University of Freiburg

Master Thesis

Periodic Optimal Control and Model Predictive Control of a Tethered Kite for Airborne Wind Energy

Author:
Jesus Lago Garcia

A thesis submitted in fulfillment of the requirements for the degree of Master of Microsystems Engineering

in the

Chair of Systems Control and Optimization
Department of Microsystems Engineering
Faculty of Engineering

July 26, 2016

Thesis period:
January 2016 - July 2016

Supervisor:
Dr. Michael Erhard
Prof. Dr. Moritz Diehl

Examineers:
Prof. Dr. Moritz Diehl
Dr. Andreas Greiner

Declaration of Authorship

I, Jesus Lago Garcia, declare that this thesis titled, "Periodic Optimal Control and Model Predictive Control of a Tethered Kite for Airborne Wind Energy" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Date: _____ Signed: _____

Acknowledgements

"I would like to thank Prof. Dr. Moritz Diehl for his support and supervision: the opportunity he gave me 18 months ago was a key factor for my academic development. When I first came to his group, I was a student with many ambitions but little knowledge; now, I realize that, looking back throughout this process, I have not only acquired valuable knowledge but I have also started the academic path that I have always wanted.

I would also like to express my sincere gratitude to my supervisor, Dr. Michael Erhard; your continuous feedback, joint work, ideas and suggestions were the most fruitful input to my research; without your enduring help this thesis would have never been possible.

I gratefully acknowledge Dr. Andreas Greiner as part of the examination committee. I am aware that the volume of this work exceeds the standards; however, your flexibility, help, advice and understanding made my work much easier.

I am also grateful to Skysails for giving me the opportunity to work and research in the field of Airborne Wind Energy and to the members of the Systems Control and Optimization Laboratory for all the support and help.

I would like to show my gratitude to my wife, my eternal source of strength; without your daily advice, understanding, and love, I would have never made it that far. I also want to give thanks to my parents, sisters, aunt and grandparents; since I can remember, you have been a continuous source of support and help, caring for me in an unconditional way; without you, none of this would have ever been possible.

Finally, I would like to thank my friend Cem; during the last three years you have been like a brother to me; in particular, without your help during the last month, writing this thesis would not have been possible. Also thank to the rest of my friends: Gizem, Pau, Ruiz, Xoa, Villar and Anton; part of my success was the joy you added to my life."

Abstract

Classical wind turbines suffer from a significant problem: while their power output scales with the square of the height, the mass does so cubically; as a result, material costs are high and the technology becomes non-competitive. Considering that the bulk of the power is generated by the outer parts of the rotor blades, airborne wind energy (AWE) tries to extract wind power by means of tethered kite or airplanes ("wings") while avoiding these high material costs. The conceptual idea is to fly these "wings" in a crosswind motion with the help of a strong cable and extract power by means of pumping cycles or small generators on board.

In this context and in collaboration with the company Skysails, this Masters thesis focuses on two research areas: computation of optimal trajectories for energy maximization by means of an optimal control problem (OCP) and design and implementation of nonlinear model predictive control (NMPC) on a real AWE system.

Using as a basis a previous research on periodic optimal trajectories, this thesis contributes to the field of optimal control and airborne wind energy with a set of four ideas: new safety conditions to augment the extracted power; the study of dynamic invariants within the periodic OCP; a proposed tethered kite model in natural coordinates and a performance comparison between the introduced model, a quaternion parameterization, and a model based on Euler angles; and the generation of different flight topologies to enhance the power efficiency.

Furthermore, in its second research domain, this thesis strengthens the field of airborne wind energy and control theory with the following three contributions: the design of a NMPC scheme on a real AWE system to track generated optimal trajectories; NMPC stability and robustness against real life perturbations such as wind gusts, delays in control inputs, parameter mismatches and realistic estimation errors; and development of the theory of warping systems, a manifold of dynamical systems for which an algorithm to perform online generation of optimal trajectories is proposed.

Kurzzusammenfassung

Klassische Windturbinen leiden unter dem entscheidenden Nachteil, dass ihre Ausgangsleistung nur quadratisch mit Ihrer Höhe steigt im Vergleich zur Gesamtmasse, die kubisch skaliert. Materialkosten werden zu groß, um diese Technologieart unter anderen Energieerzeugungsmethoden wettbewerbsfähig zu halten. Da der Hauptanteil dieser Leistung von den Spitzen der Rotorblättern erzeugt wird, bietet sich die Entwicklung von neuen Methoden an.

Die sogenannte AWE - Airborne Wind Energy (auf Deutsch Flugwindkraftenergie) zeichnet sich aus durch die Erzeugung von der erwähnten Leistung mittels am Boden gefesselter Drachen oder Segelflieger ("Flügel"), was große Materialkosten vermeidet. Das Funktionsprinzip beruht auf dem Fliegen der Flügel quer zum Wind ("crosswind"), wobei Energie mithilfe von Pumpzyklen oder kleinen Generatoren an Bord erzeugt wird.

Mit dem obengenannten Hintergrund und in Kooperation mit der Firma Skysails betrachtet diese Masterarbeit folgende zwei Teile: die Berechnung von optimalen fliegbaren Trajektorien, die Energiemaximierung mittels eines OCP - Optimal Control Problem (im Deutschen Optimalsteuerungsproblem) gewährleisten, sowie Entwurf und Umsetzung von einem NMPC - Nonlinear Model Predictive Controller (Englisch für Nichlinearer Modellprädiktiver Regler) auf einem existierenden AWE-System.

Auf der Grundlage vorher existierender Untersuchungen im Bereich OCP für optimale Trajektorien [1], wird diese Masterarbeit mit dem Ziel abgeschlossen, zu dem Anwendungshorizont der optimalen Regelung und der Flugwindkraftenergie mit vier wichtigen Ergebnissen beizutragen:

(i). Die Modifizierung von Sicherheitsbedingungen zur Steigerung der Gesamtleistung eines AWE-Systems.

(ii). Die Analyse von dynamischen Systeminvarianten innerhalb des OCP.

(iii). Die dynamische Erweiterung des Modells unter Verwendung von natürlichen Koordinaten und der daraus folgende Performancevergleich zwischen drei dynamischen Modellen.

(iv). Die Erzeugung von unterschiedlichen Topologien beim Fliegen zur Erhöhung der Energiegewinnungseffizienz.

Außerdem wird im zweiten Abschnitt dieser Arbeit ein NMPC geplant, um vorausberechneten optimalen Trajektorien zu folgen. Die daraus kommenden Erkenntnisse erweitern dann den Bereich der optimalen Regelung bzw. den der Flugwindkraftenergie mit den folgenden Kernpunkten:

(i). Das Design eines Real-time Iteration Scheme (in Echzeit-Implementierung) für ein existierendes AWE-System.

(ii). Die Verbesserung der Stabilität und Robustheit des Systems gegenüber realen Störungen wie Windstößen, Totzeiten, Regelungs-Offsets, Parameter-Missmatch und normalerweise vorhandenen Schätzfehlern der verwendeten Zustandsschätzer.

(iii). Die Entwicklung einer Theorie für "Warping"-Systeme, indem ein Algorithmus zur Onlinegenerierung von optimalen Trajektorien vorgeschlagen wird.

Acronyms

AWE	Airborne Wind Energy
DAE	Differential Algebraic Equation
DDE	Delay Differential Equation
GN	Gauss-Newton approximation
IP	Interior Point
KKT	Karush–Kuhn–Tucker
LICQ	Linear Independence Constraint Qualification
MHE	Moving Horizon Estimation
MIMO	Multiple Input Multiple Output
MOL	Method of Lines
MPC	Model Predictive Control
NLP	Nonlinear Programming (Problem)
NMPC	Nonlinear Model Predictive Control
OCP	Optimal Control Problem
ODE	Ordinary Differential Equation
PDE	Partial Differential Equation
pNLP	Parametric Nonlinear Programming (Problem)
QP	Quadratic Programming
RK4	Runge-Kutta integrator of order 4
RTI	Real Time Iteration
SISO	Single Input Single Output
SO(3)	Group representing rotations and orientations in the 3D Euclidean Space $\mathbb{R}^3$
SQP	Sequential Quadratic Programming
SWOCP	Semi-Warpable Optimal Control Problem
TP	Target Point
t-WOCP	Warpable Optimal Control Problem in a time frame t
WOCP	Warpable Optimal Control Problem

Nomenclature

General Symbols

ϕ_w	Wind direction
η_Loyd	Ratio between real and ideal power
A	Area of an airfoil
$C_\mathrm{D}, C_\mathrm{L}$	Aerodynamic drag, lift coefficients
P	Average power generated by an AWE system
P_Loyd	Maximum power generated by an ideal AWE system
R^2	Goodness of fit
v_a	Apparent airspeed
v_w	Ambient wind speed

Kite Terminology

ψ	Orientation of $\vec{e}_\mathrm{roll}$ with respect to wind direction
φ	Angle between tether projection into $\vec{e}_\mathrm{y}$-$\vec{e}_\mathrm{z}$ and $-\vec{e}_\mathrm{z}$
ϑ	Angle between $\vec{e}_\mathrm{x}$ and tether direction
δ	Steering command for the control pod (normalized)
E	Glide ratio ($=C_\mathrm{L}/C_\mathrm{D}$)
$\vec{e}_\mathrm{pitch}$	Kite pitch axis
$\vec{e}_\mathrm{roll}$	Kite roll axis
$\vec{e}_\mathrm{yaw}$	Kite yaw axis
$\vec{e}_\mathrm{x}, \vec{e}_\mathrm{y}, \vec{e}_\mathrm{z}$	x,y,z-axes of the reference frame
g_k	Parameter relating the turn rate and the steering deflection
l	Tether length
q	Quaternion vector defining the kite position (in combination with l)
q_i	Element i in the quaternion vector
R	Rotation matrix defining the kite position (in combination with l)
R_{ij}	Rotation matrix element in the row i and column j
v_winch	Reeling velocity, change in tether length
ω_pitch	Angular velocity around pitch axis
ω_roll	Angular velocity around roll axis
ω_yaw	Angular velocity around yaw axis

OCP Terminology

γ	Decay term used for invariant stabilization in periodic OCPs
Φ	System dynamics
Φ_k	Discrete system dynamics at time t_k
h	Path constraints
$\mathcal{I}(\cdot)$	Invariant in the system dynamics
N	Number of discrete time intervals
o_i	i^th topology constraint

r	Boundary constraints
T	Time horizon
t_k	k^{th} node of the time grid of a discrete OCP
$u(t)$	Control vector in a continuous time frame t
$x(t)$	State vector in a continuous time frame t
$y(t)$	Concatenation of $x(t)$ and $u(t)$
x_k	Discrete state at time t_k
u_k	Discrete control at time t_k
X	Concatenation of discrete state vectors x_k
U	Concatenation of discrete control vectors u_k
Y	Concatenation of discrete states X and controls U
Z	Basis matrix of the null space of $\nabla\mathcal{I}$

NMPC Terminology

Δt	Time step
M	Number of time intervals in the periodic optimal trajectory
N	Number of time intervals in the NMPC time horizon
Q	Weighting matrix for the states
Q_N	Weighting matrix for the end state
R	Weighting matrix for the controls
$(\cdot)_{\text{opt}}$	Subindex for the periodic optimal trajectory used as NMPC reference
$(\cdot)_{\text{track}}$	Subindex for the tracking trajectory in a single NMPC iteration
$u_{\text{opt},i}$	i^{th} control of the optimal trajectory
$x_{\text{opt},i}$	i^{th} state of the optimal trajectory
U_{opt}	Vector with the M optimal controls $u_{\text{opt},i}$
X_{opt}	Vector with the $M+1$ optimal states $x_{\text{opt},i}$
$u_{\text{track},k}$	Tracking control vector at time t_k
$x_{\text{track},k}$	Tracking state vector at time t_k
U_{track}	Vector with the N tracking controls $u_{\text{track},k}$
X_{track}	Vector with the $N+1$ tracking states $x_{\text{track},k}$,
Y_{track}	Concatenation of the tracking trajectories X_{track} and U_{track}
$\bar{x}_0$	Observed initial state

Time Warping Terminology

t	Normal time frame
τ	Warped time frame
$\dot{w}$	Warping factor
$p(t)$	Time variable parameter in a time frame t
$(\cdot)_{\text{ref}}$	Subindex used for variables in the reference time frame τ
p_{ref}	Constant parameter in a reference time frame τ
$u_{\text{ref}}(\tau)$	Control vector of a warpable system in a reference time frame τ
$x_{\text{ref}}(\tau)$	State vector of a warpable system in a reference time frame τ
$y_{\text{ref}}(\tau)$	Concatenation of the state $x_{\text{ref}}(\tau)$ and control $u_{\text{ref}}(\tau)$
$y_{\text{p}}(t)$	Feasible trajectory in a time frame t obtained by warping $y_{\text{ref}}(\tau)$
Y_{ref}	Discretization of $y_{\text{ref}}(\tau)$ used as reference in warping NMPC
u_1	Linear controls of a warpable system
u_2	Subset represented by $u \setminus u_1$
Y^*_{worst}	Optimal trajectory for the worst v_{winch}-path constraint scenario
$Y^{\text{warp}}_{v_{\text{w,k}}}$	Feasible trajectory for a wind speed $v_{\text{w,k}}$ that is obtained by warping Y^*_{worst}

Contents

Chapter 1

Introduction and Motivation

1.1 Airborne Wind Energy Foundations

As climate change continues to grow and its effects start to be present in today's societies, it is important to reconsider its causes and how to tackle its problems.

Recent studies have shown that electricity production accounts for 30% of the greenhouse emissions in the EU [1] and in the US [2]. In this framework, renewable energy sources are a sustainable and cleaner alternative to today's fossil fuel based energy production.

In particular, solar and wind energy are conceived to be the future of energy production as they seem to be capable enough to satisfy human energy needs, with wind being in general cheaper and more broadly available everywhere [3]. Nevertheless, wind power faces several issues that has to be overcome in order to become efficient enough in comparison to fossil fuel power plants.

The main concern of wind power lays within classical wind turbines: while the power scales with the square of the height, the mass does so cubically. That produces a set of material costs that make them non-competitive with respect to other technologies. In addition to that, the bulk of the power (more than 50 %) is generated by the outer 30 % of the rotor blades, while the rest of the construction (and in turn most of the material costs) is just needed to keep these wings in their fast crosswind motion [4]. Finally and due to structural limits, the turbine heights are limited to 150 [m] making them profitable only at locations with medium-high wind speed. The result of these issues is a power density per km^2 200-300 times lower than thermal plants [5].

A novel idea, which has received quite of attention during the last years, is to redesign the turbine eliminating the bulk material of the turbine (tower and the inner parts of the blades), and then model the outer parts of the rotor blades as tethered airfoils flying in a crosswind motion anchored to the ground with a strong cable or tether. This concept is known as *airborne wind energy (AWE)* and it solves the two previously mentioned problems:

(i). Since the only material needed is the airfoil and cable, the savings on material costs are tremendous and the profits much more advantageous.

(ii). AWE systems can reach much higher altitudes, regions where wind speed is stronger and more consistent, which leads in turn to more profitable power efficiency.

1.1.1 Crosswind Power

The scientific explanation behind this technology is based on the fact that the lift force on an airfoil increases with the square of the apparent airspeed at the airfoil:

$$F_{\mathrm{L}} = \frac{1}{2} C_{\mathrm{L}} \rho A {v_{\mathrm{a}}}^2$$

where v_{a} is the apparent airspeed, A the area of the airfoil, C_{L} the lift coefficient and ρ the air density.

As a result, a kite flying in crosswind direction with a velocity v_{a} five times faster than the wind speed v_{w} will produce a force on the tethered line 25 higher than a static kite. As a consequence, by maintaining the high kite velocity by means of the ambient wind flow, a huge amount of wind power can be extracted by means of the high force on the tether (Lift mode) or by using a wind turbine on this high airspeed v_{a}.

This concept is further explained in Figure 1.1 where specially the material reduction is seen in a graphical manner.

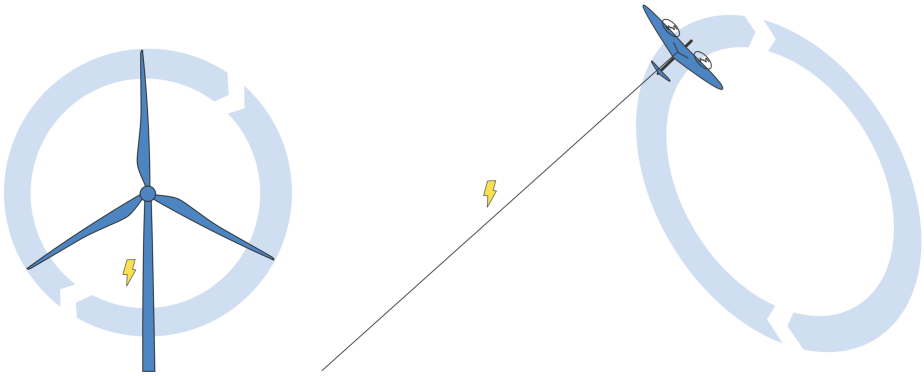


FIGURE 1.1: AWE concept: the outer part of the blades, which produces 60 % of the energy in conventional turbines but represents 30% of the material cost, are substituted by a flying airfoil to reduce costs while harvesting the same energy. [4]

It is important to remark that this idea was first developed and studied by Loyd [6] in 1980, who showed that the maximum power P that can be generated by an ideal AWE system is given by:

$$P_{\mathrm{Loyd}} = \frac{2}{27} \rho A v_{\mathrm{w}}^3 C_{\mathrm{L}} \left(\frac{C_{\mathrm{L}}}{C_{\mathrm{D}}} \right)^2 \tag{1.1}$$

with $C_{\mathrm{D}}, C_{\mathrm{L}}$ as the drag coefficient, v_{w} the wind speed, A the area of the airfoil, C_{L} the lift coefficient and ρ the air density. Nevertheless, it was not until recent years that interest and importance in AWE has risen considerably [4, 7, 8], becoming a research topic that annually gathers scientists from all over the world in the annual conference on Airborne Wind Energy and that has led to the creation of a number of companies developing diverse AWE systems, including the Google owned Makani Power [9] or the German Skysails [10] and EnerKite [11].

1.1.2 Working Principles

AWE systems have two working principles:

(i). *Drag mode*: the basic idea is to use a small wind turbine in the airfoil in order to extract the energy coming from the high apparent airspeed. This mode is also known as on-board generation due to the fact that the energy is produced at the airfoil.

(ii). *Lift mode*: the high tether force that the airfoil produces is used to pull a load at ground level, usually the load is to unroll the tether to rotate an electric generator. This method is also known as ground-based power generation because of the location of the main power elements. This mode requires a retraction phase to roll back the tether, which is usually done by moving the airfoil to an area where the lift force is small so that the energy spent in this retraction phase is much less than the energy harvested.

Since in the scope of this thesis we will work on an AWE system using the lift, we will primarily focus on this specific setup. As briefly described before, in order to harvest energy from the wind during the lift mode, the airfoil has to unroll the tether to produce energy and then to roll it back to restart the process. This periodic cycle, which is usually known as pumping cycle, is illustrated in Figure 1.2 and summarized as follows:

(i). A *power generation phase*, where the airfoil flies laying eights in crosswind motion, inducing high line forces and winching out the tether to produce energy on the ground generator.

(ii). A *transfer phase* that brings the kite to a neutral position where the orientation with respect to the wind results in a low line force.

(iii). A *return phase*, where the tether is winched in and the kite is kept at a neutral wind window position. As the tether force at the neutral position is much lower than during the power generation phase, only a small fraction of the generated energy is consumed in this retraction phase.

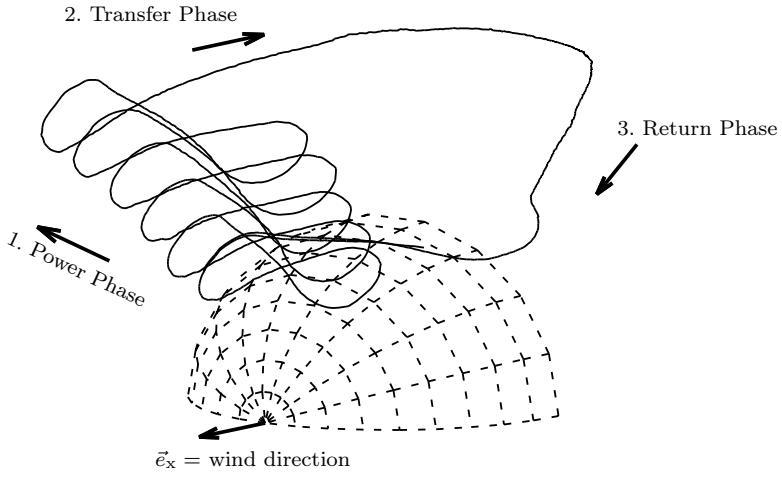


FIGURE 1.2: AWE system pumping cycle: a power phase where the airfoil flies in crosswind motion to produce energy using the high tether forces, a transfer phase to fly to a neutral wind window position of low line forces and a return phase in this neutral position to restart the cycle without consuming a large amount of energy [12].

1.1.3 Comparison with Other Sources of Energy

In order to explain the motivation of this thesis, it is very instructive to compare this novel technology with current commercialized and implemented renewable energy systems.

AWE Versus Photovoltaic Cells (PV)

As explained in [4], a modern airfoil with a drag coefficient $C_{\mathrm{D}} = 0.07$, lift coefficient $C_{\mathrm{L}} = 1$ and at a wind speed $v_{\mathrm{w}} = 13\,\mathrm{m/s}$ would be able to extract approximately $40\,\mathrm{kW}$ per m^2 of wing area. On the other hand and considering a solar irradiation of $1.3\,\mathrm{kW/m^2}$, a photovoltaic cell with a standard efficiency of 20% would produce 150 times less power per m^2. As a result, it is obvious why AWE is such a promising field of research.

AWE Versus Wind Turbines

Despite costs on material reduction have been stated as a clear AWE advantage, no specific details have been given. As reported by [4], considering an AWE wing of the size of an Airbus 380, i.e. a wing area of $845\,\mathrm{m}^2$ and $80\,\mathrm{m}$ of span that weights 30 tons, which is tethered by a modern fiber with $1\,\mathrm{GPa}$ of tensile strength, $30\,\mathrm{MW}$ could be produced by using approximately 40 tons of material. By contrast, this $30\,\mathrm{MW}$ of power is approximately the power that the biggest existing wind turbines can produce, however, these turbines require 12400 tons of material. As a result, the turbines using 300 times as much material as the AWE counterpart are an expensive and inefficient alternative.

AWE Disadvantages

Nevertheless, not everything is good news when AWE systems are regarded. Despite all the mentioned advantages, a flying airfoil that harvests energy at high altitudes is an intrinsically unstable system that requires robust automatic control algorithms that ensure safety. By contrast, photovoltaic cells and traditional wind turbines are stable systems that do not need fancy controllers to guarantee stability.

As a result, in the scope of this thesis, we will aim at overcoming this instability problem by the implementation of a robust NMPC controller that ensures stable flights under real life disturbances.

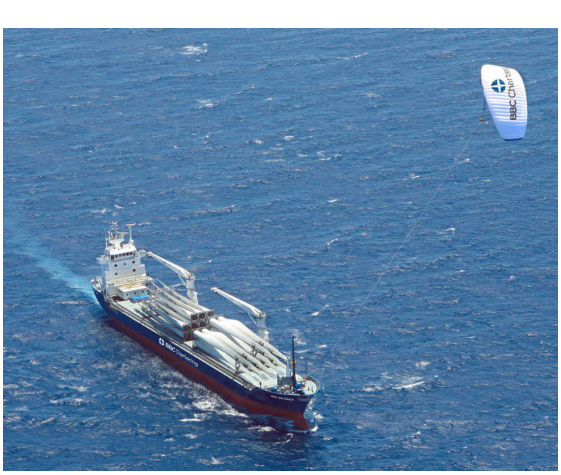
1.2 Skysails System Description

SkySails GmbH [10] is a company that was born in 2001 as a researcher and manufacturer of automated towing kite systems for vessel propulsion [13]. The idea behind the technology is, in order to save costs and reduce emissions, to use high winds as the power system for vessels. Figure 1.3a shows one of their implementation in a real vessel and in couple with Figure 1.3b they represent the main components of the SkySails system:

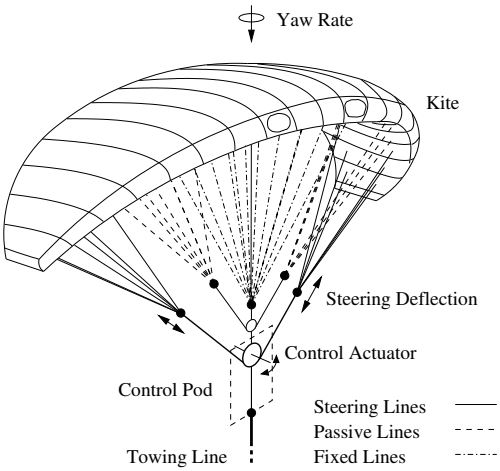
(i). The tether fiber, which is connected to the ship and transmits the wind force in the kite to the ground.

(ii). A control pod at the end of the towing line, which is the kite's main actuator and can apply kite deflections in order to obtain a curved flight. It will be referred to as the steering actuator.

(iii). The kite itself, which is the main element for energy harvesting.



(A) Real propulsion system [13].



(B) Skysails kite system schematic [13].

FIGURE 1.3: Left: BBC SkySails using a towing kite as a propulsion system. Right: Schematic of the Skysails kite system: the kite body, a towing line that transmits the wind forces to the ground and a control pod that deflects the steering lines in order to perform curved flights.

1.2.1 Power Generation

Recently, Skysails has created a company branch called SkySails Power to research and fabricate an AWE system based on their vessel propulsion kites. A first prototype was developed and tested in real life conditions using a target point controller to fly a flight topology composed by lemniscates (laying eights) [14]. Figure 1.4 shows this prototype during a real test flight.

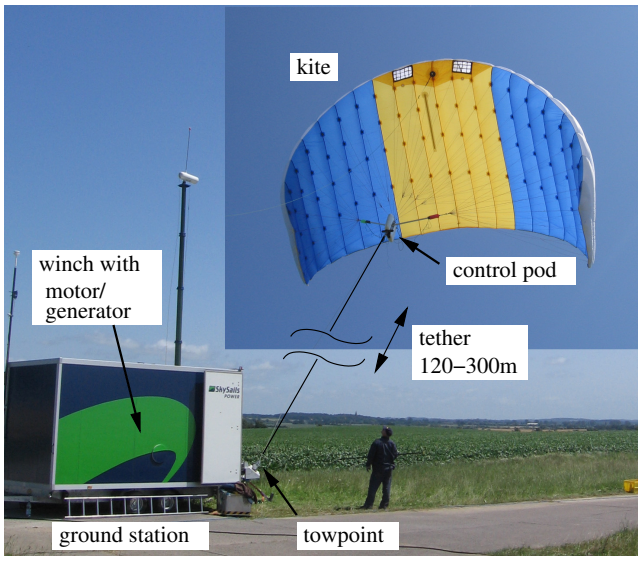


FIGURE 1.4: SkySails kite prototype for power generation in a real flight test [12].

In order to generate power, the kite flies a three stages periodic cycle as the one described in the previous section as in Figure 1.2: the three mentioned power, transfer and return phases. Using this cycle and the target point controller, their system harvests an estimated power equivalent to the 18% of the ideal maximum power P_{Loyd} given by Equation (1.1). This ratio between the real and the ideal maximum power,

$$\eta_{\text{Loyd}} = \frac{P}{P_{\text{Loyd}}}, \tag{1.2}$$

is used as the efficiency evaluation for many AWE systems, and as such, from now until the end of the thesis, Equation (1.2) will be used to evaluate the performance of the different developed models and algorithms. In this framework, [14] stated that the original SkySails system had an efficiency of 18%.

1.2.2 System Model

A simplified dynamical model was developed and validated for the original kites for vessel propulsion [13] and rescaled for the smaller size prototype. In this Section, only the model and its assumptions will be presented; for a extended derivation of the model, refer to [15].

The dynamical system is defined in the inertial reference $(\vec{e}_{\text{x}}, \vec{e}_{\text{y}}, \vec{e}_{\text{z}})$, with $\vec{e_{\text{x}}}$ always parallel to the wind direction ϕ_{w}. A fair assumption would be to think that, due to the dependence on ϕ_{w}, the reference system is not inertial. Nevertheless, ϕ_{w} is an averaged measure of the wind direction to filter wind gusts, so that in practice ϕ_{w} varies in the order of some degrees per hour and makes the reference coordinate system a good approximation of an inertial frame.

To explain the model, it has to be considered that at a constant tether length the kite performs a motion on a spherical surface, and therefore, the tether length l and two polar coordinates, ϑ and φ to determine its position. Moreover, since the tether direction is always nearly perpendicular to the kite plane ($\vec{e}_{\text{yaw}}$ parallel to the tether direction), a single angle ψ is enough to determine the orientation of the kite. As a consequence, the model state is given by $x = [\psi, \varphi, \vartheta, l]^\top$ and is illustrated in Figure 1.5.

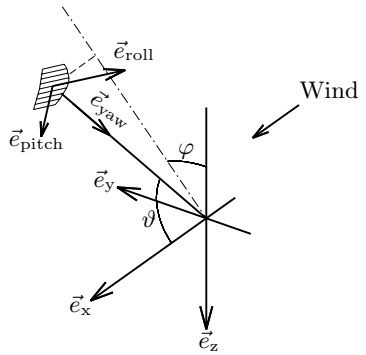


FIGURE 1.5: Kite model coordinate system [12].

Furthermore, Figure 1.5 also depicts the inertial frame $[\vec{e}_{\text{x}}, \vec{e}_{\text{y}}, \vec{e}_{\text{z}}]$ as well as the moving frame defined by the standard aircraft principal axes $[\vec{e}_{\text{yaw}}, \vec{e}_{\text{roll}}, \vec{e}_{\text{pitch}}]$. In the inertial coordinate system, the kite position is defined as:

$$\vec{r} = l \begin{bmatrix} \cos\vartheta \\ \sin\varphi \sin\vartheta \\ -\cos\varphi \sin\vartheta \end{bmatrix},$$

where ϑ is the angle between $\vec{e}_{\mathrm{x}}$ and the tether direction $-\vec{e}_{\mathrm{yaw}}$, where φ is the angle between the tether projection into the $\vec{e}_{\mathrm{y}}$-$\vec{e}_{\mathrm{z}}$ plane and the $-\vec{e}_{\mathrm{z}}$ axis, l the tether length and where the angle ψ represents the orientation of the kite longitudinal axis ($\vec{e}_{\mathrm{roll}}$) with respect to the wind direction. In particular, $\psi_0 = 0$ would represent the point where the scalar product $(\vec{e}_{\mathrm{roll}}, \vec{e}_{\mathrm{x}})$ is minimum, and then ψ would represent the angle increment/decrement of rotating the kite around the yaw axis $\vec{e}_{\mathrm{yaw}}$ starting at this reference orientation ψ_0.

For a derivation of the position using rotation matrices, the kite should begin at the position $l\vec{e}_{\mathrm{x}} = [l, 0, 0]^\top$ with $\vec{e}_{\mathrm{roll}} = -\vec{e}_{\mathrm{z}}$, and then, the following rotations should be applied:

$$R = R_{\mathrm{x}}(\varphi) R_{\mathrm{y}}(\vartheta) R_{\mathrm{x}}(-\psi),$$

i.e. a rotation $-\psi$ around $\vec{e}_{\mathrm{x}}$, followed by ϑ around $\vec{e}_{\mathrm{y}}$ ending with a rotation φ around $\vec{e}_{\mathrm{x}}$.

Model Assumptions

Before continuing with the derivation of the equations of motion, the model assumptions should be briefly explained.

- The aerodynamic forces are much larger than the system masses and therefore acceleration effects play no role.
- The tether is assumed to be massless and rigid.
- The kite is assumed to be in aerodynamic equilibrium so that the air flow is defined by the glide ratio E, which describes the ratio of the air flow between the roll and yaw axis. A strict definition of E is given by:

  $$E = \frac{C_{\mathrm{L}}}{C_{\mathrm{D}}},$$

  where C_{L} and C_{D} are the lift and drag coefficients.
- The steering of the kite is defined by a simple turn rate law.

Equations of Motion

Considering the above assumptions, the equations of motion can be described as:

$$\begin{aligned} \dot{\psi} &= g_{\mathrm{k}} v_{\mathrm{a}} \delta + \dot{\varphi} \cos\vartheta, \\ \dot{\varphi} &= -\frac{v_{\mathrm{a}}}{l \sin\vartheta} \sin\psi, \\ \dot{\vartheta} &= -\frac{v_{\mathrm{w}}}{l} \sin\vartheta + \frac{v_{\mathrm{a}}}{l} \cos\psi, \\ \dot{l} &= v_{\mathrm{winch}}, \end{aligned} \tag{1.3}$$

where $[\delta, v_{\mathrm{winch}}]^\top$ are the system control inputs, with δ representing the steering input of the control pod and v_{winch} determining directly the winching tether velocity, and where v_{a} is the

air path velocity which can also be described as:

$$v_{\mathrm{a}} = v_{\mathrm{w}} E \cos\vartheta - \dot{l} E. \tag{1.4}$$

The dynamics also have two parameters g_{k} and E, with g_{k} representing the linear dependence of the kite turn rate law and the steering deflection and E the glide ratio.

Finally, it is important to point out that because of Equation (1.4), the set of Equations (1.3) can be reformulated in several different ways representing the same dynamics.

1.2.3 System Controller

This section does not aim to give an extensive explanation of the system current controller, instead, it provides a general explanation for its working principle and its advantages and disadvantages. For a more detailed description consult [12].

The main principle of the controller is to move the kite between target points TP using the steering deflections δ, while simultaneously adjusting the winch velocity v_{winch} to reel in/out the kite and perform the desired fly phases. A TP is defined by $\mathrm{TP} = [\varphi_{\mathrm{TP}}, \vartheta_{\mathrm{TP}}]^{\top}$, and the selection of the TP and v_{winch} depends on the cycle phase (power, transfer, and return). This produces a periodic trajectory with laying eights on the power phase, and with low traction forces on the transfer and return phases. Due to the decoupling between v_{winch} and δ, the controller is designed using two independent modules:

- A first module that takes care of the trajectory phase (power, transfer and return) and selects the TP and the v_{winch}.
- The *flight* control, which manages the steering deflection to bring the kite towards the desired TP.

Figure 1.6 illustrates these two modules. Furthermore, it already outlines the fact that the flight control is composed out of the three individual controllers connected in cascade.

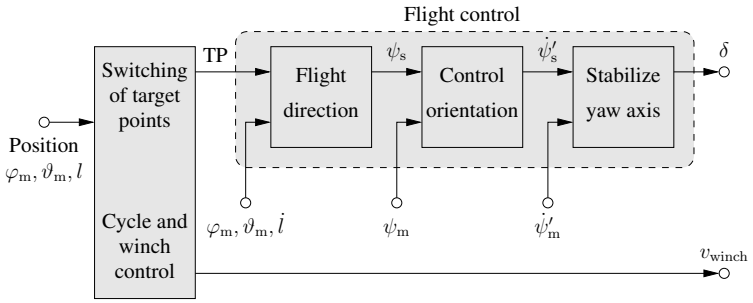


FIGURE 1.6: Classical controller overview [14].

Each of these different flight control modules has clearly defined functions:

(i). *Flight direction*: given the real measured values $[\varphi_{\mathrm{m}}, \vartheta_{\mathrm{m}}]^{\top}$ and the desired target point $[\varphi_{\mathrm{TP}}, \vartheta_{\mathrm{TP}}]^{\top}$, it computes the flight direction γ based on these four values. Then, it uses γ to compute the desired ψ_{s} angle that brings the system to the TP.

(ii). *Control orientation*: computes the error between ψ_{s} and the measured ψ_{m} and uses this value to compute the desired $\dot{\psi}_{\mathrm{s}}$ to bring ψ to ψ_{s}.

(iii). *Stabilize yaw axis*: computes the error between $\dot{\psi}_\mathrm{s}$ and $\dot{\psi}_\mathrm{m}$ and uses it to compute the required steering actuation δ.

The idea of the flight control module is represented in Figure 1.7: by setting two different TPs and switching from one to the other when getting close enough to them, a laying eight trajectory (lemniscate) is obtained; moreover, when combining the laying eights with the full controller, i.e. controlling also v_winch and the trajectory phases, the resultant trajectory resembles the desired trajectory represented in Figure 1.2.

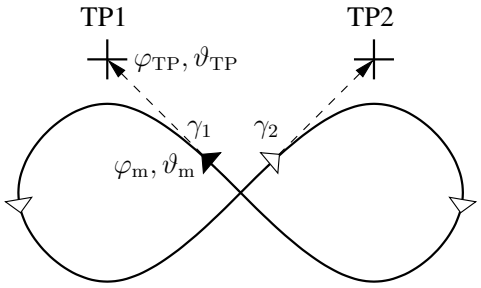


FIGURE 1.7: Pattern generation due to target points [14].

Advantages and Disadvantages

The main advantage of such a controller is quite clear: its simplicity and robustness are important features that are very hard to obtain with other approaches. In particular, due to the independence between δ and v_winch, and due to the fact that it uses two TP per laying eight, the controller is quite robust to disturbances and is able to continuously correct its trajectory to bring the kite to the different TPs. Furthermore, it has been tested under real flight conditions with successful results, achieving, as already mentioned in Section 1.2.1, an efficiency η_Loyd of 18%.

On the other hand, the simplification of having a limited number of target points and allowing the kite to fly any kind of trajectory (as long as it produces a laying eight pattern) results in trajectories, which despite producing net energy, have an efficiency that is quite below the maximum and optimal η_Loyd.

A first conclusion that can be drawn from this analysis is that a δ and v_winch decoupling affects the controller efficiency. In that framework, it is likely that better trajectories can be obtained if they are considered as a couple.

1.3 Goal

The objectives of this thesis are related with the disadvantages of the current controller and how to overcome them. The basic idea is to design and implement a new controller in order to improve the efficiency η_Loyd. In particular, we will focus on two different lines of work:

(i). We will study different methods and dynamical models to solve an *Optimal Control Problem (OCP)* to obtain a periodic trajectory that maximizes the power per cycle and in turn η_Loyd. Due to the large cycle period, solving such an OCP is a hard task which can suffer from different problems and numerical errors.

(ii). In a second focal point, we will implement a *Non-Linear Model Predictive Controller (NMPC)* to track the optimized periodic trajectories. This controller has to be robust against every sort of real life disturbance (delays, parameter and model mismatches, wind gusts, etc.) and simultaneously keep the kite flying optimal trajectories. Furthermore, the NMPC should also adapt the optimal trajectories and the dynamical model as a function of the wind speed, which can create disturbances and unstable situations if the changes are done in a discrete manner.

These two works together allow to fly optimal trajectories in a robust manner so that the extracted power is maximized. This in turn leads to a more economically attractive renewable energy product.

1.4 Mathematical Notation

In the following, a brief overview of the mathematical notation used in the thesis will be given.

Matrices and Vectors

$\mathbb{R}^n$ and $\mathbb{R}^{n\times m}$ will respectively denote the set of real vectors of dimension n and the set of matrices with n rows and m columns.

Matrices and vector symbols will not be shown visually distinct from scalars; their dimensions will follow from the context; nevertheless, square brackets will be used to present vectors and matrices elementwise.

Vectors will be considered column vectors. Concatenations of several vectors, e.g. $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^m$ yielding a vector in $\mathbb{R}^{n+m}$, will be shortened as (x, y) instead of the correct but bulkier notations $[x^\top, y^\top]^\top$.

It is important to remark that square and round brackets will also be used in a second and different context; specifically, given two real numbers $a < b$, $[a, b]$ and (a, b) will respectively denote the closed and open intervals in $\mathbb{R}$ between a and b.

The transpose of a matrix A will be defined by $A^\top$, the identity matrix of dimension n will be denoted by I_n, and diagonal matrices will be denoted by:

$$\operatorname{diag}(a_1, a_2, \ldots, a_n) = \begin{bmatrix} a_1 & & & \\ & a_2 & & \\ & & \ddots & \\ & & & a_n \end{bmatrix}$$

$A \succcurlyeq 0$ will denote that a symmetric matrix A is positive semi-definite, i.e. all its eigenvalues are larger or equal to zero; likewise, $A \succ 0$ will denote that A is positive definite.

$\|x\|$ will denote the Euclidean norm, i.e. $\|x\|^2 = x^\top x$; similarly, $\|x\|_Q$ will denote a weighted Euclidean norm with a positive definite weighting matrix $Q \in \mathbb{R}^{n\times n}$, i.e. $\|x\|_Q^2 = x^\top Q x$.

Functions

For derivatives of functions $f(x) : \mathbb{R}^n \to \mathbb{R}^m$, the Jacobian matrix will be defined by:

$$\frac{\partial f}{\partial x}(x) \in \mathbb{R}^{m\times n}.$$

For scalar functions $f : \mathbb{R}^n \to \mathbb{R}$, $\nabla f(x) \in \mathbb{R}^n$ will denote the gradient column vector. Less standard, the gradient symbol will be generalized to every function $f : \mathbb{R}^n \to \mathbb{R}^m$ to denote the transpose of the Jacobian matrix:

$$\nabla f(x) = \frac{\partial f}{\partial x}(x)^\top \in \mathbb{R}^{n \times m}.$$

Also for scalar functions $f : \mathbb{R}^n \to \mathbb{R}$, the Hessian matrix will be defined by:

$$\nabla^2 f(x) \in \mathbb{R}^{n \times n}.$$

The modulo operation to find the remainder of a division between two positive real numbers, say dividing a between b, will be defined by $a \bmod b$.

Part I

Offline Optimal Control

Chapter 2

Optimal Control in a Nutshell

Optimal control can be defined as the *optimization of dynamical systems*. As a result, it merges two important fields of research: systems theory and numerical optimization. A dynamical system can be regarded as a process evolving in time that is characterized by states x and controls u. The states define the system status and the controls modify the states evolution. OCPs use numerical optimization methods to obtain the set of controls u that optimize some objective function while respecting some given constraints. One of the most important constraints in OCPs are the system dynamics: a mathematical model ensuring that the equation of motions are fulfilled.

As explained in the previous chapter, the work performed in this thesis will require the design and solution of two types of OCPs:

(i). In order to obtain the periodic optimal trajectories, an OCP has to be solved in order to maximize the power per cycle while obeying the system dynamics and many other constraints.

(ii). The NMPC is itself a special case of an OCP: despite being a real time controller, it solves at every iteration an optimization problem in order to obtain an optimal control policy at the same time that ensures correct dynamic and safety constraints.

Therefore, due to the great importance of OCPs within this thesis, a brief overview will be given.

2.1 Continuous Optimization

Continuous optimization solves the generic class of optimization problems where the decision variables are continuous. In particular, one important class of problems that lay in this field are the so called *nonlinear programs (NLPs)*, which can be stated as:

$$\begin{aligned} \underset{w \in \mathbb{R}^n}{\text{minimize}} \quad & f(w) \\ \text{subject to} \quad & g(w) = 0, \\ & h(w) \leq 0, \end{aligned} \tag{2.1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^p$ are assumed to be at least once continuously differentiable. Before explaining how the solution of (2.1) can be obtained, a series of definitions shall be given [16].

Definition 2.1 (Active Constraint)**.** An inequality constraint $h(w) \leq 0$ is defined to be active at a point $\bar{w}$ if and only if $h(\bar{w}) = 0$.

Definition 2.2 (Active Set)**.** The set of indices $\mathcal{A}(\bar{w}) \subset \{1, \ldots, p\}$ of the active constraints at a point $\bar{w}$ is called the active set.

Definition 2.3 (Linear Independence Constraint Qualification (LICQ))**.** A general NLP defined by (2.1) it is said to satisfy the LICQ at a point $\bar{w}$ if and only if the vectors $\nabla g_i(\bar{w})$ for $i \in \{1, \ldots, m\}$ and $\nabla h_k(\bar{w})$ for $k \in \mathcal{A}(\bar{w})$ are linearly independent.

Definition 2.4 (Lagrangian function)**.** An important function associated with (2.1) is its Lagrangian function

$$\mathcal{L}(w, \lambda, \mu) = f(Y) + g(w)^\top \lambda + H(w)^\top \mu,$$

where λ and μ are the so-called Lagrange multipliers.

Considering the above definitions, the solution of the NLP can be obtained the following Theorem:

Theorem 2.5 (Karush–Kuhn–Tucker (KKT) conditions)**.** *If w^* is a local minimizer of* (2.1) *and LICQ holds at w^*, then there exists a $\lambda^* \in \mathbb{R}^{\mathrm{m}}$ and $\mu^* \in \mathbb{R}^{\mathrm{p}}$ such that:*

$$\nabla_w \mathcal{L}(w^*, \lambda^*, \mu^*) = 0, \tag{2.2a}$$
$$g(w^*) = 0, \tag{2.2b}$$
$$h(w^*) \leq 0, \tag{2.2c}$$
$$\mu^* \geq 0, \tag{2.2d}$$
$$h_k(w^*)\, \mu_k^* = 0, \quad k = 1, \ldots, p. \tag{2.2e}$$

Since all the OCPs that will be implemented follow the structure of (2.1), NLPs will be of great importance within this thesis. However, since there are NLP solvers to tackle the KKT conditions defined by (2.2), we will just use these solvers to work with the OCPs and in this section no more details on numerical algorithms will be given. Nevertheless, shall more information be required, [16] is an excellent source for optimization algorithms.

2.2 Discrete Optimal Control

Discrete OCPs solve discrete time dynamical systems. Such systems evolve in time in a predefined time grid, usually defined by a set of integers, and their dynamics can be model by:

$$x_{k+1} = \Phi_k(x_k, u_k), \quad k = 0, 1, 2, \ldots N-1 \tag{2.3}$$

A general discrete OCP is modeled as a NLP. In particular, the objective function is modeled as the sum of local non linear functions at time k that depend on u_k and x_k, i.e. $\sum_{k=0}^{N-1} L(x_k, u_k)$, plus a final special term $E(x_N)$ to penalize the last state within the system motion. Each of these $L(x_k, u_k)$ is known as a stage cost whereas $E(x_N)$ as the final cost. Furthermore, the NLP has to ensure the dynamical constraints given by (2.3), plus some other constraints including path and boundary conditions. As a result, a very generic discrete OCP can be formulated as the following NLP:

$$\underset{\substack{x_0,u_0,x_1,\ldots,\\ u_{N-1},x_N}}{\text{minimize}} \quad \sum_{k=0}^{N-1} L(x_k,u_k)+E(x_N) \tag{2.4a}$$

$$\text{subject to} \quad x_{k+1}-\Phi_k(x_k,u_k)=0, \quad k=0,\ldots,N-1, \tag{2.4b}$$

$$h(x_k,u_k)\leq 0, \quad k=0,\ldots,N-1, \tag{2.4c}$$

$$r(x_0,x_N)\leq 0. \tag{2.4d}$$

where Equation (2.4c) are the so-called path constraints and they ensure that every state x_k and control u_k ensure a certain condition along the solution. An example of these type of constraints would be safety limits on the actuations u_k. On the other hand, Equation (2.4d) represent the boundary conditions on the first and last states x_0 and x_N. Such conditions might include periodicity constraints, i.e. $x_0 = x_N$, or fixed value $\bar{x}_0$ for the initial state, i.e. $x_0 = \bar{x}_0$. In order to solve this type of problem, a general NLP solver is very often used.

Figure 2.1 illustrates the solution of a specific OCP with $x \in \mathbb{R}^2$ and $u \in \mathbb{R}$. Furthermore, the minimization function is quadratic and given by:

$$\sum_{k=0}^{N-1} L(x_k,u_k) = \sum_{k=0}^{N-1} \|x_k\|^2$$

so that the optimal solution brings the last state to $[0,0]^\top$ (assuming of course that $[0,0]^\top$ is feasible). Finally, the OCP also imposes a boundary condition on the last state x_N and a couple of path constraints to prevent the states from violating certain values.

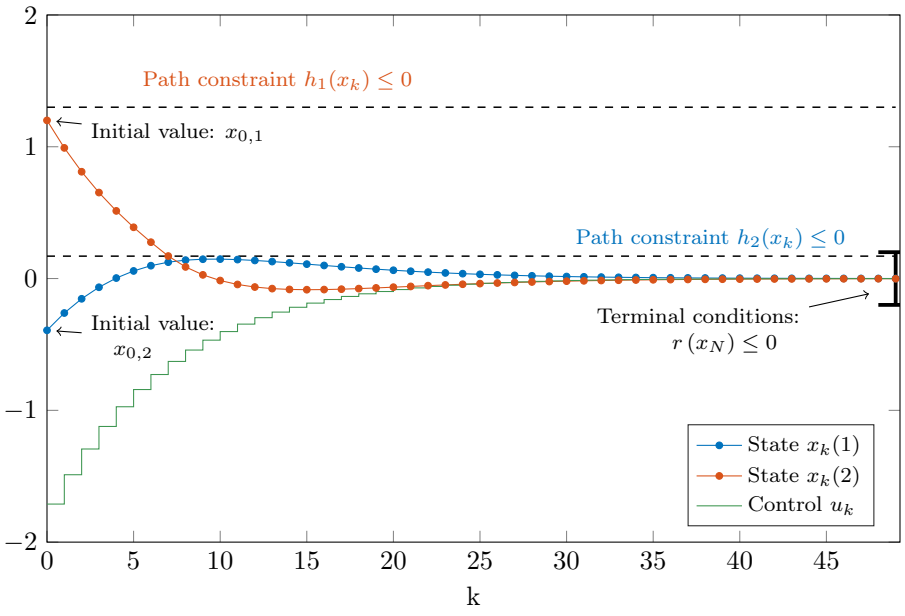


FIGURE 2.1: Discrete Time Optimal Control Problem

2.3 Continuous Time Optimal Control

As the name already states, this type of OCPs solve problems where the system lives in a continuous time frame $t \in \mathbb{R}$. As a result, the states $x(t)$ and the controls $u(t)$ are a function of time and the dynamics are defined by a differential equation $F(\dot{x}(t), x(t), u(t))$. Without loss of generality, we shall consider from now on that the dynamics are defined by an *ordinary differential equation (ODE)*, i.e. the evolution of the system is given by the following equation:

$$\dot{x}(t) = \Phi\big(x(t), u(t)\big)$$

Nevertheless, the theoretical foundations that will be covered can easily be extended to the more general class of system dynamics defined by *differential algebraic equations (DAEs)*.

The OCP structure in continuous time is very similar to the discrete NLP defined by (2.4), nevertheless there are some key differences that must be considered. First, the problem is evaluated at a time interval T, which leads to continuous optimization variables $x(t)$ and $u(t)$ having infinite dimension. Second and as a results of the previous fact, the constraints have to be ensured in a continuous time grid. Finally, the minimization function is in many cases an integration over the whole time interval and not a sum of discrete terms. As a result, the general NLP defining a continuous OCP is given by:

$$\begin{aligned}
\underset{x(\cdot),\,u(\cdot)}{\text{minimize}} \quad & \int_0^T L\big(x(t), u(t)\big)dt + E\big(x(T)\big) \\
\text{subject to} \quad & \dot{x}(t) - \Phi\big(x(t), u(t)\big) = 0, \quad t \in [0, T], \quad \text{(ODE model)}, \\
& h\big(x(t), u(t)\big) \leq 0, \quad t \in [0, T], \quad \text{(path constraints)}, \\
& r\big(x(0), x(T)\big) \leq 0, \quad \text{(boundary constraints)}
\end{aligned}$$

Figure 2.2 illustrates the solution of a specific continuous OCP. Now the state $x \in \mathbb{R}$ and the control $u \in \mathbb{R}$ are a continuous function of t. Furthermore, the OCP imposes a linear initial constraint and some terminal and path constraints.

In order to solve a continuous OCP there exists several options: the *Hamilton-Jacobi-Bellman equation*, *indirect methods* and *direct methods*. Due to their flexibility, success and widespread use nowadays, in this thesis only direct methods will be regarded.

2.4 Direct Methods

The idea of direct methods is to transform the infinite dimension continuous optimal control problem into a discrete and finite NLP. The idea behind is to use a numerical simulation method in a discrete and defined time grid $0 = t_0 < t_1 < t_2 < \dots t_N = T$ so that the continuous time dynamics can be approximated as a discrete system. As a result, instead of the infinite dimensional decision variables $u(t)$ and $x(t)$, a finite set of discrete values x_k and u_k, which represent the $u(t)$ and $x(t)$ values at the nodes of the discrete time grid, is obtained. The constraints are then evaluated at the nodes of this discrete time grid. Finally, once the problem is discretized, the problem is solved as in the general case of a discrete OCP. Because of the specific procedure, these methods are sometimes known as "first discretize, then optimize.

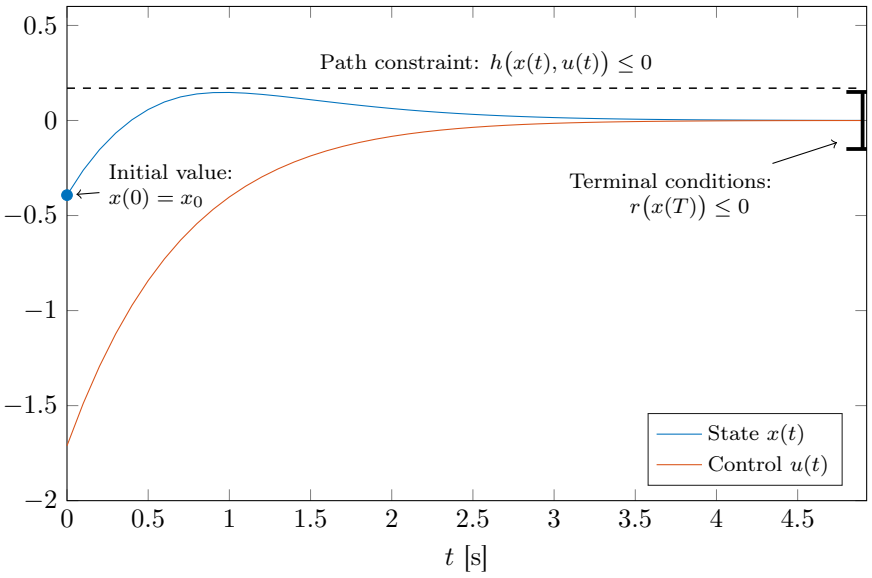


FIGURE 2.2: Continuous Time Optimal Control Problem

Depending on how the dynamics are discretized, three different direct methods can be distinguished: *single shooting*, *multiple shooting*, and *direct collocation*.

2.4.1 Single Shooting

The approach that single shooting takes is to define the discrete time grid $0 = t_0 < t_1 < \ldots < t_N = T$ and then model the infinite dimension controls $u(t)$ by piecewise polynomials. Since the most common choice is piecewise constant controls, this parameterization is normally used; in particular, the controls are referred to as $U = (u_0, u_1, \ldots, u_{N-1})$, with u_k constant in $[t_k, t_{k+1}]$. In a second step, single shooting computes the states $x(t)$ as dependent variables using a numerical integration method, an initial value x_0 and the discrete controls U. This computed trajectory can be defined as $x(U, t, x_0)$.

In this framework, single shooting substitutes $x(t)$ by the function $x(U, t, x_0)$ so that the only decision remaining variables are the discrete set of controls U and the initial value x_0. Furthermore, the constraints are evaluate in the specific nodes of the time grid. The resulting NLP is depicted below:

$$\begin{aligned} \underset{U,\, x_0}{\text{minimize}} \quad & \int_0^{t_N} L\Big(x\big(U, t, x_0\big), U\Big)\, dt + E\Big(x\big(t_N, U, x_0\big)\Big) \\ \text{subject to} \quad & h\Big(x\big(t_k, U, x_0\big), u_k\Big) \leq 0, \quad i = 0, \ldots, N-1, \\ & r\Big(x\big(t_N, U, x_0\big)\Big) \leq 0. \end{aligned}$$

Finally, Figure 2.3 illustrates this method. It can be observed how the controls are parameterized as piecewise constant in a time grid and the states are given as a function of these controls and x_0.

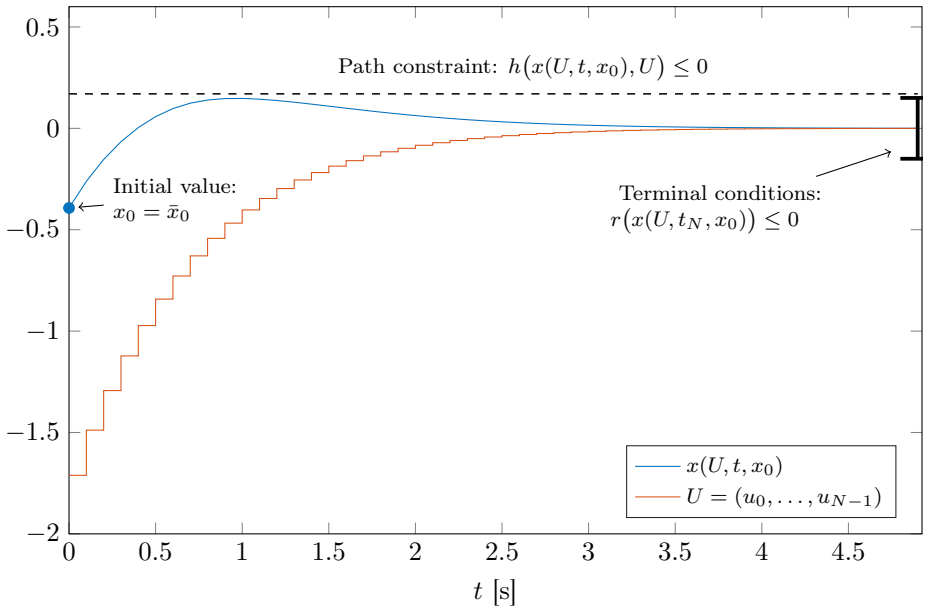


FIGURE 2.3: Single shooting OCP

2.4.2 Multiple Shooting

Originally developed by Bock and Plitt [17] and similarly to single shooting, it models piecewise controls in the discrete time grid $0 = t_0 < t_1 < t_2 < \ldots < t_N = T$, i.e.:

$$u(t) = u_k, \quad t \in [t_k, t_{k+1}].$$

The key difference with respect to single shooting is that, instead of computing $x(t)$ as a dependent variable $x(U, x_0, t)$, it discretizes the states in the time grid nodes and keeps them as independent variables. To do so, it solves the equations of motion locally on each interval $[t_k, t_{k+1}]$ and it constrains this set of local solutions to be feasible and consistent. In other words, it defines the discrete states as $X = (x_0, x_1, \ldots, x_N)$; it uses a numerical integration routine $\Phi_k(t, x_k, u_k)$ for each interval $[t_k, t_{k+1}]$to compute the integration of the system dynamics starting at x_k, with constant control u_k, and ending at time t; it ensures that each of these local numerical integration at the end of the interval equals to the next state:

$$\Phi_k(t_{k+1}, x_k, u_k) - x_{k+1} = 0.$$

Finally, the objective function integral is approximated by:

$$l_k(x_k, u_k) = \int_{t_k}^{t_{k+1}} L\big(\Phi_k(t_{k+1}, x_k, u_k), u_k\big) dt$$

and the discrete OCP can be formulated as:

$$
\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & \sum_{k=0}^{N-1} l_k(x_k, u_k) + E\left(x_N\right) \\
\text{subject to} \quad & x_0 - \bar{x}_0 = 0, \\
& \Phi_k(t_{k+1}; x_k, u_k) - x_{k+1} = 0, \quad i = 0, \ldots, N-1, \\
& h(x_k, u_k) \leq 0, \quad i = 0, \ldots, N-1, \\
& r\left(x_N\right) \leq 0.
\end{aligned}
$$

Figure 2.4 pictures this method in the middle of the optimization solver iterations. It can be seen how, as in single shooting, the controls are given as piecewise constant in a discrete time grid, but by contrast:

(i). The states are now independent variables.

(ii). The solution in the middle of the optimization algorithm is not dynamically feasible, i.e. in general $\Phi_k(t_{k+1}; x_k, u_k)$ is not equal to the variable x_{k+1} representing the state at the end of interval k. Remember that in single shooting, since $x(U, x_0, t)$ represents the model simulation at time t, starting at x_0 and using U, the dynamics are in turn always feasible and preserved.

