macrφ@ufmg

Universidade Federal de Minas Gerais
Programa de Pós-Graduação em Engenharia Elétrica
Research group MACRO - Mechatronics, Control and Robotics

FIXED-WING UAV MOTION PLANNING AND OPTIMAL CONTROL FOR CURVE TRACKING

Leonardo Anício Alves Pereira

Belo Horizonte, Brazil
2021

Leonardo Anício Alves Pereira

FIXED-WING UAV MOTION PLANNING AND OPTIMAL CONTROL FOR CURVE TRACKING

Dissertation submitted to the Graduate Program in Electrical Engineering of Escola de Engenharia at the Universidade Federal de Minas Gerais, in partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

Advisors: Luciano Cunha de Araújo Pimenta
Guilherme Vianna Raffo

Belo Horizonte, Brazil
2021

Pereira, Leonardo Anício Alves.
P436f
Fixed-wing UAV motion planning and optimal control for curve tracking [recurso eletrônico] / Leonardo Anício Alves Pereira. - 2021.
1 recurso online (xxv, 84 f. : il., color.) : pdf.

Orientador: Luciano Cunha de Araújo Pimenta.
Coorientador: Guilherme Vianna Raffo.

Dissertação (mestrado) - Universidade Federal de Minas Gerais, Escola de Engenharia.

Apêndices: f. 79-84.

Bibliografia: f. 73-78.
Exigências do sistema: Adobe Acrobat Reader.

1. Engenharia elétrica - Teses. 2. Aeronave não tripulada - Teses. 3. Controle preditivo - Teses. 4. Campos vetoriais - Teses. I. Pimenta, Luciano Cunha de Araújo. II. Raffo, Guilherme Vianna. III. Universidade Federal de Minas Gerais. Escola de Engenharia. IV. Título.
CDU: 621.3(043)

Ficha catalográfica: Biblioteca Profº Mário Werneck, Escola de Engenharia da UFMG.

"Fixed-Wing UAV Motion Planning and Optimal Control for Curve Tracking"

Leonardo Anício Alves Pereira

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 26 de fevereiro de 2021.

Por:

Prof. Dr. Luciano Cunha de Araújo Pimenta
DELT (UFMG) - Orientador

Prof. Dr. Guilherme Vianna Raffo
DELT (UFMG) - Coorientador

Vinicius Mariano

Prof. Dr. Vinícius Mariano Gonçalves
DEE (UFMG)

Prof. Dr. Armando Alves Neto
DELT (UFMG)

Rubens J. M. Afonso

Prof. Dr. Rubens Junqueira Magalhães Afonso
Divisão de Engenharia Eletrônica (ITA)

To my family, friends, and
professors, who brought me this far.

Acknowledgements

First, I would like to acknowledge my family, in special to my parents, Alessandro and Aparecida, and my girlfriend, Mayra, for their support and encouragement during the master's course. They are the ones who made this possible.

I am very grateful to my mentors Luciano Pimenta and Guilherme Raffo for their guidance and for sharing their knowledge with me in the last two years. Besides, I should thank the members of the examination committee, the professors Vinícius Gonçalves, Armando Neto, and Rubens Afonso for their valuable suggestions to improve this dissertation.

I thank the colleagues and employees of UFMG and of the School of Engineering. Thank the members of the CORO laboratory for the friendship built.

Furthermore, I acknowledge the other professors of the program for all the knowledge transmitted to me.

Finally, I acknowledge the financial support provided by CAPES and PPGEE.

> "Discontent is the first step in the progress of a man or a nation."
>
> *Oscar Wilde*

Resumo

À medida que o uso de veículos aéreos não tripulados (VANTs) vem aumentando, novas técnicas de planejamento de movimento, navegação e controle são desenvolvidas. Aplicações militares e civis geralmente requerem que um VANT seja capaz de estimar sua própria pose, processar as informações fornecidas pelo ambiente e seguir uma determinada trajetória de forma autônoma. Além disso, algumas tarefas como vigilância, mapeamento de terreno e proteção de comboio exigem uma longa vida útil em termos de consumo de energia. Nestas situações, o uso de um VANT de asa-fixa é altamente recomendado devido à sua maior autonomia quando comparado aos VANTs de asa rotativa. Este trabalho apresenta uma solução para o problema de guiar e controlar um VANT de asa-fixa para seguir uma curva fechada enquanto desvia de obstáculos dinâmicos.

A estratégia proposta pode ser dividida em duas partes. Em uma camada superior é utilizada uma estratégia de campos vetoriais que alterna entre duas formas: um campo vetorial para convergir e circular a curva alvo, e um para desviar dos obstáculos no caminho do VANT. Para a camada inferior é proposto um controle de linearização por realimentação, onde a lei de controle auxiliar é projetada através de um MPC (*Model Predictive Control*) linear para fazer com o que o VANT siga as referências fornecidas pelos campos vetoriais.

Simulações utilizando Matlab e o modelo completo do VANT, com 6 graus de liberdade e 12 estados, demonstram a eficiência da estratégia proposta para diferentes cenários. Resultados obtidos utilizando um sistema computacional embarcado demonstram que a estratégia proposta é factível de implementação em uma plataforma física.

Abstract

As the use of unmanned aerial vehicles (UAVs) is increasing, new techniques for motion planning, navigation and control are being developed. Both military and civilian applications usually require a UAV to be able to estimate its own pose, process the information provided by the environment, and follow a given trajectory autonomously. Besides, some tasks such as surveillance, terrain mapping and convoy protection require long endurance. For those tasks, the use of a fixed-wing UAV is highly recommended due to its greater endurance when compared to rotary-wing UAVs. This work presents a strategy for solving the problem of guiding and controlling a UAV to follow a closed curve while avoiding dynamic obstacles.

The proposed strategy can be divided into two parts. In a top layer, a vector field strategy is used which alternates between two forms: a vector field to converge to and circulate the target curve, and one to avoid obstacles along the UAV path. For a lower layer, a feedback linearization controller is proposed, in which a linear Model Predictive Control (MPC) is used as the auxiliary control law to make the UAV follow the references provided by the vector fields.

Simulations using Matlab and the entire UAV model, with 6 degrees of freedom and 12 states, demonstrate the efficiency of the proposed strategy for different scenarios. Results obtained using an embedded computational system demonstrate that the proposed strategy is feasible to be implemented on a physical platform.

Contents

List of Figures

List of Tables

Acronyms

EKF	Extended Kalman Filter
DOF	Degrees Of Freedom
FL	Feedback Linearization
GPS	Global Positioning System
GT	Ground Truth
IMU	Inertial Measurement Unit
KF	Kalman Filter
LQR	Linear Quadratic Regulator
LQE	Linear Quadratic Estimator
MPC	Model Predictive Control
NMPC	Nonlinear Model Predictive Control
NED	North-East-Down
ORCA	Optimal Reciprocal Collision Avoidance
PID	Proportional Integral Derivative
PRM	Probabilistic Roadmap
RRT	Rapidly-Exploring Random Tree
RHC	Receeding Horizon Control
SLPRM	Semi-lazy Probabilistic Roadmap
UAV	Unmanned Aerial Vehicle
VFRRT	Vector Field Rapidly-Exploring Random Tree

Notation

Common notation

a	Italic lower case letters denote scalars
$\boldsymbol{a}$	Boldface italic lower case letters denote vectors
$\boldsymbol{A}$	Boldface italic upper case letters denote matrices
$\boldsymbol{A}^T$	Transposed matrix
$\boldsymbol{A} \succeq 0$	Positive semi-definite matrix
t	Time
Δt	Discrete time step
$\mathbb{R}$	Set of real numbers

Model notation

g	Gravitational acceleration
m	UAV mass
$\boldsymbol{J}$	Inertia tensor
p_n	UAV North position in the NED frame
p_e	UAV East position in the NED frame
p_d	UAV Down position in the NED frame
u_r	UAV body relative velocity in the i_{body} axis
v_r	UAV body relative velocity in the j_{body} axis
w_r	UAV body relative velocity in the k_{body} axis
ϕ	Roll angle

θ	Pitch angle
ψ	Yaw angle
p	UAV angular velocity in the i_{body} axis
q	UAV angular velocity in the j_{body} axis
r	UAV angular velocity in the k_{body} axis
α	Angle-of-attack
β	Sideslip angle
v_t	UAV true speed
$\boldsymbol{\mathsf{w}}$	Wind velocity vector
$\boldsymbol{\mathsf{w}}_s$	Steady wind velocity vector
$\boldsymbol{\mathsf{w}}_g$	Gust wind velocity vector
w_n	North wind velocity
w_e	East wind velocity
w_d	Down wind velocity
δ_a	Aileron deflection
δ_e	Elevator deflection
δ_r	Rudder deflection
δ_t	Normalized throttle
δ_{el}	Left elevon deflection
δ_{er}	Right elevon deflection

Control notation

ρ	Relative degree
$\boldsymbol{\xi}$	External dynamics vector
$\boldsymbol{\eta}$	Internal dynamics vector
$\boldsymbol{\Lambda}$	Feedback Linearization decoupling matrix

$\boldsymbol{\lambda}$	Feedback Linearization vector
J_0	Cost functional
J_0^*	Optimal cost
$\boldsymbol{Q}$	State weighting matrix
$\boldsymbol{R}$	Control weighting matrix
$\boldsymbol{P}$	Terminal weighting matrix
u^*	Optimal solution
n_u	MPC control horizon
n_p	MPC prediction horizon
$\boldsymbol{A}_{eq}$	Equality matrix
$\boldsymbol{b}_{eq}$	Equality vector
$\boldsymbol{A}_{ineq}$	Inequality matrix
$\boldsymbol{b}_{ineq}$	Inequality vector
$\boldsymbol{M}_{\mathbf{u}}$	Control mapping matrix

Vector field notation

$\boldsymbol{x}$	Agent position vector
s	Circulation direction
i	Index of the closest obstacle
d_{in}	Perception range
γ	Time in transition
$\mathcal{O}$	Set of obstacles
$\mathcal{C}$	Target curve
$\mathcal{C}_i$	Target circle around the i-th obstacle
$F_c(\boldsymbol{x}, t)$	Vector field to the curve
$F_o(\boldsymbol{x}, t, s, i)$	Vector field to circulate the i-th obstacle
$F(\boldsymbol{x}, t, s, i)$	Total vector field

1 Introduction

1.1 Motivation

The field of robotics has experienced a substantial growth in the last decades. In particular, Unmanned Aerial Vehicles (UAVs) have been used in many civilian and military applications, as they are becoming cheaper and more practical than their manned counterpart. For instance, airline companies have been using UAVs for aircraft maintenance and visual inspections (Novák et al., 2020). Moreover, recent works have shown the effectiveness of UAVs in forest fire detection and monitoring (Yuan et al., 2015; Sherstjuk et al., 2018), search and rescue (Alotaibi et al., 2019), and many others applications (Shakhatreh et al., 2019).

Two of the main types of commercial UAVs are fixed-wings and rotary-wings, and both have their advantages according to their missions. For instance, fixed-wing UAVs have a few benefits over rotary-wings: *i*) they can fly faster and at higher altitudes; *ii*) they can fly over longer periods of time; *iii*) they are safer in the event of motor failure; *iv*) they are usually more stable in high winds. Due to these advantages, they have been extensively applied in surveillance, terrain mapping, and protection tasks. In contrast, rotary-wings provide more maneuverability, are easier to fly, and are usually cheaper. Practical applications of rotary-wings include 3D structure mapping, side inspections in buildings, and aerial photography. Figure 1.1 shows an example of a rotary-wing UAV, and Figure 1.2 of a fixed-wing one.

Figure 1.1: DJI Mavic Pro (*Source:* https://www.dji.com/products)

Figure 1.2: SIG Rascal 110 (*Source:* https://sigmfg.com/products).

1.2 Problem definition

Despite a large number of applications and research works, motion planning and control design are recurrent problems when dealing with UAVs. Motion planning has become one of the main research subjects as the field of robotics has been expanded beyond repetitive industrial applications. For applications with more than one goal (multi-goal applications) such as space exploration, medical surgery, and aerial photography, it may become impractical to develop a specific strategy for each new goal. Therefore, it is necessary to develop unifying algorithms that enable robots to automatically take decisions on how to achieve their goals.

In turn, robot control is about finding suitable control laws, or control sequences, to make the robot execute the decisions computed during the motion planning phase. The control of fixed-wing UAVs brings great challenges due to the nonlinear behavior of the system and mechanical constraints. For instance, such UAV cannot fly at speeds below the stall speed; moreover, in order to generate lift, the UAV must fly within a limited angle-of-attack; and control surfaces have limited deflection.

Therefore, the main problem to be addressed in this dissertation is summarized as: find a unifying strategy, to solve the problem of guiding and controlling a fixed-wing UAV to converge to and circulate a closed curve, in an environment with previously unknown obstacles.

1.3 Related works

A wide variety of works dealing with UAV control design and motion planning can be found in the literature. In this section, we intend to provide a review of both classical and state-of-the-art strategies. It should be mentioned that we focus on, but are not restricted to, strategies suited for fixed-wing UAVs.

1.3.1 Task and motion planning

Over the past years, researchers have been proposing multiple solutions for a broad class of motion planning problems. Some key issues when dealing with motion planning are obstacle avoidance, kinodynamic planning, optimality, and uncertainty. Since the robot motion field is broad, in this review we focus on vector fields, sampling-based algorithms, and Dubins paths, which are well-known, suited strategies for fixed-wing UAVs. Moreover, we recommend the reading of Yang et al. (2014) for a survey on path planning algorithms for fixed-wing UAVs.

Sampling-based algorithms

Sampling-based algorithms sample a set of nodes in the configuration space and aim to connect those points until a path is obtained. An important characteristic of these algorithms is that they might achieve probabilistic completeness, in the case of random uniform sampling, or resolution completeness, if using grid sampling. Thus, in these cases, if a solution path exists, the planner will eventually find it.

Classical algorithms such as the Probabilistic Roadmap (PRM) are well-defined and extensively used in literature. The basic PRM algorithm, described in Choset et al. (2005), divides planning into two tasks: learning and query. The first phase captures the connectivity of the free configuration space, while the second one tries to connect user-defined configurations. In Bohlin & Kavraki (2000) the authors present a lazy approach to a PRM planner. Different from the basic algorithm, the proposed strategy initially assumes that all nodes and edges in the roadmap are collision-free, removing them as they are checked, thus, enabling the algorithm to run in less time.

In Akbaripour & Masehian (2017) the authors propose a semi-lazy PRM (SLPRM) for motion planning of an industrial manipulator. The strategy differs from the previous ones by only performing collision checking of random nodes for n links of the manipulator; therefore, it can be seen as a tradeoff between the basic PRM and the lazy PRM. The results show that, because the SLPRM performs fewer collision checks, it is more time-efficient than the basic PRM. Also, since it provides a better initial roadmap, the query time is also reduced when compared with the lazy PRM.

As an alternative to the PRM algorithm, LaValle (1998) proposes the Rapidly-Exploring Random Trees (RRTs), a randomized data structure suited for a broad class of motion planning problems, which can be directly applied to nonholonomic and kinodynamic planning. Besides that, RRTs have many other advantages: *i*) the expansion can be biased towards desired regions, for instance, borders and unexplored portions of space; *ii*) the RRT algorithm is relatively simple; *iii*) it can be adapted to generate optimal paths, as in the RRT* algorithm (Karaman & Frazzoli, 2011). Figure 1.3 shows an example of the RRT algorithm applied to a cluttered environment, the obstacles are represented by the gray polygons, the target by the green dot, and the starting point by the black dot.

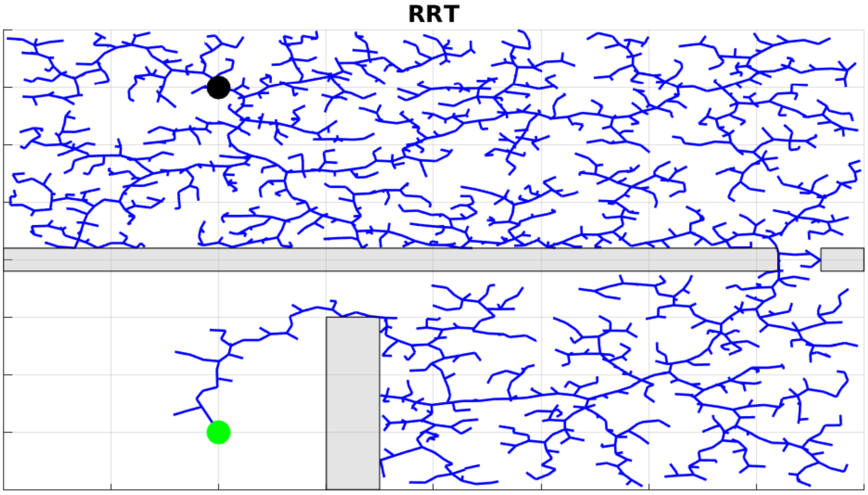

Figure 1.3: RRT in a cluttered environment.

Due to their advantages, RRTs are widely used in fixed-wing UAVs motion planning. In Lu et al. (2016), the authors propose a dynamic RRT algorithm to calculate a collision-free trajectory in an environment with dynamic obstacles. The proposed strategy uses B-splines to smooth the trajectories and adopts constraints on turning angle, diving angle, flight distance, and route leg in order to provide feasible, optimal paths.

Spline-based RRTs are also proposed in Lee & Shim (2016) and Zhang et al. (2020). The former uses the optimal bidirectional RRT* algorithm. The proposed strategy is able to respect aerodynamic characteristics such as thrust-to-weight ratio, load factor, and direction constraints. The latter proposes a real-time RRT to calculate collision-free paths in the presence of high-speed dynamic obstacles, with a growth strategy that uses the optimal reciprocal collision avoidance algorithm (ORCA), which under specific conditions ensures that the obstacle and the aircraft will never collide.

Furthermore, Ge et al. (2020) combine a kinodynamic motion planning RRT* algorithm based on the linear state space model of a fixed-wing UAV obtained by applying the feedback linearization technique. The computed trajectory is post-processed using the Gauss-Kruger projection method to obtain waypoints that can be tracked by an autopilot.

Differently from the previous strategies, Zogopoulos-Papaliakos & Kyriakopoulos (2020) propose a sampling-based algorithm for determining the Trim Flight Envelope for a fixed-wing UAV. The strategy ensures that for each point inside the envelope, there exists a control input that keeps the UAV inside of it. Also, since the flight envelope is constructed using convex intersections of half-spaces, it can be incorporated as convex constraints in MPC formulations.

Vector fields

Vector field based strategies have been extensively used in UAV guidance. One of the main advantages of vector fields oversampling algorithms relies on the fact that it provides implicit trajectories, instead of a path to be followed. In addition, it integrates the solutions to three problems in one approach (Rimon & Koditschek, 1992): path planning, trajectory planning, and control. Figure 1.4 shows an example of a vector field, in blue, to allow convergence and circulation of a desired curve, in red.

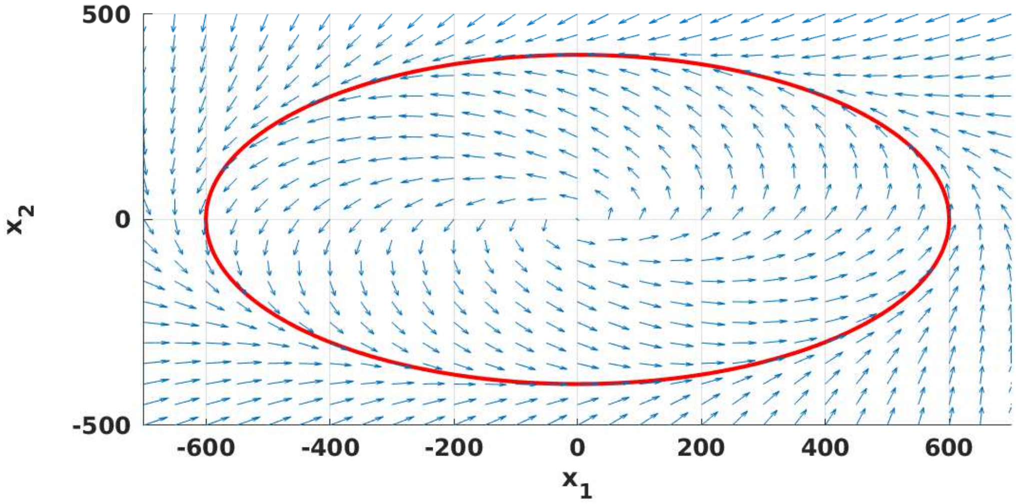

Figure 1.4: Vector field, in blue, and target curve in red.

