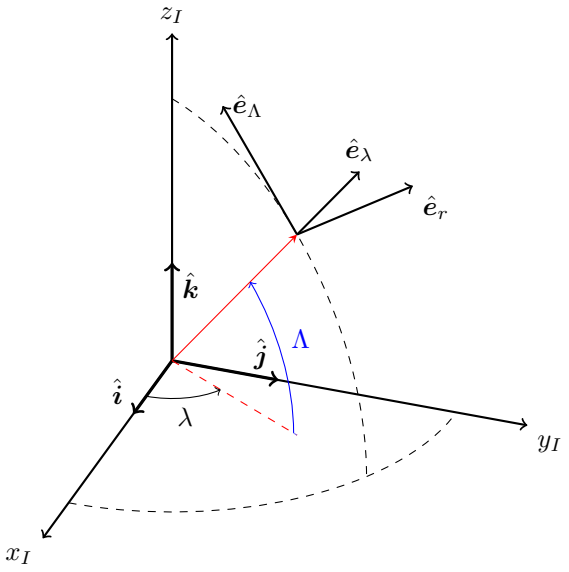


Figure 4.3: Inertial {I} and local geographic frame {G} frames of reference used in the spherical model.

4.3 Model improvement - spherical planet

The model was further expanded to allow for the sphericity of the planet to be taken into account. This allowed to simulate higher-altitude descents with added accuracy as well, when necessary.

Being Titan radius about 2575 km and considering an initial parafoil altitude of 40 km, it is straightforward to verify that the total descent accounts for about 1.55% of the radius. While this can be ignored during most simulations (especially if testing control strategies), it is not completely negligible. Plus, in case it was required to perform additional simulations from higher altitudes or for similar missions on different satellites, it is handy to have a spherical model set up for higher accuracy.

Instead of rewriting the whole dynamics from the bottom up, the spherical model has been obtained by expanding the previous model already implemented on DSENDSEdu. Using the transformations presented in Chapter 2, along with the definitions of [6], it is possible to revisit the model by considering the inertial frame $\{I\}$ placed at the center of the planet and expressing the position as a function of two angles and a distance, namely latitude, longitude and distance from the center of the planet.

The initial position must be now specified through the latitude λ, longitude Λ and initial R, which

is the sum of Titan radius R_t and initial altitude h. Note that latitude and longitude are commonly referred to as θ and ϕ in literature, this terminology was chosen to avoid confusion with the body rotations. Given these three parameters and referring to Fig. 4.3, Eq. 4.1 can be written

$$\begin{cases} \hat{\boldsymbol{e}}_r = \cos(\lambda)\cos(\Lambda)\,\hat{\boldsymbol{i}} + \sin(\lambda)\cos(\Lambda)\,\hat{\boldsymbol{j}} + \sin(\Lambda)\,\hat{\boldsymbol{k}} \\ \hat{\boldsymbol{e}}_\lambda = -\sin(\lambda)\,\hat{\boldsymbol{i}} + \cos(\lambda)\,\hat{\boldsymbol{j}} \\ \hat{\boldsymbol{e}}_\Lambda = -\cos(\lambda)\sin(\Lambda)\,\hat{\boldsymbol{i}} - \sin(\lambda)\sin(\Lambda)\,\hat{\boldsymbol{j}} + \cos(\Lambda)\,\hat{\boldsymbol{k}} \end{cases} \tag{4.1}$$

where $\hat{\boldsymbol{i}}$, $\hat{\boldsymbol{j}}$, $\hat{\boldsymbol{k}}$ are the unit vectors referred to the inertial frameof reference and $\hat{\boldsymbol{e}}_\lambda$, $\hat{\boldsymbol{e}}_\Lambda$, $\hat{\boldsymbol{e}}_r$ represent respectively latitude, longitude, and radial unit vector.

The explicit unit vectors derivates must be computed in order to express velocity and acceleration in spherical coordinates, even though only the velocity vector is needed for DSENDSEdu, since it takes care of the accelerations by itself. They are defined as follows:

$$\begin{cases} \dot{\hat{\boldsymbol{e}}}_r = \dot{\lambda}\cos(\Lambda)\,\hat{\boldsymbol{e}}_\lambda + \dot{\Lambda}\,\hat{\boldsymbol{e}}_\Lambda \\ \dot{\hat{\boldsymbol{e}}}_\lambda = -\dot{\lambda}\cos(\Lambda)\,\hat{\boldsymbol{e}}_r + \dot{\lambda}\sin(\Lambda)\,\hat{\boldsymbol{e}}_\Lambda \\ \dot{\hat{\boldsymbol{e}}}_\Lambda = -\dot{\Lambda}\,\hat{\boldsymbol{e}}_r - \dot{\Lambda}\sin(\Lambda)\,\hat{\boldsymbol{e}}_\lambda \end{cases} \tag{4.2}$$

Having defined the position at a generic time instant as $\boldsymbol{p} = R\,\hat{\boldsymbol{e}}_r$, the velocity can be obtained:

$$\boldsymbol{v} = \frac{d}{dt}(\boldsymbol{p}) = \dot{R}\,\hat{\boldsymbol{e}}_r + R\,\dot{\hat{\boldsymbol{e}}}_r = \dot{R}\,\hat{\boldsymbol{e}}_r + R\,\dot{\lambda}\cos(\Lambda)\,\hat{\boldsymbol{e}}_\lambda + R\,\dot{\Lambda}\,\hat{\boldsymbol{e}}_\Lambda \tag{4.3}$$

where $\dot{R} = \frac{d}{dt}(R_t + h) = \dot{h}$ is the descent rate of the parafoil.

The inertial coordinates can be obtained with simple projections as shown in Eq. 4.4.

$$\begin{cases} x = R\cos(\Lambda)\cos(\lambda) \\ y = R\cos(\Lambda)\sin(\lambda) \\ z = R\sin(\Lambda) \end{cases} \tag{4.4}$$

The resulting model is shown in Fig. 4.4 and, due to the spherical nature of the problem, it is now possible to simulate a more realistic landing scenario.

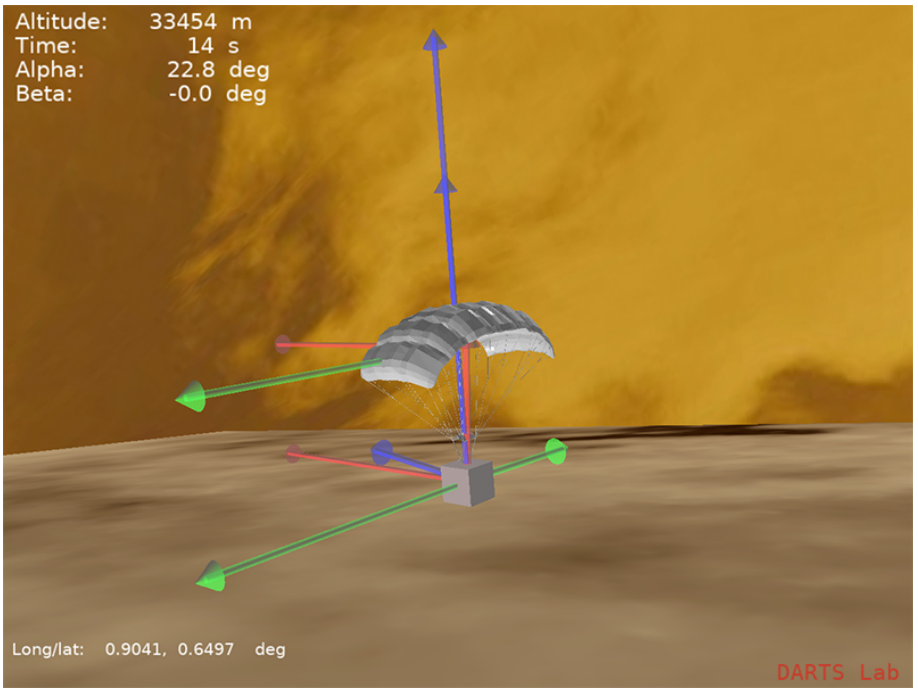


Figure 4.4: DSENDSEdu simulation – spherical planet.

4.4 Model validation

The model has been validated against other already-existing MATLAB® implementation (presented in [12]), and results were found to be consistent. A few plots of the system properties during a descent are shown here.

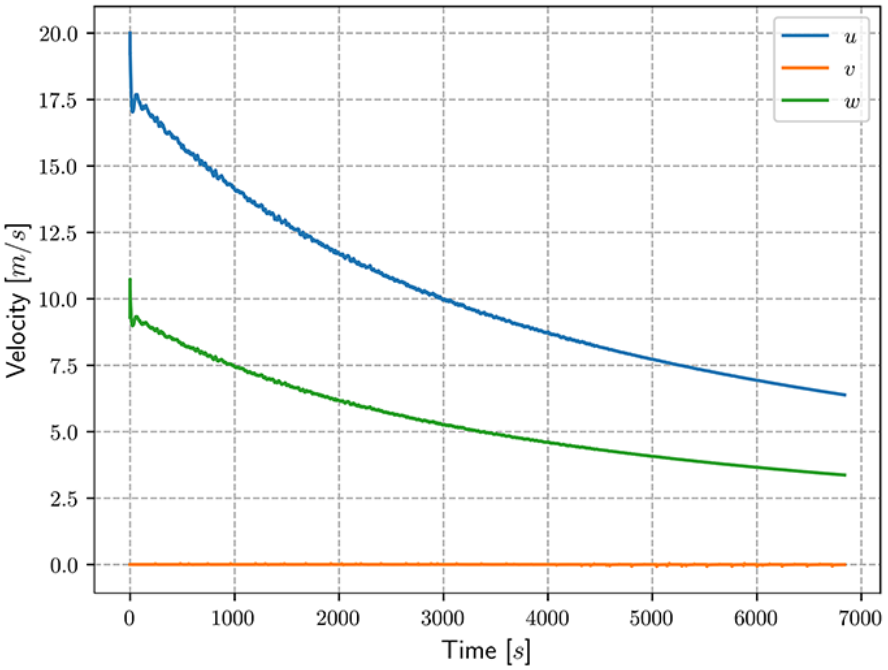


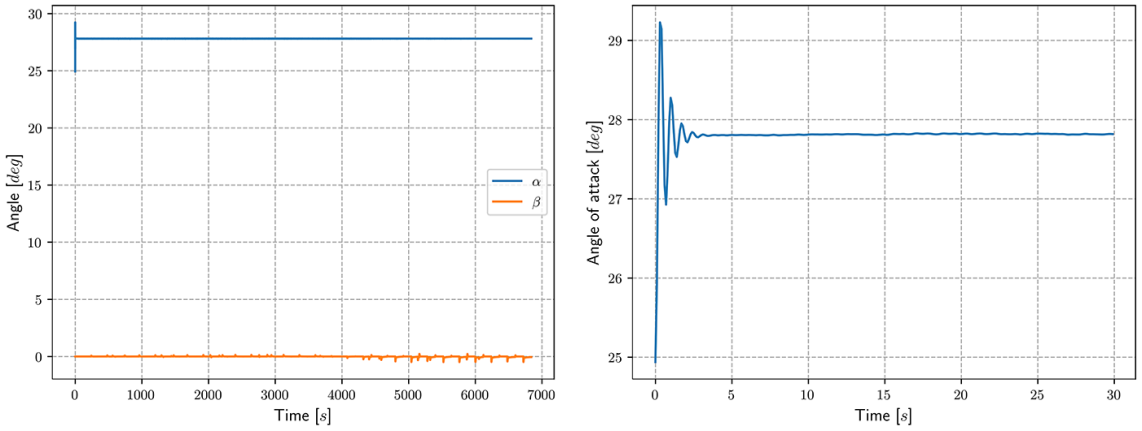
Figure 4.5: Descent velocity components.

4.5 Preliminary analysis

This chapter investigates the preliminary validations performed with DSENDSEdu, as well as the analysis of the reachable distance. The model has been compared with the one presented in [12], and results have been found to be coherent.

4.5.1 Gliding range - reachable distance

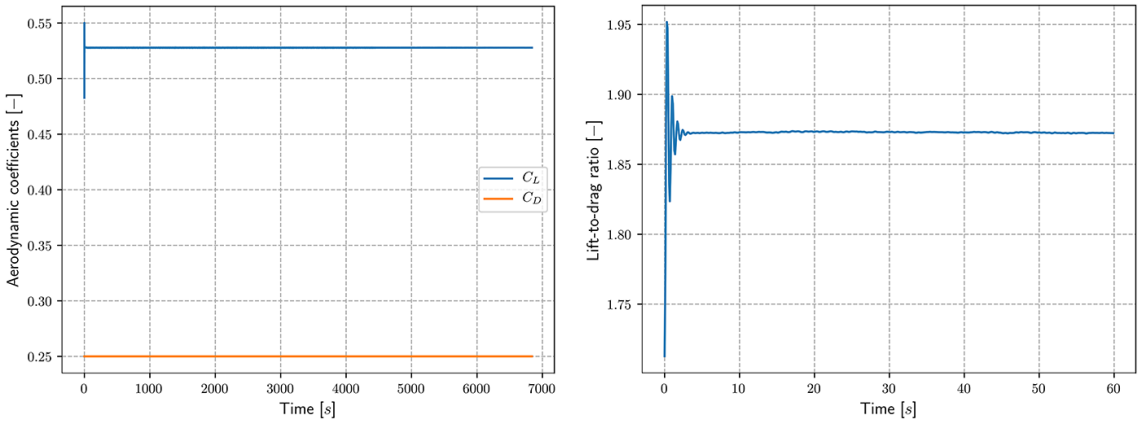
One of the most important aspects to take into account during the design phase of the Titan parafoil is its glide ratio $GR = L/D$. Different design characteristics have direct implications on this value which, in turn, directly affects the reachable distance. Beside the glide ratio, the



(a) α and β.

(b) α inititial transient.

Figure 4.6: Angle of attack and sideslip angle.



(a) C_L and C_D.

(b) C_L/C_D.

Figure 4.7: Lift and drag coefficients.

reachable distance is also influenced by wind magnitude and direction.

The aim of this section is to investigate:

1. The worst-case scenario divert range: considering a deployment condition followed by a 180 degrees turn, two different glide ratios are compared to highlight the different divert ranges
2. The influence of East-West and North-South wind on the divert range, with a fixed glide ratio

Also, since we know that given L/D and the deployment altitude h_0 we can compute the total

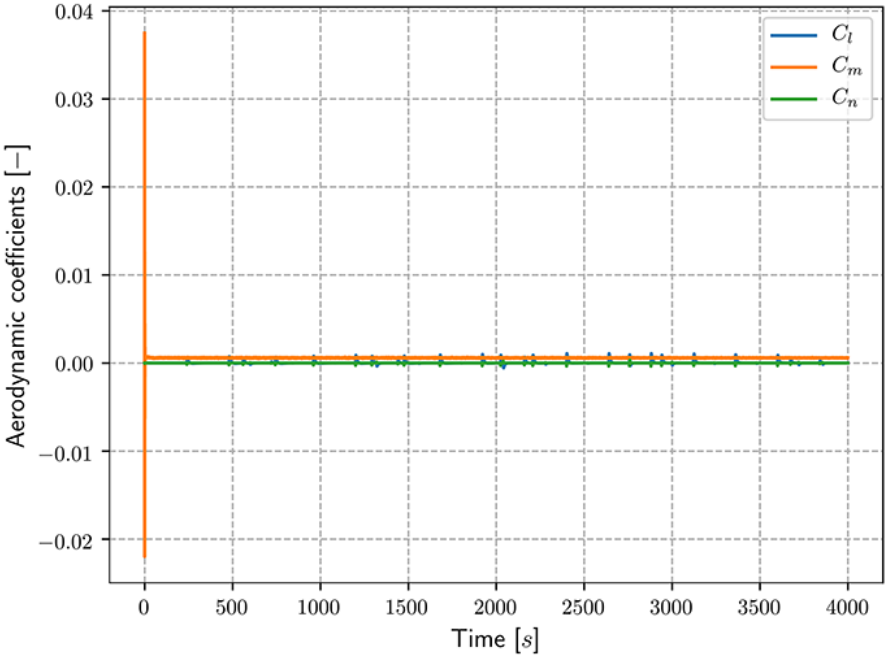


Figure 4.8: Aerodynamic moments coefficient.

gliding range as $\Delta X = \frac{L}{D} h_0$, this section serves to validate the model as well.

Titan winds mainly blow in the East-West direction, so that North-South winds can be neglected for this investigation.

The heading angle of the parafoil at deployment has been assumed to be directed towards East. As explained above, the first maneuver is a complete (180 degrees) turn in order to simulate the reachable distance in the worst possible situation. After this initial turn, the parafoil keeps pointing West and the rest of the landing involves free gliding.

The nominal wind conditions have been used for the simulations. For each investigated gliding range, three different wind profiles were considered: wind blowing towards West (parafoil moving downwind), null wind and wind blowing towards East (parafoil moving upwind).

As an example, Fig. 4.12 shows the final part of the descent of a parafoil with $L/D = 3$, to highlight the different landing points obtained in the different wind conditions.

Fig. 4.13 highlights the initial turn of the parafoil.

Table 4.1 shows the divert range for two different values of the glide ratio, starting from an initial

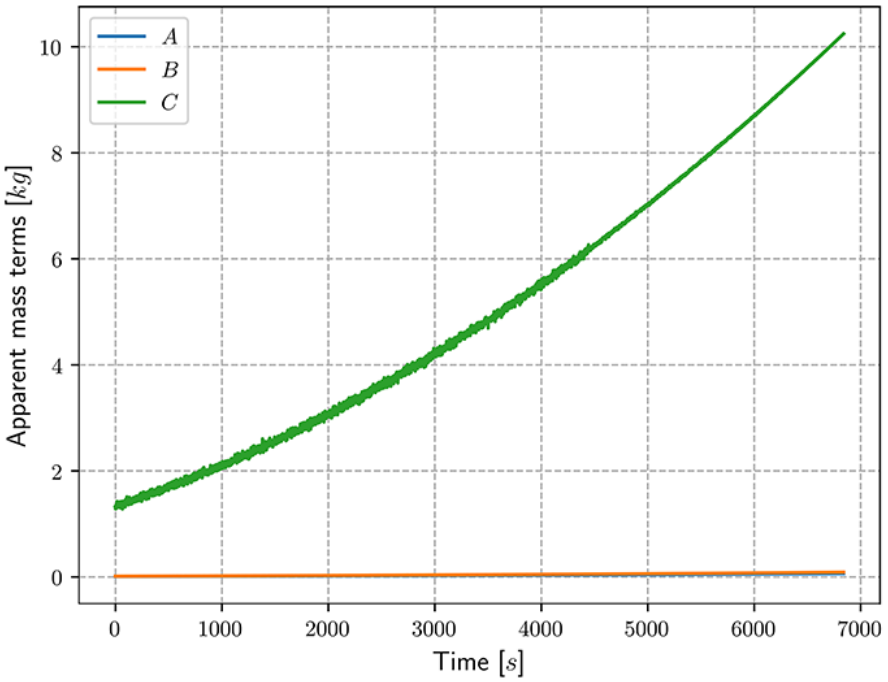


Figure 4.9: Apparent mass contributions.

altitude of 40 km.

Glide ratio (L/D)	Upwind divert range	No wind divert range	Downwind divert range
2	74406	77146	78102
3	113647	119734	122015

Table 4.1: Divert range [m] for the case of low and high glide ratio (nominal wind conditions, $h_0 = 40$ km).

4.5.2 Gliding range - wind influence

Next, the wind drift along every direction has been considered. Wind profile was changed both along the East-West and North-South direction, in the range $[-W_{max}, W_{max}]$, where W_{max} is the magnitude of the maximum wind speed. This way, the maximum wind was considered in every direction.

No initial turn was considered in the following simulations, the heading angle only changed because of lateral wind. The initial conditions were fixed, with a starting altitude of 40 km and a starting

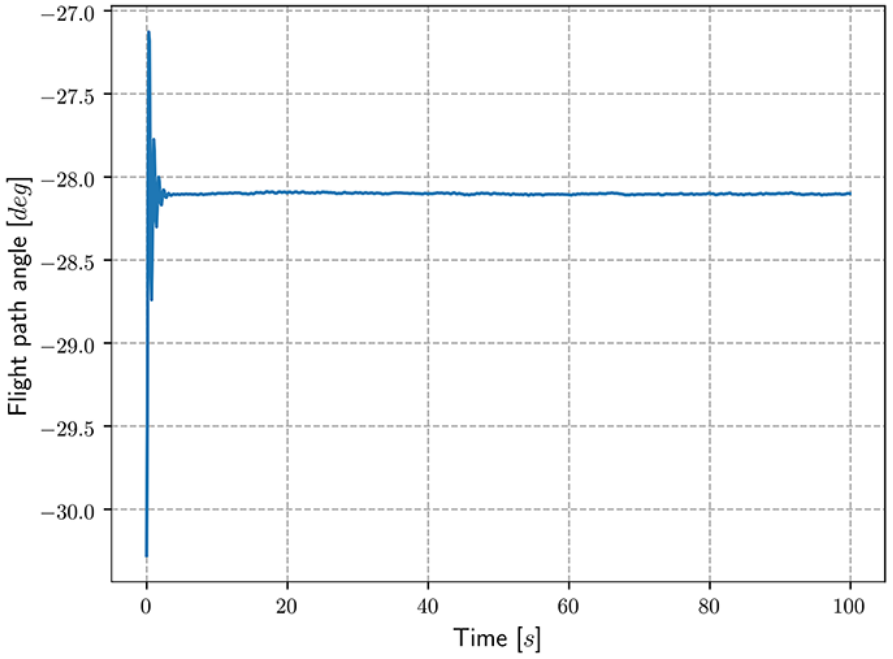


Figure 4.10: Flight path angle.

planar position of (-80, 0) km.

With a glide ratio equal to two, the no-wind landing is very close to the origin. With increasing longitudinal and lateral wind, the landing point moves further and further away from the origin. As expected, the plot is symmetrical with respect to the East-West axis.

In the ideal case with no wind, the divert range is directly proportional to the parafoil glide ratio L/D, as the theoretical model suggests. With the same initial altitude of 40 km, $L/D = 2$ and $L/D = 3$ imply a landing about 80 km and 120 km away from the starting point, respectively. Given a high initial altitude, the initial turn has little influence over the total divert range.

Also, as expected, the wind drift is *not* directly proportional to the glide ratio, meaning that doubling the glide ratio under the same wind conditions does not imply a doubling wind drift. This of course holds both for the upwind and downwind landing conditions. The reason lies in the exponential wind model whose profile, given the different landing times, has non-linear effects on the system.

Finally, lateral wind was shown to have noticeable effects on the final landing point. Although according to Fig. 4.16 wind can cause, in a few situations, the parafoil to drift as much as 50 km,

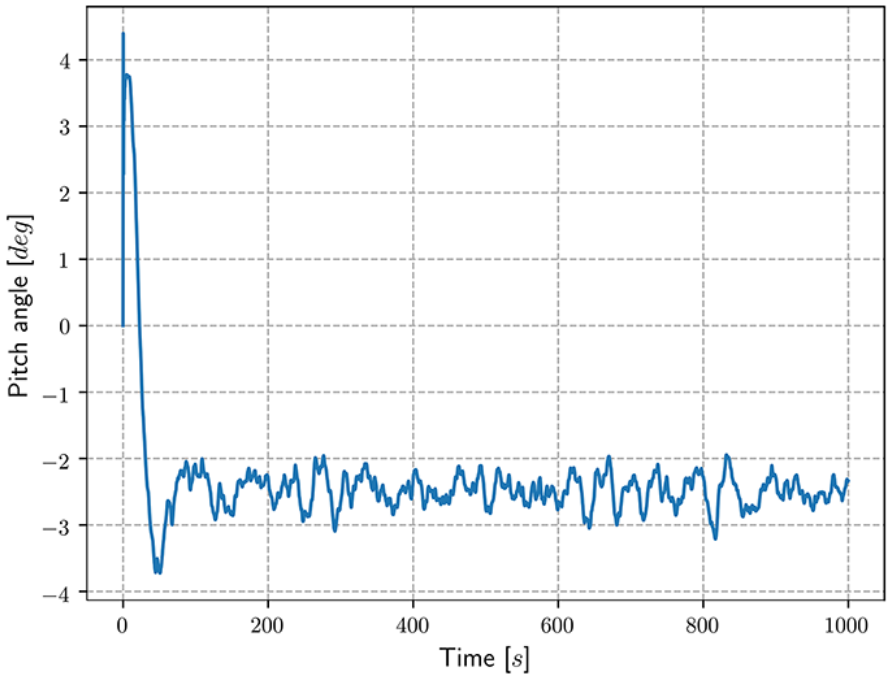


Figure 4.11: Pitch angle.