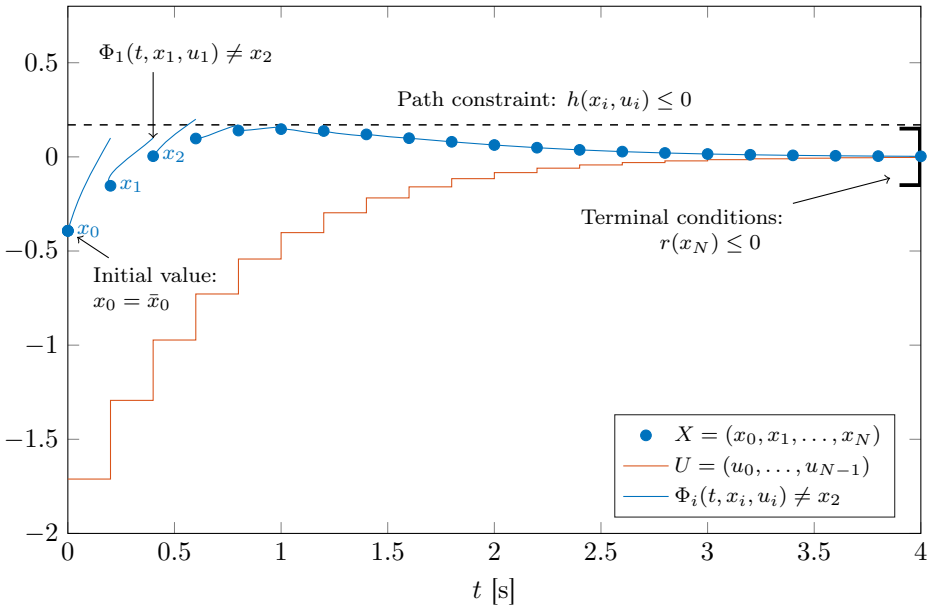


FIGURE 2.4: Multiple shooting OCP

2.4.3 Direct Collocation

Direct collocation is very similar to multiple shooting in the sense that it also uses the same discrete time grid $0 = t_0 < t_1 < t_2 < \ldots < t_N = T$ with the piecewise constant controls $U = (u_0, u_1, \ldots, u_{N-1})$ and the states as variables $X = (x_0, x_1, \ldots, x_N)$.

However, it differs with respect to multiple shooting in a key aspect: the integration of the dynamics $\Phi\big(x(t), u(t)\big)$ at every local interval $[t_k, t_{k+1}]$ is approximated by a local polynomial $p_k(t, q_k)$, with q_k as the polynomial coefficient vector. In order to ensure that the local polynomials are a good model of the system motion, a series of collocation conditions are added to the NLP.

In particular, the idea behind the collocation conditions is to divide the collocation interval $[t_k, t_{k+1}]$ into a set of m collocation points $t_{1,k}, \ldots, t_{m,k}$, and then to ensure that the derivative of the local polynomial at each one of these nodes matches the evaluation of the dynamics ODE at the same nodes. These conditions are represented for a single collocation interval by:

$$c_k(x_k, v_k, u_k) = \begin{bmatrix} \Phi\big(p_k(t_{1,k}, q_k), u_k\big) & - & \dot{p}_k(t_{1,k}, q_k) \\ \Phi\big(p_k(t_{2,k}, q_k), u_k\big) & - & \dot{p}_k(t_{2,k}, q_k) \\ & \vdots & \\ \Phi\big(p_k(t_{m,k}, q_k), u_k\big) & - & \dot{p}_k(t_{m,k}, q_k) \end{bmatrix} = 0$$

Moreover and as in multiple shooting, the continuity conditions $x_{k+1} = p_k(t_{k+1}, q_k)$ are also enforced. Finally, the objective function integral is approximated by a quadrature formula $l_k(x_k, q_k, u_k)$ using the same collocation points. Then, considering $Q = (q_0, \ldots, q_{N-1})$ as the collection of polynomial parameters, the resulting NLP can be stated as:

$$\begin{aligned} \underset{X,U,Q}{\text{minimize}} \quad & \sum_{k=0}^{N-1} l_k(x_k, q_k, u_k) + E\left(x_N\right) \\ \text{subject to} \quad & x_0 - \bar{x}_0 = 0, \\ & c_k(x_k, v_k, u_k) = 0, \quad i = 0, \ldots, N-1, \\ & p_k(t_{k+1}, q_k) - x_{k+1} = 0, \quad i = 0, \ldots, N-1, \\ & h(x_k, u_k) \leq 0, \quad i = 0, \ldots, N, \\ & r\left(x_N\right) \leq 0. \end{aligned}$$

A very common set of polynomials used for direct collocation are *Lagrange polynomials* which use two variants, *Legendre* and *Radau*, in order to perform the selection of the collocation nodes.

2.5 Online Optimal Control

The previous sections explain how to optimize a dynamical system offline, i.e. how to obtain the optimal policy U^* that can bring the system to a desired state. Nevertheless, if this policy U^* is used to control the real system, it is very likely that the result will not be as expected. This is a widely know problem of the so-called *open-loop controllers*: using a

precomputed control policy without any system feedback in a real system, which has model-plant mismatches disturbances, ends up bringing the system to an undesired state. In the specific case of unstable plants, this effect can even lead to system breakdowns.

A different approach explained in this section is to observe the system evolution and use this information to recalculate the policy U^*, so that, despite mismatches and disturbances, the system evolves as desired. This type of control is known as *close-loop control* or *feedback control*.

A basic implementation of feedback control is the well known PID controller. However, if an optimal policy U^* is desired, i.e. a set of controls that optimizes a cost function and guarantees some constraints, U^* has to be computed in real time (online) by solving an OCP at every time step.

This task is quite challenging due to the computational effort: the controller has to solve at each iteration a new optimization problem in a very small amount of time Δt, where Δt is the discretized time step. In order to tackle the high computational load of solving such a problem, online optimal control has to exploit a series of numerical/algorithmic approximations. The implementation of this set of approximations is known as *embedded optimization* or *real-time optimization*, and optimal feedback control using embedded optimization is known as model predictive control (MPC). For the particular case of non-linear dynamics, the term nonlinear MPC (NMPC) is used instead.

NMPC defines a family of control algorithms which do not designate a specific control strategy but instead share a very similar structure and make use of the same set of conceptual ideas [18]:

- They make explicit use of a system model to predict the system output in a future time interval (horizon).
- They calculate the optimal control policy U^* by minimizing an objective function $J(X, U)$.
- At each time step the first control of U^* is applied to the system and the horizon is shifted a time step into the future. This is also known as receding strategy.

With this set of conceptual ideas, the basic NMPC scheme can be summarized as follow:

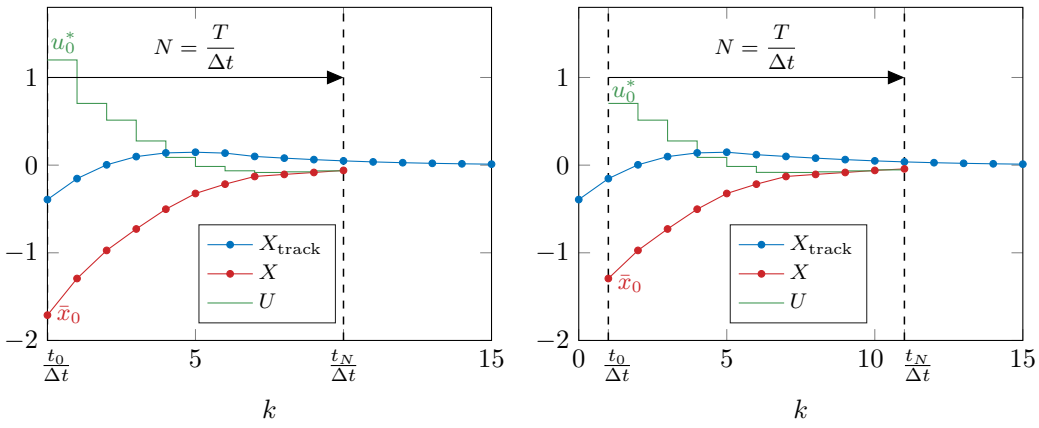
(i). Obtain the current system state $\bar{x}_0$.

(ii). Solve an OCP using a limited time window T with N steps. The OCP should impose the initial constraint $x_0 = \bar{x}_0$ as well as dynamic constraints. Path or other constraints might also be also enforced.

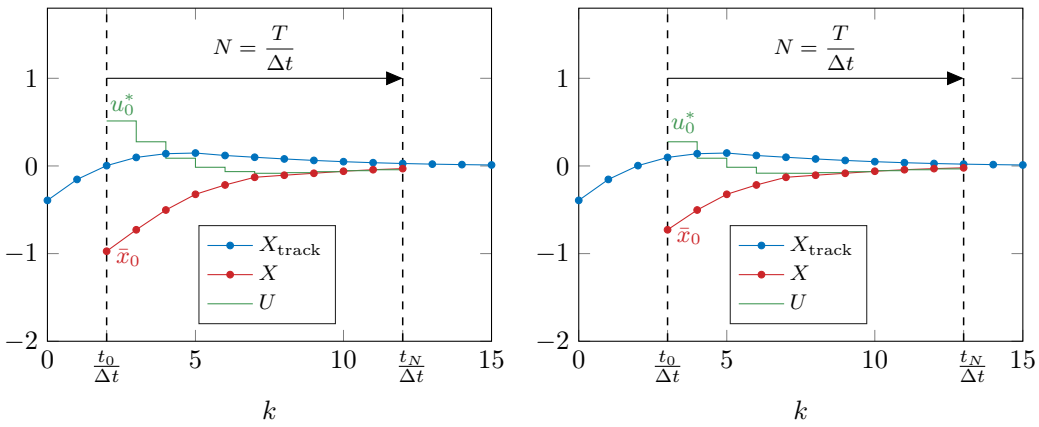
(iii). Apply the first control u_0^* of the optimal policy U^* to the system.

(iv). Move the optimization horizon a time step forward, observe the new current state $\bar{x}_0$ and repeat the procedure.

Figure 2.5 represents four consecutive iterations of a NMPC controller. This NMPC tries solve the OCP that minimize the least square error between the system state X and some state reference trajectory X_{ref}. As a result, at each iteration it computes the optimal control policy that makes the state X converge towards the reference trajectory X_{ref}.



(A) $t_0 = 0$ (B) $t_0 = \Delta t$



(C) $t_0 = 2\Delta t$ (D) $t_0 = 3\Delta t$

FIGURE 2.5: NMPC evolution in 4 consecutive time steps

Chapter 3

Offline Generation of Optimal Trajectories

As stated in the thesis introduction, the first line of work will be committed to study and research numerical methods and dynamical models to obtain periodic flight trajectories that maximize the system efficiency η_{Loyd}; in particular, we will continue the work of [12] by looking at four different issues:

(i). Reformulation of flight safety condition to increase the extracted power.

(ii). Examination of LICQ deficiency in periodical OCPs and how it might influence our specific problem.

(iii). Performance comparison between dynamical models based on quaternions, Euler angles, and rotation matrices.

(iv). Analysis of flight topologies by looking at two different topics: study of OCPs without topology constraints and generation of circular flight topologies that differ from the standard lemniscates.

3.1 Original Problem

Before diving into the different contributions of this thesis and in order to set a comprehensible background, the original problem [12] used as a base for this research shall be explained. In particular, the initially proposed OCP will be defined and the obtained results illustrated.

3.1.1 Optimal Control Problem Formulation

In order to explain the OCP in a consistent manner, this section will be divided into the main OCP parts: objective function, dynamical model and constraints; in particular, a continuous OCP will be first formulated, and then, the discretization procedure to obtain a computational algorithm will be defined.

OCP Objective Function

In order to model the objective function, two different goals were considered:

(i). The main target of maximizing the generated power per flight period.

(ii). A secondary objective to smoothen the actuation δ.

In particular, the extracted power was obtained as the generated mechanical power averaged over a periodic cycle:

$$P = \frac{1}{T}\int_0^T \dot{l}F_{\text{tether}}\mathrm{d}t = \frac{\rho A C_{\text{R}}}{2}\frac{E}{\sqrt{1+E^2}}\frac{1}{T}\int_0^T \dot{l}v_{\text{a}}^2\mathrm{d}t,$$

where T represent the time period, A the projected area of the kite in the $\vec{e}_{\text{roll}}$-$\vec{e}_{\text{pitch}}$ plane, ρ the air density and $C_{\text{R}} = \sqrt{C_{\text{L}}^2 + C_{\text{D}}^2}$ the aerodynamic force coefficient; moreover, in order to model the above equation, an extra system state W was introduced and its dynamics defined by $\dot{W} = \dot{l}v_{\text{a}}$.

Secondly, since the actuation pod has a speed limitation, the control δ was introduced as a system state and its rate $\dot{\delta}$ as a control; then, a penalty on $\dot{\delta}$ was added as part of the objective function. As a result, the objective function was modeled by:

$$f_{\text{obj}} = -\frac{1}{T}\int_0^T \dot{l}v_{\text{a}}^2\mathrm{d}t + \epsilon_\delta\int_0^T |\dot{\delta}|^2\mathrm{d}t = -\frac{W(T)}{T} + \epsilon_\delta\int_0^T |\dot{\delta}|^2\mathrm{d}t \tag{3.1}$$

Dynamical Model

Aiming at singularity-free equations of motions as well as system dynamics without trigonometric functions, [12] proposed an equivalent model of Model (1.3) by replacing the three angles ψ, φ, and ϑ by a quaternion formulation $q = [q_0,\ q_1,\ q_2,\ q_3]^\top$. The main claim to perform this change was that, by avoiding singularities and trigonometric functions, the optimization problem had a more convenient structure.

Considering the above quaternion formulation, the tether length as a fifth state, and the extra states W and δ, the equations of motion were derived as:

$$\begin{aligned}
\dot{q} = \begin{bmatrix}\dot{q_0}\\ \dot{q_1}\\ \dot{q_2}\\ \dot{q_3}\end{bmatrix} &= \frac{v_{\text{a}}}{2l}\begin{bmatrix}-q_2\\ -q_3\\ q_0\\ q_1\end{bmatrix} + \frac{v_{\text{w}}}{l}\begin{bmatrix}q_0(q_2^2+q_3^2)\\ q_1(q_2^2+q_3^2)\\ -q_2(q_0^2+q_1^2)\\ -q_3(q_0^2+q_1^2)\end{bmatrix} + \frac{g_{\text{k}}v_{\text{a}}\delta}{2}\begin{bmatrix}q_1\\ -q_0\\ -q_3\\ q_2\end{bmatrix},\\
\dot{l} &= v_{\text{winch}},\\
\dot{W} &= \dot{l}v_{\text{a}},\\
\dot{\delta} &= \dot{\delta}_{\text{c}},
\end{aligned} \tag{3.2}$$

with the system state x and the system controls u given by:

$$\begin{aligned}
x &= [q_0, q_1, q_2, q_3, l, \delta, W]^\top\\
u &= [v_{\text{winch}}, \dot{\delta}_{\text{c}}]^\top,
\end{aligned}$$

and where the air path speed was defined as:

$$v_{\text{a}} = Ev_{\text{w}}(q_0^2 + q_1^2 - q_1^2 - q_3^2) - E\dot{l}.$$

Physical Constraints

Due to physical limitations, a first set of OCP constraints had to be modeled; in particular, due to weight and geometrical design considerations, v_{winch}, δ, the rate of δ, and the tether length were bounded:

$$\begin{aligned} |\dot{\delta}| &\leq \dot{\delta}_{\max}, \\ |\delta| &\leq \delta_{\max}, \\ v_{\text{winch,min}} &\leq v_{\text{winch}}, \\ l &\leq l_{\max}, \end{aligned} \tag{3.3}$$

Furthermore, in order to keep the system tethered, a minimum air path was also imposed:

$$v_{\text{a}} = E v_{\text{w}}(q_0^2 + q_1^2 - q_1^2 - q_3^2) - E\dot{l} \geq v_{\text{a,min}}. \tag{3.4}$$

Geometric Constraints

Regarding the nature of the flight periods, a second set of OCP constraints was modeled; in particular, since the pumping cycles have to be closed, periodic boundary conditions on the system states were enforced:

$$q(T) = q(0), \tag{3.5a}$$
$$l(T) = l(0), \tag{3.5b}$$
$$\delta(T) = \delta(0). \tag{3.5c}$$

Furthermore, to ensure a proper initialization of the power integration, an initial boundary condition was considered:

$$W(0) = 0. \tag{3.6}$$

Finally, in oder to ensure a safety distance with respect to the ground surface, a minimum elevation angle $\theta_{\min}$ was derived [12, Equations (58)-(59)] as:

$$(q_0^2 + q_1^2 - q_2^2 - q_3^2)\tan\theta_{\min} + 2(q_1 q_3 - q_0 q_2) \leq 0. \tag{3.7}$$

Topological Constraints

Considering that the kite should fly lemniscates to avoid tether twisting, a third and final set of constraints was modeled; in particular, in order to ensure this flight topology, extra constraints were added to the problem to impose a specific flight direction (kite flying to the left or to the right).

In particular, in order to enforce the flight direction in a scenario with n lemniscates, the time period T was divided into a time grid $0 = t_0 < t_1 < \ldots < t_n = T$ of $n+1$ points. Then, in each of these intervals, a constraint on the flight direction was imposed by:

$$\begin{aligned} &\dot{\varphi} \leq 0 \text{ (fly left)}, \quad \text{for } t_{2i-2} \leq t \leq t_{2i-1} \text{ and } i = 1, \ldots, (n-1)/2, \\ &\dot{\varphi} \geq 0 \text{ (fly right)}, \quad \text{for } t_{2i-1} \leq t \leq t_{2i} \text{ and } i = 1, \ldots, (n-1)/2 \end{aligned} \tag{3.8}$$

Then, considering that $\dot{\varphi} = -v_{\text{a}} \sin\psi/(l \sin\vartheta)$, that $\sin\vartheta$ is always positive as the kite flies above the ground, and that $\psi = \text{arctan2}\ (q_0 q_3 - q_1 q_2,\ q_0 q_2 + q_1 q_3)$, the topology constraint was equivalently expressed by:

$$\begin{aligned} &q_0 q_3 - q_1 q_2 \geq 0, \quad \text{for } t_{2i-2} \leq t \leq t_{2i-1} \text{ and } i = 1, \ldots, (\text{n}-1)/2, \\ &q_0 q_3 - q_1 q_2 \leq 0, \quad \text{for } t_{2i-1} \leq t \leq t_{2i} \text{ and } i = 1, \ldots, (\text{n}-1)/2. \end{aligned} \tag{3.9}$$

This whole idea is better depicted in Figure 3.1. From it, the equivalence between flying to the left (right) and $\dot{\varphi} \leq 0$ ($\dot{\varphi} \geq 0$) can be easily understood.

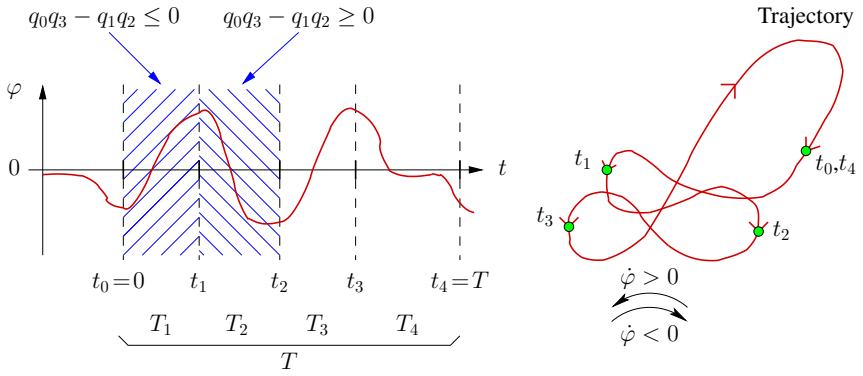


FIGURE 3.1: Topological constraints shown for a pumping cycle consisting of two lemniscates. It can be observed how the sign of $q_0q_3 - q_1q_2$ indicates whether the kite flies to the right or to the left [12].

It is important to remark that, in the topology constraints defined by Equation (3.9), no topology is defined in the last interval $t_{n-1} \leq t \leq t_n$; the reason for this lack of constraint is that in the last interval the kite is in the retraction phase, and thus, a topology constraint is not required.

As a last observation, it is important to indicate that the durations $T_i = t_i - t_{i-1}$ of the n time grid intervals were included in the optimization problem as variables; this was a mandatory requirement in order to have a flexible time grid that allows a full power optimization.

Quaternion Regularization

As it is widely known, any quaternion formulation has to ensure the well known invariant $\|q\|^2 = 1$. Moreover, since in general the quaternion norm is directly preserved by the system dynamics given by Equation (3.2), an idea to impose this condition could be to add $\|q(0)\|^2 = 1$ as an OCP constraint. However, as shown in previous studies [19], this scenario leads to LICQ deficiency and it is in practice a very poor choice.

To solve this issue, the OCP included a decay regularization in the dynamics so that $\|q\|^2 = 1$ could be ensured without incurring in LICQ deficiency; in particular, the new dynamics were defined by:

$$\begin{aligned}
\dot{q} &= \frac{v_a}{2l}\begin{bmatrix} -q_2 \\ -q_3 \\ q_0 \\ q_1 \end{bmatrix} + \frac{v_w}{l}\begin{bmatrix} q_0(q_2^2+q_3^2) \\ q_1(q_2^2+q_3^2) \\ -q_2(q_0^2+q_1^2) \\ -q_3(q_0^2+q_1^2) \end{bmatrix} + \frac{g_k v_a \delta}{2}\begin{bmatrix} q_1 \\ -q_0 \\ -q_3 \\ q_2 \end{bmatrix} - \gamma_q(\|q\|^2 - 1)q, \\
\dot{l} &= v_{\text{winch}}, \\
\dot{W} &= \dot{l} v_a, \\
\dot{\delta} &= \dot{\delta}_c,
\end{aligned} \tag{3.10}$$

where the term $\gamma_{\mathrm{q}}(\|q\|^2-1)q$ ensured that, at the end time T of the simulation, $\|q(T)\|^2=1$, and that as a result, any other quaternion state $q(t)$ would also preserve the invariant.

It is important to remark that LICQ deficiency is an important topic in periodic OCPs, and that as a result, Section 3.3 discusses the same matter but more extensively.

System and OCP parameters

In order to provide an overview of the different parameters used in the previous sections to model the dynamics, objective and constraints, Table 3.1 summarizes their numerical values and definitions.

TABLE 3.1: System parameters used for modeling and solving the original OCP.

Parameter	Value	Units	Description.
A	21	m^2	Projected area of the kite.
C_{R}	1		Aerodynamic force coefficient.
$\dot{\delta}_{\max}$	0.6	1/s	Bound of the steering actuator speed.
$\delta_{\max}$	0.7		Bound of the steering actuator.
E	5		Glide ratio.
g_{k}	0.1	rad/m	Steering constant.
$l_{\max}$	300	m	Bound on the tether length.
$\theta_{\min}$	0.35	rad	Minimum elevation angle.
ρ	1.2	$\mathrm{kg/m}^3$	Air density.
v_{w}	10	m/s	Wind speed which is assumed to be constant.
$v_{\mathrm{a,min}}$	5	m/s	Minimal air path speed to ensure stability.
$v_{\mathrm{winch,min}}$	-5	m/s	Lower bound on the winch speed.
γ_{q}	0.01	1/s	Quaternion decay to enforce the quaternion norm.

OCP Continuous Formulation

Regarding the defined objective, dynamics and constraints, the OCP can be finally summarized as:

$$
\begin{aligned}
\underset{\substack{x(\cdot),u(\cdot),\\ t_1,\ldots,t_n}}{\text{minimize}} \quad & E(x(t_n),t_n)+\int_0^{t_n} L(u(t))\mathrm{d}t \\
\text{subject to} \quad & \dot{x}-\Phi(x(t),u(t))=0, \quad t\in[t_0,t_n] && \text{(Dynamical model)},\\
& r(x(t_0),x(t_n))=0 && \text{(Boundary conditions)},\\
& h(x(t),u(t))\leq 0, \quad t\in[t_0,t_n] && \text{(Path constraints)},\\
& o_i(x(t))\leq 0, \quad t\in[t_{i-1},t_i], \quad i=1,\ldots,n, && \text{(Topology constraints)}
\end{aligned}
\tag{3.11}
$$

where the dynamical model $\Phi(\cdot)$ is defined by (3.10), the boundary constraints $r(\cdot)$ by (3.5) and (3.6), the path constraints by (3.3), (3.4) and (3.7), the topological constraints $o_i(\cdot)$ by (3.9), and where the objective terms E and L respectively represent the mechanical power and the control regularization of Equation (3.1).

OCP Discrete Formulation

In order to discretize the continuous OCP, a multiple shooting method was used. However, since the total time T of the OCP was divided into several time intervals T_i, the multiple shooting approach required some small modifications.

In particular, for each time interval T_i an independent and equally distributed time grid of m_i points was created. Then, for each of these intervals, $m_i + 1$ states, m_i controls and the time T_i were defined as problem variables:

$$w_i = (x_{i,0}, u_{i,0}, \dots, x_{i,m_i-1}, u_{i,m_i-1}, x_{i,m_i}, T_i).$$

Finally, the states within each time grid were related by the multiple shooting constraints $x_{i,k+1} = \Phi_k(x_{i,k}, u_{i,k}, T_i/m_i)$.

However, unlike the standard multiple shooting approach, the above formulation is not enough; in particular, by only considering the defined constraints, the states between time intervals would be independent of each other. As a result, continuity conditions $x_{i,m_i} = x_{i+1,0}$ between the time grids had to be added. The resulting discrete OCP can be defined by:

$$\begin{aligned}
&\underset{w_1,\dots,w_n}{\text{minimize}} \quad E(x_{n,m_n}, \sum_{i=1}^{n} T_i) + \sum_{i=1}^{n}\sum_{k=1}^{m_i} l_i(u_{i,k}, T_i) \\
&\text{subject to} \\
&x_{i,k} - \Phi_k(x_{i,k-1}, u_{i,k-1}, T_i) = 0, \quad i = 1,\dots,n,\ j = 1,\dots,m_n \quad \text{(Dynamical contraints)}, \\
&x_{i,m_i} - x_{i+1,0} = 0, \quad i = 1,\dots,n-1, \quad \text{(Continuity conditions)}, \\
&r(x_{1,0}, x_{n,m_n}) = 0, \quad \text{(Boundary conditions)}, \\
&h(x_{i,k}, u_{i,k}) \le 0, \quad i = 1,\dots,n,\ j = 1,\dots,m_n \quad \text{(Path contraints)}, \\
&o_i(x_{i,k}) \le 0, \quad i = 1,\dots,n,\ j = 1,\dots,m_n \quad \text{(Topology contraints)},
\end{aligned}$$

with:

$$l_i(u_{i,k}, T_i) = \frac{T_i}{m_i}\epsilon_\delta \dot{\delta}_{i,k}^2 + \frac{T_i}{m_i}\epsilon_v (v_{\text{winch},i,j} - v_{\text{winch,next}(i,j)})^2,$$

next(i,j) denoting the subsequent index par, and where the second term penalizing the acceleration on the winch speed was only added to the discrete formulation.

It is important to remark that, in order to discretize the dynamics and obtain the $\Phi_k(x_{i,k-1}, u_{i,k-1}, T_i)$ integrators, a Runge-Kutta integrator of order 4 (RK4) [20] was used.

3.1.2 Numerical Results

In order to solve the discrete OCP, CasADi [21, 22] and IPOPT [23] were used. In particular, using a total number of 250 discrete time points and 6 lemniscates (n=12), the OCP was solved in 487 iterations with an efficiency η_{Loyd} of 33%.

3.2 Modified Safety Conditions

In the original OCP formulation, the safety condition given by Equation (3.7) was designed to ensure a minimum distance between the kite and the ground. In particular, (3.7) imposed a constraint in the elevation angle so that a minimum altitude was achieved.

However, the above implementation has a problem: despite ensuring a minimum distance, this minimum altitude increases with the tether length (larger tether leads to larger altitude). As a result, the airpath speed v_a decreases as the tether reels out and the extracted power and η_{Loyd} factor are not fully maximized.

An alternative and more natural approach would be to impose a constraint to ensure a minimum altitude $h_{\min}$:

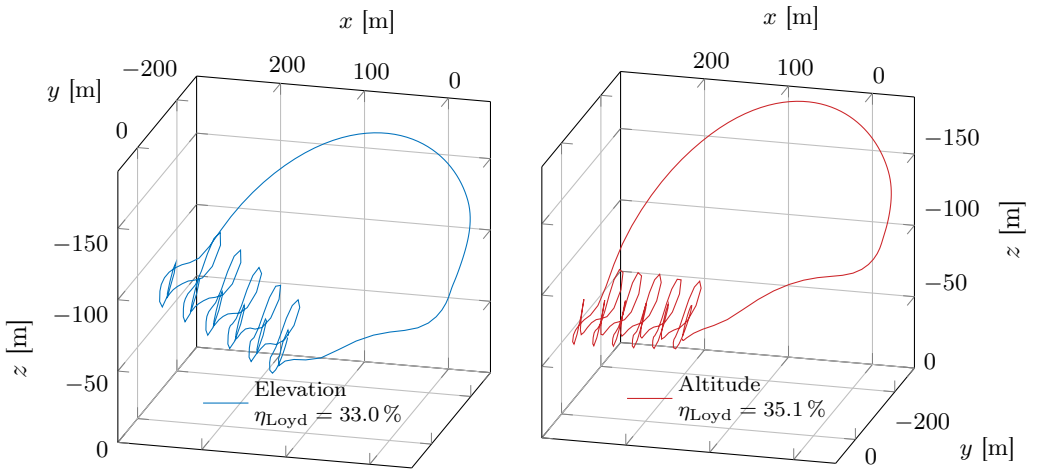
$$2l(q_0q_2 - q_1q_3) \geq h_{\min}, \tag{3.12}$$

where the altitude value $2l(q_0q_2 - q_1q_3)$ is obtained from [12, Equation (39)].

Finally, considering that in the conventional OCP solution the kite achieved a minimum altitude of 40 m, a sensible value (with a safe margin) for $h_{\min}$ could be 60 m.

Results

After replacing (3.7) by (3.12) with $h_{\min} = 60\,\text{m}$, the OCP was solved again. A comparison between the solution of this OCP and the standard one can be seen in Figure 3.2. By comparing the η_{Loyd} factor of both scenarios, it can be concluded that the new constraint does indeed increases the extracted power; in particular, the new safety condition leads to a power efficiency 2.1 % larger.



(A) Optimal solution imposing a minimum elevation angle as a safety condition.

(B) Optimal solution imposing a minimum altitude as a safety condition.

FIGURE 3.2: Optimal solution comparison considering the two different options to impose the safety condition. The altitude as safety condition leads to a more power efficient solution.

3.3 Invariants and LICQ Deficiency

As a first step to research and comprehend the original formulation, the OCP was recreated and tested. During this process, two different effects made us suspect that the original formulation could be ill-posed:

(i). The final solution was very sensitive to the initial guess: small changes in the initial conditions would lead in some cases to a divergent solution. In particular, even when the algorithm was initialized using an old solution, the OCP did not always converge.

(ii). In those cases were a solution was found, the states during the iteration procedure reached highly infeasible values.

3.3.1 Invariants in Periodic OCP

As previously explained in Section 2, the KKT conditions defined by (2.2a)-(2.2e) are not by themselves a strong enough condition to ensure optimality. In particular, it was demonstrated that a point w^* would be a local optimal of a NLP if and only if, besides solving (2.2a)-(2.2e), LICQ would also hold at w^*.

Definition 3.1 (Invariants in Dynamical Systems)**.** Regard a general dynamical model $\dot{x} = \Phi(x, u)$, where $\Phi \in \mathbb{R}^n \to \mathbb{R}^n$. The dynamics have a k-dimensional invariant $\mathcal{I}(x(t))$, with $k < n$, if:

$$\mathcal{I}(x(t)) = \text{constant} \quad \forall\ x(0), u(t).$$

In this scenario, the system state $x(t) \in \mathbb{R}^n$ is forced to evolve in a $n - k$ dimensional manifold.

Theorem 3.2 (LICQ Deficiency in Periodic Optimal Control Problems)**.** *Regard a simple and general periodic optimal control problem:*

$$\begin{aligned} \underset{x(\cdot),\,u(\cdot)}{\text{minimize}} \quad & f_{\text{obj}}(x(\cdot), u(\cdot)) \\ \text{subject to} \quad & \dot{x}(t) - \Phi(x(t), u(t)) = 0, \\ & x(T) - x(0) = 0. \end{aligned}$$

If the system dynamics $\dot{x}(t) = \Phi(x(t), u(t))$ have an invariant $\mathcal{I}(x(t))$, then the periodicity constraint leads to LICQ deficiency [19, 24].

Proof. The above OCP can be reformulated by considering an integration function $x(t) = F(x_0, u, t)$ and defining the last state $x(T)$ by $F(x_0, u, T) = G(x_0, u)$:

$$\begin{aligned} \underset{u,\,x_0}{\text{minimize}} \quad & f_{\text{obj}}(x_0, u) \\ \text{subject to} \quad & \pi = G(x_0, u) - x_0 = 0. \end{aligned}$$

Then, by the invariant definition:

$$\mathcal{I}\big(G(x_0,u)\big)-\mathcal{I}\big(x_0\big)=0,\quad \forall\ w=\begin{bmatrix}u\\x_0\end{bmatrix}$$

and thus:

$$\nabla_w\Big[\mathcal{I}\big(G(x_0,u)\big)-\mathcal{I}\big(x_0\big)\Big]=\big(\nabla_w G(x_0,u)\nabla_w x_0\big)\nabla_{x_0}\mathcal{I}(x_0)=0,$$

which finally yields:

$$\nabla_w\pi\cdot\nabla_{x_0}\mathcal{I}(x_0)=0$$

Since in general $\nabla_{x_0}\mathcal{I}(x_0)\neq 0$, the above condition implies that $\nabla_w\pi=0$ and that in turn there is LICQ deficiency in the periodic conditions [24]. □

The conceptual idea behind the above result is that, since the dynamics preserve some system condition, the periodicity constraint contains redundant information. In particular, since the dynamics evolve in a reduced $n-k$ dimensional manifold, the periodicity constraint, which was originally defined in $\mathbb{R}^n$, should be defined in $\mathbb{R}^{n-k}$ to avoid redundancy.

3.3.2 The Projection Method for Invariants

In order to avoid the LICQ deficiency, a proposed method [19] is to impose the periodicity constraints only along the directions that are tangential to the invariant. To do that, the periodicity constraints have to be *projected* into the manifold of tangential directions. In particular, regarding the Jacobian of the invariant evaluated at x_0:

$$\frac{\partial\mathcal{I}(x_0)}{\partial x}=\nabla\mathcal{I}(x_0)^\top$$

and the basis matrix Z of its null space:

$$\nabla\mathcal{I}(x_0)^\top Z=0,\quad \text{with}\ \ Z^\top Z=I,$$

LICQ failure can be avoided by reformulating the periodicity constraint $g(x_0,u)-x_0=0$ by:

$$Z^\top(g(x_0,u)-x_0)=0.$$

Note that, since $Z\in\mathbb{R}^{n\times z-k}$, the above conditions projects the original periodicity constraint into a reduced $z-k$ dimensional manifold.

In the method described above, it was assumed that the invariant was a property of the dynamics but that its specific value was not relevant for the OCP; nevertheless, in many applications, the invariant has a fixed and known value K that the OCP must ensure. In this scenario, the projection method still works; particularly, the OCP can be modified as:

$$\begin{aligned}\underset{u,\,x_0}{\text{minimize}}\quad & f_{\text{obj}}(x_0,u)\\ \text{subject to}\quad & Z^\top(g(x_0,u)-x_0)=0,\\ & \mathcal{I}(x_0)=\text{K},\end{aligned}$$

and LICQ would still hold [24].

3.3.3 Stabilization of Invariants

Given some system dynamics $\dot{x} = \Phi(x, u)$ preserving a 1-dimensional invariant $\mathcal{I}(x) \in \mathbb{R}$, if the value of the invariant is known and fixed, i.e. $\mathcal{I}(x) = \mathrm{K}$, the invariant stabilization [25] can be used as an alternative to the projection method in oder to avoid LICQ deficiency in periodic OCPs.

In particular, the working principle is to add a decay term to the system dynamics:

$$\dot{x} = \Phi(x, u) - \gamma x\big(\mathcal{I}(x) - \mathrm{K}\big),$$

so that, the new dynamics do no longer preserve the invariant $\mathcal{I}(x)$, but instead, they ensure that the value of the invariant at the end of the simulation horizon is equal to the desired value K:

$$\mathcal{I}\big(x(T)\big) = K.$$

As a result, since the dynamics do no longer preserve the invariant, the periodicity condition $x(0) = x(T)$ can be imposed without LICQ deficiency; furthermore, since $x(0) = x(T)$ is enforced, the method also ensures that:

$$\mathcal{I}\big(x(0)\big) = \mathcal{I}\big(x(T)\big) = K. \tag{3.13}$$

Finally, considering the effect of the decay in the system evolution, it should hold that:

$$\mathcal{I}\big(x(0)\big) \geq \mathcal{I}\big(x(t_1)\big) \geq \mathcal{I}\big(x(t_2)\big) \geq \mathcal{I}\big(x(T)\big), \quad \forall \;\; 0 \leq t_1 \leq t_2 \leq T$$

which considering (3.13) ensures that:

$$\mathcal{I}\big(x(t)\big) = \mathrm{K}, \quad \forall\, t \in [0, T].$$

As a result, the stabilization method avoids LICQ deficiency by removing the invariant from the dynamics, and by doing so, ensures that the invariant value is fixed to some value K along the solution.

It is important to note that, in practice, this algorithm has some drawbacks which must be outlined:

- Unlike the projection method, its performance depends on the selected value for the parameter γ. In particular, a small γ might not ensure the correct value of the invariant at the end of the horizon; by contrast, a large γ might modify the real dynamics too much.
- By adding the stabilization term, the dynamics might become stiff and the use of an implicit integrator might be required.
- Since the dynamics are modified, the integration might become harder to compute and the solver might struggle more to solve the OCP.

3.3.4 The Quaternion Case

As explained in Section 3.1, the kite dynamics have the quaternion invariant $\|q\|^2 = 1$. Therefore, any periodic OCP using the kite dynamics has to implement one of the above two methods to avoid LICQ deficiency. In particular, as described in Section 3.1, the original formulation used the stabilization method with a decay $-\gamma q(\|q\|^2 - 1)$.

Nevertheless, after observing the strange numerical performance of the original formulation and considering the mentioned drawbacks of the stabilization method, we decided that a deeper study on the topic had to be conducted. In particular, we determined that, in order to assess the quality of both approaches, the original OCP and a modified OCP including the projection method had to be compared for different test scenarios.

3.3.5 Projection Method Implementation

As a first step to implement the projection method, the Jacobian $\nabla_q \mathcal{I}(q)^\top$ has to be computed. In particular, considering $\mathcal{I}(q) = \|q\|^2$, the Jacobian can be computed as:

$$\nabla_q \mathcal{I}(q) = \nabla_q \|q\|^2 = \begin{bmatrix} 2q_0 \\ 2q_1 \\ 2q_2 \\ 2q_3 \end{bmatrix}.$$

In a second step, the basis matrix $Z \in R^{4\times 3}$ of the null-space of $\nabla_q \mathcal{I}\big(q(0)\big)$ has to be obtained. In particular, a basis matrix satisfying:

$$Z^T \nabla_q \mathcal{I}(q(0)) = Z^T \begin{bmatrix} 2q_0^{(0)} \\ 2q_1^{(0)} \\ 2q_2^{(0)} \\ 2q_3^{(0)} \end{bmatrix} = 0$$

with $q_i^{(0)} = q_i(0)$, is given by:

$$Z = \begin{bmatrix} -q_1^{(0)} & -q_2^{(0)} & -q_3^{(0)} \\ q_0^{(0)} & 0 & 0 \\ 0 & q_0^{(0)} & 0 \\ 0 & 0 & q_0^{(0)} \end{bmatrix}.$$

Finally, the projection method is directly implemented within the OCP by simply substituting the periodicity constraint (3.5) by

$$Z^T(q(0) - q(T)) = 0,$$

and then adding the boundary condition:

$$\|q(0)\|^2 = 1.$$

3.3.6 Results

In order to obtain a good evaluation of the two alternative methods, the OCPs will be solved considering combinations of the following problem variations:

(i). In order to test the performance of different initial guesses, the OCPs will consider three different initializations: the standard real flight data used in the original approach, an optimal trajectory obtained from a different OCP, and a trajectory obtained by a simulation routine.

(ii). In order to study the stiffness of the dynamics, three different direct methods will be regarded: multiple shooting with an explicit RK4 integrator, a direct collocation using a Radau scheme, and direct collocation with a Legendre scheme.

(iii). To judge the effect of the wind speed, the OCPs will be solved for three different wind speeds: 6, 10 and 15 m/s.

(iv). Finally, to measure the effect of the problem size, three different number of discrete points will be considered: the original N=250, N=500 and N=1000.

Considering every possible combination of the above variations, a total of 162 tests (81 per invariant method) have to be performed. A summary of the outcome of these 162 experiments is depicted in Figure 3.3; particularly, it can be observed that, while both approaches have low success rates, the projection method solves 60 % more cases than the invariant stabilization method does. As a result, despite both methods show problems in finding optimal solutions, the stabilization method seems to be a more erratic algorithm.

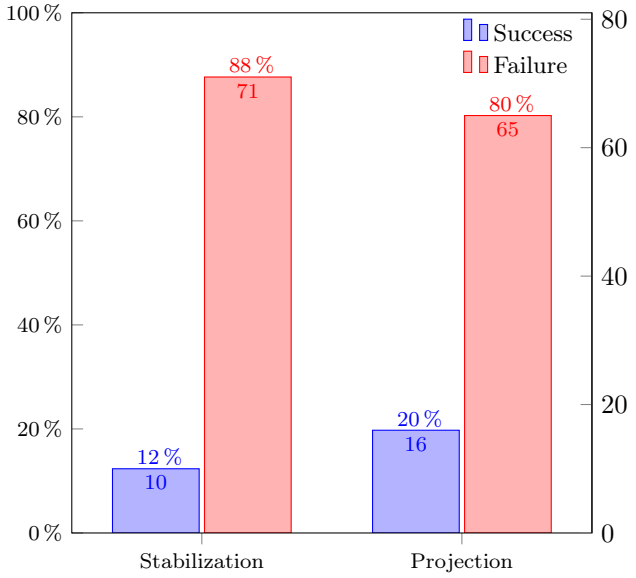
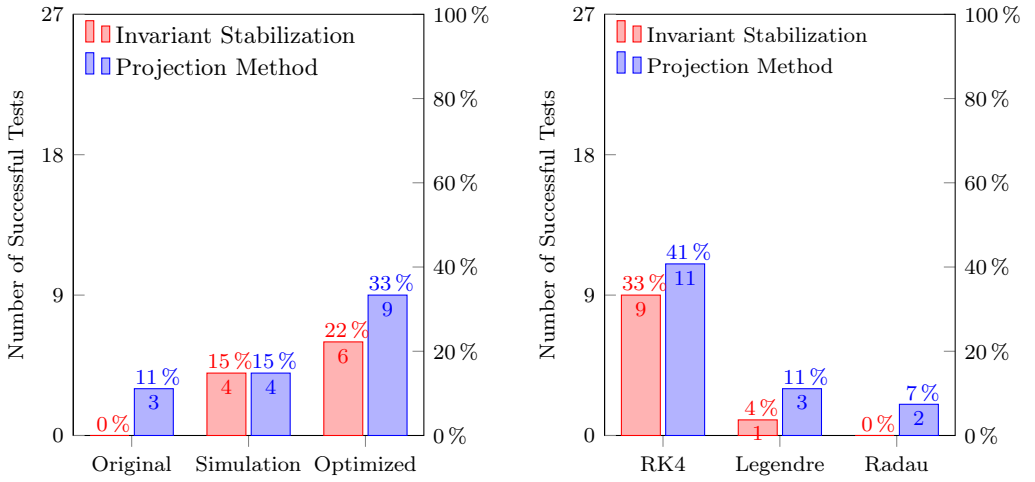


FIGURE 3.3: Performance comparison between the invariant stabilization and the projection method. A total of 81 test per invariant method are performed.

To further comprehend the performance difference between both algorithms, Figure 3.4 illustrates the individual success ratios in the four OCP variations. In particular, in addition to the better performance of the projection method, the following effects can be observed:

(i). By using the stabilization method in combination with the original initial guess, i.e. real flight data, the OCP never finds a solution. It is important to explain that, since the same real flight data was successfully used in [12] as initial guess, the outcome of this experiment might seem contradictory; nevertheless, considering the different safety condition used in [12], it is a perfectly plausible result.

(ii). The system dynamics are not stiff as the RK4 explicit integrator performs good enough.



(A) Success rate for the initial guess.

(B) Success rate for the type of integrator.

(C) Success rate for the wind speed v_w.

(D) Success rate for the number of points N.

FIGURE 3.4: Performance comparison between the invariant stabilization and the projection method. The performance is measured as a function of the success rate on periodic optimal solutions across four different OCP variations: initial guess, integrator type, number of discrete points and wind speed. Each benchmark category, e.g. $v_w = 6\,\mathrm{m/s}$ or N=20, comprises 27 test scenarios.

(iii). The type of integrator seems to be a key factor. In particular, multiple shooting with an explicit RK4 seems to outperform by far the collocation structures. In particular, it is noteworthy that, using RK4, the projection method succeeds more than 40 % of the times.

(iv). As it would be expected, using a nearly optimal trajectory as initial guess is the most successful initialization method.

In the view of the above results, it is clear that both methods depict very low success rates.

After analyzing the different sources of failure, it was concluded that the low performance of the quaternion formulation could be explained due to a combination of three effects:

- The equation of motions and OCP constraints are quite nonlinear; as a result, the solver struggles to find a good feasible local minima. In particular, it has been observed that, in several occasions, the solver fails because it stays in an infeasible point and is not able to compute a step to escape from it.
- It has also been observed that the Hessian of the Lagrangian achieves sometimes eigenvalues close to 0; as a result, the Hessian becomes almost singular, the OCP becomes numerically ill-posed, and as a result, by trying to solve that bad conditioned OCP, the solver demands too much computer memory and the solver crashes.
- Finally, the OCP might also fail because the solver exceeds the maximum number of iterations. This might be again explained due to the OCP nonlinearity.

As a result, we can conclude that, while both methods are not good, the projection method seems to be a more consistent and successful method to avoid LICQ deficiency in the context of periodic OCPs for AWE kites. Furthermore, given the poor performance of the invariant stabilization, we can justify the strange numerical behavior observed in the original approach.

Finally, it is important to remark that, due to the poor performance of both quaternion formulations, in the following section different dynamical models will be proposed and tested.

3.4 Dynamical Model Variations

In order to define the kite position, it is necessary to use a parametrization of the *3D rotation group (SO(3))*, the mathematical group representing all the rotations and orientations in the 3D Euclidean space $\mathbb{R}^3$.

In particular, since given an orthonormal basis of $\mathbb{R}^3$ any rotation can be described by an orthogonal 3×3 matrix R, the SO(3) group and its composition are identified with the group of orthogonal matrices with matrix multiplication. As a result, parametrizations of SO(3) are given by any manifold that can be used to generated the orthogonal 3×3 matrices, where typical example of these manifolds are Euler angles or quaternions.

An important thing to be considered when using Euler angles is that, despite using the minimum number of parameters to define SO(3), they often lead to highly nonlinear and singular dynamics [26]. As a result, their mathematical models are often easy to derive but when embedded in optimization problems they tend to lead to computational issues. Because of the cited problems, [12] argued that a quaternion model, when compared with the original Euler dynamics defined by (1.3), would reduce the nonlinearity and singularities of the kite dynamics and would result in a more convenient OCP formulation.

Nevertheless, considering the poor performance of the quaternion model in the previous section, the above claim might not be entirely true. In particular, we consider that, in order to effectively assess the quality of the different SO(3) parametrization, a further study on the topic is required. As a result, a third model based on rotation matrices will be first discussed, and then, its performance will be compared against the quaternion formulation and the Euler angle model.

3.4.1 Rotation Matrices Model

As suggested by [26], a model based on rotation matrices reduces the nonlinearity of the equations of motion even more than a quaternion formulation; thus, despite increasing the system state size, it is in many cases better suited for optimal control. Based on the above fact, it was concluded that it is essential to derive a model based on them, and then, to test its performance in our application.

Orientation Representation

Before deriving the kite mode, it is necessary to understand the conceptual idea behind rotation matrices. In particular, it is important to consider that, in order to define a system orientation, rotational matrices use a full 3×3 orthogonal matrix:

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}, \quad \text{with} \quad R^\top R = RR^\top = I_3,$$

where each of the columns of R defines the vectors of a moving reference frame attached to the system body.

Moreover, due to the fact that SO(3) is identified with the group of orthogonal matrices, this type of parametrization is also known as natural coordinates [26].

Dynamical Model

Considering the definition of the columns of R, the first natural step to obtain a model in natural coordinates is to select an inertial and a body reference frame for our specific application. In particular, in order to be consistent with the previous work, we decided to use again the definitions given by [12], where the inertial reference frame was defined by $\vec{e}_{\mathrm{x}}$, $\vec{e}_{\mathrm{y}}$, and $\vec{e}_{\mathrm{z}}$, the moving frame described by $\vec{e}_{\mathrm{yaw}}$, $\vec{e}_{\mathrm{pitch}}$ and $\vec{e}_{\mathrm{roll}}$], and where they were both related by a rotation matrix $R(t)$ such that:

$$\begin{aligned} \vec{e}_{\mathrm{yaw}}(t) &= -R(t)\vec{e}_{\mathrm{x}}, \\ \vec{e}_{\mathrm{pitch}}(t) &= -R(t)\vec{e}_{\mathrm{y}}, \\ \vec{e}_{\mathrm{roll}}(t) &= -R(t)\vec{e}_{\mathrm{z}}. \end{aligned}$$

To have a better picture of the above definitions, Figure 3.5 depicts both reference frames for the case of zero rotation.

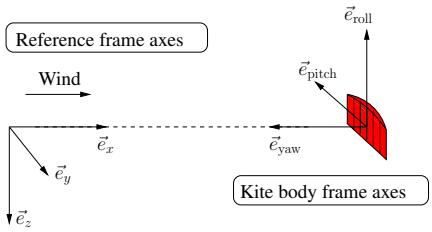


FIGURE 3.5: Reference frame axes for the derivation of the natural coordinates equations of motion. The depicted orientation corresponds to the zero-rotation, i.e. $R = I_3$ [12].

Then, by using the same definitions, the natural coordinates parameterization can be described by:

$$R(t) = \left[\begin{array}{c|c|c} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{array}\right] = \left[\begin{array}{c|c|c} & & \\ -\vec{e}_{\text{yaw}}(t) & -\vec{e}_{\text{pitch}}(t) & -\vec{e}_{\text{roll}}(t) \\ & & \end{array}\right]. \tag{3.14}$$

Once R is selected, the second step is to derive its dynamics. In particular, using (3.14), the equations of motion can be obtained by [26]:

$$\dot{R}(t) = R(t)\omega(t)_\times \tag{3.15}$$

where $\omega(t) \in \mathbb{R}^3$ represents the angular velocities of the column vectors of $R(t)$, i.e. negative angular velocities of the roll, yaw and pitch axis, and $.\times$ a skew operator that transforms a vector in $\mathbb{R}^3$ into the corresponding skew symmetric matrix:

$$\omega_\times = \begin{bmatrix} -\omega_{\text{yaw}} \\ -\omega_{\text{pitch}} \\ -\omega_{\text{roll}} \end{bmatrix}_\times = \begin{bmatrix} 0 & -\omega_{\text{roll}} & \omega_{\text{pitch}} \\ \omega_{\text{roll}} & 0 & -\omega_{\text{yaw}} \\ -\omega_{\text{pitch}} & \omega_{\text{yaw}} & 0 \end{bmatrix}. \tag{3.16}$$

In order to obtain a final expression for the dynamics, the angular velocities ω_{yaw}, ω_{pitch} and ω_{roll} have to be described as a function of $R(t)$. For simplicity reasons, the expressions for ω_{yaw}, ω_{pitch} and ω_{roll} will be directly given; nevertheless, in order to legitimize the model, Appendix A.1 provides their full derivation:

$$\omega_{\text{yaw}} = g_{\text{k}} v_{\text{a}} \delta \tag{3.17a}$$

$$\omega_{\text{pitch}} = -\frac{v_{\text{a}}}{l} + \frac{v_{\text{w}}}{l} R_{13} \tag{3.17b}$$

$$\omega_{\text{roll}} = -\frac{v_{\text{w}}}{l} R_{12}. \tag{3.17c}$$

Finally, by introducing Equations (3.17a)-(3.17c) into (3.16), Equation (3.15) can be expanded to obtain the full set of equations of motion:

$$\begin{bmatrix} \dot{R}_{11} \\ \dot{R}_{21} \\ \dot{R}_{31} \\ \dot{R}_{12} \\ \dot{R}_{22} \\ \dot{R}_{32} \\ \dot{R}_{13} \\ \dot{R}_{23} \\ \dot{R}_{33} \end{bmatrix} = \frac{v_{\text{w}}}{l} R_{12} \begin{bmatrix} -R_{12} \\ -R_{22} \\ -R_{32} \\ R_{11} \\ R_{21} \\ R_{31} \\ 0 \\ 0 \\ 0 \end{bmatrix} + g_{\text{k}} v_{\text{a}} \delta \begin{bmatrix} 0 \\ 0 \\ 0 \\ R_{13} \\ R_{23} \\ R_{33} \\ -R_{12} \\ -R_{22} \\ -R_{32} \end{bmatrix} + \frac{v_{\text{w}} R_{13} - v_{\text{a}}}{l} \begin{bmatrix} -R_{13} \\ -R_{23} \\ -R_{33} \\ 0 \\ 0 \\ 0 \\ R_{11} \\ R_{21} \\ R_{31} \end{bmatrix}, \tag{3.18}$$

with air path speed:

$$v_{\text{a}} = E v_{\text{w}} R_{11} - E\dot{l}. \tag{3.19}$$

System Constraints

In order to obtain an equivalent formulation of the system constraints, the relation between the nine elements of R and the the four quaternions will be used. However, in order to keep this section legible, the required relations for deriving the constraints will be directly used; nevertheless, should the total set of relations or its derivation be required, Appendix A.2 can be consulted.

In order to model the altitude safety constraint equivalent to Equation (3.12), the relation $R_{31} = 2(q_1 q_3 - q_0 q_2)$ can be used to obtain:

$$-lR_{31} \geq h_{\min}.$$

In the case of the constraint (3.4) ensuring that the kite is always tethered, an equivalent version can be derived using the equation $R_{11} = (q_0^2 + q_1^2 - q_2^2 - q_3^2)$:

$$v_{\mathrm{w}} E R_{11} - E\dot{l} \geq v_{\mathrm{a,min}}.$$

For the topology constraints defined by (3.9), a natural coordinates version can be obtained by considering $R_{12} = -2(q_0 q_3 - q_1 q_2)$:

$$\begin{aligned} \frac{-R_{12}}{2} &\geq 0, \quad \text{for} \;\; t_{2i-2} \leq t \leq t_{2i-1} \;\; \text{and} \;\; i = 1, \ldots, (n-1)/2, \\ \frac{-R_{12}}{2} &\leq 0, \quad \text{for} \;\; t_{2i-1} \leq t \leq t_{2i} \quad\;\; \text{and} \;\; i = 1, \ldots, (n-1)/2. \end{aligned} \tag{3.20}$$

Finally, to impose the $R(0) = R(T)$ periodicity constraint, the orthogonality of $R(t)$ is regarded to set:

$$R(0)^\top R(T) - I_3 = 0. \tag{3.21}$$

LICQ Deficiency

As with the quaternion formulation, the system dynamics have also the matrix orthogonality invariant:

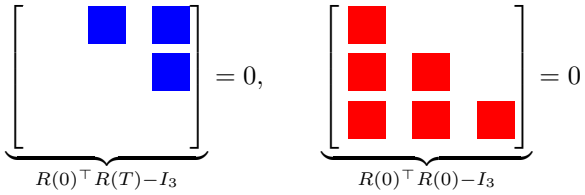
$$\mathcal{I}\big(R(t)\big) = R(t)^\top R(t) = I_3.$$

Furthermore, in order to ensure that the above equation is fulfilled an orthogonality constraint at some time point has to be added; in particular, a usual convention is to impose it in the initial point:

$$R(0)^\top R(0) - I_3 = 0. \tag{3.22}$$

As a result, considering that Equation (3.22) together with (3.21) create a set of 18 constraints on the initial state and that the $R(t) \in \mathbb{R}^{3\times 3}$, from Theorem 3.2 it follows that LICQ does not hold.

A known methodology to avoid LICQ deficiency, while still imposing the periodicity and orthogonality condition, is to select three matrix elements from (3.21) and six from (3.22); in particular, imposing the three upper triangular elements from (3.21) and the six lower triangular elements from (3.22) is a common choice [27, 24]:



3.4.2 Euler Angles

Despite leading to very non-linear and singular dynamics, Euler angles still have the advantage of representing the smallest SO(3) parametrization; consequently, the number of variables on an OCP formulation is reduced, and in turn, the complexity of the problem does not necessarily have to get worse. Moreover, due to this minimal number of states, the dynamics have no invariant and the periodicity conditions do not compromise LICQ; therefore, even with a more complex model, the lack of LICQ deficiency might lead to easier numerical computations.

Taking into account the described advantages, it is obvious the interest and necessity of including Euler angles in the SO(3) comparison.

OCP Dynamics

To model the equations of motion, the same dynamical model given by Equation (1.3) is used:

$$\begin{aligned} \dot{\psi} &= g_\mathrm{k} v_\mathrm{a} \delta + \dot{\varphi} \cos\vartheta, \\ \dot{\varphi} &= -\frac{v_\mathrm{a}}{l \sin\vartheta} \sin\psi, \\ \dot{\vartheta} &= -\frac{v_\mathrm{w}}{l} \sin\vartheta + \frac{v_\mathrm{a}}{l} \cos\psi, \\ \dot{l} &= v_\mathrm{winch}, \end{aligned} \tag{3.23}$$

with:

$$v_\mathrm{a} = v_\mathrm{w} E \cos\vartheta - \dot{l} E.$$

OCP Constraints

In order to find the equivalent set of constraints, the relations between the quaternion and the Euler angles will be used. Moreover, to keep the section legible, the required relations will be directly stated and their derivations are left for Appendix A.2.

Considering that $-\sin\vartheta\cos\varphi = 2(q_1 q_3 - q_0 q_2)$, the altitude safety constraint given by Equation (3.12) is equivalent to:

$$l \sin\vartheta \cos\varphi \geq h_\mathrm{min}.$$

In the case of (3.4), an equivalent formulation can be obtained by regarding that $\cos\vartheta = (q_0^2 + q_1^2 - q_2^2 - q_3^2)$:

$$v_\mathrm{w} E \cos\vartheta - E\dot{l} \geq v_\mathrm{a,min}.$$

For the topology constraint, an equivalent formulation was already given by (3.8):

$$\dot{\varphi} \leq 0, \quad \text{for} \quad t_{2i-2} \leq t \leq t_{2i-1} \quad \text{and} \quad i = 1, \ldots, (\mathrm{n}-1)/2,$$
$$\dot{\varphi} \geq 0, \quad \text{for} \quad t_{2i-1} \leq t \leq t_{2i} \quad \text{and} \quad i = 1, \ldots, (\mathrm{n}-1)/2.$$

Finally, $w(0) = w(T)$, with $w = [\psi,\ \varphi,\ \vartheta]^\top$, has to be added to the OCP as the periodicity constraint.

3.4.3 Linearity Comparison

Considering that the state dimension and the OCP linearity will determine the NLP performance, it is important that, before understanding and explaining the experimental results, a comparison between the linearity of Euler angles, quaternions, and natural coordinates is depicted; in particular, the equations of motion as well as the OCP constraints of the three formulations will be contrasted.

Dynamics

By comparing the three sets of equations of motion, the following conclusions can be drawn:

(i). The Euler dynamics, which are described by Equation (3.23), display the highest nonlinear structure; particularly, they include a singularity at $\vartheta = 0$ as well as several trigonometric functions that create complex equations of motion.

(ii). The natural coordinates formulation, which is represented by Equation (3.18), represents the least nonlinear dynamics; in particular, it avoids singularities and leads to uncomplicated quadratic equations of motion.

(iii). Finally, the quaternion formulation as given by Equation (3.2) display better linearity than Euler angles but less than natural coordinates; specifically, it avoids singularities but forms a set of cubic equations of motion.

Constraints

In order to have a clearer picture on the linear properties of the constraints, a full comparison on the path constraints is depicted in Table 3.2

Table 3.2: Linearity comparison between the constraints of the three studied SO(3) parametrization: Euler angles, quaternions and natural coordinates.

	Euler angles	Quaternions	Natural coordinates
Safety altitude	$l\cos\varphi\sin\vartheta$	$-2l\big(q_1q_3 - q_0q_2\big)$	$-lR_{31}$
Minimum v_a	$v_\mathrm{w}E\cos\vartheta - E\dot{l}$	$v_\mathrm{w}E\big(q_0^2 + q_1^2 - q_2^2 - q_3^2\big) - E\dot{l}$	$v_\mathrm{w}ER_{11} - E\dot{l}$
Topology	$\dot{\varphi}$ | $\sin\psi$	$q_0q_3 - q_1q_2$	$-R_{12}$

From analyzing the table, two facts can be observed:

(i). Natural coordinates represent the only linear constraint formulation.

(ii). An assessment between Euler angles and quaternion is more complicated: while quaternions lead to quadratic constraints for the three scenarios, Euler angles involve trigonometric relations in two cases but a linear constraint in the third one.

It is important to outline that, since periodicity constraints only bound a single time point, they are not expected (as long as LICQ holds) to modify the end solution as much as path constraints; as a result, their comparison was omitted in this section.