In Gonçalves et al. (2010), the authors propose a methodology for the computation of artificial vector fields in n-dimensions, using $n-1$ implicit functions whose intersection define the desired curve. The proposed vector field is based on the sum of three components: i) a convergence term, which guarantees that a controlled robot approaches the desired curve; ii) a tangent term, which makes the robot circulate the desired curve; iii) a feedforward term to compensate for the time-varying nature of the curve. The main advantages of the proposed strategy are related to the implicit-function formulation and the inclusion of the correction terms.

An obstacle avoidance strategy, based on vector fields, is proposed in Yao et al. (2019). In that work, the authors propose a static composite vector field, which uses bump curves to deviate from static obstacles in real-time. The use of bump functions significantly reduces the overlapping effect caused by the combination of two vector fields, besides guaranteeing the existence and uniqueness of solutions. Different from them, this work considers moving obstacles and an online switching strategy.

An integration of vector fields and sampling based techniques is used in Jahn & de Araújo Pimenta (2016) to find a feasible path in a 3D workspace with static obstacles for a Dubins airplane model. Simulation results show that the proposed strategy outperforms pure sampling based methods. Similarly, in Pereira et al. (2016) the authors propose an algorithm that integrates high level vector fields, responsible for convergence to a target path, and an RRT* as a local planner, responsible for avoiding obstacles and no-fly zones.

Dubins paths

Dubins paths were originally proposed in Dubins (1957). The author uses a car-like robot to show that any initial and final configuration can be connected using three path segments, called motion primitives. These motion primitives are represented by left turns (L), right turns (R), and straight lines (S). Thus, there are six possible combinations

$$\begin{pmatrix} LRL & LSR & LSL & RLR & RSR & RSL \end{pmatrix}.$$

One of the main advantages of dubins path is the possibility of including minimum turning radius, and climb (or dive) angle constraints in the formulation. One approach used when computing dubins paths for fixed-wing UAVs is to decompose the problem into two 2D paths, one for the horizontal plane, and one for the vertical plane. This approach is used in Váňa et al. (2020), where the authors develop a strategy for solving the problem of finding cost-efficient three-dimensional paths that satisfy the maximum allowed curvature and the pitch angle of the vehicle. The proposed strategy showed to provide paths with fewer turns than others, making them easier to be followed by the controller.

Alternatively, dubins paths can also be adapted to fixed-wing UAVs by assuming fixed altitude, as presented in Lugo-Cárdenas et al. (2014), or by using a modified vehicle model for 3D workspaces, called Dubins airplane. The latter was used in McLain et al. (2014), where the authors include rate-of-climb, airspeed, flight-path angle, and bank angle equations in order to make the model more consistent with fixed-wing UAVs. Also, a strategy to solve the problem of infinite solutions, when the altitude falls under specific conditions, is provided.

Furthermore, Song & Hu (2017) propose an integration of Dubins paths and A^* algorithm, (Hart et al., 1968), to generate a safe, flyable path. The strategy consists of implementing the A^* algorithm in a weighted graph, which is constructed using the Dubins paths. Finally, Ismail et al. (2018) show that for many search missions, the original Dubins curve can be reduced to a curve with only two components: straight lines and left or right turns. The authors provide an approach for calculating those components using elementary geometry.

1.3.2 Fixed-Wing Control

Fixed-wing UAVs impose big challenges on control design. Their nonlinear behavior, mechanical constraints, and coupling between lateral and longitudinal dynamics are problems to be dealt with when designing a controller. In the next sections we discuss, some of the most used strategies in the literature.

Loop closure

A widely used control design tool for fixed-wing UAVs is the successive loop closure. In Beard & McLain (2012), the authors present detailed guidelines on how to design and implement autopilots using this strategy. The basic idea is to modify the open loop dynamics of the plant by closing several simple feedback loops successively. This procedure can be seen in Figures 1.5 and 1.6, where the open-loop dynamics are given by $P(s) = P_1(s)P_2(s)...P_n(s)$. The compensators $C_1(s), C_2(s), ..., C_n(s)$ are designed in a way that each closed-loop transfer function can be modeled as a unit gain.

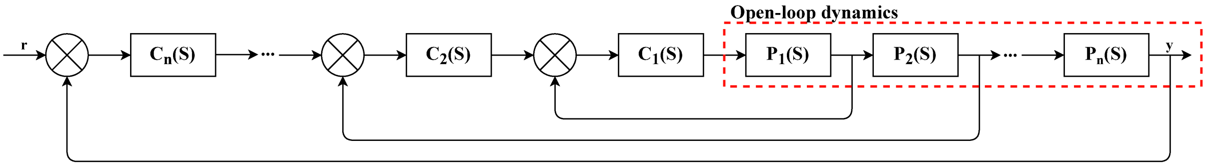

Figure 1.5: Closed loops.

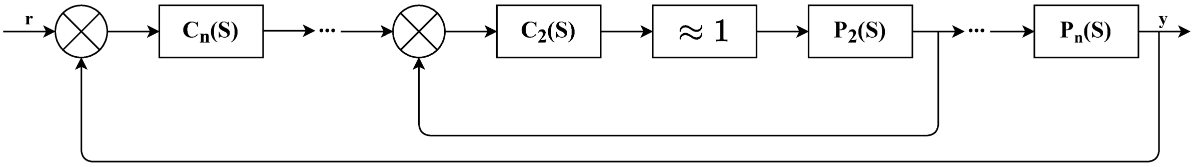

Figure 1.6: Inner loop as unit gain.

A necessary condition for this design is that the innermost loop has the highest bandwidth, with each successive loop bandwidth a factor of 5 or 10 times smaller (Beard & McLain, 2012). Also, the design of the innermost loop should consider the physical limits of the UAV control surfaces.

Despite being widely used, the successive loop closure strategy has the main drawback of becoming increasingly difficult to design as more loops are added. Also, it does not guarantee success when the dynamics are multivariable (Stevens et al., 2015).

Therefore, this strategy is mostly used to develop simplified guidance strategies, which can be used to design higher-level control laws. Several kinematic and dynamic guidance models can be found on Beard & McLain (2012).

Guidance laws

Guidance control laws usually make use of a simplified kinematic or dynamic model of the UAV to provide references for a lower-level autopilot. In Andersen & Kristiansen (2017), the authors consider a simplified model, which uses the desired angular velocities as control inputs, to present a strategy based on two quaternions, one which makes the

UAV fly towards the path and one that makes the UAV follow the path. The strategy takes advantage of the fact that summing unit-quaternions representing rotation yields averaged or blended rotations.

In Jesus et al. (2013), a reference model, which uses the desired speed, altitude, and heading angle as inputs, is applied in an inner-outer loop design to coordinate a team of n UAVs. The outer loop consists of a three-stage distributed controller responsible for making the UAVs converge to a final configuration, in which they are evenly distributed along a circle, performing uniform circular motion.

The same reference model is also used in Olavo et al. (2018) and Rezende et al. (2018). The former deals with the problem of circulating a fixed target on the ground using a robust guidance strategy with global stability. The latter develops a guidance vector-field strategy to converge to and circulate a desired, generic curve in $\mathbb{R}^3$. The proposed strategy considers model uncertainties and field singularities. The authors show that asymptotic stability is guaranteed for a bounded region.

Furthermore, Beard et al. (2014) propose a guidance law for tracking straight lines and circular orbits. The control strategy makes use of the theory of nested saturations to satisfy roll and flight path angle constraints. Specific conditions on wind speed, where the guidance strategy ensures asymptotic tracking, are derived. The authors provide conditions for when a simple switching strategy can be used to guarantee global asymptotic convergence to the orbit.

Backstepping

The main idea behind backstepping controllers can be defined as Sepulchre et al. (2012): Apply a passivation design to a small part of the system, and then reapply it step-by-step by augmenting the subsystem at each step. One of the advantages of backstepping is that it may reduce the problem complexity, since the design is split into steps. Also, its ability to deal with nonlinearities makes it a suitable strategy for UAV control. In Sartori et al. (2013), a backstepping-based autopilot is applied to stabilize the fast attitude inner-loop, while PID controls are responsible for the navigation loop. The proposed strategy ensures simultaneous control of both longitudinal and latero-directional planes.

In Espinoza et al. (2014), the authors provide an extensive comparative analysis of five different designs based on backstepping and sliding mode in order to compare which has the best performance regarding altitude, yaw, and roll angles control. The results show that by using backstepping with high order sliding mode it is possible to remove the well-known chattering of the sliding mode and to obtain smoother control inputs as compared to the other controllers.

Furthermore, Jung & Tsiotras (2008) adopt a simplified kinematic model of the UAV and apply a backstepping controller to compute roll angle commands given a desired heading rate. A parameter adaptation technique is used to increase the robustness of the system against inaccurate time constants of the roll loop. The proposed control offers the advantage of following parametric references with less rigorous initial conditions. Similarly, Ren & Atkins (2005) use a reference model with desired speed, roll angle, and altitude as command inputs to employ backstepping techniques, obtaining separated control laws for each input.

Model Predictive Control

Due to its advantages, Model Predictive Control (MPC) has become an attractive method for fixed-wing UAV control. It has a straightforward formulation, with explicit handling of states and control constraints and well understood tuning parameters. Also, it can be applied to both linear and nonlinear systems. However, its computational burden poses a big challenge.

A common strategy to overcome this drawback is to simplify the model before applying the MPC, as seen in Oettershagen et al. (2014). In that work, the authors decouple the UAV lateral and longitudinal dynamics and design a linear MPC to control the attitude of the UAV, along with an $\mathcal{L}_1$-navigation logic that provides attitude reference commands. The designed controller ensures accurate steering while respecting state and input constraints.

Tube-based strategies can also provide more efficient implementations, as some of the calculations are made offline. In Mammarella & Capello (2018), a robust tube-based MPC is proposed as an autopilot system to control both longitudinal and lateral dynamics of a UAV. The control law focuses on computational efficiency and robustness to both parametric uncertainties and bounded noises.

In contrast, using a Nonlinear MPC (NMPC) can make the formulation more intuitive, as the constraints and cost functional are more flexible. In Kang & Hedrick (2009), the authors use a simplified 2D reference model to design a higher-level tracking controller. A stability analysis for the initial conditions that assure closed-loop stability is provided. Similarly, Quintero et al. (2015) use an approximated Dubins vehicle model to design an output-feedback MPC to make two UAVs track an evasive ground target.

Furthermore, Mathisen et al. (2015) focus on the longitudinal dynamics of the UAV to design an NMPC to guide the model into a deep stall to land at minimum speed and fixed path angle. The results show that the UAV is able to perform precision landing at small places like the deck of a ship, with low speed in an accurately controlled deep stall.

Also, Yang et al. (2013) propose an adaptive NMPC to solve a reference tracking task. Instead of using a fixed horizon, the adaptive strategy changes the horizon based on the path curvature profile. Therefore, smoother paths lead to smaller horizons, which reduces the computational burden and can also improve tracking performance, as it may diminish the corner cut behavior, which happens when the cost to track the curve is higher than the cost of cutting it, by following a path close to it.

Moreover, MPC can also be used as a auxiliary control law of a Feedback Linearization (FL) controller. In Zhao & Go (2014), the authors employ this FL + MPC scheme to individual quadcopters in order to keep the leader-follower formations and avoid obstacles during flight. Also, Schnelle & Eberhard (2015) applies this strategy to a serial manipulator with a passive joint, in order to track reference trajectories. A variable constraint mapping method is used to predict future states and coherent constraints.

This design is also discussed in Deng et al. (2009) and Simon et al. (2013). In the former, the authors propose a technique for handling input constraints based on simple affine transformations of the feasible area. In the latter, the authors propose a method for handling nonlinear constraints using dynamic local polytopic approximations.

1.4 Contributions

The main contribution of this work regards a unifying solution to the problem of guiding and controlling a fixed-wind UAV to converge and circulate a closed curve, in an environment with previously unknown obstacles. At a higher level, we use a vector field switching strategy as a motion planner to compute velocity references for the UAV. The strategy is composed of two behaviors, described as follows: *i*) converge to and circulate a desired curve; *ii*) circulate the closest obstacle if it is in the UAV path. Also, we propose a transition strategy, based on the Tangent Bug algorithm presented in Choset et al. (2005), which provides a smooth switching between these two behaviors.

In the lower level, we propose a control strategy to make the UAV follow the references of the vector field strategy. First, in order to obtain a linear model of the UAV, a feedback linearization controller is applied using the generalized forces and moments as virtual control inputs. Afterwards, the auxiliary control law is performed by using a linear MPC based on the resulting chain of integrators to solve both the trajectory tracking and control allocation problems, ensuring feasible mapping between the virtual control inputs and the applied ones. A common strategy in the literature that uses MPC and deals with obstacle avoidance is to include collision constraints in the MPC formulation (Hagen et al., 2018; Zhao & Go, 2010). These are usually nonlinear quadratic constraints, which can make real-time implementation impractical due to the computational effort. In this work, we consider the vector field guidance law as responsible for the obstacle avoidance, which allows us to remove the collision constraints from the MPC formulation. Because of that, together with the fact that the formulated MPC is linear, the proposed strategy is able to run in less than 10ms, thus, making it amenable to implementation in real systems.

Figure 1.7 shows the proposed structure. The blocks inside the red dashed lines correspond to the control strategy, and the one inside the blue dashed lines represents the motion planning strategy. The *UAV Dynamics* block corresponds to the airplane model. The *Control Mapping* block represents the mapping from the virtual control inputs to the

UAV control signals, and the *State Estimation* block corresponds to the Extended Kalman Filter used in this work to estimate the UAV states.

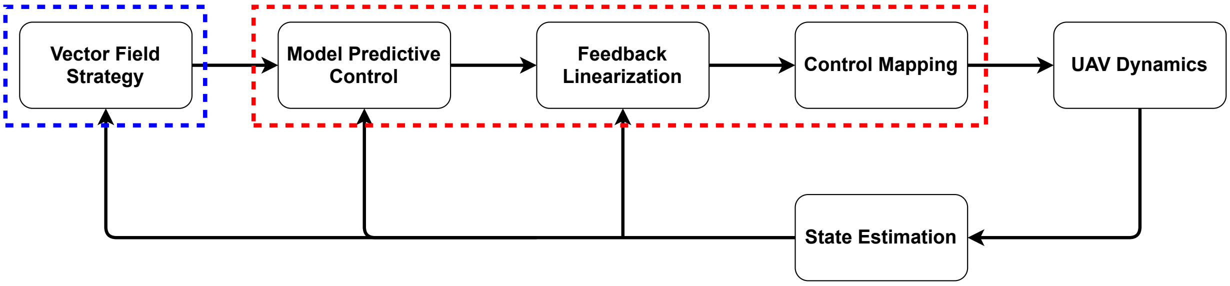

Figure 1.7: Diagram of the proposed solution.

Simulation results using the MATLAB software considering the 6DOF UAV model, wind disturbances, and sensor noise, are provided to validate the proposed strategies. We also present a computational effort analysis using an embedded computational system to validate the possibility of implementing the strategy in a real platform.

1.4.1 Publications

This dissertation originated two works: first, a publication at the *Congresso Brasileiro de Automática* (CBA), entitled *MPC Based Feedback-Linearization Strategy of a Fixed-Wing UAV* (Pereira et al., 2020). The publication is associated with the first control strategy presented in this dissertation.

Second, a work in the International Conference on Robotics and Automation (ICRA 2021), accepted for publication, entitled *Collision-free vector field guidance and MPC for a fixed-wing UAV*. This work is associated with the second control strategy, and with the motion planning strategy presented in this dissertation.

1.5 Dissertation structure

This dissertation is organized as follows: Chapter 2 anticipate some of the tools and concepts present in the text. Chapter 3 presents the vector field strategy for motion planning. Chapter 4 presents the fixed-wing control strategies. Chapter 5 presents the simulation results and finally, Chapter 6 concludes this dissertation.

2

Background

This chapter intends to present some of the tools and concepts employed throughout this dissertation. First, we introduce the fixed-wing UAV model, followed by a review of the control techniques used in this work and on the Extended Kalman Filter.

2.1 Fixed-wing model

In this section, we present a brief review of the UAV dynamics and the multiple coordinate frames used in this work. For a more detailed development of the UAV dynamics, we suggest the reading of Beard & McLain (2012) and Stevens et al. (2015).

2.1.1 Coordinate frames

To present the UAV equations of motion, we first need to define the coordinate frames in which they are expressed. The need for multiple coordinate frames arises from several reasons (Beard & McLain, 2012), among which the following stand out:

- Forces and moments actuating on the UAV are generally described in the body-fixed reference frame;

- Trajectory references are usually specified in the inertial frame;

- Sensors can provide measurements with respect to the body-fixed frame, for instance, accelerometers and gyroscopes, or with respect to the inertial frame, like GPS and barometers.

Here we seek to define the four coordinate frames used in this work: the inertial frame, the vehicle frame, the body frame, and the wind frame.

Inertial and vehicle frames

The inertial frame is an earth-fixed tangent-plane coordinate system, which has its down axis directed towards the center of the earth, and the north and east axis geographically aligned. Because of that, this frame is usually referred to as the north-east-down (NED) reference frame. It is represented by the unit vectors $\hat{\imath}$, $\hat{\jmath}$ and $\hat{k}$ in Figure 2.1. Furthermore, by translating the inertial frame to the UAV center of mass we get the vehicle frame, also represented in Figure 2.1 by the $\hat{\imath}_v$, $\hat{\jmath}_v$, $\hat{k}_v$ axis.

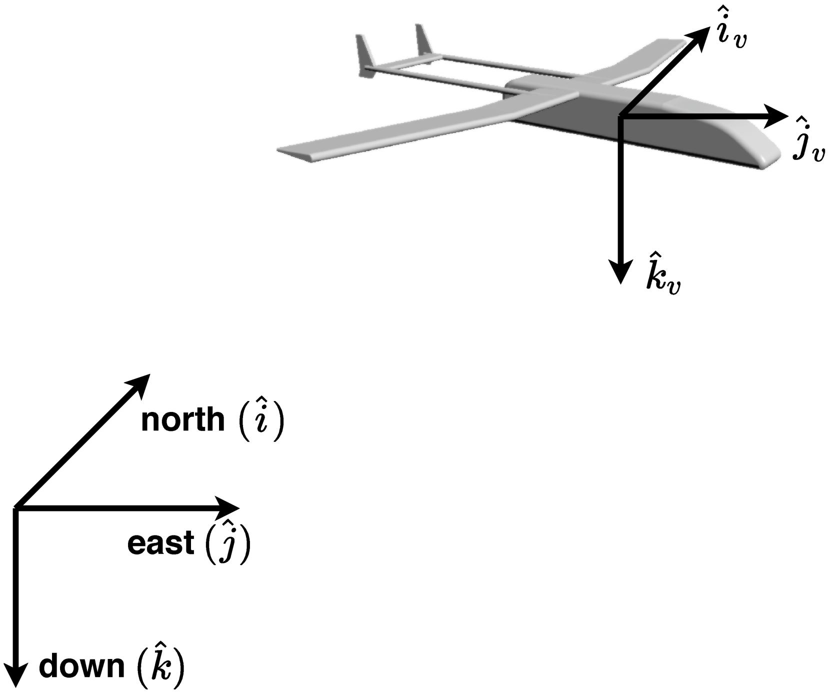

Figure 2.1: Inertial and vehicle reference frames.