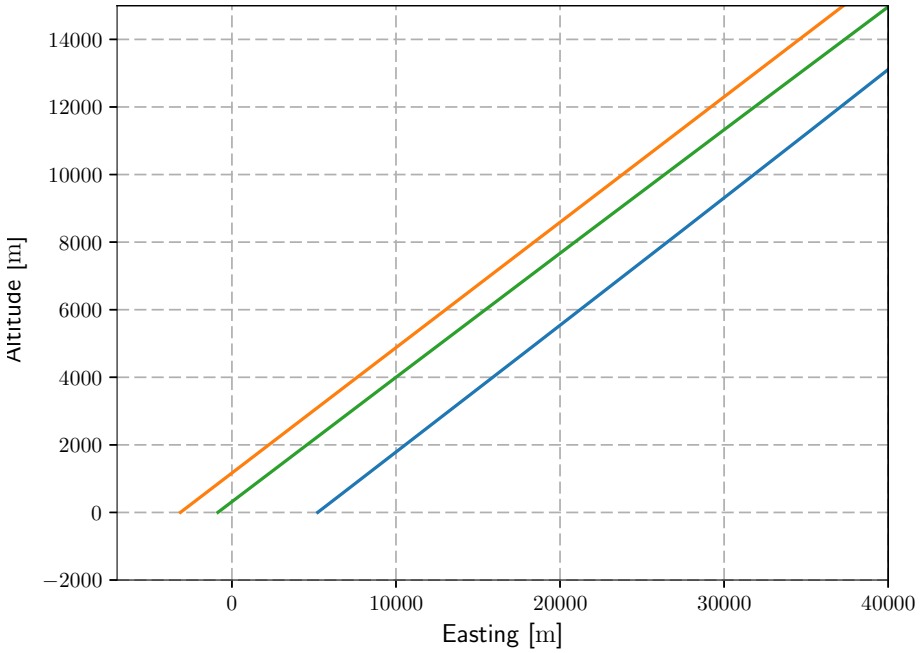


Figure 4.12: Terminal descent phase with $L/D = 3$. Downwind, no wind and upwind case.

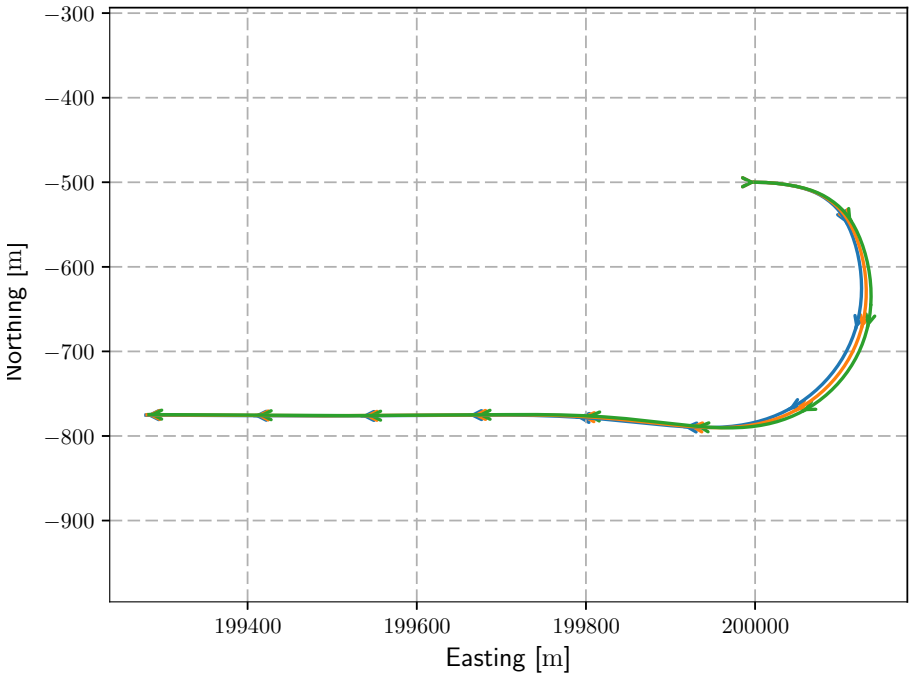


Figure 4.13: Initial turn with $L/D = 3$: downwind, no wind and upwind case.

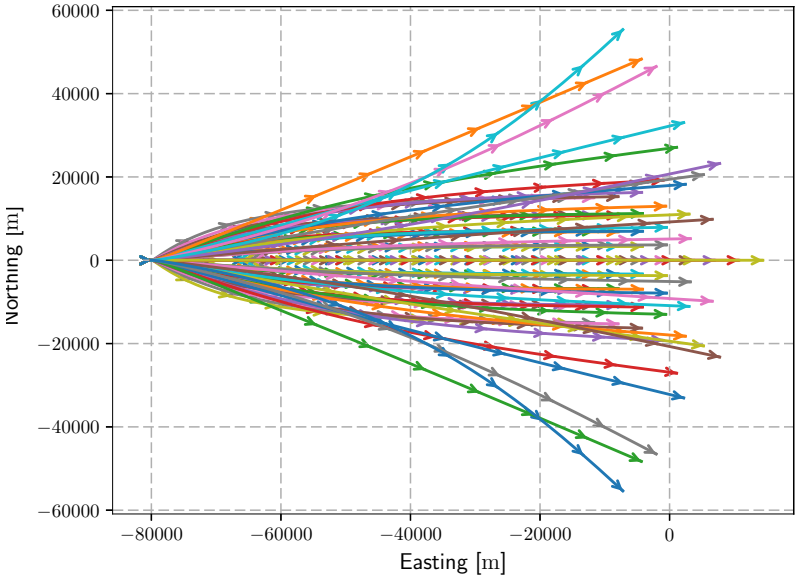
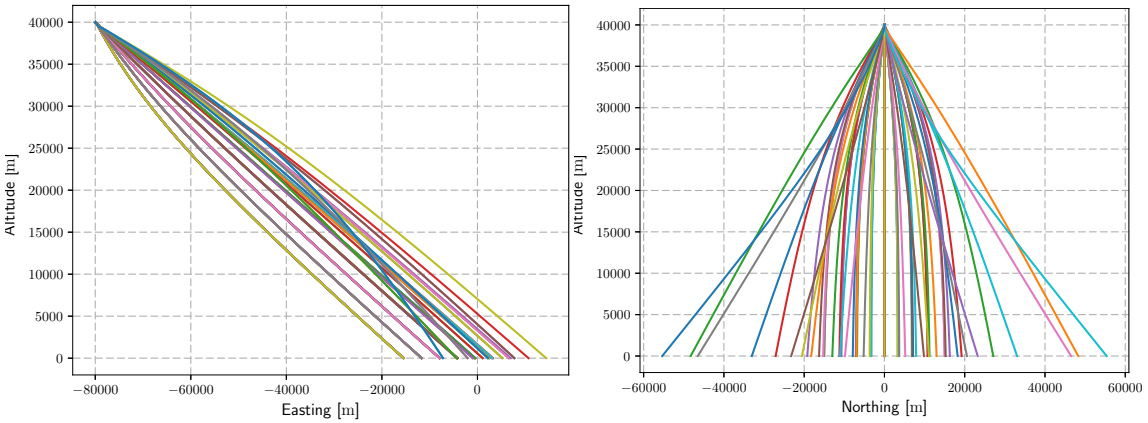


Figure 4.14: Longitudinal and lateral wind drift: full trajectories on $x - y$ plane.

it must be kept in mind that the wind speed causing those outliers is unrealistic and was only included to show a more complete overview.



(a) $x - z$ plane.

(b) $y - z$ plane.

Figure 4.15: Longitudinal and lateral wind drift.

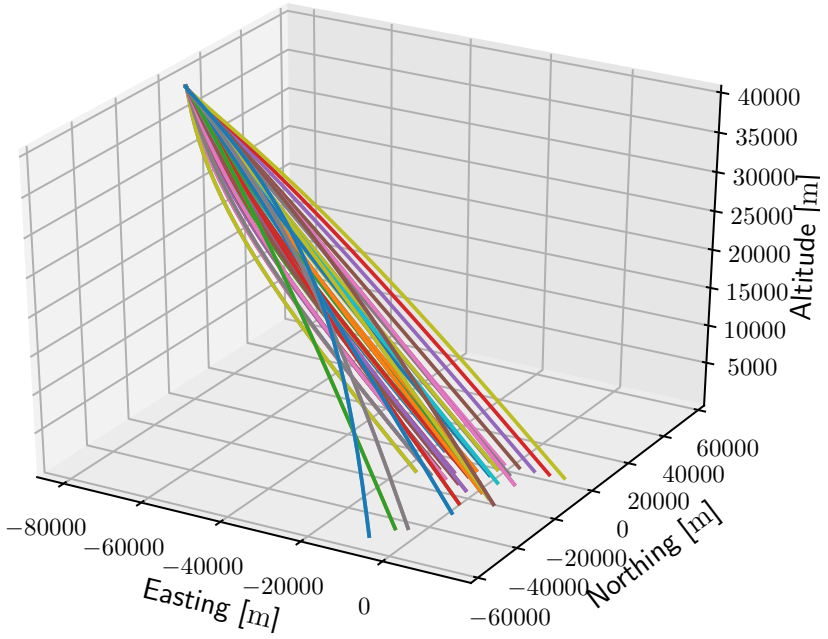


Figure 4.16: Longitudinal and lateral wind drift: full 3D trajectories.

The expected lateral drift during a real landing would likely be much lower, in particular:

- For nominal wind conditions, lateral drift is bounded to about 18 km North or South
- Nominal wind conditions refer to a West-East wind which is the worst-case scenario. If the parafoil was to follow a West-East (e.g. equatorial) trajectory, the lateral winds would be

North-South, which have been shown to be much lower.

4.5.3 Flare maneuver

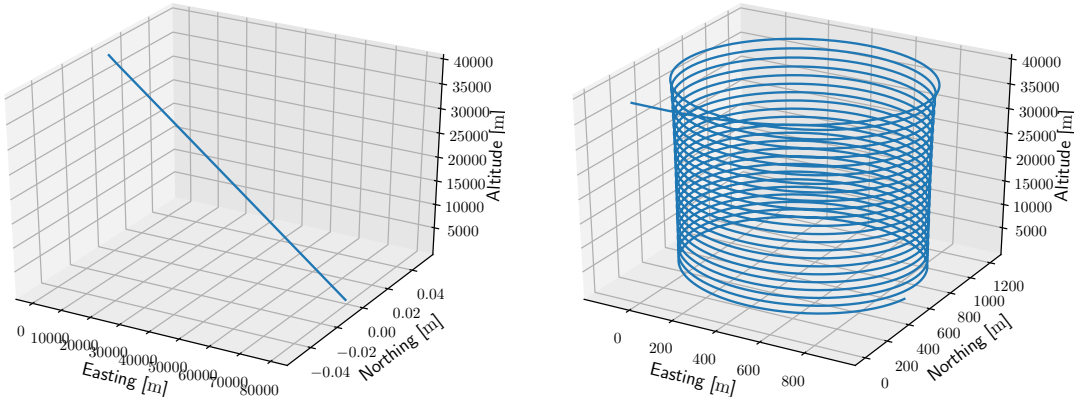
In order to minimize both the horizontal and vertical touchdown velocity, a flare maneuver must be performed during which both deflectors are actuated symmetrically at the maximum allowed value.

Thanks to this maneuver, the vertical landing speed has been shown to go, in case of slow wind, from 3.4 to 2.3 m/s, while the horizontal speed goes from 7.2 m/s to 4.5 m/s. In case of stronger winds, the effect is even more noticeable.

4.5.4 Total descent time estimation

In order to get a more accurate estimate of the final integration time, two simulations were run to compare the descent speed during a free gliding maneuver and during a constant-radius turn. Descent speed varies depending on the turning rate and, depending on the considered architecture, it can be twice as high as the one in normal conditions.

The two simulations can be seen in Fig. 4.17



(a) Free gliding.

(b) Continuous turn.

Figure 4.17: Simulations used to compute the average descent speed.

For each of the two simulations above, given a total descent time T_{tot} discretized in $N-1$ intervals, we can define Z_{vec} the vector containing the N altitude values corresponding to every time step and $V_{z,vec}$ the vector containing the corresponding N vertical velocity values. For each vector the

first element represent the value corresponding to $t_0 = 0$ s and the last element correspond the T_{tot}. It follows that every altitude can be mapped to an average descent speed for the remaining portion of the landing.

Given a generic time step n with $n \in \{1, 2, \ldots, N\}$, the corresponding altitude is $Z_{vec}(n)$ to which an average descent speed

$$V_{z,av}(n) = \frac{\sum_{i=n}^{N} V_{z,vec}(i)}{N - n + 1} \tag{4.5}$$

is associated.

This was run for both cases and, as expected, the average descent speed of the constant-radius turn was slightly higher than the one corresponding to the free gliding condition, independently of the altitude. The curves can be seen in Fig. 4.18.

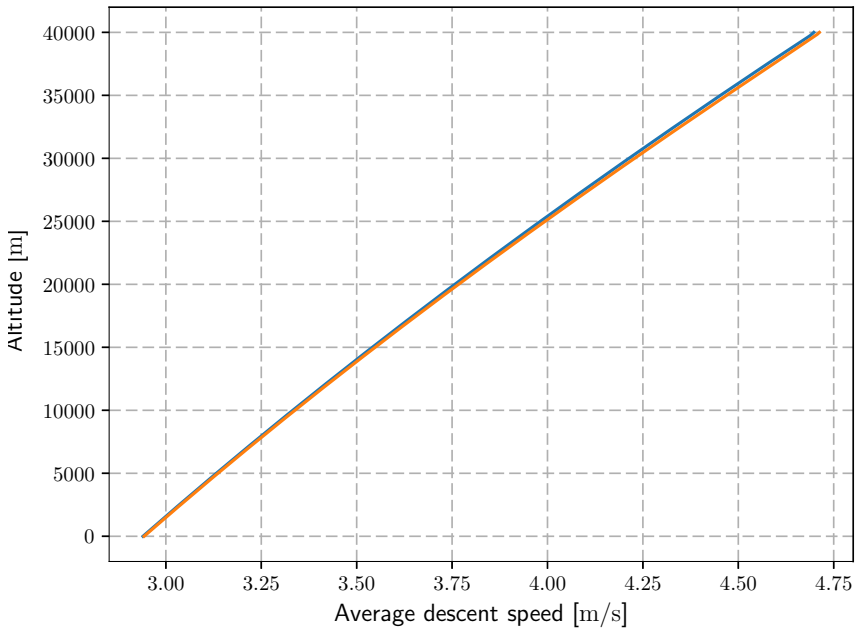


Figure 4.18: Average speed on the remaining portion of the landing. Free gliding condition (blue) and continuous turn (orange).

Using the above vectors as lookup tables is not computationally convenient, so a polynomial fitting was performed on the average of the two vectors. As Fig. 4.18 shows, the polynomial is not too far from being a line. The difference in goodness of fit for a third- and second-order polynomial

was minimal, so the latter was used. The resulting expression is:

$$V_{av}(h) = ah^2 + bh + c = 1.577 \cdot 10^{-10}h^2 + 3.791 \cdot 10^{-5}h + 2.941 \text{ [m/s]} \tag{4.6}$$

where h is the altitude considered. Given the above function and the initial altitude, the final time T_f can be estimated as h/V_{av}.

Given the local wind profile, Intended Point of Impact and deployment condition, one may wish to further formalize the above equation for taking wind speed into account. Although an in-depth analysis of this dependence is beyond the scope of this work, Fig. 4.19 shows how the average descent speed varies when moving downwind or upwind, as opposed to the no-wind condition.

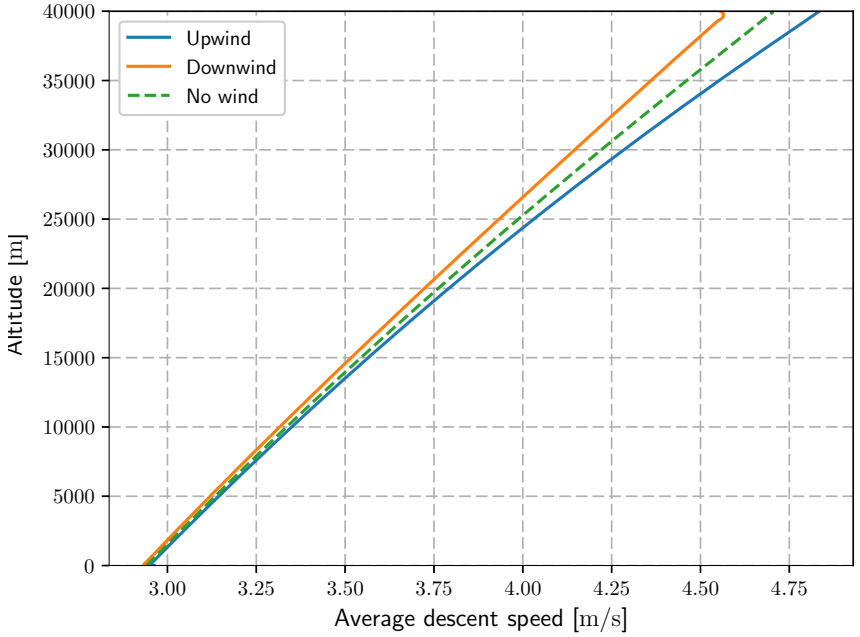


Figure 4.19: Average speed on the remaining portion of the landing: wind sensitivity.

Chapter 5

Point-to-target techniques

Having described and implemented all the necessary dynamic models in DSENDSEdu, it is possible to proceed with Guidance & Control techniques to guarantee an accurate landing. As mentioned before, the main challenge when it comes to landing on Titan (or any similar environment) is how to counteract the possibly strong winds.

Plausible operational conditions are considered in the following pages, which winds the resemble the ones adopted to estimate the gliding range of the parafoil in Chapter 4.

Not only is the presence of wind important in that it can lead to noticeable drifts, but it is also recommended that the final flare maneuver takes place against the wind to reduce the final velocity. While wind helps during this kind of maneuvers it could worsen the situation as well in case the landing happens to be downwind.

5.1 Proportional derivative control

The first controller to be implemented is based on a Proportional-Derivative feedback action which depends on χ and $\dot{\chi}$. A measure of the heading rate $\dot{\chi}$ is usually not available since it is expressed in the system wind frame. The assumption that is usually made to overcome this problem is that the sideslip angle is approximately zero during steady-flight conditions ($\beta \approx 0$). As a consequence we can write that:

$$\chi \approx \psi \qquad \text{and} \qquad \dot{\chi} \approx \dot{\psi} \tag{5.1}$$

.

In the following simulations, ψ is assumed to be known (obtained directly from the state vector) because during the real landing it would be computed by the TRN on-board system. $\dot{\psi}$ can in theory be estimated by differentiating the expression of ψ, however it is preferred to evaluate it by means of a backward difference:

$$\dot{\psi}_k = \frac{3\psi_k - 4\psi_{k-1} + \psi_{k-2}}{2\Delta t} \tag{5.2}$$

A four-point backward difference could also be used:

$$\hat{\dot{\psi}}_k = \frac{11\hat{\psi}_k - 18\hat{\psi}_{k-1} + 9\hat{\psi}_{k-2} - 2\hat{\psi}_{k-3}}{6\Delta t} \tag{5.3}$$

The generic feedback action is written as

$$\delta_a = K_p(\chi_{ref} - \chi) + K_d(\dot{\chi_{ref}} - \dot{\chi}) \approx K_p(\psi_{ref} - \psi) + K_d(\dot{\psi_{ref}} - \dot{\psi}) \tag{5.4}$$

with χ_{ref} ebing computed at every time step with `atan2`$(y_{ref} - y, x_{ref} - x)$. and using the `PID Tuner Toolbox` in MATLAB®, it is possible to compare the response of a Proportional and a Proportional-Derivative controller, as shown in Fig. 5.1. In particular, what is shown is a step response to a unitary change of the yaw angle (and consequently, due to the assumption above, heading angle).

The response above is obtained with a proportional gain $K_p = 1.95$ and a constant $K_d = 0.4$. The implementation of such a controller leads to the new poles configuration shown in Fig. 5.2. Although the system would remain stable even for a higher gain, it does not make sense to increase it, since the dominant poles have already move very close to the corresponding zeros.

Even though this controller has laid the foundations for implementing the T-approach guidance strategy, a simple control algorithm not based on waypoints has been implemented first for landing using the PD only.

The strategy works as follows. First, an initial point-to-target maneuver is performed to align the parafoil with the desired landing area, no matter where deployment happens. A "virtual" landing area (strictly downwind with respect to the Intended Point of Impact) is then selected based on the expected wind drift, and is engaged by the controller. The homing phase of the PADS lasts

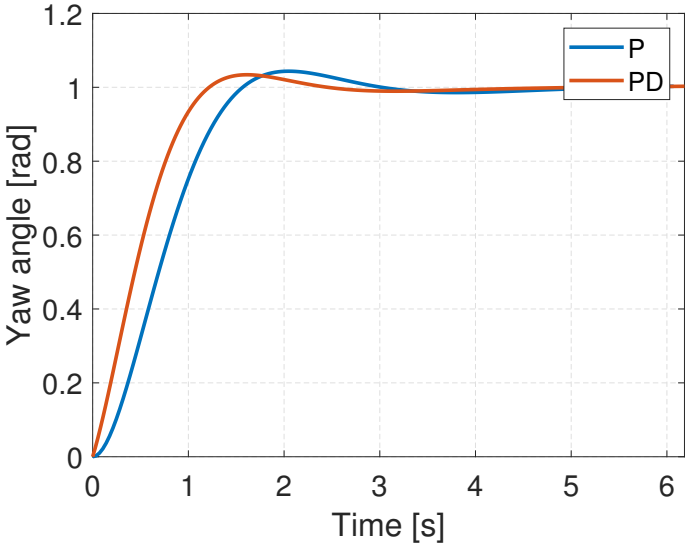


Figure 5.1: Proportional *vs* Proportional-Derivative controller.

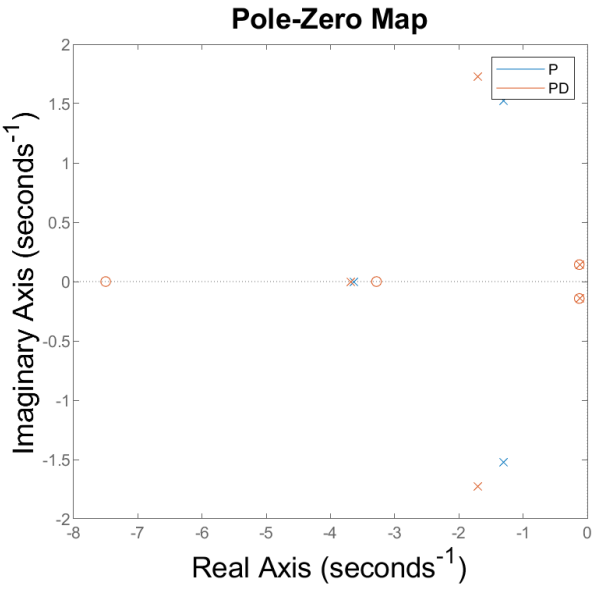


Figure 5.2: Poles of the system controlled with a Proportional and a Proportional-Derivative controller.

until the virtual landing area is reached (which is, the parafoil has come to at least 150 m from it). The energy management phase then starts around the virtual impact, which after trail and error procedures was set to be about 300 m downwind from the IPI. At this point the parafoil is performing spiraling maneuvers waiting for the appropriate time to trigger the exit maneuver for

the final approach. This is triggered by constantly checking the following condition:

$$h\mathrm{GR} < \pi R + w_{dir}\frac{h}{V_d} \tag{5.5}$$

where h is the current altitude, GR is the glide ratio, R is the minimum allowable turning radius, w_{dir} is the wind speed in the landing direction and, finally, V_d is the descent speed. The equation above assumes the average situation during which half a turn (π rad) will have to be performed during the exit maneuver to reach the IPI.