Interpretation

Regarding the previous analysis, it is possible that, when solving the OCPs, the following effects might appear:

(i). Considering their high state dimension, natural coordinates might show slower convergence than the other two methods; nevertheless, due to their linear structure, it is also likely that they will form a more robust environment: the solver might struggle less to find the optimal value and solutions might represent more efficient trajectories.

(ii). By contrast, Euler angles should provide a fast convergence model due to its low dimensional representation; nevertheless, due to the high nonlinearity, it is possible that the solver does not find a solution or, even if it does, that the solution represents less efficient optimal trajectories.

(iii). Since quaternions represent linear conditions and a state dimension that are in between Euler angles and natural coordinates, its performance should be an average of the other two. However, as it has been shown in Section 3.3, their performance is slightly poor; therefore, their results might be worse than expected.

3.4.4 Results

In order to obtain a complete evaluation of the three dynamical models, they will be tested against several OCPs variations. In particular, similarly to Section 3.3, the following problem modifications will be regarded:

(i). To test the model behavior for different initial guesses, the OCPs will consider three different initializations: the standard real flight data used in the original approach, an optimal trajectory of a slightly different OCP, and a trajectory obtained by a simulation routine.

(ii). In order to study the stiffness of the three models, three different direct methods will be used: multiple shooting with an explicit RK4 integrator, a direct collocation using a Radau scheme, and direct collocation but with a Legendre scheme.

(iii). To judge the effect of the wind speed, the OCPs will be solved for three different wind velocities: 6, 10 and 15 m/s.

(iv). To measure the effect of the problem size, two number of discrete points will be regarded: the original N=250 and N=500.

(v). Since the topology constraints were claimed to be an important part of the quaternion model, the three parameterizations will be tested with and without topology constraints in order to examine if they are indeed necessary. Nevertheless, since the results of this experiment are more extensive and independent from the type of model, its discussion and results will be left for the next section, together with a study on flight topologies.

Considering every possible combination of the above variations, a total of 332 tests (108 per dynamical model) have to be performed. A first summary of these 332 experiments is depicted in Figure 3.6; in particular, by only comparing the number of tests that are successfully solved, Euler angles seem to be the most favorable model implementation; specifically, they are able to solve two times as many cases as rotation matrices and approximately four times as many problems as quaternions.

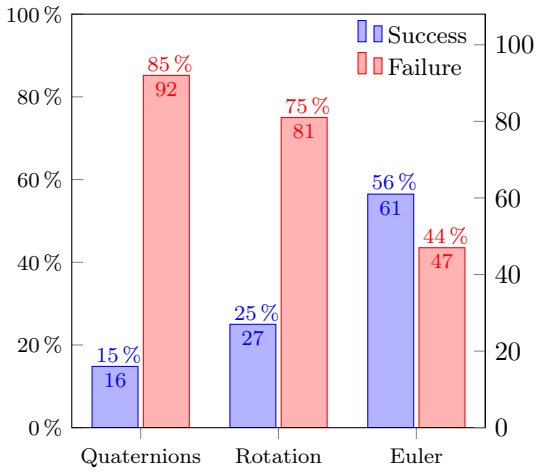


FIGURE 3.6: Performance comparison between the three different dynamical models. Each model is tested in 108 different scenarios.

It is important to remark that Figure 3.6 is just a brief compilation of all the results; therefore, to have a better understanding between the three models, the coming sections will expand the analysis.

Individual Success Ratios

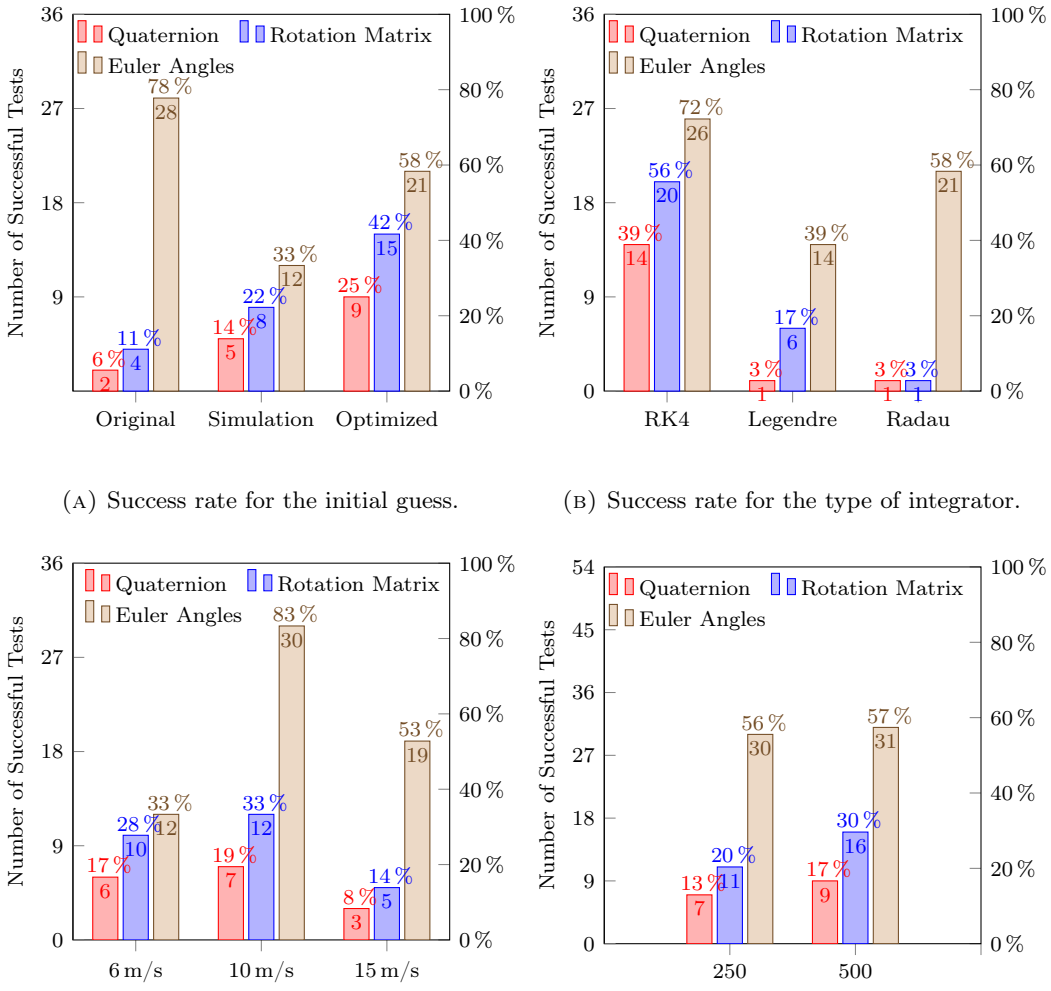
To further comprehend the performance difference between the three models, Figure 3.7 illustrates the individual success ratios considering four OCP variations. In addition to the better performance of the Euler model, the following effects can be observed:

(i). None of the system dynamics are stiff as all of them perform remarkably good with the RK4 explicit integrator.

(ii). The type of discretization method seems to be important for quaternions and rotation matrices, where multiple shooting with the explicit RK4 outperforms the collocation structures.

(iii). It was expected that the three models would show the best performance when the initial guess was a semi-optimal trajectory. Nevertheless, despite being true for quaternions and rotations matrices, Euler angles seem to perform even better when the OCP is initialized with real data.

(iv). The OCPs are easier to solve when v_{w}=10 m/s; this is quite noticeable for Euler angles. A possible explanation for this effect is the fact that the simulated trajectory and the real trajectory used as initial guesses were obtained with wind profiles whose average was close to 10 m/s.



(A) Success rate for the initial guess.

(B) Success rate for the type of integrator.

(C) Success rate for the wind speed v_w.

(D) Success rate for the number of points N.

FIGURE 3.7: Performance comparison between the three dynamical models. The performance is measured as a function of the success rate on periodic optimal solutions across four different OCP variations: initial guess, integrator type, number of discrete points and wind speed.

(v). The performance seems to be independent of the number of discrete points.

(vi). In every variation the same trend can be observed: Euler angles have the best performance, rotation matrices show an intermediate performance, and quaternions seem to represent the worst model.

Loyd Factor Versus Converge Rate

In the past sections, the success ratios of the three dynamical models were compared. Nevertheless, a model might solve a large number of tests by obtaining poor local minima, i.e. it might find optimal solutions more often but these solutions might represent trajectories with

lower power efficiencies. As a result, the success ratio alone does not provide a fair indicator of the model's performance and some other metrics must be considered.

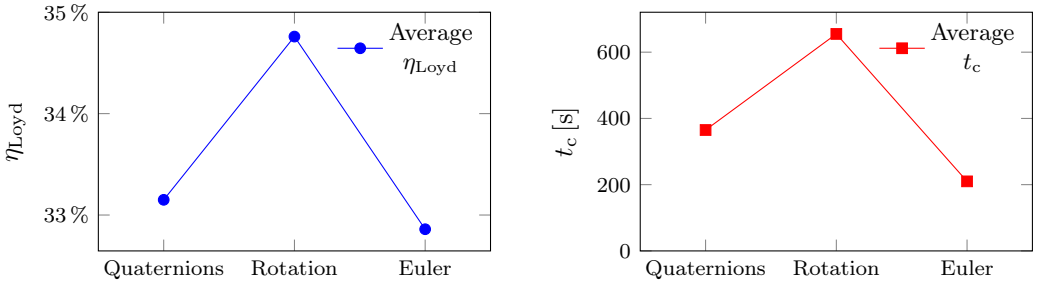
In addition, not only might the power efficiency differ from the success ratio, but also the average computation time is also critical: a method might solve many tests but at a high computational cost; therefore, it is possible that, despite solving many scenarios, a model might not be computationally tractable.

As a result, a necessary indicator to assess the goodness of the models is to compare the average efficiency factor η_{Loyd} and the average computational time t_c in those cases where the models obtain an optimal solution. The results of this comparison are depicted in Figure 3.8; by analyzing the figure, several conclusions can be drawn:

(i). Euler angles seem to be the computationally cheapest method but at the same time the one with the worst average efficiency.

(ii). Quaternions, when compared to Euler angles, depict a very similar efficiency factor but a higher computation time.

(iii). In contrast with the other two, rotation matrices are able to obtain better optimal solutions; in particular, the average η_{Loyd} is 2 % larger than the other two scenarios. However, this improvement in efficiency comes at a cost of computation time; specifically, natural coordinates require, in average, 2 to 3 times as much time as the other two models in order to solve an OCP.



(A) η_{Loyd} factor comparison.

(B) Computational time t_c comparison.

FIGURE 3.8: η_{Loyd} factor and computation time comparison between the three different dynamical models. Each model is tested in 108 different scenarios.

The fact that one of the three models is able to extract wind power with such a broad margin is quite interesting; therefore, it is necessary to understand why the natural coordinates formulation outperforms the other two by such a large difference.

A logical answer can be found in the linearity comparison of Section 3.4.3; in particular, considering that an OCP using natural coordinates had a more linear structure when compared to Euler angles or quaternions, it is plausible that, due to the more convenient NLP structure, the solver is able to find better local minima.

The above explanation is reinforced when the following observation is considered: the OCP formulations in Euler angles and quaternions were detected to find local minima that had the same number of lemniscates as the initial guess had. By contrast, natural coordinates formulations, due to its more linear structure, were found to expand the number of lemniscates in order to increase the time the kite is in crosswind motion and in turn improve the power efficiency.

This specific effect can be seen in Figure 3.9 representing the solver iterations of a natural coordinates formulation; specifically, it can be seen how, along the iteration process, the solver generates an extra lemniscate and η_{Loyd} increases.

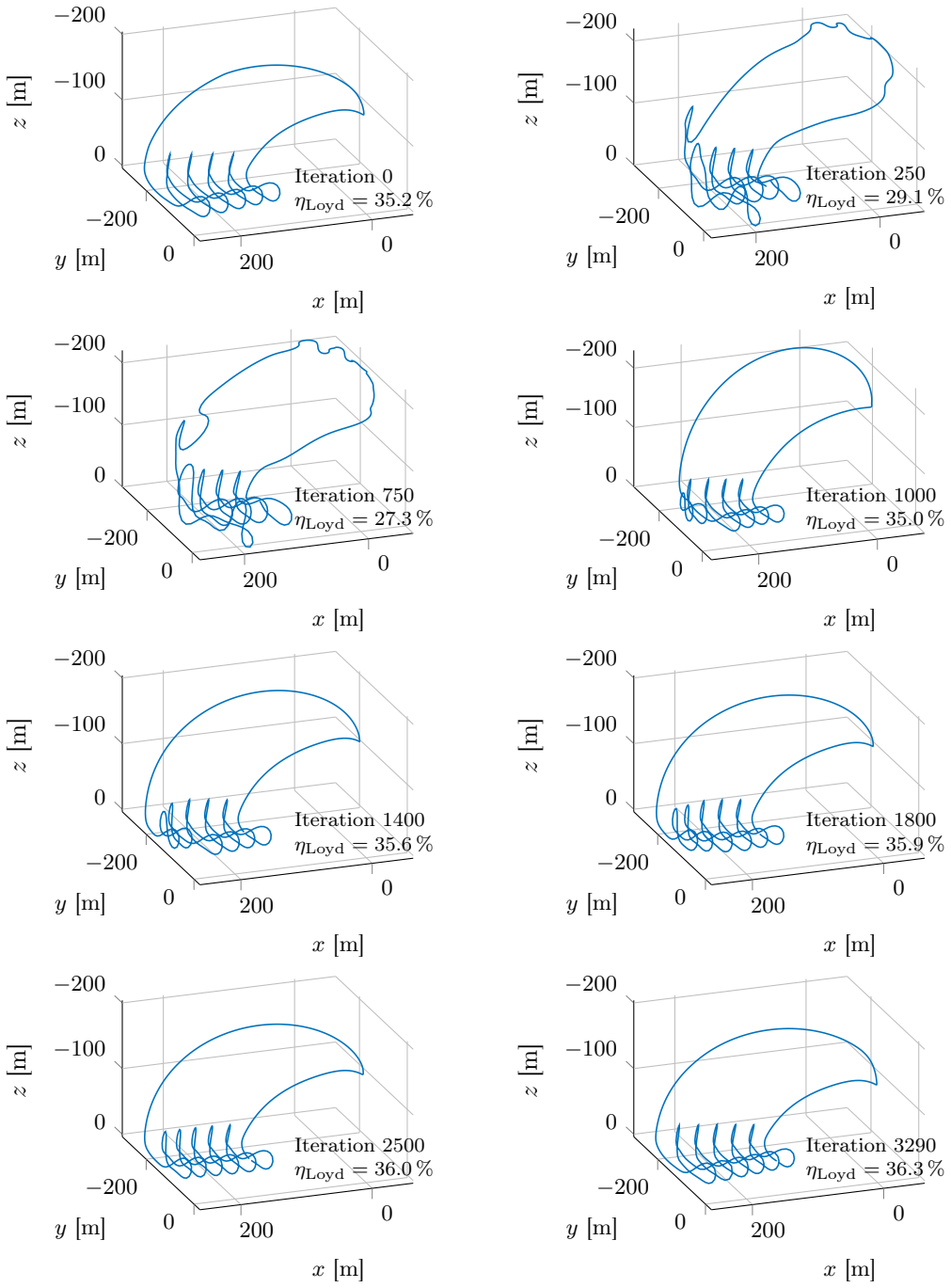


FIGURE 3.9: Generation of an extra lemniscate by using a natural coordinates formulation.

Results Analysis

Considering the above results, it can be concluded that quaternions are the worst approach with no apparent advantages; in particular, they only solve 16 % of the cases with a power efficiency similar to Euler angles, which, in comparison, are able to solve 56 % of the scenarios and in shorter times.

By contrast, Euler angles are the fastest model with computation times that outperform the other two; moreover, they represent the most successful implementation with more than half of the performed tests being solved. Nevertheless, they have the drawback that, when compared to rotation matrices, their optimal solutions represent trajectories with lower power efficiencies.

As a result, while comparing the relative quaternion performance is easy, an assessment between Euler angles and natural coordinates is harder to obtain; in particular, while the former solves a higher number of cases and it does it much faster, rotation matrices have been proven to obtain better solutions. As a consequence, Euler angles is the best candidate when many scenarios have to be solved or when a first rapid solution has to be obtained; however, after a fast and good solution is obtained with Euler angles, rotation matrices might be preferred to improve the power efficiency.

Finally, before ending this section, it is necessary to explain the failure mechanisms of the three formulations described. In particular, as already explained at the end of Section 3.3.6, quaternion breakdowns are due to three effects:

- The major source of failure is the OCP nonlinearity that leads to a solver that struggles to find a feasible local minima. In particular, the solver reaches an infeasible point and is unable to escape from it.
- As a second effect, the Hessian of the Lagrangian might become nearly singular leading to a numerically ill-posed OCP; as a consequence, by trying to solve a bad conditioned OCP, the solver demands too much computer memory and the solver crashes. This issue represents 15 % of the breakdowns.
- Finally, 15 % of the times, the OCP fails because the solver exceeds the maximum number of iterations. This is likely to be caused due to the OCP nonlinearity.

In the case of Euler angles, 95 % of the solver breakdowns are due to the nonlinear condition of the OCP. In particular, due to the trigonometric and nonlinear structure of the constraints and dynamics, the solver fails because it reaches an infeasible point from where it is unable to escape. It is important to remark that the other two sources of quaternions failure, i.e. exceeding the maximum number of iterations and memory issues, are barely observed in the Euler formulations.

Finally, in the case of natural coordinates, two are the main responsible mechanisms of failure:

- 60 % of the problems are caused by exceeding the maximum number of iterations. However, unlike quaternions, this issue is due to the large OCP size.
- 35 % of the solver miscarriages are due to an infeasible problem from which the solver is not able to escape. It can be observed that, due the more linear structure, this type of failure occurs less often than in the other two formulations.
- Finally, memory problems represented 5 % of the total failures.

Given the above explanations and resolutions, the high success ratio of the Euler angles formulation can be explained considering that it has no issues with ill-posed OCPs nor exceeding the maximum number of iterations. In particular, since it has a single source of failure, it is more likely to succeed.

In a similar fashion, considering that 60 % of the natural coordinates failures are due to an excessive number of iterations, their lower success ratios with respect to Euler angles can be clarified by the larger problem size of the former formulation.

Finally, in the case of quaternions, besides the two issues that the other formulations have, their formulation might also become ill-posed; and as a result, it is understandable why they are the least successful implementation.

3.5 Flight Topologies

In the original formulation, the kite was forced to fly lemniscates for security reasons; in particular, due to concerns of twisting the tether, lemniscates (eight shapes) were used to ensure that the angle ψ did not increase unbounded and that stresses on the tether were not built up [12]. In that particular scenario, a set of topology constraints was used to control the kite direction and ensure the desired lemniscate shape.

Considering the above facts, this section will explore two different lines of work. On the one hand, the importance of the topology constraints will be discussed; in particular, their performance will be study in order to assess if they are indeed required. On the other hand, a different flight topology will be proposed; specifically, instead of lemniscates, the optimal trajectory will be modeled by distorted circular shapes (similar to ovals) that keep the angle ψ bounded.

3.5.1 Topology Constraints

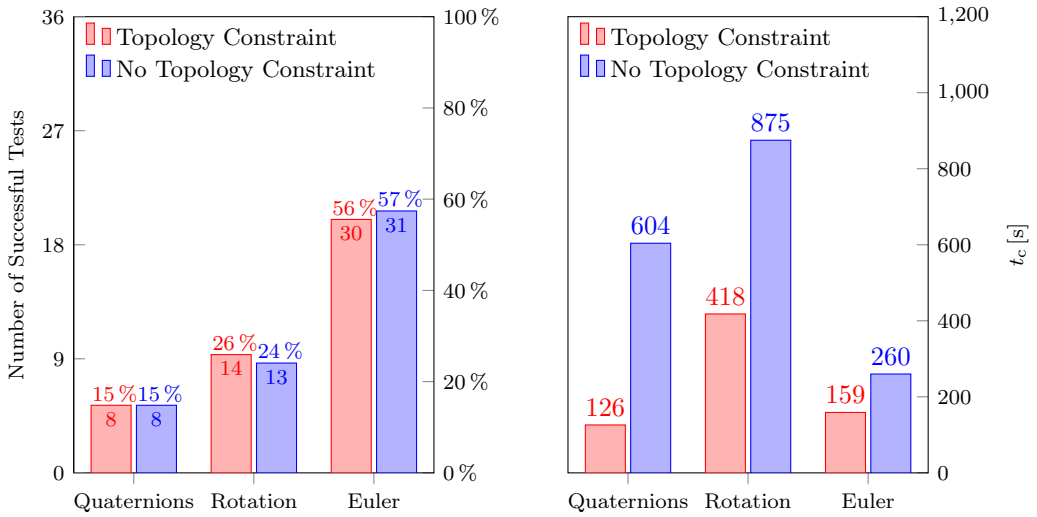
As a part of the dynamical model comparison in Section 3.4, the test scenarios included the addition or elimination of the topology constraints. The intention behind that option was to assess whether the topology constraints could indeed improve the OCP performance. In particular, since the original topology constraints did not impose the lemniscate topology, but instead, only forced the kite to sequentially fly from the right to the left, the real utility of these constraints was questioned.

The main theory was that, even without the constraints, the kite should still perform a similar motion; in particular, in order to extract the largest amount of wind power, the kite should stay in crosswind motion; as a result, the kite is expected to alternate directions in a cyclic manner so that the flight topology is symmetric with respect to the wind direction. Considering these hypothesis, the OCP might obtain a trajectory that replicates lemniscates by simply maximizing the extracted power.

In order to assess the benefits of the topology constraints, Figure 3.10 illustrates a comparison of the success ratios and the computation times between OCPs that include topology constraints and OCPs that do not.

In general, it can be observed that, in contrast with the assumptions of [12], topology constraints are not necessary to solve the OCP and obtain the desired fly topology; in particular, Figure 3.10a shows that the number of solved OCP is independent from the considered constraints.

Nevertheless, as it can be seen in Figure 3.10b, they do improve the OCP performance by reducing the computation time; in particular, by imposing in the OCP the optimal fly



(A) Success ratio.

(B) Average computation time.

FIGURE 3.10: Success ratio and computation time comparison between OCPs that differ on having topology constraints. Each category, e.g. Euler with topology, comprises 54 test scenarios.

directions, the solver avoids many non-optimal trajectories which would be feasible if no topology constraints were present.

To further illustrate this effect, Figure 3.11 compares the solver iterations of two OCPs that only differ in the topology constraints. Specifically, both OCPs model the system with a natural coordinates formulation, using a multiple shooting approach with RK4 and N=500 discrete points, considering a wind speed v_w=10 m/s and using the simulated flight data as initial guess. From the figure, it can be observed that, regardless topology constraints, the OCPs find the same optimal solution; however, the OCP that includes the constraints has a faster converge rate.

In behalf of the above results, it can be concluded that, in contrast with the hypothesis of [12], topology constraints are not required to impose the desired lemniscate topology. In particular, they might help to achieve faster convergences but they have no influence in the flight topology.

3.5.2 Circular Trajectories

Unlike lemniscates, in order to fly circles, the kite has to continuously rotate around its $\vec{e}_{\text{yaw}}$; as a consequence, if the rotation is kept in the same direction, the ψ angle grows without control and the tether might get twisted.

A possible solution to circumvent this issue is to use a double cycle optimal trajectory. The fundamental idea is that, since ψ has to be bounded by a reasonable value, the kite can fly a first periodic cycle rotating clockwise (counterclockwise) around its $\vec{e}_{\text{yaw}}$ followed by a second periodic cycle where it rotates in the opposite direction; as a result, at the end of the double periodic cycle, ψ returns to the initial value and twisting is avoided.

In order to generate a trajectory with the described topology, the parameterization of SO(3) should be first chosen; in particular, considering the outcome of Section 3.4, Euler angles are the best choice.

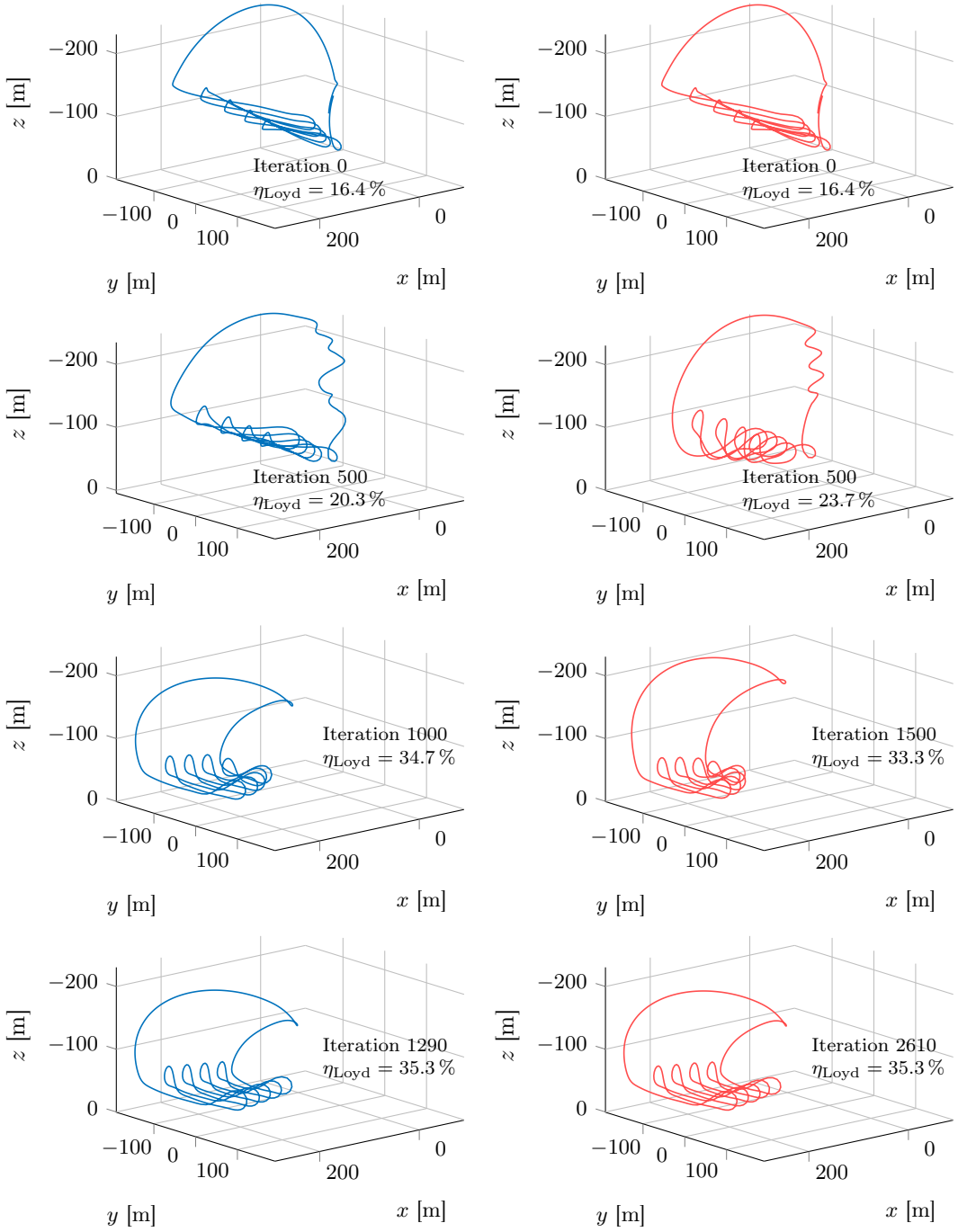


FIGURE 3.11: Solver iteration comparison for the same OCP with and without topology constraints. Left: topology constraints are used. Right: topology constraints are not considered.

Topology Constraints

In the previous section, it was shown that, using the right initial guess, topology constraints are not required to obtain lemniscates. In particular, it was proven that, since the topology constraints only imposed the flight direction, they do not influence the final flight topology.

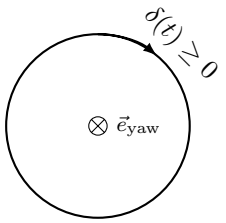
In contrast with this original scenario, it has been observed that, unless the direction of rotation around the $\vec{e}_{\text{yaw}}$ axis is enforced, the optimal solution does not achieve a circular topology. As a result, a set of constraints imposing the kite rotational direction has to be included. In particular, dividing the time period T into a time grid $t_0 < \ldots < t_4$ of 5 points, where $t_0 = 0$ and $t_4 = T$, a constraint on the kite rotational direction (clockwise or counterclockwise) should be ideally imposed by:

$$\begin{aligned} \omega_{\text{yaw}}(t) \geq 0, \quad \text{for} \quad t_0 \leq t \leq t_1 \\ \omega_{\text{yaw}}(t) \leq 0, \quad \text{for} \quad t_2 \leq t \leq t_3. \end{aligned} \tag{3.24}$$

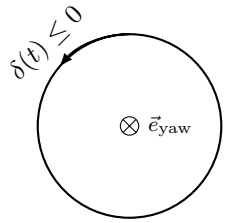
However, since ω_{yaw} is not a system state, an alternative expression must be regarded. Considering that $\omega_{\text{yaw}} = g_{\text{k}} v_{\text{a}} \delta$ and that $v_{\text{a}} > 0$, the above topology constraint is equivalent to:

$$\begin{aligned} \delta(t) \geq 0, \quad \text{for} \quad t_0 \leq t \leq t_1 \\ \delta(t) \leq 0, \quad \text{for} \quad t_2 \leq t \leq t_3. \end{aligned} \tag{3.25}$$

To have a graphical illustration, Figure 3.12 depicts the above two constraints.



(A) Clockwise rotation for first periodic cycle..



(B) Counterclockwise rotation for second periodic cycle.

FIGURE 3.12: Topology constraints for generation of circular trajectories. By enforcing the value of the control δ the rotational direction of the kite around its $\vec{e}_{\text{yaw}}$ axis can be controlled.

It is important to remark that, since the first periodic cycle is defined in the time interval $[t_0, t_2]$ and the second cycle in $[t_2, t_4]$, the above constraints only impose the required rotational orientation in the two power phases of the periodic cycles. In particular, the intervals $[t_1, t_2]$ and $[t_3, t_4]$ represent the retraction phases and they should be topology free.

Periodicity Constraints

As with the original problem, it is important to impose a periodicity constraint so that the optimal solution is a closed loop:

$$\begin{bmatrix} \psi(0) \\ \varphi(0) \\ \vartheta(0) \\ l(0) \end{bmatrix} = \begin{bmatrix} \psi(T) \\ \varphi(T) \\ \vartheta(T) \\ l(T) \end{bmatrix}. \tag{3.26}$$

Furthermore, since the circular trajectory is modeled by two periodic cycles, an extra periodicity constraint in the middle of the trajectory is also required:

$$\begin{bmatrix} \varphi(0) \\ \vartheta(0) \\ l(0) \end{bmatrix} = \begin{bmatrix} \varphi(T/2) \\ \vartheta(T/2) \\ l(T/2) \end{bmatrix}. \tag{3.27}$$

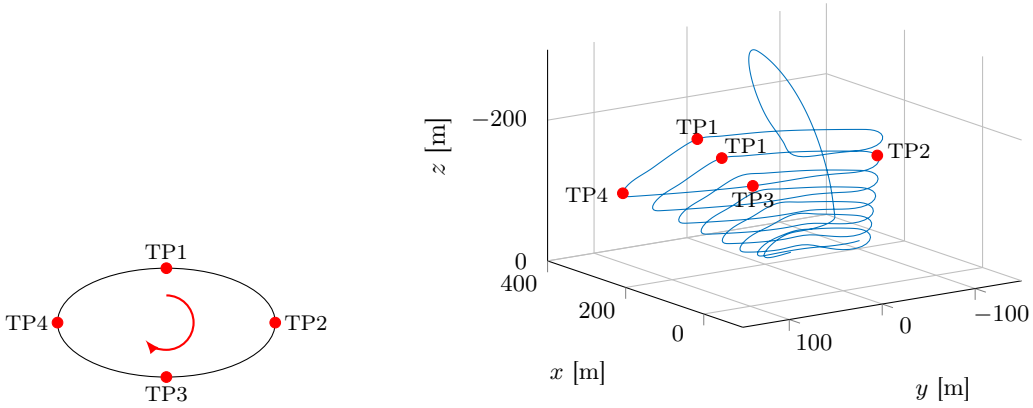
It is important to note that (3.26) in couple with (3.27) lead to a dual periodicity condition on the kite location at the beginning and end of both cycles; however, since ψ increases in the first cycle and unwinds during the second, the kite orientation is only one time periodical.

Initial Guess

One of the main problems faced when generating the circular trajectories was the lack of a decent OCP initial guess. In order to elude this problem, a framework for generating feasible trajectories was created; in particular, by using a kite simulator in couple with an implementation of the classical target point controller [14], an environment to obtain acceptable initial guesses was modeled.

The fundamental principle was to use a four target points sequence with the four points separated by 90 ° and representing an elliptical shape. This method was based on the approach represented by Figure 1.7, where a two target points strategy was used to generate lemniscates.

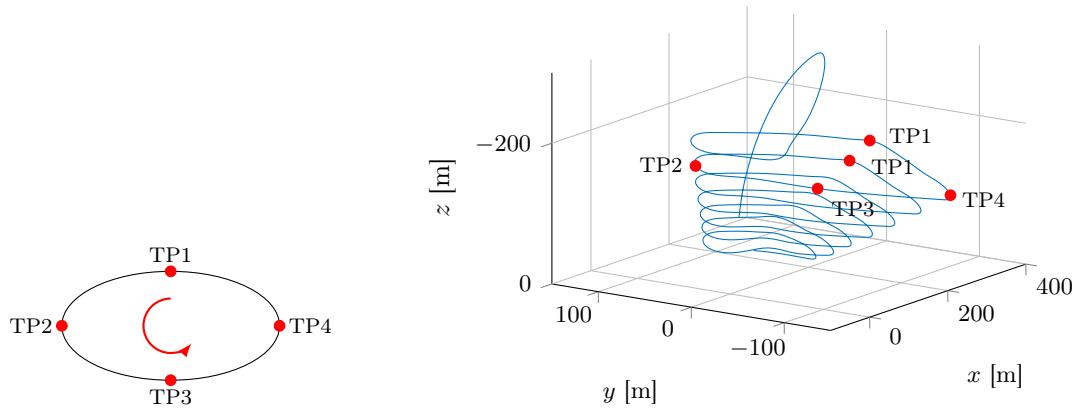
In order to provide a graphical representation, Figures 3.13 and 3.14 illustrate the generation of the two consecutive periodic cycles; in particular, the left subfigures depict the target point sequence used for the controller and the right ones the resulting simulated trajectory. It is important to note that, as observed in Figure 3.14, the sequence of target points has to be inverted at the end of the first period.



(A) Target point structure that the classical controller uses for trajectory generation.

(B) Resulting trajectory after using the classical controller with a four target points strategy.

FIGURE 3.13: Generation of the feasible trajectory for the clockwise half period of a full circular topology.



(A) Target point structure for trajectory generation.

(B) Resulting generated trajectory.

FIGURE 3.14: Generation of the feasible trajectory for the counterclockwise period.

Results

After replacing the topology constraint and the initial guess, the OCP was again solved. The results can be observed in Figure 3.15 representing the optimal trajectory at the end of the optimization process, and in Figure 3.16, which depicts the kite trajectory during eight solver iterations. It can be observed how the solver finds a very good solution after 350 iterations; however, it requires 2700 more iterations to fully optimize the result.

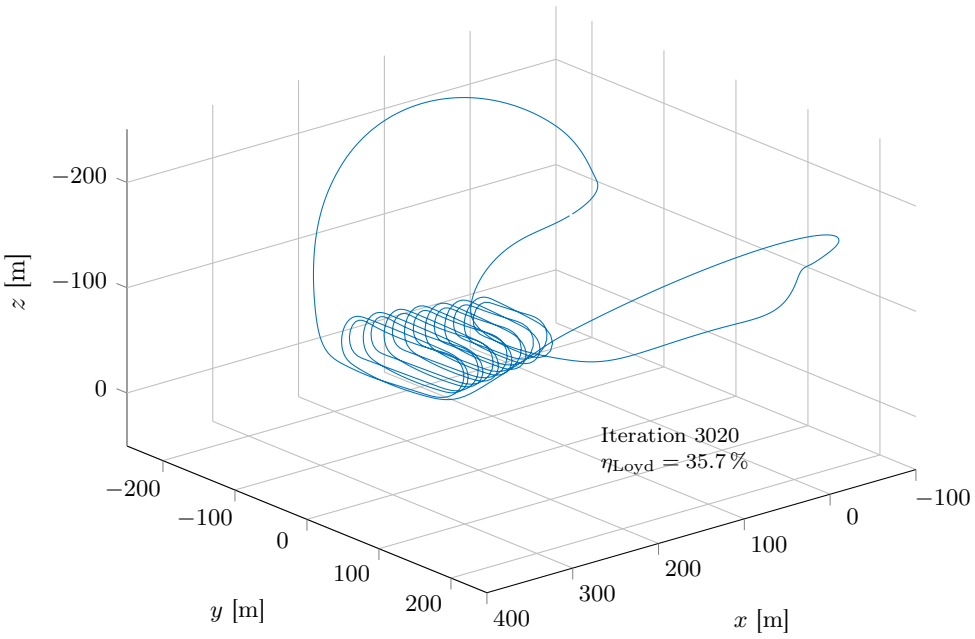


FIGURE 3.15: Fully optimized circular trajectory after 3020 iterations.

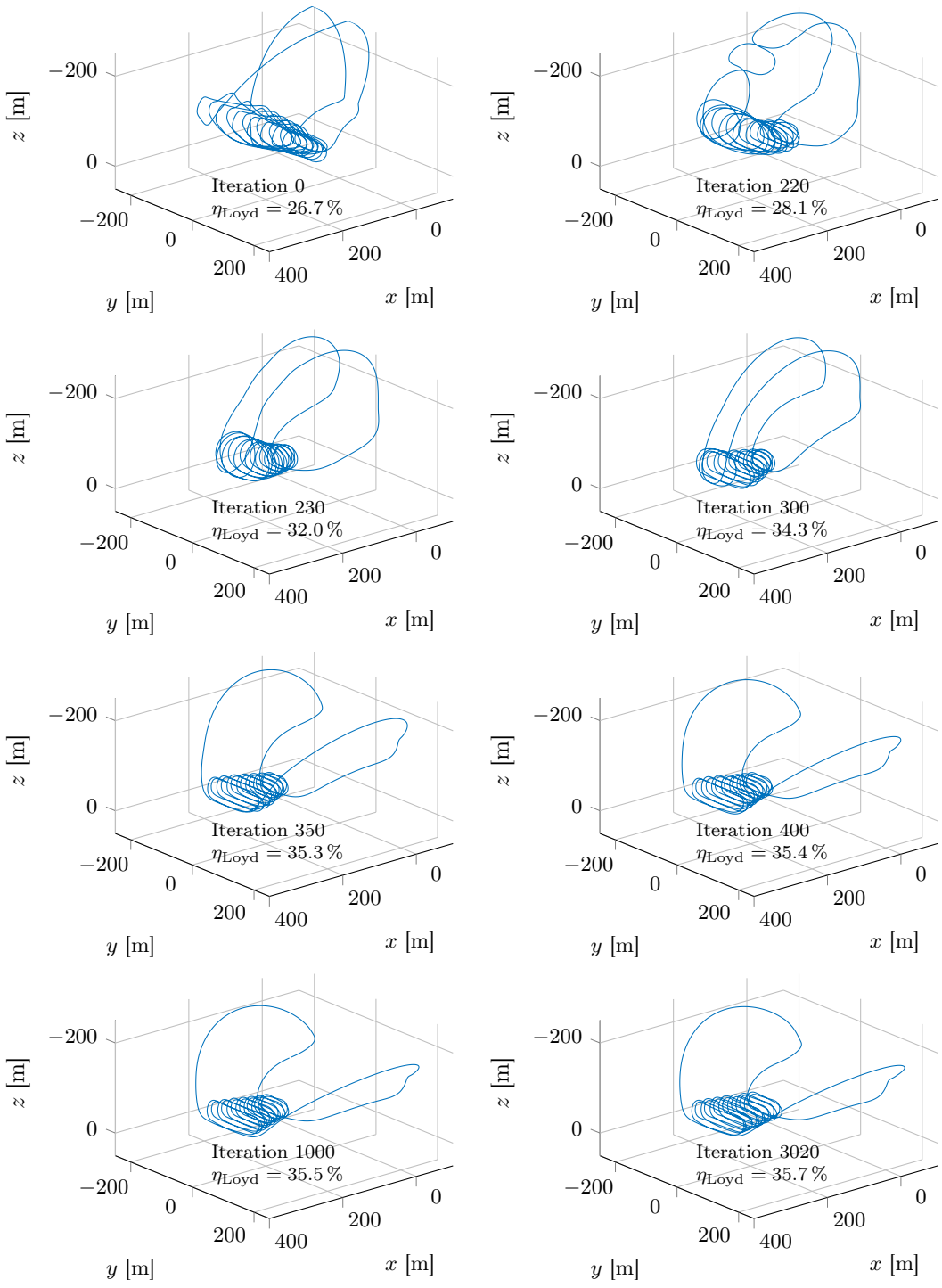


FIGURE 3.16: Solvers trajectory iterations when obtaining circular trajectories.

It is necessary to note that, by flying circular trajectories, the power efficiency achieves similar values as in lemniscate topologies, and as a result, the η_{Loyd} is still high enough to double the original 18 % efficiency of the target point controller.

Finally, it is important to make a comment on the flight topology: as observed from the results, an oval-elliptical topology seems to be more optimal than a flying circles; this result is completely expectable as the kite will always tend to fly as closer to the ground as possible to increase the air path speed v_{a} and in turn the extracted power.

3.6 Conclusion

As a result of the research conducted, several contributions to the field of AWE and periodic OCPs were made.

In a first line of work, a new model for safety conditions was proposed; this new equation was proven to increase the power efficiency by a 3%.

In a second part, the importance of LICQ in periodic OCPs was discussed. In particular, two different methods to overcome LICQ deficiency were described: the invariant stabilization method, originally used by [12], and the alternative projection method. After comparing their performance, three conclusions were drawn:

(i). Within the framework of the AWE kite, the invariant stabilization method is a more erratic algorithm. In particular, it solves half as many problems as the projection method.

(ii). Despite the projection method being a better algorithm, the quaternion formulation seems to be ill-posed. Particularly, using the projection method, the quaternion model could only solve one out of four OCPs.

(iii). The equations of motion are not stiff and explicit integrators perform better than implicit ones.

In a third section, different dynamical models were examined: a first model based on Euler angles, a second based on quaternions, and a third model using natural coordinates. The target of this research was to assess which model is better for periodic OCPs within the field of AWE. The conclusions of this experiment can be summarized as follows:

(i). Euler angles seem to be the fastest and most accurate implementation. In particular, they are able to solve more than half of the proposed OCPs with computation times that outperform the other two formulations. Their main drawback is that, since their OCP formulation is highly nonlinear, they find local minima that are not as good as natural coordinates solutions.

(ii). In contrast with Euler angles, natural coordinates were shown to be a slower and less successful approach. In particular, they solved one out of four problems but with computation times that were three times as higher as the ones in Euler angles. Nevertheless, unlike the latter, their formulation is quite linear, and as a result, when they find a solution, it is in many cases more efficient than the other two formulations.

(iii). Finally, the quaternion formulation did not show any clear advantage; particularly, it was the most unsuccessful implementation by solving one of 7 cases, with computation times that were worse than Euler angles, and with power efficiencies that were worse than natural coordinates.

Finally, in a last section, the properties of flight topologies were studied. In particular, in a first part, it was proved that topology constraints are not a requirement in order to obtain the desired lemniscates. Moreover, in a second part, it was proved that lemniscates are not the only feasible flight topology; in particular, it was demonstrated that circular trajectories can also produce power efficient flights with a high η_{Loyd} factor while simultaneously avoiding tether twist.

Part II

Nonlinear Model Predictive Control

Chapter 4

Model Predictive Control in a Nutshell

Feedback control theory encompasses two main branches of research: *classical feedback control* and *state space control*, with NMPC falling within the latter. In the coming sections, it will be explained the difference between these two approaches, the disadvantages of classical control theory with respect to NMPC and why the latter is in general a better control strategy. Finally, the theory of NMPC and its application in real time systems will be described.

Before continuing, I would like to remark that Section 4.2.2-4.2.4 and 4.3 are greatly based and inspired by [28, Chapters 18-19]. The main reasons behind this decision are two:

(i). [28] is a textbook that is currently being written by my supervisor Prof. Dr. Moritz Diehl on topics related to my Master's education. As a result, it has been a valuable asset for the development and implementation of the NMPC.

(ii). During my thesis research I have also collaborated in this work by collecting and re-editing nearly all of the exercises of the book, Latex editing of text and formulas and suggesting and implementing changes in the organization of the chapters. As a consequence, I became very familiar with the topic and it has been a very useful resource for learning NMPC.

No implicit reference will be given further on for [28], but the similarity between the mentioned sections and the cited book should be kept in mind.

4.1 Classical Feedback Control Limitations

Classical feedback control is a set of algorithms, which have a very distinctive working principle, used to control dynamical systems. In particular, their key feature is that they use the Laplace transform to model the system in the frequency domain in order to control it by means of its transfer function. An area where they are widely used is *single input single output (SISO)* linear systems. A famous example of this type of controllers is the well known PID controller.

The main goal is to control a system output to follow a given reference. To do so, they observe the specific output, they compute the error e between the output and the reference and they feed this error to the controller, which then selects the appropriate inputs to bring the output closer to the reference. Figure 4.1 illustrates a classic control scheme.

Despite being very popular in the past, due to the increase of available computational power in recent years, the use of these techniques has decayed in favor of the more modern space state control. This decline is better understood when looking at the set of disadvantages

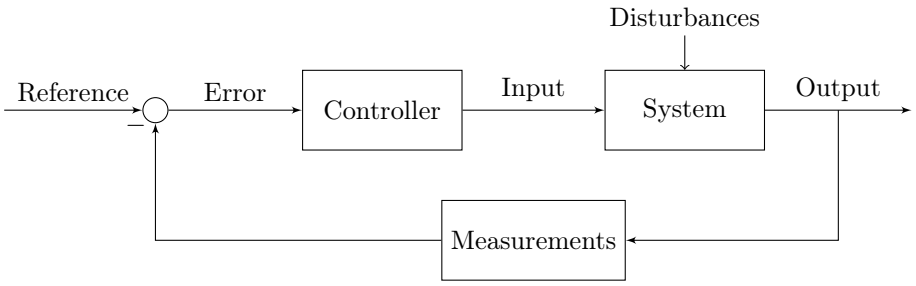


FIGURE 4.1: Schematic of a simple feedback.

and limitations that classical feedback control has always faced and which space state control can easily overcome:

(i). In general, they are not well suited for controlling *multiple input multiple output (MIMO)* systems. In particular, to control a MIMO system classical control theory could use a single controller per output. Nevertheless, if a single input influences several outputs (quite common scenario in a regular system), more than one controller will be trying to change the same input, which in turn leads to a scenario where there is no clear input control law. By contrast, modern approaches permit the evaluation of single objective function with the goal of combining different outputs in a single function.

(ii). Classic techniques use past information, i.e. the events that have already occurred. On the other hand, new modern algorithms such as NMPC make use of the dynamical model in order to predict the system behavior and optimize the controls using this predicted information; as a result, they are able to anticipate future disturbances as long as they can be modeled.

(iii). Classic control theory is in many cases limited to linear systems, which restraints the application of classic controller to a very specific set of applications. On the other hand, state space control models the dynamics of the system and can deal with the nonlinear case.

(iv). State space approaches provide a full representation of the state and its time evolution, whereas classic control theory does not. All this extra information removes several limits of classical control theory, expanding the area of application to a broader field.

(v). Space state control has a very natural way of implementing constraints, allowing the constraints to be implemented not only in the inputs, but also in the outputs and states. By contrast, in classic controllers the constraints might be imposed to the inputs by limiters, but even then, it does not prevent the controller from trying to select infeasible values in the inputs.

A very simple way of interpreting these differences can be seen with the analogy proposed by Camacho and Bordons in [18] when comparing PID with NMPC. NMPC can be thought as the natural way of driving a car looking trough the front window: the driver has a goal for the car and controls the car to achieve this goal by planning ahead the car behavior. To do so, he looks trough the front window what it is ahead of him and takes decisions accordingly.

The goal can involve several variables, e.g. fuel consumption, safety constraints, traffic rules, end destination, etc., but the driver knows how to find the balance between them and take decisions accordingly, e.g. accelerate/decelerate, choose route, etc. By contrast, PID can be though as driving the car looking through the rear mirror, taking only information from the past while trying to achieve a single objective, e.g. keep constant velocity.

Considering all these advantages, the aim of this thesis is to substitute the classical controller of the Skysails AWE kite by a NMPC, and then, test and compare their relative performance. A second motivation for this substitution is the fact that tracking optimal trajectories can only be achieved by planning ahead the kite behavior. In order to accomplish that, predictive controller is required.

4.2 NMPC Theory

As it has been already introduced in the previous sections, NMPC is a feedback predictive control algorithm which, by means of a system dynamical model, anticipates the future and selects the optimal control policy to optimize a given cost function. Its basic working principle is to solve an OCP at each time iteration so that an optimal policy is obtained while ensuring the validity of the dynamical model as well as some other constraints. In particular, the controller strategy can be described as follows:

(i). First, the controller receives information regarding the current system state $\bar{x}_0$. This information can be obtained by means of an observer, e.g. a *Kalman filter* or a *moving horizon estimation (MHE)*.

(ii). Then, it solves an OCP in order to obtain the optimal control policy U^* that optimizes a given cost function $J(U, X)$, where the decision variables $U = (u_0, u_1, \ldots, u_{N-1})$ and $X = (x_0, x_1, \ldots, x_N)$ are defined for a predictive time horizon T in a discretized time grid $0 = t_0 < t_1 < t_2 < \ldots < t_N = T$ of $N + 1$ points.

(iii). The OCP includes dynamical constraints to ensure that the system evolution is feasible. It also imposes the initial constraint $x_0 = \bar{x}_0$ to ensure that the observation of the current state is taken into account. Finally, path constraints to ensure safety conditions or some other constraint might be also enforced.

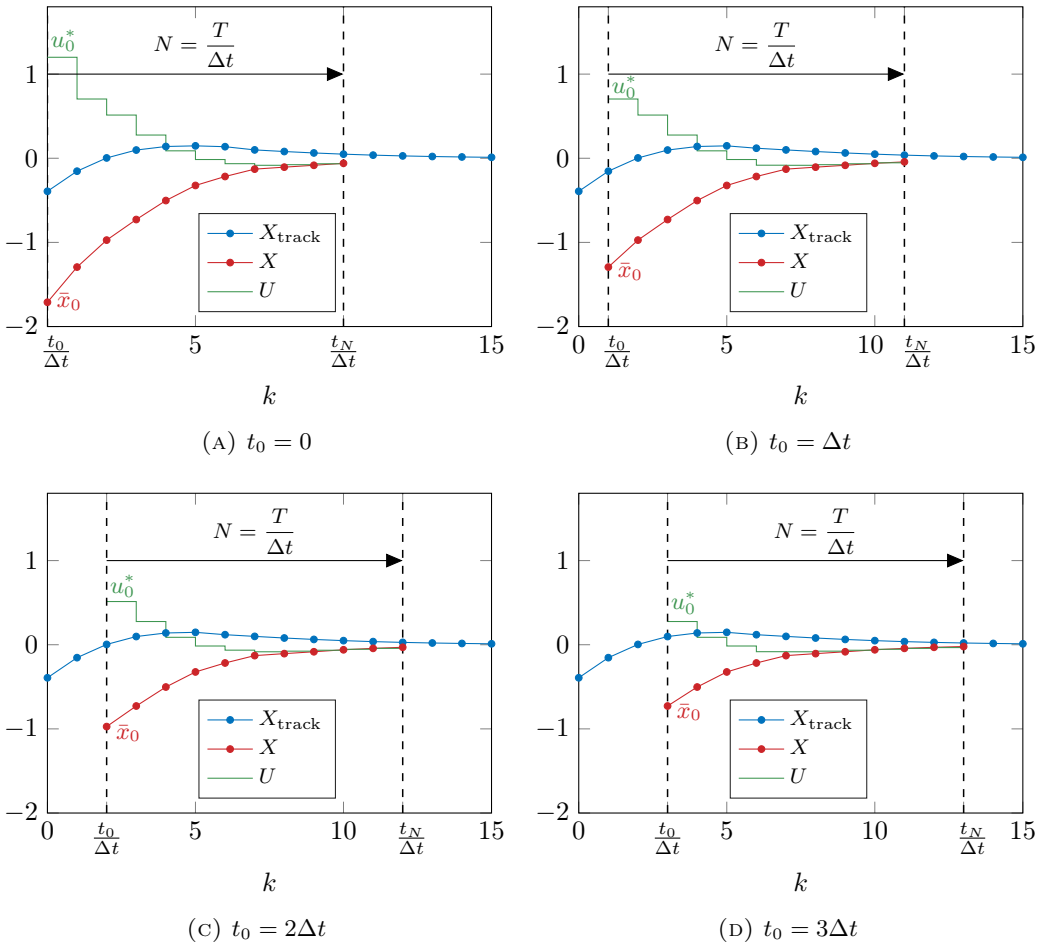
(iv). Once the OCP is solved, the algorithm applies the first optimal control u_0^* to the system, it moves the optimization horizon a time step forward, observes the new state $\bar{x}_0$ and repeats the procedure.

The above scheme is illustrated in Algorithm 1:

Algorithm 1 NMPC

1: **Input:** OCP initial guess $X_0 = (x_0, \ldots, x_N)$, $U_0 = (u_0, \ldots, u_{N-1})$.
2: **while** stop **do**
3: $\bar{x}_0 \leftarrow$ Observer()
4: U^*, $X^* \leftarrow$ Solver($\bar{x}_0$, U_0, X_0)
5: Send_to_system(u_0^*)
6: U_0, $X_0 \leftarrow$ Next_Initial_Guess(U^*, X^*)
7: **end while**

When designing the NMPC, it is important to consider that the controller time step $\Delta t = \frac{T}{N}$ is usually defined for the specific application of the NMPC. As a result, T and N have to be small enough to ensure that, with the available computational power, the NMPC computation time per iteration is smaller that Δt. Since N and T are interrelated, both are indistinctly referred to as the predictive horizon. A simple example of a NMPC can be seen in Figure 4.2, where four consecutive iterations of the algorithm are shown.



(A) $t_0 = 0$ (B) $t_0 = \Delta t$

(C) $t_0 = 2\Delta t$ (D) $t_0 = 3\Delta t$

FIGURE 4.2: NMPC evolution in 4 consecutive time steps

In this particular case, the NMPC solves at each iteration the following OCP:

$$
\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \|x_{\text{track},k} - x_k\|^2 \\
\text{subject to} \quad & x_0 - \bar{x}_0 = 0, \\
& \Phi_k(x_k, u_k) - x_{k+1} = 0, \qquad k = 0, \dots, N-1, \\
& u_{\min} \leq u_k \leq u_{\max}, \quad k = 0, \dots, N-1.
\end{aligned}
$$

Since the cost function is the least squares error between the system state X and a tracking reference trajectory $X_{\text{track}} = (x_{\text{track},0}, \ldots, x_{\text{track,N}})$, it can be seen how the time horizon T moves a time step ahead at each iteration, and while doing so, the system state X converges towards the tracking trajectory X_{track}.

As stated by [18], NMPC presents a set of advantages over other control methods, among which, seven of them are really remarkable :

(i). It is relatively easy to understand due to the intuitive nature of the NMPC concepts making it very attractive to people with limited knowledge on control theory.

(ii). It can be used to control a broad range of systems, those with very simple dynamics but also others with very complicate equation of motions and constraints.

(iii). Multivariable systems can be easily controlled.

(iv). It includes feedforward control in a natural way to make up for known disturbances that can be modeled or measured.

(v). It can easily implement system constraints.

(vi). Because of being based on basic principles, it is an open methodology that can be extended in the future.

4.2.1 Economic Versus Tracking NMPC

As stated in the introduction, NMPC does not refer to a single algorithm, but instead, to a family of control algorithms with a common structure. As a result, to understand better this set of algorithms, three very generic NMPC schemes differing in their objective functions will be explained.

Standard (or classic) NMPC is characterized for using as a cost function the sum of the least squares errors between the system states along the predictive horizon and a selected steady state $(x_{\text{s}}, u_{\text{s}})$. As a result, the OCP that the classic NMPC scheme solves at each iteration is given by:

$$
\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \Big(\|x_k - x_s\|_Q^2 + \|u_k - u_s\|_R^2 \Big) + E(x_N) \\
\text{subject to} \quad & x_0 = \bar{x}_0, \\
& x_{k+1} - \Phi_k(x_k, u_k) = 0, \quad k = 0, \ldots, N-1, \\
& h(x_k, u_k) \leq 0, \quad k = 0, \ldots, N-1, \\
& r(x_N) = 0.
\end{aligned}
$$

where $Q > 0$ and $R > 0$. Since there is plenty of theory available for the different variations of classic NMPC [29], this topic will not be examined any further.

A second scheme of NMPC is the so-called *tracking NMPC*, where an optimal state trajectory X_{track} and/or optimal control policy U_{track} are precomputed offline, and then, tracked

by means of NMPC. In particular, NMPC uses as a cost function the least squares error between the optimal and the predicted trajectory and the tracking reference trajectories X_{track} and U_{track}, which are usually obtained by means of an offline OCP. Given the trajectories $X_{\text{track}} = (x_{\text{track,0}}, \ldots, x_{\text{track,N}})$ and $U_{\text{track}} = (u_{\text{track,0}}, \ldots, u_{\text{track,N}-1})$, the tracking NMPC scheme solves at every iteration the following problem:

$$
\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \Big(\|x_{\text{track},k} - x_k\|_Q^2 + \|u_{\text{track},k} - u_k\|_R^2 \Big) + E\left(x_N\right) \\
\text{subject to} \quad & x_0 - \bar{x}_0 = 0, \\
& \Phi_k(x_k, u_k) - x_{k+1} = 0, \quad k = 0, \ldots, N-1, \\
& h(x_k, u_k) \leq 0, \quad k = 0, \ldots, N-1, \\
& r\left(x_N\right) \leq 0,
\end{aligned}
$$

where $Q > 0$ and $R > 0$. It is important to remark that standard NMPC is just a version of tracking NMPC, and as a result, the latter is in many cases used to refer to the former in the literature.

A third and more generic case is *economic NMPC*, where the objective function is the sum of some economic stage cost $l(x_k, u_k)$. As a result, the OCP that the NMPC solves at each iteration can be expressed as:

$$
\begin{aligned}
\underset{X,U}{\text{minimize}} \quad & \sum_{i=0}^{N-1} l(x_k, u_k) + E(x_N) && \text{(4.1a)} \\
\text{subject to} \quad & x_0 - \bar{x}_0 = 0, && \text{(4.1b)} \\
& \Phi_k(x_k, u_k) - x_{k+1} = 0, \quad k = 0, \ldots, N-1, && \text{(4.1c)} \\
& h(x_k, u_k) \leq 0, \quad k = 0, \ldots, N-1, && \text{(4.1d)} \\
& r\left(x_N\right) \leq 0. && \text{(4.1e)}
\end{aligned}
$$

Stability theory for economic NMPC is not as developed as for tracking NMPC and as a result, tracking NMPC is a safer and more stable choice whenever it comes to real system implementations. Therefore, despite using economic NMPC for maximizing the extracted energy could be an implementation option, we instead decided to work with tracking NMPC and offline computed optimal tracking trajectories. We expect that this choice will safeguard the safety and stability of the system while still producing very good results in the system efficiency.

4.2.2 Nominal Stability

Stability on the NMPC is a topic that has been studied for years. Diverse mathematical proofs of stability have been obtained for some specific NMPC schemes. However and in general, proving stability is a very hard and sometimes even impossible task, and in practice,

when NMPC is implemented in a real system, the easiest way to ensure stability is to test the controller under real life conditions and tune the NMPC parameters to make it robust. This task is usually realized at the simulation level by means of modeling the disturbances and mismatches of the real system. This avoids the unnecessary expenses and system breakdowns while tuning the controller in the real system.

Due to the practical methodology to ensure robustness, this section does not aim at proving or showing a general stability theory; instead, only a specific result in NMPC stability will be stated in order to motivate a method for the NMPC initialization of the next section. For an overview of the NMPC schemes with stability guarantee, [30] can be used as an excellent reference source.

Theorem 4.1 (Nominal stability of steady state NMPC)**.** *Assume that:*

- *The dynamical system has a steady state* $(\hat{x}, \hat{u})$*, i.e.* $\dot{x}(\hat{x}, \hat{u}) = \Phi(\hat{x}, \hat{u}) = 0$*, with* $(\hat{x}, \hat{u})$ *being a feasible point of the problem.*
- *The NMPC uses as a stage cost the sum of the quadratic deviations from this steady state, i.e.* $L(x, u) = (x - \hat{x})^\top Q(x - \hat{x}) + (u - \hat{u})^\top R(u - \hat{u})$*, with* $Q > 0$ *and* $R > 0$*.*
- *There is no end cost, i.e.* $E(x_N) = 0$*.*
- *The end state is constrained to the steady state, i.e.* $x_N = \hat{x}$*.*
- *There are no model-plant mismatches.*

Then, the value of the NMPC cost function is a Lyapunov function that decreases by at least $L(\bar{x}_0, u^*(\bar{x}_0))$ *at each iteration, which in turn ensures the NMPC stability.*

Proof. Assume that the NMPC starts at a point $\bar{x}_0$ and that the solution of the first iteration is $(x_0^*, u_0^*, x_1^*, u_1^*, \ldots, x_{N-1}^*, u_{N-1}^*, x_N^*)$. If u_0^* is applied to the system, the system evolves until point x_1^* due to the lack of mismatches and disturbances.