Body frame

To obtain the body frame we apply a sequence of rotations in the vehicle frame, given by:

- Positive right-hand rotation about the inertial $\hat{k}_v$ axis by the yaw angle ψ.
- Positive right-hand rotation about the $\hat{\jmath}_v$ axis by the pitch angle θ.
- Positive right-hand rotation about the $\hat{\imath}_v$ axis by the roll angle ϕ.

One should note that the mentioned rotations are intrinsic, in other words, they occur about the axes of the coordinate system attached to the moving body. Figure 2.2 shows the UAV body frame. The $\hat{\jmath}_{body}$ axis points in the direction of the right wing, $\hat{\imath}_{body}$ points in the nose direction, and $\hat{k}_{body}$ completes the triedro.

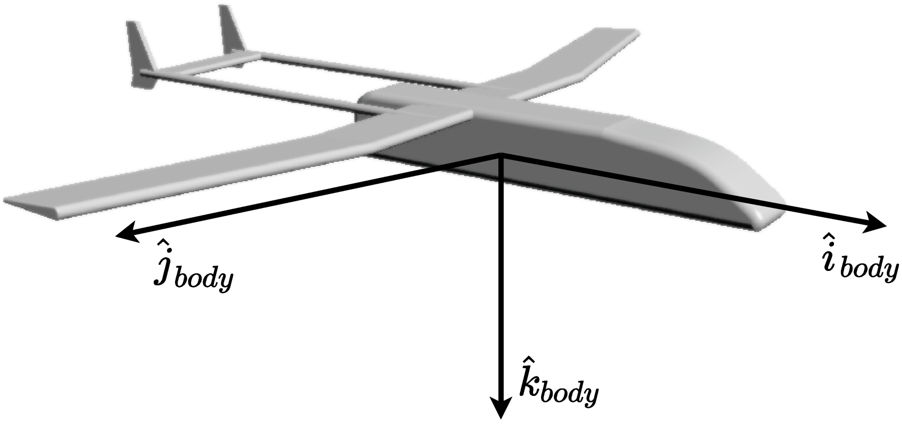

Figure 2.2: UAV body reference frame.

The roll, pitch, and yaw angles are also known as Euler angles. With them, the transformation from the vehicle frame to the body frame can be defined as

$$\boldsymbol{R}_v^b(\phi, \theta, \psi) = \begin{pmatrix} \cos\theta\cos\psi & \cos\theta\sin\psi & -\sin\theta \\ \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\sin\psi & \sin\phi\cos\theta \\ \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi & \cos\phi\cos\theta \end{pmatrix}. \quad (2.1)$$

Despite being intuitive, the Euler angle representations have a mathematical singularity when $\theta = \pm 90\text{deg}$, known as gimbal lock.

Wind frame

Given that the aerodynamics forces depend on the UAV velocity with respect to the surrounding air mass, we need to take into consideration wind effects when modeling it.

For that, we define the UAV angle-of-attack, denoted α, as the angle between the UAV $\hat{\imath}_{body}$ axis and the projection of the relative wind into the $\hat{\imath}_{body} - \hat{k}_{body}$ plane. We also define the UAV sideslip angle β, as the angle between the same projection, and the relative wind vector. By definition, α is positive when the relative wind is under the aircraft, and β is positive when the relative wind is on the right side of the aircraft.

The transformation from the body frame to the wind frame is defined by the rotation matrix (2.2). Figure 2.3 shows the wind frame, in blue, and the wind vector, in green.

$$\boldsymbol{R}_b^w(\alpha, \beta) = \begin{pmatrix} \cos\beta\cos\alpha & \sin\beta & \cos\beta\sin\alpha \\ -\sin\beta\cos\alpha & \cos\beta & -\sin\beta\sin\alpha \\ -\sin\alpha & 0 & \cos\alpha \end{pmatrix}. \quad (2.2)$$

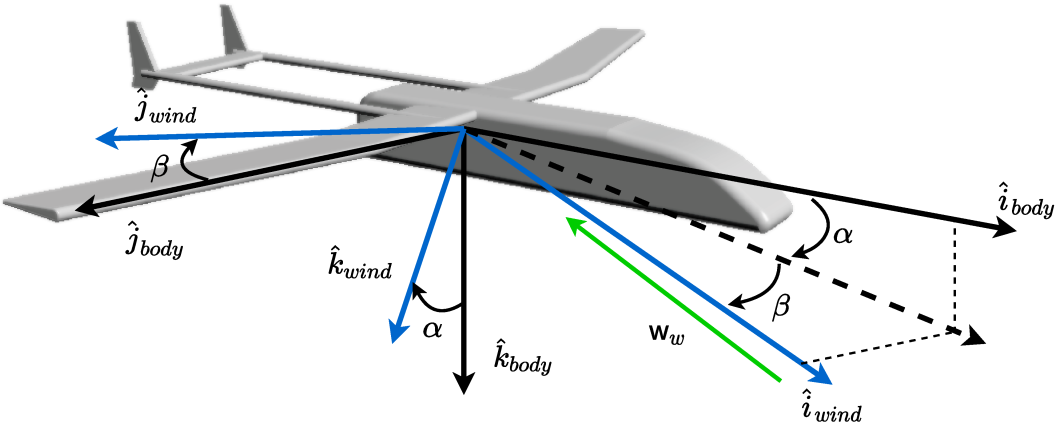

Figure 2.3: Wind reference frame, in blue, and wind vector, in green.

This way, we define the vehicle frame wind vector $\mathbf{w}_v$ as

$$\mathbf{w}_v = \begin{bmatrix} \mathsf{w}_n \\ \mathsf{w}_e \\ \mathsf{w}_d \end{bmatrix} = \boldsymbol{R}_b^v(\phi, \theta, \psi)\boldsymbol{R}_w^b(\alpha, \beta)\mathbf{w}_w, \tag{2.3}$$

where w_n, w_e and w_d are the north, east and down components of the wind vector.

2.1.2 Dynamics model

With the coordinate frames defined, we can now move on to define the UAV Equations of Motion (EOM). We will divide them into four categories: navigation, attitude, force, and moment equations.

The state variables that describes the UAV dynamics are: p_n, p_e and p_d represents the UAV inertial position, expressed in the $\hat{\imath}$, $\hat{\jmath}$ and $\hat{k}$ axis, respectively; u, v and w are the UAV body velocities, expressed in the $\hat{\imath}_{body}$, $\hat{\jmath}_{body}$ and $\hat{k}_{body}$ axis, respectively; ϕ, θ and ψ are the Euler angles (roll, pitch and yaw), and represents the UAV attitude; p, q and r are the UAV angular velocities, expressed in the $\hat{\imath}_{body}$, $\hat{\jmath}_{body}$ and $\hat{k}_{body}$ axis, respectively; $\boldsymbol{J}$ is the inertia tensor; m is the UAV mass, and g is the gravity.

Moreover, we can use (2.2) to represent the wind vector in the body frame, as follows

$$\mathbf{w}_b = \begin{pmatrix} u_{\mathsf{w}} \\ v_{\mathsf{w}} \\ w_{\mathsf{w}} \end{pmatrix} = \boldsymbol{R}_w^b(\alpha, \beta)\mathbf{w}_w. \tag{2.4}$$

This way, we can also define the UAV body frame airspeed components u_r, v_r, and w_r as

$$\begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} = \begin{pmatrix} u \\ v \\ w \end{pmatrix} - \begin{pmatrix} u_{\mathsf{w}} \\ v_{\mathsf{w}} \\ w_{\mathsf{w}} \end{pmatrix}, \tag{2.5}$$

where the subscript r indicates that we express the state variables with respect to the surrounding air mass.

Finally, f_x, f_y, f_z and l, m, n are the generalized forces and moments actuating on the UAV. These inputs comprehend the ones whose origin is related to the UAV fixed surfaces, for instance, the wings and the fuselage, and the ones that originated from the UAV control surfaces.

The navigation equations describe the UAV translational behavior in the NED frame. They are achieved by kinematics, in which the airspeed components and the wind velocity sums up as follows

$$\begin{pmatrix} \dot{p}_n \\ \dot{p}_e \\ \dot{p}_d \end{pmatrix} = \boldsymbol{R}_b^v(\phi, \theta, \psi) \begin{pmatrix} u_r \\ v_r \\ w_r \end{pmatrix} + \begin{pmatrix} \mathsf{w}_n \\ \mathsf{w}_e \\ \mathsf{w}_d \end{pmatrix}. \tag{2.6}$$

The force equations correspond to the dynamics of the UAV airspeed, and are given by

$$\begin{pmatrix} \dot{u}_r \\ \dot{v}_r \\ \dot{w}_r \end{pmatrix} = \begin{pmatrix} rv_r - qw_r \\ pw_r - r_u r \\ qu_r - pv_r \end{pmatrix} - \boldsymbol{R}_v^b(\phi, \theta, \psi) \begin{pmatrix} \dot{\mathsf{w}}_n \\ \dot{\mathsf{w}}_e \\ \dot{\mathsf{w}}_d \end{pmatrix} + \mathsf{g} \begin{pmatrix} -\sin\theta \\ \sin\phi\cos\theta \\ \cos\phi\cos\theta \end{pmatrix} + \frac{1}{\mathsf{m}} \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix}. \tag{2.7}$$

The attitude equations represent the derivatives of the angular positions in terms of the UAV angular velocities, measured in the body frame, and the angular positions themselves, given by

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{2.8}$$

One should note that the transformation is not a rotation matrix, like the one in (2.6). This is due to the fact that the angular positions are expressed with respect to three different frames, whereas the body angular velocities are expressed in the body frame.

Finally, the moment equations describe the dynamics of the UAV angular rates, measured in the body frame, as follows

$$\begin{pmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{pmatrix} = \boldsymbol{J}^{-1} \left[\begin{pmatrix} p \\ q \\ r \end{pmatrix} \times \left(\boldsymbol{J} \begin{pmatrix} p \\ q \\ r \end{pmatrix} \right) + \begin{pmatrix} l \\ m \\ n \end{pmatrix} \right], \tag{2.9}$$

Moreover, the generalized forces and moments are given by

$$f_x = \frac{1}{2}\rho_{air} v_t^2 S \left(C_X(\alpha) + C_{Xq}(\alpha)\frac{cq}{2v_t} + C_{X\delta_e}\delta_e \right) + \frac{1}{2}\rho_{air} S_{prop} C_{prop} \left(k_{motor}^2 \delta_t^2 - v_t^2 \right), \tag{2.10}$$

$$f_y = \frac{1}{2}\rho_{air} v_t^2 S \left(C_{Y0} + C_{Y\beta}\beta + C_{Yp}\frac{cp}{2v_t} + C_{Yr}\frac{br}{2v_t} + C_{Y\delta_a}\delta_a + C_{Y\delta_r}\delta_r \right), \tag{2.11}$$

$$f_z = \frac{1}{2}\rho_{air} v_t^2 S \left(C_Z(\alpha) + C_{Zq}(\alpha)\frac{cq}{2v_t} + C_{Z\delta_e}\delta_e \right), \tag{2.12}$$

$$l = \frac{1}{2}\rho_{air} v_t^2 Sb \left(C_{l0} + C_{l\beta}\beta + C_{lp}\frac{bp}{2v_t} + C_{lr}\frac{br}{2v_t} + C_{l\delta_a}\delta_a + C_{l\delta_r}\delta_r \right) - k_{Tp} k_\omega^2 \delta_t^2, \tag{2.13}$$

$$m = \frac{1}{2}\rho_{air} v_t^2 Sc \left(C_{m0} + C_{m\alpha}\alpha + C_{mq}\frac{cq}{2v_t} + C_{m\delta_e}\delta_e \right), \tag{2.14}$$

$$n = \frac{1}{2}\rho_{air} v_t^2 Sb \left(C_{n0} + C_{n\beta}\beta + C_{np}\frac{bp}{2v_t} + C_{nr}\frac{br}{2v_t} + C_{n\delta_a}\delta_a + C_{n\delta_r}\delta_r \right), \tag{2.15}$$

where the control surfaces are represented by the ailerons δ_a, the elevators δ_e, the rudder δ_r and throttle δ_t. Also, ρ_{air} is the air density, S the wing area, b the wingspan, c the mean chord, S_{prop} is the area swept by the propeller, $k_\omega \delta_t$ is the propeller speed and k_{motor} specifies the motor efficiency. Furthermore, we have

$$\begin{aligned}
C_X(\alpha) &= -C_D(\alpha)\cos(\alpha) + C_L(\alpha)\sin(\alpha) \\
C_{Xq}(\alpha) &= -C_{Dq}\cos(\alpha) - C_{Lq}\sin(\alpha) \\
C_{X\delta_e}(\alpha) &= -C_{D\delta_e}\cos(\alpha) - C_{L\delta_e}\sin(\alpha) \\
C_Z(\alpha) &= -C_D(\alpha)\sin(\alpha) - C_L(\alpha)\cos(\alpha) \\
C_{Zq}(\alpha) &= -C_{Dq}(\alpha)\sin(\alpha) - C_{Lq}(\alpha)\cos(\alpha) \\
C_{Z\delta_e}(\alpha) &= -C_{D\delta_e}(\alpha)\sin(\alpha) - C_{L\delta_e}(\alpha)\cos(\alpha),
\end{aligned}$$

with

$$\begin{aligned}
C_D(\alpha) &= C_{Dp} + \frac{\left(C_{L0} + C_{L\alpha}\alpha\right)^2}{\pi \mathsf{e} \mathrm{AR}} \\
C_L(\alpha) &= (1-\sigma)\left(C_{L0} + C_{L\alpha}\alpha\right) + \sigma\left[2\mathrm{sign}(\alpha)\sin(\alpha)^2\cos(\alpha)\right], \\
\sigma &= \frac{1 + e^{-M(\alpha-\alpha_0)} + e^{M(\alpha+\alpha_0)}}{\left(1 - e^{-M(\alpha+\alpha_0)}\right)\left(1 + e^{M(\alpha+\alpha_0)}\right)},
\end{aligned}$$

where M and α_0 are positive constants, $\mathrm{AR} = b^2/S$ is the wing aspect ratio, and e is the Oswald efficiency factor. The aerodynamic coefficients $C_{(\cdot)}$ are detailed in Appendix A.1.

In addition, we formally define the aerodynamic angles α, β and the UAV true speed

v_t as

$$\alpha = \text{arctan2}\left(w_r, u_r\right), \tag{2.16}$$

$$\beta = \arcsin\left(\frac{v_r}{v_t}\right), \tag{2.17}$$

$$v_t = \sqrt{u_r^2 + v_r^2 + w_r^2}. \tag{2.18}$$

Finally, according to Beard & McLain (2012), the wind vector can be divided into two components: a constant vector, representing the steady wind ($\mathbf{w}_s$) in the vehicle frame, and a gust wind vector ($\mathbf{w}_g$) representing atmospheric disturbances in the UAV body frame, yielding

$$\mathbf{w}_v = \mathbf{w}_s + \boldsymbol{R}_b^v(\phi, \theta, \psi)\mathbf{w}_g.$$

In this work we do not deal with the problem of measuring the wind, instead, we assume that the UAV is able to measure the steady wind vector whereas the gust wind component will be treated as an unmeasured disturbance. In practice, the steady wind could be estimated by using anemometers, as shown in Suomi & Vihma (2018).

2.2 Input-Output linearization

When dealing with tracking problems the control design can become complicated, due to the nonlinearity of the system equations. However, by using a state feedback law, we can linearize the input-output map of the system model and solve the tracking problem with linear control theory.

In this work, we use an input-output linearization strategy based on Khalil & Grizzle (2002), which we briefly review in this section. First, assume that a nonlinear system can be written in the following form

$$\begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}) + \sum_{i=1}^{m} g_i(\boldsymbol{x})u_i, \\ \boldsymbol{y} = h(\boldsymbol{x}), \end{cases} \tag{2.19}$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{y}$ is the output vector, $\boldsymbol{u} = \begin{bmatrix} u_1 & u_2 & \cdots & u_m \end{bmatrix}^T$ is the input vector, $f : D \mapsto \mathbb{R}^n$, $g_i : D \mapsto \mathbb{R}^n$ and $h : D \mapsto \mathbb{R}^j$ are sufficiently smooth on a domain $D \subset R^n$.

Now, consider the following definition (Choset et al., 2005):

Definition 1 (Diffeomorphism)**.** A function $f(x)$ that maps one differential manifold into another is said to be a diffeomorphism if $f(x)$ is bijective and both $f(x)$ and its inverse, $f^{-1}(x)$, are smooth.

The system (2.19) is said to be linearizable if there exists a diffeomorphism $T : D \mapsto \mathbb{R}^\rho$

such that the change of variables $\boldsymbol{z} = T(\boldsymbol{x})$ transforms (2.19) into the normal form

$$\begin{cases} \dot{\boldsymbol{\eta}} = f_0(\boldsymbol{\eta}, \boldsymbol{\xi}) \\ \dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{\Lambda}(\boldsymbol{x})\left(\boldsymbol{u} - \boldsymbol{\lambda}(\boldsymbol{x})\right) \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{\xi} \end{cases}, \tag{2.20}$$

where $\boldsymbol{\xi} \in \mathbb{R}^{\rho}$ determines the external dynamics, $\boldsymbol{\eta} \in \mathbb{R}^{n-\rho}$ determines the internal dynamics, $(\boldsymbol{A}, \boldsymbol{B})$ is controllable, $\boldsymbol{\lambda} : \mathbb{R}^n \mapsto \mathbb{R}^m$, $\boldsymbol{\Lambda} : \mathbb{R}^n \mapsto \mathbb{R}^{\rho \times m}$, and ρ is the relative degree. Furthermore, it is assumed that $\boldsymbol{\Lambda}(\boldsymbol{x})$ is nonsingular for all $\boldsymbol{x} \in D$.

According to Khalil & Grizzle (2002), for a system in the form of (2.19), each output y_k is said to have relative degree ρ_k in a region $D_o \subset D$ if

$$L_g L_f^{i-1} h_k(\boldsymbol{x}) = 0,\ \ i = 1, 2, ..., \rho_k;\ \ L_g L_f^{\rho_k - 1} h_k(\boldsymbol{x}) \neq 0, \quad \forall \boldsymbol{x} \in D_0$$

where $L_f h_k(\boldsymbol{x}) = \frac{\partial h(\boldsymbol{x})}{\partial \boldsymbol{x}} f(\boldsymbol{x})$ is the Lie derivative of $h_k(\boldsymbol{x})$, with respect to f throughout f.

Therefore, for a system with j outputs, the total relative degree will be the sum of each relative degree as follows

$$\rho = \sum_{k=1}^{j} \rho_k.$$

The relative degree ρ_k can also be seen as the number of times needed to differentiate y_k with respect to time, so that at least one input u_i explicitly appears in the derivative.