This strategy, although simple in its formulation, allows to obtain promising results, as Fig. 5.3 shows by highlighting the behavior of a bank of trajectory generated varying the entry point within a 5 km-radius.

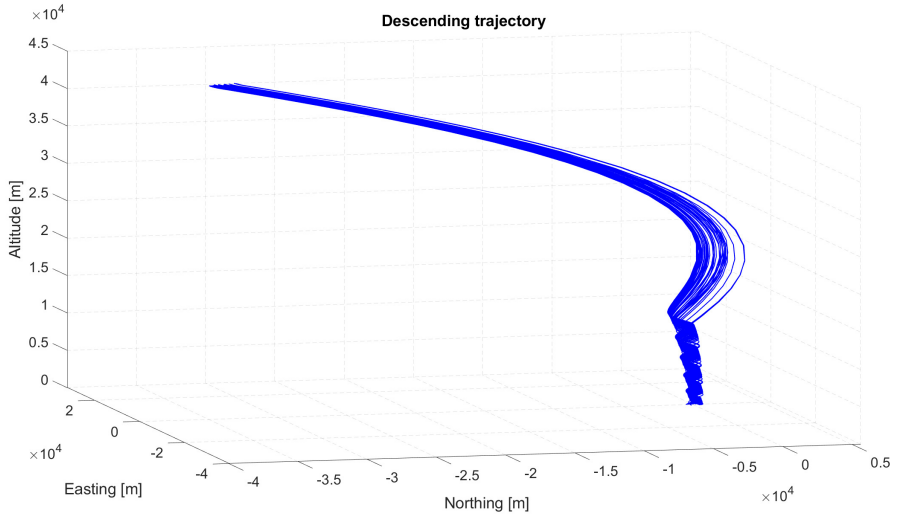


Figure 5.3: Bank of descent trajectories obtained with a PD controller: entry point within a 5 km radius.

5.2 T-approach guidance strategy

Although more advanced solutions (in particular optimal trajectories) will be discussed in later sections, a heuristic approach needs has been investigated first. This approach allows for a simpler solution which, nonetheless, can lead to good results. The T-approach guidance strategy was originally developed by DLR ([14]), and the trajectory is continuously updated based on the most recently available information about the environment. This way, uncertainties coming from wind or from some system parameters can be constantly compensated.

The main advantage of using a T-approach is the relatively simple implementation when compared to the generation of optimal trajectories, which translates to faster computational time that, in turn, allow online computation. Additionally, the algorithm allows to accommodate any sort of control strategies: the only information returned by the T-approach algorithm is which waypoint the parafoil should currently point to, but the control algorithm could be of any kind.

Different attempts have been made to use similar approaches to update the trajectory. Many of them rely on flying circles during the energy management phase, which allow to maximize the descent speed.[1] A T-approach adopts eight-shaped maneuvers instead. This allows, among other things, for a more reliable wind estimation [3]. While the following is based on the general idea presented in [3], the algorithm has been slightly modified, with the main difference being how wind is taken into account. Instead of basing the computation on a "virtual" Intended Point of Impact (IPI) in the wind-fixed frame of reference, the expected wind shift is computed at each step and an apparent IPI is created accordingly. This apparent IPI is then used as a base to place the remaining waypoints. All of this is performed directly in the inertial frame, as shown in Fig. 5.4.

Five waypoints are needed for performing the T-approach, as shown in Fig. 5.4 (although, depending on the phase, only one or two may be needed):

- **IPI**: Intended Point of Landing
- **FTP**: Final Turn Point
- **EMC**: Energy Management Center
- **EMTPs**: Energy Management Turning Points

A waypoint is considered cleared once the parafoil reaches the area inside a circumference centered in the physical waypoint.

[1] The descent speed during a turn might be as much as twice the one during a linear descent.

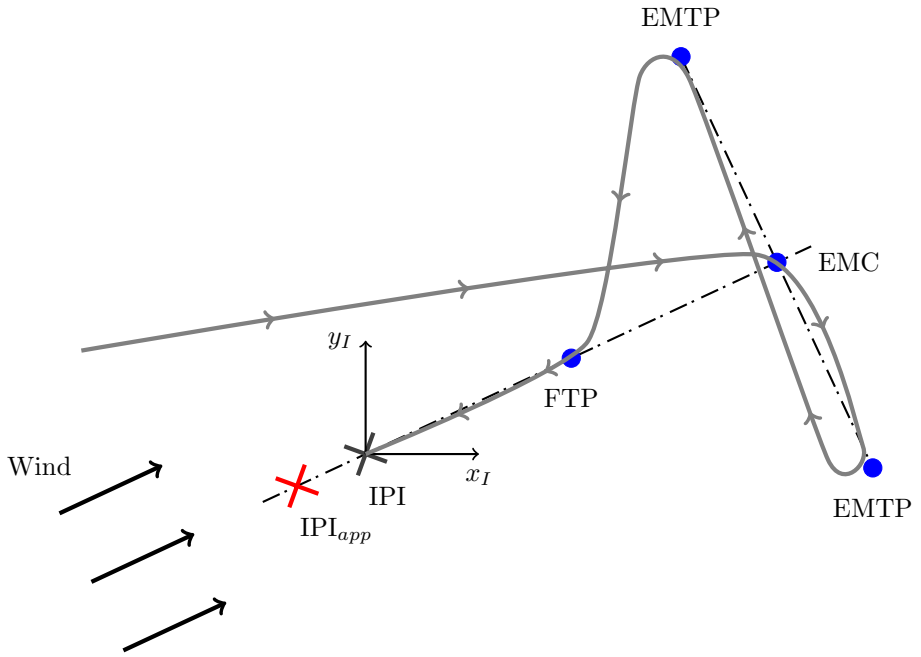


Figure 5.4: Waypoints needed for a T-approach landing.

The main idea is to divide the descent and landing in the following phases:

1. **Homing**: the parafoil navigates towards the EMC, if residual altitude is sufficient. If not, backup mode is entered.
2. **Energy management**: the surplus altitude is reduced by flying around EMTPs. When a prescribed exit altitude is reached or some exit condition is met, the parafoil starts flying towards FTP.
3. **Landing**: divided into three phases.
   (a) *Approach FTP*: fly towards FTP
   (b) *Turn into wind*: after passing FTP, keep pointing against IPI and prepare to land against the wind
   (c) *Flare maneuver*: avoid payload rollover and reduce final speed

At each step, based on the new estimation and available measurements, a new expected wind drift (in the inertial frame) is computed and the IPI is shifted accordingly to a new point called IPI$_{app}$ (apparent IPI). This allows to counteract the wind effect and land close to the IPI even in presence

of previously-unknown winds.

The most important aspect affecting the final landing accuracy is the exit condition, which must be triggered at the right instant. The decision to initiate this final maneuver is based on the condition:

$$h\mathrm{GR} < \pi R + s + w_x \frac{h}{V_d} \tag{5.6}$$

5.2.1 Implementation and results

Algorithm 1 was implemented in DSENDSEdu and is executed at every step until the parafoil has landed. The control architecture used to point at the correct waypoint is the Proportional-Derivative controller implemented in the previous section. Whatever flying mode or condition the parafoil is undergoing, Algorithm 1 will return the most appropriate waypoint to point at. Fig. 5.5 shows the detail of the eight-pattern executed during the energy management phase, which slowly advances towards the Intended Point of Impact.

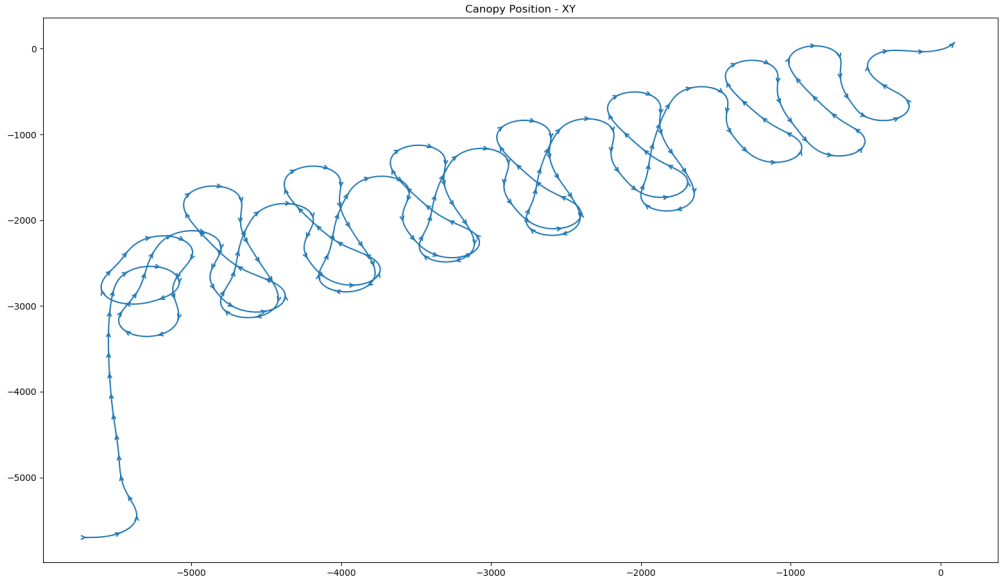


Figure 5.5: T-approach obtained trajectory: x–y.

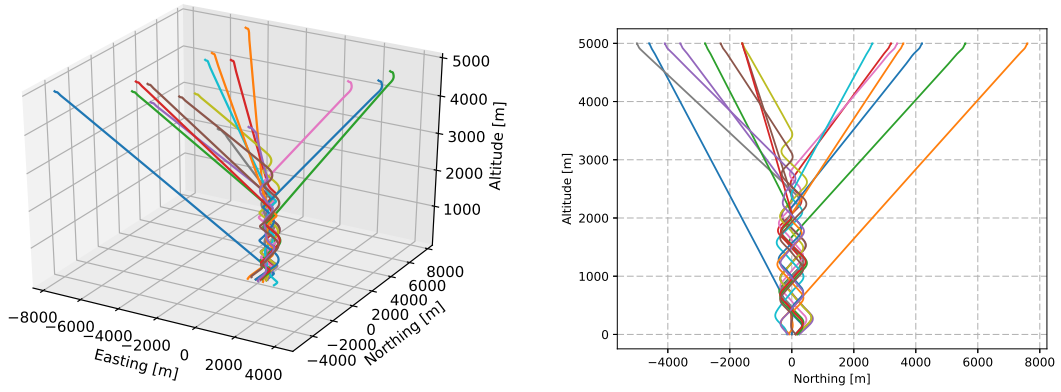
The algorithm has then been validated by letting the initial conditions vary randomly in the x-y plane and fixing the initial altitude at 5000 m, with c constant wind profile. Stochastic Gaussian-like wind gusts were added as well, with a standard deviation equal to 1 m/s. Fig. 5.6 shows

Algorithm 1 T-approach algorithm

```
1: xy_drift ← drift caused by the wind                          ▷ Computed at each step
2: IPI_app ← IPI − xy_drift                                     ▷ Apparent IPI
3: ϵ ← wind orientation
4: if (mode == "homing") or (mode == "energy-management") then
5:     FTP, EMC, EMTP_1, EMTP_2 ← waypoints
6:     if mode == "homing" then
7:         if z 'too low' then
8:             mode ← "backup"
9:         else if (EMC reached) or (z 'low enough') then
10:            mode ← "energy-management"
11:        else
12:            WP ← EMC
13:    if mode == "energy-management" then
14:        if remaining gliding distance 'too low' then
15:            mode ← "approach-FTP"
16:        else
17:            if heading exists then
18:                if heading == "EMTP_1" then
19:                    if EMTP_1 reached then
20:                        WP ← EMTP_2
21:                        heading ← "EMTP_2"
22:                    else
23:                        WP ← EMTP_1
24:                else
25:                    if EMTP_2 reached then
26:                        WP ← EMTP_1
27:                        heading ← "EMTP_1"
28:                    else
29:                        WP ← EMTP_2
30:            else
31:                heading ← initialize
32:                if EMTP_1 closer than EMTP_2 then
33:                    WP ← EMTP_1
34:                    heading ← "EMTP_1"
35:                else
36:                    WP ← EMTP_2
37:                    heading ← "EMTP_2"
38: if mode == "approach-FTP" then
39:    FTP ← waypoint
40:    if FTP reached then
41:        mode ← "turn-into-wind"
42:    else
43:        WP ← FTP
```

```
44: if mode == "turn-into-wind" then
45:     if z < h_flare then
46:         mode ← "flare"
47:     else
48:         if 'close' to IPI_app then
49:             WP ← IPI
50:         else WP ← IPI_app
51:     if mode == "backup" then
52:         if (z < h_backup) and (z > h_flare) then
53:                                          ▷ turn against wind and prepare for landing
54:         else if z <= h_flare then
55:             mode ← "flare"
56:         else
57:             FTP ← waypoint
58:             WP ← FTP
59:     if mode == "flare" then
60:         WP ← IPI
61: return WP
```

the resulting trajectories, while Fig. 5.7 shows the polar landing dispersion from which it can be inferred that the maximum landing error is limited to about 500 m.



(a) 3D trajectories.

(b) x-z view.

Figure 5.6: T-approach validation with different initial conditions.

Two full-descent trajectories were then simulated assuming a deployment altitude equal to 40 km. The same wind profile blowing East to West was used for both landings, with the only difference being that diametrically opposed starting positions were chosen on the x-y plane. Fig. 5.8 shows the resulting trajectories.

The invariance of the implemented algorithm to different wind conditions was tested next, by fixing

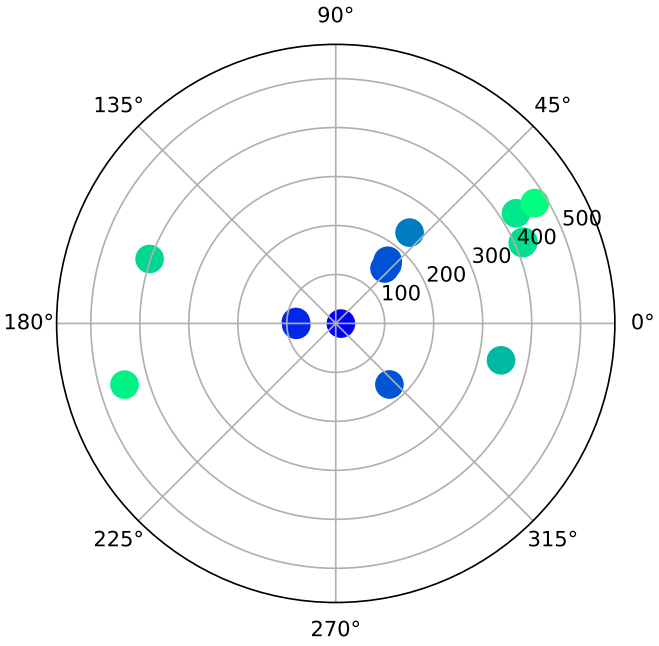
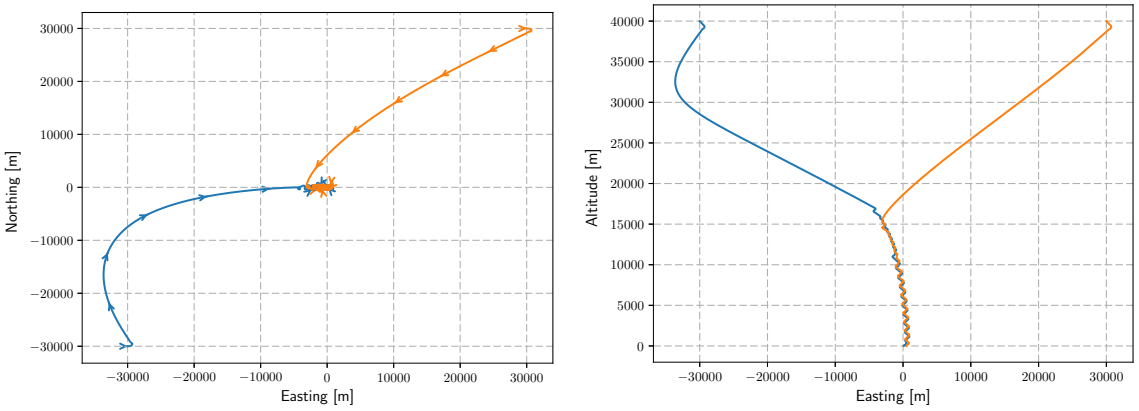


Figure 5.7: Polar landing dispersion (wind blowing from 180° towards 0°).



(a) x-y view.

(b) x-z view.

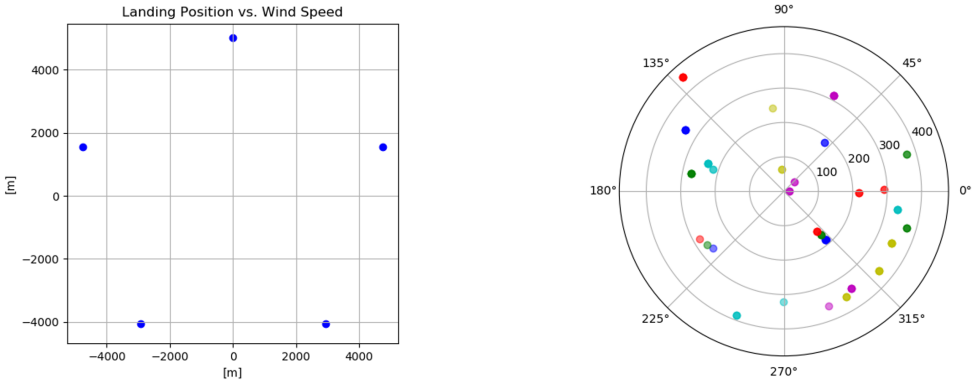
Figure 5.8: 40 km descent using the T-approach.

five different deployment point at a 5 km altitude and letting the wind speed vary. The five chosen configurations allow the parafoil to experience wind coming from any direction during the descent, since the wind direction was kept constant blowing towards East. Stochastic Gaussian-like wind gusts were added also in this simulation, with the same standard deviation specified above. Fig. 5.9a shows where the five deployment points have been placed on the plane, while Fig. 5.9b shows

the landing dispersion of multiple drops under six different wind conditions:

- *blue* → wind speed = 0 m/s
- *green* → wind speed = 0.25 m/s at 5 km altitude
- *red* → wind speed = 0.5 m/s at 5 km altitude
- *light blue* → wind speed = 1.0 m/s at 5 km altitude
- *purple* → wind speed = 2.0 m/s at 5 km altitude
- *yellow* → wind speed = 4.0 m/s at 5 km altitude

No matter what wind condition is faced, the landing points are still all within the same less-than-500-meters radius.



(a) Deployment conditions.

(b) Landing dispersion (different colors represent different wind conditions).

Figure 5.9: T-approach invariance to wind speed – plots.

Fig. 5.10 presents quantitatively the results shown in the previous plots. In particular, 5.10a reports the landing coordinates x and y (needed because of the wind dominance towards East) and 5.10b shows the total landing error $\rho = \sqrt{x^2 + y^2}$.

			Wind [m/s]											
			0.00		0.25		0.50		1.00		2.00		4.00	
			x	y	x	y	x	y	x	y	x	y	x	y
Initial x, y Position [m]	4755	1545	-286	179	-267	52	-293	332	-220	80	144	279	312	-151
	0	5000	123	-141	109	-126	96	-116	328	-52	196	-282	277	-231
	-4755	1545	121	-140	356	-106	218	-5	-137	-360	15	-0.5	181	-306
	-2939	-4045	118	140	357	107	290	6	-207	64	29	27	-7	63
	2939	-4045	-205	-164	-222	-155	-222	-139	-2	-320	131	-334	-34	241

(a) Landing coordinates – multiple starting points with different wind speeds.

			Wind [m/s]					
			0.00	0.25	0.50	1.00	2.00	4.00
Initial x, y Position [m]	4755	1545	336	274	443	234	314	347
	0	5000	187	167	150	332	343	361
	-4755	1545	184	372	218	386	15	356
	-2939	-4045	184	373	290	217	40	63
	2939	-4045	263	270	282	320	359	243

(b) Landing error – multiple starting points with different wind speeds.

Figure 5.10: T-approach invariance to wind speed – results.

Chapter 6

Optimal trajectory

Two different approaches can be used to deal with the control problem. One is to adopt a motion planning approach and design a controller to track the generated trajectory in the best possible way; the other is to design a controller that plans part of the trajectory at every step and constantly adjusts the prediction based on current noise (mainly wind action).

This chapter focuses on the first approach, with different motion planning techniques being considered. In Chapter 7 the appropriate control algorithms needed to maintain these trajectories will be described, along with control schemes that make use of the second approach.

Trajectory planning or *guidance* (i.e. the generation of a descent trajectory) is one of the most important aspects of the mission. This part is critical because, assuming the deployment conditions will not be known in advance, the computation of the most appropriate trajectory must be performed online. Computation is performed following atmosphere entry, and might have to be performed more than once depending on the circumstances. The optimal trajectory to follow depends on multiple factors, with the main ones being:

- deployment conditions (i.e. position, attitude and velocity)
- atmospheric properties (wind speed, density, wind gusts if present)
- physical constraints (maximum control actions, terrain obstacles, etc.)
- energy consumption (i.e. control effort)
- desired landing conditions (mainly position, but also attitude to land upwind)

In general, it is not necessary to plan the whole descent trajectory at once. The descent and landing part of the mission can be divided into five distinct phases:

1. **Initial turn**: since the deployment condition is not known with certainty, the first step to take is to redirect the parafoil towards the correct direction with an initial turn-to-target maneuver.
2. **Homing phase**: this phase involves a straight-line navigation towards the Intended Point of Impact (IPI).
3. **Energy management**: once the parafoil is close enough to the IPI, the aim is to stay as close as possible to it. During this phase eight-shape (or S-like) maneuvers are performed to dissipate potential energy while staying above the IPI (and, as it will be shown, the optimal trajectory will convergence to this pattern by itself).
4. **Final approach**: this is the most critical and interesting part to address, since the final landing error mainly depends on its robustness and noise-rejection properties. Different approaches have been carried out in literature and others are presented in the present work.
5. **Flare maneuver**: needed to reduce touchdown velocity and avoid payload roll-over.

The classification above describes the most general case, in reality the sequence may change slightly. For example, if deployment is above the IPI, the energy management phase may trigger immediately, with no need to perform an initial turn nor an homing phase. In a similar manner, the deployment is far enough from the IPI, the energy management phase may be skipped due to lack of altitude.

The initial homing phase (i.e. flying towards the Intended Point of Impact) is similar in almost all guidance maneuvers, due to the fact that the objective is trying to maximize the altitude reserve available for the final approach, which is handled by the energy management phase.

Usually energy management is performed by repeating a series of similar patterns until a switch to the final landing phase is triggered.

Once terminal guidance is engaged, additional precision is required because multiple constraints must be satisfied together at touchdown (e.g. landing must be as close as possible to the Intended Point of Impact, with a desired final heading that depends upon wind conditions).

The present chapter will mainly address aspects concerning the optimization of the final approach.

Before the different techniques are presented, a word must be spent on how wind effects are included in the model.

6.1 Wind accommodation

Taking into account wind is crucial when dealing with PADS, especially if winds are strong. While very detailed atmospheric models for wind are available on Earth, the same can't be said for Titan. Nonetheless, having a general idea of the average component of winds allows to plan the PADS trajectory in the wind-fixed coordinate system $\{W\}$ and consider a virtual parafoil position based on the predicted wind drift. Once this shift has been considered, the PADS trajectory is computed as if there was no wind.