Starting at x_1^* in the new iteration, the decision variable vector X defined by shifting the previous solution one step backwards and adding the steady state at the end:

$$X_{0,\text{new}} = (x_1^*, u_1^*, \ldots, x_{N-1}^*, u_{N-1}^*, x_N^*, \hat{u}, \hat{x})$$

is not only a feasible point but it also improves the cost function with respect to the previous iteration. This can be easily proved considering that by removing the first state and control from the vector, the objective is decreased by an amount $L(\bar{x}_0, u^*(\bar{x}_0))$, and that the stage cost of the added states $L(\hat{x}, \hat{u})$ is zero.

Finally, by starting the optimization problem at this value $X_{0,\text{new}}$ the objective can only get better, and as a result, it can be stated that the cost function decreases by at least $L(\bar{x}_0, u^*(\bar{x}_0))$ in each iteration.

□

4.2.3 NMPC Initialization

It is important to consider that the optimization problems that NMPC solves at each iteration are not independent from each other. As a result, a good NMPC implementation would use information from previous iterations to obtain a better solution. The simplest way to transfer information is considering that the solution of the previous NMPC iteration, $U^* = (u_0^*, \ldots, u_{N-1}^*)$, $X^* = (x_0^*, \ldots, x_N^*)$, is a very good initialization guess X_0, U_0 of the next initialization problem.

An even better way to transfer information is to consider that, since the system moved a time step into the future, a shifted version of U^* and X^* would actually represent a better solution at the current time. This shifting is performed similarly as it was done for the nominal stability guarantee, i.e. by removing the first value of U^* and X^* and adding a value to the end:

$$X_0 = (x_1^*, \ldots, x_N^*, x_{\text{new}})$$
$$U_0 = (u_1^*, \ldots, u_{N-1}^*, u_{\text{new}})$$

where x_{new} and u_{new} can be chosen in different ways.

This concept is called *shift initialization* and it is motivated by the *principle of optimality of subarcs* and its application on *receding horizon NMPC* without disturbances: since the horizon N (and in turn the size of X and U) decreases by a time step at every iteration, if the solution at iteration k is given by $(x_0^*,\ u_0^*,\ x_1^*,\ u_1^*,\ \ldots,\ x_{N-1}^*,\ u_{N-1}^*,\ x_N^*)$, then, the shifted vector $(x_1^*,\ u_1^*,\ \ldots,\ x_{N-1}^*,\ u_{N-1}^*,\ x_N^*)$ is a solution of the NMPC at iteration $k+1$. Bear in mind that at iteration $k+1$ the system is at x_1^* due to the lack of disturbances.

In the case of *moving horizon NMPC*, i.e. N is constant, the principle of optimality of subarcs can not be strictly used and the selection of the appended values at the end plays an important role. A common policy is to copy the last control value, i.e. $u_N = u_{N-1}$, or to use $\hat{u}$ in the case of steady state control, and then simulate the system from x_N^* to obtain a initialization of x_{N+1}.

As proved in [31], shift initialization does not always provide an advantage when compared with directly copying the old solution from the previous iteration. Nevertheless, in the particular case of periodic tracking NMPC, shift initialization has been proved to have a significant contribution as showed in [32].

4.2.4 Real-Time Optimization

As it has been already described before, special algorithms and methods are required to cope with the timing and computational requirements of NMPC. This section will give a brief overview of some ideas, but it would be in Section 4.3 when some of them will be described in more detail.

- *Offline computations*: in order to save online computational resources, some parts of the optimization problem can be computed offline. In particular, Hessian and Jacobian matrices, reference trajectories and compiled computer code are among common entities that are precomputed offline.
- *Iterate before convergence*: the idea is to start the optimization problem of the next iteration without fully convergence at the current iteration, and then use the shifted solution of the current iteration as initialization guess for the next one. More details will be given in Section 4.3.
- *Delay compensation*: if the computation time is too long, the NMPC does not use the current state as initial point but instead a predictive state of where the system will be after the NMPC computations.
- *Phase division*: in each iteration, the computations are divided into two phases. In the *preparation phase*, the NMPC performs heavy computation while it waits for the observer to provided an estimate of the current state x_0. In a second phase called

feedback phase, the NMPC computes a quick solution using the precomputations and the state estimation.

4.3 Real Time Iteration Scheme

When the NMPC initialization was explained in the previous section, it was assumed that the system would evolve without disturbances. As a result, it was expected that, at the beginning of each iteration, the system would be located at the point predicted by the previous iteration when applying the first optimal control u_0^*. In reality, disturbances occur between two iterations and the current initial point $\bar{x}_0$ differs from the predicted one.

In order to model this disturbance, the initial point x_0 is included as a problem variable and the equation $x_0 - \bar{x}_0 = 0$ is added as a constraint (as in Equation (4.1b)). Then, modeling $\bar{x}_0$ as a parameter that the observer has to estimate, each NMPC iteration can be seen as a case of *parametric nonlinear program (pNLP)* with variables $Y = (X, U)$ and parameter $\bar{x}_0$:

$$\begin{aligned} \text{pNLP}(\bar{x}_0): \qquad & \underset{Y}{\text{minimize}} \quad F(Y) \\ & \text{subject to} \quad G(\bar{x}_0, Y) = 0, \\ & \qquad\qquad\quad H(Y) \leq 0. \end{aligned} \tag{4.2}$$

Initial Value Embedding

Recalling Definition 2.4, the Lagrangian function $\mathcal{L}(Y, \lambda, \mu)$ of Problem (4.2) is defined as:

$$\mathcal{L}(Y, \lambda, \mu) = F(Y) + G(\bar{x}_0, Y)^\top \lambda + H(Y)^\top \mu \tag{4.3}$$

where λ and μ are the so-called Lagrange multipliers. Furthermore, considering Theorem 2.2, if Y^* is the optimal solution of Problem (4.2) it has to satisfy the KKT conditions:

$$\nabla_Y \mathcal{L}(Y^*, \lambda^*, \mu^*) = 0, \tag{4.4a}$$
$$G(\bar{x}_0, Y^*) = 0, \tag{4.4b}$$
$$H(Y^*) \leq 0, \tag{4.4c}$$
$$\mu^* \geq 0, \tag{4.4d}$$
$$H_k(Y^*)\, \mu_k^* = 0, \quad k = 1, \ldots, n_H. \tag{4.4e}$$

When looking at Equation (4.1b) it can seen that $\bar{x}_0$ enters G linearly, and that as a result, neither the Jacobian of G nor any other derivative are $\bar{x}_0$-dependent. The term *initial value embedding* refers to this specific linear parameter dependence, and as it will be seen in the next section, it will be the base of the path-following algorithms and the real time iteration scheme.

Path-Following Methods

Regard $W^*(\bar{x}_0) = \big(Y^*(\bar{x}_0), \lambda^*(\bar{x}_0), \mu^*(\bar{x}_0)\big)$ as the solution manifold of Equations (4.4a)-(4.4e) for different values of $\bar{x}_0$.

In the case of having only equality constraints, the KKT conditions (4.4a)-(4.4b) can be summarized as a root finding problem $R(\bar{x}_0, W) = 0$, and as a result, $W^*(\bar{x}_0)$ becomes a smooth solution map at all the points where the KKT matrix $\frac{\partial R}{\partial W}$ is invertible.

In a real time optimization framework the last result means that, if a NMPC iteration is solved with a parameter $\bar{x}_0$ and a solution $W = W^*(\bar{x}_0)$ is reached, a *tangential predictor* W' for the new solution $W^*(\bar{x}_0')$ can be obtained by linearizing $R(\bar{x}_0, W) = 0$ at $(\bar{x}_0, W)$. In that case, $\bar{x}_0'$ and W' represent the parameter and solution at the next iteration and the linearized root finding problem is given by:

$$R(\bar{x}_0, W) + \frac{\partial R}{\partial \bar{x}_0}(\bar{x}_0, W)(\bar{x}_0' - \bar{x}_0) + \frac{\partial R}{\partial W}(\bar{x}_0, W)(W' - W) = 0.$$

Rearranging this last equation and considering that due to the initial value embedding:

(i). The dependence on the $\bar{x}_0$ derivatives can be omitted.

(ii). $\frac{\partial R}{\partial \bar{x}_0}(W)(\bar{x}_0' - \bar{x}_0)$ can be written $R(\bar{x}_0', W) - R(\bar{x}_0, W)$.

Then, W' is given by:

$$W' = W - \left(\frac{\partial R}{\partial W}(W)\right)^{-1} R(\bar{x}_0, W).$$

These algorithms to predict the optimal solution for successive NMPC iterations are called *path-following methods*, and whenever the NMPC solves a equality constraint OCP they represent a fast computation of successive solutions for real time optimization.

In the case of inequality constrained NMPC, Equation (4.4e) is non-smooth and the previous approach is not that simple. There are two main approaches to this issue: one based on *interior point (IP)* methods and a second one based on *sequential quadratic programming (SQP)* algorithms.

Interior Point Path-Following Methods

IP methods solve the general KKT conditions by substituting the non-smooth equation (4.4e) by a smooth approximation:

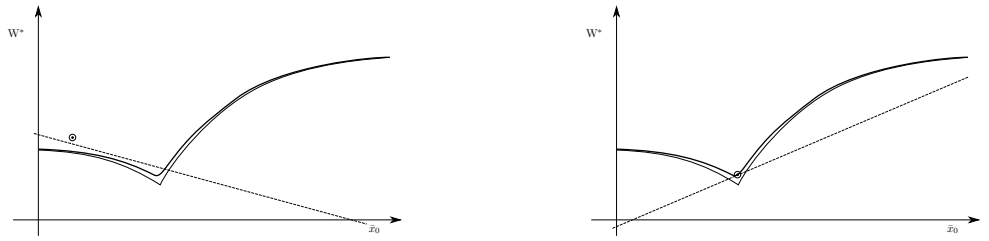
$$\begin{aligned} \nabla_Y \mathcal{L}(Y^*, \lambda^*, \mu^*) &= 0, \\ G(\bar{x}_0, Y^*) &= 0, \\ H_k(Y^*)\, \mu_k^* + \tau &= 0, \quad k = 1, \ldots, n_H, \quad \tau > 0. \end{aligned}$$

Since for small values of τ the problem above approximates the original one, the general idea of IP solvers is to iteratively solve the smooth root finding problem for decreasing values of τ.

In the framework of path following methods, if τ was constant, the above equations would represent a parametric root finding problem $R(\bar{x}_0, W) = 0$ that would provide the solution manifold $W^*(\bar{x}_0')$. As a consequence, the theory from the previous section could be used to obtain the tangential predictor W' at each iteration, and then, by reducing τ at every step, the solution manifold would be expected to approximate better and better the real NMPC solution.

The main drawback of this type of methods is that the tangential predictor is a very bad approximation for small τ values at the points where the active set (set of inequality

constraints holding with equality) changes. This problem can be seen in Figure 4.3 where the solution manifold $W^*(\bar{x}_0)$ as a function of $\bar{x}_0$ is represented.



(A) Linearizing at approximate solution (B) Linearizing at active set change

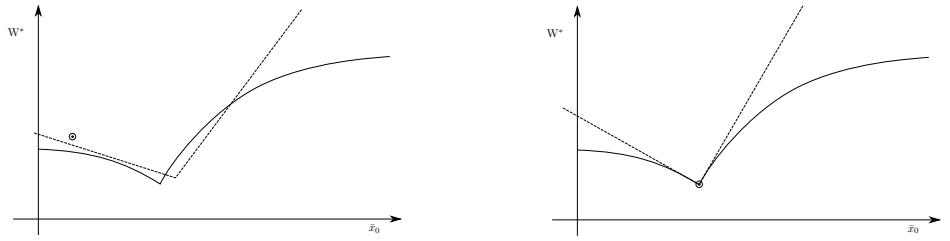
FIGURE 4.3: Tangential predictors for interior point method using a small τ [28].

Some approaches have been proposed to avoid this last issue [33, 34], nevertheless, in this thesis we decided to rely on the methods presented in the next two sections as they have a specific software environment for their implementation and they ensure a more accurate tangential predictor.

SQP Path-Following Methods

The main issue with IP methods is that, as they try to solve a smooth root finding problem, they have to approximate the KKT conditions, and as it has been shown, this approximation is not good at the points where the active set changes.

A way to avoid the previous problem is to use *directional derivatives* of the solution manifold [35, Thm 3.3.4] as tangential predictor. In the NMPC context, they are usually referred to as *generalized tangential predictors*, and Figure 4.4b provides an example of them.



(A) Linearizing at approximate solution (B) Linearizing at active set change

FIGURE 4.4: Generalized tangential predictors [28].

A practical algorithm using the above concept is proposed in [36]: given the current solution guess W^k for a parameter $\bar{x}_0$, the next solution W^{k+1} for a parameter $\bar{x}_0'$ is obtained as the solution of the following parametric *quadratic programming (QP)* problem:

$$\begin{aligned} \text{pQP}(\bar{x}_0', W^k): \quad & \underset{Y}{\text{minimize}} \quad \nabla F(Y^k)^\top Y + \frac{1}{2}(Y - Y^k)^\top \nabla_Y^2 \mathcal{L}(Y^k, \lambda^k, \mu^k)(Y - Y^k) \\ & \text{subject to} \quad G(\bar{x}_0', Y^k) + \nabla G(Y^k)^\top (Y - Y^k) = 0, \\ & \qquad\qquad H(Y^k) + \nabla H(Y^k)^\top (Y - Y^k) \leq 0. \end{aligned} \tag{4.5}$$

Where as proved in [36, Thm. 3.6] and illustrated in Figure 4.4b, when starting at the solution $W^k = W^*(\bar{x}_0)$ the above QP delivers the exact generalized tangential predictor. Moreover, as shown in Figure 4.4a, when starting close to the solution or using a Hessian or Jacobian approximation W^{k+1} is still a good predictor.

Considering the above, the idea behind this type of approaches is to solve a QP problem at each iteration to obtain the predictor W'. For the same reason, they are usually referred to as *SQP path-following methods*.

Condensing

A specific technique to solve Problem 4.5 would be to use the so-called *simultaneous approach* [28, Section 8.1.3] for discrete optimal control. The main idea would be to leave only the controls U and the initial state x_0 in the decision variable Y and substitute the states X by some integration routine $f(U, x_0)$, so that as a result, the resulting parametric QP is a *condensed* version of the original problem:

$$\begin{aligned} \text{pQPcond}(\bar{x}_0', W^k): \qquad & \underset{U}{\text{minimize}} \quad f_{\text{condQP},k}(\bar{x}_0', U) \\ & \text{subject to} \quad \bar{r}_k + \bar{R}_k^{x_0}\bar{x}_0 + \bar{R}_k^U U \leq 0. \end{aligned}$$

In real time optimization, the importance of this condensed QP relies on the fact that if the vector U is not too large, there are dense QP solvers that can exploit this specific structure and solve the QP very quickly at the same time that keep explicit expressions for the parameter x_0 and the first control u_0 (needed for the NMPC feedback).

Real Time Iteration Scheme

The *real-time iteration (RTI)* scheme is a type of NMPC specifically designed for real time computing first presented in [36, 37]. Its core features and concepts are based on all the previously presented ideas and can be summarized as follows:

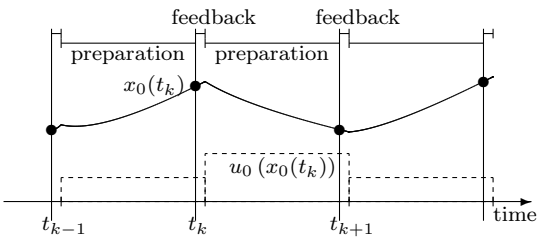
(i). The NMPC is implemented using a SQP approach with a Gauss-Newton Hessian approximation where at each iteration a single QP (as given by Equation 4.5) is solved. As a result, only one system linearization is required per sampling time.

(ii). The NMPC is modeled using multiple shooting with full derivatives and condensing.

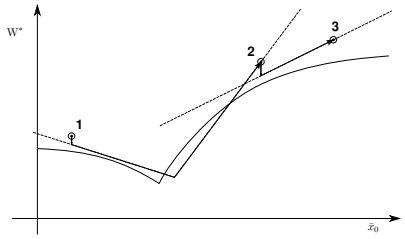
(iii). The generalized tangential predictor is used to correct for the mismatch between the expected x_0 and real initial state $\bar{x}_0$.

(iv). The algorithm is divided into a *preparation phase*, where the system linearization and condensing are performed (heavy computations), and a *feedback phase*, where the single condensed QP is solved. The preparation phase is usually performed while the controller waits for the new initial state $\bar{x}_0$ and it is in general several orders of magnitude larger than the feedback phase.

Figure 4.5 illustrates these concepts: on the left, the division between preparation and feedback phase, and on the right, the solution representation for three iterations. From 4.5a



(A) Division of one real-time iteration into preparation and feedback phase.



(B) Subsequent solution approximations

FIGURE 4.5: Real time iteration scheme [28].

it can be observed that thanks to the division in two phases, the control feedback to the system occurs right after obtaining the new $\bar{x}_0$ value.

Considering the described features of the RTI scheme, the full NMPC algorithm using this set of ideas can be depcited by the following algorithm:

Algorithm 2 Real Time Iteration NMPC

1: **Input:** initial guess $X^0 = (x_0^0, \ldots, x_N^0)$, $U^0 = (u_0^0, \ldots, u_{N-1}^0)$ and $i = 0$.
2: **while** stop **do**
3: Obtain system trajectory sensitivities required for linearization at the grid nodes.
4: Prepare the QP problem and condense it.
5: Wait for new current state $\bar{x}_0$.
6: Solve the QP and send the new control u_0^i to the system.
7: Increment i and shift the initial guess X^i, U^i for the next iteration.
8: **end while**

It is important to remark that error bounds and close loop stability of the RTI scheme have been established for shrinking horizon problems [38] and for NMPC with shifted and non-shifted initializations [39, 40].

The main drawback of the RTI scheme is that, in order to overcome the issues of the conventional tangential predictor of IP methods, it has to use a more computationally costly QP predictor. Due to this high QP cost in fast NMPC applications, a new technique called *online active set strategy* was proposed by [41]. The idea is, based on the fact that the QP solution depends affinely on $\bar{x}_0$ for constant active sets, to create some solution homotopy that only changes when it crosses an active set boundary. The online active set strategy is available in the QP solver package qpOASES [42], and will be used for our own NMPC implementation.

Chapter 5

Controller Implementation

As stated in the introduction, one of the main contributions of this thesis is the development of a kite controller to track optimal trajectories. Moreover, it has been explained that a tracking NMPC approach using the RTI scheme would be an optimal choice for this particular application.

As a result, this chapter will aim at providing the derivation of such a controller. In particular, a software toolbox used for the implementation will be introduced in a first section; then, in a second section, the different parts of the NMPC will be described; finally, in the last two sections, a system simulation will be modeled and a small disturbance in the form of a delay introduced.

It is important to note that this chapter will not cover test simulations, but instead, it will focus on the NMPC implementation leaving the testing scenarios for the next chapter.

5.1 ACADO Toolbox

ACADO Toolbox is a C++ open-source package that provides a user-friendly environment to work with different OCP algorithms. In particular, it implements a code generation tool that helps with the design of RTI NMPC schemes (as given by Algorithm 2), and then, it exports highly efficient C code for real-time optimal control [43]. Furthermore, it features a MATLAB interface, highly accurate integrators optimized for real time optimization [44] as well as the inclusion of qpOASES, the QP solver designed for implementing the active set strategy. As a result, there is a series of clear advantages that makes ACADO the optimal environment to perform our NMPC implementation:

- The NMPC algorithm can be designed, implemented and tested in a user-friendly MATLAB environment.
- A full RTI scheme can be prototyped without having to worry about numerical errors.
- ACADO offers out of the box accurate integrators optimized for real time implementations.
- Easy interface with qpOASES for implementing the active set strategy.
- The RTI scheme can be exported in optimized C code that can later be used in the real time controller of the Skysails system.

It is important to remark that the code generation tool might not be flexible enough for every application; in particular, there are some limitations regarding the type of NMPC scheme that can be implemented:

- The objective function of the NMPC is limited to nonlinear least squares schemes $\|M(x)\|^2$.
- Since nonlinear constraints are linearized at each iteration and the RTI scheme does not solve the problem until convergence, nonlinear path constraints do not always ensure a good NMPC performance.

Nevertheless, since we want to implement tracking NMPC with boundary constraints, ACADO and its code generation tool are an optimal choice for our application. The resultant NMPC scheme is defined by:

$$\underset{X,U}{\text{minimize}} \quad \sum_{k=0}^{N-1}\Big(\|x_{\text{track},k} - x_k\|_Q^2 + \|u_{\text{track},k} - u_k\|_R^2\Big) + (x_N - x_{\text{track},N})^\top Q_N (x_N - x_{\text{track},N}) \tag{5.1a}$$

$$\text{subject to} \qquad x_0 - \bar{x}_0 = 0, \tag{5.1b}$$

$$\Phi_k(x_k, u_k) - x_{k+1} = 0, \qquad k = 0, \dots, N-1, \tag{5.1c}$$

$$x_{\min} \le x_k \le x_{\max}, \quad k = 0, \dots, N, \tag{5.1d}$$

$$u_{\min} \le u_k \le u_{\max}, \quad k = 0, \dots, N-1. \tag{5.1e}$$

where:

$$X = \big(x_0, \cdots, x_N\big), \qquad X_{\text{track}} = \big(x_{\text{track},0}, \cdots, x_{\text{track},N}\big),$$
$$U = \big(u_0, \cdots, u_{N-1}\big), \quad U_{\text{track}} = \big(u_{\text{track},0}, \cdots, u_{\text{track},N-1}\big),$$

5.2 Base NMPC Implementation

In order to define a basic NMPC scheme that obeys (5.1), a set of different options have to be selected and defined:

- The objective function, i.e. the weighting matrices Q, R and Q_N and the tracking trajectory X_{track}, U_{track}.
- The boundary conditions for the control and states.
- The predictive horizon N, T as well as the controller sampling time Δt, which will determine the required computational power.
- Specific algorithms that the NMPC will use: type of integrator, type of Hessian approximation, etc.

5.2.1 Objective Function

As it has been explained in the previous section, the NMPC has to continuously track periodic optimal trajectories, $X_{\text{opt}} = (x_{\text{opt},0}, \dots, x_{\text{opt},M})$ and $U_{\text{opt}} = (u_{\text{opt},0}, \dots, u_{\text{opt},M-1})$, that are

generated as the solution of a periodic OCP maximizing the power per cycle. Considering that, due to the periodicity condition $x_{\text{opt},0} = x_{\text{opt},M}$, the NMPC tracking trajectories at time $t_0 = k\Delta t$ are defined as:

$$\begin{aligned}
&X_{\text{track}} = (x_{\text{opt},i}, \ldots, x_{\text{opt},j}),\\
&U_{\text{track}} = (u_{\text{opt},i}, \ldots, u_{\text{opt},j-1}),\\
&\text{with: } i = k \bmod M,\\
&\qquad j = \Big((i + N - 1) \bmod M\Big) + 1,
\end{aligned}$$

where i, j can be regarded as the indexes to go through the whole optimal trajectory one step at a time and in a periodic fashion. The specific results for generating X_{opt} and U_{opt} can be examined in Chapter 3 and no more details about them will be given here. Furthermore, the selection of the weighting matrices will be explained in Section 5.2.4.

5.2.2 Dynamics

Considering that, in order to ensure safer flight conditions, the original controller implemented a rate-limiter on the control δ, the system dynamics (originally defined by Equation 1.3 and 1.4) have to be extended in order to keep this safety limit in our NMPC approach. In particular, the control δ has to be defined as a system state and its rate of change $\dot{\delta}_{\text{c}}$ as a system control; then, by setting boundaries in $\dot{\delta}_c$, the rate limiter of the original controller can be replicated. As a result, the extended dynamics can be defined as:

$$\begin{aligned}
\dot{\psi} &= g_{\text{k}} v_{\text{a}} \delta + \dot{\varphi} \cos\vartheta,\\
\dot{\varphi} &= -\frac{v_{\text{a}}}{l \sin\vartheta} \sin\psi,\\
\dot{\vartheta} &= -\frac{v_{\text{w}}}{l} \sin\vartheta + \frac{v_{\text{a}}}{l} \cos\psi,\\
\dot{l} &= v_{\text{winch}},\\
\dot{\delta} &= \dot{\delta}_{\text{c}},\\
\text{with:}\quad v_{\text{a}} &= v_{\text{w}} E \cos\vartheta - \dot{l} E,
\end{aligned} \tag{5.2}$$

where the two system parameters, E and g_{k}, are chosen to be 5 and 0.01 respectively considering the results of a previous parameter estimation study. Moreover, the above dynamics were discretized by means of a fourth order implicit Runge-Kutta integrator with four integration steps.

5.2.3 Constraints

Considering that the NMPC scheme has to comply with the constraint structure given by (5.1b)-(5.1e), it was decided that the only required path constraints were based on the safety boundaries of the original controller formulation. In that approach, the controller implemented upper and lower limits on the controls v_{winch} and δ as well as in the rate of δ. Therefore, to imitate this behavior, only the following constraints are considered:

$$\begin{array}{rcccl} \delta_{\min} & \leq & \delta & \leq & \delta_{\max}, \\ v_{\text{winch,min}} & \leq & v_{\text{winch}} & \leq & v_{\text{winch,max}}, \\ \dot{\delta}_{\text{c,min}} & \leq & \dot{\delta}_{\text{c}} & \leq & \dot{\delta}_{\text{c,max}}. \end{array}$$

Furthermore, in order to select the bound values, we considered that in the original formulation they were imposed due to hardware limitations and concluded as a result that they should therefore be respected in our approach. The specific bound values are represented in Table 5.1

Table 5.1: NMPC control bounds.

Boundary	$\delta_{\min}$	$\delta_{\max}$	$v_{\text{winch,min}}$	$v_{\text{winch,max}}$	$\dot{\delta}_{\text{c,min}}$	$\dot{\delta}_{\text{c,max}}$
Value	-0.7	0.7	$-5\,\text{m/s}$	$3.5\,\text{m/s}$	-0.6	0.6

It is important to remark that including only control bounds was a decision which was initially taken in order to achieve a good trade-off between safety and NMPC simplicity. Nevertheless, during the different tests that were performed, we always checked whether including state constraints would improve the NMPC stability; however, since no remarkable difference was ever observed, they were kept out of the algorithm.

Finally, the NMPC also defines the constraints to ensure correct dynamics. In particular, it discretizes the dynamics by using a function per time step, $\Phi_k(t, x_k, u_k)$, that models the state evolution after a time t when starting at x_k and applying a constant control u_k. As a consequence, if the controller time step is Δt, the NMPC dynamical constraints can be defined as:

$$x_{k+1} = \Phi_k(\Delta t, x_k, u_k). \tag{5.3}$$

5.2.4 Numerical Algorithms and Parameters

In order to implement an NMPC controller, there is a set of numerical algorithms and parameters that have to be selected. A clear example would be the already chosen qpOASES as the QP solver for each NMPC iteration; however, the rest of them were not yet defined so they will be described in this section.

Dynamics Discretization

In order to model the continuous dynamics given by (5.2) with an equivalent discrete formulation $\Phi_k(t, x_k, u_k)$, and then, model the dynamical constraints defined by Equation 5.3, we decided to use multiple shooting (Section 2.4) and integrate the state at each time interval by a fourth order implicit Runge-Kutta integrator with four time steps.

The choice of an implicit integrator over an explicit one was done due to concerns on the system stiffness. It is relevant to point out that, before taking this decision, the viability of such implementation was studied taking into account the increase in computational power demand of implicit methods with respect to explicit ones. For a better understanding on implicit and explicit integrators, stiff dynamics and integration stability, [28, Chapter 11] provides an excellent reference.

Hessian Approximation

As it has been seen in Section 4.3, the solution of a continuous optimization problem is determined by the KKT conditions. In particular, in order to solve the root finding problem given by (4.4a)-(4.4e), the Hessian of the Lagrangian $\nabla^2_Y \mathcal{L}(Y, \lambda, \mu)$ has to be computed. However, Hessian computations suffer from a big drawback: second order derivatives are very expensive to compute and they can prevent the controller from meeting the real time requirements.

To overcome this limitation, some techniques to calculate the Hessian based on first order derivatives have been developed. In particular, since the method to solve the KKT conditions using the exact Hessian is known as *Newton method*, this set of approximations was called *quasi-Newton methods* [45]. One of these methods, called the *Gauss-Newton approximation (GN)*, is used when the objective is a least-squares function, i.e. $\|R(Y)\|^2_W = \|W^{1/2}R(Y)\|^2_2$, and approximates the Hessian by:

$$\nabla^2_Y \mathcal{L}(Y, \lambda, \mu) = \Big(W^{1/2}\nabla_Y R(Y)\Big)\Big(W^{1/2}\nabla_Y R(Y)\Big)^\top.$$

Since according to Equation (5.4a) the cost function can be written as the least squares function $\|Y - Y_{\text{track}}\|^2_P$, the GN approach seems to be a good choice for approximating the Hessian. In particular, considering GN, the Hessian approximation is described by:

$$\nabla^2_Y \mathcal{L}(Y, \lambda, \mu) = \Big(P^{1/2}\nabla_Y (Y - Y_{\text{track}})\Big)\Big(P^{1/2}\nabla_Y (Y - Y_{\text{track}})\Big)^\top = \Big(P^{1/2}\mathbb{I}\Big)\Big(P^{1/2}\mathbb{I}\Big)^\top = P.$$

Furthermore, in order to obtain a more robust approximation, the *Levenberg-Marquardt* approximation [46] was included:

$$\nabla^2_Y \mathcal{L}(Y, \lambda, \mu) = P + \alpha\mathbb{I},$$

where $\alpha\mathbb{I}$ is a regularization term to ensure positive definiteness in case the Hessian is ill posed. For the NMPC, α was chosen to be 10^{-6}.

Predictive Horizon and Time Step

The most critical parameters to ensure feasibility of a real time implementation are the predictive horizon N, T and the controller time step Δt.

Since the current Skysails controller, observer and communication protocols are already implemented with a time interval of 100 ms, the NMPC had to stick to this timing and make sure that the predictive horizon was short enough to make $\Delta t = 100\,\text{ms}$ a feasible implementation.

Considering that at 10 m/s the kite needs about 6 s to perform a half lemniscate cycle, we decided to use $T = 6\,\text{s}$ as the predictive horizon; in particular, it was regarded that this period was long enough to ensure system stability and account for enough future behavior, but also short enough to achieve a real time implementation. As a result, if T is chosen to be 6 s and Δt is required to be 100 ms, the number of points of the horizon has to be selected as $N = 60$.

Considering the above choice, a first test result was performed and it was observed that the NMPC computation time per iteration t_c was roughly 10-20 times smaller than Δt. Therefore, it was concluded that the selected T and N were small enough.

Weighting Matrices

A second set of critical parameters are the weighting matrices Q, R and Q_N. In order to model them, their two main functionalities have to be considered:

- The weighting matrices have to normalize the different errors. Whenever minimizing the norm of a vector which has elements with different dimensions and/or units, all the vector quantities should be normalized.
- Once the values are within the same range, the weights can be modified to increase the importance of a variable in the tracking problem.

Considering these two weighting matrices purposes, Q and R can be computed in two steps:

$$Q = \underbrace{\begin{bmatrix} \frac{1}{\sigma^2(\psi)} & & & & & \\ & \frac{1}{\sigma^2(\varphi)} & & & & \\ & & \frac{1}{\sigma^2(\vartheta)} & & & \\ & & & \frac{1}{\sigma^2(l)} & & \\ & & & & \frac{1}{\sigma^2(\delta)} \end{bmatrix}}_{Q_1} \underbrace{\begin{bmatrix} w_\psi & & & & & \\ & w_\varphi & & & & \\ & & w_\vartheta & & & \\ & & & w_l & & \\ & & & & w_\delta \end{bmatrix}}_{Q_2},$$

$$R = \underbrace{\begin{bmatrix} \frac{1}{\sigma^2(v_{\text{winch}})} & \\ & \frac{1}{\sigma^2(\dot{\delta})} \end{bmatrix}}_{R_1} \underbrace{\begin{bmatrix} w_{v_{\text{winch}}} & \\ & w_{\dot{\delta}} \end{bmatrix}}_{R_2}.$$

As it can be seen from the above equation, Q_1 and R_1 are obtained as a first step to perform the normalization; in particular, they are computed as diagonal matrices with the diagonal elements equal to the inverse of the variance of each tracking variable; as a result, since the tracking reference trajectory is constant, they can be set at the beginning and not be modified. Considering the specific values of the tracking reference trajectory, they are defined by:

$$Q_1 = \text{diag}(0.4650,\ 1.7972,\ 2.7055,\ 0.0004,\ 14.2066), \qquad R_1 = \text{diag}(0.1169,\ 9.5951).$$

On the other hand, Q_2 and R_2 set the relative weight of each state and control; thus, they will be the matrices used for tuning the NMPC and we will refer to them whenever the weighting matrices should be modified. Considering that the state tracking is more important than following optimal controls, the matrices are initialized as:

$$Q_2 = \text{diag}(1,\ 1,\ 1,\ 1,\ 0.001), \qquad R_2 = \text{diag}(0.001,\ 0.001).$$

In contrast with Q and R, selecting the matrix Q_N might be a more critical step: since it penalizes the last state x_N, using a big enough penalty might ensure that a steady point is reached at the end of the horizon, convergence is improved and stability increased; however, on the other hand, too big values on Q_N might also result in very small penalties on the other states and controls and lead to a bad NMPC performance. In theory, as shown in [32], $Q_N(t)$ should be a time dependent matrix that is computed from a time periodic differential Riccati equation. In this particular case, this approach would mean linearize the system dynamics at each of the points $(x_{\text{opt},k},\ u_{\text{opt},k})$ of the periodic optimal trajectory, and then,

use these linearized systems to compute the matrices $Q_N(k\Delta t) = P_k$ as the solution of the periodic differential Riccati equation. Ideally, this approach should increase the system stability and robustness; however, due to the long horizon $N = 60$, it was observed that selecting $Q_N(k\Delta t) = P_k$ or selecting $Q_N(k\Delta t) = Q_N = Q$ had no real repercussion in the performance of the NMPC; as a result and for the sake of simplicity, we opted for the latter approach $Q_N = Q$ for defining the NMPC weighting matrix.

5.2.5 Formulation

Considering the algorithms and parameters defined in the previous section, the OCP that the NMPC solves at each time iteration is represented by:

$$\begin{aligned}
\underset{Y}{\text{minimize}} \quad & \|Y - Y_{\text{track}}\|_P^2 && && \text{(5.4a)}\\
\text{subject to} \quad & \Phi_k(\Delta t, x_k, u_k) - x_{k+1} = 0, && k = 0, \dots, N-1, && \text{(5.4b)}\\
& v_{\text{winch,min}} \le v_{\text{winch},k} \le v_{\text{winch,max}}, && k = 0, \dots, N-1, && \text{(5.4c)}\\
& \dot{\delta}_{\text{c,min}} \le \dot{\delta}_{\text{c},k} \le \dot{\delta}_{\text{c,max}}, && k = 0, \dots, N-1, && \text{(5.4d)}\\
& \delta_{\text{min}} \le \delta_k \le \delta_{\text{max}}, && k = 0, \dots, N, && \text{(5.4e)}\\
& x_0 - \bar{x}_0 = 0, && && \text{(5.4f)}
\end{aligned}$$

where:

$$\begin{aligned}
&Y = \big(X, U\big), && Y_{\text{track}} = \big(X_{\text{track}}, U_{\text{track}}\big)\\
&X = \big(x_0, \cdots, x_N\big), && X_{\text{track}} = \big(x_{\text{track},0}, \cdots, x_{\text{track},N}\big),\\
&U = \big(u_0, \cdots, u_{N-1}\big), && U_{\text{track}} = \big(u_{\text{track},0}, \cdots, u_{\text{track},N-1}\big),\\
&x_k = \big[\psi_k, \varphi_k, \vartheta_k, l_k, \delta_k\big]^\top, && u_k = \big[\dot{\delta}_k, v_{\text{winch},k}\big]^\top,\\
&P = \text{diag}\big(Q,\ Q,\ \underbrace{\cdots}_{(N-4)\times Q},\ Q,\ Q_N,\ R,\ \underbrace{\cdots}_{(N-3)\times R},\ R\big).
\end{aligned}$$

5.3 System Simulator

In order to test and tune the NMPC, we implemented a system simulator that at each time step receives from the NMPC the optimal control u_0^*, simulates the system with u_0^* and finally sends back the new current state $\bar{x}_0$ to the NMPC. Moreover, in order to test the NMPC against real life conditions, in subsequent sections of Chapter 6 this simulator will be expanded by including the two main sources of disturbances:

(i). A NMPC model lacking some real life effects.

(ii). An observer that provides an imperfect estimation of the current state.

5.3.1 First Simulation Results

As a first necessary step to test whether the numerical algorithm, the NMPC code and the interface between simulator and controller were correctly implemented, an experiment without simulation of disturbances was conducted.

In particular, the NMPC was set to track the optimal trajectories obtained for $v_w = 10\,\mathrm{m/s}$ using a simulator with the same wind speed. The results of this experiment are depicted in Figure 5.1: since the NMPC perfectly tracks the reference trajectories of the four system states ψ, φ, ϑ and l (the difference between the reference and the NMPC trajectories can not be distinguished), it can be inferred that the implementation of the numerical algorithms is correct and proceed to the next sections where a robust NMPC will be designed.

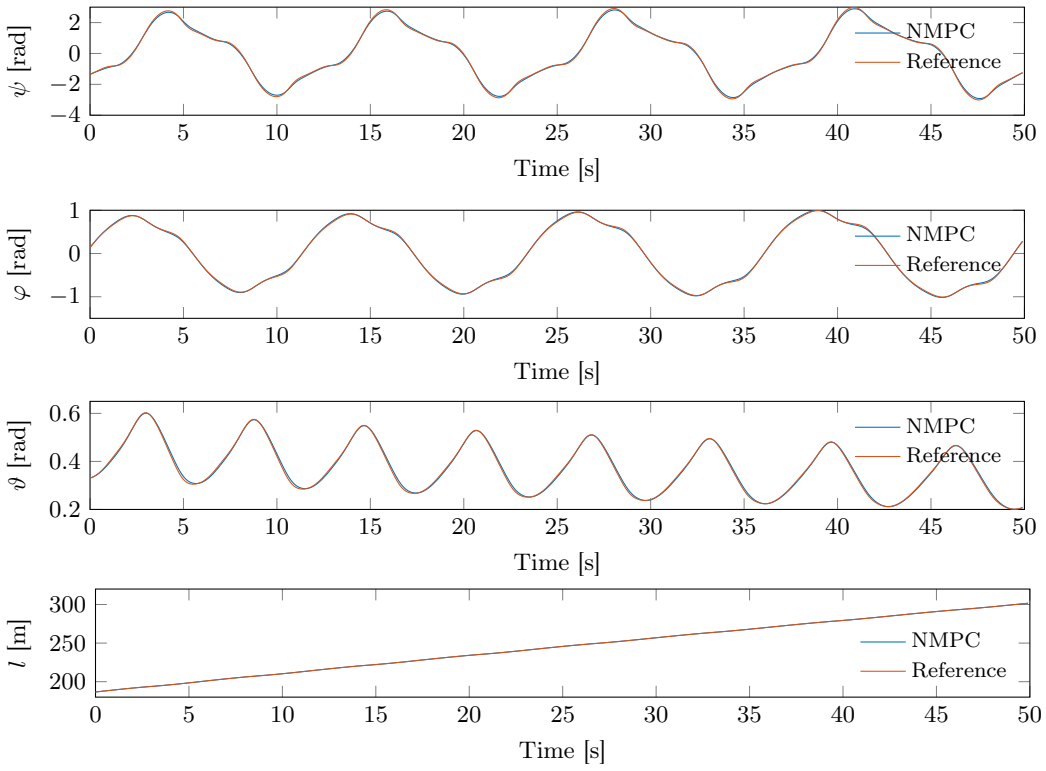


FIGURE 5.1: First NMPC test using a perfect observer.

5.3.2 Evaluation Metrics

Considering that in subsequent sections the NMPC will be simulated against several disturbances, it is necessary to define here a set of different metrics that will assess the controller performance when facing these perturbations; in particular, the R^2 will be defined as a measure of the tracking quality and the Loyd factor η_{Loyd} as a power efficiency indicator.

R^2 Goodness of Fit

Given a predefined trajectory V_{pre}, the R^2-goodness of fit of another trajectory V with respect to V_{pre} is defined as:

$$R^2 = \left(1 - \frac{\|Y - Y_{\mathrm{pre}}\|^2}{\|Y_{\mathrm{pre}}\|^2}\right) \cdot 100. \tag{5.5}$$

As its name indicates, this metric is in general used to assess the fitting performance of some algorithm. In our specific case, we intend to use it in order to measure the tracking performance of the NMPC by measuring the goodness of fit between the tracking reference states ψ_{track}, φ_{track}, ϑ_{track} and l_{track} and the NMPC-driven states ψ, φ, ϑ and l. However, since the trajectories of each of these four states have different ranges of values, all the trajectory vectors have to be normalized in order to have a meaningful indicator of the fit:

$$\psi' = \frac{\psi}{\mu_{|\psi|}}, \qquad \varphi' = \frac{\varphi}{\mu_{|\varphi|}}, \qquad \vartheta' = \frac{\vartheta}{\mu_{|\vartheta|}}, \qquad l' = \frac{l}{\mu_{|l|}},$$
$$\psi'_{\text{track}} = \frac{\psi_{\text{track}}}{\mu_{|\psi|}}, \qquad \varphi'_{\text{track}} = \frac{\varphi_{\text{track}}}{\mu_{|\varphi|}}, \qquad \vartheta'_{\text{track}} = \frac{\vartheta_{\text{track}}}{\mu_{|\vartheta|}}, \qquad l'_{\text{track}} = \frac{l_{\text{track}}}{\mu_{|l|}},$$

with $\mu_{|\gamma|}$ representing the mean of the absolute value of the trajectory γ_{track}.

Once the values are normalized, $V = (\psi',\ \varphi',\ \vartheta', l')$ and $V_{\text{pre}} = (\psi'_{\text{track}}, \varphi'_{\text{track}}, \vartheta'_{\text{track}}, l'_{\text{ref}})$ can be defined, and then, Equation (5.5) can be directly used to provide a general indicator of how much the NMPC trajectory differs from the optimal one.

Loyd Factor

The Loyd factor η_{Loyd}, which was already defined in Equation (1.2), is an indicator of how much power the system is harvesting compared to an ideal maximum power. In the following sections, this metric will be used in order to quantify the NMPC efficiency. In particular, in order to do a meaningful comparison, two previously obtained η_{Loyd} should be recalled:

(i). The 35.3% efficiency of the optimal tracking trajectory.

(ii). The η_{Loyd} of the original Skysails controller which was approximately 18%.

It is important to remark that, in order to avoid local effects of a single flight kite period, these indicators will be computed using a test flight of 10000 time points, which is roughly equivalent to 8-9 flight periods.

5.4 Control Delay

An assumption done in order to derive the equations of motion given by (5.2) was to assume that the control δ would modify the system dynamics right after it was selected; nevertheless, in real life experiments it has been observed that the kite reacts to the control δ with a lag of approximately 0.5 s.

Since in our NMPC framework half second retardment is equivalent to a delay of 5 control actions, it is expected that such a delay might have the potential to make the controller unstable. As a result, in order to test how such a mismatch affects the stability of the NMPC, the option to delay the controls δ by any delay d was implemented in the simulator.

Figure 5.2 and Table 5.2 depicts the results of testing the NMPC against a 0.5 s delay; it can be observed that, despite not crashing the kite, the delay has the potential to make the NMPC unstable and unable to track the optimal trajectories. On top of that, it has to be considered that, so far, no disturbances were introduced, therefore, in the event of having some additional real perturbations, it is quite likely that the controller would become unstable and the kite would crash.

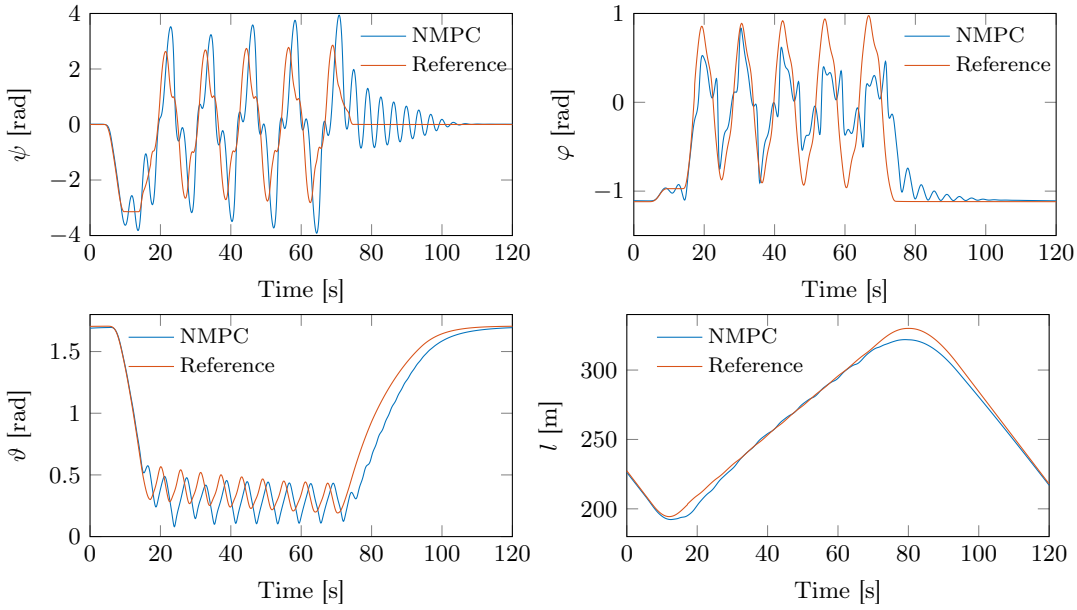


FIGURE 5.2: NMPC tracking performance when facing a 0.5 s delay on δ.

TABLE 5.2: NMPC metrics considering a δ delay of half second.

R^2	η_{Loyd}
32.88%	30.21%

As a final note, it is important to remark that a dynamic delay is ultimately a type of real life disturbance; as a result, it could be argued that the best place for its discussion is in Chapter 6, where all the other known disturbances will be covered. Nevertheless, since the dynamic delay will be explicitly modeled within the NMPC, it was decided that discussing it in this chapter, together with the NMPC implementation, was more sensible.

5.4.1 Delay Differential Equation

In order to solve the previous issue, we decided to model the delay as part of the kite dynamics by using a *delay differential equation (DDE)* [47, 48].

In a general ODE or *partial differential equation (PDE)*, the rate of change $\dot{s}(t)$ of any state $s(t)$ at time t can only depend on variables at the same time point t; by contrast, DDEs can model systems where the rate of change $\dot{s}(t)$ depends on the state values at prior times. A general DDE formulation is given by:

$$\dot{s}(t) = \Phi(t, s(t), s_t), \qquad \text{with: } s_t = \{s(\tau) : \tau \leq t\}.$$

The main problem of DDEs is that, like PDEs, some of their variables are infinite dimensional. In order to confront this issue, the dynamics of the infinite dimensional variable s_t should be first modeled using a PDE, and then, a PDE specific algorithm should be used to discretize s_t at several time nodes in the past and obtain a set of ODEs that model the dynamics of s_t at these discretized time nodes.

In particular, in order to model the delay with a PDE, the time delay shall be regarded as a *pipe flow*, where the position inside the pipe represents a specific time in the past and the flow at that location the state at that specific time. Then, in order to model a delay that varies between $[0, d]$, i.e. $(t - \tau) \in [0, d]$, a variable $z \in [0, d]$ should be used to model the position inside the pipe (the time in the past) and a second variable $v(z, t)$ to represent the flow within the pipe (the state at that past time). Then, the desired variable $s_t = s(\tau)$ is equal to $v(t - \tau, t)$ and its dynamics are obtained by solving the following PDE:

$$\frac{\partial v}{\partial t} = -\frac{1}{d}\frac{\partial v}{\partial z}. \tag{5.6}$$

The variable $v(z, t)$, which represents the state s at a time point in the past $(t-z)$, can be regarded as the continuous past memory of the state s flowing backwards in the pipe of time; as a result, the entrance of the pipe would represent the state s at time t, i.e. $v(0, t) = s(t)$, and the output would be state at some point in past $t - d$, i.e. $v(d, t) = s(t - d)$, where d is the maximum time delay.

To include and solve Equation (5.6) within the dynamical equation of motions, the *method of lines (MOL)* [49] for PDEs have to be used. The main conceptual idea of MOL is based on a three steps algorithms:

(i). In a first step, all the continuous states except one are discretized. The standard convection in PDEs is to discretize the spatial derivatives and keep the time derivatives as continuous functions.

(ii). Then, in a second step, MOL uses algebraic approximations to model the spatial derivatives of these discretized variables.

(iii). Finally, the result of this procedure is a system of ODEs that approximates the original PDE at discrete nodes and which can be easily integrated with modern ODE algorithms.

In the specific case of Equation (5.6), the described procedure can be implemented by first, discretizing the state $v(z, t)$ in a set of $m + 1$ nodes, $V(t) = [v_0(t), \ldots, v_m(t)]^\top$, equally separated by a distance $\frac{d}{m}$; then, by approximating the spatial derivative $\dot{v}_k(t)$ to generate the system of ODEs. In particular, an algebraic approximation for such derivatives is given by finite differences as [47]:

$$\dot{v}_k(t) \approx -\frac{v_k - v_{k-1}}{\Delta z} = -\frac{v_k - v_{k-1}}{d/m}., \quad k = 1, \ldots m,$$

where, in order to obtain the missing ODE for $\dot{v}_0(t)$, the condition $\dot{s}(t) = \dot{v}_0(t)$ can be used so that:

$$\dot{s}(t) = \dot{v}_0(t) \approx \Phi(t, s(t), V(t)).$$

The above equations represent the system of ODEs that model the time evolution of the states at different time points in the past, and as a consequence, if including them as part of the NMPC dynamics, delayed values of any state or control become available to the controller.

5.4.2 DDE Implementation

Regard the system dynamics defined by Equation (5.2) but with a delay d on the actuation δ:

$$
\begin{aligned}
\dot{\psi}(t) &= g_{\mathrm{k}} v_{\mathrm{a}} \delta(t-d) + \dot{\varphi}(t)\cos\vartheta(t),\\
\dot{\varphi}(t) &= -\frac{v_{\mathrm{a}}(t)}{l(t)\sin\vartheta(t)}\sin\psi(t),\\
\dot{\vartheta}(t) &= -\frac{v_{\mathrm{w}}(t)}{l(t)}\sin\vartheta(t) + \frac{v_{\mathrm{a}}(t)}{l(t)}\cos\psi(t),\\
\dot{l}(t) &= v_{\mathrm{winch}}(t),\\
\dot{\delta}(t) &= \dot{\delta}_{\mathrm{c}}(t),\\
\text{with:}\quad v_{\mathrm{a}}(t) &= v_{\mathrm{w}}(t) E\cos\vartheta(t) - \dot{l}(t)E.
\end{aligned}
$$

Considering the theory of DDEs described above, this system can be easily approximated by the following system of ODEs:

$$
\begin{aligned}
\dot{\psi}(t) &= g_{\mathrm{k}} v_{\mathrm{a}} \delta_m(t) + \dot{\varphi}(t)\cos\vartheta(t),\\
\dot{\varphi}(t) &= -\frac{v_{\mathrm{a}}(t)}{l(t)\sin\vartheta(t)}\sin\psi(t),\\
\dot{\vartheta}(t) &= -\frac{v_{\mathrm{w}}(t)}{l(t)}\sin\vartheta(t) + \frac{v_{\mathrm{a}}(t)}{l(t)}\cos\psi(t),\\
\dot{l}(t) &= v_{\mathrm{winch}}(t),\\
\dot{\delta}(t) &= \dot{\delta}_{\mathrm{c}}(t),\\
\dot{\delta}_1(t) &= -\frac{\delta_1(t)-\delta(t)}{d/m},\\
\dot{\delta}_2(t) &= -\frac{\delta_2(t)-\delta_1(t)}{d/m},\\
\vdots \quad &= \qquad \vdots\\
\dot{\delta}_m(t) &= -\frac{\delta_m(t)-\delta_{m-1}(t)}{d/m},\\
\text{with:}\quad v_{\mathrm{a}}(t) &= v_{\mathrm{w}}(t) E\cos\vartheta(t) - \dot{l}(t)E.
\end{aligned}
$$

In order to perform a first test, we selected $m = 6$ to implement the above model, and then, the controller was tested. The results of such an experiment can be seen in Figure 5.3 and Table 5.3, where as before, the table shows the results of a 9-periods flight and the figure represents one of these periods.

Table 5.3: Performance of a δ delay using a DDE.

R^2	η_{Loyd}
98.59	33.78

By evaluating the above results a main conclusion can be drawn: modeling the delay by extending the dynamics with a DDE stabilizes the NMPC and leads to nearly the same performance as if the delay was non-existent; in particular, this is an excellent result since it practically removes every delay effects from the controller.

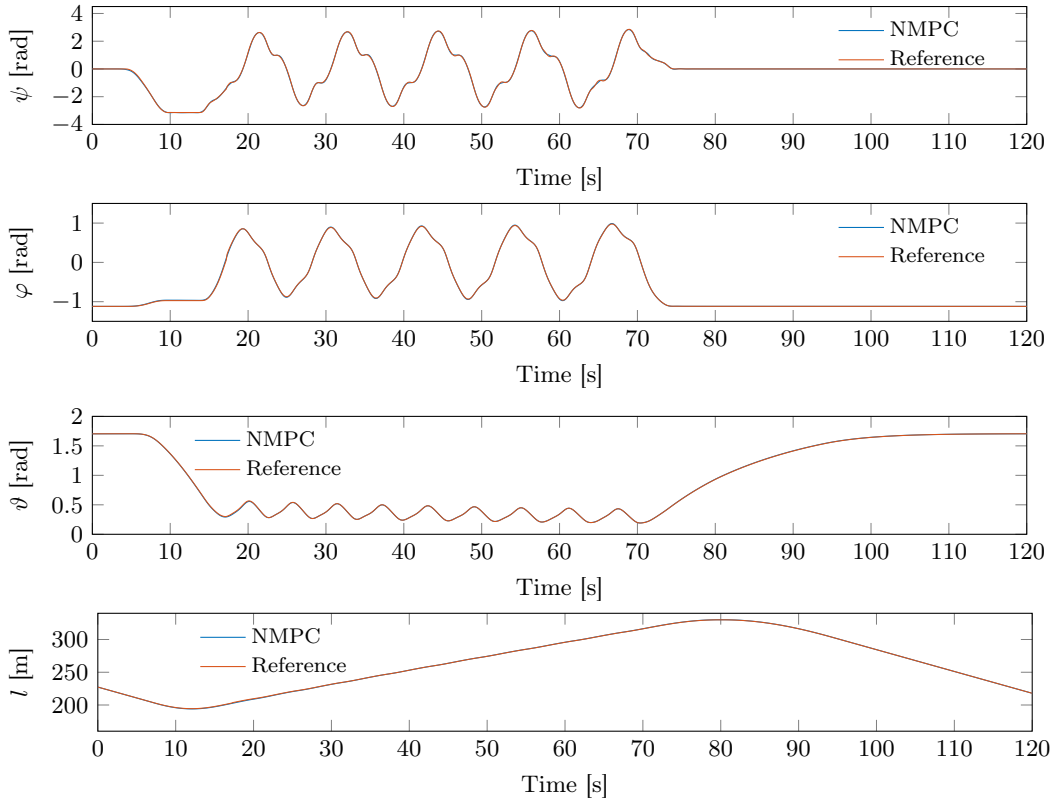


FIGURE 5.3: NMPC with 0.5 s delay on δ and with the delay modeled by a DDE.

DDE Complexity

A very critical step when designed the DDE extension is to select the right number m of states δ_k: a high number leads to a higher accuracy of the model but at the same time it can make the NMPC computation very expensive; in order to choose the optimal number of extra states for our specific application, the NMPC performance was compared for different m values by means of the R^2 goodness of fit and the computation time per iteration t_c. From Figure 5.4 and Table 5.4 it can be observed that the optimal value for m is around 6: higher values would make t_c more expensive without improving the fitting and smaller values would decrease the fitting performance; as a result, $m = 6$ was chosen for our application.

TABLE 5.4: NMPC performance as a function of the number m of extra states δ_k.

m	2	3	4	5	6	7	8	10
R^2 [%]	95.31	97.05	97.97	98.38	98.59	98.69	98.71	98.66
t_c[ms]	5.2	5.1	6.2	6.2	6.3	7.5	9.2	11.6

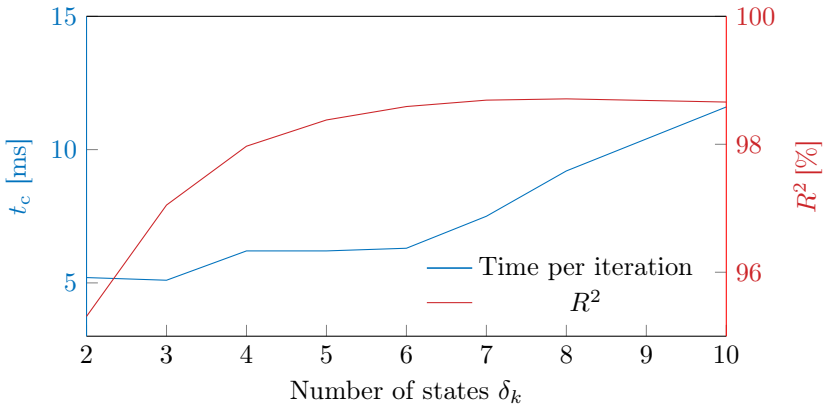


FIGURE 5.4: NMPC metrics as a function of m.

5.4.3 Delay Mismatch

When analyzing the delay effect in the previous sections it was assumed to be deterministic and equal to 0.5 s; however, in a real flight scenario, the nature of the delay and how it might affect to the total delay duration have to be considered. In particular, it should be regarded that the delay appears as the combination of two different effects:

(i). Due to hardware limitations, a deterministic communication delay of 0.2 s is present.

(ii). A second delay is produced due to aerodynamics effects; in particular, this lag is not constant but might achieve a maximum value of 0.3 s and a minimum of 0.1 s.

As a result, since a real delay will oscillate between 0.3 s and 0.5 s, the effects of this variation must be studied.

Table 5.5 illustrates the results of controlling a system with a 0.3 s delay that is modeled within the NMPC as 0.5 s; by comparison with Table 5.3 it can be observed that, while the Loyd efficiency and R^2 become slightly worse, the decrease is too small to raise any real concern regarding controller stability.

TABLE 5.5: NMPC performance when overestimating the δ delay.

R^2	η_{Loyd}
94.16	33.25

5.4.4 Conclusion

As it has been shown in this section, delays are a very critical effect that has to be modeled within the controller. Moreover, if perfect estimation can not be done, a delay with the largest possible value can be assumed to still obtain a close performance to exact estimation.

As a result, in order to have realistic simulations, a 0.5 s delay will be permanently included in the simulator so that all testing scenarios in the following chapters will model the most realistic flight condition.

Chapter 6

Simulation of Real Conditions

In the previous chapter, the core of the NMPC was designed and its performance tested assuming ideal conditions. Moreover, a control delay as a first disturbance was included and then solved by using a DDE.

The aim of this section is to continue this work by modeling the remaining real flight disturbances and adapting the NMPC to stay stable against them. In order to perform this task in a consistent manner, the chapter will be divided in several sections where each of them will cover one of the following effects:

(i). A real wind profile where the wind speed is not constant but includes wind gusts.

(ii). Parameter mismatches to model that in real conditions the glide ratio E and the steering constant g_{k} are not the ideal estimated parameters.

(iii). A wind direction profile using real wind data.

(iv). A offset error on the control δ.

(v). A realistic observer that makes estimation errors.

Finally, to avoid repeating the same information, it is important to point out that all the performed tests in the coming sections will use as tracking optimal trajectory the solution of the offline OCP for $v_{\mathrm{w}} = 10\,\mathrm{m/s}$.

6.1 Wind Gusts

To solve the different OCPs in Chapter 3 as well to perform the first tests in Chapter 5, a wind profile with constant wind speed was assumed. Nevertheless, in real life, wind is by far not steady and two main effects should be considered:

(i). Wind gusts: variations of the wind speed in a short time scale (seconds or 1-2 minutes). Predicting wind gusts is not an easy task: their behavior is stochastic and anticipating these variations within seconds is almost impossible. However, the average wind speed in this small time scale is roughly constant.