If $\rho \leq n$, then for all $\boldsymbol{x}_0 \in D$, there is a neighborhood N, and $n - \rho$ scalar functions $\Phi_1(\boldsymbol{x}), \Phi_2(\boldsymbol{x}), ..., \Phi_{n-\rho}(\boldsymbol{x})$ such that

$$L_g \Phi_i(\boldsymbol{x}) = 0, \quad 1 \leq i \leq n - \rho, \quad \forall \boldsymbol{x} \in N,$$

and the mapping

$$\boldsymbol{z} = T(\boldsymbol{x}) = \begin{bmatrix} \Phi_1(\boldsymbol{x}) \\ \vdots \\ \Phi_{n-\rho}(\boldsymbol{x}) \\ \hdashline h(\boldsymbol{x}) \\ \vdots \\ L_f^{\rho-1} h(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\eta} \\ \hdashline \boldsymbol{\xi} \end{bmatrix}, \tag{2.21}$$

restricted in N, is a diffeomorphism in N. Then, it is clear that the external dynamics can be linearized by using the state feedback control law

$$\boldsymbol{u} = \boldsymbol{\lambda}(\boldsymbol{x}) + \boldsymbol{\Lambda}(\boldsymbol{x})^{-1}\boldsymbol{\nu}, \tag{2.22}$$

thus, obtaining

$$\begin{cases} \dot{\boldsymbol{\eta}} = f_0(\boldsymbol{\eta}, \boldsymbol{\xi}), \\ \dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{\nu}, \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{\xi}, \end{cases} \tag{2.23}$$

where $\boldsymbol{\nu}$ is the auxiliary control variable. Furthermore, $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ are given by

$$\boldsymbol{\Lambda} = \begin{bmatrix} L_{g_1} L_f^{\rho-1} h_1(\boldsymbol{x}) & \cdots & L_{gm} L_f^{\rho-1} h_1(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{\rho-1} h_j(\boldsymbol{x}) & \cdots & L_{gm} L_f^{\rho-1} h_j(\boldsymbol{x}) \end{bmatrix}, \qquad \boldsymbol{\lambda} = -\boldsymbol{\Lambda}^{-1} \begin{bmatrix} L_f^{\rho} h_1(\boldsymbol{x}) \\ \vdots \\ L_f^{\rho} h_j(\boldsymbol{x}) \end{bmatrix}. \tag{2.24}$$

For the special case where $\rho = n$, the internal dynamics is removed, $\boldsymbol{z} = \boldsymbol{\xi}$, and (2.23) is reduced to

$$\begin{cases} \dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{\nu}, \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{\xi}. \end{cases}$$

2.3 Optimal control

In this section, we aim to provide a brief introduction to optimal control theory. One should have in mind that the field of optimal control is broad, so we focus this introduction on linear optimal control with a quadratic cost.

2.3.1 Optimization problem

An optimization problem is a problem of finding the best solution, according to some criterion, from a set of feasible solutions (Borrelli et al., 2017). For instance, suppose that we want to minimize a function f, the optimization problem can be formulated as

$$\begin{aligned} \min_{\boldsymbol{z}} \quad & f(\boldsymbol{z}), \\ s.t. \quad & G(\boldsymbol{z}) \leq \boldsymbol{0}, \\ & E(\boldsymbol{z}) = \boldsymbol{0}, \end{aligned} \tag{2.25}$$

where $f : \mathbb{R}^n \mapsto \mathbb{R}$ is the objective function, $\boldsymbol{z}$ is the vector of decision variables, $G(\boldsymbol{z}) \leq \boldsymbol{0}$ are the inequality constraints, $E(\boldsymbol{z}) = \boldsymbol{0}$ are the equality constraints, and $\boldsymbol{0}$ are zero vectors of appropriate size.

Solving (2.25) means finding the optimal value $f^* = f(\boldsymbol{z}^*)$, corresponding to the optimal solutions $\boldsymbol{z}^*$, such that

$$f(\boldsymbol{z}) \geq f(\boldsymbol{z}^*), \quad \forall \boldsymbol{z} \in \{\boldsymbol{z} \in \mathbb{R}^n : G(\boldsymbol{z}) \leq \boldsymbol{0},\ E(\boldsymbol{z}) = \boldsymbol{0}\}.$$

If the solution to (2.25) does not exist, we say that the problem is unfeasible. If for all $\boldsymbol{z} \in \mathbb{R}^n$, both $G(\boldsymbol{z}) \leq \boldsymbol{0}$ and $E(\boldsymbol{z}) = \boldsymbol{0}$, the problem is said to be unconstrained. Also, if $f^* = \pm\infty$ the problem is said to be unbounded. Furthermore, note that (2.25) can be written in the compact form

$$\min_{\boldsymbol{z} \in Z} \quad f(\boldsymbol{z}), \tag{2.26}$$

where $Z \subset \mathbb{R}^n$ is the set of feasible solutions, in other words

$$Z = \{\boldsymbol{z} \in \mathbb{R}^n : G(\boldsymbol{z}) \leq \boldsymbol{0},\ E(\boldsymbol{z}) = \boldsymbol{0}\}.$$

Convex optimization

A set $Z \subset \mathbb{R}^n$ is said to be convex if:

$$\sigma \boldsymbol{z}_1 + (1-\sigma)\boldsymbol{z}_2 \in Z, \quad \forall \boldsymbol{z}_1, \boldsymbol{z}_2 \in Z \text{ and } \sigma \in [0,1]. \tag{2.27}$$

A function $f : Z \mapsto \mathbb{R}$ is said to be convex if

$$f(\sigma \boldsymbol{z}_1 + (1-\sigma)\boldsymbol{z}_2) \leq \sigma f(\boldsymbol{z}_1) + (1-\sigma) f(\boldsymbol{z}_2), \quad \forall \boldsymbol{z}_1, \boldsymbol{z}_2 \in Z \text{ and } \sigma \in [0,1]. \tag{2.28}$$

Thus, the optimization problem (2.26) is said to be convex if (2.27) and (2.28) are met. A fundamental property of convex optimization problems is that local optimal solutions are also global optimal. We suggest the reading of Borrelli et al. (2017) for a proof.

Quadratic optimization

Quadratic optimization is one of the most well-known classes of convex optimization problems. Its popularity comes from the fact that many problems can be formulated and efficiently solved using this approach.

A problem in the form of (2.26) is said to be quadratic if the constraint functions are affine and the cost function is a convex quadratic function. A convex quadratic function can be defined as

$$f(\boldsymbol{z}) = \boldsymbol{z}^T \boldsymbol{H} \boldsymbol{z} + 2\boldsymbol{q}^T \boldsymbol{z} + r, \tag{2.29}$$

where $\boldsymbol{H}^T = \boldsymbol{H} \succeq 0 \in \mathbb{R}^{n \times n}$, $\boldsymbol{q} \in R^n$ and $r \in \mathbb{R}$. An illustration of the level curves of (2.29) is presented in Figure 2.4 by the dashed lines and the feasible solution set Z is represented by the blue polygon. The interior point indicates the optimal solution $\boldsymbol{z}^*$

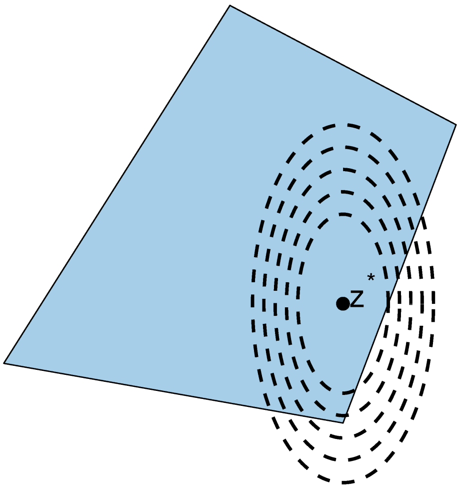

Figure 2.4: Curve lines of the cost function and feasible solution set.

2.3.2 Optimal control problem

With the definition of a quadratic optimization problem given, we can now move on to define the optimal control problem (OCP). First, consider the discrete time system

$$\begin{cases} \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \\ \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{x}_k, \end{cases} \tag{2.30}$$

where $\boldsymbol{y} \in \mathbb{R}^l$, $\boldsymbol{x} \in \mathbb{R}^n$, and $\boldsymbol{u} \in \mathbb{R}^m$. Now, suppose that we want to stabilize $\boldsymbol{x}$ at the origin from the initial time $k = 0$ to $k = \infty$. Thus, we can define a performance index given by the functional

$$J_0(\boldsymbol{x}_0, \hat{\boldsymbol{u}}) = f(\boldsymbol{x}_0, \vec{\boldsymbol{u}}) = \sum_{i=0}^{\infty} \boldsymbol{x}_i^T \boldsymbol{Q}\boldsymbol{x}_i + \boldsymbol{u}_i^T \boldsymbol{R}\boldsymbol{u}_i, \tag{2.31}$$

where $\boldsymbol{Q} = \boldsymbol{Q}^T \succeq \boldsymbol{0}$ and $\boldsymbol{R} = \boldsymbol{R}^T \succ \boldsymbol{0}$ are the state and control weighting matrices, and $\hat{\boldsymbol{u}} = \begin{bmatrix} \boldsymbol{u}_0^T & \boldsymbol{u}_1^T & \cdots \end{bmatrix}^T$. Thus, we can express the optimal control problem as

$$\begin{aligned} J_0^*(\boldsymbol{x}_0, \hat{\boldsymbol{u}}) = \quad & \min_{\hat{\boldsymbol{u}}} \quad J_0(\boldsymbol{x}_0, \hat{\boldsymbol{u}}), \\ & s.t. \quad \boldsymbol{x}_k \in \mathcal{X}, \qquad \forall k \in \{1, 2, ...\}, \\ & \qquad \boldsymbol{u}_k \in \mathcal{U}, \qquad \forall k \in \{1, 2, ...\}, \\ & \qquad \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, \quad \forall k \in \{1, 2, ...\}, \end{aligned} \tag{2.32}$$

where $\mathcal{X}$ is the state constraints set, and $\mathcal{U}$ is the control constraints set. In the case that $\mathcal{X} = \mathbb{R}^n$ and $\mathcal{U} = \mathbb{R}^m$ the problem is unconstrained, and the solution can be obtained as

$$\boldsymbol{u}_k^* = -\left(\boldsymbol{B}^T \boldsymbol{P}\boldsymbol{B} + \boldsymbol{R}\right)^{-1} \boldsymbol{B}^T \boldsymbol{P}\boldsymbol{A}\boldsymbol{x}_k, \quad k = \{1, 2, ...\}, \tag{2.33}$$

where $\boldsymbol{P}$ is the solution of the associated Algebraic Riccati Equation

$$\boldsymbol{P} = \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} + \boldsymbol{Q} - \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{B} \left(\boldsymbol{B}^T \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R}\right)^{-1} \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{A}.$$

Note that in (2.33) we have an optimal control law, instead of a control sequence. This controller is also referred to as the Linear Quadratic Regulator (LQR). For the proof of (2.33), we suggest the reading of Kirk (2004) and Borrelli et al. (2017).

Now, suppose that we want to solve problem (2.32) from $k = 0$ to $k = N$, with $N \in \mathbb{Z}$. This way, we can rewrite the OCP as

$$\begin{array}{rlll} J_0^*(\boldsymbol{x}_0, \hat{\boldsymbol{u}}) = & \min_{\hat{\boldsymbol{u}}} & \boldsymbol{x}_N^T \boldsymbol{P} \boldsymbol{x}_N + \sum_{i=0}^{N-1} \boldsymbol{x}_i^T \boldsymbol{Q} \boldsymbol{x}_i + \boldsymbol{u}_i^T \boldsymbol{R} \boldsymbol{u}_i, & \\ & s.t. & \boldsymbol{x}_k \in \mathcal{X}, & \forall k \in \{1, ..., N-1\}, \\ & & \boldsymbol{u}_k \in \mathcal{U}, & \forall k \in \{1, ..., N-1\}, \\ & & \boldsymbol{x}_N \in \mathcal{X}_f, & \\ & & \boldsymbol{x}_{k+1} = \boldsymbol{A}\boldsymbol{x}_k + \boldsymbol{B}\boldsymbol{u}_k, & \forall k \in \{1, ..., N-1\}, \end{array} \tag{2.34}$$

where $\boldsymbol{P}$ is the terminal weighting matrix, and $\mathcal{X}_f$ is the terminal set, or terminal region, that we want the states to reach at the end of the horizon N. The addition of a terminal set is mandatory in order to ensure closed-loop stability (Limón et al., 2008; Mayne et al., 2000). Typically, the terminal region is chosen as the maximum invariant set of the system.

To solve problem (2.34), we will use an alternative approach and explicitly write it as a function of the future inputs by using successive substitutions, as follows

$$\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{x}_N \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} \\ \boldsymbol{A}^2 \\ \vdots \\ \boldsymbol{A}^N \end{bmatrix} \boldsymbol{x}_0 + \begin{bmatrix} \boldsymbol{B} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{A}\boldsymbol{B} & \boldsymbol{B} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}^{N-1}\boldsymbol{B} & \boldsymbol{A}^{N-2}\boldsymbol{B} & \cdots & \boldsymbol{B} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_0 \\ \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_{N-1} \end{bmatrix}, \tag{2.35}$$

which can be reduced to

$$\boldsymbol{x} = \boldsymbol{T}\boldsymbol{x}_0 + \boldsymbol{S}\hat{\boldsymbol{u}}.$$

This way, the cost functional can be expressed as

$$J_0(\boldsymbol{x}_0, \hat{\boldsymbol{u}}) = \hat{\boldsymbol{u}}^T \boldsymbol{H} \hat{\boldsymbol{u}} + 2\boldsymbol{x}_0^T \boldsymbol{F} \hat{\boldsymbol{u}} + \boldsymbol{x}_0^T \boldsymbol{Y} \boldsymbol{x}_0,$$

where

$$\begin{aligned} \boldsymbol{H} &= \boldsymbol{S}^T \bar{\boldsymbol{Q}} \boldsymbol{S} + \bar{\boldsymbol{R}}, \\ \boldsymbol{F} &= \boldsymbol{T}^T \bar{\boldsymbol{Q}} \boldsymbol{S}, \\ \boldsymbol{Y} &= \boldsymbol{T}^T \bar{\boldsymbol{Q}} \boldsymbol{T}. \end{aligned}$$

Moreover, $\bar{\boldsymbol{Q}} = \text{blkdiag}\overbrace{(\boldsymbol{Q},\ \boldsymbol{Q},\ ...,\ \boldsymbol{P})}^{N}$ and $\bar{\boldsymbol{R}} = \text{blkdiag}\overbrace{(\boldsymbol{R},\ \boldsymbol{R},\ ...,\ \boldsymbol{R})}^{N}$. Therefore, the optimal control problem resumes to

$$\begin{aligned} J_0^*(\boldsymbol{x}_0, \hat{\boldsymbol{u}}) = \quad & \min_{\hat{\boldsymbol{u}}} \quad \hat{\boldsymbol{u}}^T \boldsymbol{H} \hat{\boldsymbol{u}} + 2\boldsymbol{x}_0^T \boldsymbol{F} \hat{\boldsymbol{u}} + \boldsymbol{x}_0^T \boldsymbol{Y} \boldsymbol{x}_0, \\ & s.t. \quad\ G(\hat{\boldsymbol{u}}) \leq \boldsymbol{0}, \\ & \qquad\quad\ E(\hat{\boldsymbol{u}}) = \boldsymbol{0}. \end{aligned} \tag{2.36}$$

The optimal solution $\hat{\boldsymbol{u}}^*$ is achieved by solving (2.36). Note that the term $\boldsymbol{x}_0^T \boldsymbol{Y} \boldsymbol{x}_0$ does not affect the result, as it does not depend on $\hat{\boldsymbol{u}}$, therefore it can be removed from the problem. Furthermore, if the restrictions $E(\hat{\boldsymbol{u}}) = \boldsymbol{0}$ and $G(\hat{\boldsymbol{u}}) \leq \boldsymbol{0}$ are affine, they can be written in the compact form

$$\begin{aligned} \boldsymbol{A}_{eq} \hat{\boldsymbol{u}} &= \boldsymbol{b}_{eq}, \\ \boldsymbol{A}_{ineq} \hat{\boldsymbol{u}} &\leq \boldsymbol{b}_{ineq}. \end{aligned}$$

We will use this compact notation throughout the text.

2.3.3 Receding Horizon Control

In the previous section, we presented the solution of a constrained, finite time optimal control problem. Here we seek to propose a different approach, using the Receding Horizon Control (RHC) formulation. For that, an optimal control problem with a finite horizon is solved at the initial time $k = 0$. Then, the first component of the optimal control sequence u_0^* is applied to the system during the interval $[k\Delta t, (k+1)\,\Delta t)$, where Δt is the sampling period. At the next time step, $k+1$, the initial states are updated, and we solve a new optimal control problem over a shifted horizon, and we keep repeating this procedure. An RHC, where the optimal control sequences are computed online, is called Model Predictive Control (MPC).

2.4 Extended Kalman Filter

One of the main challenges in UAV control is the state measurement. Generally, a UAV is equipped with onboard sensors to measure some of its states, however, some of the UAV states cannot be directly measured, and the ones we can measure can be subjected to noise and uncertainty. One solution to this problem is to use the available sensors and an observer to estimate the UAV states. A well-known observer is the Kalman Filter (KF), also known as the Linear Quadratic Estimator (LQE), (Kalman et al., 1960). The KF is an algorithm that produces estimates of unknown variables from a series of measurements by estimating a joint probability distribution for each time frame. For nonlinear systems, it is common to use the Extended Kalman Filter (EKF), which we briefly discuss in this section.

For that let $\boldsymbol{\Gamma}_p$ and $\boldsymbol{\Gamma}_o$ be the process noise and the observation noise covariance matrices, and $\boldsymbol{\Gamma}_e$ the predicted covariance estimate. Moreover, let a nonlinear system be defined as

$$\begin{cases} \dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{w}, \\ \boldsymbol{y} = h(\boldsymbol{x}), \end{cases} \tag{2.37}$$

where $\boldsymbol{x}$ is the state vector, $\boldsymbol{y}$ the output vector, $\boldsymbol{u}$ is the input vector, $\boldsymbol{w}$ is the noise vector, and f, h are differentiable functions. Thus, assuming that both f and h are a good approximation of the real model, we can use them to compute the predicted state from previous estimates.

The EKF algorithm may then be defined in two steps: prediction and correction. In the prediction step, we estimate the UAV states using the sensors readings as inputs of the model, and update the covariance matrix, $\boldsymbol{\Gamma}_{\boldsymbol{e}}$. In the correction step, we compare the estimated outputs with the data provided by another set of sensors to update the state estimates and $\boldsymbol{\Gamma}_{\boldsymbol{e}}$. Algorithm 1 resumes this procedure, which was modified from Beard & McLain (2012). The function *SensorData*() returns the data from the sensors used to correct the estimates. Furthermore, $\tilde{\boldsymbol{x}}$ are the state estimates, $\tilde{\boldsymbol{u}}$ the sensor inputs, Δt_{EKF} is the sensor sampling time, used to discretize (2.37), and $\mathbb{I}$ is the identity matrix.

Algorithm 1: Extended Kalman Filter algorithm.

```
while True do
    if Sensor data for prediction available then
        $\tilde{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}} + f(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})\Delta t_{EKF}$ ;                 ▷ Predict step
        $\boldsymbol{F} \leftarrow \frac{\partial f}{\partial \boldsymbol{x}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$ ;
        $\boldsymbol{\Gamma}_e \leftarrow \boldsymbol{\Gamma}_e + \Delta t_{EKF}\left(\boldsymbol{F}\boldsymbol{\Gamma}_e + \boldsymbol{\Gamma}_{\boldsymbol{e}}\boldsymbol{F}^T + \boldsymbol{\Gamma}_{\boldsymbol{p}}\right)$ ;
    if Sensor data for correction available then
        SD $\leftarrow$ SensorData() ;                                  ▷ Correction step
        $\boldsymbol{H} \leftarrow \frac{\partial h}{\partial \boldsymbol{x}}(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{u}})$;
        $\boldsymbol{L} \leftarrow \frac{\boldsymbol{\Gamma}_{\boldsymbol{e}}\boldsymbol{H}^T}{\boldsymbol{\Gamma}_{\boldsymbol{o}} + \boldsymbol{H}\boldsymbol{\Gamma}_{\boldsymbol{e}}\boldsymbol{H}^T}$ ;
        $\boldsymbol{\Gamma}_e \leftarrow (\mathbb{I} - \boldsymbol{L}\boldsymbol{H})\,\boldsymbol{\Gamma}_e$ ;
        $\tilde{\boldsymbol{x}} \leftarrow \tilde{\boldsymbol{x}} + \boldsymbol{L}\left(\mathrm{SD} - h(\tilde{\boldsymbol{x}})\right)$ ;
```

3

Motion planning strategy

In this chapter, we present the development of the proposed guidance strategy. Although we apply it on a fixed-wing UAV, the proposed strategy can also be used in the guidance of a broad class of mobile robots.