Wind drift is defined as follows

$$\Delta \boldsymbol{s} = \begin{bmatrix} \Delta x^W \\ \Delta y^W \end{bmatrix} = \int_{t_0}^{t_f} \begin{bmatrix} w_x(t) \\ w_y(t) \end{bmatrix} dt \tag{6.1}$$

Wind is usually expressed as a function of altitude, and the position is not known a priori at every time step, until the optimization is done. Assuming $t_0 = 0$ and expressing dt as a function of the vertical descent speed, Eq. 6.1 is rewritten as

$$\Delta \boldsymbol{s} = \int_0^h \begin{bmatrix} w_x(z) \\ w_y(z) \end{bmatrix} \frac{dz}{V_v + w_z(z)} \approx \frac{hW^{bal}(h)}{V_v} \begin{bmatrix} \cos\left(\chi_w(h)\right) \\ \sin\left(\chi_w(h)\right) \end{bmatrix} \tag{6.2}$$

where $\approx$ is used due to assumption that the descent rate V_d is about the same as the vertical component of the airspeed vector V_v: $V_d(h) = V_v(h) + w_z(h) \approx V_v(h)$ (again, w_z is usually negligible) [3].

Δx^W and Δy^W are shown in Fig. 6.1, which also shows a rotation of the coordinate system needed to align the x axis with the wind direction, so landing against the wind always means landing along the x direction.

6.2 Optimal terminal trajectory

The aim of this section is to compute optimal trajectories suitable for the terminal guidance of the parafoil. Given an initial configuration defined by the initial position and heading of the parafoil

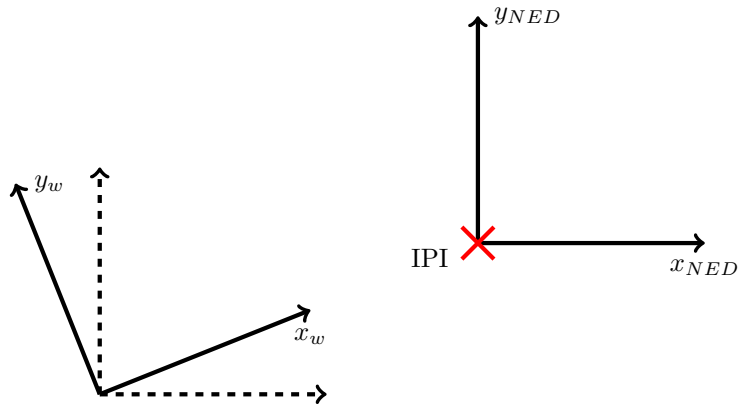


Figure 6.1: Wind-fixed frame used for motion planning.

$\boldsymbol{X}_0 = [x_0, y_0, z_0, \psi_0]^T$, as well as the desired final configuration $\boldsymbol{X}_f = [x_f, y_f, z_f, \psi_f]^T$ (where h_f is usually equal to zero), the goal is to generate an optimal trajectory to guide a parafoil from $\boldsymbol{X}_0$ to $\boldsymbol{X}_f$. Note that $\boldsymbol{X}_0$ and $\boldsymbol{X}_f$ don't necessarily correspond to the full initial states $\boldsymbol{x}_0$ and $\boldsymbol{x}_f$, but are used to collect the variables that we are usually interested in bringing to a certain value.

This section starts by considering a three-DOF representation of the parafoil. Such a representation allows to obtain an optimal trajectory in a relatively short time and, provided the appropriate parameters are available, it can mirror correctly the real behavior of the full parafoil. The three-DOF model also allows to present the matrices in their full form since the expressions are relatively simple. In the case of the six-DOF model which will be presented later, the matrices have been obtained in symbolic form using MATLAB® instead.

The optimal strategy is based on reaching an area above the Intended Point of Impact (IPI) as quickly as possible through minimum-path trajectories and then continue the descent by tracking the optimal trajectories obtained by integrating the Euler-Lagrange equations. The techniques used to track the generated trajectories are presented in Chapter 7.

This approach decouples the two guidance and control phases, so that motion planning and trajectory following can be tackled independently. The two strategies can also be used independently and waypoint-tracking techniques can still be employed independently of how the optimal trajectories are obtained.

Techniques such as infinite horizon Nonlinear Model Predictive Control have been studied for similar problems, but the they require to solve a nonlinear, possibly non-convex problem online for

an extended time horizon, which can require a tremendous amount of computational resources. By decoupling the problem, multiple optimal trajectories can be computed in advance and the most suitable one can be tracked appropriately by means, for example, of a standard Finite-Horizon Linear MPC.

6.2.1 Initial homing phase

During the first part of the descent – homing – it is highly desirable to reach a x-y position close to the Intended Point of Impact (IPI) as fast as possible. This, in fact, allows to maximize the residual altitude available during the energy management phase and can, in turn, be obtained by minimizing the time needed to approach the landing point. The minimum time problem is presented in [15] and [3].

Since, due to the deployment conditions, the initial phase of the descent is at high altitudes, it is here assumed that enough residual altitude is available to reach the $x-y$ area without touching the ground before. Even if this hypothesis didn't hold, the fact that we are dealing with a minimum-time trajectory implies that there would be no better solution to get as close as possible to the Intended Point of Impact (IPI).

Keeping in mind that, as shown previously, we must deal with wind and plan the trajectory in the wind frame $\{W\}$ whose shift is obtained through the estimated average components of wind, the trajectory can be planned considering a planar problem. As little as three differential equations (shown in Eq. 6.3) are sufficient to describe the problem.

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V_h \cos(\psi) \\ V_h \sin(\psi) \\ u \end{bmatrix} \tag{6.3}$$

We are interested in bringing the PADS from $\boldsymbol{X}_0 = [x_0, y_0, \psi_0]^T$ to $\boldsymbol{X}_f = [x_f, y_f, \psi_f]$ by varying a bounded control input $u < u_{max}$ which, in the specific case, corresponds to $\delta_a < \delta_{a,max}$ (responsible of controlling the lateral dynamics).

Pontryagin's Maximum Principle allows to find the optimal path by minimizing

$$J = \int_{t_0}^{t_f} L(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt = \int_{t_0}^{t_f} dt \tag{6.4}$$

The Hamiltonian can be written as (note the negative sign in front of L due to the adoption of the Maximum Principle formulation):

$$H = \boldsymbol{\lambda}^T \cdot f\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) - L\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) = \lambda_x V_h \cos\left(\psi\right) + \lambda_y V_h \sin\left(\psi\right) + \lambda_\psi u - 1 \tag{6.5}$$

with

$$\begin{bmatrix} \dot{\lambda_x} \\ \dot{\lambda_y} \\ \dot{\lambda_\psi} \end{bmatrix} = - \begin{bmatrix} \partial H/\partial x \\ \partial H/\partial y \\ \partial H/\partial \psi \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ V_h\left(\lambda_x \sin\left(\psi\right) - \lambda_y \cos\left(\psi\right)\right) \end{bmatrix} \tag{6.6}$$

The optimality condition of maximizing the derivative $\partial H/\partial u$ allows to reduce the problem to ([3]):

$$u_{opt} = \max_{|u| \leq u_{max}} \left(\lambda_\psi u\right) \tag{6.7}$$

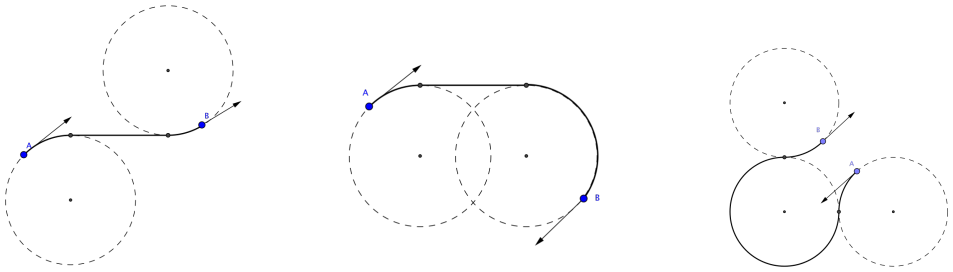
The solution to this problem shows that any time-optimal trajectory consists of at most three motion primitives, obtained with any of the following control actions:

$$u_{opt} = \begin{cases} u_{max}, & \text{if} \quad \lambda_\psi > 0 \\ 0, & \text{if} \quad \lambda_\psi = 0 \\ -u_{max}, & \text{if} \quad \lambda_\psi < 0 \end{cases} \tag{6.8}$$

Depending on which control action is applied, the resulting motion will either be a minimum-radius (maximum-curvature) turn (either left or right, depending on the sign of the control action) or will follow a straight path. This solution is known as bang-bang control and is equally obtainable by solving the shortest-path problem described by Eq. 6.9

$$J = \int_{t_0}^{t_f} \sqrt{\dot{x}^2 + \dot{y}^2} dt \tag{6.9}$$

which was used to obtain the curves commonly known as Dubins path, shown in Fig. 6.2 (where R, L and S indicate a right turn, left turn and straight portion, respectively).



(a) RSL path (b) RSR path (c) LRL path

Figure 6.2: Examples of Dubins paths.

6.2.2 Problem formulation

As specified above, the problem will be presented using the three-DOF model which allows to compute a fast online solution, with the six-DOF model results being presented in the next section. The equations governing the dynamics of the system are the same presented in Eq. 3.8, and are shown in Eq. 6.10

$$\begin{cases} \dot{V}_a = -\dfrac{1}{m}\left(D + mg\sin\left(\gamma_a\right)\right) \\ \dot{\gamma}_a = \dfrac{1}{mV_a}\left(L\cos\left(\phi_a\right) - mg\cos\left(\gamma_a\right)\right) \\ \dot{\chi}_a = \dfrac{L\sin\left(\phi_a\right)}{mV_a\cos\left(\gamma_a\right)} \\ \dot{x} = V_a\cos\left(\gamma_a\right)\cos\left(\chi_a\right) + w_x \\ \dot{y} = V_a\cos\left(\gamma_a\right)\sin\left(\chi_a\right) + w_y \\ \dot{z} = V_a\sin\left(\gamma_a\right) + w_z \end{cases} \tag{6.10}$$

where, for convenience, the translational kinematics has been included directly in the system dynamics.

A cost function

$$J = \phi\left(\boldsymbol{x}(t_f)\right) + \int_{t_0}^{t_f} L\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) dt \tag{6.11}$$

subject to the dynamics $\dot{\boldsymbol{x}} = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)$ is defined.

$f(\boldsymbol{x}(t), \boldsymbol{u}(t))$ follows from the equations above and J is written as

$$J = \frac{1}{2} P \left(\boldsymbol{x}_f - \boldsymbol{x}_{f,des}\right)^2 + \int_{t_0}^{t_f} \frac{1}{2} \left(\boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u} + \boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x}\right) dt \tag{6.12}$$

where P, Q and R are the weights matrices of the final state, state, and control action, respectively.

Defining the Hamiltonian as

$$H = L\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right) + \boldsymbol{\lambda}^T \cdot \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right). \tag{6.13}$$

the optimality conditions can be obtained [15].

The first variation of the cost ΔJ^* must be zero in order for the trajectory to be optimal. This leads to the Euler-Lagrange equations for optimality

1.
$$\boldsymbol{\lambda}(t_f) = \left(\frac{\partial \phi}{\partial \boldsymbol{x}}\right)^T \Bigg|_{t_f} \tag{6.14}$$

2.
$$\dot{\boldsymbol{\lambda}}(t) = -\left(\frac{\partial L}{\partial \boldsymbol{x}} + \boldsymbol{\lambda}^T \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)^T = -(\boldsymbol{L}_x + \boldsymbol{\lambda}^T \cdot \boldsymbol{F}) \tag{6.15}$$

3.
$$\frac{\partial H}{\partial \boldsymbol{u}} = \boldsymbol{L}_u + \boldsymbol{\lambda}^T \cdot \boldsymbol{G} = \boldsymbol{0} \tag{6.16}$$

where

$$\boldsymbol{F} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \qquad \boldsymbol{G} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}} \tag{6.17}$$

and

$$L_x = \frac{\partial L}{\partial \boldsymbol{x}} \qquad L_u = \frac{\partial L}{\partial \boldsymbol{u}} \tag{6.18}$$

When writing the above matrices it is necessary to express the dependence on the air density and, as a consequence, on the current altitude. The state of the system depends on drag D and lift L

$$\begin{aligned} L &= \tfrac{1}{2} \rho S C_L V_a^2 \\ D &= \tfrac{1}{2} \rho S C_D V_a^2 \end{aligned} \tag{6.19}$$

where

$$\rho = 5.43\exp\left(\frac{-0.0512 * z}{1000}\right) \tag{6.20}$$

and C_L and C_D can be obtained step-by-step depending on the instant altitude, through a lookup table based on the results presented in Chapter 4.

Computing the expressions above $\boldsymbol{F}$ is:

$$\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 0 & c(\chi)c(\gamma) & -Vc(\chi)s(\gamma) & -Vs(\chi)c(\gamma) \\ 0 & 0 & 0 & s(\chi)c(\gamma) & -Vs(\chi)s(\gamma) & Vc(\chi)c(\gamma) \\ 0 & 0 & 0 & s(\gamma) & Vc(\gamma) & 0 \\ 0 & 0 & -\frac{5.43}{2m}ke^{kz}SC_DV^2 & -\frac{1}{m}SC_D\rho V & -gc(\gamma) & 0 \\ 0 & 0 & \frac{5.43}{2m}SC_LV \\ & & c(\phi)ke^{kz} & \frac{1}{2m}SC_L\rho c(\phi) & \frac{1}{V}gs(\gamma) & 0 \\ 0 & 0 & \frac{5.43}{2mc(\gamma)}SC_LV \\ & & s(\phi)ke^{kz} & \frac{1}{2mc(\gamma)}SC_LVs(\phi) & \frac{1}{2mc(\gamma)}SC_L\rho V \\ & & & & s(\phi)t(\gamma) & 0 \end{bmatrix} \tag{6.21}$$

where, for the sake of brevity, $k = \frac{-0.0512}{1000}$, $c(\cdot) = cos(\cdot)$, $s(\cdot) = sin(\cdot)$, $t(\cdot) = tan(\cdot)$ and the $_a$ subscripts have been omitted.

Similarly,

$$\boldsymbol{G} = \begin{bmatrix} 0 & 0 & 0 & 0 & -\frac{1}{mV_a}L\sin(\phi_a) & \frac{L\cos(\phi_a)}{mV_a\cos(\gamma_a)} \end{bmatrix}^T. \tag{6.22}$$

Fig. 6.3 shows the process used for the numerical solution of the Euler-Lagrange equations.

6.2.3 Control action vector initialization

One critical aspect influencing the convergence (and convergence speed) of the algorithm is the goodness of the initial guess for the sequence of control actions. The vector $\boldsymbol{u}$, in fact, should be initialized with values that are somewhat consistent with the final solution, as to avoid divergence due to the gradient descent algorithms. To address this problem, the control action was initialized by letting the already-implemented PD controller run first to perform the initial guess.

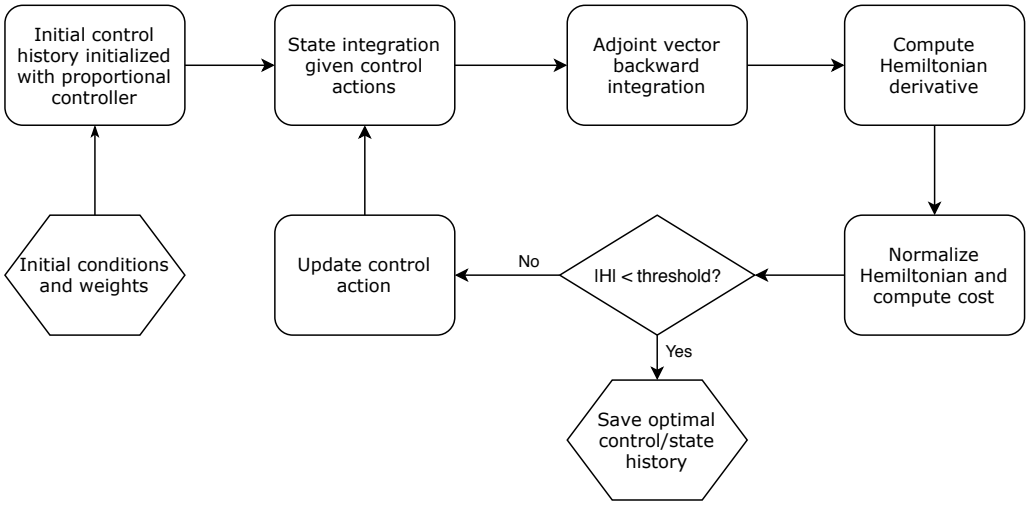


Figure 6.3: Euler-Lagrange optimization process.

6.2.4 Numerical simulations and results

The initial conditions were imposed on x, y and z as $[-1300, -1100, 1500]$ m.

The desired final conditions have been imposed to be null for x, y, z and χ.

$$P = \begin{bmatrix} 10^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10^2 \end{bmatrix} \tag{6.23}$$

weights the final position, especially z which is driven to zero through a soft constraint and χ_a which is driven to the desired final value.

$$Q = \begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{6.24}$$

constantly weights the distance from the Intended Point of Impact (IPI), which in this case is $(0, 0)$ and contributes to moving the parafoil closer to the IPI over time. Generally speaking, Q is not necessarily needed as long as the parafoil lands correctly in the desired position. Cost-wise, in fact, it does not make any difference for the parafoil to reduce its potential energy (spiral) at the beginning of a descent or close to the landing. However, for this problem, it is required by the Terrain Relative Navigation (TRN) team that te parafoil reaches as fast as possible an area close to the Intended Point of Impact for practical optical navigation purposes. The vision algorithms, in fact, turn out to be more reliable close to the IPI since the area has more contrast areas surrounding it (mainly lakes).

Finally,

$$R = 1 \tag{6.25}$$

constantly weights the control action (banking angle).

Finally, t_f has been estimated based on an average descent velocity from the results presented in the previous chapter.

The cost function J decreases over time, as shown in Figure 6.4.

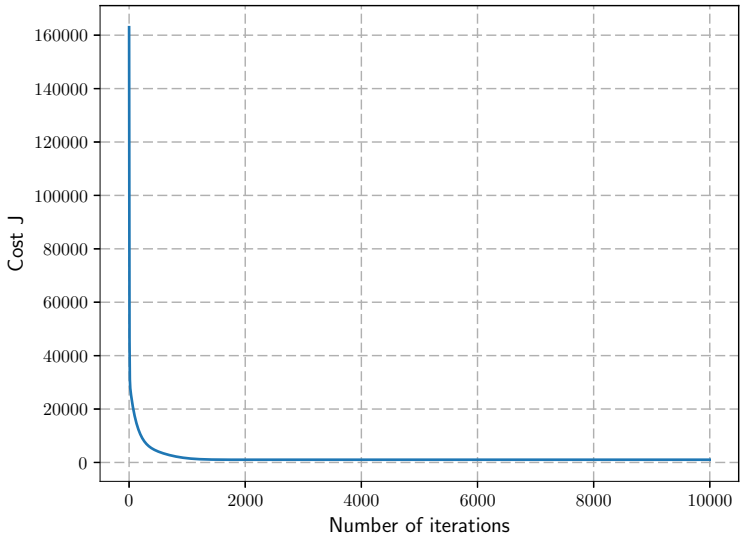


Figure 6.4: Total cost J vs number of iterations N.

Fig. 6.5 shows how x, y and z converge to zero over time:

The remaining states, along with the banking angle, can be seen in Fig. 6.6: Figs. **??** and 6.8

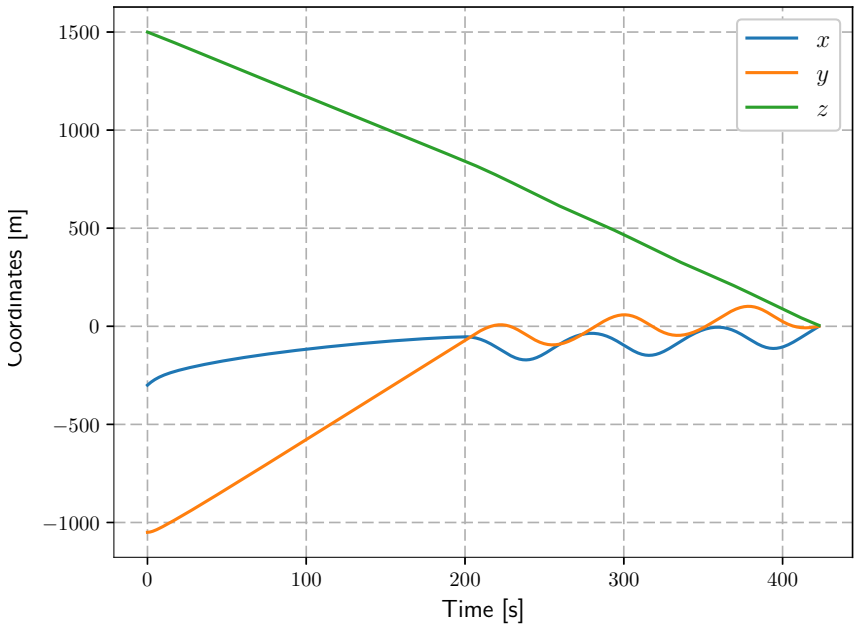


Figure 6.5: x, y and z state history.

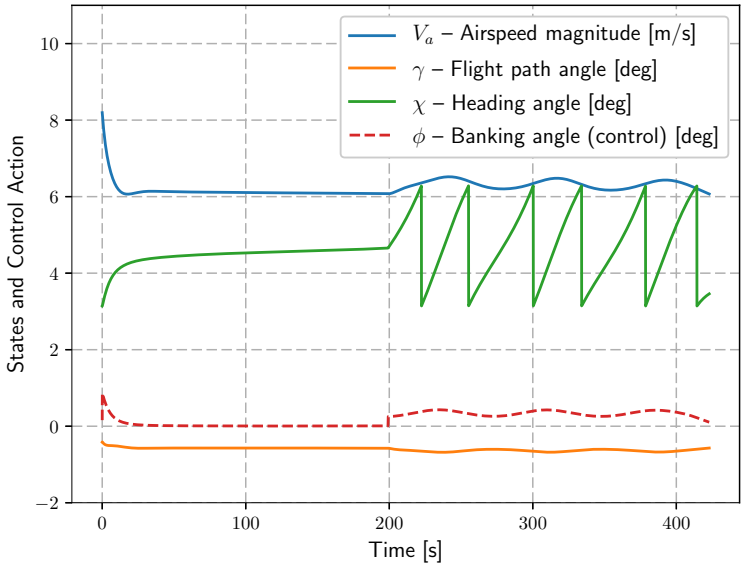


Figure 6.6: Temporal evolution of the states during the optimal descent (χ bound between π and 2π).

show the obtained trajectory.

Despite the model being a three-DOF simplification of the real system, this kind of formulation is used for formulating multiple other guidance strategies (see, for example, [16]).