(ii). Long term variations: in contrast to wind gusts, whenever looking at wind in a long time scale (several minutes, hours or days) the wind speed average can change dramatically. However, in contrast to wind gusts, obtaining an estimator of this averaged wind speed can be easily done by low pass filtering wind measurements.

In this section, the first case will be treated by proving NMPC stability against wind gusts. The second case will be considered later, in Chapter 7, where a strategy to obtain optimal trajectories in real time as a function of the wind speed will be proposed.

6.1.1 Wind Profile Generation

A specific strategy to model wind gusts is to consider that the wind speed can be modeled by a Kaimal power spectrum. In particular, based on the mentioned principle, [50] describes a specific algorithm to obtain a discrete time series $[v_{\mathrm{w},0}, \ldots, v_{\mathrm{w},m}]^\top$ of wind speed values with an average $v_{\mathrm{w}}^{(0)}$.

For the sake of comprehensibility, this section will be confined to prove the validity of this Kaimal model as well as testing the NMPC against the generated wind data; for further details, the methodology for generating wind speed is described in Appendix B.

To prove the validity of the model, the frequency spectrum of real wind measurements obtained for the Skysails kite were compared with the Kaimal spectrum; moreover, the same wind measurements were plotted in the time domain against simulated values from the Kaimal model. The results of these two comparisons can be seen in Figure 6.1: it can be clearly observed that the data generated by this methodology models quite accurately real wind data.

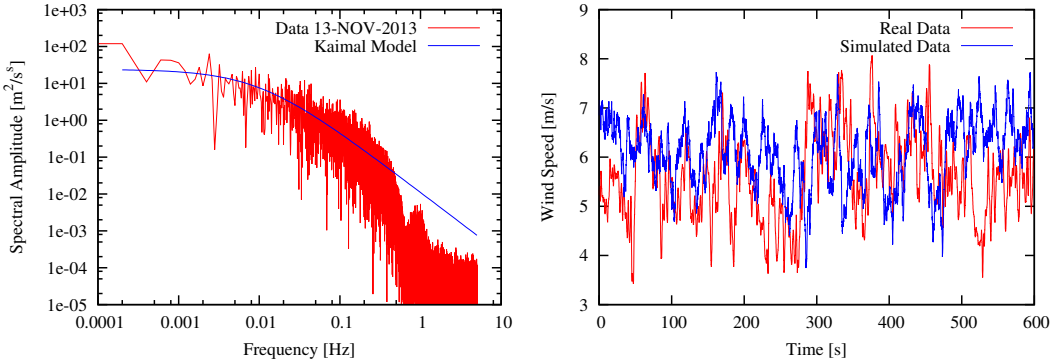


FIGURE 6.1: Left: Kaimal versus measured wind speed spectrum. The deviation for $f > 0.3\,\mathrm{Hz}$ can be explained due to a low pass filter in the kite sensor. Right: generated versus measured wind speed for a period of 10 minutes.

6.1.2 NMPC with Real Wind Profile

In order to test the NMPC stability against real wind gusts, realistic wind profiles are generated with an average wind speed $v_{\mathrm{w}}^{(0)}$ of $10\,\mathrm{m/s}$; this particular $v_{\mathrm{w}}^{(0)}$ selection is important so that it matches the velocity of the optimal reference trajectory. Figure 6.2 depicts an example of such a wind profile.

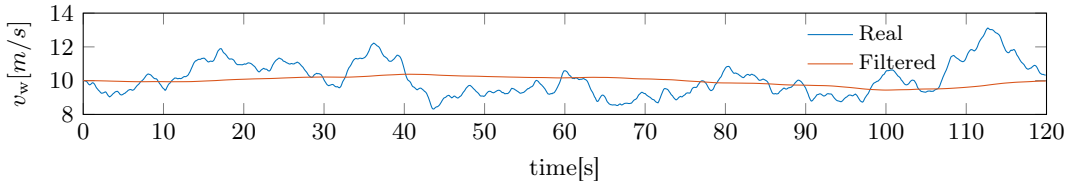


FIGURE 6.2: Wind profile for $v_{\mathrm{w}}^{(0)} = 10\,\mathrm{m/s}$ and a 10% turbulence.

Once the generation was done, the NMPC was tested against a wind profile by considering the worst possible case in the observer. In particular, it was assumed that, since estimation of the real wind speed is a very hard task, the only information available to the NMPC was an average value of the wind velocity during the last 60 seconds; this consideration was done to ensure stability given a poor estimation, i.e. if the NMPC is stable in these extreme conditions it should also be stable given a real observer. The results of the experiment are illustrated in Table 6.1 and Figure 6.3, where the table represents the performance metrics during a 9 cycles flight and the figure illustrates the tracking performance of one of these cycles.

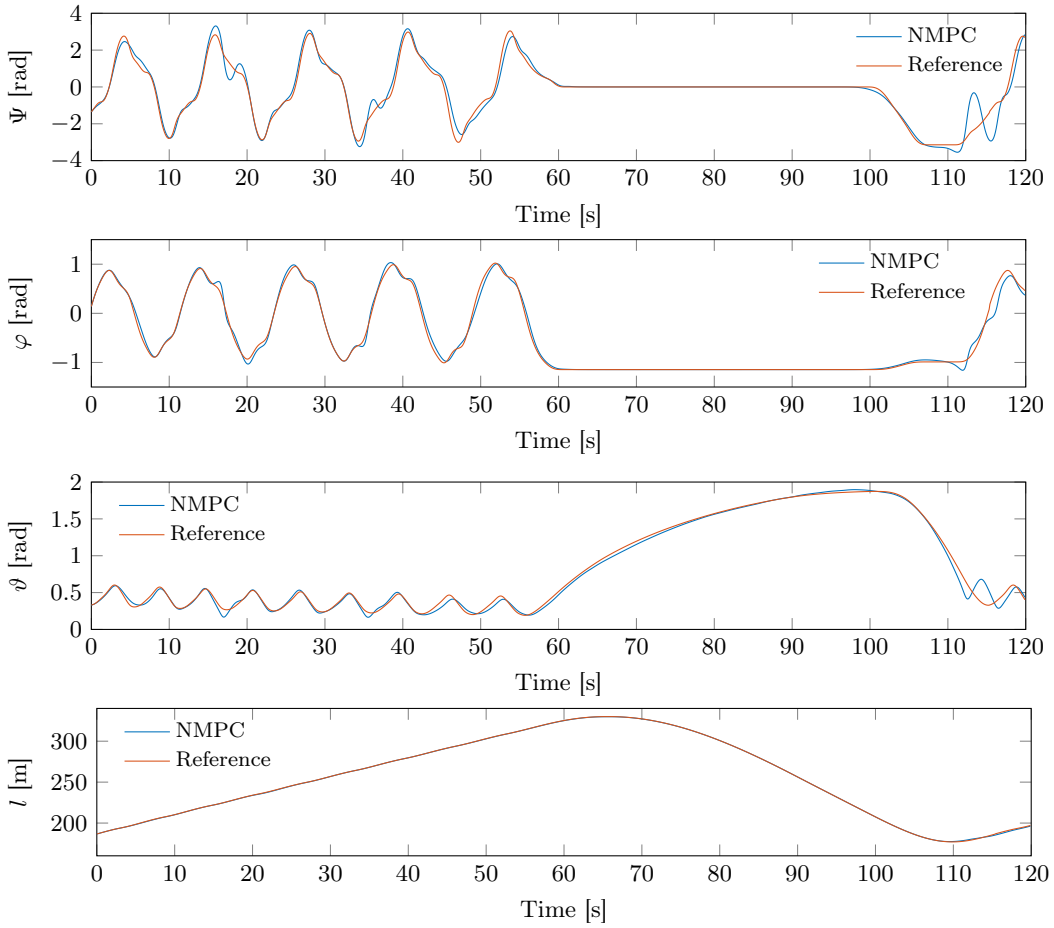


FIGURE 6.3: NMPC controlling system under the influence of a real wind gust.

TABLE 6.1: Metrics for wind gusts.

R^2	η_{Loyd}
82.48	33.25

Looking at these result and comparing them with Table 5.3 and Figure 5.3, three conclusion can be drawn :

- Even though the wind speed can not be estimated, provided that the average wind speed is similar to the velocity of the tracking reference trajectory, the NMPC remains stable against wind gusts.
- Wind gusts seems to have a small impact on the tracking performance: despite a decrease in R^2, the NMPC still tracks the reference optimal trajectory quite good and the small error between the NMPC and the mentioned trajectory is remarkable.
- Wind gusts reduce the system efficiency by roughly 1% leading to a controller that almost doubles the original 18% η_{Loyd}.

As a result, we can claim that the NMPC shows a great performance against short term wind speed variations and that they are not a subject of concern. Nevertheless, as it will be seen in Chapter 7, long term wind speed variations are a very different case that will be treated differently.

Since these wind profiles are a very accurate representation of real situations, they will be included as standard wind conditions in all further tests; furthermore, to have a common ground truth, the same wind profile will be used for every testing scenario. However, in order to account for the fact that wind speed estimation is not possible, the wind speed will be always computed as a 60 s average.

6.2 Parameter Mismatch

As it has been previously mentioned, the NMPC implementation assumes that $E = 5$, $g_{\text{k}} = 0.1$ and that they remain constant during flight conditions. However, it has been observed that, since the glide ratio E and the steering constant g_{k} might oscillate in a real flight environment, this assumption does not hold .

In a perfect scenario this should not be a problem: the parameters would be ideally estimated online and the NMPC model readapted. Nevertheless, online estimation of the kite parameters is not currently implemented in the observer; therefore, it has to be considered that the plant might have some parameter mismatches which are unknown to the NMPC.

In order to study how the previous result affects the NMPC, this section will study the controller performance considering the different parameter mismatches. In particular, in order to carry out this study, it is necessary to know the maximum value of the parameter mismatches; nevertheless, due to a lack of previous data regarding this issue, there are no mismatches references and we have to make an educated guess about the maximum parameters variations. Specifically, based on the following two hypotheses, we decided to assume nominal parameter variations of 10 % and a maximum of 20 %:

(i). The kite model was validated several times using $E = 5$ and $g_{\text{k}} = 0.1$. If the variations of the parameters were larger, they should have influenced these validations in a stronger manner and the assumption of having constant parameters would not have hold.

(ii). If higher variations would occur, the current observer would not be able to perform good and the estimation of the system states would be wrong. Since from past flight tests it has been observed that state estimation is pretty accurate, high parameters mismatches are not likely.

As a result, in order to carry out the NMPC analysis in an organized and consistent manner, the following steps will be executed:

(i). Model the NMPC with the expected values $E = 5$ and $g_\mathrm{k} = 0.01$.

(ii). Study the NMPC performance considering a nominal 10 % parameter mismatch in the system simulator.

(iii). Select the worst scenarios out of the nominal cases and repeat the experiments with the maximum 20% mismatch.

It is important to remark that, considering the independence of the equation $\dot{l} = v_\mathrm{winch}$ with respect to E and g_k, it is reasonable to assume that parameter mismatches will not have a direct and big effect on the tether length l; as a result, when displaying tracking performance, the trajectory of l will be substituted by a 2D projection of the flight trajectory.

Finally, before starting the analysis and in order to interpret the specific results, three concepts shall be first defined:

- We define *nominal dynamics* as a system of equations of motions used for reference purposes. In the NMPC case, they will refer to the kite dynamics with the system parameters $E = 5$ and $g_\mathrm{k} = 0.1$.
- We define *faster dynamics* if, by changing the system parameters, using the same actuation input the dynamics lead to a larger state displacement compared to the nominal dynamics.
- By contrast, we will refer to *slower dynamics* when the opposite behavior appears: for the same actuation input, the dynamics displace the state in a smaller amount compared to the nominal ones.

6.2.1 Nominal Mismatch of 10 %

In order to test the NMPC against the four 10 % mismatch worst case scenarios, the following possible values of E and g_k should be considered:

$$\{[E,\ g_\mathrm{k}]^\top \mid E \in \{4.5,\ 5.5\},\ g_\mathrm{k} \in \{0.09,\ 0.11\}\}.$$

First Nominal Test: $E = 4.5,\ g_\mathrm{k} = 0.09$

The results of this test case are shown in Figure 6.4 and Table 6.2; by looking especially at the ψ angle and the retraction phase on θ, it can be seen that these parameters lead to slower dynamics when compared to the nominal dynamics. The result is a controller that struggles to track the optimal trajectory because the controls that it chooses are never big enough to follow this tracking trajectory; nevertheless, it can also be observed that if the dynamics are slower than expected, the efficiency decreases because the NMPC can not perform the full lemniscates cycle but the controller remains stable.

TABLE 6.2: NMPC performance considering $E = 4.5$ and $g_\mathrm{k} = 0.09$.

R^2	η_Loyd
67.96%	30.95%

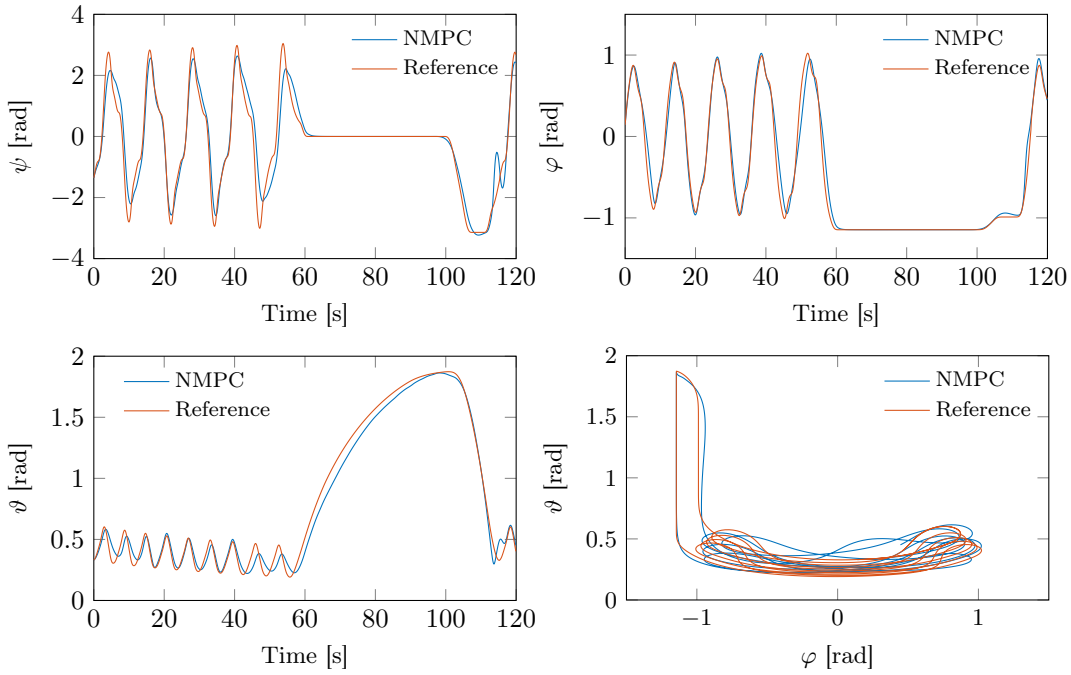


FIGURE 6.4: NMPC tracking with $E = 4.5$ and $g_k = 0.09$.

Second Nominal Test: $E = 4.5$, $g_k = 0.11$

As in the previous testing scenario, this parameter mismatch creates slower dynamics and lead to a NMPC that struggles to follow the tracking trajectory. Therefore, to avoid plot redundancy, in this section the standard plots will be omitted.

By looking at the results in Table 6.3 and comparing them with Table 6.2 it can be observed that: the lower the values of E and g_k the slower the dynamics become and that the importance of E in this phenomena is larger than g_k.

TABLE 6.3: NMPC performance considering $E = 4.5$ and $g_k = 0.11$.

R^2	η_{Loyd}
77.52%	32.20%

Third Nominal Test: $E = 5.5$, $g_k = 0.11$

As before, the results are illustrated in Figure 6.5 and Table 6.4; in this particular case, the opposite behavior of the first and second scenarios can be observed: the dynamics have become faster and the NMPC performs larger motions that it intends.

The result of this situation is a controller that tends to overshoot and that as a consequence has to continuously correct its trajectory. By looking at Figure 6.5 and comparing with Figure 6.4, it can be seen that, despite not being unstable, the controller seems to struggles more whenever the parameters are underestimated.

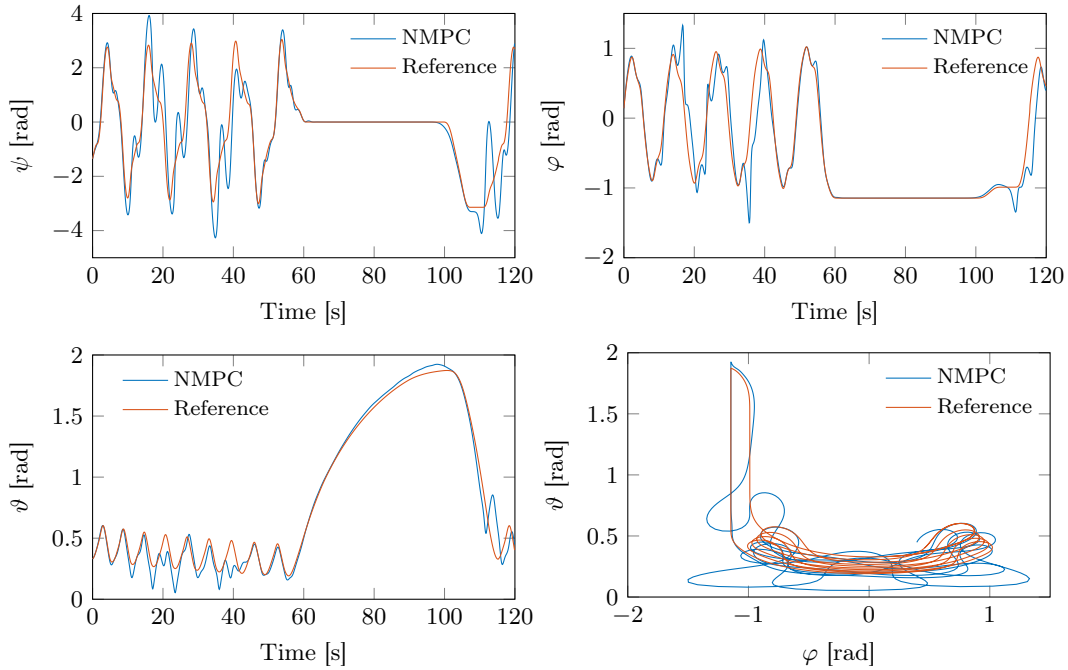


FIGURE 6.5: NMPC tracking with $E = 5.5$ and $g_k = 0.11$.

TABLE 6.4: NMPC performance considering $E = 5.5$ and $g_k = 0.11$.

R^2	η_{Loyd}
60.14%	34.99%

Fourth Nominal Test: $E = 5.5$, $g_k = 0.09$

Since the result of this test scenario are again very akin to the third case, only the NMPC metrics will be depicted to avoid plot redundancy.

As expected, the results in Table 6.5 confirm the assumed hypothesis: the importance of a mismatch in E is relatively larger than in g_k and that the higher the values of E and g_k the faster the dynamics become.

TABLE 6.5: NMPC performance considering $E = 5.5$ and $g_k = 0.09$.

R^2	η_{Loyd}
82.5%	33.99%

Nominal Mismatch Analysis

By analyzing nominal mismatches, the following patterns can be clearly observed:

(i). The higher E or g_k are, the faster the real dynamics become with respect to the NMPC model; as a consequence, the controller assumes a dynamical model that is slower than

the real one causing the NMPC to select too high inputs, overshoot, and struggle quite significantly to follow the tracking reference trajectory.

(ii). By contrast, the lower the real E and g_k are, the slower the real dynamics become; as a result,the NMPC assumes dynamics that are faster than the real ones resulting in a kite performing movements that are smaller than expected. Surprisingly, this effect does not seem to make the system unstable; instead, it just leads to smaller lemniscates and a loss in the power efficiency.

(iii). Mismatches in E seems to be more critical than in g_k.

6.2.2 Maximum Mismatch of 20 %

In order to study the two worst case scenarios under maximal mismatches, it has to be considered that, whenever E and g_k are minimal the dynamics become as slow as possible and, on the contrary, if they are maximized the fastest dynamics are obtained.

First Maximal Test: $E = 4,\ g_k = 0.08$

In Figure 6.6 and Table 6.6 it can be observed that, as already concluded in the previous section, if the NMPC overestimates the parameters (even if it is by a 20 %) the kite flies shorter lemniscates and the system efficiency decreases but the system remains unstable.

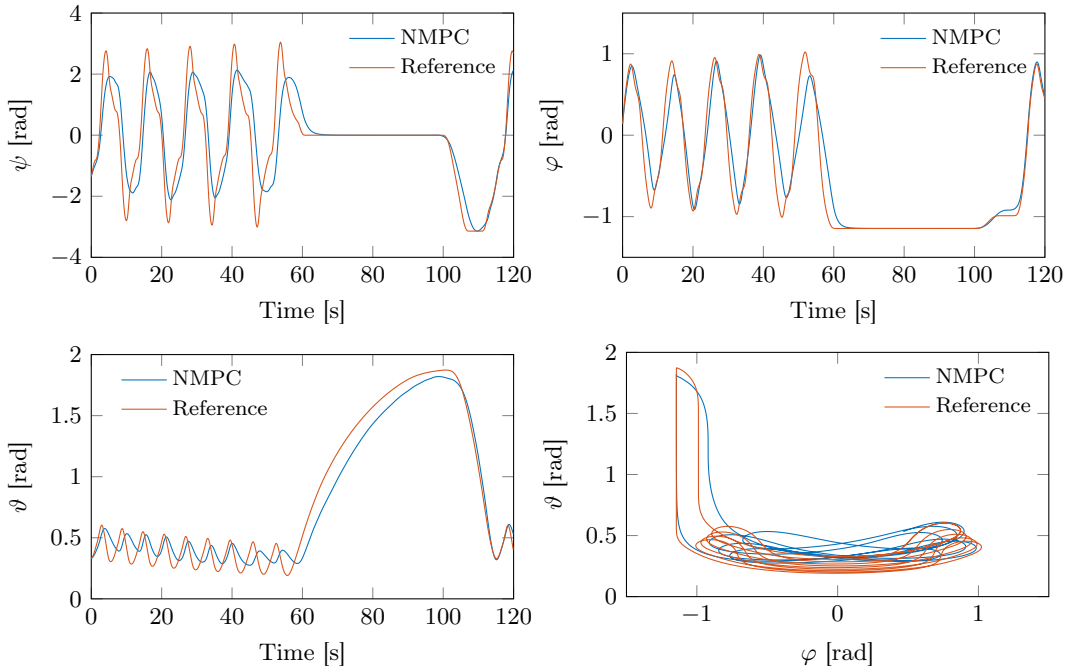


FIGURE 6.6: NMPC tracking with $E = 4$ and $g_k = 0.08$.

TABLE 6.6: NMPC performance considering a 20 % parameter mismatch $E = 4$ and $g_k = 0.08$.

R^2	η_{Loyd}
45.2%	25.58%

Second Maximal Test: $E = 6$, $g_k = 0.12$

As with the nominal mismatch, dealing with underestimated parameters is a critical issue; in particular, for the particular case of a 20 % underestimated mismatch, it can be observed in Figure 6.7 and Table 6.7 how the system dynamic become that fast that the NMPC can no longer control the trajectory and the system become unstable.

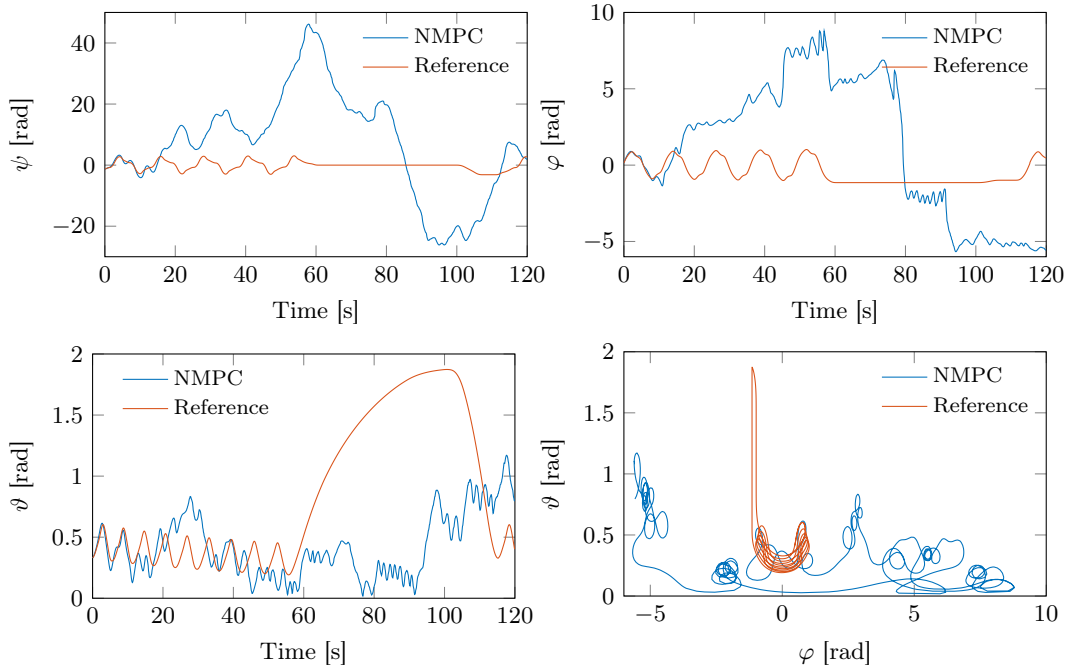


FIGURE 6.7: NMPC with $E = 6$ and $g_k = 0.12$.

TABLE 6.7: NMPC performance considering a 20 % parameter mismatch with $E = 6$ and $g_k = 0.12$.

R^2	η_{Loyd}
-294.5%	18.21%

Maximal Mismatch Analysis

From analyzing the previous test studies a main conclusion can be drawn: whereas underestimating parameters is a very critical and unstable situation that should always be avoided, overestimation creates quite stable flight conditions and provide a security measure to avoid kite crashes. In view of the previous facts and in order to ensure controller stability and

safety conditions, it might be reasonable to slightly overestimate the parameters E and g_k so that, despite decreasing the efficiency, unstable situations are avoided and a kite that fly smaller but safer lemniscates is obtained.

As a result, we propose a methodology to avoid controller failures in the case of having parameter mismatches; specifically, we consider that, since overestimating is an easier task than obtaining an accurate estimation, any decent observer should be always able to provide a parameter overestimation with a 20 % margin; then, it would be just necessary to use the NMPC with this overestimation to make the system stable.

In order to add an extra value to the previous methodology, the following add-on result is provided: even if the observer assumption was incorrect, i.e the system observer was not able to provide a parameter overestimation, by simply using the maximum parameter values $E = 6$ and $g_k = 0.12$ the controller would remain stable; this can be easily proven by testing the described NMPC against a system with the minimum parameter values $E = 4$ and $g_k = 0.08$. As seen in Figure 6.8 and Table 6.8, even for a 50 % overestimation mismatch and despite having a very poor efficiency, the controller remains stable.

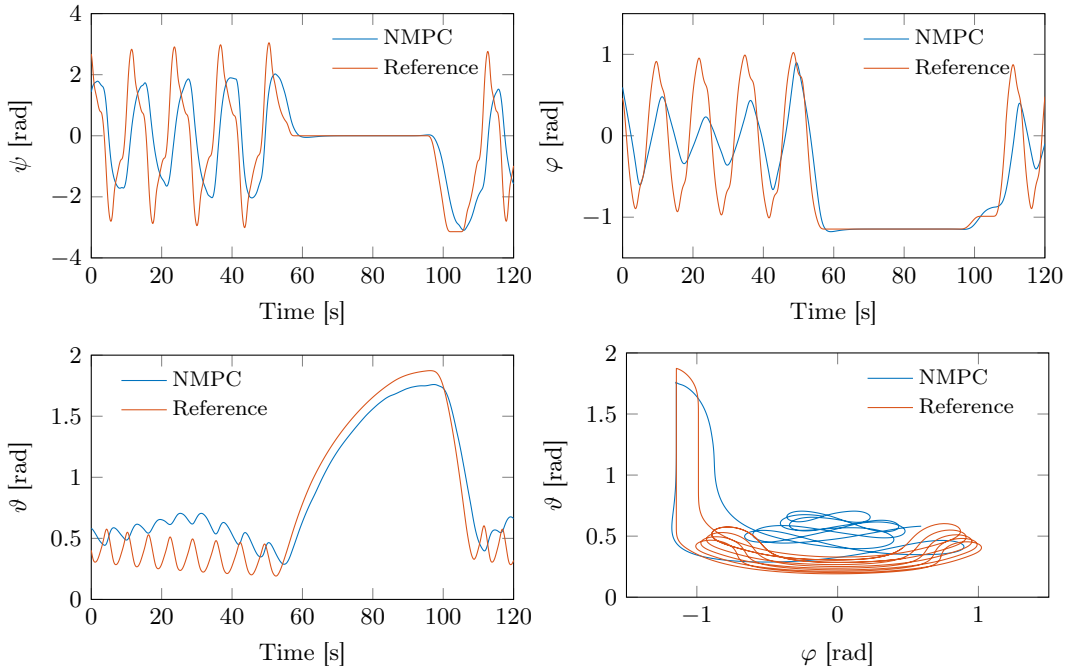


FIGURE 6.8: System with $E = 4$ and $g_k = 0.08$ and NMPC overestimating parameters with $E = 6$ and $g_k = 0.12$.

TABLE 6.8: NMPC performance considering a 50 % overestimated parameter mismatch.

R^2	η_{Loyd}
23.53%	15.39%

6.2.3 Online Parameter Estimation

In the previous two sections, the effects of parameter mismatches between the NMPC and the real system have been analyzed and explained; particularly, it was shown that this type of disturbances make the system dynamics faster or slower depending on whether the NMPC is under or overestimating the parameters; moreover, it was also proposed that overestimation was a safe procedure to ensure controller stability but with the drawback of leading to low power efficiencies.

In this section, we will aim at system efficiency while keeping stability; in particular, an algorithm to do online parameter estimation will be proposed and implemented so that the NMPC can always rely on accurate parameters and target stability while maintaining efficiency.

Moving Horizon Estimation

After becoming familiar with the NMPC concepts described in Chapter 4 and so far in Chapter 5, it is easy to understand the idea behind moving horizon estimation (MHE) [29].

Using the same foundations as MPC, MHE solves an OCP problem per time step in order to obtain an accurate estimation of the system parameters p and/or the current system state $\bar{x}_0$. In particular, by using a set of functions $y_k = m_k(x_k, u_k, p)$ that map the control u_k, system state x_k and parameters p into the system output y_k, MHE computes the optimal states $X = (x_1, \dots, x_N)$, controls $U = (u_0, \dots, u_{N-1})$, and parameters p that, while ensuring dynamic and path constraints, lead to the most accurate map with respect to a given set of past system measurements $Y = (y_0, y_1, y_2, \dots, y_N)$. As a result, the OCP per time step of the MHE can be defined as:

$$
\begin{aligned}
\underset{X,U,p}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \|m_k(x_k, u_k, p) - y_k\|_Q^2 + \|(m_N(x_N) - \bar{y}_N)\|_{Q_N}^2 \\
\text{subject to} \quad & \Phi_k(x_k, u_k, p) - x_{k+1} = 0, \qquad k = 0, \dots, N-1, \\
& h_k(x_k, u_k, p) \leq 0, \qquad k = 0, \dots, N-1, \\
& u_{\min} \leq u_k \leq u_{\max}, \quad k = 0, \dots, N-1, \\
& x_{\min} \leq x_k \leq x_{\max}, \quad k = 0, \dots, N,
\end{aligned}
$$

where, by considering in the above scheme that $\bar{y}_N$ is the last received measurement, $\bar{x}_0$ can be directly obtained as $\bar{x}_0 = x_N^*$ and the system parameters directly as p^*.

Finally, considering a fixed horizon N, MHE could also perform the switching strategy described for NMPC; in particular, it could use the RTI scheme to shift the solution from the previous iteration and use it as initial guess for the new iteration.

Implementation

As done with the NMPC, the MHE was implemented using the ACADO code generation toolbox in order to, first, have generated C-code ready to be used in an embedded platform, and second, to obtain all the benefits of algorithms specifically designed for embedded optimization and the RTI scheme. In particular, the following MHE scheme was implemented:

$$
\begin{aligned}
\underset{\substack{X,U,\\ \bar{E},\bar{g}_\mathrm{k},v_\mathrm{w}}}{\text{minimize}} \quad & \sum_{k=0}^{N-1} \Big\| (x_k,u_k) - y_k \Big\|_Q^2 + \Big\| x_N - y_N \Big\|_{Q_N}^2 + \Big\| [E_N,\ g_{\mathrm{k},N}]^\top - [E_\mathrm{prev}, g_{\mathrm{k,prev}}]^\top \Big\|_P^2 \\
\text{subject to} \quad & \Phi_k(x_k,u_k,E_k,g_{\mathrm{k},k}) - x_{k+1} = 0, && k=0,\dots,N-1, \\
& v_{\mathrm{w,min}} \le v_{\mathrm{w},k} \le v_{\mathrm{w,max}}, && k=0,\dots,N-1, \\
& \Delta v_{\mathrm{w,min}} \le v_{\mathrm{w},k} - v_{\mathrm{w},k-1} \le \Delta v_{\mathrm{w,max}}, && k=1,\dots,N-1, \\
& g_{\mathrm{k,min}} \le g_{\mathrm{k},k} \le g_{\mathrm{k,max}}, && k=0,\dots,N, \\
& E_\mathrm{min} \le E_k \le E_\mathrm{max}, && k=0,\dots,N, \\
& E_k = E_{k+1}, && k=0,\dots,N-1, \\
& g_{\mathrm{k},k} = g_{\mathrm{k},k+1}, && k=0,\dots,N-1,
\end{aligned}
$$

where:

$$
\begin{aligned}
&X = \big(x_0,\cdots,x_N\big), && U = \big(u_0,\cdots,u_{N-1}\big), \\
&\bar{E} = \big[E_0,\cdots,E_N\big]^\top, && \bar{g}_\mathrm{k} = \big[g_{\mathrm{k},0},\cdots,g_{\mathrm{k},N}\big]^\top, \\
&x_k = \big[\psi_k,\varphi_k,\vartheta_k,l_k\big]^\top, && u_k = \big[\delta_k, v_{\mathrm{winch},k}\big]^\top, \\
&y_k = \big[\psi_{k,\mathrm{m}},\ \varphi_{k,\mathrm{m}},\ \vartheta_{k,\mathrm{m}},\ l_{k,\mathrm{m}},\ v_{\mathrm{winch,k,m}},\ \delta_{k,\mathrm{m}}\big]^\top, \\
&y_N = \big[\psi_{N,\mathrm{m}},\ \varphi_{N,\mathrm{m}},\ \vartheta_{N,\mathrm{m}},\ l_{N,\mathrm{m}}\big]^\top, \\
&v_\mathrm{w} = [v_{\mathrm{w},0},\dots,v_{\mathrm{w},N-1}]^\top.
\end{aligned}
$$

and where the following features and conventions have been used:

(i). To avoid destroying the sparse structure of the problem, E and g_k were modeled as system states with no dynamics, i.e. $\dot{E} = \dot{g_\mathrm{k}} = 0$; in particular, the vectors $\bar{E} = \big[E_0,\cdots,E_N\big]^\top$ and $\bar{g}_\mathrm{k} = \big[g_{\mathrm{k},0},\cdots,g_{\mathrm{k},N}\big]^\top$ were used to represent the parameters at each time node.

(ii). The initial constraint $x_0 - \bar{x}_0 = 0$ was removed since the information on the first state x_0 is similar as the information for any other state x_k.

(iii). The wind speed v_w was included as an OCP variable in order to obtain a more accurate parameter estimation. Furthermore, the upper and lower value as well as the time derivative of v_w was limited to obtain a smooth wind speed distribution.

(iv). E_prev and $g_\mathrm{k,prev}$ represent the optimal parameters of the previous iterations. The last term of the objective function including these two values has to be added to avoid big variations of parameters between consecutive iterations.

(v). $\gamma_{k,\mathrm{m}}$ represents the measurement of the state/control γ at the time point k.

(vi). Q, Q_N and P were obtained by tuning the MHE and their values are represented by:

$$\begin{aligned} Q &= \mathrm{diag}(0.4645,\ 1.7972,\ 2.7057,\ 0.0042,\ 141.9234,\ 14.1923),\\ Q_N &= \mathrm{diag}(0.4645,\ 1.7972,\ 2.7057,\ 0.0042),\\ P &= \mathrm{diag}(0.25, 50). \end{aligned}$$

First MHE Test: $E = 4,\ g_{\mathrm{k}} = 0.08$

To test the performance of the MHE, it was first run against the easier case where the parameters are minimal, i.e. without estimation the NMPC would overestimate the parameters. Considering Figure 6.9 illustrating the estimation of the parameters during a nine periods flight, it seems that the MHE can estimate parameters with a decent performance.

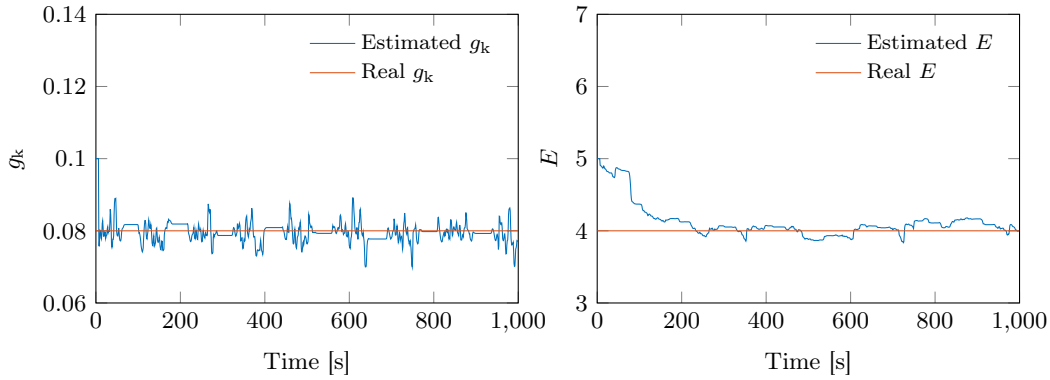


FIGURE 6.9: Parameter estimation using MHE with $E = 4$ and $g_{\mathrm{k}} = 0.08$.

Figure 6.10 depicts the state trajectories for the usual case; by comparing them with Figure 6.6 representing the trajectories without MHE, it can be rapidly appreciated the algorithm improvement.

Furthermore, it can be observed in Table 6.9 that, by using these estimated parameters instead of the fixed $E = 5$ and $g_{\mathrm{k}} = 0.01$, the NMPC improves its tracking performance as well as the system efficiency. In order to have a glance at the system performance, Figure 6.10 depicts the state trajectories for the usual case; by comparing them with Figure 6.6 representing the trajectories without MHE, it can be rapidly appreciated the algorithm improvement.

TABLE 6.9: NMPC performance with & without MHE with $E = 4$ and $g_{\mathrm{k}} = 0.08$.

	R^2 [%]	η_{Loyd} [%]
Without MHE	45.2%	25.58%
With MHE	69.51%	28.71%

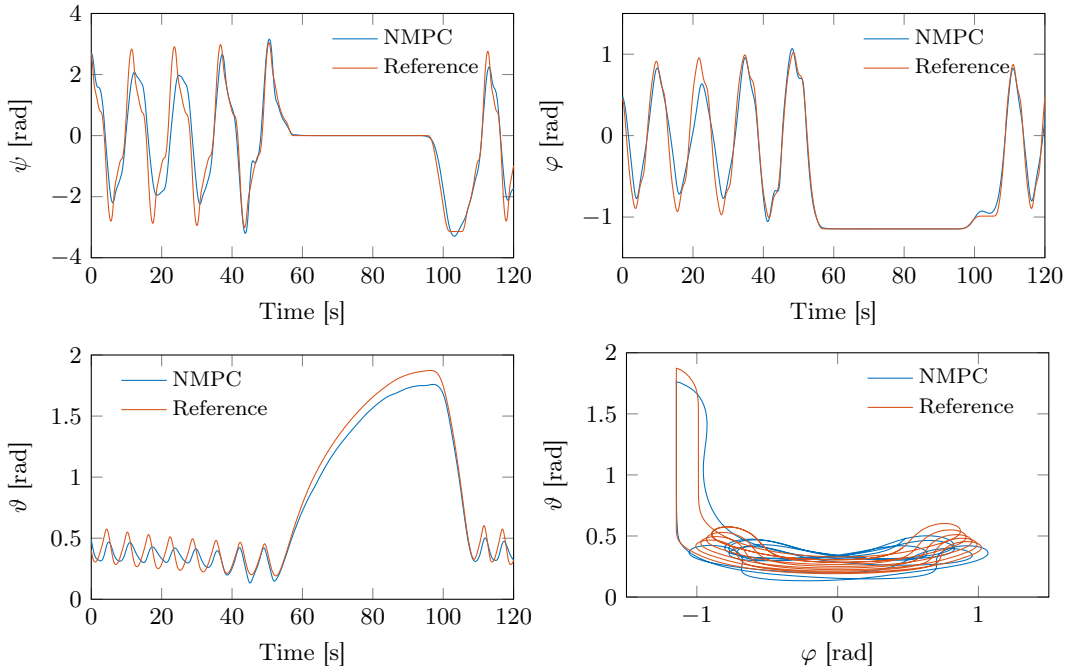


FIGURE 6.10: NMPC using MHE and a system with $E = 4$ and $g_k = 0.08$.

Second MHE Test: $E = 6$, $g_k = 0.12$

After observing the stability and efficiency enhancement obtained by using the MHE against minimal parameters, the performance of the NMPC-MHE should be tested against the more critical situation of having maximal parameters. By considering Figure 6.11 representing the estimated parameters during a nine-periods flight, it can be again concluded that the MHE can also accurately estimate the system parameter when their value is maximum.

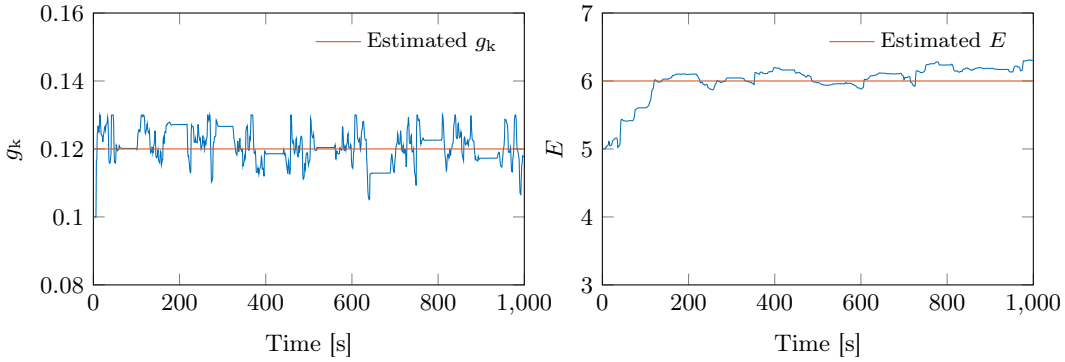


FIGURE 6.11: Parameter estimation using MHE with $E = 6$ and $g_k = 0.12$.

Regarding the NMPC performance, in Figure 6.12 it can be observed how, by using more accurate parameters, the controller becomes stable and the performance of tracking the optimal trajectory is no longer a problem; specifically, by comparing Figures 6.12 and

6.7, the tremendous difference of using versus not using MHE as well as the importance of MHE to bring stability to the system can be seen.

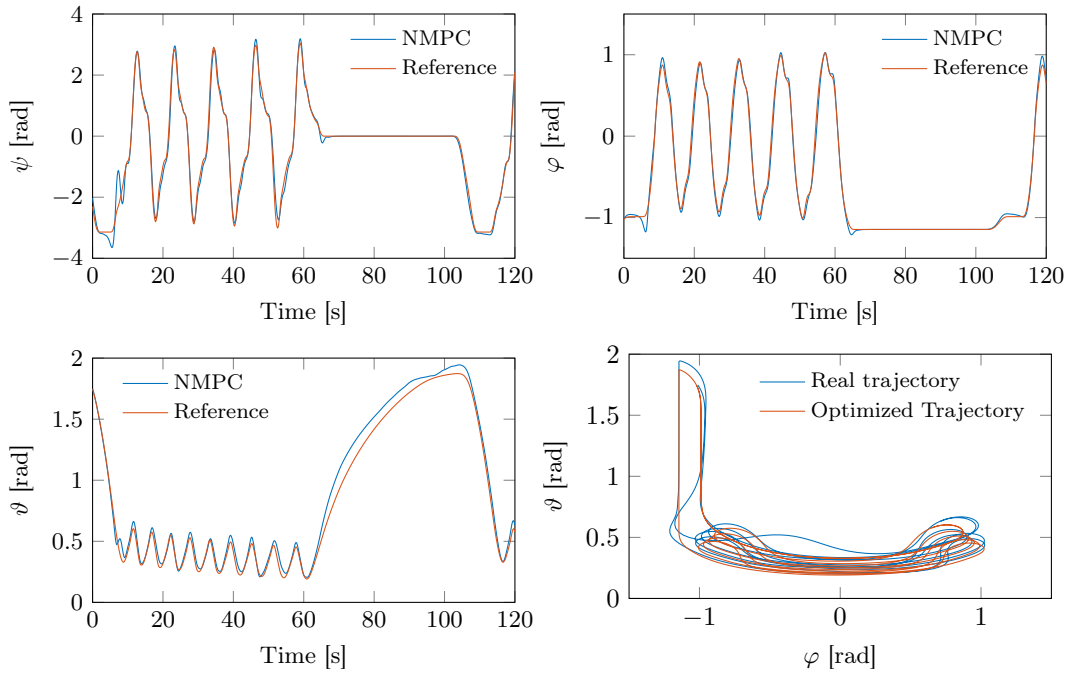


FIGURE 6.12: NMPC tracking performance in a system with $E = 6$ and $g_k = 0.12$.

Finally, in order to observe the controller improvements by means of numerical indicators, Table 6.10 compares the NMPC performance with and without MHE.

TABLE 6.10: NMPC performance with & without MHE considering a system with $E = 6$ and $g_k = 0.12$.

	R^2 [%]	η_{Loyd} [%]
Without MHE	-294.5%	18.21%
With MHE	86.75%	34.22%

Estimated Wind Speed

A remarkable result is that, while implementing the MHE for parameter estimation, excellent results on the wind speed estimation were observed; in particular, Figure 6.13 represents the online wind speed using the MHE; it can be clearly observed that, despite not being a perfect fit, the wind speed estimation seems to be a more accurate representation of the real wind speed than the currently used wind speed filter.

Nevertheless, despite obtaining this excellent outcome, we believe that, since real wind conditions might involve further effects, a further study on the topic must be carried before any real implementation; therefore, the results are illustrated here but the wind estimation

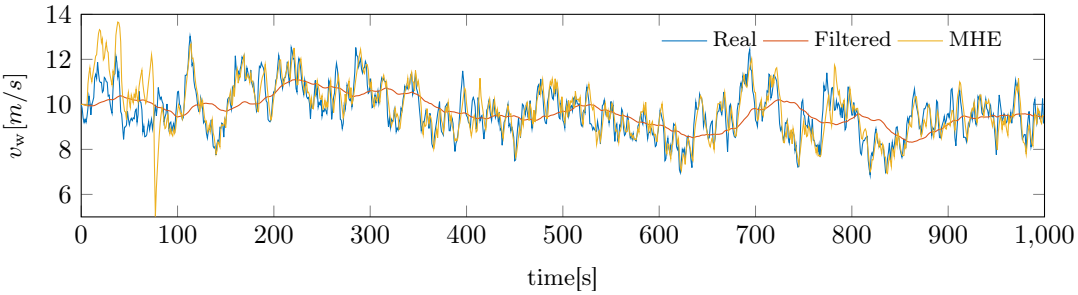


FIGURE 6.13: Wind speed estimation using MHE.

will not be used for the NMPC implementation. In particular, the average wind speed will be kept as the observed value in order to remain in the worst case scenario.

6.2.4 Conclusion and Remark

Parameter mismatches are a critical phenomenon that has to be considered in order to obtain a stable controller. In general, by overestimating the real parameters a stable system with lower efficiency can be obtained; however, if high efficiency is targeted, overestimating is no longer a solution and online parameter estimation is required. Table 6.11 summarizes the results of the previous experiments.

TABLE 6.11: NMPC performance summary for parameter mismatches.

	Without MHE					With MHE	
Real value	Larger		Lower			Larger	Lower.
of E, g_k	10 %	20 %	10 %	20 %	40 %	20 %	
R^2 [%]	60.14	-294.5	67.96	45.2	23.53	69.51	86.75
η_{Loyd} [%]	34.99	18.21	30.95	25.58	15.39	28.71	34.22

It is important to point out that the online parameter estimation algorithm uses system states that were already estimated by the real observer; this algorithm by no means tries to replace the current observer and it should be only used as a tool to improve the controller stability by providing some knowledge regarding the system parameters.

As a final note, it is important to remark the better performance of using MHE with the parameter mismatches $E = 6$ and $g_k = 0.12$ versus not having parameter mismatches at all; particularly, it is remarkable that the former, despite having extra mismatches, is able to perform better. However, this effect is actually expected: for the case of having higher system parameters, the system dynamics are faster and, due to MHE estimation, the NMPC is aware of it; as a result, the NMPC can exploit the faster dynamics and choose more convenient and smoother controls so that the tracking efficiency is improved.

6.3 Wind Direction

One of the assumptions done to obtain the system dynamics was that the wind direction ϕ_w is always constant and aligned with respect to the $\vec{e}_x$ axis. In real life, the average of the

wind direction has indeed very low dynamics and the resulting variable of low pass filtering ϕ_{w} varies only a couple of degrees per hour. Nevertheless, gusts in real-world wind fields may come along with changes in wind direction for several seconds (usually up to several degrees depending on environmental conditions).

As a result, the average of ϕ_{w} can be used as part of the reference system, and indeed, the current system observer dynamically detects the slow motions of ϕ_{w}, realigns the reference system and updates the estimated values according to the new reference; nevertheless, the real wind direction is not always aligned with the $\vec{e}_{\mathrm{x}}$ axis and the controller has to deal with it. For consistency reasons, ϕ_{w} will be kept as the average wind direction and, as illustrated in Figure 6.14, the real direction will be modeled as an offset $\Delta\phi_{\mathrm{w}}$ with respect to the $\vec{e}_{\mathrm{x}}$.

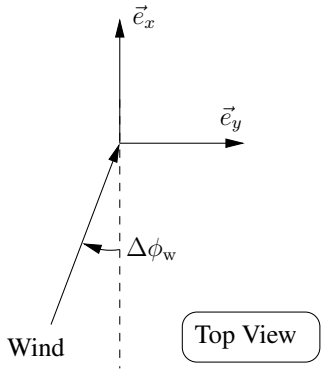


FIGURE 6.14: Reference system considering wind direction.

It is important to consider that, from a controller perspective, the system observer can not provide any estimation of $\Delta\phi_{\mathrm{w}}$; as a result, the controller has no means to obtain $\Delta\phi_{\mathrm{w}}$ and the best thing that can be done is to make the NMPC robust and stable against the fast motions of ϕ_{w}. In this section, NMPC stability against wind directions will be proven first, by modeling extended dynamics that include $\Delta\phi_{\mathrm{w}}$, second, by generating some $\Delta\phi_{\mathrm{w}}$ profile, and third, by using these dynamics and profile in the simulator and running the NMPC against it.

6.3.1 Extended Dynamics

In order to extend the system dynamics, the derivation of the equations of motion of [15] can be repeated but now considering the real wind speed vector. In particular, assuming that, as illustrated in Figure 6.14, $\Delta\phi_{\mathrm{w}}$ is defined in the $\vec{e}_{\mathrm{x}}$-$\vec{e}_{\mathrm{y}}$ plane, the wind vector is $[v_{\mathrm{w}}\cos\Delta\phi_{\mathrm{w}}, v_{\mathrm{w}}\sin\Delta\phi_{\mathrm{w}}, 0]^{\top}$ instead of $[v_{\mathrm{w}}, 0, 0]^{\top}$; then, by repeating the derivation, the equations of motion for φ, ϑ can be rewritten as:

$$\dot{\varphi} = -\frac{v_{\mathrm{a}}}{l\sin\vartheta}\sin\psi + \frac{v_{\mathrm{w}}}{l\sin\vartheta}\cos\varphi\sin\Delta\phi_{\mathrm{w}}, \tag{6.1a}$$

$$\dot{\vartheta} = \frac{v_{\mathrm{a}}}{l}\cos\psi - \frac{v_{\mathrm{w}}}{l}\sin\vartheta\cos\Delta\phi_{\mathrm{w}} + \frac{v_{\mathrm{w}}}{l}\sin\varphi\cos\vartheta\sin\Delta\phi_{\mathrm{w}}, \tag{6.1b}$$

where the air path velocity is also updated as:

$$v_{\mathrm{a}} = v_{\mathrm{w}}E\cos\vartheta\cos\Delta\phi_{\mathrm{w}} - \dot{l}E + v_{\mathrm{w}}E\sin\varphi\sin\vartheta\sin\Delta\phi_{\mathrm{w}}. \tag{6.1c}$$

6.3.2 Wind Direction Profile

In order to test the NMPC with the most accurate wind direction profile, real wind direction measurements, which were obtained by Skysails during a flight test, will be used. In particular, Figure 6.15 represents the distribution of real $\Delta\phi_w$ data during a time window of two hours. From the figure, it can be clearly observed that the $\Delta\phi_w$ distribution can be roughly approximated with a Gaussian distribution $\mathcal{N}(0, \sigma_\phi)$, where, from the real data, it was inferred that $\sigma_\phi \approx 7.25°$.

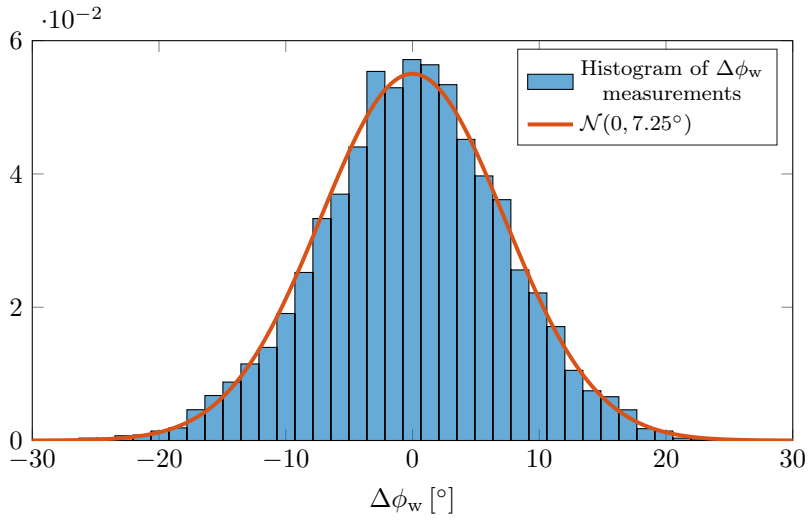


FIGURE 6.15: Wind direction distribution in a time window of two hours.

To have a better picture of how this wind direction profile looks like, Figure 6.16 represents the same $\Delta\phi_w$ measurements in a time window of 10 minutes.

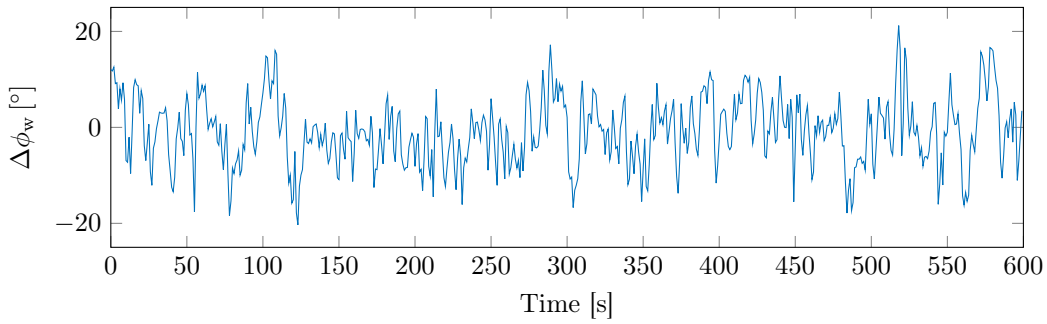


FIGURE 6.16: Wind direction$\Delta\phi_w$ measurements as a function of time.

6.3.3 Implementation

By extending the simulator dynamics with Equations (6.1a)-(6.1c) and using the real measurements for the wind direction profile, the stability and robustness of the NMPC against $\Delta\phi_w$ can be easily tested; moreover, to be consistent and always test the controller against the

worse case scenario, the real wind speed profile, the system delay, and the worst parameter mismatch with the MHE estimation will be all added to the system simulator.

Figure 6.17 and Table 6.12 illustrate the system performance under the above described conditions; by comparing them with Figure 6.12 and Table 6.10 representing the same testing conditions but without the $\Delta\phi_w$ disturbances, it can be immediately observed that wind angle variations have an insignificant effect on the NMPC performance.

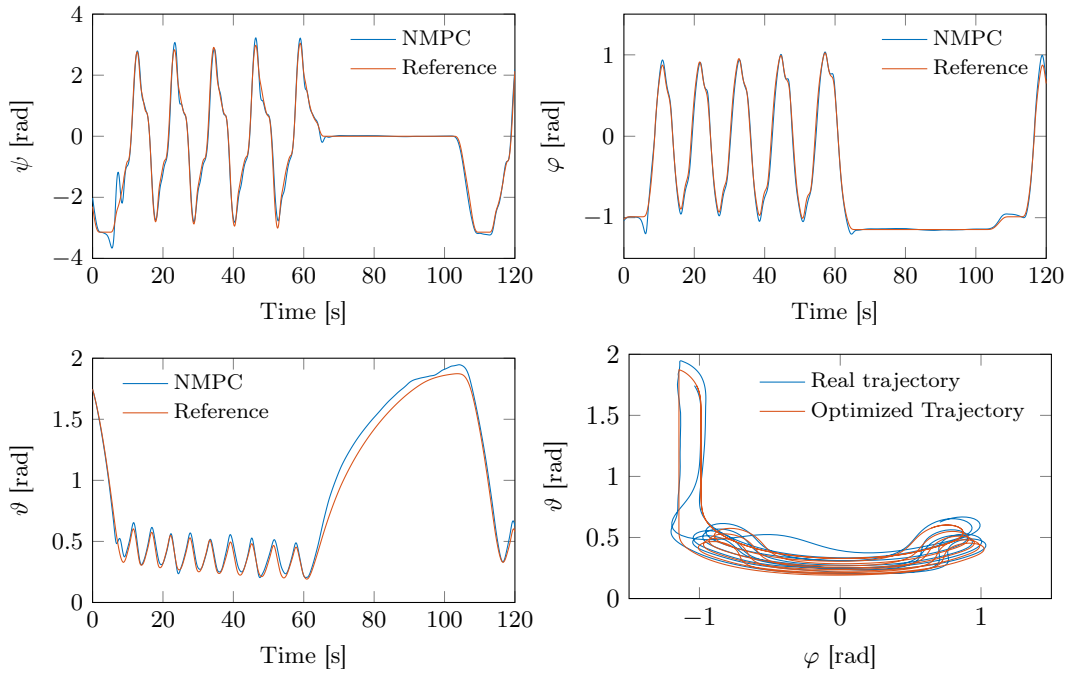


FIGURE 6.17: NMPC tracking performance with a real wind gusts (speed and direction), parameter mismatches and a control delay.

TABLE 6.12: NMPC efficiency with for real wind gusts (direction and speed), maximum parameter mismatches and a control delay.

R^2	η_{Loyd}
86.61%	34.22%

6.4 Control Bias

Recalling from Section 1.2, the kite uses a control pod at the end of the towing line to apply deflections and obtain a curved flight; this device was named the steering actuator and its behavior within the dynamics was modeled by a control δ.

Nevertheless, in the real system setup, modeling the kite aerodynamic effect of the control pod by a control δ is an idealization; in particular, it has been observed that, due to an asymmetry in the setup of the control pod, there is a mismatch between the model and the reality and the control δ suffers from an unknown bias error $\Delta\delta$. In theory, this mismatch

could be estimated and then included it in the controller by simply extending the $\dot{\psi}$ equation of motion as:

$$\dot{\psi} = g_{\mathrm{k}} v_{\mathrm{a}} (\delta + \Delta\delta) + \dot{\varphi} \cos\vartheta.$$

In reality, it has been observed that online estimation of the offset $\Delta\delta$ is very hard; in particular, the effect is complicated to quantify and, as with the wind direction, the best a controller can do is tune itself to remain stable. As a consequence, in this section the NMPC stability and performance will be tested by including the bias error in the system simulator; moreover, like in the previous sections, all the previously defined disturbances will be included in order to keep consistency.