3.1 Problem formulation

In this work, we consider the task of guiding a fixed-wing UAV to converge to and circulate a closed curve $\mathrm{C}(t) \in \mathbb{R}^3$ while deviating from moving obstacles.

Assumption 1. The obstacles can be represented as rigid vertical cylinders that can move only in the horizontal plane, with velocities bounded below by the UAV stall speed, v_{stall}.

Let $\mathcal{O}(t)$ be the set of cylindrical obstacles, indexed by $i, \mathcal{O}(t) = \{\mathcal{O}_1(t),\ \mathcal{O}_2(t), ...\}$, that can be represented by their position in the horizontal plane, which can be time varying, and by the radius $\mathcal{O}_{i,r}$ of a corresponding circumscribed cylinder.

Assumption 2. The moving obstacles have some intelligence, so that two obstacles $\mathcal{O}_i$ and $\mathcal{O}_j$ do not move towards each other when inside a collision range.

Assumption 1 is needed to ensure that an obstacle will not move faster than the UAV, so it can be circulated. Assumption 2 ensures that obstacles will not collide with each other and, once the UAV starts circulating an obstacle, it will have space to finish the maneuver. Furthermore, in this work we do not consider the problem of detecting the obstacles, thus we also have the following assumption:

Assumption 3. For each obstacle $\mathcal{O}_i \in \mathcal{O}$ within a given range d_{in} from the UAV, both position and velocity of $\mathcal{O}_i$ are known.

In practice, the obstacle position and velocity could be estimated using a camera, as shown in Pester & Schrittesser (2019); Ali & Ragb (2019). If they are robots, they can also communicate this information. With these assumptions, we can formally define the motion planning problem as follows:

Problem 1. Given a target curve $\mathrm{C}(t)$ and a set of obstacles $\mathcal{O}$, find a vector field $F(\boldsymbol{x}, t)$ to guide the UAV to converge to and circulate C while avoiding the obstacles in $\mathcal{O}$ under Assumptions 1, 2 and 3.

In order to find a solution to Problem 1, we will first divide it into minor problems, or steps. This way, we are able to develop the solution to each one separately. The three minor problems are defined as follows:

- Find a vector field $F_c(\boldsymbol{x}, t)$ to converge to and circulate the target curve C;
- Find a vector field $F_o(\boldsymbol{x}, t)$ to converge to and circulate any obstacle $\mathcal{O}_i \in \mathcal{O}$;
- Find a switching law to combine $F_c(\boldsymbol{x}, t)$ and $F_o(\boldsymbol{x}, t)$ in order to solve Problem 1.

3.2 Curve field

In this section, we present the solution for the first step of Problem 1. The proposed solution relies on the implementation of a vector field strategy, initially proposed in Gonçalves et al. (2010) for n dimensions. Since we are dealing with a three dimensional workspace, we will briefly recall the strategy using $n = 3$.

First, consider a robot represented by a configuration vector $\boldsymbol{x} = [x_1 \ x_2 \ x_3]^T$. Note that $\boldsymbol{x} \in \mathbb{R}^3$ is the trajectory position, and can be related to the NED coordinate system. Moreover, we initially consider that the configuration space is collision-free. Also, we assume that the robot can be modeled as a single integrator, $\dot{\boldsymbol{x}} = \boldsymbol{u}_{\mathrm{x}}$, where $\boldsymbol{u}_{\mathrm{x}}$ is the control input.

Now, let $\kappa_i(\boldsymbol{x}, t) : \mathbb{R}^4 \to \mathbb{R}, \ i = 1, 2$ be scalar functions with bounded second-partial derivatives and with linearly independent gradients at the target curve. With that, we can formally define the target curve $\mathrm{C}(t)$ as

$$\mathrm{C} = \{[x_1 \ x_2 \ x_3 \ t]^T \in \mathbb{R}^4 : \kappa_i(x_1, x_2, x_3, t) = 0, \ i = 1, 2\}.$$

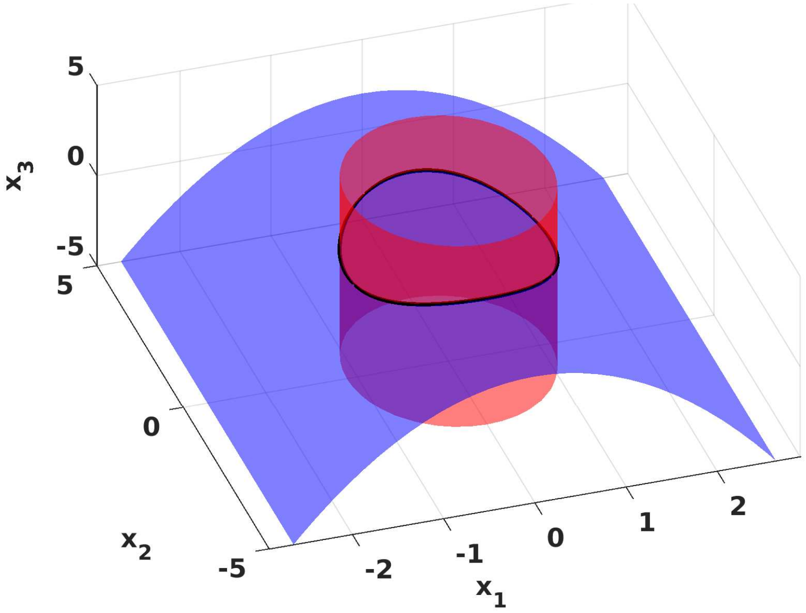

Figure 3.1: Target curve, in black, and zero sets of $\kappa_1(\boldsymbol{x}, t) = -x_3 + x_1^2 + 2 = 0$, in blue, and $\kappa_2(\boldsymbol{x}, t) = 4x_1^2 + x_2^2 - 5 = 0$ in red.

Figure 3.1 shows an example of a target curve, in black, defined by the zero level sets of the respective $\kappa_i(\boldsymbol{x}, t)$ functions.

With those definitions, we can decompose the vector field in three components: one which converges to the curve, one to circulate it, and one to compensate for the time-varying behavior. Now, let $V(\kappa_1, \kappa_2) : \mathbb{R}^4 \to \mathbb{R}$ be a differentiable positive-definite function, such that $V = \nabla V = 0 \iff \kappa_1 = \kappa_2 = 0$. Therefore, by setting $V = \nabla V = 0$ we ensure convergence. Furthermore, we have that $\dot{V}$ is given by

$$\frac{dV}{dt} = \nabla V^T \dot{\boldsymbol{x}} + \frac{\partial V}{\partial t} = \nabla V^T \boldsymbol{u}_{\mathbf{x}} + \frac{\partial V}{\partial t}, \tag{3.1}$$

where ∇ is the gradient with respect to $\boldsymbol{x}$. Thus, we can set

$$\boldsymbol{u}_{\mathbf{x}} = G(x_1, x_2, x_3, t)\nabla V + \boldsymbol{F_f}, \tag{3.2}$$

where $G(x_1, x_2, x_3, t) : \mathbb{R}^4 \to \mathbb{R}$ is a non-positive scalar function, with possible null value only when $\nabla V = 0$, and with bounded partial derivatives. Also, $\boldsymbol{F_f}$ is the time-varying compensation component. It should be mentioned that in the case of a static curve, we can set $\boldsymbol{F_f} = 0$. Otherwise, according to Gonçalves et al. (2010), it must satisfy the following equality:

$$\nabla \kappa_i^T \boldsymbol{F_f} = -\frac{\partial \kappa_i}{\partial t}, \; i = 1, 2. \tag{3.3}$$

We can rewrite (3.3) in matrix form as

$$\underbrace{\begin{bmatrix} \nabla\kappa_1^T \\ \nabla\kappa_2^T \end{bmatrix}}_{\boldsymbol{M_*}} \boldsymbol{F_f} = -\underbrace{\begin{bmatrix} \frac{\partial\kappa_1}{\partial t} \\ \frac{\partial\kappa_2}{\partial t} \end{bmatrix}}_{\boldsymbol{a_*}}. \tag{3.4}$$

Therefore, as long as (3.4) is ensured, the time-varying nature of the curve will be compensated. Moving on, by replacing (3.2) and (3.4) in (3.1) we obtain

$$\frac{dV}{dt} = G\|\nabla V\|^2.$$

Since G is negative when $\nabla V \neq 0$, then $V \to 0$ and the convergence to the curve is ensured. Suppose now that $\boldsymbol{x}$ has converged to the desired curve. Thus, we have $\nabla V = 0$ and (3.2) reduces to $\boldsymbol{u_x} = \boldsymbol{F_f}$, which will only maintain the robot in the curve. Therefore we need a third therm $\boldsymbol{F_r}$ to make it circulate C.

For that, it is necessary to ensure $\|\boldsymbol{F_r}\| \neq 0$ and that $\dot{\kappa}_1 = \dot{\kappa}_2 = 0$. Thus, we need

$$\frac{d\kappa_i}{dt} = \nabla\kappa_i^T(\boldsymbol{F_r} + \boldsymbol{F_f}) + \frac{\partial\kappa_i}{\partial t} = 0,\ \ i = 1, 2. \tag{3.5}$$

With (3.3) and (3.4) we can rewrite (3.5) as

$$\boldsymbol{M_*}\boldsymbol{F_r} = 0,$$

where $\boldsymbol{M_*}$ is a 2×3 matrix, such that the i-th row is given by $\nabla\kappa_i^T$. If $\boldsymbol{M_*}$ has full row rank, then there is only one solution to $\boldsymbol{F_r}$, which we define as

$$\boldsymbol{F_r} = H(x_1, x_2, x_3, t)\left(\nabla\kappa_1 \times \nabla\kappa_2\right), \tag{3.6}$$

where $H(x_1, x_2, x_3, t) : \mathbb{R}^4 \mapsto \mathbb{R}$ is a continuous, strictly positive or negative, scalar function. Thus, the sign of H will define the circulation direction.

Finally, note that as C is time varying, it may have a tangent velocity component, which would imply that both $\boldsymbol{F_f}$ and $\boldsymbol{F_r}$ could contribute to guide the system in the tangent direction of C. In order to avoid that behavior, (3.4) is modified to ensure that $\boldsymbol{F_r}$ and $\boldsymbol{F_f}$ are orthogonal, thus obtaining

$$\underbrace{\begin{bmatrix} \nabla\kappa_1^T \\ \nabla\kappa_2^T \\ (\nabla\kappa_1 \times \nabla\kappa_2)^T \end{bmatrix}}_{\boldsymbol{M_c}} \boldsymbol{F_f} = -\underbrace{\begin{bmatrix} \frac{\partial\kappa_1}{\partial t} \\ \frac{\partial\kappa_2}{\partial t} \\ 0 \end{bmatrix}}_{\boldsymbol{a_c}} \to \boldsymbol{F_f} = -\boldsymbol{M_c}^{-1}\boldsymbol{a_c}.$$

With that, circulation can be ensured by choosing a proper H, so that $\|\boldsymbol{F_r}\|$ is greater than any tangent velocity component. Therefore, we can define $\boldsymbol{u}_{\mathbf{x}}$ as

$$\boldsymbol{u}_{\mathbf{x}} = G\nabla V + H(\nabla\kappa_1 \times \nabla\kappa_2) - \boldsymbol{M}_c^{-1}\boldsymbol{a}_c. \tag{3.7}$$

Proof of convergence and circulation for (3.7) can be found in Gonçalves et al. (2010). Furthermore, in this work, we desire to converge to and circulate C with a speed close to a given reference speed ϑ. For that, we define the vector field to the target curve $F_c(x_1, x_2, x_3, t)$ as

$$F_c(x_1, x_2, x_3, t) = \vartheta G \frac{\nabla V}{\|\nabla V\|} + \vartheta H \frac{\nabla\kappa_1 \times \nabla\kappa_2}{\|\nabla\kappa_1 \times \nabla\kappa_2\|} - \boldsymbol{M}_c^{-1}\boldsymbol{a}_c, \tag{3.8}$$

with G and H defined as

$$G = -\frac{2}{\pi}\arctan(V), \tag{3.9}$$

$$H = \sqrt{1 - G^2}, \tag{3.10}$$

$$V = K_1\kappa_1^2 + K_2\kappa_2^2, \tag{3.11}$$

where K_1 and K_2 are positive constants. Moreover, note that in (3.8) we do not normalize the feedforward component, as by doing it, the time varying component of the curve may not be compensated.

3.3 Obstacle avoidance

With the vector field to converge to and circulate the pre-defined curve, we can now move on to the obstacle avoidance problem. As mentioned before, we consider that each obstacle $\mathcal{O}_i \in \mathcal{O}$ can be fully represented by the position in the horizontal plane, denoted by $\mathcal{O}_{i,x_1}(t)$ and $\mathcal{O}_{i,x_2}(t)$, and by the radius $\mathcal{O}_{i,r}$ of a corresponding cylinder circumscribing it. Figure 3.2 shows an example of a triangular obstacle, in red, and its corresponding cylinder, in blue.

It should be noted that though we are able to extend the method to a broad class of objects, for some shapes, this strategy can be very conservative, as shown in the example above. However, this approach allows us to reduce the dimension of the problem from $\mathbb{R}^3$ to $\mathbb{R}^2$. It is also important to mention that in some scenarios it is not interesting to allow maneuvers above or under a dynamic obstacle.

Note that since we are considering obstacles represented by cylinders, we can define a target circle around the i-th obstacle, such that by following this circle the UAV will circulate the obstacle. Moreover, since circulating the obstacle at any height does not affect avoidance, we can reduce our problem to a two dimensional one. This way, for each

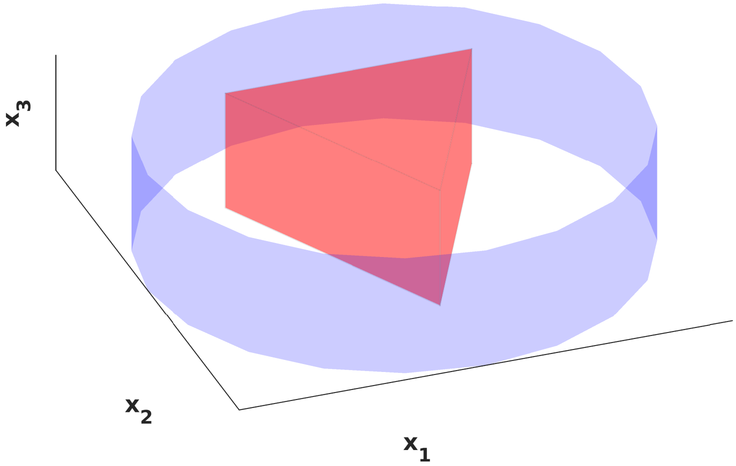

Figure 3.2: Example of an obstacle, in red, and its corresponding circumscribed cylinder, in blue.

obstacle $\mathcal{O}_i$, we can define a correspondent scalar function $\mathcal{O}_{i,\kappa}(x_1, x_2, t)$ as

$$\mathcal{O}_{i,\kappa}(x_1, x_2, t) = \left(x_1 - \mathcal{O}_{i,x_1}(t)\right)^2 + \left(x_2 - \mathcal{O}_{i,x_2}(t)\right)^2 - \mathcal{O}_{i,R}^2. \tag{3.12}$$

Note that $\mathcal{O}_{i,\kappa}(x_1, x_2, t)$ defines the target circle C_i, of radius $\mathcal{O}_{i,R}$, around the i-th obstacle. Also, it should be noted that $\mathcal{O}_{i,R}$ must be greater than $\mathcal{O}_{i,r}$, otherwise by following (3.12) the UAV could hit the obstacle.

We can now use the same strategy presented in Section 3.2 to define the vector field $F_o(\boldsymbol{x}, t, i, s)$ that leads to the avoidance of the i-th obstacle, where s is the circulation direction. First, let us define an auxiliary, bidimensional vector field $\bar{F}_o(\boldsymbol{x}, t, i, s)$ which only converges to and circulates C_i. For the convergence component, we can use the $\mathcal{O}_{i,\kappa}$ functions to define

$$V_i = \mathcal{O}_{i,\kappa}.$$

As for the circulation component, we need to find a suitable $\boldsymbol{F_r}$ such that the following equality is satisfied:

$$\begin{bmatrix} \frac{\partial \mathcal{O}_{i,\kappa}}{\partial x_1} & \frac{\partial \mathcal{O}_{i,\kappa}}{\partial x_2} \end{bmatrix} \boldsymbol{F_r} = 0.$$

An evident choice is the Hamiltonian Gradient ∇_H of $\mathcal{O}_{i,\kappa}$, given by

$$\nabla_H \mathcal{O}_{i,\kappa} = \begin{bmatrix} -\frac{\partial \mathcal{O}_{i,\kappa}}{\partial x_2} & \frac{\partial \mathcal{O}_{i,\kappa}}{\partial x_1} \end{bmatrix}^T.$$

With that, we can set $\boldsymbol{M}_o$ and $\boldsymbol{a}_o$ to

$$\boldsymbol{M}_o = \begin{bmatrix} \nabla \mathcal{O}_{i,\kappa} & \nabla_H \mathcal{O}_{i,\kappa} \end{bmatrix}^T, \qquad \boldsymbol{a}_o = -\begin{bmatrix} \frac{\partial \mathcal{O}_{i,\kappa}}{\partial t} & 0 \end{bmatrix}^T.$$

This way, we ensure that $\boldsymbol{F_r}$ and $\boldsymbol{F_f}$ are orthogonal. Now, we can define the obstacle circulation field as

$$\bar{F}_o(\boldsymbol{x}, t, i, s) = \vartheta G_i \frac{\nabla \mathcal{O}_{i,\kappa}}{\|\nabla \mathcal{O}_{i,\kappa}\|} + s\vartheta H_i \frac{\nabla_H \mathcal{O}_{i,\kappa}}{\|\nabla_H \mathcal{O}_{i,\kappa}\|} - \boldsymbol{M}_o^{-1}\boldsymbol{a}_o, \tag{3.13}$$

with G_i and H_i given by

$$G_i(x_1, x_2, t) = -\frac{2}{\pi}\arctan(K_{G_i} V_i), \tag{3.14}$$

$$H_i(x_1, x_2, t) = \sqrt{1 - G_i(x_1, x_2, t)^2}, \tag{3.15}$$

where K_{G_i} is a positive constant. Furthermore, as following F_c is the main task, s is chosen in order to generate the smallest field's change:

$$s = \begin{cases} 1, & \boldsymbol{x_3}(\boldsymbol{r}_i \times F_c(\boldsymbol{x}, t)) \leq 0, \\ -1, & \textit{otherwise}. \end{cases} \tag{3.16}$$

$$\boldsymbol{x_3} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix},$$

with $\boldsymbol{r}_i$ being the vector that points from the UAV to the i-th obstacle center.