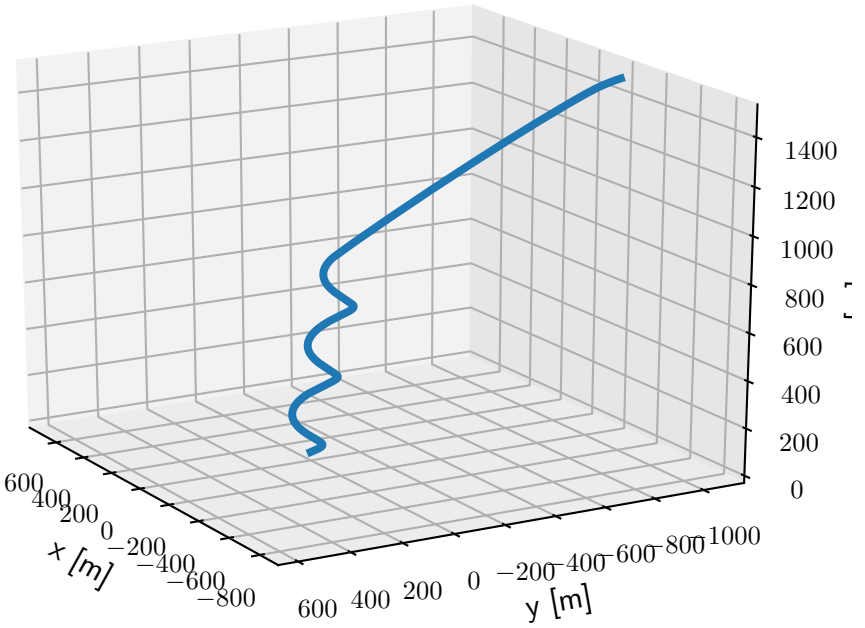
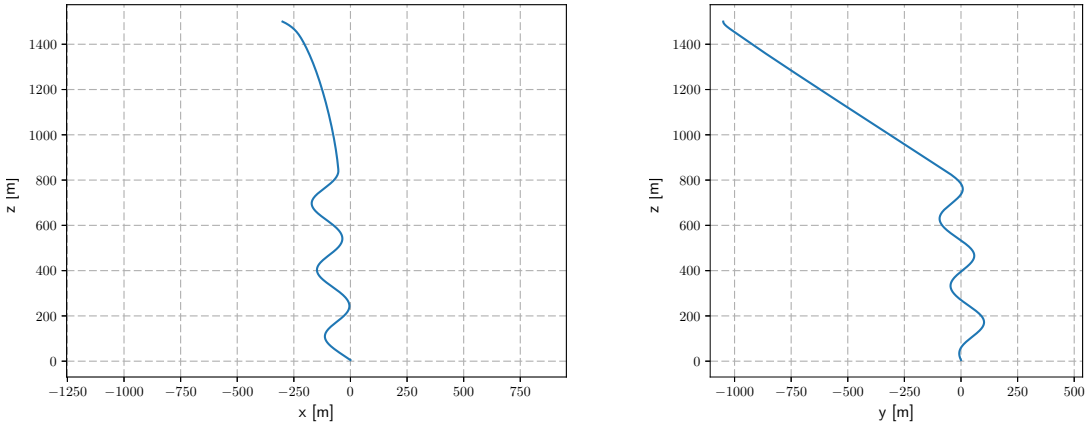


Figure 6.7: Optimal trajectory result for a 1500 m descent



(a) x-z plane.

(b) y-z plane.

Figure 6.8: Side views of the complete trajectory.

6.3 Six-DOF optimal descent trajectory

The six-DOF model was implemented next. The process for solving the trajectory is the same shown in Section **??**, with the only difference in the implementation being that matrices F and

G are obtained in symbolic form using MATLAB®. They are computed once at the beginning of the script, then the constants are substituted (constant folding) and whenever the new state is available they are quickly computed. The optimal trajectory is seen in Fig. 6.9.

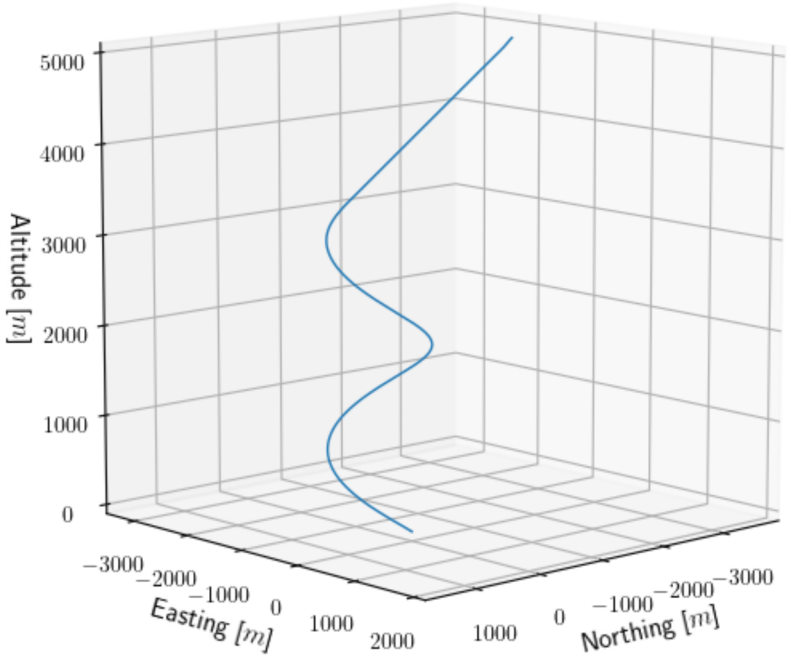


Figure 6.9: Full six-DOF optimal trajectory.

Note how using higher weights on the control action with respect to the three-DOF model brings the parafoil to make much wider turns during the spiraling phase. This also allows for more safety since reducing the banking angle is desirable whenever performing long turns.

The above trajectory has been used as a reference for future simulations of waypoint-tracking algorithms, and involves a 5 km descent. It is visible how the initial portion of the descent is a straight line (no control effort) and the subsequent part is constituted by a spiraling maneuver that leads (upwind) to the desired final position.

6.3.1 Robustness

In order to test the robustness of the algorithm to different initial states, multiple initial heading angles were imposed while the desired final heading (and whole state in general) was kept the same. As Fig. 6.10 shows, the algorithm proves to be robust to different initial heading angles.

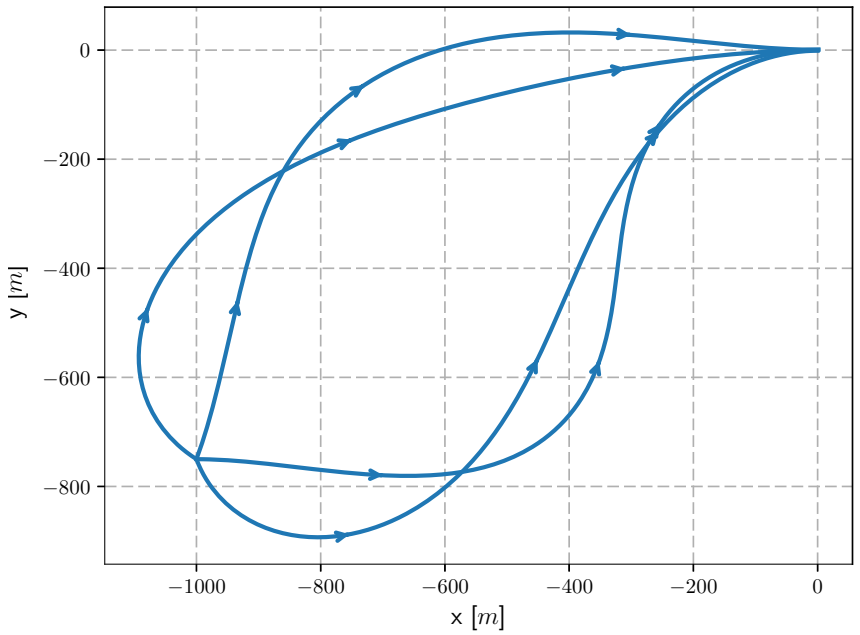


Figure 6.10: Optimal trajectories for different initial heading angle values.

In a similar way, the robustness to the final desired heading was tested. Fig. 6.11 shows the computed trajectories for fixed initial conditions and varying final heading angles which, again, turns out to be robust.

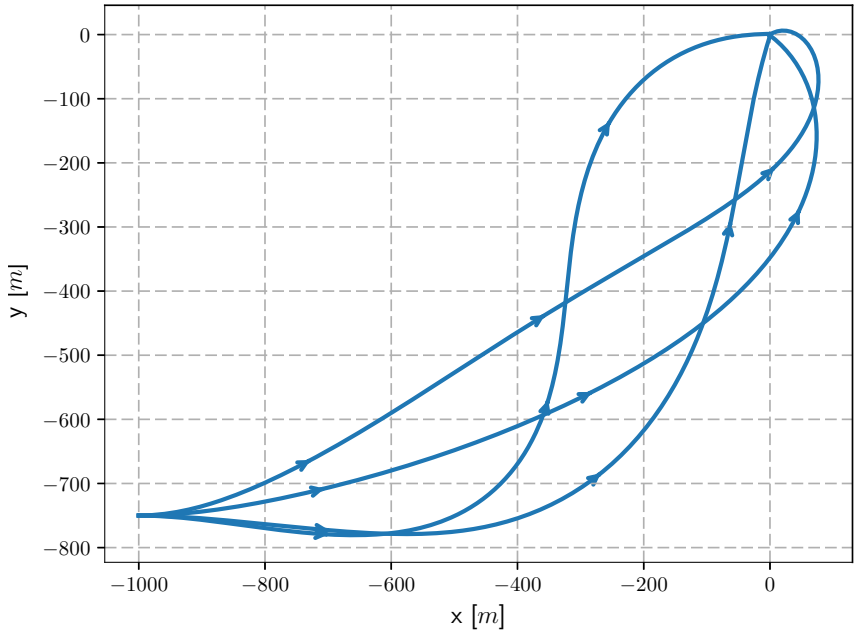


Figure 6.11: Optimal trajectories for different final heading angle values.

Chapter 7

Trajectory tracking

Once an optimal trajectory (or, perhaps, any kind of trajectory) has been computed, a trajectory-tracking controller is needed in order for the parafoil to follow the desired trajectory.

A number of studies that concern this topic have been studied in literature. Parametric approaches that assume constant properties have been used in the past for short and simple trajectories but, due to the complexity of PADS trajectories, these approaches would be ineffective.

Three different solutions are implemented to perform trajectory tracking:

1. PD controller: based on the previously implemented Proportional-Derivative controller, presented to perform waypoint navigation.
2. Linear-Quadratic Regulator: provides the optimal feedback action whenever the full state/control history is available.
3. Model Predictive Control: also used to perform waypoint navigation, requires additional computations but offers multiple advantages over the other two solutions, like the possibility of explicitly including constraints on states and control actions.

7.1 Proportional-Derivative controller

This controller can be used whenever a succession of waypoints has been generated in advance, and it is agnostic to how these waypoints have been generated. The architecture follows the principles presented in [17].

Fig. 7.1 shows a scheme of how the implemented controller works [17].

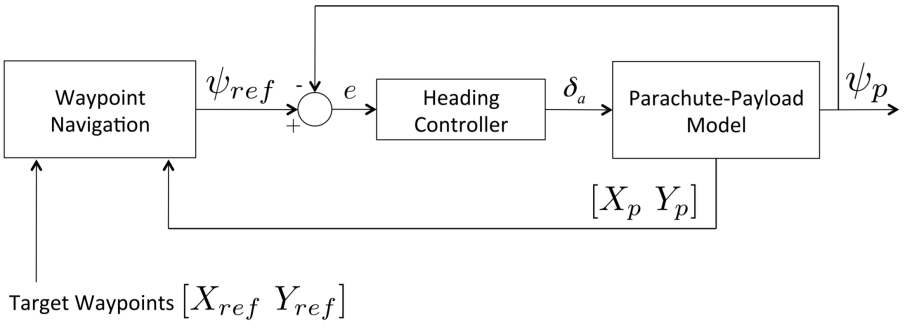


Figure 7.1: Proportional-Derivative control for waypoint tracking [17].

The assumption is that $\chi \approx \psi$ for the reasons presented before. The lateral dynamics is controlled by acting on the asymmetric flap deflection

$$\delta_a = K_p(\chi_{ref} - \chi) + K_d(\dot{\chi_{ref}} - \dot{\chi}) \approx K_p(\psi_{ref} - \psi) + K_d(\dot{\psi_{ref}} - \dot{\psi}) \tag{7.1}$$

although a second controller might be implemented to control the longitudinal dynamics as well, but is usually not done if the correct glide ratio is used for generating the waypoints. χ_{ref} is obtained by computing $\texttt{atan2}(y_{ref} - y, x_{ref} - x)$ at every time instant.

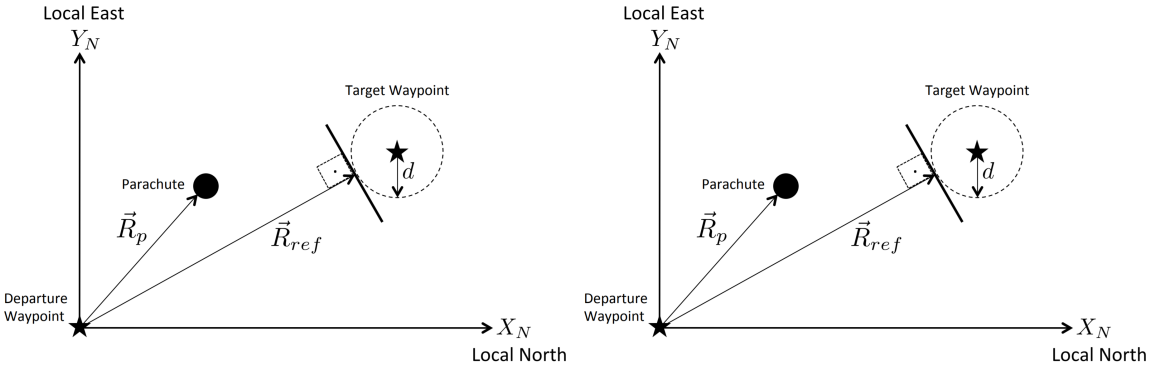
When dealing with waypoint navigation based on a PD controller, as opposed to usual signal tracking, two main problems arise: overshoot and erratic engagement.

Overshooting is caused by the fact that a PD controller does not have any information about the future, beside the information about the currently-engaged waypoint. The controller will try to reach the specified waypoint and, only after that, will it engage the next waypoint. To avoid this overshooting condition, a "waypoint clearance condition" is introduced.

Engagement of the wrong waypoint is directly related to the robustness of the controller. If for example a waypoint is missed due to high winds, the controller must move on and engage the next waypoint. Failure in doing so may lead to spiraling around a single waypoint, with subsequent loss of altitude which makes it impossible to reach the upcoming waypoints at the nominal altitude.

The waypoint clearance condition is shown in Fig. 7.2a: if the parafoil is moving from "Departure Waypoint" to "Target Waypoint", it must recognize in advance that it has entered the circle with diameter d. So, as soon as $|\vec{R_p} - \vec{R_{ref}}| < d$, the waypoint tracker moves to the next waypoint.

To recognize a missed waypoint, instead, a miss condition like the one shown in Fig. 7.2b is used. A circle located at the departure point with radius R_{dt} is defined, with R_{dt} corresponding to the distance between the two waypoints. If the PADS exits this circle, the waypoint is assumed to have been missed.



(a) Clearance condition.

(b) Miss condition.

Figure 7.2: Waypoint clearance and miss condition [17].

Waypoints don't have to be "too close" to each other to avoid continuous corrections and/or engagement errors, nor "too far", since the deviation from the computed trajectory might become too high. The clearance condition was chosen to be satisfied if the parafoil comes to at least 50 m from the intended waypoint.

Fig. 7.3 shows a full 40 km descent followed using the above approach, with the reference trajectory being the blue one and the real trajectory being the orange one.

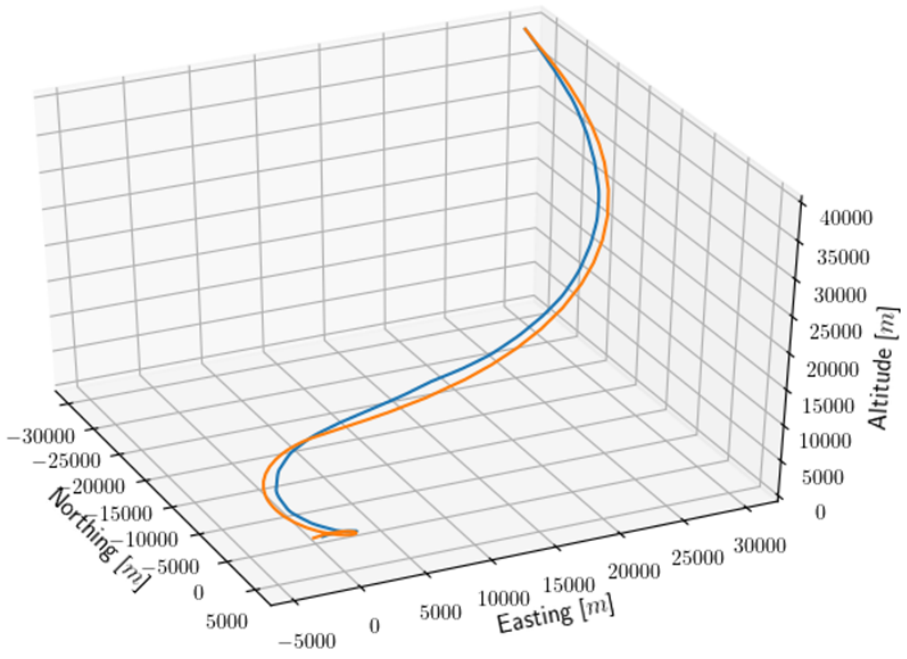


Figure 7.3: Trajectory following with a PD controller.

7.2 Model Predictive Control

A possible improvement over a PD controller is Model Predictive Control (MPC). MPC allows to exploit the model of a system to predict its future states and decide what the best control action to apply is accordingly. The steps required to solve a Model Predictive Control problem at every iteration are as follows:

1. Measure/estimate the current state $\boldsymbol{x}(t)$
2. Solve a Constrained Finite Time Optimal Control (CFTOC) problem to determine the best sequence of future control actions
3. Apply the *first* control action $\boldsymbol{u}(t)$ of the obtained sequence
4. Wait until the next measure of the state is available

The CFTOC is generally truncated to a predefined time horizon, so that only N future steps are considered. These steps are shown in Fig. 7.4.

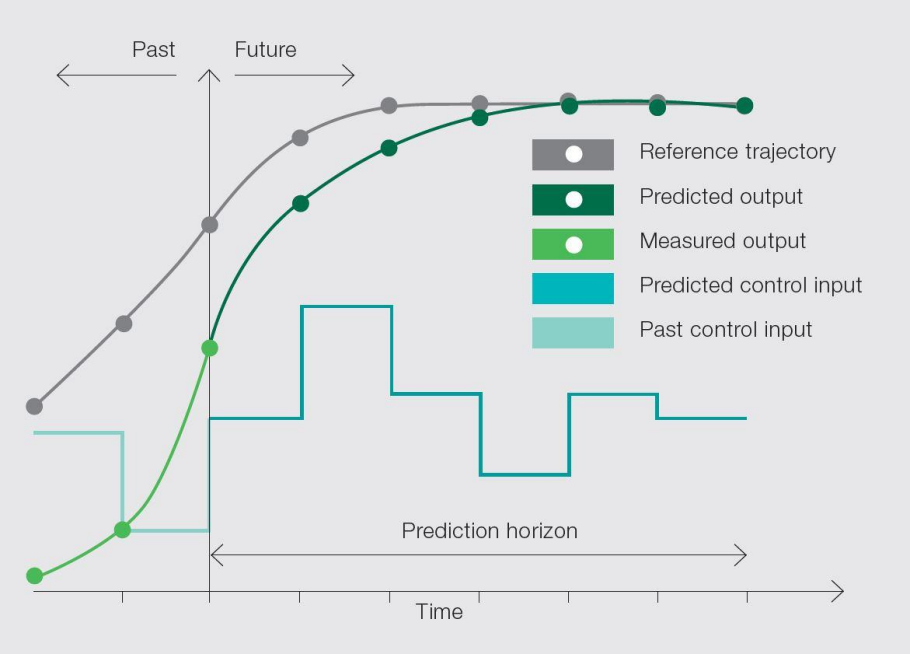


Figure 7.4: Model Predictive Control: past and future trajectories. Credit: *ABB*.

MPC offers numerous advantages over classical control techniques: among the most important ones, it allows to weight multiple cost variables to find the most appropriate combination of cost

terms, can handle multiple constraints (both on inputs and outputs) in an explicit way and allows to deal with multiple control variable at once. Furthermore, a complete nonlinear dynamic model can be handled by MPC algorithms, provided enough computational power is available to compute the solution within a sufficiently low time interval.

This advantages though come at a cost, the computational load involved in the solution MPC problems is high, and the implementation is often not trivial in the case of complex systems. Solving a nonlinear MPC (often referred to as NMPC) requires even more computational power and is seldom accomplished online, especially when considering multi-DOF multi-body systems such as a canopy and payload system. [18] analyzes in more detail advantages and disadvantages of linear and nonlinear MPCs, with particular focus on the computational requirements. For this reason, the obvious choice is to implement a linear MPC.

7.2.1 Linear Model Predictive Control

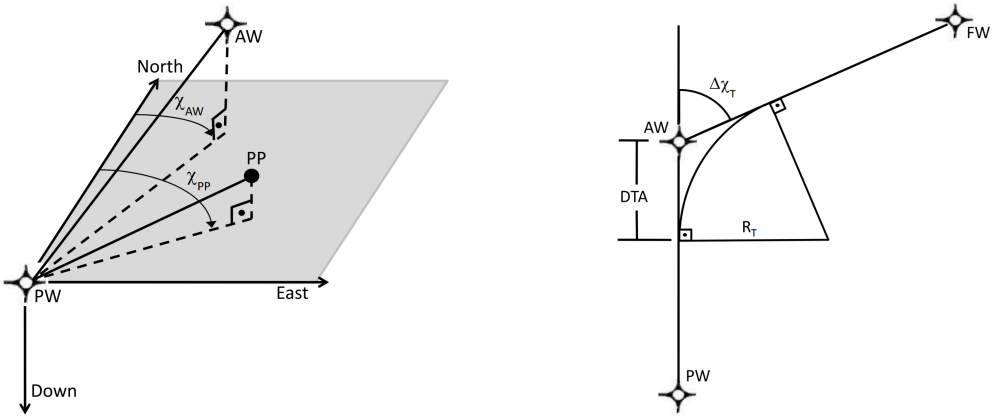
The implementation a linear model suitable to predict the future states of the PADS is not trivial since the dynamics of a parafoil is highly nonlinear due to aerodynamic forces and the rotation matrices involved when moving from a frame of reference to the other. Another reason is the complexity of the chain that connects a deflection of the control surfaces to the resulting aerodynamic forces and then, again, to the resulting dynamic changes.

Given the quasi-steady-state nature of a parafoil descent, the system is linearized at every time step before initializing the optimization procedure. With the assumption that the linearization remains valid for the whole sequence of N steps, the CFTOC is solved and the first control action is applied. This avoids integrating the dynamics of the system and allows to solve the problem faster.

A Single-Input Single-Output MPC meant for directing a parafoil is presented in [2], which also adopts a dynamic model very similar to the one presented here. A cost function is presented that weights the distance from a reference trajectory as well as the control action. However, being the presented problem unconstrained, it is possible to invert the rewrite the state-space representation of the system to explicitly extract the control action sequence $\boldsymbol{U}^*$ that minimizes the chosen cost function J. In presence of constraints the approach is not as straightforward: the problem is to be solved iteratively as shown later on in this chapter.

[19] considers the inclusion of constraints by implementing a constrained infinite horizon model

predictive control. In that case, however, a passenger aircraft undergoing engine limitations is examined, with the main objective being obtaining an appropriate tracking of waypoints that are far from each other. A single waypoint at a time is engaged by augmenting the state vector with two additional angles describing the elevation and azimuth error. A turn anticipation strategy is also implemented to avoid waypoint overshooting. Both aspects are highlighted in Fig. 7.5.



(a) Navigation system based on the next waypoint.

(b) Turn anticipation strategy.

Figure 7.5: Constrained infinite horizon MPC waypoint engagement strategy presented in [19].

The strategy presented herein, however, keeps track of multiple upcoming waypoints and does not require state augmentation.

Given a generic dynamical system described by

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t), \tag{7.2}$$

it can be rewritten in a discrete linearized form

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{A}_d\, \boldsymbol{x}_k + \boldsymbol{B}_d\, \boldsymbol{u}_k \\ \boldsymbol{y}_k = \boldsymbol{C}_d\, \boldsymbol{x}_k \end{cases} \tag{7.3}$$

where $\boldsymbol{A}_d$, $\boldsymbol{B}_d$ and $\boldsymbol{C}_d$ represent the discrete state-space matrices and are obtained from the

continuous-time matrices $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$ by applying the zero-order-hold input:

$$\begin{aligned}\boldsymbol{A}_d &= e^{\boldsymbol{A}\Delta t} \approx \boldsymbol{I} + \boldsymbol{A}\Delta t \\ \boldsymbol{B}_d &= \int_0^{\Delta t} e^{\boldsymbol{A}\tau} d\tau \boldsymbol{B} \approx \boldsymbol{B}\Delta t + \frac{1}{2}\boldsymbol{A}\boldsymbol{B}\Delta t^2 \\ \boldsymbol{C}_d &= \boldsymbol{C}\end{aligned} \tag{7.4}$$

exploiting the fact that $e^{\lambda t} = 1 + \lambda t + \frac{\lambda^2}{2!}t^2 + \cdots + \frac{\lambda^n}{n!}t^n$ when expanded using Taylor series.