In order to model the bias error it has to be considered that, despite $\Delta\delta$ being hard to estimate online, it is known that its maximum value will never be larger than 10 % the value of δ and that, for the considered NMPC time horizon, it is roughly constant.

Positive Constant Bias

The results of continuously perturbing δ with a positive 10 % disturbance are shown in Figure 6.18 and Table 6.13; by comparing them with Figure 6.17 and Table 6.12 we can immediately conclude that, as wind direction, a maximum positive bias on δ does barely influence the tracking performance and system efficiency.

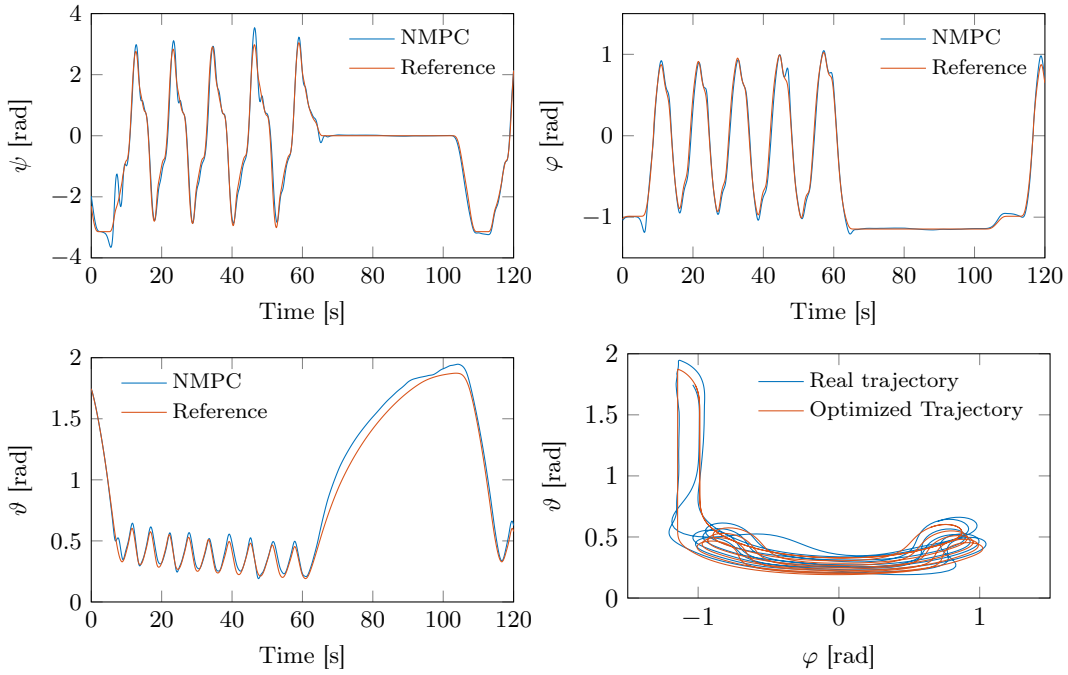


FIGURE 6.18: NMPC tracking performance under the influence of a bias $\Delta\delta = 0.1\delta$.

Table 6.13: NMPC efficiency with a bias error $\Delta\delta = 0.1\delta$.

R^2	η_{Loyd}
85.76%	34.25%

Negative Constant Bias

The results of continuously perturbing δ with a negative 10 % disturbance are almost equal to using the positive counterpart; as a result, for the sake of simplicity and to avoid repetition, the standard figure with the four states trajectories will not be depicted; instead, only the performance metrics as given in Table 6.14 will be shown. Looking at them it can be seen that, as with the positive bias, the system efficiency and performance remain almost unaffected.

Table 6.14: NMPC efficiency with a bias error $\Delta\delta = -0.1\delta$.

R^2	η_{Loyd}
86.5%	34.23%

6.5 Real Observer

In all the previous sections, it has always been considered that the observer could provide the NMPC an accurate representation of the current state $\bar{x}_0 = [\bar{\psi}_0, \bar{\varphi}_0, \bar{\vartheta}_0, \bar{l}_0]^\top$; in particular, all sort of different real disturbances and mismatches between NMPC and simulator were modeled, but the system observer was assumed to provide the perfect estimation of the current state $\bar{x}_0$ after simulation.

However, in real life there is no such a thing as a perfect observer, i.e. any type of estimation will always incur in some errors and the current state $\bar{x}_0$ will never be an exact representation of the reality. As a result, in this section realistic estimation errors will be modeled and the NMPC performance regarding them will be measured.

6.5.1 Estimation Error Models

In order to perform an assessment of the errors that the real observer undergoes, we had a talk with the Skysails team and discussed the different influences of real flight conditions on each of the four estimated states; in the following sections, we will first discuss and explain the outcome of this talk, i.e. how to model the estimation errors, and then, we will model a realistic observer to test the NMPC. It is important to keep in mind that, in order to avoid an overwhelming amount of figures, only the R^2 and η_{Loyd} indicators will be used for most part of the study and, only at the end of the section, a figure will be used to graphically illustrate the NMPC tracking performance considering all sources of error.

Tether Length l

When looking at a real tether there are three main physical effects affecting the estimation error:

(i). The first effect is due to the inaccuracy of the sensor itself; in particular, length measurements are obtained by counting the revolutions of the ground generator/motor that

reels in/out the tether and then inferring the tether length based on the radius of the generator; since the radius can not be accurately measured, an unknown permanent offset Δl builds up.

(ii). A second effect is a tether deformation due to pulling forces; in particular, it has been observed that, after several hours of continuous flight and high pulling forces, the tether suffers from creep deformation and the tether length changes. The result of these phenomenon is second unknown and permanent offset Δl. According to the Skysails team, whenever the two offsets act together, they can achieve a maximum value of $\pm 0.5\,\mathrm{m}$.

(iii). A third effect is related to a catenary shape on the tether: one of the assumptions done by the estimation model is that the tether is always completely straight; however, as depicted in Figure 6.19, under the influence of gravity in combination with low wind speed the tether deforms creating some sort of catenary so that the model assumption does not hold. The result is an extra estimation error on the tether length that, according to the Skysails team, can be modeled as a zero mean Gaussian noise with a standard deviation $\sigma_l = 0.5\,\mathrm{m}$.

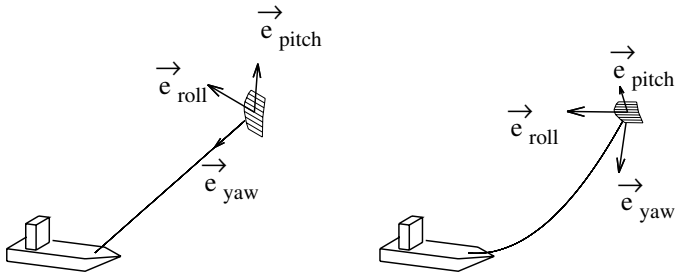


FIGURE 6.19: Left: estimation model assumption. Right: catenary effect under low wind forces.

Therefore, in order to model a realistic estimator l of the tether length, the described error sources should be added to the simulation output $\bar{l}$:

$$l = \bar{l} + \Delta l + \omega_l, \quad \text{where } |\Delta l| \leq 0.5\,\mathrm{m} \text{ and } \omega_l \sim \mathcal{N}(0, 0.5\,\mathrm{m}).$$

Angle ψ

In order to estimate the angle ψ, the Skysails observer has to heavily rely on gyroscope measurements; as a result, it is forced to tackle the standard issue of gyroscopes: since they provide turn rate measurements with a bias error, when these measurements are integrated, a drift error on the estimated angles builds up.

As it is expected, the kite observer estimates and compensates the error; nevertheless, due to the fast dynamics of the drift, the observer estimation has a small lag with respect to the real error and the drift estimation is not accurate. According to the Skysails team, the latter means that the estimated angle differs from the real angle by some offset $\Delta\psi$; in particular, they consider that this offset error can be modeled as a random walk with very low dynamics (1 % variation per hour), and in our case, since simulations last only some minutes, they believed that a realistic observed ψ should be modeled by adding a constant

offset to the simulated state $\bar{\psi}$:

$$\psi = \bar{\psi} + \Delta\psi, \quad \text{with } |\Delta\psi| \leq 0.2\,\text{rad}.$$

Angle ϑ

Since the estimation of ϑ, in contrast with ψ, is done based on angular sensors located at the ground station, the estimation of ϑ does not need to perform the standard drift error estimation. Nevertheless, the Skysails team has observed that, in some cases, an offset error $\Delta\vartheta$ is also built up.

The general agreement regarding this error is that it is created by the tether catenary effect depicted in Figure 6.19; in particular, due to the shape of the catenary, the estimator observes an apparent angle ϑ that is lower than the real elevation angle, i.e. due to the curvature of the tether the offset $\Delta\vartheta$ is expected to be negative; this particular effect can be observed in Figure 6.20.

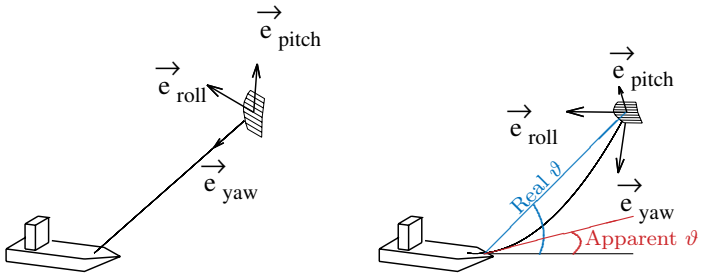


FIGURE 6.20: Left: ideal situation for ϑ estimation. Right: wrong ϑ estimation due to the catenary effect.

As with the angle ψ, the Skysails team estimates that, first, this offset error can achieve a minimum value of -0.1 rad, and second, that due the horizon lengths used for the NMPC, $\Delta\vartheta$ can be assumed to be constant. As a result, the model for a real ϑ estimation can be given by:

$$\vartheta = \bar{\vartheta} + \Delta\vartheta, \quad \text{with } -0.1\,\text{rad} \leq \Delta\vartheta \leq 0\,\text{rad}.$$

Angle φ

Similar to ϑ, the angle φ uses an angular sensor at the ground estimation to perform the estimation; as a result, it has also been reported that an offset error $\Delta\varphi$ appears in the estimation procedure. Nevertheless, the key difference with respect to ϑ is that, due to the flight symmetry in φ, the offset error might be positive or negative. In particular, the Skysails team estimates that the maximum offset is bounded to 0.1 rad and that, as before, $\Delta\varphi$ can be modeled as a constant offset. As a consequence, the real estimation of φ can be modeled by:

$$\varphi = \bar{\varphi} + \Delta\varphi, \quad \text{with } |\Delta\varphi| \leq 0.1\,\text{rad}.$$

6.5.2 Simulation Results

In order to fully test the NMPC performance against estimation errors, two different approaches will be implemented. First, the individual estimation errors will be tested so that their specific effect on the controller performance can be characterized; then, in order to test whether the NMPC is robust in every possible situation, the controller will be tested against each worst case scenario.

Individual Errors

Considering the described models above, it is clear that a total of seven test scenarios shall be performed in order to study the particular effect of positive and negative offsets in l, ψ, and φ and a negative offset in ϑ. For the sake of consistency, these seven scenarios will consider the same Gaussian noise profile in the l estimation. The outcome of these particular tests is depicted in Table 6.15.

Table 6.15: NMPC performance considering individual estimation errors.

$\Delta\vartheta$ [rad]	$\Delta\varphi$ [rad]	$\Delta\psi$ [rad]	Δl [m]	R^2 [%]	η_{Loyd} [%]
0	0	0	-0.5	86.08	34.22
			0.5	85.94	34.21
		-0.2	0	82.41	33.89
		0.2		82.93	34.08
	-0.1	0		85.97	34.22
	0.1			85.84	34.21
-0.1	0			84.22	33.03

By looking at the data and comparing it with Table 6.14 (illustrating the results when estimation errors are not present) several conclusions can be drawn:

(i). The NMPC seems to be quite robust and efficient against every estimation error when they act individually.

(ii). In particular, the offsets Δl and $\Delta\varphi$ seem to have barely no effect.

(iii). The negative offset $\Delta\vartheta$ appears to perform slightly worse than Δl and $\Delta\varphi$; however, the effect is too small to be even noticed.

(iv). Finally, the offset $\Delta\psi$ seems to influence the controller the most; nevertheless, it can also be observed that the loss in performance it is quite small compared with the case of having no estimation errors at all.

Multiple Errors

Considering that l, ψ and φ have two different offset scenarios whereas ϑ a single one, a total of $2 \times 2 \times 2 \times 1 = 8$ test simulations should be performed in order to evaluate every worst possible case. The results of these specific experiments are depicted in Table 6.16.

From analyzing the data, the following conclusions can be drawn:

TABLE 6.16: NMPC performance considering the eight possible worst case scenarios of real estimation errors.

$\Delta\vartheta$ [rad]	$\Delta\varphi$ [rad]	$\Delta\psi$ [rad]	Δl [m]	R^2 [%]	η_{Loyd} [%]
-0.1	-0.1	-0.2	-0.5	81.07	31.87
			0.5	81.10	31.93
		0.2	-0.5	80.76	32.88
			0.5	80.68	32.83
	0.1	-0.2	-0.5	81.33	31.87
			0.5	81.18	31.93
		0.2	-0.5	80.02	32.97
			0.5	80.04	32.77

(i). The NMPC seems once again to be robust and efficient against every possible estimation disturbance.

(ii). All scenarios show very similar behavior: the efficiency η_{Loyd} seems to experience a drop between 1.5-2.5 % and the goodness of fitting R^2 a decrease of 5 %.

(iii). The offset in the angle ψ seems to be once again the most determinant factor; in particular, positive offsets appear to have better efficiency but worse tracking performance.

(iv). An interesting characteristic to point out is the symmetric behavior of positive and negative $\Delta\varphi$ offsets: since the trajectory of the angle φ is quite symmetrical, it can be observed that tests that only differ in the sign of $\Delta\varphi$ seem to have very alike indicators.

(v). Despite being all the results quite similar, it can be detected that the worst scenario occurs when all the offsets are negative; in particular, this test displays the worst η_{Loyd} factor while having a R^2 goodness of fit that is just 0.07 % better than the worst result.

In order to provide a graphical illustration of the NMPC performance under estimation errors, Figures 6.21, 6.22 and 6.23 represent the system trajectories when the NMPC tackles the previously described worst case scenario. In particular, Figure 6.21 represents the trajectory of the four system states, 6.22 the 3D trajectory representation, and finally, 6.23 the system controls v_{winch} and δ.

From analyzing the NMPC graphical results in Figure 6.21 and 6.23, further statements and remarks can be made:

(i). The good NMPC performance observed by the metrics η_{Loyd} and R^2 are corroborated by the graphical results.

(ii). The estimation offsets do not seem to affect the controller stability much.

(iii). The depicted controls are quite smooth, i.e. the controller does not need to perform big control changes to keep the kite stable, at the same time that they are kept within the safety boundaries.

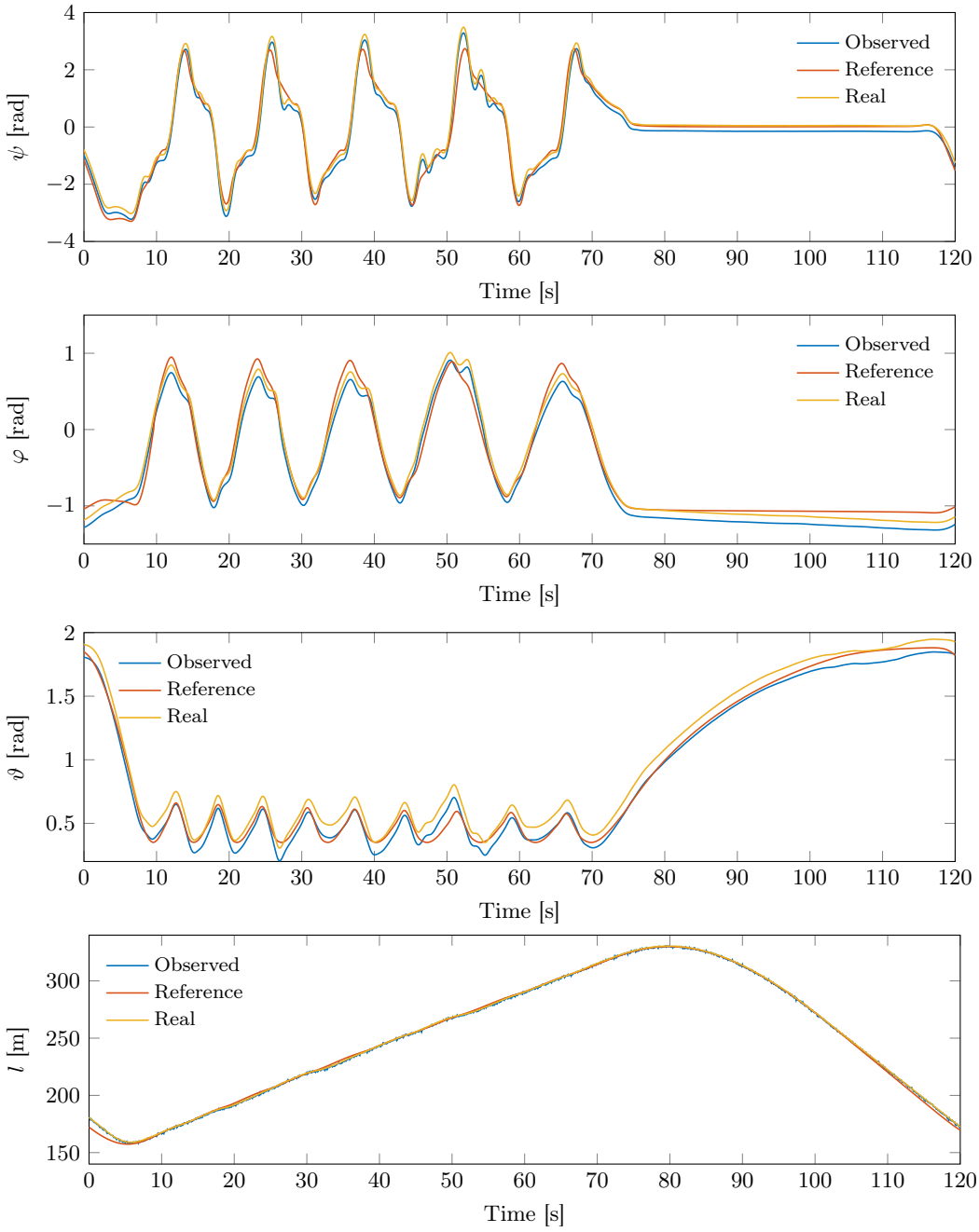


FIGURE 6.21: NMPC performance on the four system states considering the worst possible scenarios for estimation errors, system disturbances and model mismatches.

(iv). The good NMPC performance can be especially appreciated when looking at the 2D projection: despite the real trajectory differing with respect to the reference one, the former tracks the latter with a high degree of accuracy.

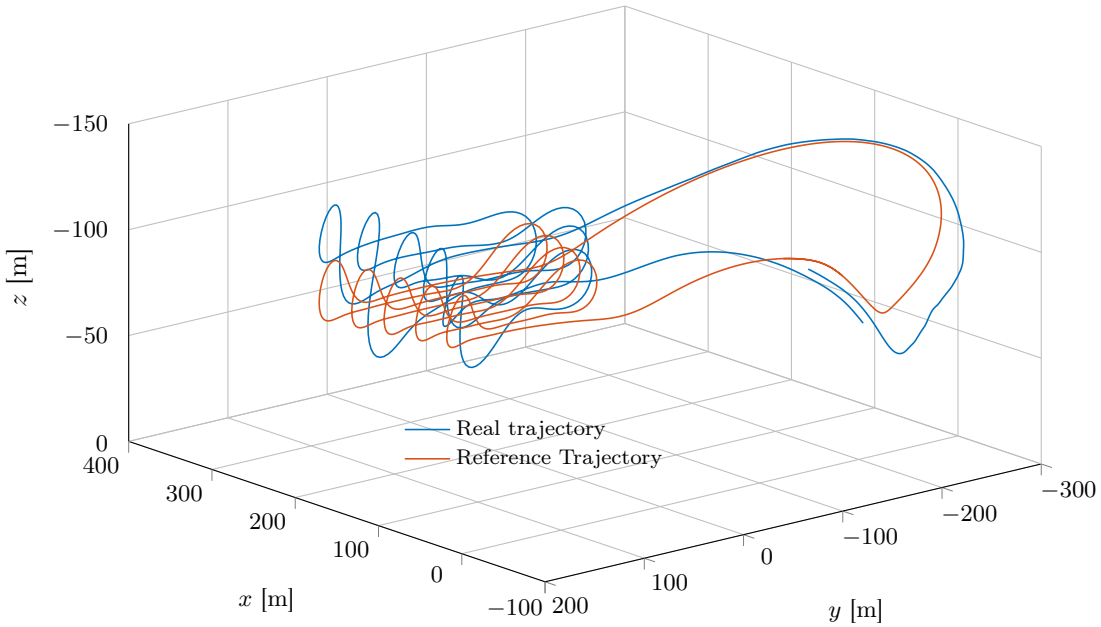
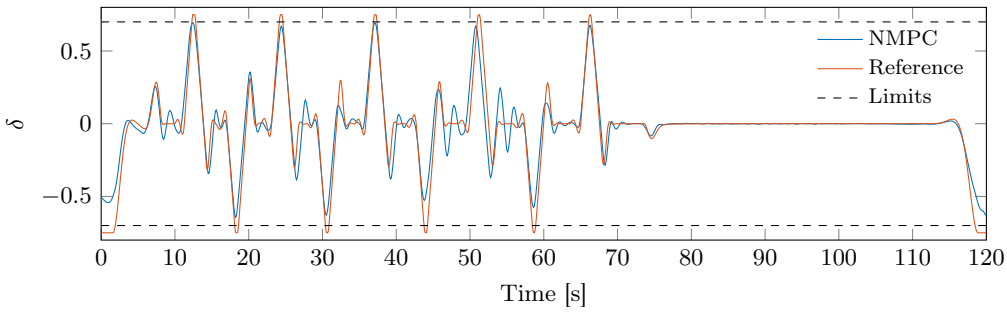
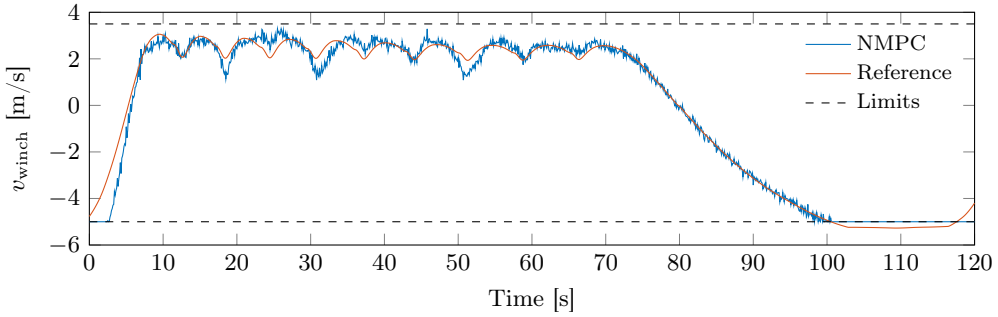


FIGURE 6.22: 3D trajectory when the NMPC performs against the worst possible scenarios for estimation errors, system disturbances and model mismatches.



(A) Steering command δ.



(B) Winch speed v_{winch}.

FIGURE 6.23: 2D trajectory projection and system controls when the NMPC addresses the worst possible scenarios in estimation errors, system disturbances and model parameter mismatches.

Result's implications

It is very important to reflect how outstanding the previous results are: the controller has been tested considering the three main sources of errors: real life disturbances, mismatches between the controller and the real model, and finally, a system observer with realistic observation errors; in all these test scenarios the controller was quite robust, keeping the system stable while accurately tracking the offline precomputed optimal trajectory and achieving a high power efficiency η_{Loyd}.

6.5.3 Analysis of a Positive ϑ Offset

According to the Skysails team, a positive offset in the angle ϑ is not possible. Nevertheless, for the sake of security, we decided to study its implications.

The first performed test was to include the positive offset as the only error in the estimation. Surprisingly, compared with the other seven individual errors, a positive $\Delta\vartheta$ increases the η_{Loyd} factor more than any other individual estimator error and decreases the tracking performance as any other does. To illustrate this effect, Table 6.17 depicts the performance of a positive $\Delta\vartheta$ offset equal to 0.1 rad compared with the best ($\Delta l = -0.5\,\text{m}$) and worst ($\Delta\psi = -0.2\,\text{rad}$) results from the previous individual tests.

TABLE 6.17: Comparison of positive $\Delta\varphi$ offset with respect to the best and worst individual estimator errors.

$\Delta\vartheta$ [rad]	$\Delta\varphi$ [rad]	$\Delta\psi$ [rad]	Δl [m]	R^2 [%]	η_{Loyd} [%]
0	0	0	-0.5	86.08	34.22
0	0	-0.2	0	82.41	33.89
0.1	0	0	0	78.48	35.84

In order to further investigate the implications of the previous result, the NMPC was tested considering the positive $\Delta\vartheta$ offset in couple with all the other plausible estimation errors. The results, which are depicted in Table 6.18, are really interesting. When comparing similar test scenarios which only differ by the sign of the offset $\Delta\vartheta$, it can be observed how positive offsets, despite increasing the system efficiency, lead to a worse tracking performance. Furthermore, it can also be seen that, if the positive $\Delta\vartheta$ acts in couple with $\Delta\varphi = -0.1\,\text{rad}$, $\Delta\psi = 0.2\,\text{rad}$, and $\Delta\varphi = 0.5\,\text{m}$ the kite even becomes unstable and crashes.

Despite positive offsets in ϑ are not a thing to worry about in the current estimator, they might occur in future implementations of the system observer. Therefore, it is important to explain and understand why they are so critical in order to avoid future failure of the controller.

Whenever a positive offset in ϑ occurs, the NMPC regards an observed state ϑ_{obs} that is higher than the real ϑ; as a result, while the NMPC thinks that is bringing the observed trajectory to meet the reference, in reality it is moving the kite to a lower angular position. The outcome of this effect is that the kite tends to fly at lower altitudes (lower angles), which leads in turn to a higher apparent wind speed v_{a}, which is then responsible for the larger harvested energy and the higher power efficiency η_{Loyd}.

Nevertheless, despite the increase in efficiency, a major risk appears as the controller becomes unstable; in particular, by analyzing in detail the system dynamics, it can be clearly

TABLE 6.18: Comparison of performance between positive and negative $\Delta\varphi$.

$\Delta\vartheta$ [rad]	$\Delta\varphi$ [rad]	$\Delta\psi$ [rad]	Δl [m]	R^2 [%]	η_{Loyd} [%]
-0.1	-0.1	-0.2	-0.5	81.07	31.87
			0.5	81.10	31.93
		0.2	-0.5	80.76	32.88
			0.5	80.68	32.83
	0.1	-0.2	-0.5	81.33	31.87
			0.5	81.18	31.93
		0.2	-0.5	80.02	32.97
			0.5	80.04	32.77
0.1	-0.1	-0.2	-0.5	75.67	35.85
			0.5	75.70	35.88
		0.2	-0.5	75.87	35.92
			0.5	-86.66	7.66
	0.1	-0.2	-0.5	76.02	35.85
			0.5	76.37	35.94
		0.2	-0.5	73.58	35.40
			0.5	74.37	35.74

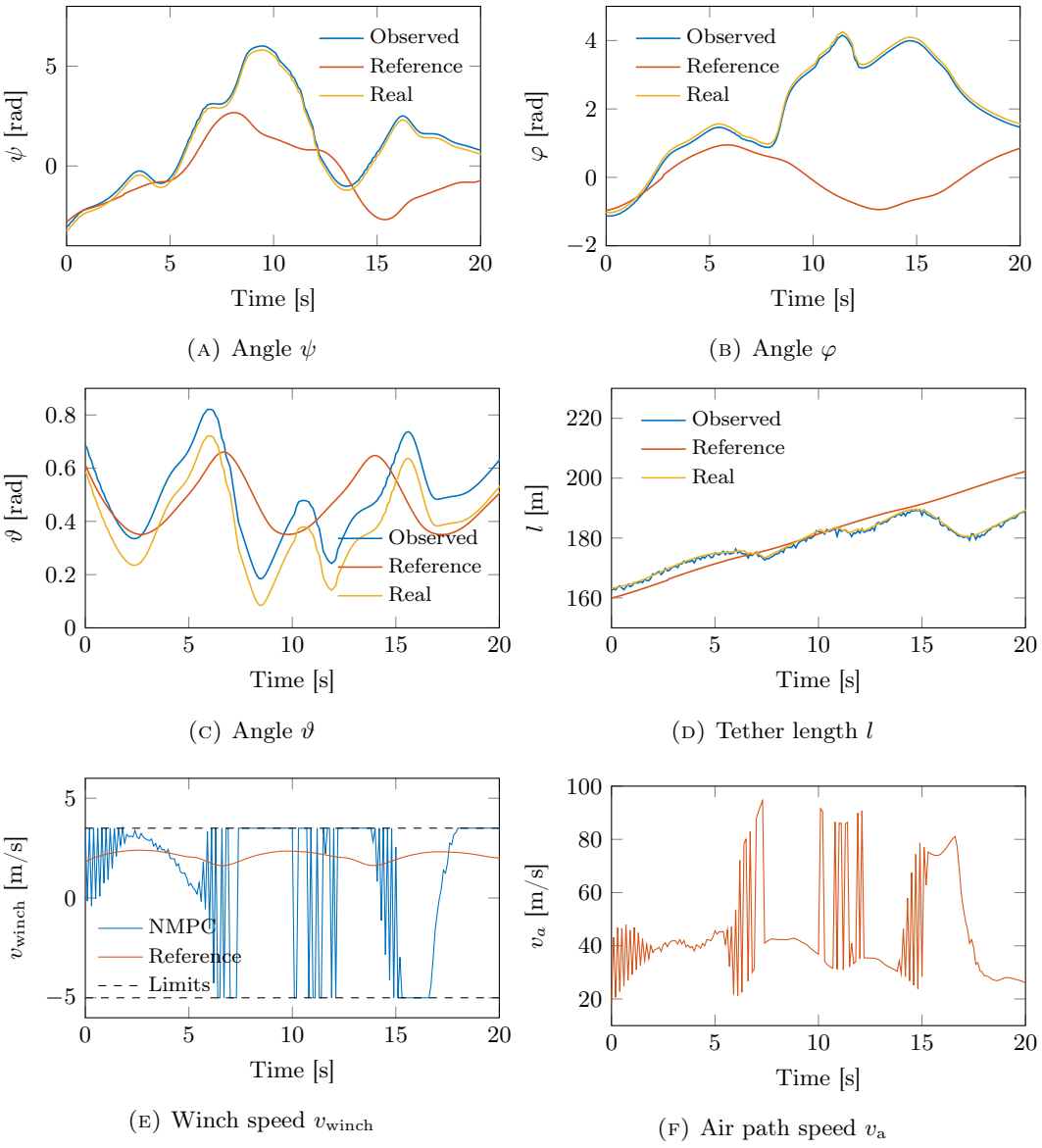
observed how flying at low altitudes can bring the system to unstable situations. Regarding once again the kite equations of motion:

$$\begin{aligned}
\dot{\psi} &= g_{\text{k}} v_{\text{a}} \delta + \dot{\varphi} \cos\vartheta, \\
\dot{\varphi} &= -\frac{v_{\text{a}}}{l \sin\vartheta} \sin\psi, \\
\dot{\vartheta} &= -\frac{v_{\text{w}}}{l} \sin\vartheta + \frac{v_{\text{a}}}{l} \cos\psi, \\
\dot{l} &= v_{\text{winch}}, \\
\text{with:} \quad v_{\text{a}} &= v_{\text{w}} E \cos\vartheta - \dot{l} E,
\end{aligned}$$

it can be perceived that, if the real angle ϑ is lower than presumed, the $\cos\vartheta$ is higher than what the NMPC infers and v_{a} becomes higher than expected. As a result, the derivate $\dot{\psi}$ becomes larger than the NMPC consideration, which in turn leads to a kite that starts to steer without much control and which eventually might crash. For illustration purposes, Figure 6.24 illustrates a kite crash due to positive offsets in ϑ.

Considering the analysis done regarding the dynamics and the positive $\Delta\vartheta$, the exact process leading to the kite crash in Figure 6.24 can be explained as follows:

- After approximately 5 s, the NMPC considers that ϑ is higher than in reality and as a result pushes down the angle lower than necessary.



(A) Angle ψ

(B) Angle φ

(C) Angle ϑ

(D) Tether length l

(E) Winch speed v_{winch}

(F) Air path speed v_{a}

FIGURE 6.24: NMPC crashing due to positive offset in the estimation of ϑ.

- As immediate result, the angles ψ, the angle φ and the air path speed v_{a} start to increase.
- To correct for this disturbance, the NMPC tries to lower the length l so that the derivative $\dot{\varphi}$ becomes negative and the $\dot{\vartheta}$ positive.
- Due to the correction in the length, the winch speed is set to its lowest value and v_{a} increases even more.

- This last effect creates a loop that initiates a fast and unstoppable increase in ψ and φ: the kite starts to steer without control and eventually φ reaches the 90° representing the ground floor and the kite crashes.

It is really important to remark once again that this issue is not something to be worried about or concern with the current controller. The Skysails team has assured us that, in our implementation using the current observer, this event is extremely unlikely. Nevertheless, we decided to study and document it so that it serves as a reference in the case of any future modification in the controller or the observer. As a simple suggestion, if the positive offset is ever present, a very plain solution would be to add an artificial offset of -0.1 rad to the ϑ estimation so that the estimated values are always above the real ϑ; the main drawback of such an approach would be that a negative offset of -0.2 rad might build up; nevertheless, considering the little effect on the NMPC performance of negative $\Delta\vartheta$ offset, we believe that this disadvantage would have small implications.

6.6 Conclusion and Remarks

Through the last two chapters, a real NMPC controller has been modeled in order to track optimal trajectories that maximizes the extracted energy; in particular, a robust and stable controller was obtained which is able to follow optimal trajectories under real life conditions. In addition, the controller was designed using a software that generates C-code so that, despite designing the controller in MATLAB, the algorithm can be directly implemented in the real embedded platform.

Furthermore, in order to ensure that the NMPC was being tested in a realistic framework, we hold several meetings with the Skysails team where the full range of possible influences was determined, and then, a system simulator regarding them all was implemented. By tuning the controller and adding different features, a robust NMPC against the following real issues was achieved:

(i). Realistic wind gusts: the NMPC was designed to be stable against fast changes in wind speed as well as wind direction.

(ii). Delay perturbations and offset bias in the control δ.

(iii). Realistic mismatches between the controller model and the real system.

(iv). Pragmatical estimation errors that appear in the real observer.

Nonetheless, in spite of apparently having a very accurate controller that has achieved the initial goal, an extra issue is yet to be addressed: the controller has been shown to be robust against wind gusts which had a constant average wind speed similar to the one used for generating the optimal trajectory. While this analysis is important and was required, the NMPC should also be tested against longer time horizons where the average wind speed is not constant but changes with slow dynamics. As a result, it is critical to understand and address the following questions:

- Is the system still robust/efficient when the average wind speed differs too much from the wind speed used for computing the optimal trajectory?
- If a single reference trajectory can not be used for the total range of wind speeds, how can online feasible and optimal trajectories be generated so that stability and efficiency is always ensured?

To solve the previous inquiries, the next chapter will examine the consequences of long term wind speed profiles as well as propose a general theoretical result to cope with them: *Time Warpable Dynamical Systems*.

Chapter 7

Time Warping

7.1 Motivation

Regard a tracking NMPC using a real time iteration scheme [37], where the tracking trajectory Y_{track} is updated by a shifting strategy to increase the controller stability (at each iteration only the last value of the trajectory is updated whereas the others are shifted one time point forward). Consider as well that the equations of motion depend on a term $p(t) \in \mathbb{R}$, where $p(t)$ can represent a time variable parameter or an uncontrolled input.

Since the controller needs to track feasible trajectories in order to ensure stability, the NMPC requires a way of obtaining feasible tracking trajectories as a function of the possible values of $p(t)$. Moreover, if the tracking trajectories are also required to be optimal with respect to some cost function, the algorithm to generate trajectories online should also enforce optimality.

A simple solution would be to forget about $p(t)$ and treat it as a disturbance, relying on the performance of the NMPC to keep the system stable. However, this solution is not only too risky but in many cases also practically infeasible. The disturbances can achieve too large values and make the system unstable. Furthermore, even if the controller was stable enough, this solution could not guarantee any sort of optimality since the tracking trajectory would be unique and independent of $p(t)$.

A more complicated solution would be to precompute in a first step a discrete set of feasible and optimal trajectories $\Omega_{\text{Y}} = \{Y_{\text{ref}_{\text{p}_1}}, Y_{\text{ref}_{\text{p}_2}}, \ldots, Y_{\text{ref}_{\text{pm}}}\}$ as a function of a discrete set of p-values $\Omega_{\text{p}} = \{p_1, p_2, \ldots, p_m\}$. Then, in a second step, make the NMPC switch between tracking trajectories in a discrete manner according to the relative value of $p(t)$ and the discrete values in Ω_{p}. This solution has three main drawbacks:

(i). A discrete change of trajectories is never optimal since the trajectories are discrete and the disturbance $p(t)$ varies in a continuous manner. On the one hand, this forces the NMPC to track infeasible trajectories when $p(t)$ is not exactly one of the discrete values, creating in turn difficulties to ensure stability. On the other hand, even though stability could be ensured, if the NMPC is tracking optimal trajectories but $p(t)$ is not exactly one of the discrete values in Ω_{p}, the NMPC will work in a regime where the tracking trajectories are no longer optimal and the system efficiency decreases.

(ii). A second disadvantage is the complexity of precomputing the discrete set of feasible trajectories. Specially, if the trajectories are generated via an Optimal Control Problem to achieve performance maximization, the time required to solve each individual OCP can become extremely long.

(iii). Finally, whenever the NMPC has to switch between two tracking trajectories of the discrete sets, the NMPC has also to find the two points in the trajectories that, when

switching from one to the other, guarantee a clean switch without incurring unstable situations. In practice, finding such points is not a trivial task and guaranteeing stability is even harder.

A much better approach would be to use the novel concepts proposed and developed in this thesis. We define this set of new ideas as *Time Warping Theory*, and they will be used to generate optimal and feasible trajectories as a continuous function of $p(t)$ and with very little computational effort. This innovative approach overcomes the disadvantages of the previously mentioned methods and leads to a fully optimal and stable system.

As a drawback, it is important to remark that the developed theory and algorithms will not apply to every dynamical system, but, only to those with a specific structure in the equations of motion. Nevertheless, we still believe that many systems can benefit from this algorithm, and in particular, its application and large benefits will be illustrated in the kite system of the Skysails company, where time warping will be the solution to obtain optimal and feasible online trajectories as a function of the wind speed v_{w}.

7.2 Warping Theory

7.2.1 Theoretical Idea

Definition 7.1 (Warped Time Frame $\boldsymbol{\tau}$)**.** Consider a real time frame t which is used to describe any motion of a dynamical system. A *warped time frame* τ with respect to t can be defined by formulating the relation between the time variations $\mathrm{d}t$ and $\mathrm{d}\tau$ in both frames. This relation is called warping factor $\dot{w}(t)$ and is defined as:

$$\frac{\mathrm{d}\tau}{\mathrm{d}t} = \dot{w}(t),$$

with $\dot{w}(t) > 0,\ \mathrm{d}t > 0$ and $\mathrm{d}\tau > 0$. Looking at the above equation, it is easy to see that given an analytical expression for $\dot{w}(t)$, τ can be easily computed as a function of t by integrating $w(t) = \int_0^t \dot{w}(t)\mathrm{d}t$.

Figure 7.1 illustrates the time motion differences between t and τ for different warping factors. In particular, the blue line represents the case where $\dot{w}(t)$ is constant and bigger than one leading to a motion in the time frame τ relatively faster than in t. In contrast, the red line represents the opposite behavior: $\dot{w}(t)$ is still constant but the motion in t is now faster than in τ. Finally, the yellow line illustrates a general case where the warping factor varies as a function of time and the time variation in t with respect to τ is not constant.

It is important to remark that the warping operation is bidirectional, i.e. any real time frame t can be warped to obtain the time frame τ, but τ could simply be warped back to obtain t by using $\frac{\mathrm{d}t}{\mathrm{d}\tau} = \frac{1}{\dot{w}(t)}$. Moreover, it is also important to note that, according to our definition, time is a strictly positive monotonic function, i.e. t and $\tau = w(t)$ must be strictly positive monotone.

Definition 7.2 (Warpable Dynamical System)**.** Let's regard a general dynamical system defined by the equations of motion $\dot{x}(t) = \Phi(x(t), u(t), p(t), t) = \Theta(t)$, where t represents the time variable defining the time frame, $x \in \mathbb{R}^{n_x}$ is the state of the system, $u \in \mathbb{R}^{n_u}$ the inputs of the system and where $p \in \mathbb{R}^{n_p}$ represents time dependent parameters. A different but equally valid interpretation of p could be the uncontrolled inputs or system disturbances.

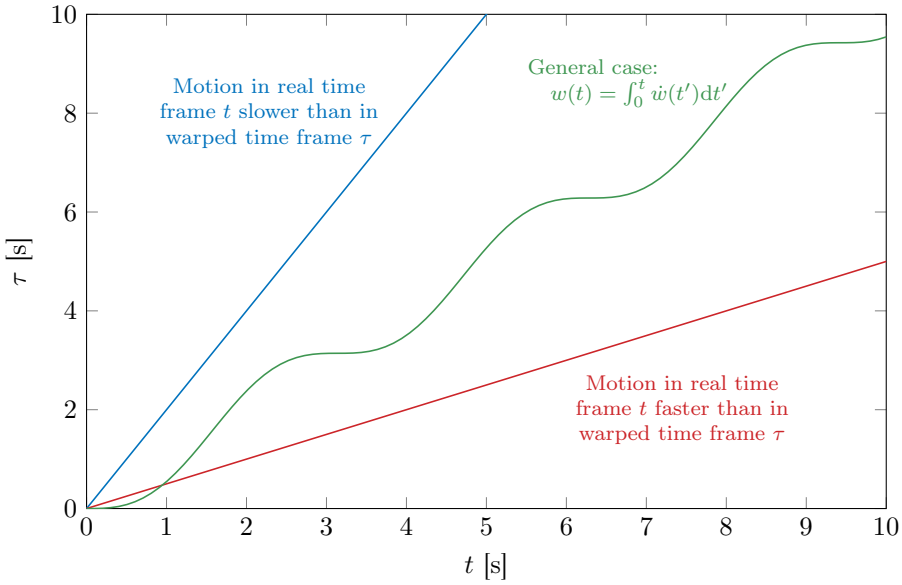


FIGURE 7.1: Warping factor and time variation in time frames τ, t

If there exist a decomposition of the equations of motion such as:

$$\begin{aligned}
\dot{x}(t) &= p(t) \cdot f\big(x(t), u_2(t)\big) + C \cdot u_1(t) \cdot l\big(x(t), u_2(t)\big) \\
&= \quad p(t) \cdot g(t) \quad + \quad C \cdot u_1(t) \cdot s(t), \\
&\text{with :} \\
&\quad p(t) \in \mathbb{R}, \\
&\quad u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix}, \\
&\quad u_1(t) \in \mathbb{R}^m \text{ and } C \in \mathbb{R}^{n_x \mathrm{x}(n_u - m)} \text{ as a selection matrix,}
\end{aligned} \tag{7.1}$$

then we define the system as a *Warpable Dynamic System.*

Theorem 7.3 (Time warped dynamical system)**.** *Given a warpable dynamical system as defined by Equation*(7.1)*, then:*

(i). *The system dynamics can be defined in a different and warped time frame τ as:*

$$\begin{aligned}
\dot{x}_{\mathrm{ref}}(\tau) = \frac{\mathrm{d}x}{\mathrm{d}\tau} &= p_{\mathrm{ref}} \cdot f\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref}2}(\tau)\big) + C \cdot u_{\mathrm{ref}1}(\tau) \cdot l\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref}2}(\tau)\big) \\
&= \quad p_{\mathrm{ref}} \cdot g_{\mathrm{ref}}(\tau) \quad + \quad C \cdot u_{\mathrm{ref}1}(\tau) \cdot s_{\mathrm{ref}}(\tau),
\end{aligned} \tag{7.2}$$

with linear dependent controls:

$$u_{\mathrm{ref}1}(\tau) = \frac{u_1(t)}{\dot{w}(t)}, \tag{7.3}$$

where the warping factor between t and τ is defined as:

$$\dot{w}(t) = \frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{p(t)}{p_{\mathrm{ref}}}, \tag{7.4}$$

$$w(t) = \int_0^t \frac{p(t')}{p_{\mathrm{ref}}} \mathrm{d}t' = \tau, \tag{7.5}$$

and where the time dependency of the parameter $p(t)$ is removed and substituted by a constant parameter $p_{\mathrm{ref}} > 0$.

(ii). *$g(t) = g_{\mathrm{ref}}\big(w(t)\big)$, $s(t) = s_{\mathrm{ref}}\big(w(t)\big)$, i.e. the state derivative with respect to t of the system in the real time frame is equivalent to the the state derivate with respect to τ in the warped time frame.*

(iii). *$x(t) = x_{\mathrm{ref}}\big(w(t)\big)$, i.e. the state of the system at time t in the real time frame is equivalent to the state at a time $w(t)$ in the warped time frame.*

(iv). *$u_2(t) = u_{\mathrm{ref}2}\big(w(t)\big)$ if f is injective, otherwise $u_2(t) = u_{\mathrm{ref}2}\big(w(t)\big)$ provides one of the possible solutions to achieve $x(t) = x_{\mathrm{ref}}\big(w(t)\big)$.*

Proof - (ii). Considering that τ can be expressed as a function of t by $\tau = w(t)$:

$$\begin{aligned}
\frac{\mathrm{d}x}{\mathrm{d}\tau} &= p_{\mathrm{ref}} \cdot g_{\mathrm{ref}}\big(w(t)\big) + C \cdot u_{\mathrm{ref}1}\big(w(t)\big) \cdot s_{\mathrm{ref}}\big(w(t)\big) \\
\frac{\mathrm{d}x}{\mathrm{d}\tau} = \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = \dot{x}\frac{1}{\dot{w}(t)} &= \frac{1}{\dot{w}(t)}\Big(\quad p(t) \cdot g(t) \quad + \quad C \cdot u_1(t) \cdot s(t) \quad \Big) \\
&\overset{(7.4)}{=} \quad \frac{p_{\mathrm{ref}}}{p(t)} p(t) \cdot g(t) \quad + \quad C \cdot \frac{u_1(t)}{\dot{w}(t)} \cdot s(t) \\
&\overset{(7.3)}{=} \quad p_{\mathrm{ref}} \cdot g(t) \quad + \quad C \cdot u_{\mathrm{ref}1}(\tau) \cdot s(t)
\end{aligned}$$

$$\implies p_{\mathrm{ref}} \cdot \Big(g(t) - g_{\mathrm{ref}}\big(w(t)\big)\Big) + C \cdot u_{\mathrm{ref}1}\big(w(t)\big) \cdot \Big(s(t) - s_{\mathrm{ref}}\big(w(t)\big)\Big) = 0. \tag{7.6}$$

Then, since $u_{\mathrm{ref}1}\big(w(t)\big)$ is a system control and can take any value (positive, negative or zero), and since the dynamics $g(t)$, $s(t)$, $g_{\mathrm{ref}}\big(w(t)\big)$ and $s_{\mathrm{ref}}\big(w(t)\big)$ are not a function of $u_{\mathrm{ref}1}\big(w(t)\big)$:

$$\begin{aligned}
&\begin{cases} u_{\mathrm{ref}1}\big(w(t)\big) \text{ can be } = 0 \\ p_{\mathrm{ref}} > 0 \end{cases} && \overset{(7.6)}{\implies} g(t) = g_{\mathrm{ref}}\big((w(t)\big). \\
&\begin{cases} u_{\mathrm{ref}1}\big(w(t)\big) \text{ can be } \neq 0 \\ g(t) = g_{\mathrm{ref}}\big((w(t)\big) \end{cases} && \implies s(t) = s_{\mathrm{ref}}\big(w(t)\big).
\end{aligned} \tag{7.7}$$

□

Proof - (i). Using the previous result:

$$\begin{aligned}
\dot{x}_{\text{ref}}(\tau) &= \frac{\mathrm{d}x}{\mathrm{d}\tau} = \frac{\mathrm{d}x}{\mathrm{d}t}\frac{\mathrm{d}t}{\mathrm{d}\tau} = \dot{x}\frac{1}{\dot{w}(t)} \\
&\overset{(7.4)}{=} \frac{p_{\text{ref}}}{p(t)}p(t)\cdot g(t) \quad + \quad C\cdot\frac{u_1(t)}{\dot{w}(t)}\cdot s(t) \\
&\overset{(7.3)}{=} p_{\text{ref}}\cdot g(t) \quad + \quad C\cdot u_{\text{ref}1}(\tau)\cdot s(t) \\
&\overset{(7.7)}{=} p_{\text{ref}}\cdot g_{\text{ref}}(\tau) \quad + \quad C\cdot u_{\text{ref}1}(\tau)\cdot s_{\text{ref}}(\tau).
\end{aligned}$$

□

Proof - (iii). By definition:

$$\begin{aligned}
x(t) &= \int_0^t \big(p(t')\cdot g(t') + u_1(t')\cdot s(t')\big)\mathrm{d}t', \\
x_{\text{ref}}(\tau) &= \int_0^\tau \big(p_{\text{ref}}\cdot g_{\text{ref}}(\tau') + C\cdot u_{\text{ref}1}(\tau')\cdot s_{\text{ref}}(\tau')\big)\mathrm{d}\tau'.
\end{aligned}$$

Therefore:

$$\begin{aligned}
x_{\text{ref}}(w(t)) &= \int_{w(0)=0}^{w(t)=\int_0^t \frac{p(t')}{p_{\text{ref}}}\mathrm{d}t'} \big(p_{\text{ref}}\cdot g_{\text{ref}}(\tau') + C\cdot u_{\text{ref}1}(\tau')\cdot s_{\text{ref}}(\tau')\big)\mathrm{d}\tau' \\
&= \int_0^t \Big(p_{\text{ref}}\cdot g_{\text{ref}}\big(w(t')\big) + C\cdot u_{\text{ref}1}\big(w(t')\big)\cdot s_{\text{ref}}\big(w(t')\big)\Big)\dot{w}(t')\mathrm{d}t' \\
&= \int_0^t \Big(p_{\text{ref}}\cdot g_{\text{ref}}\big(w(t')\big)\cdot\frac{p(t')}{p_{\text{ref}}} + C\cdot u_{\text{ref}1}\big(w(t')\big)\cdot\dot{w}(t')\cdot s_{\text{ref}}\big(w(t')\big)\Big)\mathrm{d}t' \qquad (7.8) \\
&= \int_0^t \Big(p(t')\cdot g_{\text{ref}}\big(w(t')\big) + C\cdot u_1(t')\cdot s_{\text{ref}}\big(w(t')\big)\Big)\mathrm{d}t' \\
&\overset{(7.7)}{=} \int_0^t \Big(p(t')\cdot g(t') + C\cdot u_1(t')\cdot s(t')\Big)\mathrm{d}t' \\
&= x(t).
\end{aligned}$$

□

Proof - (iv). By considering (ii):

$$\begin{aligned} g(t) = g_{\text{ref}}\big(w(t)\big) &\overset{(7.1)}{\iff} f\big(x(t), u_2(t)\big) = f\Big(x_{\text{ref}}\big(w(t)\big), u_{\text{ref2}}\big(w(t)\big)\Big) \\ &\overset{(7.8)}{\iff} f\big(x(t), u_2(t)\big) = f\Big(x(t), u_{\text{ref2}}\big(w(t)\big)\Big) \\ &\overset{f \text{ injective}}{\implies} u_2(t) = u_{\text{ref2}}\big(w(t)\big). \end{aligned}$$

$$\begin{aligned} s(t) = s_{\text{ref}}\big(w(t)\big) &\overset{(7.1)}{\iff} l\big(x(t), u_2(t)\big) = l\Big(x_{\text{ref}}\big(w(t)\big), u_{\text{ref2}}\big(w(t)\big)\Big) \\ &\overset{(7.8)}{\iff} l\big(x(t), u_2(t)\big) = l\Big(x(t), u_{\text{ref2}}\big(w(t)\big)\Big) \\ &\overset{h \text{ injective}}{\implies} u_2(t) = u_{\text{ref2}}\big(w(t)\big). \end{aligned}$$

Thus, if f or h are injective, $u_2(t) = u_{\text{ref2}}\big(w(t)\big)$. If not, it is still true that:

$$\begin{aligned} u_2(t) = u_{\text{ref2}}\big(w(t)\big) \quad &\implies \quad \begin{cases} f\big(x(t), u_2(t)\big) = f\Big(x(t), u_{\text{ref2}}\big(w(t)\big)\Big) \\ l\big(x(t), u_2(t)\big) = l\Big(x(t), u_{\text{ref2}}\big(w(t)\big)\Big) \end{cases} \\ &\overset{(7.8),(7.1)}{\iff} \begin{cases} g(t) = g_{\text{ref}}\big(w(t)\big) \\ s(t) = r_{\text{ref}}\big(w(t)\big) \end{cases} \overset{(7.7)}{\implies} x_{\text{ref}}(w(t)) = x(t), \end{aligned} \tag{7.9}$$

thus, $u_2(t) = u_{\text{ref2}}\big(w(t)\big)$ is one of the possible solutions to obtain the desired result $x_{\text{ref}}(w(t)) = x(t)$. □

7.2.2 Warped Time Frame Interpretation

When looking at the system defined by Equation (7.2) three main questions arise:

(i). What does it mean to warp a time frame?

(ii). What do $\tau = w(t)$ and p_{ref} represent?

(iii). What is the meaning of $u_{\text{ref1}}\big(w(t)\big)$?

A time warping is a distortion on any given time frame. Another interpretation would be a change on the time velocity of the time frame. In this new time frame $\tau = w(t)$, $d\tau$ would be the new time velocity, so that the ratio $\dot{w}(t)$ between $p(t)$ and p_{ref} (ratio between $d\tau$ and dt) would characterize the ratio of the time velocities of the two time frames. An example of a motion in two different time frames warped with respect to each other is given in Figure 7.2. In that particular example, the ratio $\dot{w}(t)$ is constant leading to a linear time distortion.

$u_{\text{ref1}}\big(w(t)\big) = \frac{u_1(t)}{\dot{w}(t)} = u_1(t)\frac{p_{\text{ref}}}{p(t)}$ could be interpreted as the set of inputs in the warped time frame, which not only have to be warped on time, but also have to be amplified or attenuated with respect to the inputs $u_1(t)$ to account for the fact that the second term of Equation (7.1) is independent from the parameter $p(t)$. Figure 7.3 illustrates the difference between $u_1(t)$ and $u_{\text{ref1}}\big(w(t)\big)$.

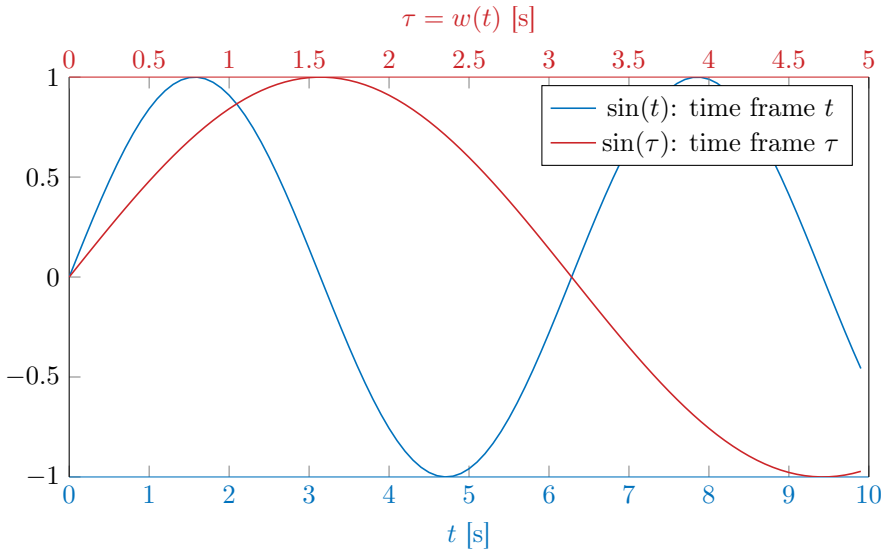


FIGURE 7.2: Linear warping of a sin function between two time frames τ, t, where $\dot{w}(t) = \frac{d\tau}{dt} = \frac{1}{2}$.

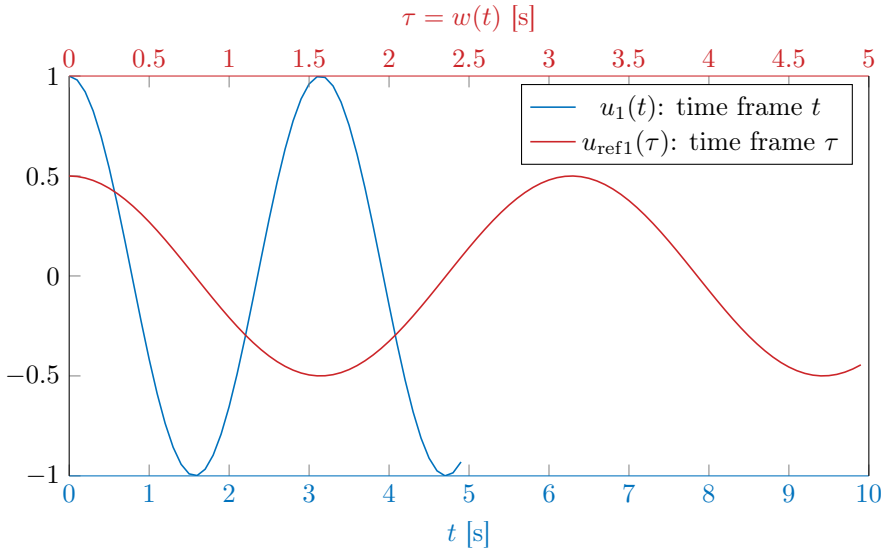


FIGURE 7.3: Comparison of controls u_1 in two time frames with a warp ratio with $\dot{w}(t) = \frac{p(t)}{p_{\text{ref}}} = \frac{1}{2}$.

It is important to notice that, the integral over time of these specific controls is constant and independent of warping effects:

$$\int_{w(0)=0}^{w(t)=\tau} u_{\text{ref1}}(\tau')\mathrm{d}\tau' = \int_0^t u_{\text{ref1}}(w(t'))\dot{w}(t')\mathrm{d}t' = \int_0^t u_1(t')\mathrm{d}t'.$$

7.2.3 Optimality of Warped Trajectories

Definition 7.4 (Warpable Optimal Control Problem (WOCP))**.** Regard a general OCP defined in a time frame t and in continuous time:

$$\begin{aligned}
\underset{y(\cdot)}{\text{minimize}} \quad & J\big(y(t)\big) = \int_0^T L\big(x(t),u(t),p(t)\big)dt + E\big(x(T)\big) \\
\text{subject to} \quad & \dot{x}(t) - \Phi\big(x(t),u(t),p(t)\big) = 0, \quad t\in[0,T], \quad \text{(dynamical model)}, \\
& h\big(x(t),u(t)\big) \le 0, \quad t\in[0,T], \quad \text{(path constraints)}, \\
& r\big(x(0),x(T)\big) \le 0, \quad \text{(boundary constraints)}
\end{aligned}$$

with $x \in R^{n_x}$, $u \in R^{n_u}$, $y(t) = \big(x(t),u(t)\big) \in \mathbb{R}^{n_y}$, $\Phi : \mathbb{R}^{n_y+1} \to \mathbb{R}^{n_x}$, $h : \mathbb{R}^{n_y} \to \mathbb{R}^{n_h}$, $r : \mathbb{R}^{n_x} \to \mathbb{R}^{n_r}$ and $p \in R^{n_p}$ representing a time varying parameter or an uncontrolled input. If it holds that:

(i). The dynamical system of the OCP is warpable, i.e.

$$\Phi\big(x(t),u(t),p(t)\big) = p(t)\cdot f\big(x(t),u_2(t)\big) + C\cdot u_1(t)\cdot l\big(x(t),u_2(t)\big).$$

(ii). The linear parameter $p(t)$ has very slow dynamics, i.e. p can be considered constant or independent of time.