Now, as mentioned before, since the x_3 component does not affect the avoidance behavior, we can combine (3.13) and (3.8) to obtain $F_o(\boldsymbol{x}, t, i, s)$ as

$$F_o(\boldsymbol{x}, t, i, s) = \begin{bmatrix} \vartheta' G_i \frac{\nabla \mathcal{O}_{i,\kappa}}{\|\nabla \mathcal{O}_{i,\kappa}\|} + s\vartheta' H_i \frac{\nabla_H \mathcal{O}_{i,\kappa}}{\|\nabla_H \mathcal{O}_{i,\kappa}\|} \\ \boldsymbol{x_3} F_c(\boldsymbol{x}, t) \end{bmatrix} - \begin{bmatrix} \boldsymbol{M}_o^{-1}\boldsymbol{a}_o \\ 0 \end{bmatrix}, \tag{3.17}$$

with ϑ' given by

$$\vartheta' = \sqrt{\vartheta^2 - \|\boldsymbol{x_3} F_c(\boldsymbol{x}, t)\|^2}.$$

With (3.17) we obtain a vector field that circulates the i-th obstacle while following the x_3 reference of $F_c(\boldsymbol{x}, t)$. Also, note that by multiplying only the x_1 and x_2 component of F_o by ϑ' we are able to maintain the norm of the field close to ϑ. Note also that though we multiply the feedforward component of F_c, it will not affect the obstacle avoidance, given that we keep the feedforward component of F_o unchanged.

3.4 Composite vector field

Now, in order to provide a soft transition between the vector field behaviors, we need to define a transition law. For that, consider that each transition lasts for T_γ seconds. Also, let t_γ be the time during transition, and γ a function that maps the interval $[0,\ T_\gamma]$ into $[0,\ 1]$, for instance $\gamma = t_\gamma T_\gamma^{-1}$. This way, we can define the composite vector field $F(\boldsymbol{x}, t, i, s)$

as a convex combination of F_o and F_c by setting

$$F(\boldsymbol{x}, t, i, s) = \bar{\vartheta}\frac{\bar{F}(\boldsymbol{x}, t, i, s)}{\|\bar{F}(\boldsymbol{x}, t, i, s)\|}, \tag{3.18}$$

where $\bar{\vartheta}$ and $\bar{F}(\boldsymbol{x}, t, i, s)$ are given by

$$\bar{\vartheta} = (1-\bar{\gamma})\|F_c(\boldsymbol{x}, t)\| + \bar{\gamma}\|F_o(\boldsymbol{x}, t, i, s)\| \tag{3.19}$$

$$\bar{F}(\boldsymbol{x}, t, i, s) = (1-\bar{\gamma})F_c(\boldsymbol{x}, t) + \bar{\gamma}F_o(\boldsymbol{x}, t, i, s), \tag{3.20}$$

$$\bar{\gamma}(\gamma) = 3\gamma^2 - 2\gamma^3. \tag{3.21}$$

Note that in (3.19) and (3.20) we consider the transition from F_c to F_o. If considering the opposite scenario F_c and F_o must be exchanged. Also, for $\gamma \in [0,\ 1]$, the function $\bar{\gamma}(\gamma)$ ensures a smooth transition between the vector fields, since $\dot{\bar{\gamma}}(0) = \dot{\bar{\gamma}}(1) = 0$. We also use this transition to perform a convex combination of the norms of the field, (3.19). This is necessary to ensure continuity of the norm of F, since $\|F_o\| \neq \vartheta$ when $\mathcal{O}_i$ is moving.

Lemma 1. A sufficient condition to ensure that the trajectory will not reach the obstacles is that

$$T_\gamma < \frac{d_{in} - (\mathcal{O}_{i,R} - \mathcal{O}_{i,r})}{\bar{\vartheta}_{max}}, \tag{3.22}$$

where $\bar{\vartheta}_{max}$ is the maximum norm of the vector field.

Proof. In the beginning of the transition phase, the distance from the UAV to C_i will be $d_{in} - (\mathcal{O}_{i,R} - \mathcal{O}_{i,r})$. In the worst hypothetical scenario, the UAV will be moving towards it with speed $\bar{\vartheta}_{max}$. Therefore it will take $\frac{d_{in} - (\mathcal{O}_{i,R} - \mathcal{O}_{i,r})}{\bar{\vartheta}_{max}}$ seconds to reach C_i. Thus, if (3.22) is respected, the transition will be completed before UAV reaches C_i. Also, by continuing to follow F_o, the UAV will reach C_i from the outside and thus will not reach the obstacle. □

Therefore, when no obstacles are detected, the vector field $F(\boldsymbol{x}, t, s, i)$ will be the one to the curve, $F_c(\boldsymbol{x}, t)$, as shown in Figure 3.3. The robot is represented by the blue dot, the obstacle by the brown disk, with the target circle in yellow, d_{in} in green, and the target curve in black. When an obstacle $\mathcal{O}_i$ is detected inside the perception range d_{in}, and $F_c(\boldsymbol{x}, t)$ points in a direction that decreases the distance to the obstacle, a transition from F_c to F_o is initiated. This transition is shown in Figure 3.4, where the curve field, F_c, is represented in blue, the obstacle field, F_o, in red, and the composite field, F, in dark green. The transition will last for T_γ seconds. After that, the vector field $F(\boldsymbol{x}, t, s, i)$ will be the one to circulate the obstacle, $F_o(\boldsymbol{x}, t, s, i)$, as illustrated in Figure 3.5. This behavior lasts until $F_c(\boldsymbol{x}, t)$ starts to point in a direction that increases the distance from $\mathcal{O}_i$. At this moment, a transition from F_o to F_c begins, as depicted in Figure 3.6.

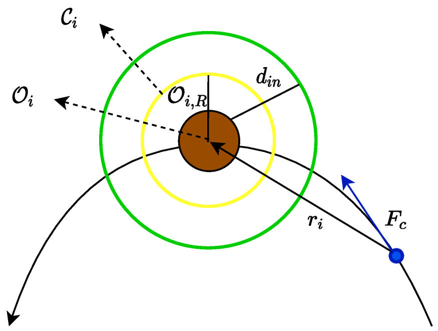

Figure 3.3: Robot in follow curve state.

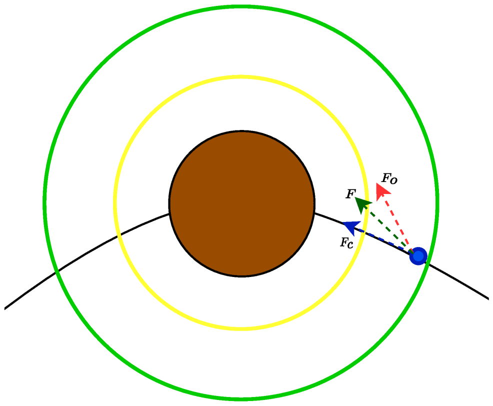

Figure 3.4: Robot in transition to obstacle state.

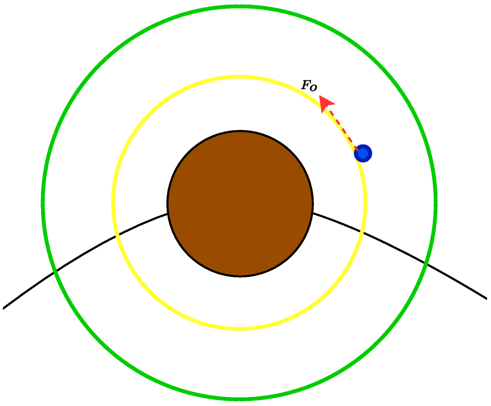

Figure 3.5: Robot in circulate obstacle state.

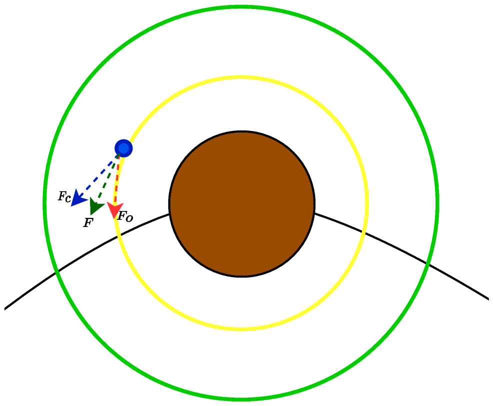

Figure 3.6: Robot in transition to curve state.

Algorithm 2 formally defines the proposed strategy. The function $ClosestObstacle(t, \mathcal{O})$ returns the index of the closest obstacle to the UAV, inside the perception range d_{in}, at time t. If there are no obstacles, it returns -1.

After defining the guidance strategy, a control strategy is needed in order to steer the UAV to the vector field references. In the next chapter, we propose two strategies as solutions to the control problem.

Algorithm 2: Online Switching Vector Field

```
$state \leftarrow follow\_curve$    ▷ Initial state
while $True$ do
    $i \leftarrow ClosestObstacle(t, \mathcal{O})$
    switch $state$ do
        case $follow\_curve$
            $F(\boldsymbol{x}, t, s, i) \leftarrow F_c(\boldsymbol{x}, t)$
            if $\boldsymbol{r}_i^T F_c(\boldsymbol{x}, t) > 0$ then
                $state \leftarrow transition\_to\_obstacle$
                $s \leftarrow getCircDir(\boldsymbol{x}, t, i)$
                $\gamma \leftarrow 0$
        case $transition\_to\_obstacle$
            $\bar{\gamma} \leftarrow 3\gamma^2 - 2\gamma^3$
            $\bar{\vartheta} \leftarrow (1 - \bar{\gamma})\|F_c(\boldsymbol{x}, t)\| + \bar{\gamma}\|F_o(\boldsymbol{x}, t, s, i)\|$
            $F(\boldsymbol{x}, t, s, i) \leftarrow (1 - \bar{\gamma})F_c(\boldsymbol{x}, t) + \bar{\gamma}F_o(\boldsymbol{x}, t, s, i)$
            $F(\boldsymbol{x}, t, s, i) \leftarrow \bar{\vartheta}F(\boldsymbol{x}, t, s, i)\|F(\boldsymbol{x}, t, s, i)\|^{-1}$
            $\gamma \leftarrow \gamma + T_\gamma^{-1}\Delta t$
            if $\gamma \geq 1$ then
                $state \leftarrow circulate\_obstacle$
        case $circulate\_obstacle$
            $F(\boldsymbol{x}, t, s, i) \leftarrow F_o(\boldsymbol{x}, t, s, i)$
            if $\boldsymbol{r}_i^T F_c(\boldsymbol{x}, t) < 0$ then
                $state \leftarrow transition\_to\_curve$
                $\gamma \leftarrow 0$
        case $transition\_to\_curve$
            $\bar{\gamma} \leftarrow 3\gamma^2 - 2\gamma^3$
            $\bar{\vartheta} \leftarrow (1 - \bar{\gamma})\|F_o(\boldsymbol{x}, t, s, i)\| + \bar{\gamma}\|F_c(\boldsymbol{x}, t)\|$
            $F(\boldsymbol{x}, t, s, i) \leftarrow (1 - \bar{\gamma})F_o(\boldsymbol{x}, t, s, i) + \bar{\gamma}F_c(\boldsymbol{x}, t)$
            $F(\boldsymbol{x}, t, s, i) \leftarrow \bar{\vartheta}F(\boldsymbol{x}, t, s, i)\|F(\boldsymbol{x}, t, s, i)\|^{-1}$
            $\gamma \leftarrow \gamma + T_\gamma^{-1}\Delta t$
            if $\gamma \geq 1$ then
                $state \leftarrow follow\_curve$
    $t \leftarrow t + \Delta t$
```

4
Control

In this chapter, we detail the development of the control strategies proposed in this work. The proposed strategies enable the UAV to follow the references given by the guidance strategy developed in Chapter 3.

4.1 Problem definition

First, let us formally define the control problem addressed in this work as:

Problem 2. Find an control law to make a fixed-wing UAV, modeled by (2.6)-(2.9), converge to the references provided by the solution of Problem 1, assuming that it exists.

In this chapter, we propose two solutions to Problem 2. First, we will use the linear velocities in the NED frame and the time derivative of the Euler angles as outputs of the system. With this approach, we intend to directly compare the vector field velocity references with the controlled outputs in the controller. In the second approach, we maintain the time derivative of the Euler angles but we exchange the linear velocities for the aerodynamic angles and true speed. This change allows us to impose linear constraints on the UAV's true speed and on the angle-of-attack.

The proposed solutions act in two stages: *i*) first, a Feedback Linearization controller is applied to the UAV model (2.6)-(2.9) using the generalized forces and moments as virtual control inputs; *ii*) after, the auxiliary control law is designed using a Model Predict Control to efficiently solve the reference tracking and the control mapping problems simultaneously, ensuring feasible transformations between the virtual control inputs and the applied ones.

4.2 Control using the linear velocities

In this section, we present the first control strategy. In this approach, we seek to control the UAV linear velocities in the NED frame and the Euler angles derivatives, as mentioned before. With that, we are able to compare the velocities references from the vector field strategy presented in Section 3.1 with the controlled outputs directly in the controller.

First, note that in (2.10)-(2.15) we can isolate the forces and moments that originated from the UAV control surfaces and the ones that originated from the UAV fixed surfaces, obtaining

$$\begin{aligned} f_x &= f_{x\boldsymbol{u}} + f_{x\boldsymbol{s}}, \\ &\vdots \\ n &= n_{\boldsymbol{u}} + n_{\boldsymbol{s}}, \end{aligned}$$

where the subscripts $\boldsymbol{u}$ and $\boldsymbol{s}$ indicate where the forces or moments are originated from the control surfaces or from the fixed surfaces, respectively. For the ones that originated from the control surfaces, we can summarize them as

$$f_{x\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S C_{X\delta_e}\delta_e + \frac{1}{2}\rho S_{prop} C_{prop} k_{motor}^2 \delta_t^2, \tag{4.1}$$

$$f_{y\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S \left(C_{Y\delta_a}\delta_a + C_{Y\delta_r}\delta_r\right), \tag{4.2}$$

$$f_{z\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S C_{Z\delta_e}\delta_e, \tag{4.3}$$

$$l_{\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S b \left(C_{l\delta_a}\delta_a + C_{l\delta_r}\delta_r\right) - k_{Tp} k_\omega^2 \delta_t^2, \tag{4.4}$$

$$m_{\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S c C_{m\delta_e}\delta_e, \tag{4.5}$$

$$n_{\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S b \left(C_{n\delta_a}\delta_a + C_{n\delta_r}\delta_r\right). \tag{4.6}$$

Note that both (4.1) and (4.4) are nonlinear with respect to δ_t. However, we can linearize those equations around a given input $\hat{\delta}_t$ by applying the following transformation

$$\delta_t^2 = \hat{\delta}_t^2 + 2\hat{\delta}_t(\delta_t - \hat{\delta}_t) = 2\hat{\delta}_t\delta_t - \hat{\delta}_t^2. \tag{4.7}$$

This way, by replacing (4.7) in (4.1) and (4.4) we obtain

$$f_{x\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S C_{X\delta_e}\delta_e + \rho S_{prop} C_{prop} k_{motor}^2 \hat{\delta}_t \delta_t - \frac{1}{2}\rho S_{prop} C_{prop} k_{motor}^2 \hat{\delta}_t^2, \tag{4.8}$$

$$l_{\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S b \left(C_{l\delta_a}\delta_a + C_{l\delta_r}\delta_r\right) - 2k_{Tp}k_\omega^2 \hat{\delta}_t \delta_t + k_{Tp}k_\omega^2 \hat{\delta}_t^2. \tag{4.9}$$

Now, note that the terms multiplying $\hat{\delta}_t^2$ in (4.8) and (4.9) are constant. Therefore, since we are interested only in the terms multiplying the throttle input, we can move these terms to $f_{x\boldsymbol{s}}$ and $l_{\boldsymbol{s}}$, obtaining

$$f_{x\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S C_{X\delta_e}\delta_e + \rho S_{prop} C_{prop} k_{motor}^2 \hat{\delta}_t \delta_t, \tag{4.10}$$

$$l_{\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S b \left(C_{l\delta_a}\delta_a + C_{l\delta_r}\delta_r\right) - 2k_{Tp}k_\omega^2 \hat{\delta}_t \delta_t. \tag{4.11}$$

This way, we are able to obtain a relationship between the virtual control inputs and the applied ones, given by

$$\bar{\boldsymbol{u}} = \boldsymbol{M}_{\boldsymbol{u}}\boldsymbol{u}, \tag{4.12}$$

with

$$\boldsymbol{M}_{\boldsymbol{u}} = \frac{1}{2}\rho v_t^2 S \begin{bmatrix} 0 & C_{X\delta_e} & 0 & \frac{S_{prop}C_{prop}}{v_t^2 S}k_{motor}^2\hat{\delta}_t \\ C_{Y\delta_a} & 0 & C_{Y\delta_r} & 0 \\ 0 & C_{Z\delta_e} & 0 & 0 \\ bC_{l\delta_a} & 0 & bC_{l\delta_r} & -\frac{4k_{Tp}k_\omega^2\hat{\delta}_t}{\rho v_t^2 S} \\ 0 & cC_{m\delta_e} & 0 & 0 \\ bC_{n\delta_a} & 0 & bC_{n\delta_r} & 0 \end{bmatrix}.$$

$$\bar{\boldsymbol{u}} = \begin{bmatrix} f_{x\boldsymbol{u}} & f_{y\boldsymbol{u}} & f_{z\boldsymbol{u}} & m_{\boldsymbol{u}} & l_{\boldsymbol{u}} & n_{\boldsymbol{u}} \end{bmatrix}^T,$$

$$\boldsymbol{u} = \begin{bmatrix} \delta_a & \delta_e & \delta_r & \delta_t \end{bmatrix}^T,$$

Note that $\boldsymbol{M}_{\boldsymbol{u}}$ contains the aerodynamic coefficients of the control surfaces. With (4.12), we are able to map restrictions of the applied inputs into the virtual ones.

Now, in order to apply a linear MPC to track the desired references, initially, we will use the input-output linearization technique presented in Chapter 2.2 with the generalized forces and moments vector $\bar{\boldsymbol{u}}$ as virtual control inputs. Furthermore, we define the state vector $\boldsymbol{x}$ and the output vector $\boldsymbol{y}$ as follows

$$\boldsymbol{x} = \begin{bmatrix} u_r & v_r & w_r & \phi & \theta & \psi & p & q & r \end{bmatrix}^T,$$

$$\boldsymbol{y} = \begin{bmatrix} \dot{p}_n & \dot{p}_e & \dot{p}_d & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T.$$

From (2.7)-(2.9) it is straightforward to see that at least one virtual control input appears in the first derivative of each output y_k, thus $\rho_k = 1$, $\forall k \in \{1, ..., 6\}$, and $\rho = 6$. This way, by applying the feedback linearization technique presented in Chapter 2, we can use the control law

$$\bar{\boldsymbol{u}} = \boldsymbol{\lambda}(\boldsymbol{x}) + \boldsymbol{\Lambda}(\boldsymbol{x})^{-1}\boldsymbol{\nu}, \tag{4.13}$$

to obtain a partially linearized system in the following normal form

$$\begin{cases} \dot{\boldsymbol{\eta}} = f_0(\boldsymbol{\eta}, \boldsymbol{\xi}) \\ \dot{\boldsymbol{\xi}} = \boldsymbol{A}\boldsymbol{\xi} + \boldsymbol{B}\boldsymbol{\nu} \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{\xi} \end{cases}, \tag{4.14}$$

with

$$\begin{aligned} &\boldsymbol{\eta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T, \\ &\boldsymbol{\xi} = \begin{bmatrix} \dot{p}_n & \dot{p}_e & \dot{p}_d & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T, \\ &\boldsymbol{A} = \boldsymbol{0},\ \boldsymbol{B} = \mathbb{I},\ \boldsymbol{C} = \mathbb{I}, \end{aligned}$$

where $\boldsymbol{0}$ and $\mathbb{I}$ are appropriate sized zero and identity matrices. Note that here we do not include the UAV position in the dynamics, as it is controlled in an external loop by the vector field strategy.