Of course, since the propagated dynamics is linear, the linearization condition must hold throughout the whole predicted horizon. This condition won't hold for $N >> 1$, especially if there are quick maneuvers to be performed during the predicted horizon (e.g. a rapid switch in turning direction). In this case the MPC won't give accurate predictions and the obtained control action sequence will be misleading.

The goal at every step is to obtain the optimal control action sequence $\boldsymbol{U}^* = \left[u_0^*, u_1^*, \ldots, u_{N-1}^*\right]$ so that the quadratic cost function shown in Eq. 7.5 is minimized.

$$\begin{aligned}J(\boldsymbol{x}_0, \boldsymbol{u}) = &(\boldsymbol{x}_N - \boldsymbol{x}_{N,ref})^T \boldsymbol{P} (\boldsymbol{x}_N - \boldsymbol{x}_{N,ref}) \\ &+ \sum_{k=0}^{N-1} \left[(\boldsymbol{x}_k - \boldsymbol{x}_{k,ref})^T \boldsymbol{Q} (\boldsymbol{x}_k - \boldsymbol{x}_{k,ref}) \right. \\ &\left. + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k + (\boldsymbol{u}_k - \boldsymbol{u}_{k-1})^T \boldsymbol{R}_d (\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \right]\end{aligned} \tag{7.5}$$

where $\boldsymbol{P}$ weights the final condition, $\boldsymbol{Q}$ weights the distance from the reference at every step, $\boldsymbol{R}$ weights the control action and $\boldsymbol{R}_d$ weights the absolute value of the difference between two successive control actions.

Since the problem involves tracking a trajectory, the first quadratic term $(\boldsymbol{x}_N - \boldsymbol{x}_{N,ref})^T \boldsymbol{P} (\boldsymbol{x}_N - \boldsymbol{x}_{N,ref})$ is not included in the selected cost function, which only depends on the distance from the nominal trajectory at every time step and on the applied control action. The same result can be achieved by only including the $\boldsymbol{Q}$ matrix.

The Python convex optimizer module `cvxpy` is used to solve the optimization problem at every time step given the appropriate constraints to be satisfied. All of the following ones have to be included:

- Initial condition

$$\boldsymbol{x}(0) = \boldsymbol{x}_0 \tag{7.6}$$

- Dynamic constraint

$$\boldsymbol{x}_{k+1} = \boldsymbol{A}_d\, \boldsymbol{x}_k + \boldsymbol{B}_d\, \boldsymbol{u}_k \tag{7.7}$$

- Allowable control actions

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}_k \leq \boldsymbol{u}_{max} \qquad \forall k \tag{7.8}$$

- Allowable control actions difference

$$|\boldsymbol{u}_{k+1} - \boldsymbol{u}_k| \leq \delta \boldsymbol{u}_{max} \qquad \forall k \tag{7.9}$$

7.2.2 Online computation of the state matrices

Although the linear MPC formulation is compact, the state matrices A_{12x12} and B_{12x2} are not easy to obtain online. Nonlinearities depending upon aerodynamic forces which, in turn, depend on the control action, lead to intensive computations.

The chosen approach was to compute the matrices with MATLAB® in symbolic form using the `jacobian` function and then substitute the real-time values online. Furthermore, the part of the state vector containing the dynamics and the one that contains the kinematics can be solved in succession.

$$\boldsymbol{x} = \Big[u, v, w, p, q, r, x, y, z, \phi, \theta, \psi\Big]^T \tag{7.10}$$

in fact, as explained previously, can be divided into the dynamic and kinematic part.

$$\boldsymbol{x}_{dyn} = \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \qquad \boldsymbol{x}_{kin} = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix} \tag{7.11}$$

The first term of Eq. 7.11 is obtained solving

$$\boldsymbol{A}_{6x6}\dot{\boldsymbol{x}}_{6x1} = \boldsymbol{b}_{6x1} \tag{7.12}$$

while $\boldsymbol{x}_{kin}$ can be further divided into a translational and a rotational part so that the following relationships can be applied

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} = \boldsymbol{R}_{BN}^{T} \begin{bmatrix} u \\ v \\ w \end{bmatrix} \tag{7.13}$$

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} 1 & \sin(\phi)\dfrac{\sin(\theta)}{\cos(\theta)} & \cos(\phi)\dfrac{\sin(\theta)}{\cos(\theta)} \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\dfrac{1}{\cos(\theta)} & \cos(\phi)\dfrac{1}{\cos(\theta)} \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{7.14}$$

The computation can then be further sped up with constant folding (i.e. substituting in advance every value that remains constant during the descent).

7.2.3 Algorithm

The first step is to equalize the presence of waypoints along the track. This is done by interpolating the original waypoints with a spline (smooth path line) which, among all the possible interpolation functions, is the one with minimum curvature (i.e. minimum acceleration).

Given a sequence of n points, $n-1$ cubic polynomials can be used to interpolate them in the x-y planes, each one defined by four coefficients:

$$y(x) = a_0 + a_1\, x + a_2\, x^2 + a_3\, x^3 \tag{7.15}$$

The Python module `scipy` is used to perform the appropriate interpolations, while the `numpy` module's `linspace` command is used to match the discretization along z with the same present along the other directions. Given the spline function, the new evenly spaced waypoints are generated. The spacing between waypoints has been varied between 5 and 20 m, and 10 m has been chosen for the final implementation.

The implemented algorithm keeps track of more than one waypoint at a time, as opposed to the

usual implementations that only keep track of one at a time. Usually, it is possible to feed the optimizer with an objective function and a list of constraints to obtain $\boldsymbol{U}^*$. If more than one waypoint is involved, a different strategy must be devised. The chosen approach is the following:

1. The initial condition is set to the present state, around which the system is linearized
2. The dynamics is propagated and a list of n waypoints is stored
3. An initial guess is made for the sequence of control actions (previous optimal sequence)
4. A cost function is defined, along with the necessary constraints
5. The optimal problem is solved finding J^* so that

$$J^* = \min_{\boldsymbol{u} \in U} \left\{ \sum_{k=0}^{N} \left[(\boldsymbol{x}_k - \boldsymbol{x}_{k,ref})^T \boldsymbol{Q} (\boldsymbol{x}_k - \boldsymbol{x}_{k,ref}) + \boldsymbol{u}_k^T \boldsymbol{R} \boldsymbol{u}_k + (\boldsymbol{u}_k - \boldsymbol{u}_{k-1})^T \boldsymbol{R}_d (\boldsymbol{u}_k - \boldsymbol{u}_{k-1}) \right] \right\} \tag{7.16}$$

6. The first control action of the optimal sequence is applied, and the minimization process starts again

If the optimal solution J^* is not find within a predefined maximum allowable number of iterations, the optimizer is stopped and whatever solution it came up with is accepted as the one to be applied.

7.2.4 Predicted horizon

As mentioned above, a longer predicted horizon implies a less accurate linearization condition. Not only that, but a longer horizon also means heavier and longer computations. On the other hand, a too short horizon length is not as accurate and may overshoot sharp turns.

Fig. 7.6 shows how the convex optimizer converges to a good solution in only one iteration even starting from a null initial guess, and keeps converging toward the optimal solution with an increasing number of iterations.

Fig. 7.7 shows how to obtained solution changes as a function of the predicted horizon length. In particular, it is clear that when $N = 10$ the solution is found very quickly, although as explained above it might be prone to overshooting. With $N = 30$ the optimizer still converges with a fairly low number of iterations, although the solution takes longer to be computed. Finally, with $N = 50$ the optimizer takes much longer to converge to the correct solution and the linearization condition

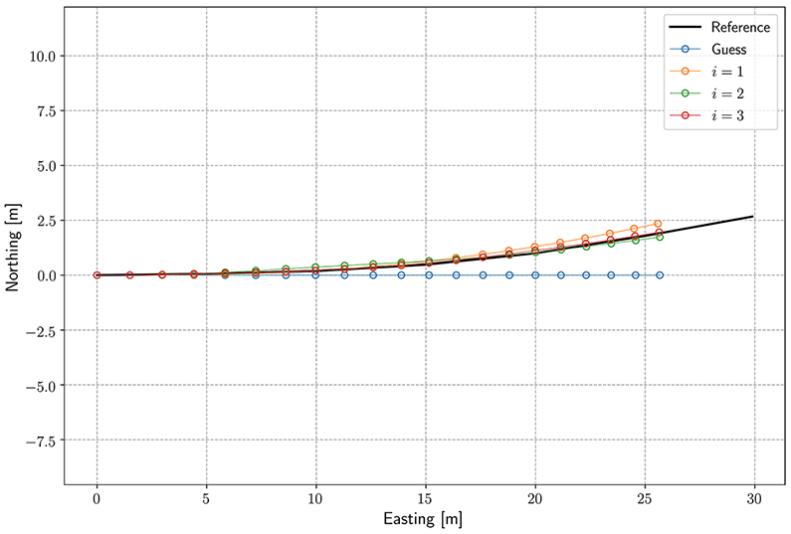
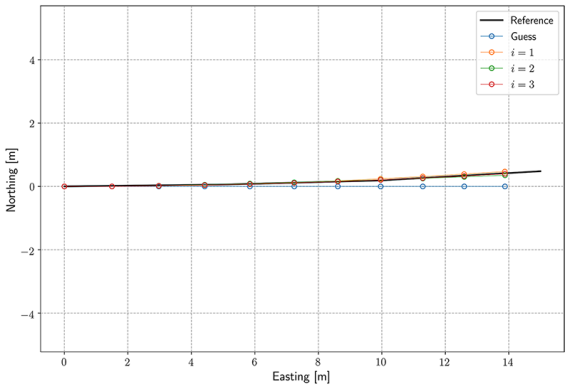


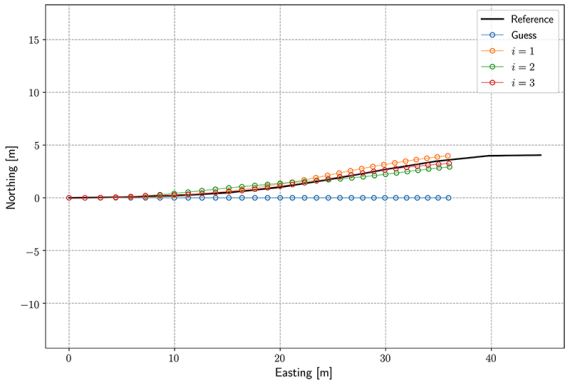
Figure 7.6: MPC solution as a function of the number of iterations with $N = 20$.

is likely not to be reliable anymore.

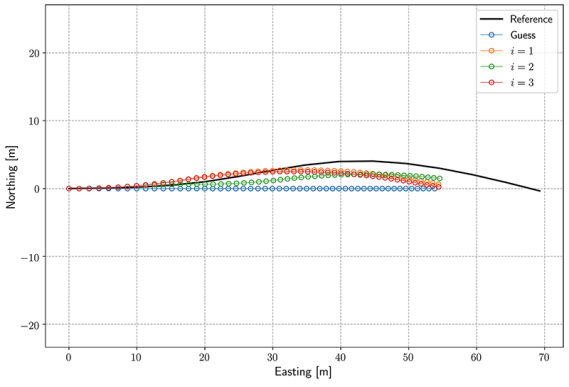
For the reasons above, after trial and error procedures and an evaluation of the convergence times, the horizon is chosen to consider twenty upcoming time steps. Fig. 7.8 shows the performance obtained when tracking a generic trajectory, while Fig. 7.9 emphasizes the sharp turn of the trajectory: the higher N is, the more accurate the tracking is. Both $N = 20$ and $N = 30$ guarantee precise tracking, so the previous choice of setting $N = 20$ is confirmed to be reliable.



(a) $N = 10$.



(b) $N = 30$.



(c) $N = 50$.

Figure 7.7: MPC as a function of the number of iterations with varying horizon lengths.

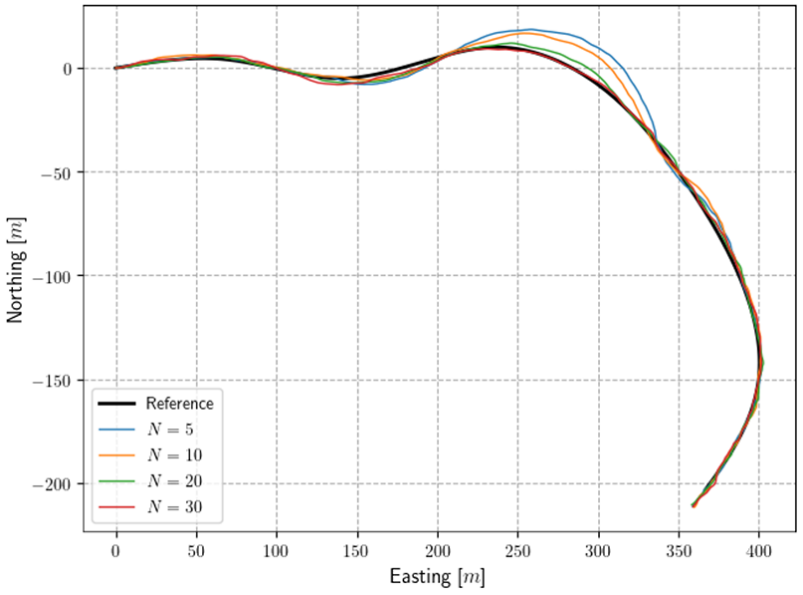


Figure 7.8: MPC tracking with gusts of wind: varying prediction horizon.

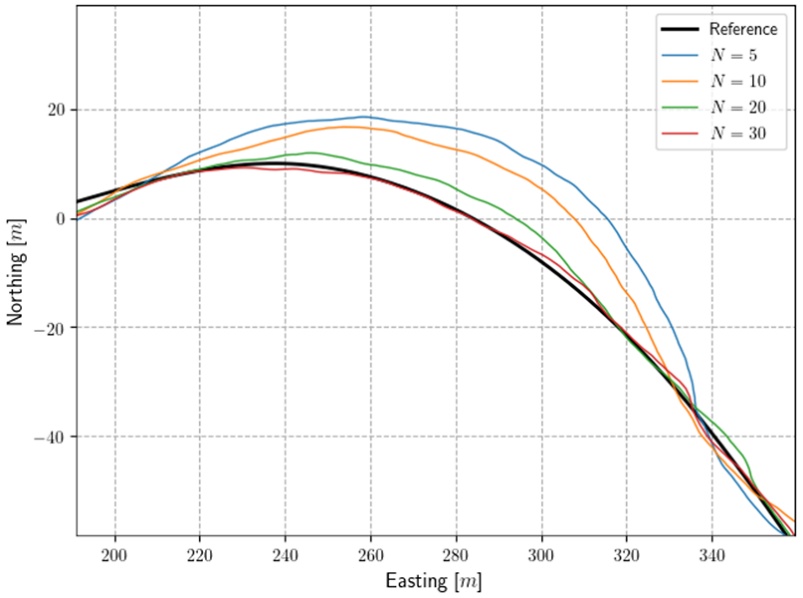


Figure 7.9: MPC tracking with gusts of wind: detail.

7.2.5 Weight for the control action

Different weights have been tested for the control action, as well as for the control action difference. Fig. 7.10, as an example, shows how the obtained solution changes when changing R during a double turn maneuver. If R is too low the actuator arrives at saturation, and if it is too high the control action won't be effective and tracking won't be guaranteed.

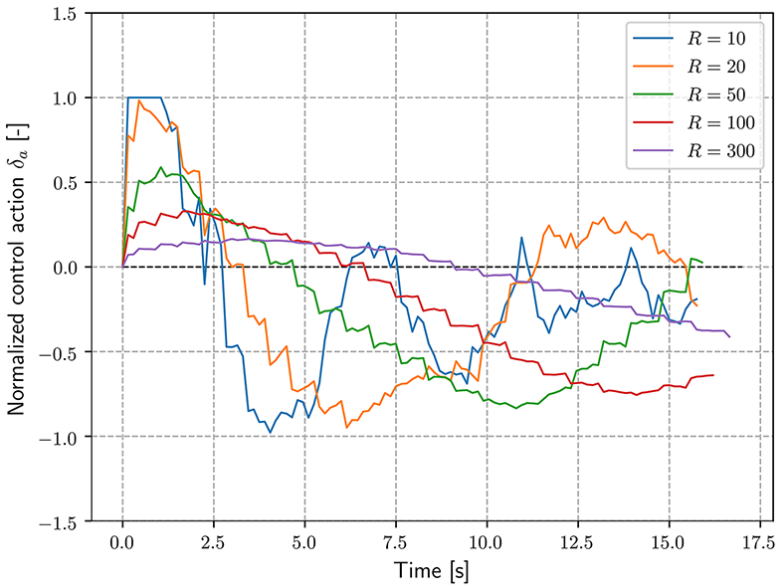


Figure 7.10: MPC control action as a function of different weights R.

R was set equal to 50, while the weight for the control action difference, R_d, was set equal to 10.

7.3 Linear-Quadratic Regulator

Provided the full state/control history of a pre-computed optimal trajectory is available, a Linear-Quadratic Regulator (LQR) can also obtained to provide the optimal feedback action needed to reject external disturbances in the neighborhood of the trajectory itself.

If the reference trajectory satisfies the three Euler-Lagrange conditions (i.e. it is optimal), it minimizes the functional shown in Eq. 7.17.

$$J = \phi\left(\boldsymbol{x}(t_f)\right) + \int_{t_0}^{t_f} L\left[\boldsymbol{x}(t), \boldsymbol{u}(t)\right] dt \tag{7.17}$$

subject to the dynamics of the system $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)$, which in turn implies that the first variation of the cost J is zero with respect to a control variation.

Given a set containing the optimal trajectory and control action history defined as

$$\{\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)\} \quad \text{for } t \in [t_0, t_f] \tag{7.18}$$

a new set $\{\boldsymbol{x}(t), \boldsymbol{u}(t)\}$ will be obtained if the system is subjected to small linear perturbations, as shown in Eq. 7.19.

$$\{\boldsymbol{x}(t), \boldsymbol{u}(t)\} = \{\boldsymbol{x}(t) + \Delta\boldsymbol{x}(t), \boldsymbol{u}(t) + \Delta\boldsymbol{u}(t)\} \quad \text{for } t \in [t_0, t_f] \tag{7.19}$$

which means that

$$\dot{\boldsymbol{x}}(t) = \dot{\boldsymbol{x}}^*(t) + \Delta\dot{\boldsymbol{x}}(t) = f\left(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)\right) + \boldsymbol{F}(t)\Delta\boldsymbol{x}(t) + \boldsymbol{G}(t)\Delta\boldsymbol{u}(t) \tag{7.20}$$

where, as usual:

$$\boldsymbol{F}(t) = \frac{\partial f\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)}{\partial \boldsymbol{x}} \qquad \boldsymbol{G}(t) = \frac{\partial f\left(\boldsymbol{x}(t), \boldsymbol{u}(t)\right)}{\partial \boldsymbol{u}} \tag{7.21}$$

If the cost function J is be expanded into a Taylor series around the reference optimal trajectory:

$$J \approx J^*\left(\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)\right) + \cancelto{0}{\Delta J\left(\Delta\boldsymbol{x}(t), \Delta\boldsymbol{u}(t)\right)} + \Delta^2 J\left(\Delta\boldsymbol{x}(t), \Delta\boldsymbol{u}(t)\right) \tag{7.22}$$

where $\Delta J\left(\Delta\boldsymbol{x}(t), \Delta\boldsymbol{u}(t)\right) = 0$ since $\{\boldsymbol{x}^*(t), \boldsymbol{u}^*(t)\}$ is optimal.

The only term to be analyzed is, then, $\Delta^2 J\left(\Delta \boldsymbol{x}(t), \Delta \boldsymbol{u}(t)\right)$, which is expanded in Eq. 7.23

$$\Delta^2 J\left(\Delta \boldsymbol{x}(t), \Delta \boldsymbol{u}(t)\right) = \frac{1}{2}\left[\Delta \boldsymbol{x}^T \frac{\partial^2 \phi(t_f)}{\partial \boldsymbol{x}^2} \Delta \boldsymbol{x} + \int_{t_0}^{t_f} \begin{bmatrix} \Delta \boldsymbol{x}^T \\ \Delta \boldsymbol{u}^T \end{bmatrix}^T \begin{bmatrix} \boldsymbol{L}_{xx} & \boldsymbol{L}_{xu} \\ \boldsymbol{L}_{ux} & \boldsymbol{L}_{uu} \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{u} \end{bmatrix} dt\right] \tag{7.23}$$

with $\boldsymbol{L}_{xu} = \boldsymbol{L}_{ux} = \boldsymbol{0}$. The remaining matrices can then be redefined as $\boldsymbol{L}_{xx} = \boldsymbol{Q}$ and $\boldsymbol{L}_{uu} = \boldsymbol{R}$. The condition we have available is and $\frac{\partial^2 \phi(t_f)}{\partial \boldsymbol{x}^2} = \boldsymbol{P}(t_f)$.

$$\implies L\left(\Delta \boldsymbol{x}(t), \Delta \boldsymbol{u}(t)\right) = \Delta \boldsymbol{x}(t)^T \boldsymbol{Q} \Delta \boldsymbol{x}(t) + \Delta \boldsymbol{u}(t)^T \boldsymbol{R} \Delta \boldsymbol{u}(t) \tag{7.24}$$

The goal is now to minimize $\Delta^2 J$ subject to the dynamic constraint $\Delta \dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t)\Delta \boldsymbol{x}(t) + \boldsymbol{G}(t)\Delta \boldsymbol{u}(t)$.

Defining the Hamiltonian as:

$$H\left(\Delta \boldsymbol{x}(t), \Delta \boldsymbol{u}(t), \Delta \boldsymbol{\lambda}(t)\right) = L\left(\Delta \boldsymbol{x}(t), \Delta \boldsymbol{u}(t)\right) + \Delta \boldsymbol{\lambda}^T(t) f\left(\Delta \boldsymbol{x}(t), \Delta \boldsymbol{u}(t)\right) \tag{7.25}$$

and applying the Euler-Lagrange conditions with the hypothesis that $\Delta \boldsymbol{\lambda} = \boldsymbol{P}(t)\Delta \boldsymbol{x}(t)$, the non-linear Differential Riccati Equation (DRE) is obtained:

$$\dot{\boldsymbol{P}}(t) = -\left(\boldsymbol{F}(t)^T \boldsymbol{P}(t) + \boldsymbol{P}(t)\boldsymbol{G}(t)\boldsymbol{R}^{-1}\boldsymbol{G}(t)^T \boldsymbol{P}(t) + \boldsymbol{Q}\right) \tag{7.26}$$

which, once solved given the boundary condition $\boldsymbol{P}(t_f) = \phi_{xx}$, provides the optimal feedback gain matrix $\boldsymbol{K}(t) = \boldsymbol{R}^{-1}\boldsymbol{G}(t)\boldsymbol{P}(t)$ through which one can obtain the optimal control action to compensate external disturbances, given that $\Delta \boldsymbol{u}^*(t) = -\boldsymbol{K}(t)\Delta \boldsymbol{x}(t)$. The regulator architecture is shown in Fig. 7.11

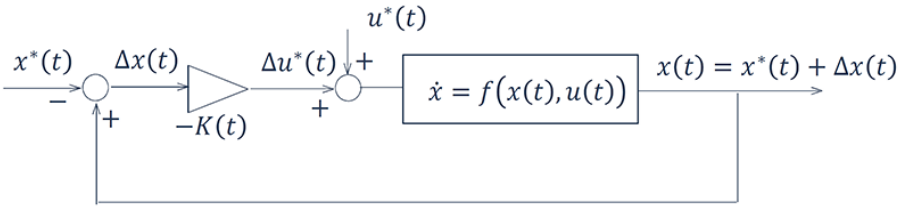


Figure 7.11: Linear Quadratic Regulator architecture.

Solving the nonlinear Differential Riccati Equation is often time-requiring but the whole solution

can be computed offline.