(iii). The OCP path constraints are independent of $u_1(t)$, i.e.

$$h\big(x(t),u(t)\big) = h\big(x(t),u_2(t)\big).$$

(iv). The Lagrange cost of the OCP can be written as:

$$L\big(x(t),u(t),p(t)\big) = \frac{1}{T}\sum_{m=0}^{N} p^{j_m}\cdot u_1(t)^{K-j_m}\cdot f_m\big(x(t),u_2(t)\big),$$

$$\text{with}: K \text{ constant warping exponent},$$

$$j_m \in \mathbb{R} \text{ exponent coefficient of each } m\text{-sum term}.$$

Then, we define the OCP to be a *Warpable Optimal Control Problem (WOCP)*. For the sake of completeness the entire WOCP optimization problem is stated below:

$$\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad \frac{1}{T}\int_0^T \sum_{m=0}^{N} {p_1}^{j_m}\cdot u_1(t)^{K-j_m}\cdot f_m\big(x(t),u(t)\big)dt + E\big(x(T)\big) \tag{7.10a}$$

$$\text{subject to} \quad \dot{x}(t) - p\cdot g(t) + C\cdot u_1(t)\cdot s(t) = 0, \quad t\in[0,T], \tag{7.10b}$$

$$h\big(x(t),u_2(t)\big) \le 0, \quad t\in[0,T], \tag{7.10c}$$

$$r\big(x(0),x(T)\big) \le 0, \tag{7.10d}$$

with $g(t) = f\big(x(t),u_2(t)\big)$ and $s(t) = l\big(x(t),u_2(t)\big)$

Definition 7.5 (Semi-Warpable Optimal Control Problem (SWOCP))**.** Consider a general WOCP as given by Equation (7.10). The resultant OCP of adding $u_1(t)$-dependent path constraints,

$$h_2\big(x(t), u_1(t), u_2(t)\big) \leq 0, \quad t \in [0, T],$$

to the original WOCP is defined as *Semi-Warpable Optimal Control Problem (SWOCP).* A full description of the general SWOCP scheme is depicted by the following optimization problem:

$$\begin{aligned}
\underset{x(\cdot),\,u(\cdot)}{\text{minimize}} \quad & \frac{1}{T}\int_0^T \sum_{m=0}^{N} {p_1}^{j_m} \cdot u_1(t)^{K-j_m} \cdot f_m\big(x(t), u(t)\big)dt + E\big(x(T)\big) && \text{(7.11a)}\\
\text{subject to} \quad & \dot{x}(t) - p \cdot g(t) + C \cdot u_1(t) \cdot s(t) = 0, \quad t \in [0, T], && \text{(7.11b)}\\
& h\big(x(t), u_2(t)\big) \leq 0, \quad t \in [0, T], && \text{(7.11c)}\\
& h_2\big(x(t), u_1(t), u_2(t)\big) \leq 0, \quad t \in [0, T], && \text{(7.11d)}\\
& r\big(x(0), x(T)\big) \leq 0, && \text{(7.11e)}
\end{aligned}$$

with $g(t) = f\big(x(t), u_2(t)\big)$ and $s(t) = l\big(x(t), u_2(t)\big)$.

Theorem 7.6 (Optimality of time warping dynamical systems)**.** *Consider the solution of a general WOCP in a time frame t to be defined by:*

$$y^*(t) = \begin{bmatrix} x^*(t) \\ u_1^*(t) \\ u_2^*(t) \end{bmatrix}.$$

Regard also a warped trajectory of $y^*(t)$ with a warping factor $\dot{w}(t) = \frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{p}{p_{\mathrm{ref}}}$ defined in a time frame τ:

$$y^*_{\mathrm{ref}}(\tau) = y^*_{\mathrm{ref}}\big(w(t)\big) = \begin{bmatrix} x^*_{\mathrm{ref}}\big(w(t)\big) \\ u^*_{\mathrm{ref1}}\big(w(t)\big) \\ u^*_{\mathrm{ref2}}\big(w(t)\big) \end{bmatrix},$$

It holds that $y^*_{\mathrm{ref}}(\tau)$ is an optimal solution of the same WOCP but defined in a warped time frame τ and with a time horizon $\bar{\tau} = w(T)$:

$$\begin{aligned}
y^*_{\mathrm{ref}}(\tau) = \underset{y_{\mathrm{ref}}(\cdot)}{\arg\min} \quad & \frac{1}{\bar{\tau}}\int_0^{\bar{\tau}} \sum_{m=0}^{N} {p_{\mathrm{ref}}}^{j_m} \cdot u_{\mathrm{ref1}}(\tau)^{K-j_m} \cdot f_m\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref2}}(\tau)\big)d\tau + E\big(x_{\mathrm{ref}}(\bar{\tau})\big)\\
\text{subject to} \quad & \dot{x}_{\mathrm{ref}}(\tau) - p_{\mathrm{ref}} \cdot g_{\mathrm{ref}}(\tau) + C \cdot u_{\mathrm{ref1}}(\tau) \cdot s_{\mathrm{ref}}(\tau) = 0, \;\; \tau \in [0, \bar{\tau}],\\
& h\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref2}}(\tau)\big) \leq 0, \;\; \tau \in [0, \bar{\tau}],\\
& r\big(x_{\mathrm{ref}}(0), x_{\mathrm{ref}}(\bar{\tau})\big) \leq 0,
\end{aligned}$$

with: $y_{\mathrm{ref}}(\tau) = \big(x(\tau), u(\tau)\big)$, $g_{\mathrm{ref}}(\tau) = f\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref2}}(\tau)\big)$ and $s_{\mathrm{ref}}(\tau) = l\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref2}}(\tau)\big)$. For clarity purposes and from now on, this modified WOCP will be defined as τ-WOCP and the original one as t-WOCP.

Proof. Since p is constant, the warping factor also has to be:

$$\dot{w}(t) = \frac{p}{p_{\text{ref}}} = \dot{w}, \tag{7.12}$$

and because of that, the warping operation becomes a linear transformation:

$$\tau = \int_0^t \frac{p}{p_{\text{ref}}} t' dt' = \frac{p}{p_{\text{ref}}} t \implies \bar{\tau} = w(T) = \frac{p}{p_{\text{ref}}} T. \tag{7.13}$$

As a result, the cost function of the τ-WOCP can be expressed as:

$$\begin{aligned}
&J\big(y_{\text{ref}}(\tau)\big) \\
&\quad = \frac{1}{\bar{\tau}} \int_0^{\bar{\tau}=w(T)} \sum_{m=0}^{N} {p_{\text{ref}}}^{j_m} \cdot u_{\text{ref1}}(\tau)^{K-j_m} \cdot f_m\big(x_{\text{ref}}(\tau), u_{\text{ref2}}(\tau)\big) d\tau + E\big(x_{\text{ref}}(\bar{\tau})\big) \\
&\quad = \frac{1}{\bar{\tau}} \int_0^{T} \sum_{m=0}^{N} {p_{\text{ref}}}^{j_m} \cdot u_{\text{ref1}}\big(w(t)\big)^{K-j_m} \cdot f_m\Big(x_{\text{ref}}\big(w(t)\big), u_{\text{ref2}}\big(w(t)\big)\Big) \dot{w}(t) dt + E\big(x_{\text{ref}}(\bar{\tau})\big) \\
&\quad \overset{\substack{(7.12)\\(7.13)}}{=} \frac{1}{T} \int_0^{T} \sum_{m=0}^{N} {p_{\text{ref}}}^{j_m} \cdot u_{\text{ref1}}\big(w(t)\big)^{K-j_m} \cdot f_m\Big(x_{\text{ref}}\big(w(t)\big), u_{\text{ref2}}\big(w(t)\big)\Big) dt + E\big(x_{\text{ref}}(\bar{\tau})\big) \\
&\quad \overset{\substack{(7.8)\\(7.9)}}{=} \frac{1}{T} \int_0^{T} \sum_{m=0}^{N} {p_{\text{ref}}}^{j_m} \cdot u_{\text{ref1}}\big(w(t)\big)^{K-j_m} \cdot f_m\big(x(t), u_2(t)\big) dt + E\big(x(T)\big) \\
&\quad \overset{\substack{(7.3)\\(7.12)}}{=} \frac{1}{T} \int_0^{T} \sum_{m=0}^{N} \Big(\frac{p}{\dot{w}}\Big)^{j_m} \cdot \Big(\frac{u_1(t)}{\dot{w}}\Big)^{K-j_m} \cdot f_m\big(x(t), u_2(t)\big) dt + E\big(x(T)\big) \\
&\quad = \frac{1}{T \cdot \dot{w}^K} \int_0^{T} \sum_{m=0}^{N} p^{j_m} \cdot u_1(t)^{K-j_m} \cdot f_m\big(x(t), u_2(t)\big) dt + E\big(x(T)\big) \\
&\quad = \frac{1}{\dot{w}^K} \cdot J\big(y(t)\big).
\end{aligned} \tag{7.14}$$

In other words, the cost function, $J\big(y(t)\big)$, of a general WOCP and the cost function, $J\big(y_{\text{ref}}(\tau)\big)$, of its warped version τ-WOCP just differ in a constant factor $\frac{1}{\dot{w}^K}$. As a result, given $y^*(t)$ such that:

$$y^*(t) = \arg\min_{y(\cdot)} \quad J\big(y(t)\big),$$

then, $y^*(t)$ also minimizes $J(y_{\text{ref}}\big(w(t)\big))$, where $y_{\text{ref}}(w(t)) = y_{\text{ref}}(\tau)$:

$$y^*(t) = \arg\min_{y(\cdot)} \quad J\big(y(t)\big) = \arg\min_{y(\cdot)} \frac{J\big(y(t)\big)}{\dot{w}^K} = \arg\min_{y_{\text{ref}}(\cdot)} J\big(y_{\text{ref}}(w(t))\big) = y^*_{\text{ref}}(w(t)).$$

This means that, if an unconstrained minimization problem was being solved, the solution that minimizes $J\big(y(t)\big)$ should be the same as the solution that minimizes $J\big(y_{\text{ref}}(\tau)\big)$. This specific solution can be either represented by $y^*(t)$ or by $y^*_{\text{ref}}\big(w(t)\big) = y^*_{\text{ref}}(\tau)$, and where whether using one or the other will depend on the time frame (t or τ) used, i.e. it will depend on whether the cost function of the t-WOCP or the τ-WOCP is being solved.

Nevertheless, since an OCP is not an unconstrained minimization problem (dynamics, path, and boundary constraints are also enforced), optimality does not follow directly from Equation (7.14). However, if the constraints of both WOCPs were proven to be equivalent, together with (7.14), these two facts would lead to both WOCPs sharing the same optimal solution.

Boundary constraints:

$$\begin{aligned} & r\big(x_{\text{ref}}(0), x_{\text{ref}}(\bar{\tau})\big) && \leq 0 \\ \overset{\substack{(7.5)\\(7.13)}}{=} \; & r\Big(x_{\text{ref}}\big(w(0)\big), x_{\text{ref}}\big(w(T)\big)\Big) && \leq 0 \\ \overset{(7.8)}{=} \; & r\big(x(0), x(T)\big) && \leq 0. \end{aligned} \tag{7.15}$$

Path constraints:

$$\begin{aligned} & h\big(x_{\text{ref}}(\tau), u_{\text{ref2}}(\tau)\big) \leq 0, \quad \tau \in [0, \bar{\tau}] \\ \overset{\substack{(7.8)\\(7.9)\\(7.13)}}{=} \; & h\big(x(t), u_2(t)\big) \leq 0, \quad \frac{p}{p_{\text{ref}}} t \in [0, \frac{p}{p_{\text{ref}}} T] \\ = \; & h\big(x(t), u_2(t)\big) \leq 0, \quad t \in [0, T]. \end{aligned} \tag{7.16}$$

Dynamic constraints:

$$
\begin{aligned}
& \dot{x}_{\mathrm{ref}}(\tau) && - p_{\mathrm{ref}} \cdot g_{\mathrm{ref}}(\tau) + C \cdot u_{\mathrm{ref1}}(\tau) \cdot s_{\mathrm{ref}}(\tau) && = 0, && \tau \in [0, \bar{\tau}] \\
\overset{\substack{(7.7)\\(7.12)}}{\underset{\substack{(7.13)\\(7.3)}}{=}} & \frac{\mathrm{d}x}{\mathrm{d}\tau} && - \frac{p}{\dot{w}} \cdot g(t) + C \cdot \frac{u_1(t)}{\dot{w}} \cdot s(t) && = 0, && \frac{p}{p_{\mathrm{ref}}} t \in [0, \frac{p}{p_{\mathrm{ref}}} T] \\
\underset{(7.4)}{=} & \dot{w} \cdot \frac{\mathrm{d}x}{\mathrm{d}\tau} && - p \cdot g(t) + C \cdot u_1(t) \cdot s(t) && = 0, && t \in [0, T] \\
= & \frac{\mathrm{d}\tau}{\mathrm{d}t} \cdot \frac{\mathrm{d}x}{\mathrm{d}\tau} && - p \cdot g(t) + C \cdot u_1(t) \cdot s(t) && = 0, && t \in [0, T] \\
= & \dot{x}(t) && - p \cdot g(t) + C \cdot u_1(t) \cdot s(t) && = 0, && t \in [0, T].
\end{aligned}
\tag{7.17}
$$

As it can be seen from Equations (7.15-7.17), the constraints in both WOCPs represent the exact same information: both sets of constraints are fully equivalent with the difference of being defined in different time frames. Considering also Equation (7.14), it can be seen that solving:

$$
\begin{aligned}
\underset{x_{\mathrm{ref}}(\cdot),\, u_{\mathrm{ref}}(\cdot)}{\text{minimize}} \quad & \frac{1}{\bar{\tau}} \int_0^{\bar{\tau}} \sum_{m=0}^{N} {p_{\mathrm{ref}}}^{j_m} \cdot u_{\mathrm{ref1}}(\tau)^{K-j_m} \cdot f_m\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref2}}(\tau)\big) d\tau + E\big(x_{\mathrm{ref}}(\bar{\tau})\big) \\
\text{subject to} \quad & \dot{x}_{\mathrm{ref}}(\tau) - p_{\mathrm{ref}} \cdot g_{\mathrm{ref}}(\tau) + C \cdot u_{\mathrm{ref1}}(\tau) \cdot s_{\mathrm{ref}}(\tau) = 0, \;\; \tau \in [0, \bar{\tau}], \\
& h\big(x_{\mathrm{ref}}(\tau), u_{\mathrm{ref2}}(\tau)\big) \leq 0, \;\; \tau \in [0, \bar{\tau}], \\
& r\big(x_{\mathrm{ref}}(0), x_{\mathrm{ref}}(\bar{\tau})\big) \leq 0,
\end{aligned}
$$

in a time frame τ, is equivalent to solving:

$$
\begin{aligned}
\underset{x(\cdot),\, u(\cdot)}{\text{minimize}} \quad & \frac{1}{T\dot{w}} \int_0^{T} \sum_{m=0}^{N} {p_1}^{j_m} \cdot u_1(t)^{K-j_m} \cdot f_m\big(x(t), u(t)\big) dt + E\big(x(T)\big) \\
\text{subject to} \quad & \dot{x}(t) - p \cdot g(t) + C \cdot u_1(t) \cdot s(t) = 0, \quad t \in [0, T], \\
& h\big(x(t), u_2(t)\big) \leq 0, \quad t \in [0, T], \\
& r\big(x(0), x(T)\big) \leq 0,
\end{aligned}
$$

in a time frame t, which in turn is identical to solving:

$$\begin{aligned}
\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & \frac{1}{T}\int_0^T \sum_{m=0}^{N} {p_1}^{j_m}\cdot {u_1(t)}^{K-j_m}\cdot f_m\big(x(t),u(t)\big)dt + E\big(x(T)\big) \\
\text{subject to} \quad & \dot{x}(t) - p\cdot g(t) + C\cdot u_1(t)\cdot s(t) = 0, \quad t\in[0,T], \\
& h\big(x(t),u_2(t)\big) \le 0, \quad t\in[0,T], \\
& r\big(x(0),x(T)\big) \le 0.
\end{aligned}$$

As a result, it can be seen that the t-WOCP and the τ-WOCP solve the exact same problem, with the minor differences of being defined in different time frames and having cost functions that differ in a constant parameter. Nevertheless, these variations do not change the solution of the optimization problem, and thus, they share the same optimal solution:

$$y^*(t) = y^*_{\text{ref}}\big(w(t)\big) = y^*_{\text{ref}}(\tau).$$

As before, using $y^*(t)$ or $y^*_{\text{ref}}(\tau)$ will depend on the time frame used, or equivalently, on which WOCP (t-WOCP or τ-WOCP) is being solved. □

Corollary 7.7 (Optimality extension to Semi-Warpable Optimal Control Problems)**.** *Recalling the notation of the above sections, t-SWOCP will be used for referring to the original SWOCP, and τ-SWOCP for the same SWOCP but defined in a time frame τ and with a time horizon $\bar{\tau} = w(T)$.*

If the u_1-dependent path constraints of the t-SWOCP hold with strict inequality at the optimal solution $y^*(t)$, i.e.

$$h_2\big(x^*(t),u_1^*(t),u_2^*(t)\big) < 0, \quad t\in[0,T],$$

and at the warped version of $y^*(t)$, $y^*_{\text{ref}}(\tau)$, the u_1-dependent path constraints of the τ-SWOCP also hold with strict inequality, i.e.

$$h_2\big(x^*_{\text{ref}}(\tau),u^*_{\text{ref1}}(\tau),u^*_{\text{ref2}}(\tau)\big) < 0, \quad \tau\in[0,\bar{\tau}],$$

then, $y^*_{\text{ref}}(\tau)$ is also an optimal solution of τ-SWOCP.

In general and in contrast with the above result, if the controls $u_1(t)$ appear in the path constraints of a WOCP, i.e. if a SWOCP is being solved, the corollary does not hold: if $y^*(t)$ is the optimal solution of a t-SWOCP, $y^*_{\text{ref}}(\tau)$ is not necessarily an optimal solution for the τ-SWOCP. This can be easily proven considering that, whenever the controls $u_1(t)$ are warped to a time frame τ, they are not only warped but also amplified or attenuated by the factor $\frac{1}{\dot{w}}$, and as a result, the new controls $u_{\text{ref1}}(\tau)$ might lead to infeasible path constraints or suboptimal solutions in the original SWOCP.

Proof. The proof holds directly from the fact that, in any NLP, if an inequality constraint is inactive at the optimal solution, i.e. it holds with strict inequality, this constraint can be removed from the original NLP without modifying the optimal solution.

As a result, since the u_1-dependent path constraints hold with strict inequality at the optimal solution, they can be removed without modifying the problem solution, transforming the original SWOCPs into WOCPs. Then, having two WOCPs, Corollary 7.7 holds directly due to Theorem 7.6.

□

Due to Corollary 7.7, treating WOCPs and SWOCPs with the u_1 inequalities inactive is exactly the same. As a result, we will not distinguish between both cases and they will be referred to as WOCP.

7.3 Warping NMPC

In the field of tracking NMPC, Theorem 7.3 and 7.6 have a very important implication on the controller stability and efficiency. In particular, consider a tracking NMPC using a Real Time Iteration scheme [37] in order to control a warpable dynamical system. In this context, since the system equations of motions can be formulated as (7.1) and thus are linearly influenced by a parameter $p(t)$, the NMPC requires a way of obtaining feasible and optimal tracking trajectories as a function of the different $p(t)$ values.

In the motivation section, a couple of known methods to tackle this problem were stated. In this section, we introduce and develop a novel algorithm called *Warping NMPC* as an alternative solution to generate fully optimal and stable tracking trajectories online for the specific set of warpable dynamical systems.

7.3.1 Theoretical Foundations

Warping NMPC foundations are built up using the theoretical results of Section 7.2. In particular, Theorem 7.3 is applied to generate feasible trajectories, and then, Theorem 7.6 is exploited to show that, under some conditions, these trajectories are also optimal. As a consequence, its use is limited to warpable dynamical systems; and if feasibility is not enough and optimality has to be ensured, the range of applications is narrowed to tracking trajectories that are obtained via (S)WOCPs.

Feasible Trajectories Generation

Corollary 7.8 (Feasible trajectory transformation between warpable systems)**.** *Regard a dynamical system S_t defined by Equation* (7.1)*, i.e. linearly dependent on $p(t)$ and in the time frame t. Consider also a second dynamical system S_τ defined by Equation* (7.2)*, i.e. linearly dependent on p_ref and in the time frame τ. Regard as well that the controls $u_1(t)$ and $u_{\mathrm{ref},1}(\tau)$ are limited by the following general constraint equations:*

$$h_{u_1}\big(u_1(t)\big) \leq 0, \quad h_{u_\mathrm{ref}}\big(u_{\mathrm{ref},1}(\tau)\big) \leq 0.$$

Then:

(i). *Given that:*

$$h_{u_\mathrm{ref}}\left(u_1(t) \cdot \frac{p_\mathrm{ref}}{p(t)}\right) \leq 0, \quad t \in [0, T].$$

Then, any feasible trajectory $y_p(t) = \big(x(t),\, u_1(t),\, u_2(t)\big)$ of S_t, with $t \in [0, T]$, can be warped to obtain a feasible trajectory $y_\mathrm{ref}(\tau) = \big(x_\mathrm{ref}(\tau), u_{\mathrm{ref},1}(\tau), u_{\mathrm{ref},2}(\tau)\big)$ in S_τ, with $\tau \in [0, w(T)]$. In this case, the warping factor is $\dot{w}(t) = \frac{p(t)}{p_\mathrm{ref}}$.

(ii). *Given that:*

$$h_{u_1}\left(u_{\mathrm{ref},1}(\tau)\cdot\frac{p(t)}{p_{\mathrm{ref}}}\right)\leq 0,\quad \tau\in[0,\bar{\tau}]\ \ \textit{and}\ \ t=w^{-1}(\tau).$$

Any feasible trajectory $y_{\mathrm{ref}}(\tau)=\big(x_{\mathrm{ref}}(\tau),u_{\mathrm{ref},1}(\tau),u_{\mathrm{ref},2}(\tau)\big)$ *of* S_τ*, with* $\tau\in[0,\bar{\tau}]$*, can be warped to obtain a feasible trajectory* $y_p(t)=\big(x(t),\,u_1(t),\,u_2(t)\big)$ *in* S_{t}*, with* $t\in[0,w^{-1}(\bar{\tau})]$*. In this scenario, the warping factor is simply given by* $\dot{w}^{-1}(t)=\frac{1}{\dot{w}(t)}=\frac{p_{\mathrm{ref}}}{p(t)}$.

Proof.

(i). Part (i) holds directly as a direct result of Theorem 7.3, where $y_p(t)=\big(x(t),\,u_1(t),\,u_2(t)\big)$ and $y_{\mathrm{ref}}(\tau)=\big(x_{\mathrm{ref}}(\tau),u_{\mathrm{ref},1}(\tau),u_{\mathrm{ref},2}(\tau)\big)$.

(ii). Part (ii) holds by the bidirectional property of time warping, i.e if time is warped from t to τ with a warping factor $\dot{w}(t)=\frac{p(t)}{p_{\mathrm{ref}}}$, time can also be warped back from τ to t using a warping factor $\frac{1}{\dot{w}(t)}=\frac{p_{\mathrm{ref}}}{p(t)}$.

□

In order to generate feasible trajectories as a continuous function of $p(t)$, the result (ii) of Corollary 7.8 could be exploited. In particular, regarding t as the real time frame where the controller time grid is defined and τ as a warped version of t, a reference trajectory $y_{\mathrm{ref}}(\tau)$, obtained in a dynamical system proportional to the constant p_{ref}, could be warped to obtain a feasible trajectory $y_p(t)$ for any given $p(t)$ value. In particular, to avoid infeasible solutions due to the attenuations and amplifications of $u_{\mathrm{ref},1}(\tau)$ when warping, the reference trajectory $y_{\mathrm{ref}}(\tau)$ is generated using the parameter p_{ref} that models the worst case scenario for the constraint:

$$h_{u_{\mathrm{ref}}}\big(u_{\mathrm{ref},1}(\tau)\big)\leq 0. \tag{7.18}$$

That way, if the controls $u_{\mathrm{ref},1}(\tau)$ of the reference trajectory validate Equation (7.18), then, any warped control $u_1(t)=u_{\mathrm{ref},1}(\tau)\cdot\frac{p(t)}{p_{\mathrm{ref}}}$ should also validate it and, as a result, the warped trajectories $y_p(t)$ are always feasible.

It is important to remark that, in the warping NMPC algorithm as well as in the Corollary 7.8, only the limitations in $u_1(t)$ and $u_{\mathrm{ref},1}(\tau)$ have to be considered since they are the only quantities that are attenuated or amplified while warping. By contrast, $x(t)$, $x_{\mathrm{ref}}(\tau)$, $u_2(t)$ and $u_{\mathrm{ref},2}(\tau)$ are warped but their amplitudes do not change.

An interesting concept to look at is the relation between the time velocities and the parameters in t and τ. In the time frame t, the time velocity $\mathrm{d}t$ is constant and the parameter $p(t)$ changes with time, leading to a time dependent feasible tracking trajectory $y_p(t)$. By contrast, in time frame τ, $y_{\mathrm{ref}}(\tau)$ is constant because p_{ref} is also so, and to account for this time independence, the time velocity $\mathrm{d}\tau$ has to vary so that at every moment $\mathrm{d}\tau=\dot{w}(t)\mathrm{d}t$ and the two systems can be equivalent.

Figure 7.4 illustrates the warping concept between a reference and a feasible trajectory: on the one hand, the blue line represents the reference trajectory, y_{ref}, which is feasible for the constant p_{ref} value and in the warped system τ. On the other hand, the red line would show the warped trajectory, $y_p(t)$, which is a feasible trajectory for the real parameter $p(t)$ and which is obtained by time warping y_{ref} with a ratio $\dot{w}(t)=\frac{p(t)}{p_{\mathrm{ref}}}=0.5$.

So far, only the warping concept in continuous trajectories has been illustrated. Nevertheless, the equivalent algorithm for discrete trajectories is exactly the same and Figure 7.4

depicts it as well. Considering that the NMPC uses $N = 14$ points, then, the discrete real trajectory that the NMPC should track at the current time is given by $Y_{\text{track}} = (y_{\text{track},0}, y_{\text{track},1}, \ldots, y_{\text{track},13})$ and is represented by the circles. This trajectory is obtained by warping the discrete precomputed tracking trajectory $Y_{\text{pre}} = (y_{\text{pre},0}, \ldots, y_{\text{pre},13})$ that the NMPC would use if the real parameter $p(t)$ was exactly p_{ref} but with an NMPC horizon $T_{\text{ref}} = T\frac{p(t)}{p_{\text{ref}}}$.

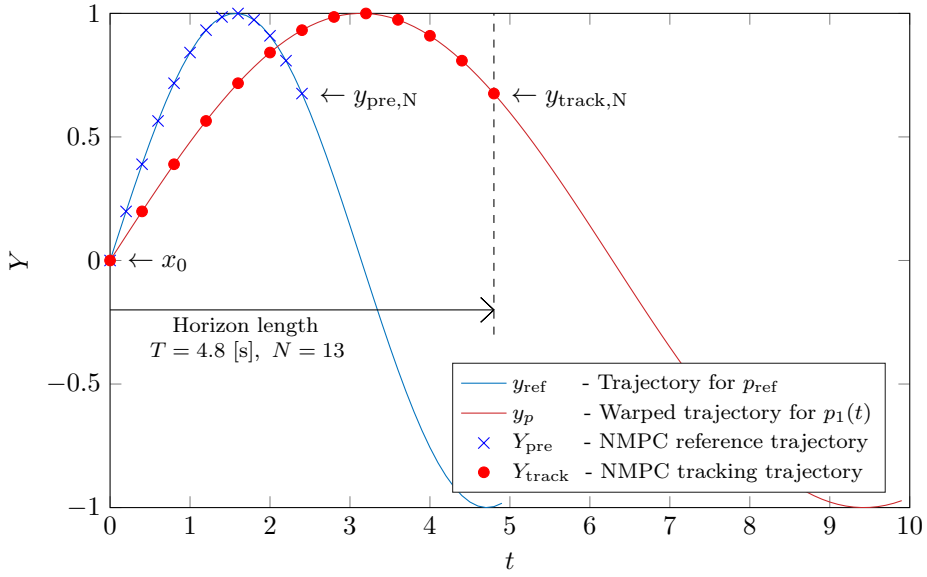


FIGURE 7.4: Warped trajectory for NMPC with a time horizon of 4.8 [s], 14 points and with a warp ratio $\frac{p(t)}{p_{\text{ref}}} = \frac{1}{2}$.

Optimal Trajectories Generation

Feasibility is in many cases a requirement that is strong enough to ensure stability of the tracking NMPC scheme. Therefore, the previous result on its own can be used to generate a variant of warping NMPC that is stable but does not ensure optimality. Nevertheless, if optimality of the warped trajectories has to be ensured, the following corollaries should be regarded:

Corollary 7.9 (Optimal trajectory transformation in WOCP)**.** *Regard a trajectory* $y_p^*(t) = \big(x^*(t), u_1^*(t), u_2^*(t)\big)$ *which optimizes a certain* t*-WOCP, which in turn considers a warpable dynamical system with a variable parameter* $p(t)$*. Consider as well the trajectory* $y_{\text{ref}}^*(\tau) = \big(x_{\text{ref}}^*(\tau), u_{\text{ref},1}^*(\tau), u_{\text{ref},2}^*(\tau)\big)$ *which optimizes the equivalent warped* τ*-WOCP, which regards a dynamical system with a constant parameter* p_{ref}*. Then, the optimal and constant trajectory* $y_{\text{ref}}^*(\tau)$ *can be warped to obtain the optimal trajectory* $y_p^*(t)$*.*

Proof. This result is an immediate consequence of Theorem 7.6 and Corollary 7.8, where no constraints of $u_1(t)$ and $u_{\text{ref},1}(\tau)$ have to be considered due to the WOCP definition. □

Corollary 7.10 (Optimal trajectory transformation in SWOCP)**.** *Regard the general* u_1*-path constraint of a SWOCP which is given by Equation* (7.11d)*. If the reference trajectory* $y_{\text{ref}}^*(\tau)$

is generated using the parameter p_{ref} that models the worst case scenario of this u_1-constraint, and in this scenario the constraint is still inactive, i.e.

$$l_2\big(x^*_{\text{ref}}(\tau), u^*_{\text{ref},1}(\tau), u^*_{\text{ref},2}(\tau)\big) < 0,$$

then, $y^*_{\text{ref}}(\tau)$ can again be warped to obtain the optimal trajectory $y^*_p(t)$ for any value of $p(t)$.

Proof. This result is an immediate consequence of Corollary 7.9, Corollary 7.7 and the fact that if p_{ref} models the worst case scenario for the constraints, the attenuations or amplifications on $u^*_{\text{ref},1}(\tau)$ should still make Equation (7.11d) inactive. □

Corollary 7.9 establishes that, in order to ensure that the warped feasible solutions $y_p(t)$ are also optimal, the base trajectory $y_{\text{ref}}(\tau)$, which is used to generate these feasible trajectories, has to be obtained as the solution of a τ-WOCP.

Corollary 7.10 establishes that the previous result can also be extended to SWOCP-generated trajectories as long as two conditions are met:

(i). The reference parameter p_{ref} creates the worst case scenario of the u_1-path constraint.

(ii). The u_1-path constraint remains inactive at the reference solution $y^*_{\text{ref}}(\tau)$.

Despite the second condition being violated, it is important to note that if the first condition is met, the warped trajectories $y^*_p(t)$ can not represent a worse u_1-path constraint value than $y^*_{\text{ref}}(\tau)$, and as a result, $y^*_p(t)$ has to be feasible. In this case, $y^*_p(t)$ would be a suboptimal solution, which despite not being fully optimal, it would still be a better approximation of the true solution than a random feasible trajectory.

7.3.2 Algorithm Implementation

In a tracking NMPC where the time frame is invariant, i.e. where the time steps on the horizon are constant, the update of the tracking trajectory is usually done by shifting backwards the previous trajectory one time step and adding a new point at the end of the trajectory. The following equations illustrate that:

$$\begin{aligned} &\text{At time } t_{\text{k}} \quad : \quad Y_{\text{track,k}} = (y_0, y_1, \ldots, y_{N-1}, y_N) \\ &\text{At time } t_{\text{k+1}}\text{: } Y_{\text{track,k+1}} = (y_1, y_2, \ldots, y_N, y_{\text{new}}) \end{aligned} \tag{7.19}$$

The selection of y_{new} can be done in different ways, but the main idea is to just update the last value and shift the others so that big changes on the optimization problem are avoided and stability is not compromised.

In order to preserve that in the proposed NMPC approach, it was decided to update the trajectory by just warping the last time interval to obtain the value y_{new} and shift the rest. The limitation of this approach is that the adaptability to the changes on $p(t)$ has a delay equal to the NMPC horizon length which in principle could arise concerns over the NMPC stability due to the late reaction. Nevertheless, this reaction delay is a common problem to the classic shift represented by Equation (7.19); and in practice, due to the short length of the horizon times on NMPC, this is rarely a real problem.

In order the explain the specific update of the last value y_{new}, let's first define several things:

(i). The continuous reference trajectory y_{ref} is obtained from solving an OCP and therefore is approximated by its discrete version $Y_{\text{ref}} = (y_{\text{ref},0}, y_{\text{ref},1}, \ldots, y_{\text{ref,M}})$.

(ii). The full discrete reference trajectory $Y_{\text{ref}} = (y_{\text{ref},0}, y_{\text{ref},1}, \ldots, y_{\text{ref,M}})$ is defined in a time grid $t_{\text{ref}} = \{t_{\text{ref},0}, t_{\text{ref},1}, \ldots, t_{\text{ref,M}}\}$, where $t_{\text{ref},0} < t_{\text{ref},1} < \ldots < t_{\text{ref,M}}$.

(iii). The time step of the NMPC controller is constant and defined by Δt.

(iv). If Δt is small enough:

$$|p(t) - p(t + \Delta t)| << p(t)$$

and therefore, $p(t)$ can be assumed to be constant between two NMPC time steps.

Lemma 7.11. *Consider a random point $y_0 \in y_p$. Then:*

(i). $y_0 \in y_{\text{ref}} \iff y_0 \in y_p$

(ii). *A time displacement of Δt on y_p starting at $y_{\text{ref},0}$ would bring the system to the same point $y_{ref,1}$ as a time displacement $\Delta t_{\text{w}} = \Delta t \frac{p}{p_{\text{ref}}}$ also starting at $y_{\text{ref},0}$ on y_{ref}.*

Proof.
(i) is proved directly because y_p and y_{ref} are warped versions of each other.
(ii) is proved considering that $p(t) = p$ is constant on Δt, i.e. $w(\Delta t) = \frac{p}{p_{\text{ref}}}\Delta t$, and by using the result (iii) of Theorem 7.3 with $t = \Delta t$:

$$y_p(\Delta t) = x(\Delta t) = x_{\text{ref}}\left(w(\Delta t)\right) = x_{\text{ref}}\left(\frac{p}{p_{\text{ref}}}\Delta t\right) = y_{\text{ref}}\left(\frac{p}{p_{\text{ref}}}\Delta t\right).$$

□

Considering Lemma 7.11, the algorithm to update y_{new} can be explained as a set of several steps/concepts:

(i). Regarding $y_{\text{track},N}$ as the last point of the real tracking trajectory Y_{track}, the algorithm uses a continuous time variable t_{c} as a continuous time index to track the equivalent point of $y_{\text{track},N}$ within the reference trajectory y_{ref}. We define this point by $y_{\text{ref}}(t_c) = y_{\text{t}_\text{c}}$.

(ii). t_{c} is updated at each iteration by increasing its value the warped time step Δt_{w}.

(iii). Then, the algorithm finds in Y_{ref} the upper y_{up} and lower y_{low} closest points to $y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}}$, and then, it approximates the new value of y_{t_c} by linear interpolation between these two points.

(iv). To find y_{up} and y_{low}, the algorithm searches in t_{ref} for the upper t_{up} and lower t_{low} time points that are closest to t_{c} , and then, it uses their indices, $k_{\text{up}}, k_{\text{low}}$, to obtain $Y_{\text{ref}}(k_{\text{up}}) = y_{\text{up}}$ and $Y_{\text{ref}}(k_{\text{low}}) = y_{\text{low}}$.

(v). Finally, the new value $y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}}$ can be easily approximated by:

$$y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}} = y_{\text{low}} + \frac{y_{\text{up}} - y_{\text{low}}}{t_{\text{up}} - t_{\text{low}}}(t_{\text{c}} - t_{\text{low}})$$

Due to result (ii) of Lemma 7.11, $y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}}$ represents the same point as the point obtained in the real time frame t by advancing a time step Δt in the continuous tracking trajectory y_p starting at $y_{\text{track},N}$, i.e. $y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}} = y_{\text{new}} = y_{\text{track},N+1}$. Algorithm 3 illustrates this procedure and table 7.1 summarizes all the variables used to do this computation.

y_{ref}	Continuous reference trajectory obtained for a certain parameter p_{ref}.
y_p	Continuous feasible trajectory obtained by warping y_{ref} using a warping factor $\dot{w}(t) = \frac{p(t)}{p_{\text{ref}}}$.
Y_{ref}	Discretized version of y_{ref}.
Y_{track}	Tracking NMPC trajectory obtained by Algorithm 3.
$y_{\text{track},k}$	k point of the real tracking trajectory Y_{track}.
t_{c}	Continuous index of y_{ref} to track the relative time position of $y_{\text{track},N}$ within the reference system y_{ref}.
y_{t_c}	Approximated value of y_{ref} at time t_{c}.
$y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}}$	Approximated value of y_{ref} at time $t_{\text{c}} + \Delta t_w$ obtained by linearly interpolating $y_{\text{up}}, y_{\text{low}}$ and $t_{\text{up}}, t_{\text{low}}$.
$t_{\text{up}}(t_{\text{low}})$	Closest upper (lower) time point in t_{ref} to t_{c}.
$y_{\text{up}}(y_{\text{low}})$	Y_{ref} value at time t_{up} (t_{low}).
Δt	Time grid interval for the NMPC controller.
Δt_{w}	Warped version of Δt with a warping factor $\dot{w}(t) = \frac{p}{p_{\text{ref}}}$.
y_{new}	Point added to the tracking trajectory Y_{track} at every iteration. $y_{\text{new}} = y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}}$

TABLE 7.1: Parameters and concepts for updating the last values of the tracking trajectory using the warping theory.

Algorithm 3 Trajectory update on Warping NMPC

$p \leftarrow$ Update_Disturbance()
$\Delta t_{\text{w}} \leftarrow \Delta t \frac{p_{\text{ref}}}{p}$
$t_{\text{c}} \leftarrow t_{\text{c}} + \Delta t_{\text{w}}$
$y_{\text{new}} \leftarrow$ Interpolating_Y$(t_{\text{c}}, Y_{\text{ref}}, t_{\text{ref}})$
return y_{new}

Function Interpolating_Y$(t_{\text{c}}, Y_{\text{ref}}, t_{\text{ref}})$:
    $t_{\text{low}} \leftarrow \arg\min_t \quad |t - t_{\text{c}}|, \quad \text{s.t. } t \leq t_{\text{c}}, \quad t \in t_{\text{ref}}$
    $t_{\text{up}} \leftarrow \arg\min_t \quad |t - t_{\text{c}}|, \quad \text{s.t. } t > t_{\text{c}}, \quad t \in t_{\text{ref}}$
    $y_{\text{low}} \leftarrow Y_{\text{ref}}(t_{\text{low}}), \quad y_{\text{up}} \leftarrow Y_{\text{ref}}(t_{\text{up}})$
    return $y_{\text{low}} + \frac{y_{\text{up}} - y_{\text{low}}}{t_{\text{up}} - t_{\text{low}}}(t_{\text{c}} - t_{\text{low}})$

Figure 7.5 illustrates the algorithm in a graphical manner: at a random time t_k, the tracking trajectory is represented by the red dots. At this time, the algorithm uses t_{c} and the point y_{t_c} to keep track of the equivalent position of the last tracking point, $y_{\text{track},N}$, within the reference trajectory Y_{ref}. Then, the algorithm computes the point $y_{\text{t}_\text{c}+\Delta\text{t}_\text{w}}$, which is ahead of $y_{\text{ref},\text{t}_\text{c}}$ by an interval Δt_{w}. Finally, it adds this point to the new tracking trajectory. This point is represented in the Figure by y_{new} and the tracking trajectory at time t_{k+1} is represented by the pentagons.

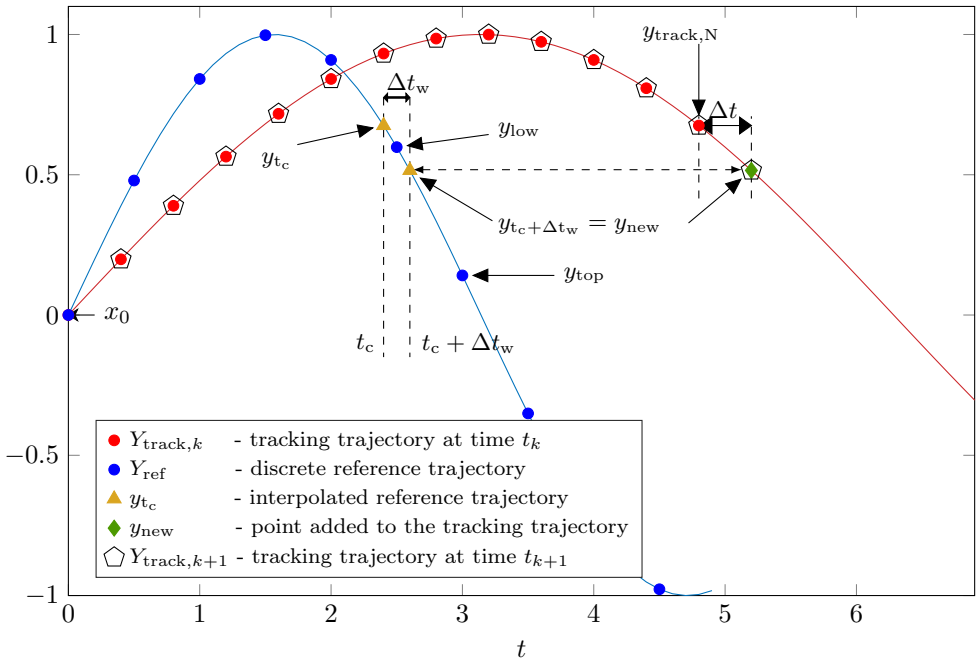


FIGURE 7.5: NMPC shifting strategy: at time t_k $Y_{\text{track},k}$ is tracked. In parallel, t_{c} and $y_{\text{t_c}}$ are used to follow the last tracking point within the reference trajectory Y_{ref}. Then, $y_{\text{t_c}+\Delta t_{\text{w}}}$ is computed by interpolation using Y_{ref}, and finally, it is added to the new tracking trajectory $Y_{\text{track},k+1}$.

As a result, Algorithm 3 establishes a way to update the tracking NMPC trajectories Y_{track} so that they remain feasible independently of the parameter value $p(t)$. Furthermore, it also establishes that if Y_{ref} is obtained using a WOCP (or via a SWOCP but validating the conditions of Corollary 7.10), then Y_{track} is not only feasible but optimal with respect to $p(t)$.

7.4 Warping Kite

The previously developed theory is of special importance when considering that, as we will prove in the coming sections, the Skysails kite is a system with warpable dynamics, and the reference trajectories of the implemented NMPC are the solutions of a SWOCP. In particular, considering the theory and algorithms developed in the previous sections, warping NMPC can be implemented in order to improve the stability and robustness of the kite controller against long term variations of wind speed.

7.4.1 Theory

In order to implement warping NMPC, a first step is to prove that the kite is indeed a warpable system and that the tracking trajectories are obtained as solutions of a SWOCP.

Warpable Kite Dynamics

Let's recall the equation of motions of the Skysails kite system:

$$\dot{\psi} = g_{\mathrm{k}} v_{\mathrm{a}} \delta + \dot{\varphi} \cos\vartheta, \tag{7.20a}$$
$$\dot{\varphi} = -\frac{v_{\mathrm{a}}}{l \sin\vartheta} \sin\psi, \tag{7.20b}$$
$$\dot{\vartheta} = -\frac{v_{\mathrm{w}}}{l} \sin\vartheta + \frac{v_{\mathrm{a}}}{l} \cos\psi, \tag{7.20c}$$
$$\dot{l} = v_{\mathrm{winch}}, \tag{7.20d}$$

with :

$$v_{\mathrm{a}} = v_{\mathrm{w}} E \cos\vartheta - \dot{l} E. \tag{7.20e}$$

By substituting Equation (7.20e) into Equations (7.20a),(7.20b) and (7.20c), (7.20b) into (7.20a) and (7.20d) into (7.20e), the following equivalent equations of motion are obtained:

$$\dot{\psi} = v_{\mathrm{w}} \Big(\cos\vartheta E g_{\mathrm{k}} \delta - \frac{E \sin\psi}{l \tan\vartheta} \cos\vartheta \Big) + v_{\mathrm{winch}} \Big(\frac{E}{l \tan\vartheta} - E g_{\mathrm{k}} \delta \Big) \tag{7.21a}$$
$$\dot{\varphi} = -v_{\mathrm{w}} \frac{E \sin\psi}{l \tan\vartheta} + v_{\mathrm{winch}} \frac{E}{l \sin\vartheta}, \tag{7.21b}$$
$$\dot{\vartheta} = v_{\mathrm{w}} \Big(-\frac{\sin\vartheta}{l} + \frac{E \cos\vartheta \cos\psi}{l} \Big) - v_{\mathrm{winch}} \frac{E \cos\psi}{l}, \tag{7.21c}$$
$$\dot{l} = v_{\mathrm{winch}}. \tag{7.21d}$$

Looking at the structure of Equations (7.21a)-(7.21d), it is obvious that the Skysails kite is a warpable dynamic system; in particular, defining the state of the system as $x = [\psi, \varphi, \vartheta, l]^\top$, the controls as $u = [v_{\mathrm{winch}}, \delta]^\top$ and the disturbance as $p = v_{\mathrm{w}}$, the equation:

$$\dot{x} = \begin{bmatrix} \dot{\psi} \\ \dot{\varphi} \\ \dot{\vartheta} \\ \dot{l} \end{bmatrix} = v_{\mathrm{w}} g_1(x, \delta) + v_{\mathrm{winch}} g_2(x, \delta), \tag{7.22}$$

with:

$$g_1(x,\delta) = \begin{bmatrix} \cos\vartheta E g_{\mathrm{k}} \delta - \frac{E \sin\psi}{l \tan\vartheta} \cos\vartheta \\ -\frac{E \sin\psi}{l \tan\vartheta} \\ -\frac{\sin\vartheta}{l} + \frac{E \cos\vartheta \cos\psi}{l} \\ 0 \end{bmatrix}, \qquad g_2(x,\delta) = \begin{bmatrix} \frac{E}{l \tan\vartheta} - E g_{\mathrm{k}} \delta \\ \frac{E}{l \sin\vartheta} \\ -\frac{E \cos\psi}{l} \\ 1 \end{bmatrix},$$

has the same structure as Equation (7.1), where $u_1 = v_{\mathrm{winch}}$, $u_2 = \delta$ and $p = v_{\mathrm{w}}$.

It is important to remark that, for the sake of clarity and simplicity, in the above derivations the time dependence was omitted; however, it is necessary to keep in mind that any of

the system states, controls, and parameters are time dependent, i.e. $x = x(t)$, $u = u(t)$ and $p = p(t)$.

Tracking Trajectories as SWOCP Solution

Let's regard now the OCP defined by (3.11) which was used to obtain the tracking trajectories for the NMPC. A clearer representation of (3.11) is given by considering the following simplifications:

(i). The control $\dot{\delta}$ was only used to obtain a smooth trajectory on the real control δ. As a result, without loss of generality, it should be possible to use δ instead of $\dot{\delta}$ as the system control.

(ii). The regularization term of the objective function did not aim at maximizing the extracted energy, but instead, at obtaining a smooth control δ. As a result, without loss of generality, the term could be removed from the OCP and the extracted energy would still be maximized.

Considering the above simplifications, (3.11) can be reformulated as:

$$\begin{aligned}
\underset{x(\cdot),u(\cdot)}{\text{minimize}} \quad & -\frac{1}{T}\int_0^T v_{\mathrm{a}}(t)^2 \dot{l}(t)dt && && \text{(7.23a)}\\
\text{subject to} \quad & \dot{x}(t) - \big(v_{\mathrm{w}}(t)f_1(t) + v_{\mathrm{winch}}(t)f_2(t)\big) = 0, && t \in [0,T],, && \text{(7.23b)}\\
& h\big(x(t)\big) \leq 0, && t \in [0,T], && \text{(7.23c)}\\
& \delta_{\min} \leq \delta(t) \leq \delta_{\max}, && t \in [0,T], && \text{(7.23d)}\\
& v_{\min} \leq v_{\mathrm{winch}}(t) \leq v_{\max}, && t \in [0,T], && \text{(7.23e)}\\
& r\big(x(T), x(0)\big) \leq 0 && && \text{(7.23f)}
\end{aligned}$$

where for notation simplicity we defined $f_1(t) = g_1\big(x(t), \delta(t)\big)$ and $f_2(t) = g_2\big(x(t), \delta(t)\big)$. Then, by expanding the cost function as:

$$\begin{aligned}
& -\frac{1}{T}\int_0^T v_{\mathrm{a}}(t)^2 \dot{l}(t)dt \\
= \; & -\frac{1}{T}\int_0^T \big(v_{\mathrm{w}}(t)E\cos\vartheta(t) - \dot{l}(t)E\big)^2 \dot{l}(t)dt \\
= \; & -\frac{1}{T}\int_0^T \big(v_{\mathrm{w}}(t)F_1\big(x(t)\big) - \dot{l}(t)E\big)^2 \dot{l}(t)dt \\
= \; & -\frac{1}{T}\int_0^T \big(v_{\mathrm{w}}(t)^2 F_2(x(t)) + \dot{l}(t)^2 E^2 - v_{\mathrm{w}}(t)\dot{l}(t)F_3\big(x(t)\big)\dot{l}(t)dt \\
= \; & -\frac{1}{T}\int_0^T \Big(v_{\mathrm{w}}(t)^2 \dot{l}(t)F_2\big(x(t)\big) + \dot{l}(t)^3 E^2 - v_{\mathrm{w}}(t)\dot{l}(t)^2 F_3\big(x(t)\big)\Big)dt, \qquad \text{(7.24)}
\end{aligned}$$

with:

$$F_2\big(x(t)\big) = F_1\big(x(t)\big)^2 \quad \text{and} \quad F_3\big(x(t)\big) = F_2\big(x(t)\big)E,$$

it can be observed that, defining $u_1(t) = v_{\text{winch}}(t)$, $p(t) = v_{\text{w}}(t)$ and K=3, the expanded cost function (7.24) has the same structure as Equation (7.10a) representing the cost function of a WOCP. Therefore, considering the v_{winch} path constraint given by (7.23e), the OCP that obtains the tracking reference trajectory is proven to be a SWOCP.

In view of the above results, given a reference trajectory Y_{ref} for a constant reference wind speed $v_{\text{w,ref}}$, warping NMPC can be used to obtain a continuous spectrum of feasible and maybe optimal tracking trajectories $Y_{v_{\text{w}}}$ for any v_{w} value.

Warping Illustration

In order to illustrate this concept better, the offline OCP solutions for 3 different wind speed values are depicted in Figure 7.6 and 7.7; in particular, Figure 7.6 illustrates the 3D view of these optimal trajectories and 7.7 the individual solution of the four system states. Looking at them, the conclusion is clear: in a 3D space, the three trajectories make the kite fly through the same physical location; however, since the wind speed is different in the three scenarios, the kite dynamics have different velocities and the state trajectories are warped versions of each other.

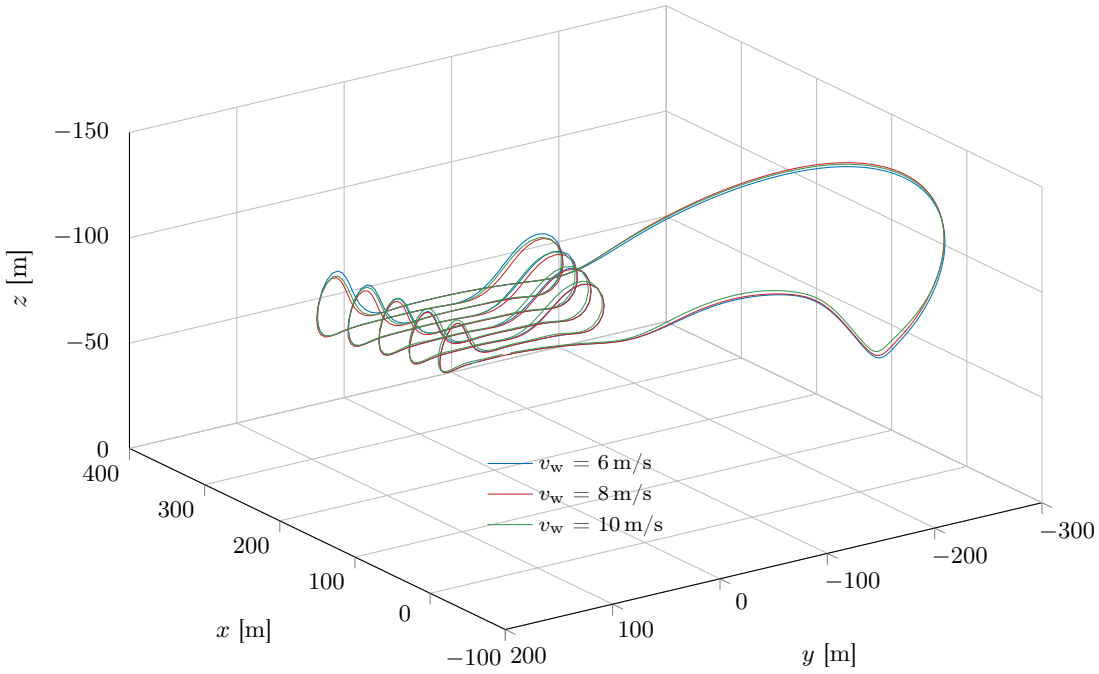


FIGURE 7.6: OCP 3D trajectory solutions for v_{w} equal to 6, 8 and 10 m/s.

To further comprehend the warping theory, Figure 7.8 represents the control inputs for the trajectories depicted in Figure 7.6 and 7.7. As previously explained, since v_{winch} represents the linear input u_1 of a warpable dynamic system, it is expected that, between warped trajectories, v_{winch} would not just warp but also be attenuated or amplified. In contrast with v_{winch}, the second control δ should only be warped. Figure 7.8 confirms and illustrates the previous hypothesis and, together with Figure 7.6 and 7.7, exemplifies the warping theory applied to the Skysails AWE system.

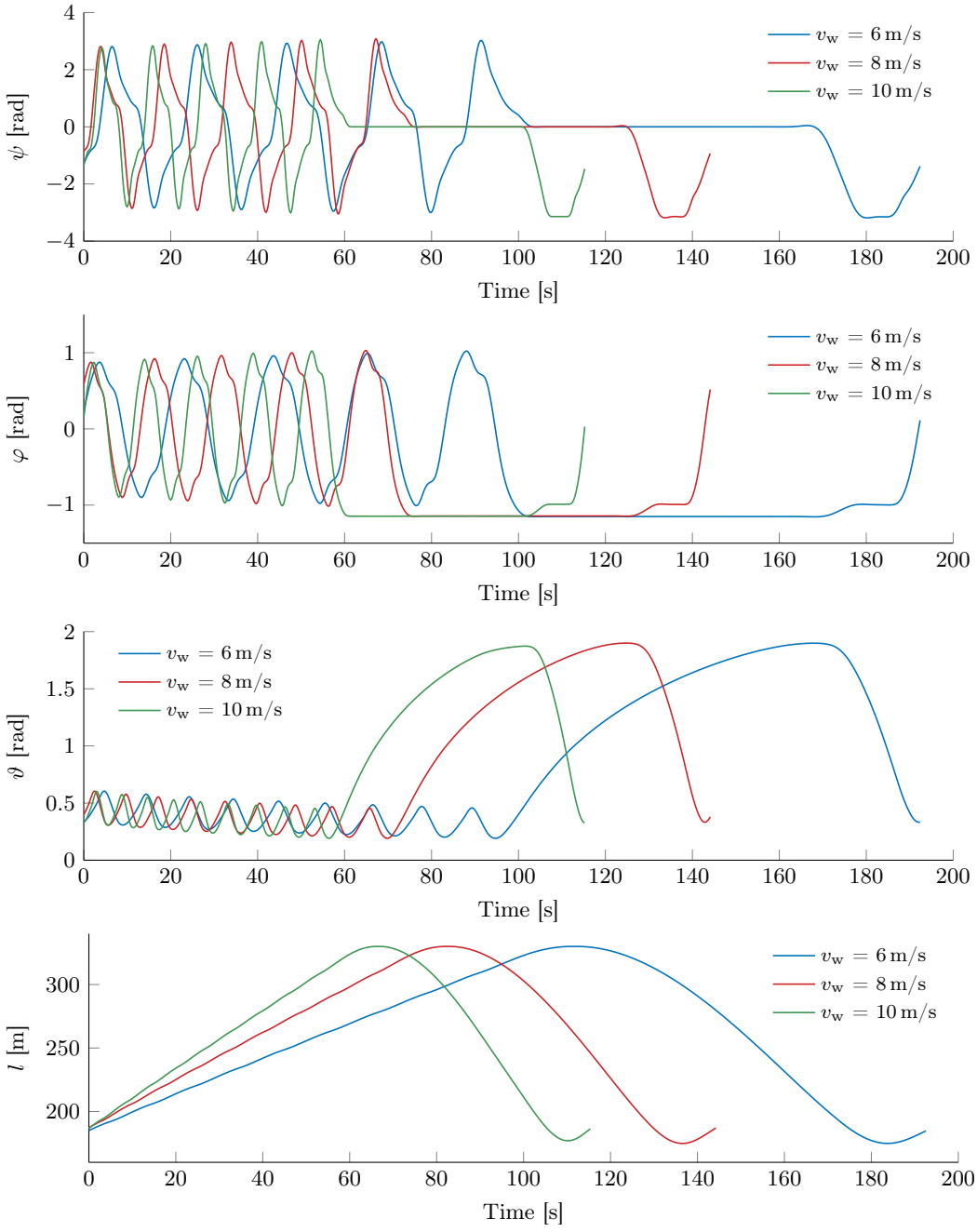


FIGURE 7.7: OCP optimal solutions for v_w equal to 6, 8 and 10 m/s. All of them are warped versions of each other.

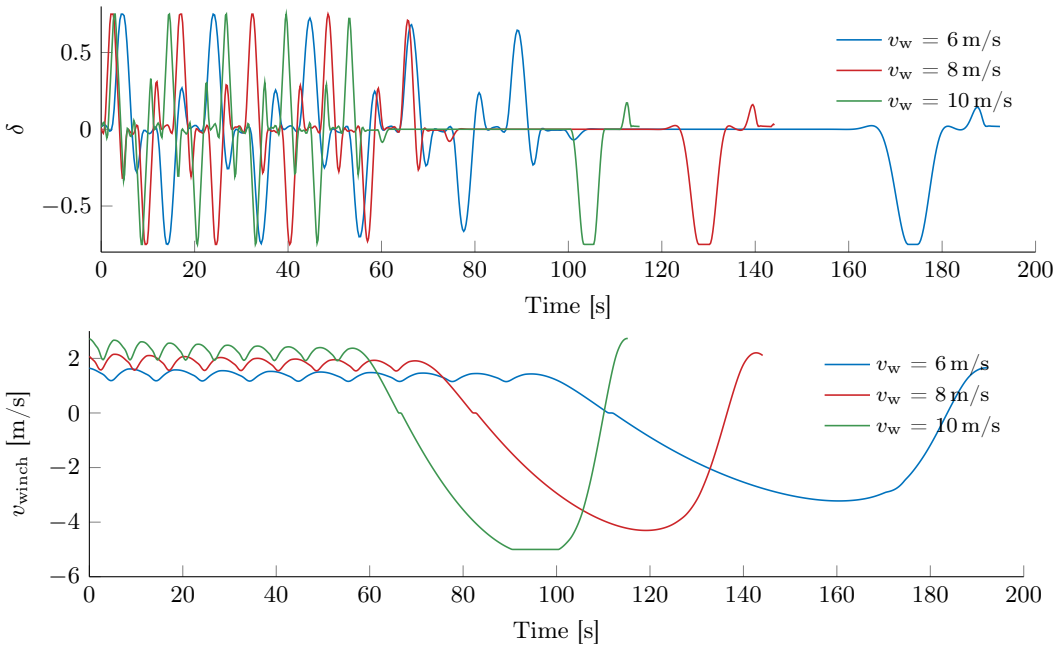


FIGURE 7.8: OCP system controls for v_w equal to 6, 8 and 10 m/s.

7.4.2 Optimality of Warped Trajectories

If the path constraint $v_{min} \leq v_{winch}(t) \leq v_{max}$ did not exist, the SWOCP would become a WOCP and warping NMPC would generate reference trajectories that are not only feasible but also optimal. In the case of having a path constraint and considering Corollary 7.10, optimality can only be obtained if this path constraint is inactive in its worst case scenario.

Looking at Figure 7.8, it can be observed that the control v_{winch} reaches a bound for $v_w = 10\,m/s$; as a consequence, the path constraint becomes active and optimality of the warped trajectories can not be guaranteed. Nevertheless, the warped trajectories might still represent suboptimal solutions; therefore, in order to evaluate if they are worthy to be used in the NMPC framework, their quality should be assessed in the following sections.