Furthermore, $\boldsymbol{\Lambda}$ and $\boldsymbol{\lambda}$ are given by

$$\boldsymbol{\Lambda} = \begin{bmatrix} L_{g1}h_1(\boldsymbol{x}) & \cdots & L_{g6}h_1(\boldsymbol{x}) \\ \vdots & \ddots & \vdots \\ L_{g1}h_6(\boldsymbol{x}) & \cdots & L_{g6}h_6(\boldsymbol{x}) \end{bmatrix}, \quad \boldsymbol{\lambda} = -\boldsymbol{\Lambda}^{-1} \begin{bmatrix} L_f h_1(\boldsymbol{x}) \\ \vdots \\ L_f h_6(\boldsymbol{x}) \end{bmatrix}. \tag{4.15}$$

For the complete version of these matrices please refer to Appendix B.1.

By computing $\boldsymbol{\Lambda}$ in (4.15) we find that $\det(\boldsymbol{\Lambda}) \neq 0$ if, and only if, $\cos(\theta) \neq 0$, thus implying $\theta \neq \pm 0.5\pi$. Therefore, for the proposed feedback linearization control law to be feasible, we first need to ensure that: *i)* the pitch angle θ will be such that $|\theta| < \frac{\pi}{2}$; and *ii)* the mapping between the virtual control inputs and the applied ones is invertible, and therefore, feasible. To solve both problems, besides the tracking problem, we propose to use a linear Model Predictive Control.

Initially, let k be the discrete sampling instant. Then, by discretizing the external dynamics of (4.14) using the Euler method, we obtain

$$\begin{cases} \boldsymbol{\xi}^+ = \boldsymbol{A}_d\boldsymbol{\xi}_k + \boldsymbol{B}_d\boldsymbol{\nu}_k \\ \boldsymbol{y}_k = \boldsymbol{C}\boldsymbol{\xi}_k \end{cases}, \tag{4.16}$$

where $\boldsymbol{\xi}^+ = \boldsymbol{\xi}_{k+1}$. Now, by augmenting (4.16) using the control increments $\Delta\boldsymbol{\nu}_k = \boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}$

as inputs, yields

$$\begin{cases} \underbrace{\begin{bmatrix} \boldsymbol{\xi}^{+} \\ \boldsymbol{\nu}^{+} \end{bmatrix}}_{\boldsymbol{z}^{+}} = \underbrace{\begin{bmatrix} \boldsymbol{A}_d & \boldsymbol{B}_d \\ \boldsymbol{0} & \mathbb{I} \end{bmatrix}}_{\mathcal{A}} \underbrace{\begin{bmatrix} \boldsymbol{\xi}_k \\ \boldsymbol{\nu}_{k-1} \end{bmatrix}}_{\boldsymbol{z}} + \underbrace{\begin{bmatrix} \boldsymbol{B}_d \\ \mathbb{I} \end{bmatrix}}_{\mathcal{B}} \Delta\boldsymbol{\nu}_k, \\ \boldsymbol{y}_k = \underbrace{\begin{bmatrix} \boldsymbol{C} & \boldsymbol{0} \end{bmatrix}}_{\mathcal{C}} \boldsymbol{z}. \end{cases} \tag{4.17}$$

This Incremental MPC Framework is discussed in Borrelli et al. (2017); Rossiter (2003).

In order to incorporate the control allocation problem into the optimal control problem of the MPC, the virtual and applied control inputs should also be discretized. For that, consider that $\boldsymbol{\lambda}_0 = \boldsymbol{\lambda}|_{k=0}$ and $\boldsymbol{\Lambda}_0 = \boldsymbol{\Lambda}|_{k=0}$, with $\boldsymbol{\lambda}$ and $\boldsymbol{\Lambda}$ from (4.15), are kept constant during the entire prediction horizon. This way, with (4.12) and (4.13), we can define

$$\begin{aligned} \bar{\boldsymbol{u}}_k &= \boldsymbol{\lambda}_0 + \boldsymbol{\Lambda}_0^{-1}\boldsymbol{\nu}_k, \\ \Delta\bar{\boldsymbol{u}}_k &= \bar{\boldsymbol{u}}_k - \bar{\boldsymbol{u}}_{k-1} = \boldsymbol{\lambda}_0 - \boldsymbol{\lambda}_0 + \boldsymbol{\Lambda}_0^{-1}\left(\boldsymbol{\nu}_k - \boldsymbol{\nu}_{k-1}\right) = \boldsymbol{\Lambda}_0^{-1}\Delta\boldsymbol{\nu}_k, \\ \Delta\boldsymbol{u}_k &= \boldsymbol{u}_k - \boldsymbol{u}_{k-1} = \boldsymbol{M}_{\boldsymbol{u}}^{+}\left(\bar{\boldsymbol{u}}_k - \bar{\boldsymbol{u}}_{k-1}\right) = \boldsymbol{M}_{\boldsymbol{u}}^{+}\Delta\bar{\boldsymbol{u}}_k, \end{aligned}$$

where $\boldsymbol{M}_{\boldsymbol{u}}^{+}$ is the left pseudo-inverse of $\boldsymbol{M}_{\boldsymbol{u}}$, also computed at $k = 0$ and kept constant. It should be noted that by by doing that, we also avoid nonlinear constraints. Now, suppose that for a given input $\Delta\boldsymbol{\nu}_k$, there is an increment in the applied controls given by

$$\Delta\boldsymbol{u}_k = \boldsymbol{M}_{\boldsymbol{u}}^{+}\Delta\bar{\boldsymbol{u}}_k = \boldsymbol{M}_{\boldsymbol{u}}^{+}\boldsymbol{\Lambda}_0^{-1}\Delta\boldsymbol{\nu}_k. \tag{4.18}$$

Since $\boldsymbol{M}_{\boldsymbol{u}}^{+}$ only gives the minimum squared error for $\Delta\boldsymbol{u}_k$, the actual incremental inputs applied are

$$\Delta\boldsymbol{\nu}_k' = \boldsymbol{\Lambda}_0\boldsymbol{M}_{\boldsymbol{u}}\Delta\boldsymbol{u}_k = \boldsymbol{\Lambda}_0\boldsymbol{M}_{\boldsymbol{u}}\boldsymbol{M}_{\boldsymbol{u}}^{+}\boldsymbol{\Lambda}_0^{-1}\Delta\boldsymbol{\nu}_k. \tag{4.19}$$

However, the valid solution must ensure that $\Delta\boldsymbol{\nu}_k' = \Delta\boldsymbol{\nu}_k$. For that, we can reorganize (4.19) to obtain

$$(\mathbb{I} - \boldsymbol{M}_{\boldsymbol{u}}\boldsymbol{M}_{\boldsymbol{u}}^{+})\boldsymbol{\Lambda}_0^{-1}\Delta\boldsymbol{\nu}_k = 0. \tag{4.20}$$

Therefore, as long as $\Delta\boldsymbol{\nu}_k$ is in the null space of $(\mathbb{I} - \boldsymbol{M}_{\boldsymbol{u}}\boldsymbol{M}_{\boldsymbol{u}}^{+})\boldsymbol{\Lambda}_0^{-1}$, there is a feasible mapping between the virtual inputs $\bar{\boldsymbol{u}}$ and the applied ones $\boldsymbol{u}$. Thus, we can formally define the control mapping restriction as

$$(\mathbb{I} - \boldsymbol{M}_{\boldsymbol{u}}\boldsymbol{M}_{\boldsymbol{u}}^{+})\boldsymbol{\Lambda}_0^{-1}\Delta\boldsymbol{\nu}_k = 0, \quad \forall k \in \{0, ..., n_u\}, \tag{4.21}$$

where n_u is the control horizon.

Furthermore, given that the control surfaces have physical and rating limits, we need to impose constraints so that $\Delta\boldsymbol{\nu}_k$ will respect those. For that, let $\boldsymbol{u}_0$ be the value of $\boldsymbol{u}$ at

$k = 0$. Then, we can use the relationship found in (4.18) to obtain

$$\Delta \boldsymbol{u}_{min} \leq \boldsymbol{M}_{\boldsymbol{u}}^{+} \boldsymbol{\Lambda}_0^{-1} \Delta \boldsymbol{\nu}_k \leq \Delta \boldsymbol{u}_{max}, \qquad \forall k \in \{0, ..., n_u\}, \tag{4.22}$$

and

$$\begin{cases} \boldsymbol{u}_{min} \leq \boldsymbol{u}_0 + \boldsymbol{M}_{\boldsymbol{u}}^{+} \boldsymbol{\Lambda}_0^{-1} \sum_{j=0}^{k} \Delta \boldsymbol{\nu}_j \leq \boldsymbol{u}_{max}, & \forall k \in \{0, ..., n_u\}, \\ \boldsymbol{u}_{min} \leq \boldsymbol{u}_0 + \boldsymbol{M}_{\boldsymbol{u}}^{+} \boldsymbol{\Lambda}_0^{-1} \left(\sum_{j=0}^{nu} \Delta \boldsymbol{\nu}_j + (k - n_u) \Delta \boldsymbol{\nu}_{nu} \right) \leq \boldsymbol{u}_{max}, & \forall k \in \{n_u + 1, ..., n_p\}, \end{cases} \tag{4.23}$$

where n_p is the prediction horizon, $\boldsymbol{u}_{max}$ and $\boldsymbol{u}_{min}$ are the upper and lower bounds of $\boldsymbol{u}$, and $\Delta \boldsymbol{u}_{max}$ and $\Delta \boldsymbol{u}_{min}$ are the rating bounds. Thus, given that (4.22) and (4.23) are met, then $\Delta \boldsymbol{\nu}_k$ will respect the control surfaces amplitude limits and rating limits.

Moreover, we need to ensure that the feedback linearization control law is always feasible. For that, we need to ensure that $|\theta| < \frac{\pi}{2}$. Assume for now that θ can be measured, and it is available to the controller at any instant k. Now, given that $\theta_{k+1} = \theta_k + \dot{\theta}_k \Delta t$, where Δt is the sampling time, we can write the following constraint on $\dot{\theta}$

$$-\frac{\pi}{2} < \theta_0 + \sum_{j=1}^{k} \dot{\theta}_j \Delta t < \frac{\pi}{2}, \ \forall k \in \{1, ..., n_p\}. \tag{4.24}$$

One should note that a strict restriction cannot be solved by using numeric methods, therefore, it must be approximated as follows

$$-\frac{\pi}{2 + \epsilon} \leq \theta_0 + \sum_{j=1}^{k} \dot{\theta}_j \Delta t \leq \frac{\pi}{2 + \epsilon}, \ \forall k \in \{1, ..., n_p\}, \tag{4.25}$$

where ϵ is a small, positive constant.

Furthermore, the UAV must fly above a minimum speed, denoted v_{stall}. Thus, consider the following constraint

$$\dot{p}_{n_k}^2 + \dot{p}_{e_k}^2 + \dot{p}_{d_k}^2 \geq v_{stall}^2. \tag{4.26}$$

Note that (4.26) is a nonlinear restriction. Therefore, we modify it in order to use a linear constraint in the MPC formulation. For that, consider that the UAV speed at time $k + i$ is given by

$$\dot{p}_{n_{k+i}} = \dot{p}_{n_k} + \Delta t \sum_{j=0}^{i-1} \ddot{p}_{n_{k+j}},$$

$$\dot{p}_{e_{k+i}} = \dot{p}_{e_k} + \Delta t \sum_{j=0}^{i-1} \ddot{p}_{e_{k+j}},$$

$$\dot{p}_{d_{k+i}} = \dot{p}_{d_k} + \Delta t \sum_{j=0}^{i-1} \ddot{p}_{d_{k+j}}.$$

Replacing the above equations in (4.26) we obtain

$$\dot{p}_{n_k}^2 + \dot{p}_{e_k}^2 + \dot{p}_{d_k}^2 + \Delta t^2 \left[\left(\sum_{j=0}^{i-1} \ddot{p}_{n_j} \right)^2 + \left(\sum_{j=0}^{i-1} \ddot{p}_{e_j} \right)^2 + \left(\sum_{j=0}^{i-1} \ddot{p}_{d_j} \right)^2 \right] + \\ 2\Delta t \sum_{j=0}^{i-1} \left(\ddot{p}_{n_{k+j}} \dot{p}_{n_k} + \ddot{p}_{e_{k+j}} \dot{p}_{e_k} + \ddot{p}_{d_{k+j}} \dot{p}_{d_k} \right) \geq v_{stall}^2, \ \forall i \in \{1, ..., n_p\}. \tag{4.27}$$

Now, since the term multiplying Δt^2 in (4.27) is positive, we can remove it to obtain

$$\vartheta_0 + 2\Delta t \sum_{j=0}^{i-1} \left(\ddot{p}_{n_{k+j}} \dot{p}_{n_k} + \ddot{p}_{e_{k+j}} \dot{p}_{e_k} + \ddot{p}_{d_{k+j}} \dot{p}_{d_k} \right) \geq v_{stall}^2, \ \forall i \in \{1, ..., n_p\}, \tag{4.28}$$

where

$$\vartheta_0 = \dot{p}_{n_k}^2 + \dot{p}_{e_k}^2 + \dot{p}_{d_k}^2.$$

Since $\dot{p}_{n_k}$, $\dot{p}_{e_k}$ and $\dot{p}_{d_k}$ are known, (4.28) is a convex, linear constraint. Moreover, from (4.17), we have that $\ddot{p}_{n_k}$, $\ddot{p}_{e_k}$ and $\ddot{p}_{d_k}$ can be replaced by the respective desired accelerations, composing the first three elements of $\boldsymbol{\nu}_k$. Finally, the solution of (4.28) is also a solution to (4.27). However, it should be noted that by using this conservative constraint, the feasible solution set can be drastically reduced. This concludes the formulation of the constraints for the controller.

Now, let $\boldsymbol{y}_r$ be a given desired reference, yet to be defined. By defining the tracking error as $\bar{\boldsymbol{y}} = \boldsymbol{y}_r - \boldsymbol{y} = \boldsymbol{y}_r - \boldsymbol{C}\boldsymbol{z}$ we can define the cost functional as

$$J(\Delta\boldsymbol{\nu}, \bar{\boldsymbol{y}}) = \min_{\Delta\nu} \sum_{i=0}^{nu} \Delta\boldsymbol{\nu}_i^T \boldsymbol{R} \Delta\boldsymbol{\nu}_i + \sum_{j=0}^{np-1} \bar{\boldsymbol{y}}_j^T \boldsymbol{Q} \bar{\boldsymbol{y}}_j + \bar{\boldsymbol{y}}_{np}^T \boldsymbol{P} \bar{\boldsymbol{y}}_{np}, \tag{4.29}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are the control and output weighting matrices, and $\boldsymbol{P}$ is the terminal cost. Note that $\boldsymbol{y}_k$ can be written as an explicit function of the initial state $\boldsymbol{z}_0$ and the future inputs $\Delta\boldsymbol{\nu}_0$, $\Delta\boldsymbol{\nu}_1$, ..., $\Delta\boldsymbol{\nu}_{n_u-1}$, as follows

$$\boldsymbol{y}_k = \mathcal{C}\mathcal{A}^k \boldsymbol{z}_0 + \mathcal{C} \sum_{i=0}^{k-1} \mathcal{A}^i \mathcal{B} \Delta\boldsymbol{\nu}_i. \tag{4.30}$$

Therefore, by defining n_u and n_p, we follow a similar approach to the one presented in Chapter 2.3 to write (4.30) in the compact form

$$\vec{\boldsymbol{y}} = \boldsymbol{T}\boldsymbol{z}_o + \boldsymbol{S}\Delta\vec{\boldsymbol{\nu}}, \tag{4.31}$$

with

$$
\begin{aligned}
&\vec{\boldsymbol{y}} = \begin{bmatrix} \boldsymbol{y}_1^T & \boldsymbol{y}_2^T & \cdots & \boldsymbol{y}_{np}^T \end{bmatrix}^T, \\
&\Delta\vec{\boldsymbol{\nu}} = \begin{bmatrix} \Delta\boldsymbol{\nu}_0^T & \Delta\boldsymbol{\nu}_1^T & \cdots & \Delta\boldsymbol{\nu}_{nu-1}^T \end{bmatrix}^T, \\
&\boldsymbol{T} = \begin{bmatrix} \mathcal{C}\mathcal{A} \\ \mathcal{C}\mathcal{A}^2 \\ \vdots \\ \mathcal{C}\mathcal{A}^{np} \end{bmatrix}, \quad \boldsymbol{S} = \begin{bmatrix} \mathcal{C}\mathcal{B} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathcal{C}\mathcal{A}\mathcal{B} & \mathcal{C}\mathcal{B} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{C}\mathcal{A}^{np-1}\mathcal{B} & \mathcal{C}\mathcal{A}^{np-2}\mathcal{B} & \cdots & \mathcal{C}\sum_{i=1}^{np-nu}\mathcal{A}^i\mathcal{B} \end{bmatrix}.
\end{aligned}
$$

Differently from the notation used in Chapter 2.3, here we are obtaining the future outputs, instead of future states. Moreover, the cost functional can now be rewritten as

$$
J(\Delta\vec{\boldsymbol{\nu}}, \boldsymbol{z}_0) = \Delta\vec{\boldsymbol{\nu}}^T \boldsymbol{H} \Delta\vec{\boldsymbol{\nu}} + \boldsymbol{F}^T \Delta\vec{\boldsymbol{\nu}} + \boldsymbol{Y}, \tag{4.32}
$$

with

$$
\begin{aligned}
\boldsymbol{H} &= \bar{\boldsymbol{R}} + \boldsymbol{S}^T \bar{\boldsymbol{Q}} \boldsymbol{S}, \\
\boldsymbol{F} &= 2\left(\boldsymbol{T}\boldsymbol{z}_0 - \vec{\boldsymbol{y}}_r\right)^T \bar{\boldsymbol{Q}} \boldsymbol{S}, \\
\boldsymbol{Y} &= (\vec{\boldsymbol{y}}_r - \boldsymbol{T}\boldsymbol{z}_0)^T \bar{\boldsymbol{Q}} (\vec{\boldsymbol{y}}_r - \boldsymbol{T}\boldsymbol{z}_0),
\end{aligned}
$$

where $\vec{\boldsymbol{y}}_r$ is the reference output vector, $\bar{\boldsymbol{Q}} = \text{blockdiag}\,\overbrace{(\boldsymbol{Q}, ..., \boldsymbol{Q}, \boldsymbol{P})}^{np}$ and $\bar{\boldsymbol{R}} = \text{blockdiag}\,\overbrace{(\boldsymbol{R}, ..., \boldsymbol{R})}^{nu}$. Thus, (4.32) is a quadratic convex function of $\Delta\vec{\boldsymbol{\nu}}$ and $\boldsymbol{z}_0$.