7.3.1 Straight descent trajectory tracking

To evaluate the noise rejection properties of the LQR, a 5 km optimal trajectory descent was simulated in presence of gusts of wind. The following plots report the behavior in presence of white noise-like gusts, with $\mu = 0$ and $\sigma = 1$ m/s.

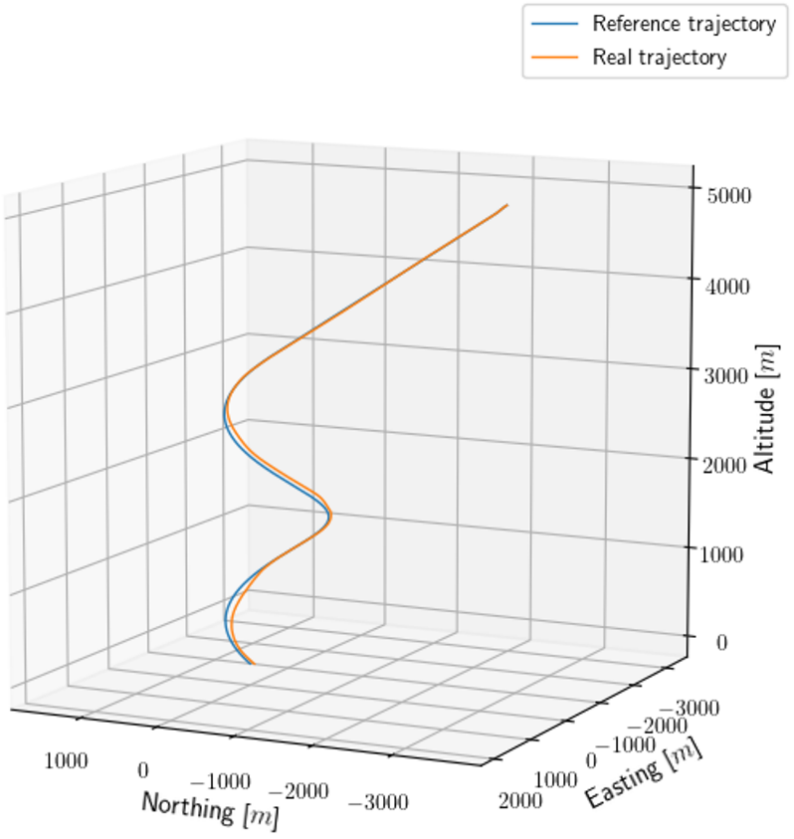


Figure 7.12: Reference *vs* real descent trajectory.

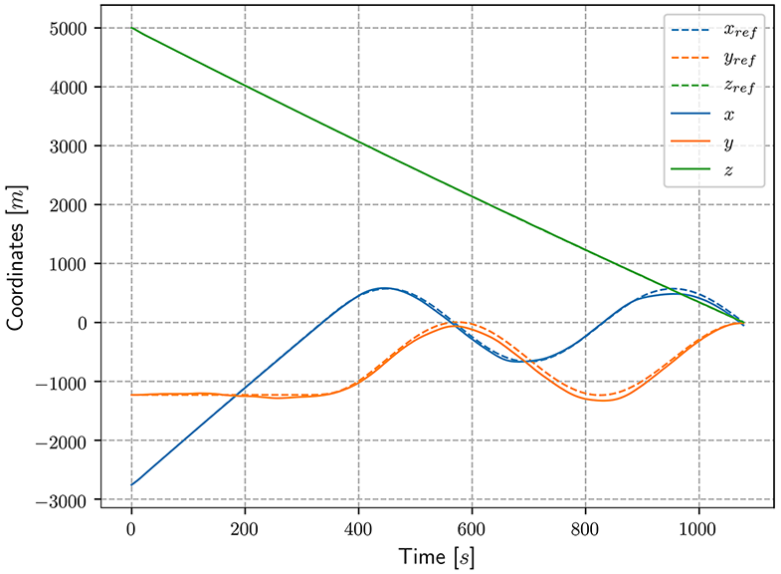
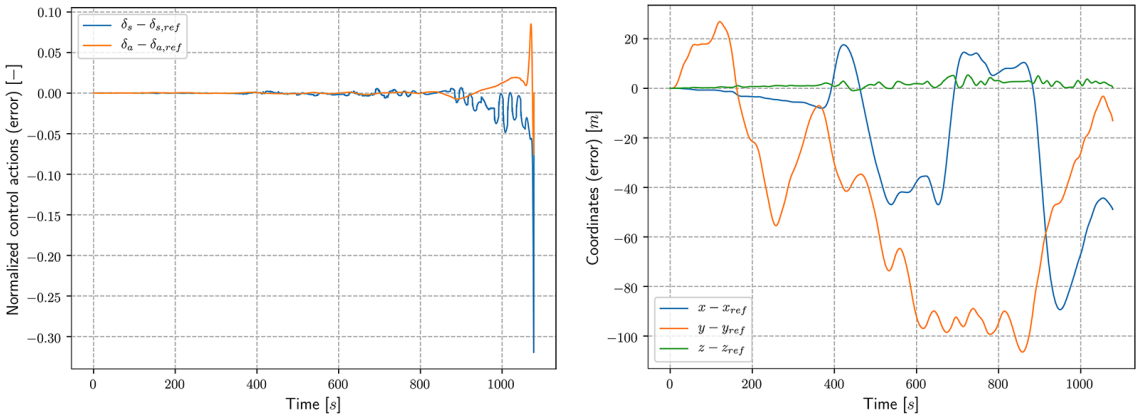


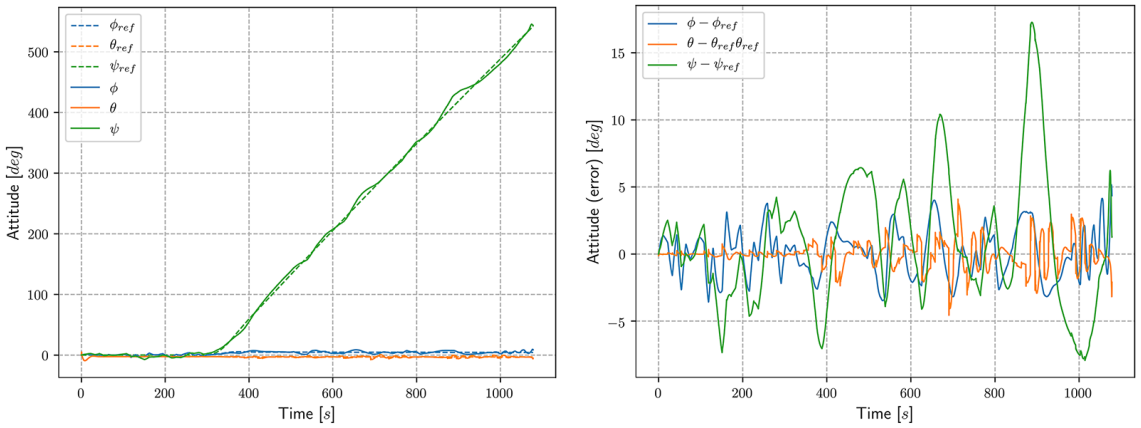
Figure 7.13: Reference *vs* real x, y and z during the descent.



(a) $\Delta\delta_s$ and $\Delta\delta_a$.

(b) Δx, Δy and Δz.

Figure 7.14: LQR – detail of control action and position.



(a) ϕ, θ and ψ.

(b) $\Delta\phi$, $\Delta\theta$ and $\Delta\psi$.

Figure 7.15: Effect of LQR on ϕ, θ and ψ.

Chapter 8

Dynamic Programming

Dynamic Programming (DP) finds its foundation in the optimality, which says that an optimal policy has the property that, whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision. More intuitively, referring to Fig. 8.1, if the optimal solution to go from $(\boldsymbol{x}_0, t_0)$ to $(\boldsymbol{x}_f, t_f)$ passes through an intermediate point $(\boldsymbol{x}_1, t_1)$, then the optimal solution starting from $(\boldsymbol{x}_1, t_1)$ goes through the same path.

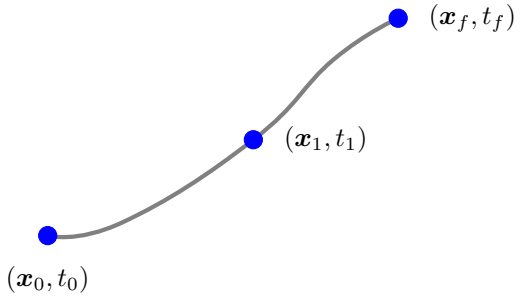


Figure 8.1: Illustration of Bellman's principle of optimality.

Dynamic programming was first proposed by Richard Bellman and leads to the solution of an iterative functional equation. Instead of exhaustively evaluating every possible solution with a brute-force approach, DP allows to reduce the computational complexity by several orders of magnitude. The solution is computed backwards in time by evaluating a so-called residual cost function at every step. Despite the elegance of its formulation, the main difficulty with applying DP

to complex problems is still the excessive computational power required, a direct consequence of the *curse of dimensionality*, even for problems of moderate complexity. Many successive publications (among which **??**, edited by Richard Bellman himself, is worth mentioning) focused on the reduction of memory requirements or computational power needed.

Here, a three-DOF reduced-order dynamic programming algorithm has been investigated to both study the feasibility of its implementation for the control during terminal guidance and verify its effectiveness. Although with a reduced model, the goal is to show that it is feasible to reduce computational time by exploiting high parallelization and successive grid refinement. The problem is first solved by applying dynamic programming to a wider grid whose density is going to dictate the precision of the final solution and then, through successive grid-refinement, a denser grid is solved around the tentative trajectory to guarantee higher accuracy and a more frequent update of the feedback control action.

Dynamic programming provides a *sufficient* condition for optimality [15]. Given the set of nonlinear differential equations that govern the dynamics of the PADS

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \phi_a(t), t) \tag{8.1}$$

where ϕ_a is the control variable (banking angle), we wish to perform an optimization by minimizing a cost function to obtain a solution that, besides the above dynamics, satisfies $\boldsymbol{x}(t_0) = \boldsymbol{x}_0$, $\boldsymbol{x}(t_f) = \boldsymbol{x}_f$, and respects the possible constraints.

Since it is necessary to propagate the state in a discrete form, an approximation must be introduced. The simplest approximation that may be used is to propagate the dynamics based on the local derivative, as already pointed out in previous chapters:

$$\boldsymbol{x}(t + \Delta t) = \boldsymbol{x}(t) + \boldsymbol{f}\left(\boldsymbol{x}(t), \phi_a(t), t\right) \Delta t \tag{8.2}$$

where Δt represents the time interval over which the control $\phi_a(t)$ is applied.

The cost function to be minimized has the following continuous form

$$J = \phi[\boldsymbol{x}(t_f), t_f] + \int_{t_0}^{t_f} L\left(\boldsymbol{x}(t), \phi_a(t), t\right) dt \tag{8.3}$$

where ϕ is not to be confused with ϕ_a, in that the former represents the cost associated to the final

condition, while the latter is the control variable.

Again, having discretized the problem, it is necessary to approximate the integration step-by-step:

$$\int_{t_k}^{t_k+\Delta t} L\left(\boldsymbol{x}(t), \phi_a(t), t\right) dt \approx L\left(\boldsymbol{x}_k, \phi_{a,k}, t_k\right) \Delta t \tag{8.4}$$

Constraints can be easily included by limiting the set of admissible states to $\boldsymbol{X}(t)$ and that of admissible control actions to $\boldsymbol{\Phi}_a(\boldsymbol{x}, t)$, which in this case are not time-dependent and will be denoted as $\boldsymbol{\Phi}_a$ and $\boldsymbol{X}$.

An iterative procedure can be applied that makes use of a functional equation to lead to the solution. A *value function* or *cost-to-go* $V(\boldsymbol{x}, t)$ is needed to determine the minimum cost to go from the present state-time pair $(\boldsymbol{x}, t)$ to the final time. Starting from Eq. 8.5

$$V(\boldsymbol{x}, t) = \min_{\phi_a(\tau) \in \boldsymbol{\Phi}_a} \left\{ \phi\left(\boldsymbol{x}(t_f), t_f\right) + \int_t^{t_f} L\left(\boldsymbol{x}(t), \phi_a(t), t\right) dt \right\} \tag{8.5}$$

Once the system is discretized, the iterative equation is expressed by Eq. 8.6

$$V(\boldsymbol{x}, t) = \min_{\phi_a \in \boldsymbol{\Phi_a}} \left\{ L\left(\boldsymbol{x}, \phi_a, t\right) \Delta t + V\left[\boldsymbol{x} + \boldsymbol{f}(\boldsymbol{x}, \phi_a, t) \Delta t, t + \Delta t\right] \right\} \tag{8.6}$$

where the costs are multiplied by Δt in order for them to be invariant to the choice of a time interval, since a final cost is present.

Eq. 8.6 implies that the minimum cost at a given $(\boldsymbol{x}, t)$ pair is found by minimizing, through the choice of the most suitable control action, the sum of the cost over the next time interval Δt and the minimum cost needed to go to t_f once in the next state $\boldsymbol{x} + \boldsymbol{f}\left(\boldsymbol{x}, \phi_a, t\right) \Delta t$. The iterative functional equation can then be solved backwards in time since $V(\boldsymbol{x}, t)$ depends on future minimum values of the cost function. The starting condition (initialization) of the algorithm is given by Eq. 8.7

$$V(\boldsymbol{x}, t_f) = \phi\left(\boldsymbol{x}, t_f\right) \tag{8.7}$$

The set of admissible states and control actions is quantized in a finite number of elements. That is, admissible states are x_i $(i = 1, 2, \ldots, n)$ and admissible control actions are u_i $(i = 1, 2, \ldots, m)$. In a similar manner, the infinite number of time instants such that $t_0 \leq t \geq t_f$ is defined by $t_k = t_0 + k\Delta t$ $(k = 0, 1, \ldots, K)$ so that $K\Delta t = t_f - t_0$.

Eqs. 8.6 and 8.7 then become

$$V(\boldsymbol{x},k)=\min_{\phi_a\in\boldsymbol{\Phi}_a}\left\{L\left(\boldsymbol{x},\phi_a,k\right)\Delta t+V\left[\boldsymbol{x}+\boldsymbol{f}(\boldsymbol{x},\phi_a,k)\Delta t,k+1\right]\right\} \tag{8.8}$$

and

$$V(\boldsymbol{x},K)=\phi[\boldsymbol{x},t_f] \tag{8.9}$$

It can be shown [20] that, because $V(\boldsymbol{x},k)$ is determined by exhaustively searching over all possible (quantized) values of $\boldsymbol{u}$ at each $(\boldsymbol{x},k)$, an absolute minimum is always found.

8.1 Dynamic Programming for autonomous PADS

Although a few applications of dynamic programming to parafoils are found in literature (see [3] and [21]), they rely on some simplificative assumptions in order to reduce the impact of the *curse of dimensionality*. Some examples limit the space assuming that the PADS will follow a trajectory that lies on a plane, while others rely on the concept of feasible arcs, which restrict the set of possible "landing" points onto the next layer to an arc. This, however, does not necessarily take into account the possibly strong wind gusts that may be present on Titan. Most approaches also simplify the problem by assuming a constant descent speed $\dot{z}$. In fact, if the two equations describing the evolution of x and y are divided by the one describing the evolution of z, it is possible to exclude the time variable from the model and recover it after the computation is completed:

$$x_h\prime=\frac{dx}{dh}=-\frac{dx}{dz}\,,\ y_h\prime=\frac{dy}{dh}=-\frac{dy}{dz}\implies\begin{bmatrix}x_h\prime\\y_h\prime\end{bmatrix}=\frac{-V_h}{V_v+w_z}\begin{bmatrix}\cos\left(\chi_a\right)\\\sin\left(\chi_a\right)\end{bmatrix}-\frac{1}{V_v+w_z}\begin{bmatrix}w_x\\w_y\end{bmatrix} \tag{8.10}$$

In the present approach, the nonlinear equations of motion are used to propagate the dynamics, assuming a varying speed over time along z and without restricting the searchable space. The approach is computationally intensive but the very nature of dynamic programming makes it a so-called *embarassingly parallel problem*, for which massive parallel computation can lead to enormous savings in computational time. Massive parallelization is adopted in many fields and, in particular, it has been analyzed for other PADS terminal guidance techniques([22]). Furthermore, a successive grid refinement is presented to divide the problem solution in two parts:

1. First, a wider grid can solved offline (i.e. during the homing phase) before reaching the

proximity of the landing site

2. After the entry point in the hypercube is known more precisely, a nominal optimal solution can be extracted and a more accurate grid can be solved only around this trajectory

The second step also allows to obtain better disturbance-rejection by ensuring a more frequent control action update, and can be recomputed online if needed, provided sufficient computational power is available on-board. Furthermore, the extension of the smaller grid is dictated by the amount of expected disturbances, so it can be physically correlated to the problem.

As mentioned above, the implementation relies on a three-DOF representation of the parafoil. The state is then represented by $\boldsymbol{x} = [x, y, z, V_a, \gamma_a, \chi_a]^T$ so, theoretically, it is necessary to discretize the state space along six directions, plus the time discretization which leads to seven grid variables. In reality however, assuming steady-state conditions, γ_a can be considered constant and V_a only depends on the altitude z. Given the flight path angle γ it is then possible to compute the horizontal and vertical components of the velocity V_h and V_v through simple trigonometric projections.

This reduces the discretization to only $[x, y, z, \chi_a]$ and time, which can be excluded by assuming a constant control action between two sufficiently close layers. The discretization then is only to be performed in four direction, which can be envisioned as a 3D grid in which every point can be approached from multiple directions. This also implies that $k = \{0, 1, \dots, K\}$ indicates the layer number here, instead of the time interval.

Different combinations of grid discretizations have been tested and, in the end, layers along z have been placed at a distance of 20 m from each other, x and y have been discretized with an equal spacing of 50 m between successive points and, finally, χ_a has been discretized with $\pi/6$ intervals. The control action, on the other hand, remains the banking angle ϕ_a (given that the considered model only has three-DOF) and must be discretized as well (in this case assuming 5 deg increments).

8.2 Computational requirements

During the computation process, $V(\boldsymbol{x}, k)$ must be computed for every quantized state $\boldsymbol{x} \in \boldsymbol{X}$. The number of grid points is

$$N_h = \prod_{i=1}^{n} N_i = K \cdot N_x \cdot N_y \cdot N_{\chi_a} \tag{8.11}$$

where N_i is the number of values in which the i-th state variable is quantized.

Assuming a 1000 m descent with 3000 m allowable space both along x and y, and assuming the discretization presented above the total number of points in every layer is

$$N_{layer} = \left(\frac{2\pi}{\pi/6} + 1\right)\left[\text{ceil}\left(\frac{3000}{50}\right) + 1\right]^2 \tag{8.12}$$

which gives almost 50000 nodes. Accounting for every possible layer leads to more than two and a half million grid points. Although it is possible to reduce the number of total operations with a few considerations about symmetry (which will be presented later), it is clear that a Dynamic Programming approach can only be used to control the last portion of a terminal descent if the problem has to be solved during the homing phase. If solved offline, a greater workspace may be allowable depending on the available resources, keeping into account the fact that due to the *curse of dimensionality* the problem does not scale well. Nonetheless dynamic programming has the huge advantage that, for every possible position in the grid, the optimal feedback control action is available.

8.3 Algorithm

A schematization of the implemented algorithm is shown in 8.2. Note that it is necessary to perform an interpolation after every transition since, in general, the parafoil will not belong exactly to a node of the successive layer after moving from one layer to the next. Being the chosen discretization fairly dense, it was chosen to adopt a linear interpolation. If the discretization was more rough, however, higher-order interpolation techniques can be used (see [20]).

If a dynamic system is affine and written in a discrete form, it is usually convenient to compute the control action that allows to reach to optimal successive state at every point in time. Even though as presented in Section 3.6 the system dynamics can be rewritten in the affine form shown in Eq. 8.13, the control action $\boldsymbol{u} = \phi_a$ won't be obtained directly to solve the dynamic programming problem.

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}(\boldsymbol{x})\boldsymbol{u} \tag{8.13}$$

Being the parafoil underactuated, in fact, it is not allowed to reach any given combination of x, y and χ onto the next layer grid, but only a certain combination of points which, in general, will not be part of the grid. This means that an approach such as the one presented in [20] should be

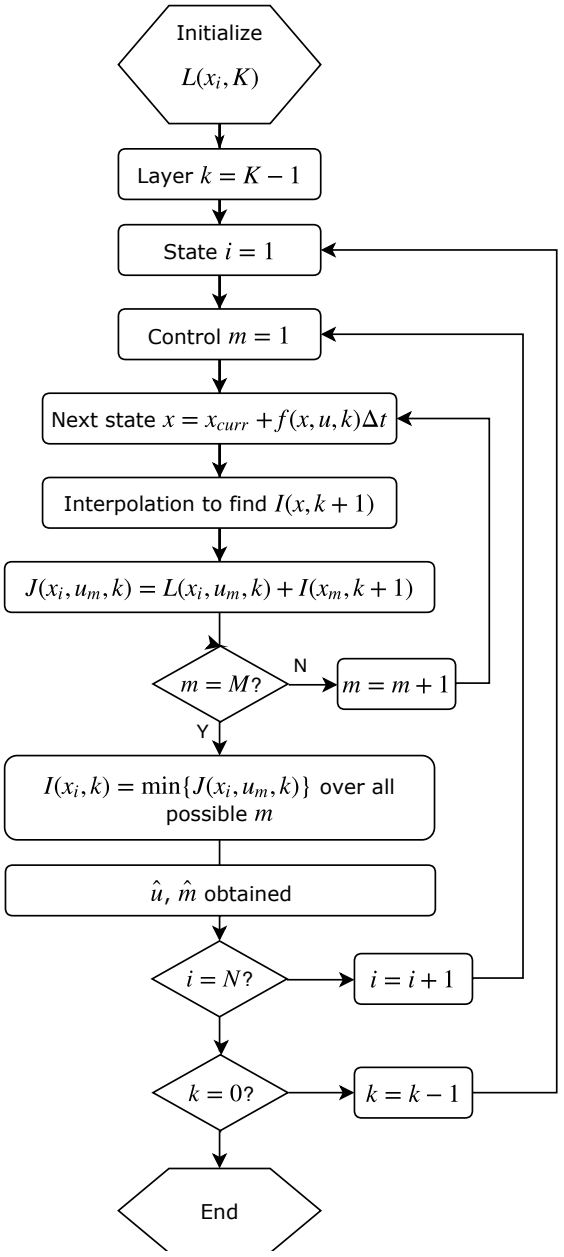


Figure 8.2: Iterative procedure for solving the dynamic programming.

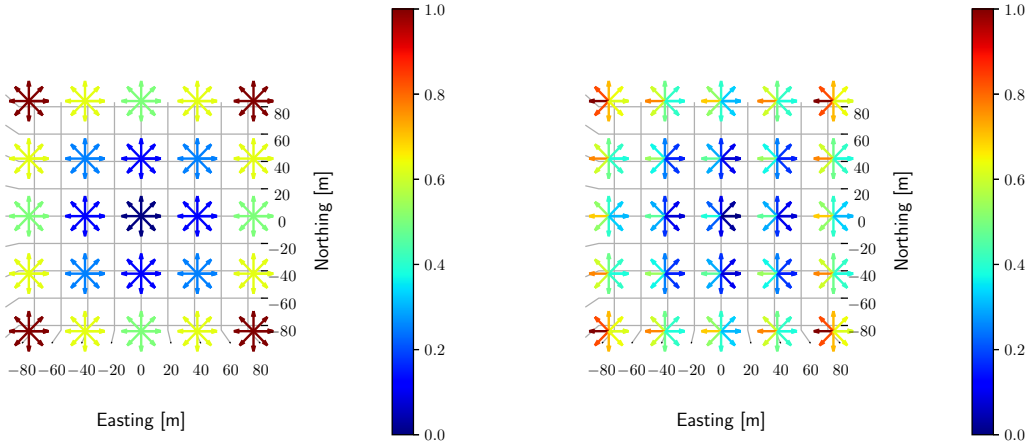
followed, where every possible control action is applied in every point of the grid, and the optimal action is the one that minimizes the sum of the value function associated to the next "landing" point and the running cost. This algorithm is still far more efficient than a brute force approach ([20]) since the usual iteration of dynamic programming can be exploited.