Warped Versus Optimal Offline Trajectories

The first required study to assess the quality of the suboptimal trajectories is to compare their efficiency with respect to their optimal counterpart. Particularly, to perform this evaluation, the following steps must be conducted:

(i). Compute the optimal trajectory Y^*_{worst} considering the worst v_{winch} constraint scenario. In particular, it has been observed that the higher the wind speed the closer the constraint (7.23e) is to become active; therefore, Y^*_{worst} should be computed considering the highest expected wind speed (15 m/s in this case).

(ii). Warp this trajectory Y^*_{worst} to obtain different trajectories $Y^{warp}_{v_{w,k}}$ at different wind speeds $v_{w,k}$.

(iii). Compare the efficiency of the $Y^{warp}_{v_{w,k}}$ trajectories with respect to $Y^*_{v_{w,k}}$, with $Y^*_{v_{w,k}}$ being obtained by solving the SWOCP at constant wind speed $v_{w,k}$.

Therefore, to perform the evaluation, the optimal trajectories at $v_w = 6$, 8, 10, 12, 14 and 15 m/s were computed; then, the last one was warped to obtain feasible trajectories at 6, 8, 10, 12 and 14 m/s and their efficiencies were compared. Table 7.2 illustrates this efficiency comparison.

TABLE 7.2: Efficiency comparison between trajectories obtained for a fixed v_w by solving the OCP, and trajectories obtained by warping a reference trajectory obtained at $v_w = 15$ m/s.

v_w	6 m/s	8 m/s	10 m/s	12 m/s	14 m/s	15 m/s
η_{Loyd} OCP	35.4 %	35.4 %	35.3 %	34.9 %	34.2 %	33.7 %
η_{Loyd} warping	33.7 %					

Considering that the warped trajectories have a maximum efficiency decrease of less than 2%, it seems that they represent a very good approximation of their optimal counterpart and warping NMPC should, in theory, bring several advantages to the controller. However, to have a full assessment, an evaluation of their online performance will be conducted in the next section.

Warped Versus Optimal NMPC Trajectories

Despite having already compared warped and optimal trajectories, their specific performance in a NMPC framework still remains unknown. Therefore, using the simulation scenario of Section 6.5, the NMPC performance of tracking an optimal trajectory at $v_w = 10$ m/s should be compared against the controller performance of tracking the warped counterpart obtained from an optimal trajectory at $v_w = 15$ m/s.

As depicted in Table 7.3, the decrease in the tracking performance and power efficiency is again smaller than 2 %. As a result, we can claim that warped trajectories are a very good approximation of optimal trajectories and that, as a consequence, warping NMPC can be used to model a controller that tracks nearly optimal feasible trajectories.

TABLE 7.3: NMPC performance comparison between tracking an offline trajectory obtained at $v_w = 10$ m/s and tracking a trajectory warped from $v_w = 15$ m/s to $v_w = 10$ m/s.

	R^2	η_{Loyd}
Optimal Trajectory	81.07 %	31.87 %
Warped Trajectory	79.82 %	30.32 %

7.4.3 Warping NMPC on Skysails Kite

After proving the exceptional quality of warped trajectories in the NMPC context, warping NMPC, as described in Algorithm 3, can be finally implemented.

Before doing so, it is important to make the following remark: in an ideal scenario, it would be desirable to have a NMPC that always tracks feasible trajectories; moreover, in order to achieve that, an accurate estimation of wind gusts to warp and adjust the tracking

trajectory is required. Nonetheless, as already explained in Section 6.1, the current observer is unable to estimate wind speed variations in short time intervals and the wind speed is given as a minute average. Although the former could be a problem, in Section 6.1 it was also proved that, as long as the v_w average is similar to the wind speed of the tracking trajectory, the NMPC remains stable.

In behalf of the above consideration, warping NMPC will be tested and implemented using the average v_w to continuously generate tracking trajectories. It is important to point out that this assumption sets the controller in the worst case scenario; as a result, in a future observer where the wind speed might be estimated in shorter time horizons, warping NMPC will produce even better results.

To illustrate the benefits of warping NMPC, long term variations of the wind speed should be considered. In particular, it is important to test the NMPC performance considering wind speed profiles for which the standard NMPC can no longer track the trajectory, so that it can be evaluated whether warping NMPC is able to adapt the tracking reference and stay stable. As a result, the following two controllers will be compared in two different test scenarios:

- A normal NMPC as defined in Section 6.5, i.e this controller will track a trajectory that is feasible and optimal at $10\,\mathrm{m/s}$.
- A warping NMPC where the trajectories are warped and re-adapted using the optimal trajectory at $15\,\mathrm{m/s}$.

Parameters and MHE Influence

Before explaining the specific results, it is necessary to disclose an important effect. In Section 6.2, the consequences of parameter mismatches were explained by showing that larger parameter values made the system dynamics faster and smaller parameters slower. Moreover, to cope with these parameter mismatches, an MHE was implemented.

When performing the different tests on warping NMPC, it was observed that the MHE could estimate wrong system parameters in order to improve the NMPC performance. In particular, the following two effects were observed:

(i). Independently of the real parameters values, when the real wind speed was lower than the wind speed of the reference trajectory, the MHE estimated smaller parameter values to slow down the NMPC dynamics and in turn reduce the mismatch between the real dynamics and the model used for obtaining the reference trajectory.

(ii). By contrast, when the real wind speed was higher, the MHE estimated higher parameters to make the dynamics of the NMPC faster.

In view of the previous effect, it is necessary to compare the warping NMPC performance considering two different test setups:

(i). A simulator that does not implement MHE and that considers all the disturbance except parameter mismatches so that, as a result, a fair evaluation of the warping NMPC performance without external influences can be obtained.

(ii). A simulator that considers parameter mismatches and a MHE so that, despite not having a lonely evaluation of the warping effects, the NMPC performance is studied in a more realistic scenario.

Nominal Wind Velocity Increase

A first test to analyze warping NMPC is to consider an increase of wind speed where a normal NMPC might fail due to infeasible tracking trajectories; as a consequence, a test scenario was modeled where the wind velocity increased in a time horizon of 25 minutes from a nominal 10 m/s to the maximum 15 m/s. The resulting wind profile is depicted in Figure 7.9.

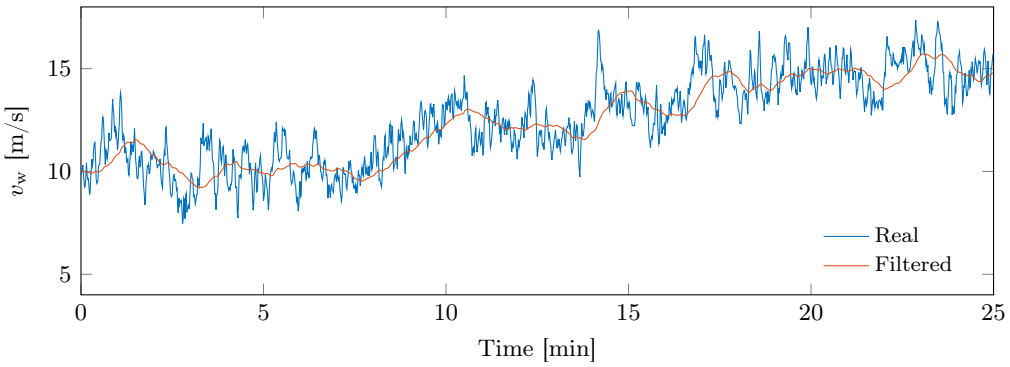


FIGURE 7.9: Wind profile considering a nominal increase from 10 m/s to 15 m/s.

The results of this experiment can be seen in Table 7.4; by analyzing them, it can be concluded that, in the case of having a NMPC facing wind speeds that steadily increase with respect to the reference value:

(i). Warping NMPC seems to be quite beneficial for power efficiency. In particular, it can be observed that, regardless of MHE and parameter mismatches, by warping the tracking trajectories an efficiency increase of 3-4 % is obtained.

(ii). Warping NMPC does not seem to really improve the tracking performance.

(iii). By using MHE an extra increase on the NMPC performance is obtained as a result of parameter tuning and of balancing the dynamics.

TABLE 7.4: Warping NMPC comparison considering a nominal wind speed profile increase from 10 m/s to 15 m/s.

	Without MHE		With MHE	
NMPC	Normal	Warping	Normal	Warping
η_{Loyd} [%]	24.95	27.88	25.67	30.09
R^2 [%]	75.59	73.99	77.5	76.63

To provide a graphical illustration, Figures 7.10 and 7.11 depict a comparison in the two states, ϑ and ψ, where the performance difference was the largest; moreover, to observe the real benefits of warping NMPC, the comparison is done for the last 100 seconds of the experiment, the moment where the wind speed is at the highest level.

By analyzing Figure 7.10 illustrating the ϑ tracking performance, the efficiency improvement of warping NMPC can be explained:

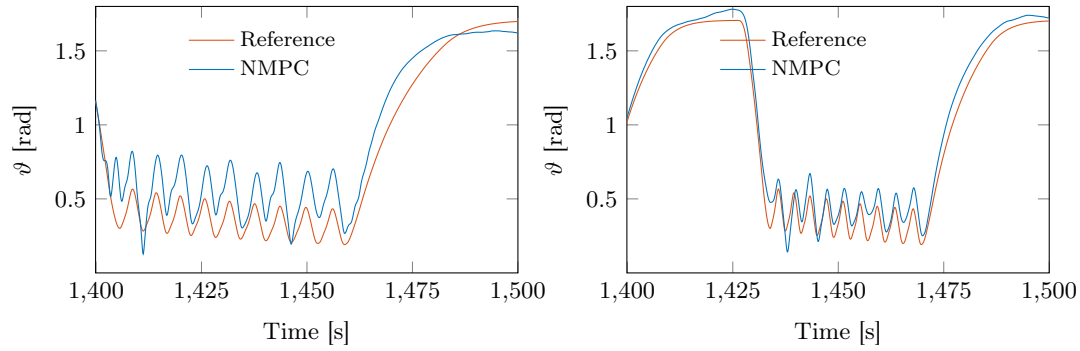


FIGURE 7.10: ϑ performance considering wind speed increases. Left: normal NMPC. Right: warping NMPC.

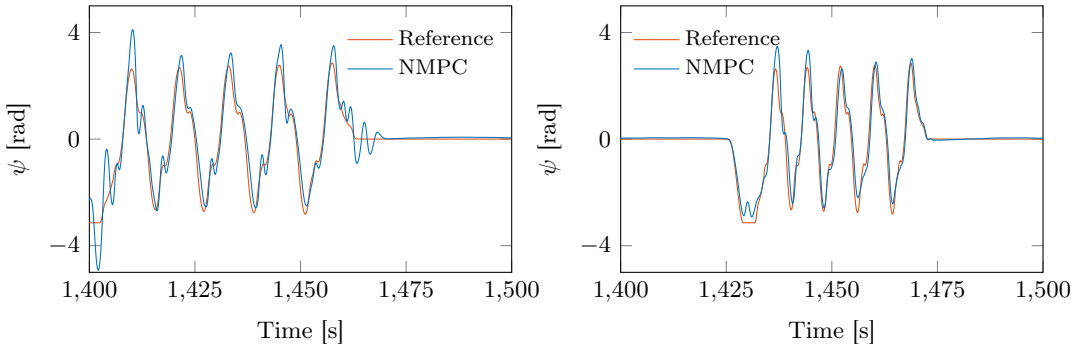


FIGURE 7.11: ψ performance considering nominal wind speed increase. Left: normal NMPC. Right: warping NMPC.

- Due to the wind speed increase, the real ϑ dynamics become faster.
- In the case of warping NMPC, since the controller warps and adapts the trajectory to the new dynamics, this is not a problem and the tracking and efficiency performance is quite good.
- However, in the standard NMPC, the tracking trajectory is too slow for the real wind speed, resulting in a controller that overshoots and struggles to follow the reference.
- As a result, due to the overshoots, the normal controller flies the kite at higher altitudes (higher ϑ) than expected, decreasing in turn the air path speed v_a and the extracted power.

Figure 7.11 illustrates a similar behavior in the angle ψ; in particular, it can be seen how, whereas standard NMPC leads to overshooting, warping NMPC solves the problem by warping and accommodating the tracking reference.

Nominal Wind Velocity Decrease

Similarly to the previous test scenario, warping NMPC should now be tested against a decreasing wind profile; in particular, a wind speed profile that drops in a time horizon of

25 minutes from the nominal 10 m/s to the minimum of 6 m/s is considered. The resulting profile is depicted in Figure 7.12.

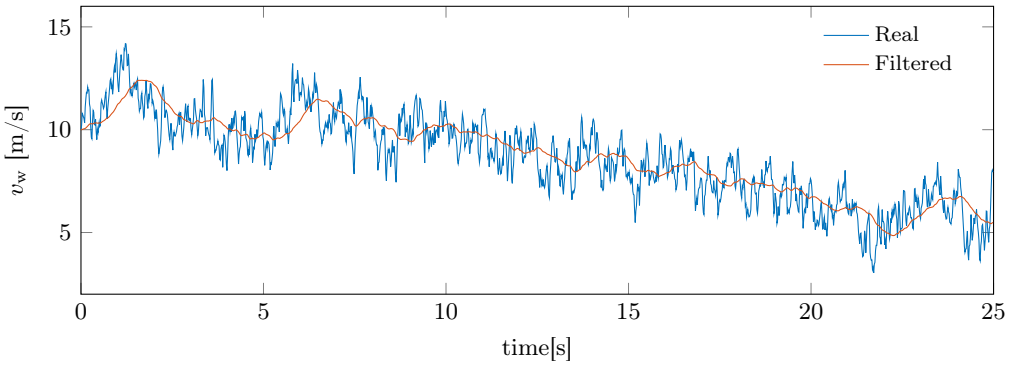


FIGURE 7.12: Wind profile considering a nominal decrease from 10 m/s to 6 m/s.

As before, the results are first illustrated by comparing the two standard metrics; in particular, considering the results depicted in Table 7.5, the following conclusions can be made:

(i). In the case of lower wind speeds than the reference, the benefits of warping NMPC seem to be larger.

(ii). In particular, as before, there is an improvement in the power efficiency; however, instead of the previous 2-3 % increase, η_{Loyd} improves by 5-8 %.

(iii). Moreover, unlike the previous scenario, warping NMPC is proven to be of extreme utility for tracking performance. This effect can be seen when comparing the 10-20 % difference in R^2 between normal and warping NMPC.

(iv). By using MHE, the controller performance can again be improved.

TABLE 7.5: Warping NMPC comparison considering a nominal wind speed profile decrease from 10 m/s to 6 m/s.

	Without MHE		With MHE	
NMPC	Normal	Warping	Normal	Warping
η_{Loyd} [%]	25.10	30.47	25.77	33.6
R^2 [%]	61.31	72.99	63.98	80.75

Finally, the graphical tracking performance is illustrated and compared in Figures 7.13-7.15. In particular, in order to show the real benefits of warping NMPC, the comparison is made at the end of the simulation horizon, time when the wind speed is at the lowest level. By observing the tremendous difference of performance in the three angles ϑ, φ and ψ, it seems clear why using warping NMPC is critical.

Related with this experiment, it is important to point out that, despite showing a terrible performance, normal NMPC still seemed to be stable. Nevertheless, it is important to explain

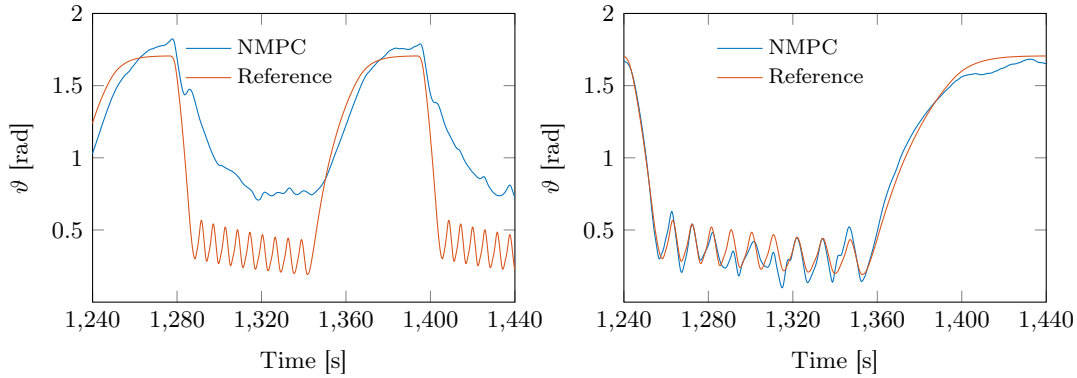


FIGURE 7.13: ϑ performance considering a nominal wind speed decrease. Left: normal NMPC. Right: warping NMPC.

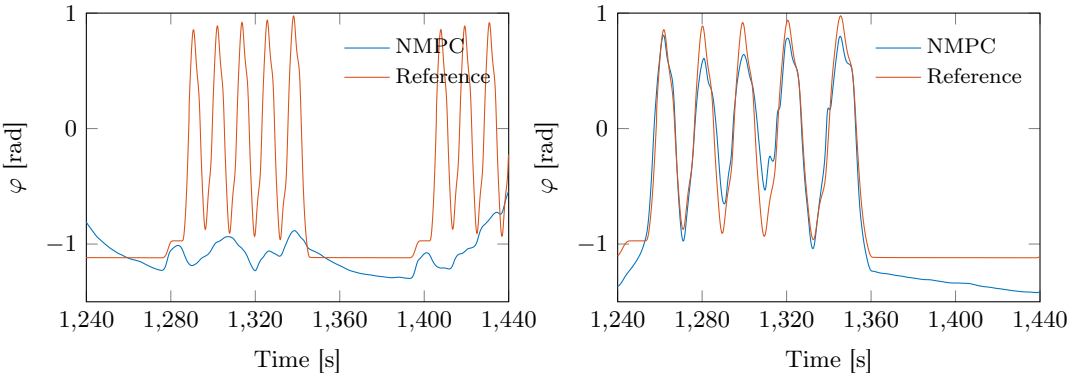


FIGURE 7.14: φ performance considering a nominal wind speed decrease. Left: normal NMPC. Right: warping NMPC.

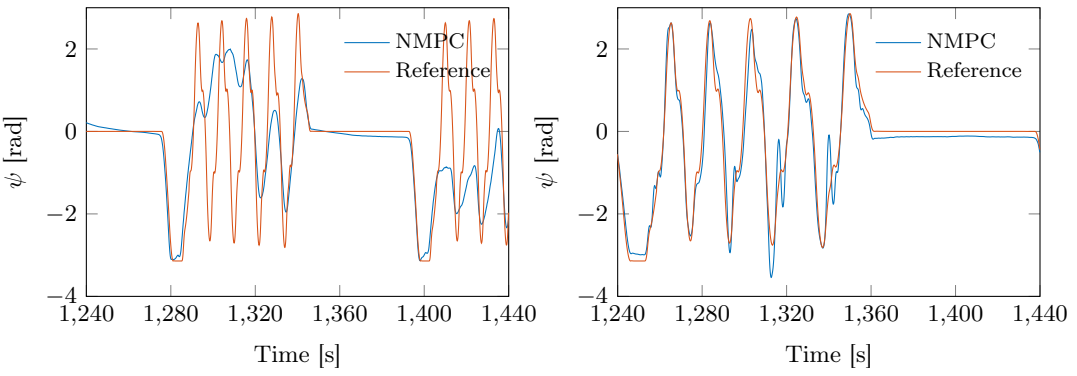


FIGURE 7.15: ψ performance considering a nominal wind speed decrease. Left: normal NMPC. Right: warping NMPC.

that it does so at the expenses of extracting a very small amount of power and performing

extremely tiny periodic flight cycles. Figure 7.16 illustrates and compares the 3D trajectory of normal NMPC versus warping NMPC; in particular, it can observed how, whereas warping NMPC is able to track the trajectory quite decently, normal NMPC just stays at a high altitude and performs very slow motions.

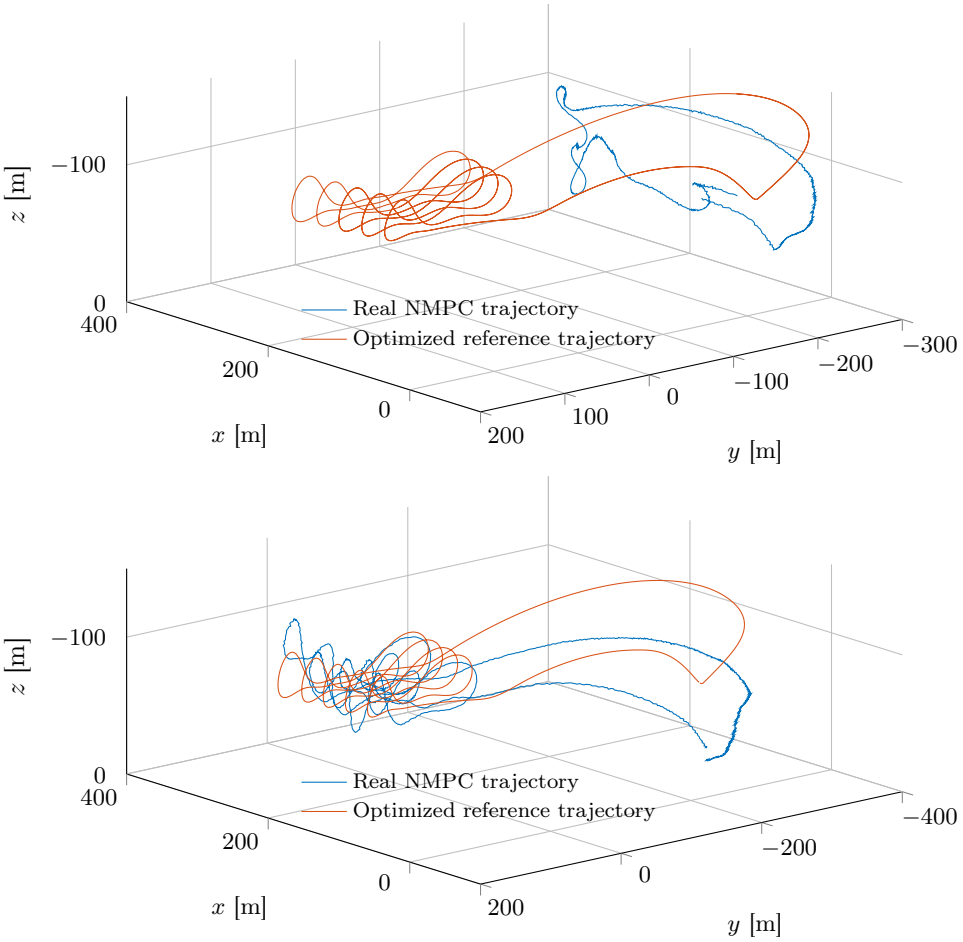


FIGURE 7.16: 3D trajectory comparison when the NMPC faces a nominal wind speed decrement. Top: normal NMPC. Bottom: warping NMPC.

7.5 Conclusion

Warpable dynamics systems have been introduced as a manifold of systems whose equations of motion are represented by (7.1). In particular, they were shown to be of special interest in the field of tracking NMPC, where they form the basis for warping NMPC, a proposed new algorithm that was proved to be useful for online generation of feasible trajectories.

Of tracking NMPC interest was also the described OCP class of warpable optimal control problems (WOCP); in particular, it was proven that, if the NMPC trajectories are optimal with respect to an WOCP, warping NMPC generates not only feasible but also optimal trajectories.

Based on WOCP, a broader OCP class called semi-warpable optimal control problems (SWOCP) was also introduced. In particular, for this type of problem it was demonstrated that, while the previous theory of tracking NMPC also applies, the online generated trajectories might be suboptimal in some cases.

As a last step, considering that the main motivation for developing the theory was overcoming long term wind disturbances, a warping NMPC scheme was implemented in the kite system in order to fly optimal and feasible trajectories independently of the wind speed.

Finally, the standard NMPC was tested against warping NMPC considering long term wind profiles. In the view of the experimental results, three aspects can be concluded:

(i). Warping NMPC is required to fly power efficient trajectories as well as to bring stability and robustness to the controller.

(ii). The tracking and efficiency improvements of warping NMPC seem to be more important for decreasing wind speeds. In particular, if the wind speed steadily decreases and warping is not present, the tracking ability of the controller is seriously compromised.

(iii). Wind speed, system parameters, and the velocity of the dynamics are strongly related. As a result, the implemented observer might estimate wrong system parameters in order to balance wind speed effects.

Chapter 8

Towards Real Life Experiments

In the past three chapters, a NMPC scheme was designed with the intention of substituting the current Skysails controller by a more efficient implementation. Fortunately, at the simulation level, the result of this design was a controller that could approximately achieve two times the efficiency rates of the current controller. Furthermore, the NMPC was not only efficient, but also proven to be robust and stable against the expected real life perturbations:

(i). A delay on the control δ that was solved by DDE modeling.

(ii). Real wind gusts considering short term variations of wind speed and direction.

(iii). Model parameter mismatches which were tackled by online estimation using MHE.

(iv). An offset error on the control δ.

(v). A real observer that produced estimation errors.

(vi). Long term variations of the wind speed for which a special theory was developed.

In behalf of the above results, the designed NMPC controller is expected to be a good candidate for replacing the classical approach; however, to have a final assessment of the possibilities that it offers, real life experiment using the Skysails prototype must be conducted.

8.1 Flight Permissions

Performing real life experiments in the field of AWE is rather complicated. A common problem that many researches have to face is the bureaucratic procedure of obtaining flight permissions; in particular, since AWE systems fly at high altitudes, any experimental setup must comply with a series of legal requirements so that accidents are avoided.

During the last years, Skysails has been obtaining periodic flight permissions that entitled them to perform some tests for short periods of time. As a result, in order to assess the work of this thesis, flight permissions also had to been requested. Unfortunately, despite a permission was approved in recent weeks for a first flight test in the middle of August of 2016, the six month time window of this Master's thesis was not enough to include these results as a part of this work.

Nevertheless, to prove and ensure that the system is ready to run in the real Skysails hardware, the NMPC will perform a last experiment in the Skysails official simulator.

8.2 Skysails Simulation Framework

Skysails has a simulation framework to simulate and visualize kite flights in 3D which implements the same communication protocol as the real kite does; as a result, by successfully testing the NMPC using this software, it is expected that the new controller will not face communication and hardware issues the day of the real experiment.

In particular, the hardware on the kite prototype employs a TCP communication protocol using an Ethernet cable to send and receive a predefined set of packages. Specifically, the system sends a data package every 100 ms including all the relevant information regarding the system state; furthermore, the data package contains 33 fields which are modeled as a floating-point number of 4 bytes. Finally, after sending the data, it waits for an incoming 2-fields package including the control values δ and v_{winch}.

As stated before, in order to create a realistic simulation environment, the simulator uses the same Ethernet-TCP connection and the same data structure. Therefore, in order to test the NMPC in a real communication framework, the simulator and the visualizer were set in a second computer, then, a TCP instance was added within the NMPC, and finally, both computers were connected with an Ethernet cable and the TCP communication performed the data exchange. Figure 8.1 depicts the TCP communication schematic between the NMPC and the kite systems (simulator and prototype).

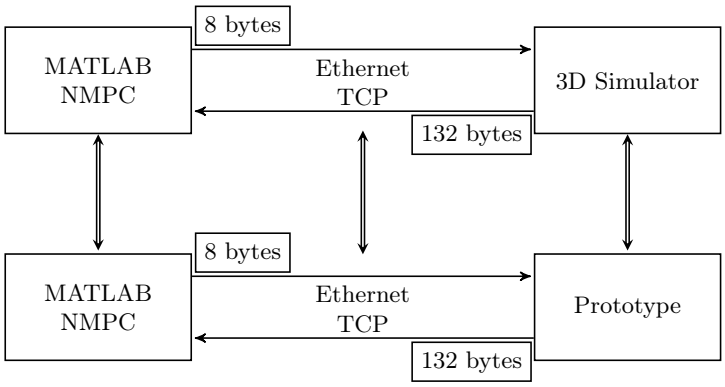


FIGURE 8.1: Schematic of the TCP connection between NMPC and the kite. Top: communication between the NMPC and the 3D simulator. Bottom: real connection with the kite prototype.

The outcome of this experimental test was excellent: the NMPC was connected without difficulties and the controller performed stable pumping cycles. To have a graphical representation of the test scenario, Figure 8.2 shows the real experimental setup and Figure 8.3 depicts the 3D visualizer with the kite being controlled by the NMPC.

It is important to remark that the NMPC was not implemented on the embedded hardware due to timing considerations; in particular, since the current NMPC computation time is approximately 10 times lower than the time step of the real hardware, it was decided that the computer was a good platform for running the NMPC in a first experimental test.

Before concluding this chapter, it is important to note that, given the performance showed in the simulation framework, we have high expectations that the NMPC will successfully control the AWE kite system.

FIGURE 8.2: Experimental setup to test the NMPC using a TCP connection with the official Skysails simulator.

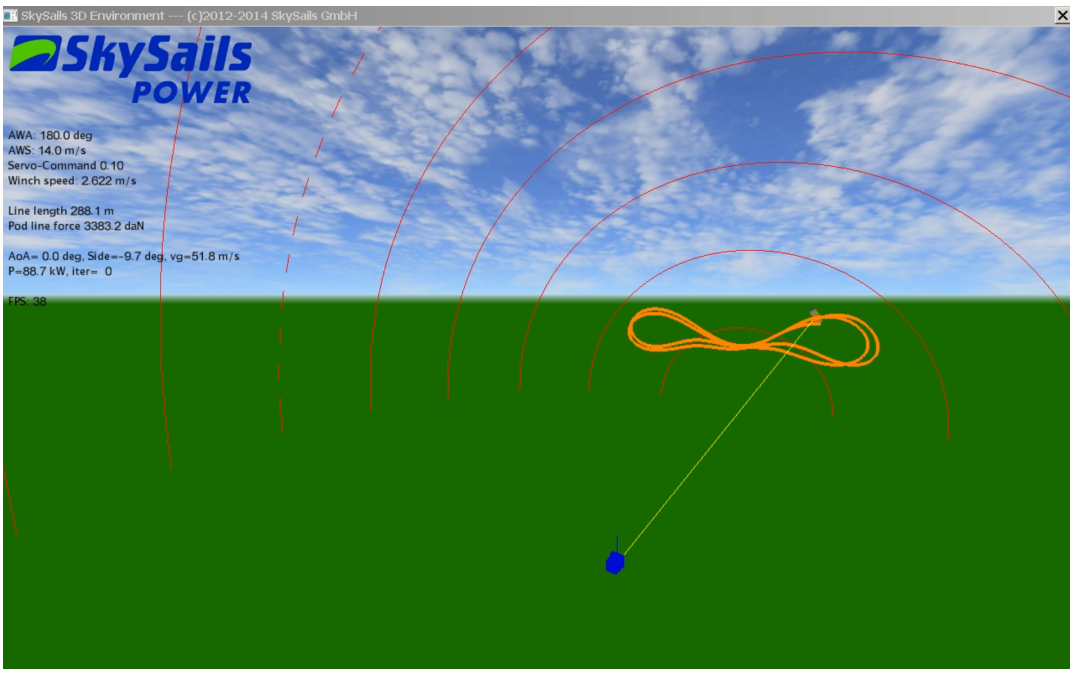


FIGURE 8.3: Official Skysails simulator controlled by the NMPC.

Chapter 9

Conclusion and Future Work

This thesis examined two different research areas in the context of optimal control and airborne wind energy. On the one hand, it aimed at modeling and analyzing diverse optimal control problems with the motivation of obtaining a robust framework to generate power efficient trajectories for a tethered kite. On the other hand, it targeted the design and implementation of a nonlinear model predictive controller intended to regulate a tethered kite to track the optimal trajectories resulting from the OCP framework.

9.1 Conclusion

In order to achieve the described objectives, a two-part approach was adopted. In particular, an independent Chapter 1 introduced and motivated the two research contributions by describing the theoretical foundations of AWE, discussing the stability and efficiency of AWE systems, and presenting the different features of a tethered kite prototype.

Part 1: Offline Optimal Control

Then, in a first part, Chapter 2 introduced a brief theoretical analysis in numerical optimization methods and optimal control.

Chapter 3 focused on the OCP framework implementation; in particular, as the basis of the research, it introduced the previous work [12] based on a quaternion formulation. Then, as a first contribution, it reformulated the flight safety conditions to improve the power efficiency by 2 %.

Within the same chapter and as a second contribution, LICQ deficiency due to model invariants was studied; particularly, the projection method was proposed as an alternative to the original invariant stabilization; then, the new method was shown to be more robust to overcome LICQ issues. Furthermore, it was also displayed that, due to the bad conditioning of the quaternion formulation, both methods are in general quite erratic and some alternative parameterization might perform better.

As a third contribution, two better alternatives that outperform the quaternion performance were proposed: a first parameterization based on Euler angles, which was shown to be more robust and stable; and a second model based on rotation matrices, which achieved larger power efficiencies. In particular, the following characteristics were proved:

- Euler angles are the most robust and efficient implementation solving more than half of the proposed OCPs while simultaneously requiring the lowest computational effort.

- In contrast with Euler angles, natural coordinates are slower and have lower success ratios. However, unlike the former, their formulation is quite linear, which, as a result, leads to more power efficient trajectories.
- The quaternion formulation was shown to be the most erratic implementation with no clear advantages over the other two.

Finally, as for the fourth contribution of Chapter 3, a study on flight topologies was conducted. In particular, it was proved that, unlike the original hypothesis of [12], topology constraints are not a requirement to derive optimal flight trajectories based on lemniscates. Moreover, an alternative flight topology was proposed; specifically, it was demonstrated that, besides lemniscates, circular trajectories are also a power efficient and safe implementation for tethered kites.

Part 2: Nonlinear Model Predictive Control

In a second part, Chapter 4 described and introduced the required theory of a real time implementation of NMPC; particularly, tracking NMPC using a real time iteration scheme was described.

Chapter 5 introduced the controller implementation using the ACADO framework; specifically, it derived the required objective function, constraints, and numerical algorithms in order to obtain a real time controller. Moreover, the dynamical model was extended to included control delays; in particular, controller stability was ensured by modeling steering command delays by a DDE.

Chapter 6 proved and tested the stability and robustness of the implemented NMPC given real life disturbances. Particularly, the controller was modeled to be robust against the following real life disturbances:

(i). Real wind gusts: fast changes in the wind speed and direction.

(ii). An offset bias in couple with a time delay in the control δ.

(iii). Parameter mismatches between the controller and the real system. A MHE was implemented to reduce the effect of this type of perturbation.

(iv). Realistic estimation errors that are created by the real system observer.

Chapter 7 introduced and derived time warping, a series of theorems and concepts that were used to tackle the ultimate and most important type of disturbances: long term wind speed variations. In particular, warpable systems and warpable OCPs were respectively defined as types of dynamical systems and optimal control problems with a very specific structure. In this context, warping NMPC was proposed as a successful and robust algorithm for online generation of optimal trajectories regardless of parameter disturbances. Specifically, the algorithm was used to generate optimal tracking trajectories as a function of the wind speed, resulting in a NMPC controller that was proved to be stable against long term wind speed variations.

Finally, Chapter 8 introduced the possibilities for real life experiments. Furthermore, it illustrated the controller performance in a real AWE simulation framework, a software environment where the NMPC displayed an excellent performance.

9.2 Future Work

The main contribution of any future work is to perform real flight tests. In particular, a first test date was set in the middle of August 2016 to conduct a first experimental setup with a real tethered kite.

Furthermore, warpable systems were shown to have very interesting properties in the field of state space control. Therefore, a deeper research must be conducted to fully explore all their features and applications. In particular, we believe that the implementation of warping NMPC can be extended to other fields such as chemical processes, areas where tracking NMPC on linearly disturbed systems might require online generation of optimal trajectories.

Finally, a more extensive research on the offline OCP properties shall be conducted; in particular, considering the frequency of appearance of SO(3) parameterizations in periodic OCPs, a more detailed analysis on the three dynamical models might be highly beneficial.

Appendices

Appendix A

Derivation of Alternative SO(3) Parameterizations

A.1 Angular Velocities in Natural Coordinates

To have an organized and consistent derivation, this appendix will be divided into three parts: first, some auxiliary relations will be defined, then, the linear velocities of the reference frame will be obtained, and finally, the angular speeds will be derived. As a first step, let's define the explicit relation between the moving frame axis and the rotational matrix elements:

$$\vec{e}_{\text{yaw}} = -\begin{bmatrix} R_{11} \\ R_{21} \\ R_{31} \end{bmatrix}, \quad \vec{e}_{\text{pitch}} = -\begin{bmatrix} R_{12} \\ R_{22} \\ R_{32} \end{bmatrix}, \quad \vec{e}_{\text{roll}} = -\begin{bmatrix} R_{13} \\ R_{23} \\ R_{33} \end{bmatrix}.$$

Then, regarding the reference frames defined in Figure 3.5, the position of the kite is given by:

$$r = -l\vec{e}_{\text{yaw}} \tag{A.1}$$

Furthermore, the air path speed vector $\vec{v}_{\text{a}}$ can be calculated as the sum of the ambient wind vector $v_{\text{w}}\vec{e}_{\text{x}}$ and the reversed kinematic velocity vector $-\dot{r}$. In particular, regarding the time derivative of A.1 and introducing the speeds v_{roll} and v_{pitch}:

$$\vec{v}_{\text{a}} = v_{\text{w}}\vec{e}_{\text{x}} - v_{\text{pitch}}\vec{e}_{\text{pitch}} - v_{\text{roll}}\vec{e}_{\text{roll}} + \dot{l}\vec{e}_{\text{yaw}}. \tag{A.2}$$

Model Assumptions

In oder to derive the aerodynamic model, two assumptions have to be done [12]:

(i). The aerodynamic forces are typically larger that the masses, and as a result, accelerations can be neglected.

(ii). The kite is assumed to always be in its aerodynamic equilibrium.

Given the above two assumptions, the aerodynamics of the system are reduced to the two conditions depicted in Figure A.1 and stated in the following:

(i). As depicted by the top view illustration of Figure A.1, the kite experiences no transversal wind flow:

$$\langle \vec{e}_{\text{pitch}}, \vec{v}_{\text{a}} \rangle = 0.$$

As a result, inserting the above expression into A.2 results in:

$$\langle \vec{e}_{\text{pitch}}, v_{\text{w}}\vec{e}_{\text{x}} \rangle - v_{\text{pitch}} + \dot{l} \underbrace{\langle \vec{e}_{\text{pitch}}, \vec{e}_{\text{yaw}} \rangle}_{=0} = 0,$$

which in turn leads to:

$$v_{\text{pitch}} = v_{\text{w}} \langle \vec{e}_{\text{pitch}}, \vec{e}_{\text{x}} \rangle = -v_{\text{w}} R_{12}$$

(ii). The lift-to-drag ratio (given by E) is constant; consequently, the kite is always assumed to fly at the same angle of attack. Considering the geometry of the side view in Figure A.1:

$$\langle \vec{e}_{\text{roll}}, \vec{v}_{\text{a}} \rangle = E \langle \vec{e}_{\text{yaw}}, \vec{v}_{\text{a}} \rangle.$$

Inserting A.2 into the above expression results in:

$$v_{\text{roll}} = v_{\text{w}} \langle R(E\vec{e}_{\text{x}} - \vec{e}_{\text{z}}), \vec{e}_{\text{x}} \rangle - E\dot{l} = v_{\text{w}} E R_{11} - v_{\text{w}} R_{13} - E\dot{l}$$

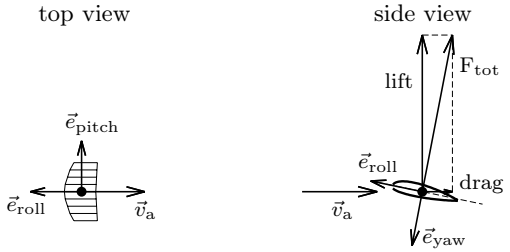


FIGURE A.1: The kite in the aerodynamic equilibrium state. The kite top view represents the absence of transversal air flow and the side view the constant lift-to-drag-ratio [12].

Angular Velocities

Considering that any steering actuation δ leads to a turn rate around the $\vec{e}_{\text{yaw}}$ axis, the angular velocity around the $\vec{e}_{\text{yaw}}$ axis can be described by the following relation:

$$\omega_{\text{yaw}} = g_{\text{k}} v_{\text{a}} \delta$$

where v_{a} is defined as the component of $\vec{v}_{\text{a}}$ in the $-\vec{e}_{\text{roll}}$ direction:

$$v_{\text{a}} = \langle -\vec{e}_{\text{roll}}, \vec{v}_{\text{a}} \rangle = -\langle \vec{e}_{\text{roll}}, v_{\text{w}}\vec{e}_{\text{x}} \rangle + v_{\text{roll}} = v_{\text{w}} E R_{11} - E\dot{l}.$$

It is important to note that the validity of this turn-rate law has been experimentally shown for different kites [13, 51, 52].

Finally, due to the nature of kite motion in a sphere, a motion in the tangent plane, as depicted in Figure A.2, can be considered; as a result, the following relations for the angular velocities of the roll and pitch axis can be established:

$$\omega_{\text{roll}} = \frac{v_{\text{pitch}}}{l} = -v_{\text{w}} R_{12}$$
$$\omega_{\text{pitch}} = -\frac{v_{\text{roll}}}{l} = \frac{E\dot{l}}{l} - \frac{v_{\text{w}} E}{l} R_{11} + \frac{v_{\text{w}}}{l} R_{13} = -\frac{v_{\text{a}}}{l} + \frac{v_{\text{w}}}{l} R_{13}$$

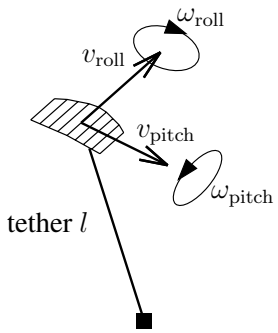


FIGURE A.2: Kite motion in the tangent plane [12].

As a final remark it is important to state that, as in Section 3.4 and due to simplicity reasons, the explicit time dependency has been omitted in the above derivations; however, it should be kept in mind that it is still present.

A.2 Relation between Euler Angles, Quaternions and Natural Coordinates

To derive the relation between natural coordinates and Euler angles, the rotations that ψ, φ and ϑ represented in Section 1.2.2 should be recalled:

$$R = R_{\text{x}}(\varphi) R_{\text{y}}(\vartheta) R_{\text{x}}(-\psi).$$

Then, expanding the above definition:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi \\ 0 & \sin\varphi & \cos\varphi \end{bmatrix} \begin{bmatrix} \cos\vartheta & 0 & \sin\vartheta \\ 0 & 1 & 0 \\ \sin\vartheta & 0 & \cos\vartheta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix}$$

$$= \begin{bmatrix} \cos\vartheta & -\sin\vartheta\sin\psi & \sin\vartheta\cos\psi \\ \sin\varphi\sin\vartheta & \cos\varphi\cos\psi + \sin\varphi\cos\vartheta\sin\psi & \cos\varphi\sin\psi - \sin\varphi\cos\vartheta\cos\psi \\ -\sin\vartheta\cos\varphi & \sin\varphi\cos\psi - \cos\varphi\cos\vartheta\sin\psi & \sin\varphi\sin\psi + \cos\varphi\cos\vartheta\cos\psi \end{bmatrix}.$$

To obtain the relation between the above expressions and the quaternion formulation, the derivations of [12] can be used to state:

$$\begin{array}{rcccl}
R_{11} & = & \cos\vartheta & = & q_0^2+q_1^2-q_2^2-q_3^2,\\
R_{21} & = & \sin\varphi\sin\vartheta & = & 2(q_1q_2+q_0q_3),\\
R_{31} & = & -\sin\vartheta\cos\varphi & = & 2(q_1q_3-q_0q_2),\\
R_{12} & = & -\sin\vartheta\sin\psi & = & 2(q_1q_2-q_0q_3),\\
R_{22} & = & \cos\varphi\cos\psi+\sin\varphi\cos\vartheta\sin\psi = & = & q_0^2-q_1^2+q_2^2-q_3^2,\\
R_{32} & = & \sin\varphi\cos\psi-\cos\varphi\cos\vartheta\sin\psi & = & 2(q_2q_3+q_0q_1),\\
R_{13} & = & \sin\vartheta\cos\psi & = & 2(q_1q_3+q_0q_2),\\
R_{23} & = & \cos\varphi\sin\psi-\sin\varphi\cos\vartheta\cos\psi & = & 2(q_2q_3-q_0q_1),\\
R_{33} & = & \sin\varphi\sin\psi+\cos\varphi\cos\vartheta\cos\psi & = & q_0^2-q_1^2-q_2^2+q_3^2.
\end{array} \tag{A.3}$$

Moreover, by inverting the above relations, the Euler angle formulation can be derived as a function of the natural coordinates and quaternions:

$$\begin{array}{rcccl}
\vartheta & = & \arccos\left(R_{11}\right) & = & \arccos\left(q_0^2+q_1^2-q_2^2-q_3^2\right),\\
\varphi & = & \operatorname{arctan2}\left(R_{21},-R_{31}\right) & = & \operatorname{arctan2}\left(q_0q_3+q_1q_2,q_0q_2-q_1q_3\right),\\
\psi & = & \operatorname{arctan2}\left(-R_{12},R_{13}\right) & = & \operatorname{arctan2}\left(q_0q_3-q_1q_2,q_0q_2+q_1q_3\right).
\end{array} \tag{A.4}$$

Finally, to be consistent and self contained, the quaternion formulation can be derived as a function of the other two parameterization. In particular, consider first that:

$$R_{11}+R_{22}+R_{33}=3q_0^2-q_1^2-q_2^2-q_3^2.$$

By the quaternion invariant $q_0^2+q_1^2+q_2^2+q_3^2=1$, the above expression is equivalent to:

$$R_{11}+R_{22}+R_{33}+1=4q_0^2.$$

Therefore, an expression for q_0 is given by:

$$q_0=\frac{1}{2}\sqrt{R_{11}+R_{22}+R_{33}+1}. \tag{A.5}$$

As a result, the other three relations follow directly as:

$$\begin{aligned}
q_1&=\frac{R_{32}-R_{23}}{4q_0}\\
q_2&=\frac{R_{13}-R_{31}}{4q_0}\\
q_3&=\frac{R_{21}-R_{12}}{4q_0}
\end{aligned} \tag{A.6}$$

Finally, considering Equations A.5 and A.6 in couple with [12, Equation (38)] expressing the quaternion formulation in Euler angles, the full set of quaternion relations is given by:

$$\begin{array}{rcl}
q_0=\frac{1}{2}\sqrt{R_{11}+R_{22}+R_{33}+1} & = & \cos\frac{\varphi}{2}\cos\frac{\vartheta}{2}\cos\frac{\psi}{2}+\sin\frac{\varphi}{2}\cos\frac{\vartheta}{2}\sin\frac{\psi}{2}\\
q_1=\frac{R_{32}-R_{23}}{2\sqrt{R_{11}+R_{22}+R_{33}+1}} & = & \sin\frac{\varphi}{2}\cos\frac{\vartheta}{2}\cos\frac{\psi}{2}-\cos\frac{\varphi}{2}\cos\frac{\vartheta}{2}\sin\frac{\psi}{2}\\
q_2=\frac{R_{13}-R_{31}}{2\sqrt{R_{11}+R_{22}+R_{33}+1}} & = & -\sin\frac{\varphi}{2}\sin\frac{\vartheta}{2}\sin\frac{\psi}{2}+\cos\frac{\varphi}{2}\sin\frac{\vartheta}{2}\cos\frac{\psi}{2}\\
q_3=\frac{R_{21}-R_{12}}{2\sqrt{R_{11}+R_{22}+R_{33}+1}} & = & \sin\frac{\varphi}{2}\sin\frac{\vartheta}{2}\cos\frac{\psi}{2}+\cos\frac{\varphi}{2}\sin\frac{\vartheta}{2}\sin\frac{\psi}{2}
\end{array} \tag{A.7}$$

Appendix B

Wind Speed Profile Generation

As stated in Section 6.1, [50] describes a specific methodology to obtain a discrete time series $[v_{\mathrm{w},0}, \ldots, v_{\mathrm{w},m}]$ of wind speed values which follow a Kaimal power spectrum. The specific algorithm can be implemented as a series of different steps:

(i). In a first step, white noise using a uniform distribution $\mathcal{U}[0,1]$ is generated and then averaged so that discretization effects are removed; the result of this step is a times series of values $[x_0, \ldots, x_m]$, where each time value x_k is computed by the following equation:

$$x_k = \frac{\sum_{i=1}^{N} r_i - N/2}{0.29\sqrt{N\Delta t}},$$

where r_i represent the random values equally distributed in $[0,1]$, N=100 the number of averaged values and Δt the time step.

(ii). This data is then filtered using a Kaimal filter so that the time series has a Kaimal power spectrum. In order to understand the filter design, the Kaimal power spectrum definition by the standard IEC61400-1 [53] has to be considered:

$$\frac{fS_{\mathrm{k}}(f)}{\sigma_{\mathrm{w}}^2} = \frac{4fL_{\mathrm{t}}/v_{\mathrm{w}}^{(0)}}{(1+6fL_{\mathrm{t}}/v_{\mathrm{w}}^{(0)})^{\frac{5}{3}}},$$

where $v_{\mathrm{w}}^{(0)}$ represents the wind gusts average speed, L_{t} the turbulence length of the wind gusts and σ_{w} the standard deviation of the turbulences. Based on the above equation, the filter is then modeled in the frequency domain by the following continuous transfer function [50]:

$$H_{\mathrm{kaimal}}(s) = K\frac{0.0182c^2s^2 + 1.3653cs + 0.9846}{1.3463c^2s^2 + 3.7593cs + 1.0},$$

where $K = 2.0313\sqrt{L_{\mathrm{t}}\sigma_{\mathrm{w}}^2/v_{\mathrm{w}}^{(0)}}$ and $c = L_{\mathrm{t}}/v_{\mathrm{w}}^{(0)}$. Finally, the filter is transformed to the time domain (without the filter gain K) by using the following discrete IIR filter:

$$y_k = \frac{\mathrm{b}_0 x_k + \mathrm{b}_1 x_{k-1} + \mathrm{b}_2 x_{k-2} - \mathrm{a}_1 y_{k-1} - \mathrm{a}_2 y_{k-2}}{\mathrm{b}_0},$$

with:

$$\begin{aligned}
&\mathrm{b}_2 = \mathrm{B}_0\Delta t^2 - \mathrm{B}_1\Delta t + \mathrm{B}_2, && \mathrm{b}_1 = \mathrm{B}_1\Delta t - 2\mathrm{B}_2, && \mathrm{b}_0 = \mathrm{B}_2,\\
&\mathrm{B}_2 = 0.0182c^2, && \mathrm{B}_1 = 1.3653c, && \mathrm{B}_0 = 0.9846,\\
&\mathrm{a}_2 = \mathrm{A}_0\Delta t^2 - \mathrm{A}_1\Delta t + \mathrm{A}_2, && \mathrm{a}_1 = \mathrm{A}_1\Delta t - 2\mathrm{A}_2, && \mathrm{a}_0 = \mathrm{A}_2,\\
&\mathrm{A}_2 = 1.3463c^2, && \mathrm{A}_1 = 3.7593c, && \mathrm{B}_0 = 1.
\end{aligned}$$

(iii). The output data of the Kaimal filter is filtered again by the following two cascade low pass filters:

$$\begin{aligned}
y_k^{(\mathrm{LP1})} &= e^{-\omega\Delta t} y_{k-1}^{(\mathrm{LP1})} + (1 - e^{-\omega\Delta t}) y_k,\\
y_k^{(\mathrm{LP2})} &= e^{-\omega\Delta t} y_{k-1}^{(\mathrm{LP2})} + (1 - e^{-\omega\Delta t}) y_k^{(\mathrm{LP1})},
\end{aligned}$$

with $\omega = 3.6\,\mathrm{rad/sec}$.

(iv). Finally the factor K of the continuous filter is applied and the average wind speed summed to obtain the wind speed time series:

$$v_{\mathrm{w},k} = v_{\mathrm{w}}^{(0)} + y_k^{(\mathrm{LP2})}.$$

In order to generate wind speed values for the kite simulator, the algorithm was modeled with a turbulence length $L_{\mathrm{t}} = 100\,\mathrm{m}$ and a $\sigma_{\mathrm{w}} = 0.1 v_{\mathrm{w}}^{(0)}\,\mathrm{m/s}$; this selection leads to realistic wind gusts with turbulence's levels of roughly 10 %.

Bibliography

[1] EEA. En01 energy related greenhouse gas emissions. Technical report, European Environment Agency, 2008.

[2] Kahn Ribeiro, S. Kobayashi, M. Beuthe, J. Gasca, D. Greene, D. S. Lee, Y. Muromachi, P. J. Newton, S. Plotkin, D. Sperling, R. Wit, and P. J. Zhou. Transport and its infrastructure. in climate change 2007: Mitigation. Technical report, Contribution of Working Group III to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change [B. Metz, O.R. Davidson, P.R. Bosch, R. Dave, L.A. Meyer (eds)], Cambridge University Press, Cambridge, United Kingdom and New York, NY, USA., 2007.

[3] Cristina L. Archer and Mark Z. Jacobson. Evaluation of global wind power. *J. Geophys Res.*, 110(D12110), 2005.

[4] U. Ahrens, M. Diehl, and R. Schmehl, editors. *Airborne Wind Energy*. Springer-Verlag Berlin Heidelberg, 2013.

[5] M. Canale, L. Fagiano, and M. Milanese. High Altitude Wind Energy Generation Using Controlled Power Kites. *IEEE Transactons On Control Systems Technology*, 18:168 – 180, 2010.

[6] M.L. Loyd. Crosswind Kite Power. *Journal of Energy*, 4(3):106–111, July 1980.

[7] L. Fagiano, M. Milanese, and D. Piga. Optimization of Airborne Wind Energy Generators. *International Jounal of Robust and Nonlinear Control*, 2011.

[8] B. Houska and M. Diehl. Optimal control of towing kites. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 2693–2697, San Diego, USA, 2006.

[9] Makani Power. Makani Power Website. `http://www.google.com/makani/`.

[10] Skysails. Skysails Power Website. `http://www.skysails.info/english/power/`.

[11] Enerkite. Enerkite - Airborne Wind Energy. `http://www.enerkite.de/en/`.

[12] M. Erhard, G. Horn, and M. Diehl. A quaternion-based model for optimal control of the SkySails airborne wind energy system. *Angewandte Mathematik und Mechanik*, 2015.

[13] Michael Erhard and Hans Strauch. Control of towing kites for seagoing vessels. arXiv http://arxiv.org/abs/1202.3641.

[14] Michael Erhard and Hans Strauch. Flight control of tethered kites in autonomous pumping cycles for airborne wind energy. *Control Engin*, (40):13–26, 2015.

[15] M. Erhard and H. Strauch. Theory and experimental validation of a simple comprehensible model of tethered kite dynamics used for controller design. In *Airborne Wind Energy*, chapter 8, pages 141–165. Springer, 2013.

[16] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, 2 edition, 2006.

[17] H. G. Bock and K. J. Plitt. A multiple shooting algorithm for direct solution of optimal control problems. In *Proceedings of the IFAC World Congress*, pages 242–247. Pergamon Press, 1984.

[18] E.F. Camacho and C. Bordons. *Model Predictive Control*. Springer, 2nd edition, 2007.

[19] J. Sternberg, S. Gros, B. Houska, and M. Diehl. Approximate robust optimal control of periodic systems with invariants and high-index differential algebraic systems. In *Proceedings of the 7th IFAC Symposium on Robust Control Design*, pages 678–683, 2012.

[20] J. C. Butcher. *Numerical Methods for Ordinary Differential Equations*. 2003.

[21] J. Andersson. *A General-Purpose Software Framework for Dynamic Optimization*. PhD thesis, K.U. Leuven, October 2013.

[22] J. Andersson, J. Åkesson, and M. Diehl. CasADi – a symbolic package for automatic differentiation and optimal control. In S. Forth, P. Hovland, E. Phipps, J. Utke, and A. Walther, editors, *Recent Advances in Algorithmic Differentiation*, Lecture Notes in Computational Science and Engineering, pages 297–307, Berlin, 2012. Springer.

[23] Andreas Wächter and Lorenz T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[24] S. Gros and M. Diehl. Numerical optimal control with differential algebraic equations - lecture notes. 2016.

[25] U. Ascher. Stabilization of invariants of discretized differential systems. *Numerical Algorithms*, 14, 1997.

[26] S. Gros and M. Diehl. Modeling of airborne wind energy systems in natural coordinates. In *Airborne Wind Energy*. Springer-Verlag Berlin Heidelberg, 2013.

[27] G. Horn, S. Gros, and M. Diehl. Numerical trajectory optimization for airborne wind energy systems described by high fidelity aircraft models. In *Airborne Wind Energy*. Springer-Verlag Berlin Heidelberg, 2013.

[28] Moritz Diehl and Sebastien Gros. *Numerical Optimal Control*. Cambridge University Press, expected to be published in 2017.

[29] J.B. Rawlings and D.Q. Mayne. *Model Predictive Control: Theory and Design*. Nob Hill, 2009.

[30] D. Q. Mayne, J. B. Rawlings, C. V. Rao, and P. O. M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica*, 26(6):789–814, 2000.

[31] H. G. Bock, M. Diehl, D. B. Leineweber, and J.P. Schlöder. Efficient direct multiple shooting in nonlinear model predictive control. In F. Keil, W. Mackens, H. Voß, and J. Werther, editors, *Scientific Computing in Chemical Engineering II*, volume 2, pages 218–227. Springer, 1999.

[32] Moritz Diehl, Lalo Magni, and Giuseppe De Nicolao. Efficient NMPC of unstable periodic systems using approximate infinite horizon closed loop costing. *Annual Reviews in Control*, 28(1):37 – 45, 2004.

[33] Toshiyuki Ohtsuka. A continuation/GMRES method for fast computation of nonlinear receding horizon control. *Automatica*, 40(4):563–574, 2004.

[34] V. M. Zavala and L.T. Biegler. The Advanced Step NMPC Controller: Optimality, Stability and Robustness. *Automatica*, 45:86–93, 2009.

[35] J. Guddat, F. Guerra Vasquez, and H.T. Jongen. *Parametric Optimization: Singularities, Pathfollowing and Jumps*. Teubner, Stuttgart, 1990.

[36] M. Diehl. *Real-Time Optimization for Large Scale Nonlinear Processes*, volume 920 of *Fortschritt-Berichte VDI Reihe 8, Meß-, Steuerungs- und Regelungstechnik*. VDI Verlag, Düsseldorf, 2002. PhD Thesis.

[37] M. Diehl, H. G. Bock, J.P. Schlöder, R. Findeisen, Z. Nagy, and F. Allgöwer. Real-time optimization and nonlinear model predictive control of processes governed by differential-algebraic equations. *Journal of Process Control*, 12(4):577–585, 2002.

[38] M. Diehl, H. G. Bock, and J. P. Schlöder. A real-time iteration scheme for nonlinear optimization in optimal feedback control. *SIAM Journal on Control and Optimization*, 43(5):1714–1736, 2005.

[39] M. Diehl, R. Findeisen, F. Allgöwer, H. G. Bock, and J. P. Schlöder. Nominal stability of the real-time iteration scheme for nonlinear model predictive control. *IEE Proc.-Control Theory Appl.*, 152(3):296–308, 2005.

[40] M. Diehl, R. Findeisen, and F. Allgöwer. A stabilizing real-time implementation of nonlinear model predictive control. In L. Biegler, O. Ghattas, M. Heinkenschloss, D. Keyes, and B. van Bloemen Waanders, editors, *Real-Time and Online PDE-Constrained Optimization*, pages 23–52. SIAM, 2007.

[41] H. J. Ferreau, H. G. Bock, and M. Diehl. An online active set strategy to overcome the limitations of explicit MPC. *International Journal of Robust and Nonlinear Control*, 18(8):816–830, 2008.

[42] H.J. Ferreau. *qpOASES User's Manual*, 2007–2011. http://www.qpOASES.org/.

[43] B. Houska, H. J. Ferreau, and M. Diehl. ACADO toolkit – an open source framework for automatic control and dynamic optimization. *Optimal Control Applications and Methods*, 32(3):298–312, 2011.

[44] R. Quirynen, S. Gros, and M. Diehl. Fast auto generated ACADO integrators and application to MHE with multi-rate measurements. In *Proceedings of the European Control Conference (ECC)*, pages 3077–3082, 2013.

[45] C. G. Broyden. Quasi-Newton methods and their application to function minimization. *Maths. Comp.*, 21:368–381, 1967.

[46] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[47] Moritz Diehl. *Modelling and System Identification Lecture Script*. University of Freiburg, 2015.

[48] Rodney D. Driver. *Ordinary and Delay Differential Equations*. Springer, 1977.

[49] William E. Schiesser. *The Numerical Method of Lines: Integration of Partial Differential Equations*. Academic Press, 1991.

[50] C. Gavriluta, S. Spataru, I. Mosincat, C. Citro, I. Candela, and P. Rodriguez. Complete methodology on generating realistic wind speed profiles based on measurements. 2012.

[51] L. Fagiano, A. Zgraggen, M. Morari, and M. Khammash. Automatic crosswind flight of tethered wings for airborne wind energy: Modeling, control design, and experimental results. *Control Systems Technology*, 2013.

[52] C. Jehle. Automatic flight control of tethered kites for power generation. Master's thesis, Technical Univeristy of Munich, 2012.

[53] International Electrotechnical Commission. IEC 61400-1: Wind turbines design requirements, 2005.