8.4 Choice of the final weights

Since the last layer must be solved to start with, the first choice to be made is that of a suitable cost function for the landing condition at $z = 0$. The cost function is $J_f = 0.5\boldsymbol{x}^T P\boldsymbol{x}$ as usual and, since the interest is not only to land in $(x, y) = (0, 0)$ but also to do so while moving upwind, the cost function must include the final heading angle as well. Fig. 8.3 shows how the difference is reflected in the final cost. Each point on the grid represents one of the possible landing points and, for each of them, eight possible heading angles are displayed (represented by the arrows) so that the color of an arrow represents the normalized cost associated with landing in a specific position with a given angle. Note that the χ was discretized at 45° intervals for representation purposes but, in the final script, is implemented with a 20° step. In particular, Fig. 8.3a represents a condition invariant to the final heading (see Eq. 8.14), while 8.3b takes into account the final direction of the parafoil (see Eq. 8.15).

$$J_{f(1)} = 0.5\left\{P_x\left[(x_f - x_{f,des})/x_{max}\right]^2 + P_y\left[(y_f - y_{f,des})/y_{max}\right]^2\right\} \tag{8.14}$$

$$J_{f(2)} = J_{f(1)} + P_\chi\left[(\chi_f - \chi_{f,des})/\chi_{max}\right]^2 \tag{8.15}$$



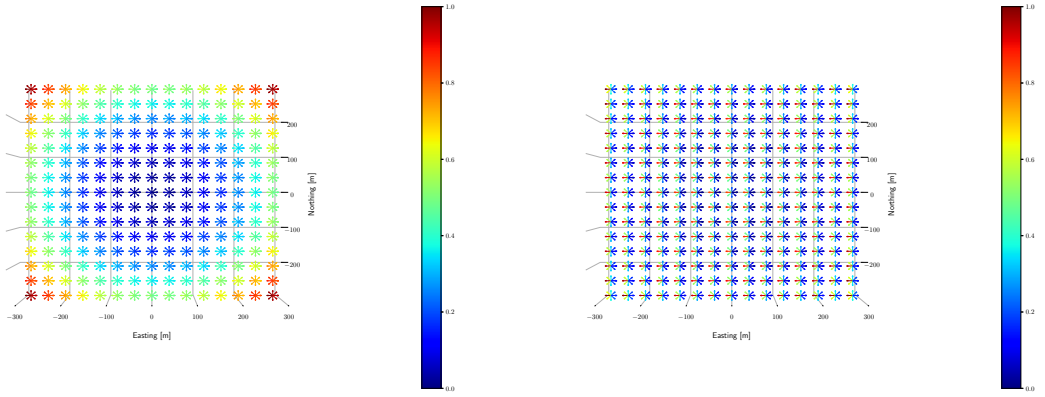
(a) Cost function invariant to χ_f.

(b) Cost function taking χ_f into account.

Figure 8.3: Different cost functions for the final layer.

Using the above cost function, however, tends to completely ignore the effect of χ_f, provided the impact point is far enough from the intended one. Off-diagonal terms come to help in this case

since they allow to weight a combination of the states, so that a wrong final heading will weight more and more when moving away from the origin, as shown in Fig. 8.4. In Fig. **??** the effect of $(x_f - x_{f,des})$ is not visible, while in Fig. **??** it is exaggerated on purpose.



(a) No off-diagonal terms in the P matrix.

(b) Cost coupling between miss distance and heading error.

Figure 8.4: Effect of P extra-diagonal terms.

Once the final condition weights have been tuned correctly, it is possible to proceed building the next layers as shown in Fig. 8.5.

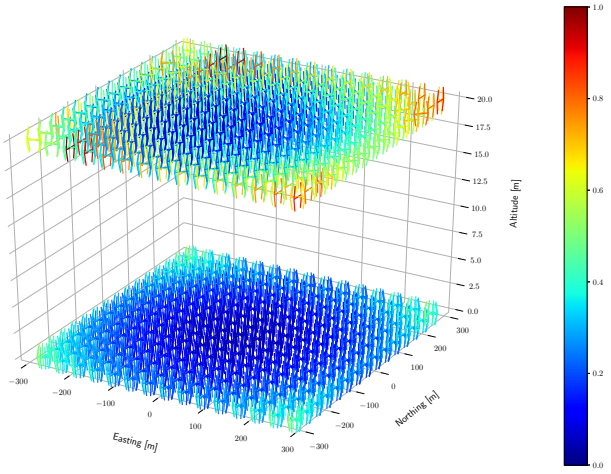


Figure 8.5: First two layers of the dynamic programming solution (workspace reduced for representation).

8.5 Parallelization

Given a generic layer, the computations to be performed in every point belonging to the layer itself only depend upon the cost-to-go associated to the next layer. The problem is then highly parallelizable and different portions of the same layer can be solve simultaneously by means of multiple threads/processes. This concept, synthetically shown in Fig. 8.6, was implemented and lead to substantial speed improvements (approximately fivefold) when the grid was split in eight different parts. Processors with a higher number of cores or that allow for a higher number of threads (including GPUs) would likely reduce this time much more.

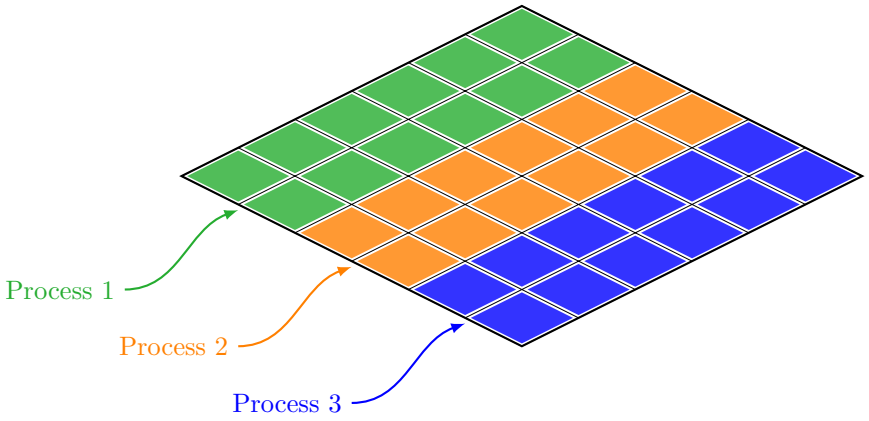


Figure 8.6: Dynamic programming grid discretization for parallel processing.

8.6 Grid refinement

In the implemented algorithm, once a "rough" trajectory is obtained, the grid is refined around it. This allows multiple advantages:

- By increasing the number of layers, a more frequent feedback action is obtained
- Increasing the discretization along x, y, χ_a and ϕ_a allows to reduce the interpolation errors and obtain a more accurate feedback
- The new grids are centered around the first trajectory, meaning that their span can be regulated according to the expected wind drifts (i.e. based on the standard deviation)
- If waypoints are chosen around the first trajectory, it is possible to compute the solution online by iteratively addressing only the upcoming descent portion

- For more complex approaches, multiple iterations could be used to bring to greater grid refinements

Figs. 8.7 and 8.8 visualize the concept presented above. Fig. 8.7 shows three generic layers belonging to the original grid (used during the first iteration). Three points P_1, P_2 and P_3 represent the intersection of the recovered trajectory with each of these layers. Fig. 8.8 shows the update refined grid: in particular, the red grids are built on the original layers, while the blue ones are added during the first iteration in an arbitrary number of intermediate layers.

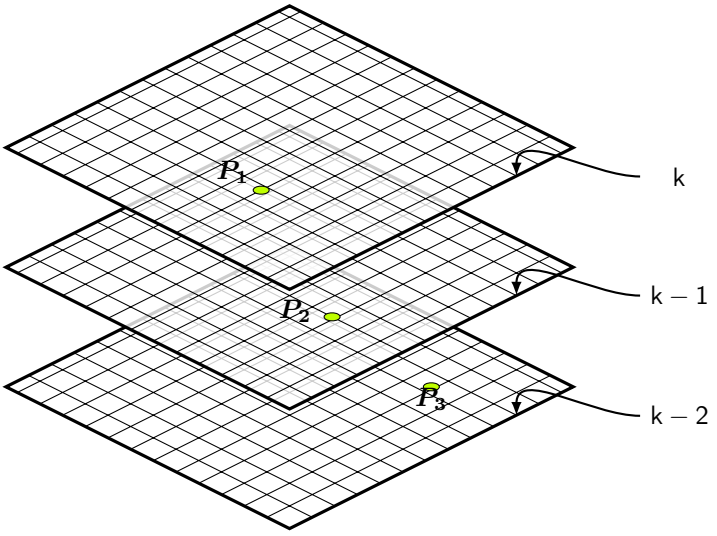


Figure 8.7: Dynamic programming grid during first iteration.

8.7 Results

A 1600 m wide, 2000 m long and 700 m high grid was solved. Discretization is as follows: 40 m spacing along Northing and Easting direction, 20 m spacing along z and 20° intervals along χ. The final solution is shown in Fig. 8.9 which represents the normalized cost-to-go associated with every point of the grid. The solution is symmetric as expected due to the symmetry of the parafoil and cost. This property allows to only solve half of the grid (namely, either the left or right part along the Northing axis) as also presented in [3], which allows to save millions of operations.

Fig. 8.10 shows how the grid gets locally refined after the initial conditions are fixed, by reducing the spacing along x, y and z to only 10 m and assuring a more accurate interpolation procedure. The obtained grid forms an envelope around the nominal trajectory, with the dimensions of the

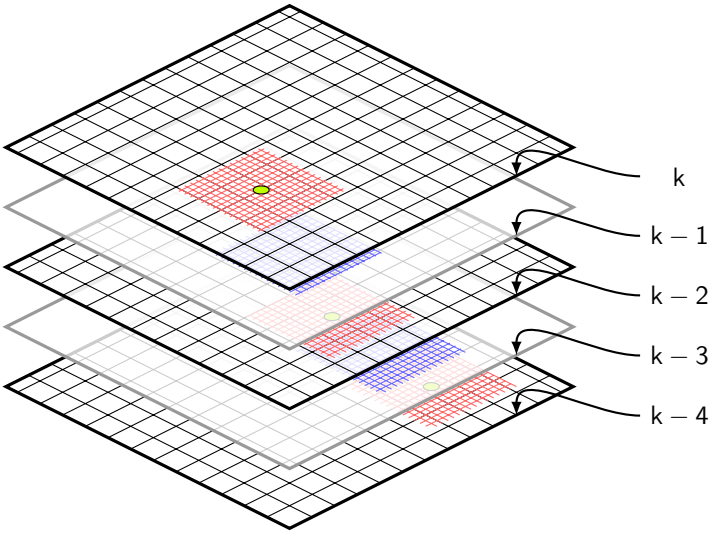


Figure 8.8: Refined dynamic programming grid.

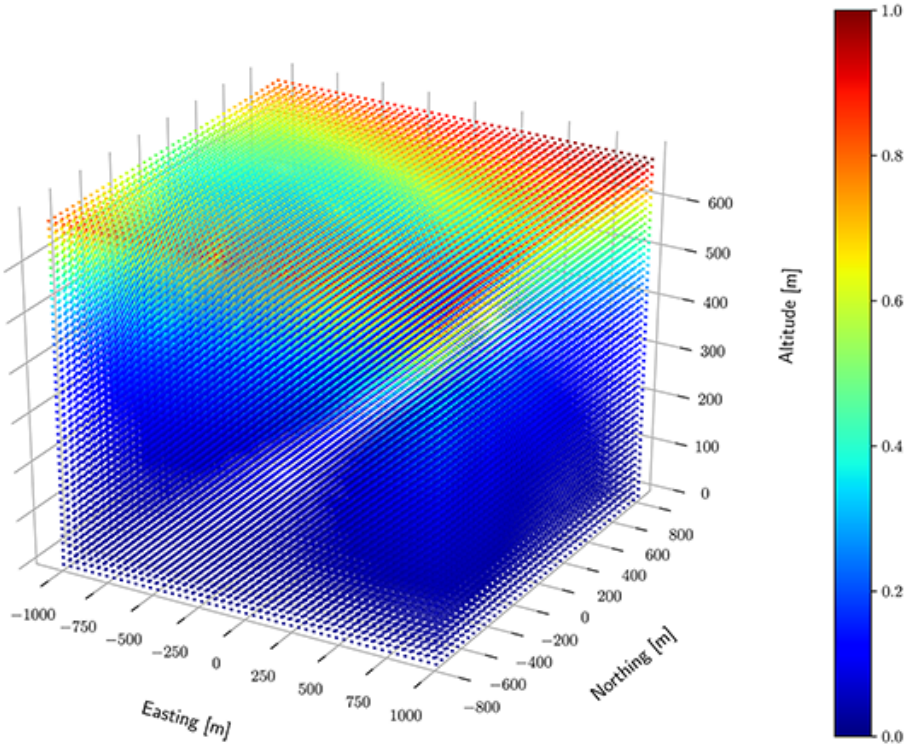


Figure 8.9: Dynamic programming – full grid generation (normalized cost of being in a specific position).

enveloped representing the maximum expected drift. Notice that, even in case the PADS was falling out of the finer grid, the original grid could still be used to recover a new trajectory.

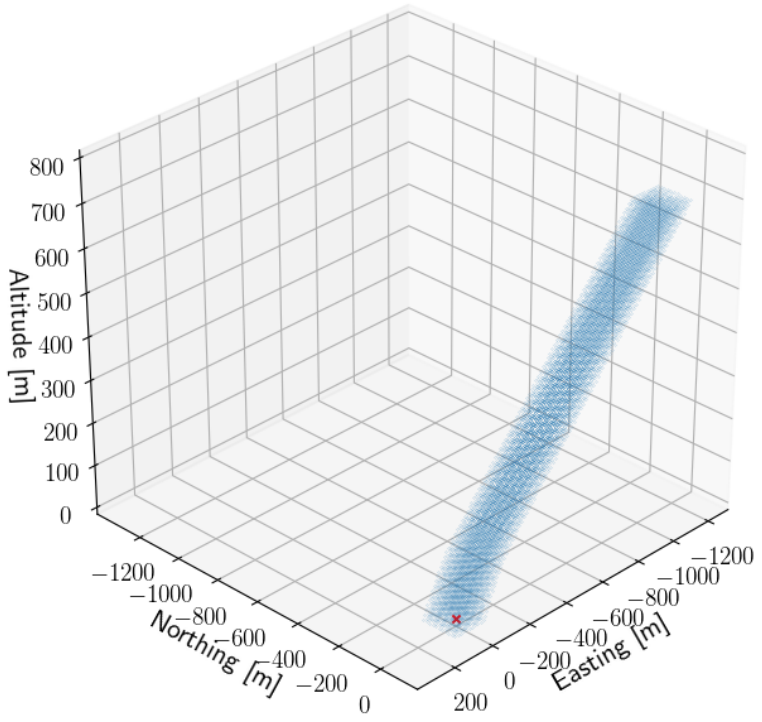


Figure 8.10: Dynamic programming – refined grid.

Finally, an example of recovered trajectory is shown in Fig. 8.11.

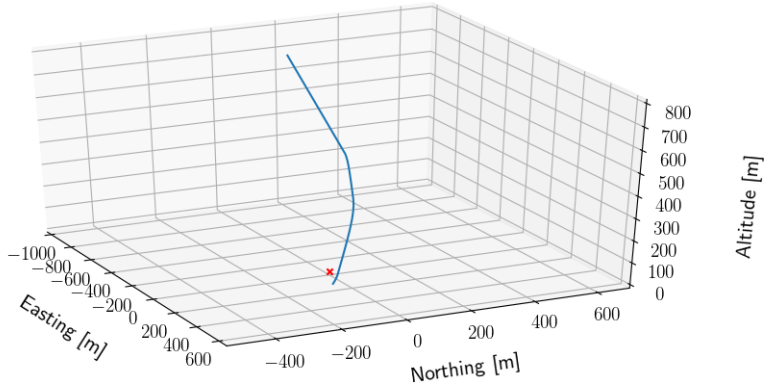


Figure 8.11: Dynamic programming – recovered trajectory.

Chapter 9

Conclusions and future work

An accurate six degrees of freedom model of a Precision Aerial Delivery System (PADS) for landing on Titan was presented, along with a reduced-order model. Its stability and controllability properties were investigated and the full system was implemented using DSENDSEdu, which allowed to simulate the real behavior with high fidelity and a lower-than-usual computational cost.

The overall approach of accurately landing on Titan using a parafoil is promising and certainly suitable for similar missions. The gliding range, provided a release altitude of 40 km, allows to reach areas about 80 km away from the deployment zone with good accuracy. The importance of landing upwind while performing a flare maneuver to reduce the ground speed was also highlighted: failure in doing so, in fact, could lead to impact speeds in the order of 5 m/s horizontally and 4 m/s vertically.

An initial turn-to-target maneuver to begin the homing phase was presented, followed by terminal descent-specific Guidance & Control techniques. It ways shown that, as expected, wind plays an important role on the final landing accuracy, but its effect can be compensated. Even the most basic implemented guidance architecture (namely, the heuristic T-approach) allowed to consistently hit a 450 m landing zone around the Intended Point of Impact during the simulations.

More advanced techniques were then presented, starting from the computation of optimal descent trajectories (whose cost function can be adapted as needed). While the computational requirements needed to solve the Euler-Lagrange problem are relatively high, there would be almost an hour available during the homing phase to produce a suitable bank of optimal trajectories. Nonetheless, a reduced three-DOF model was implemented as well, which leads to faster convergence.

A Linear-Quadratic Regulator (LQR) was presented to follow the obtained optimal trajectories (provided they were computed solving the six-DOF model) or perhaps any kind of trajectory for which the complete state/control history is available. The presented LQR is able to perform path-following correctly by constantly computing the neighboring-optimal feedback action.

Two other techniques were then presented for cases where only x-y-z coordinates of the waypoints are available (or perhaps it is decided not to use the LQR): a proportional controller which constantly compensates the drift effects and a Model Predictive Controller. While the proportional controller is able to follow the nominal trajectory correctly, it is prone to overshooting waypoints and compensating this effect leads to the new problem of having to heuristically decide the radii needed for the clearance and miss conditions. The MPC, while more requiring from a computational point of view, is much more robust to overshooting and allows to smoothly follow a given trajectory. It also allows to include any constraint relative to the state and/or the control actions.

A Dynamic Programming approach was also investigated for terminal descent. While its computational cost is extremely high even with a reduced three-DOF model, results were promising. A DP approach would allow to have a feedback available no matter where wind blows the system.

Nonetheless, multiple aspects of the project still need to be investigated:

- State estimation must be included in the simulation once the on-board instrumentation is known (IMUs, gyroscopes, Pitot tubes, etc.)
- Wind estimation and/or more accurate wind models should be included as well, since it plays a crucial role on the final landing accuracy
- Some examples are available (mainly for Mars, like [23]) that show how a feedback action can be promptly obtained by using deep-learning based techniques for research purposes. Training requires a tremendous amount of data, but results come close to achieving pinpoint accuracy for landers.
- Once a final canopy design is selected, Computational Fluid Dynamics (CFD) studies should be performed on the inflated canopy to compute aerodynamic coefficients more precisely, along with the stability derivatives
- The behavior of the canopy during the initial inflation should be analyzed
- Drop tests need to be performed to obtain the parameters needed to implement more accurate nine-DOF (payload attitude) or seven-DOF (payload yawing) models. The extension to this

kind of models should be fairly straightforward starting from what has been implemented in DSENDSEdu.

- The most computationally-intensive parts of the implemented code should be implemented in C/C++ to improve code speed and, t a more advanced stage, the whole code should be rewritten in a lower-level language to investigate real-time implementation.

References

[1] M. B. Quadrelli, "A novel approach to planetary precision landing using parafoils," 2005. 1

[2] N. Slegers and O. Yakimenko, ch. Optimal Control for Terminal Guidance of Autonomous Parafoils. Aerodynamic Decelerator Systems Technology Conferences, American Institute of Aeronautics and Astronautics, May 2009. 0. 2, 86

[3] O. A. Yakimenko, *Precision Aerial Delivery Systems: Modeling, Dynamics, and Control.* AIAA (American Institute of Aeronautics and Astronautics), 2015. 3, 9, 15, 17, 19, 20, 32, 33, 57, 67, 69, 70, 104, 111

[4] J. Waite, J. Bell, R. Lorenz, R. Achterberg, and F. Flasar, "A model of variability in titan's atmospheric structure," *Planetary and Space Science*, vol. 86, pp. 45 – 56, 2013. 5

[5] R. D. Lorenz and C. E. Newman, "Twilight on ligeia: Implications of communications geometry and seasonal winds for exploring titan's seas 2020–2040," *Advances in Space Research*, vol. 56, no. 1, pp. 190 – 204, 2015. 5

[6] P. H. Zipfel, *Modeling and Simulation of Aerospace Vehicle Dynamics.* AIAA (American Institute of Aeronautics and Astronautics) Education Series, 2000. 12, 38

[7] V. Nguyen, *Optimal trajectories in atmospheric flight.* Elsevier, 2012. 15, 16

[8] M. B. Quadrelli and A. B. Acikmese, "Planetary aeromaneuvering for precision landing," 16

[9] Seminar Presentation at the 14th AIAA Aerodynamic Decelerator Systems Technology Conference & Seminar, San Francisco, CA, USA, June 2-5, 1997., *Case Study 1. Computer Based Modeling and Analysis of a Parafoil-Load Vehicle.*, 1997. 17, 20, 31

[10] *The Added Mass and Inertia of Lightly Loaded Aricraft.*, 1982. 20

[11] *Apparent Mass Effects on Parafoil Dynamics.*, Proceedings of the 12th RAeS/AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, American Institute of Aeronautics and Astronautics, 1993. 20

[12] J. Rimani, "High lift systems for planetary descent and landing.," October 2018. 33, 41

[13] "Darts (dynamics algorithms for real-time simulation)." `https://dshell.jpl.nasa.gov/DARTS/`. Accessed: 2018-02-19. 35

[14] T. Jann, ch. Advanced Features for Autonomous Parafoil Guidance, Navigation and Control. Aerodynamic Decelerator Systems Technology Conferences, American Institute of Aeronautics and Astronautics, May 2005. 0. 57

[15] R. F. Stengel, *Optimal Control and Estimation.* Dover Publications, 1994. 69, 72, 102

[16] D. Carter, L. Singh, L. Wholey, M. McConley, S. Tavan, B. Bagdonovich, T. Barrows, C. Gibson, S. George, and S. Rasmussen, ch. Band-Limited Guidance and Control of Large Parafoils. Aerodynamic Decelerator Systems Technology Conferences, American Institute of Aeronautics and Astronautics, May 2009. 0. 76

[17] I. Y. Gonenc Gursoy, Anna Prach, "Design of a waypoint tracking control algorithm for parachute-payload systems," April 2013. 81, 82, 83

[18] M. Kamel, M. Burri, and R. Siegwart, "Linear vs nonlinear mpc for trajectory tracking applied to rotary wing micro aerial vehicles," 2017. 86

[19] F. de Almeida, ch. Waypoint Navigation Using Constrained Infinite Horizon Model Predictive Control. Guidance, Navigation, and Control and Co-located Conferences, American Institute of Aeronautics and Astronautics, Aug 2008. 0. 86, 87

[20] R. E. Larson, *State Increment Dynamic Programming.* American Elsevier Publishing Company, 1968. 104, 106, 107

[21] J. R. Cleminson, ch. Path Planning for Autonomously-guided Parafoils: A Dynamic Programming Approach. Aerodynamic Decelerator Systems Technology Conferences, American Institute of Aeronautics and Astronautics, Mar 2013. 0. 104

[22] J. Rogers and N. Slegers, "Robust parafoil terminal guidance using massively parallel processing," *Journal of Guidance, Control, and Dynamics*, vol. 36, pp. 1336–1345, Jun 2013. 104

[23] B. Gaudet, R. Linares, and R. Furfaro, "Deep reinforcement learning for six degree-of-freedom planetary powered descent and landing," *CoRR*, vol. abs/1810.08719, 2018. 116