To write the constraint in the compact form, we can follow a similar procedure. Starting with the constraints on the pitch angle, let $\boldsymbol{z}_\theta$ and $\boldsymbol{Z}_\theta$ be defined as

$$
\begin{aligned}
\boldsymbol{z}_\theta &= \begin{bmatrix} 0 & 0 & 0 & 0 & \Delta t & 0 & \cdots & 0 \end{bmatrix}, \\
\boldsymbol{Z}_\theta &= \begin{bmatrix} \boldsymbol{z}_\theta & \mathbf{0} & \cdots & \mathbf{0} \\ \boldsymbol{z}_\theta & \boldsymbol{z}_\theta & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{z}_\theta & \boldsymbol{z}_\theta & \cdots & \boldsymbol{z}_\theta \end{bmatrix}.
\end{aligned} \tag{4.33}
$$

With that, we can rewrite (4.24) in the compact form

$$
\begin{bmatrix} \boldsymbol{Z}_\theta \boldsymbol{S} \\ -\boldsymbol{Z}_\theta \boldsymbol{S} \end{bmatrix} \Delta\vec{\boldsymbol{\nu}} \leq \begin{bmatrix} [\boldsymbol{\theta}_{max}]_{\times np} \\ -[\boldsymbol{\theta}_{min}]_{\times np} \end{bmatrix} + \begin{bmatrix} -[\boldsymbol{\theta}_0]_{\times np} \\ [\boldsymbol{\theta}_0]_{\times np} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{Z}_\theta \boldsymbol{T} \boldsymbol{z}_o \\ \boldsymbol{Z}_\theta \boldsymbol{T} \boldsymbol{z}_o \end{bmatrix}, \tag{4.34}
$$

with

$$
\theta_{max} = -\theta_{min} = \frac{\pi}{2+\epsilon}.
$$

Moreover, the operator $[\cdot]_{\times} n$ means that we stack the variable n times, for instance

$$[\boldsymbol{\theta}_0]_{\times np} = \begin{bmatrix} \overbrace{\theta_0 \quad \cdots \quad \theta_0}^{np} \end{bmatrix}^T.$$

Now, for the speed constraint (4.28), we have

$$-\boldsymbol{Z}_\vartheta \boldsymbol{S} \Delta\vec{\boldsymbol{\nu}} \le [\boldsymbol{\vartheta}_0]_{\times np} - [\boldsymbol{v}^2_{stall}]_{\times np} + \boldsymbol{Z}_\vartheta \boldsymbol{T} \boldsymbol{z}_0, \tag{4.35}$$

where

$$\boldsymbol{z}_\vartheta = 2\Delta t \begin{bmatrix} 0 & 0 & \cdots & \dot{p}_{n_k} & \dot{p}_{e_k} & \dot{p}_{d_k} & 0 & 0 & 0 \end{bmatrix},$$
$$\boldsymbol{Z}_\vartheta = \begin{bmatrix} \boldsymbol{z}_\vartheta & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{z}_\vartheta & \boldsymbol{z}_\vartheta & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{z}_\vartheta & \boldsymbol{z}_\vartheta & \cdots & \boldsymbol{z}_\vartheta \end{bmatrix}.$$

Similarly, (4.22) and (4.23) can be written in the compact form

$$\begin{bmatrix} \boldsymbol{U}_{rate} \\ -\boldsymbol{U}_{rate} \end{bmatrix} \Delta\vec{\boldsymbol{\nu}} \le \begin{bmatrix} [\Delta\boldsymbol{u}_{max}]_{\times nu} \\ -[\Delta\boldsymbol{u}_{min}]_{\times nu} \end{bmatrix}, \tag{4.36}$$

$$\begin{bmatrix} \boldsymbol{U}_{sum} \\ -\boldsymbol{U}_{sum} \end{bmatrix} \Delta\vec{\boldsymbol{\nu}} \le \begin{bmatrix} [\boldsymbol{u}_{max}]_{\times np} \\ -[\boldsymbol{u}_{min}]_{\times np} \end{bmatrix} + \begin{bmatrix} -[\boldsymbol{u}_0]_{\times np} \\ [\boldsymbol{u}_0]_{\times np} \end{bmatrix}, \tag{4.37}$$

where $\boldsymbol{U}_{rate}$ and $\boldsymbol{U}_{sum}$ are matrices such that

$$\boldsymbol{U}_{rate} = \begin{bmatrix} \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} \end{bmatrix},$$
$$\boldsymbol{U}_{sum} = \begin{bmatrix} \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \cdots & \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} \\ \vdots & \vdots & \vdots & \vdots \\ \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} & \cdots & (n_p - n_u + 1)\boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} \end{bmatrix}.$$

Since all the constraints and the cost functional are explicit functions of $\Delta\vec{\boldsymbol{\nu}}$, we can formally define the optimal control problem as

$$\begin{aligned} J(\Delta\vec{\boldsymbol{\nu}}) = \min_{\Delta\vec{\boldsymbol{\nu}}} \Delta\vec{\boldsymbol{\nu}}^T \boldsymbol{H} \Delta\vec{\boldsymbol{\nu}} + \boldsymbol{F}^T \Delta\vec{\boldsymbol{\nu}}, \\ s.t.\ \boldsymbol{A}_{eq} \Delta\vec{\boldsymbol{\nu}} = \boldsymbol{0}, \\ \boldsymbol{A}_{ineq1} \Delta\vec{\boldsymbol{\nu}} \leq \boldsymbol{b}_{ineq1}, \end{aligned} \tag{4.38}$$

where the matrices $\boldsymbol{A}_{eq}$ and $\boldsymbol{A}_{ineq1}$, and the vector $\boldsymbol{b}_{ineq1}$ are given by

$$\boldsymbol{A}_{eq} = \begin{bmatrix} \left(\mathbb{I} - \boldsymbol{M}_{\boldsymbol{u}} \boldsymbol{M}_{\boldsymbol{u}}^{+}\right) \boldsymbol{\Lambda}_0^{-1} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \left(\mathbb{I} - \boldsymbol{M}_{\boldsymbol{u}} \boldsymbol{M}_{\boldsymbol{u}}^{+}\right) \boldsymbol{\Lambda}_0^{-1} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \cdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \left(\mathbb{I} - \boldsymbol{M}_{\boldsymbol{u}} \boldsymbol{M}_{\boldsymbol{u}}^{+}\right) \boldsymbol{\Lambda}_0^{-1} \end{bmatrix},$$

$$\boldsymbol{A}_{ineq1} = \begin{bmatrix} \boldsymbol{Z}_{\theta} \boldsymbol{S} \\ -\boldsymbol{Z}_{\theta} \boldsymbol{S} \\ -\boldsymbol{Z}_{\vartheta} \boldsymbol{S} \\ \boldsymbol{U}_{rate} \\ -\boldsymbol{U}_{rate} \\ \boldsymbol{U}_{sum} \\ -\boldsymbol{U}_{sum} \end{bmatrix}, \quad \boldsymbol{b}_{ineq1} = \begin{bmatrix} \vec{\theta}_{max} - \vec{\theta}_0 - \boldsymbol{Z}_{\theta} \boldsymbol{T} \boldsymbol{z}_o \\ -\vec{\theta}_{min} + \vec{\theta}_0 + \boldsymbol{Z}_{\theta} \boldsymbol{T} \boldsymbol{z}_o \\ \vec{\vartheta}_0 - \vec{v}_{stall}^{\,2} + \boldsymbol{Z}_{\vartheta} \boldsymbol{T} \boldsymbol{z}_0 \\ \Delta\vec{\boldsymbol{u}}_{max} \\ -\Delta\vec{\boldsymbol{u}}_{min} \\ \vec{\boldsymbol{u}}_{max} - \vec{\boldsymbol{u}}_0 \\ -\vec{\boldsymbol{u}}_{min} + \vec{\boldsymbol{u}}_0 \end{bmatrix}.$$

Moreover, the OCP has $(6 \times n_u)$ equality constraints, and $(11 \times n_p + 4 \times n_u)$ inequality constraints. Note that in (4.38) we do not include the term $\boldsymbol{Y}$ from (4.32), as it does not contributes to the solution of the problem. Therefore, given the optimal control sequence $\Delta\vec{\boldsymbol{\nu}}^*$, resulting from (4.38), the applied control inputs will be given by

$$\boldsymbol{u} = \boldsymbol{u}_0 + \boldsymbol{M}_{\boldsymbol{u}}^{+} \boldsymbol{\Lambda}_0^{-1} \Delta\boldsymbol{\nu}_0^*. \tag{4.39}$$

4.3 Control using the aerodynamic states

In this section, we present an alternative approach to solve Problem 2, using the aerodynamics angles and the UAV's true speed in place of the linear velocities expressed in the NED frame. This will allow us to replace the conservative constraints (4.28) with more direct ones. First, let the derivative of those new states be given by

$$\dot{v}_t = \frac{u_r \dot{u}_r + v_r \dot{v}_r + w_r \dot{w}_r}{v_t}, \tag{4.40}$$

$$\dot{\alpha} = \frac{u_r \dot{w}_r - w_r \dot{u}_r}{u_r^2 + w_r^2}, \tag{4.41}$$

$$\dot{\beta} = \frac{v_t \dot{v}_r - v_r \dot{v}_t}{v_t \sqrt{u_r^2 + w_r^2}}, \tag{4.42}$$

in which $\dot{u}_r$, $\dot{v}_r$, $\dot{w}_r$ and $\dot{v}_t$ are obtained from (2.7). Therefore, by choosing

$$\boldsymbol{y} = \begin{bmatrix} v_t & \alpha & \beta & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T,$$

and applying the same feedback linearization procedure as in the first strategy, we obtain a partially linearized system in the form of (4.14) with

$$\begin{aligned} \boldsymbol{\eta} &= \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T, \\ \boldsymbol{\xi} &= \begin{bmatrix} v_t & \alpha & \beta & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T, \\ \boldsymbol{A} &= \boldsymbol{0},\ \boldsymbol{B} = \mathbb{I},\ \boldsymbol{C} = \mathbb{I}. \end{aligned}$$

With (4.40)-(4.42) we find that $\rho_k = 1\ \forall k \in 1, ..., 6$ and $\rho = 6$. Therefore, $\boldsymbol{\lambda}$ and $\boldsymbol{\Lambda}$ are given in the same form as in (4.15). However, in this case, by computing $\boldsymbol{\Lambda}$ we find out that

$$\det(\boldsymbol{\Lambda}) \neq 0 \iff \cos(\theta) \neq 0,\ v_t^2 > 0,\ \cos(\beta) \neq 0. \tag{4.43}$$

Therefore, we need two additional conditions in the MPC formulation. Note that $v_t^2 > 0$ can be replaced with $v_t > v_{stall}$ in (4.43) without loss of generality, given that $v_{stall} > 0$. This way, we will be limiting the UAV velocity by its stall speed. Therefore, we can directly set the restrictions on v_t as

$$-\boldsymbol{Z}_{v_t}\boldsymbol{S}\Delta\vec{\boldsymbol{\nu}} \leq -\left[\boldsymbol{v}_{stall}\right]_{\times n_p} + \boldsymbol{Z}_{v_t}\boldsymbol{T}\boldsymbol{z}_0, \tag{4.44}$$

with $\boldsymbol{S}$ and $\boldsymbol{T}$ given by (4.31), and

$$\begin{aligned} \boldsymbol{z}_{v_t} &= \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}, \\ \boldsymbol{Z}_{v_t} &= \begin{bmatrix} \boldsymbol{z}_{v_t} & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{z}_{v_t} & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{z}_{v_t} \end{bmatrix}. \end{aligned}$$

Then, we can remove (4.28) from the MPC problem. As for the restriction $\cos(\beta) \neq 0$ we can set

$$\begin{bmatrix} \boldsymbol{Z}_\beta\boldsymbol{S} \\ -\boldsymbol{Z}_\beta\boldsymbol{S} \end{bmatrix} \Delta\vec{\boldsymbol{\nu}} \leq \begin{bmatrix} \left[\boldsymbol{\beta}_{max}\right]_{\times n_p} \\ -\left[\boldsymbol{\beta}_{min}\right]_{\times n_p} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{Z}_\beta\boldsymbol{T}\boldsymbol{z}_0 \\ \boldsymbol{Z}_\beta\boldsymbol{T}\boldsymbol{z}_0 \end{bmatrix}, \tag{4.45}$$

where

$$\boldsymbol{z}_\beta = \begin{bmatrix} 0 & 0 & 1 & \cdots & 0 \end{bmatrix},$$
$$\boldsymbol{Z}_\beta = \begin{bmatrix} \boldsymbol{z}_\beta & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{z}_\beta & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{z}_\beta \end{bmatrix}.$$

Moreover, since we are controlling α, we can impose constraints on it in order to avoid stall situations, which are given by

$$\begin{bmatrix} \boldsymbol{Z}_\alpha \boldsymbol{S} \\ -\boldsymbol{Z}_\alpha \boldsymbol{S} \end{bmatrix} \Delta\vec{\nu} \leq \begin{bmatrix} [\boldsymbol{\alpha}_{max}]_{\times np} \\ -[\boldsymbol{\alpha}_{min}]_{\times np} \end{bmatrix} + \begin{bmatrix} -\boldsymbol{Z}_\alpha \boldsymbol{T} \boldsymbol{z}_0 \\ \boldsymbol{Z}_\alpha \boldsymbol{T} \boldsymbol{z}_0 \end{bmatrix}, \tag{4.46}$$

where α_{min} and α_{max} are constructive parameters, and $\boldsymbol{Z}_\alpha$ is given by

$$\boldsymbol{z}_\alpha = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \end{bmatrix},$$
$$\boldsymbol{Z}_\alpha = \begin{bmatrix} \boldsymbol{z}_\alpha & \boldsymbol{0} & \cdots & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{z}_\alpha & \cdots & \boldsymbol{0} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{0} & \boldsymbol{0} & \cdots & \boldsymbol{z}_\alpha \end{bmatrix}.$$

Thus, the optimal control problem is posed as

$$\begin{aligned} J(\Delta\vec{\boldsymbol{\nu}}) = \min_{\Delta\vec{\boldsymbol{\nu}}} & \Delta\vec{\boldsymbol{\nu}}^T \boldsymbol{H} \Delta\vec{\boldsymbol{\nu}} + \boldsymbol{F}^T \Delta\vec{\boldsymbol{\nu}} \\ s.t. \ & \boldsymbol{A}_{eq} \Delta\vec{\boldsymbol{\nu}} = \boldsymbol{0} \\ & \boldsymbol{A}_{ineq2} \Delta\vec{\boldsymbol{\nu}} \leq \boldsymbol{b}_{ineq2}, \end{aligned} \tag{4.47}$$

where

$$\boldsymbol{A}_{ineq2} = \begin{bmatrix} \boldsymbol{Z}_\theta \boldsymbol{S} \\ -\boldsymbol{Z}_\theta \boldsymbol{S} \\ -\boldsymbol{Z}_{v_t} \boldsymbol{S} \\ \boldsymbol{Z}_\beta \boldsymbol{S} \\ -\boldsymbol{Z}_\beta \boldsymbol{S} \\ \boldsymbol{Z}_\alpha \boldsymbol{S} \\ -\boldsymbol{Z}_\alpha \boldsymbol{S} \\ \boldsymbol{U}_{rate} \\ -\boldsymbol{U}_{rate} \\ \boldsymbol{U}_{sum} \\ -\boldsymbol{U}_{sum} \end{bmatrix}, \quad \boldsymbol{b}_{ineq2} = \begin{bmatrix} \vec{\theta}_{max} - \vec{\theta}_0 - \boldsymbol{Z}_\theta \boldsymbol{T} \boldsymbol{z}_o \\ -\vec{\theta}_{min} + \vec{\theta}_0 + \boldsymbol{Z}_\theta \boldsymbol{T} \boldsymbol{z}_o \\ -\vec{v}_{stall} + \boldsymbol{Z}_{v_t} \boldsymbol{T} \boldsymbol{z}_0 \\ \vec{\beta}_{max} - \boldsymbol{Z}_\beta \boldsymbol{T} \boldsymbol{z}_0 \\ -\vec{\beta}_{min} + \boldsymbol{Z}_\beta \boldsymbol{T} \boldsymbol{z}_0 \\ \vec{\alpha}_{max} - \boldsymbol{Z}_\alpha \boldsymbol{T} \boldsymbol{z}_0 \\ -\vec{\alpha}_{min} + \boldsymbol{Z}_\alpha \boldsymbol{T} \boldsymbol{z}_0 \\ \Delta \vec{\boldsymbol{u}}_{max} \\ -\Delta \vec{\boldsymbol{u}}_{min} \\ \vec{\boldsymbol{u}}_{max} - \vec{\boldsymbol{u}}_0 \\ -\vec{\boldsymbol{u}}_{min} + \vec{\boldsymbol{u}}_0 \end{bmatrix}.$$

Note that here we only change the inequality matrices, while $\boldsymbol{A}_{eq}$ is the same as in the first strategy. Moreover, the OCP has $(6 \times n_u)$ equality constraints, and $(15 \times n_p + 4 \times n_u)$ inequality constraints.

Similarly to the first controller, given the optimal control sequence $\Delta \vec{\boldsymbol{\nu}}^*$, resulting from (4.47), the applied control inputs will be given by

$$\boldsymbol{u} = \boldsymbol{u}_0 + \boldsymbol{M}_{\boldsymbol{u}}^+ \boldsymbol{\Lambda}_0^{-1} \Delta \boldsymbol{\nu}_0^*. \tag{4.48}$$

4.4 Internal dynamics

For both presented controllers, the chosen output vectors lead to a relative degree less than the number of states. In this case, the input-output map is linearized, but the state equations are only partially linearized. Because of that, we are not able to eliminate the internal dynamics, which, for both strategies, is represented by the state vector

$$\boldsymbol{\eta} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}.$$

As $\boldsymbol{\eta}$ is not observable from the controller, we need to analyze it in order to know its behavior. From Khalil & Grizzle (2002), we have that if $\dot{\boldsymbol{\eta}}$ has an asymptotically stable equilibrium point, then the system is said to be minimum phase.

Thus, assuming that the proposed MPC controllers are able to stabilize the external dynamics, then $\dot{\phi}$, $\dot{\theta}$ and $\dot{\psi}$ converge to zero. Consequently, we have $\dot{\boldsymbol{\eta}} = f_0(\boldsymbol{\eta}, \boldsymbol{0}) = \boldsymbol{0}$ and the system is minimum phase for both cases.

4.5 Reference trajectory

With the control strategy defined, we can now formulate our reference trajectory $\boldsymbol{y}^r$. From Chapter 3, the vector field $F(\boldsymbol{x}, t, s, i)$ is a three-element vector, in which each element contains the reference speed in the x_1, x_2 and x_3 axis, respectively. Let us denote this vector by $\boldsymbol{F} = [F_{x_1} \quad F_{x_2} \quad F_{x_3}]^T$.

Therefore, we can write $\boldsymbol{F}$ in the NED reference frame as

$$\boldsymbol{F}_{NED} = \begin{bmatrix} F_{x_1} & F_{x_2} & F_{x_3} \end{bmatrix}^T, \tag{4.49}$$

and use it as reference to $\dot{p}_n$, $\dot{p}_e$ and $\dot{p}_d$.

Furthermore, $\boldsymbol{F}$ can also be used as a reference to v_t. For that, we replace (4.49) in the left side of (2.6) with (2.16), (2.17) and (2.18) to obtain

$$v_t^r = \|\boldsymbol{F}_{NED} - \mathbf{w}_v\|. \tag{4.50}$$

As for the Euler angles, it is desired that the UAV is aligned with the field. For that, let $F_\psi = \text{arctan2}(F_{x_2}, F_{x_1})$ be the field orientation in the x_1x_2 plane. Thus, we define the reference to $\dot{\psi}$ as a first order error dynamics, given by

$$\dot{\psi}^r = K_{1,\psi}\left(F_\psi - \psi\right) + K_{2,\psi}\dot{F}_\psi, \tag{4.51}$$

with $K_{1,\psi}$ and $K_{2,\psi} > 0$.

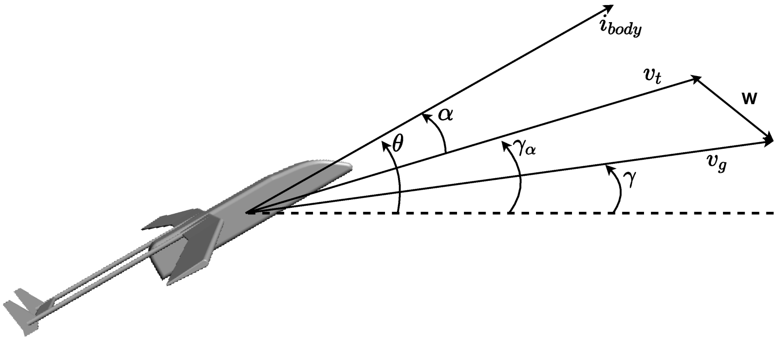

Figure 4.1: Wind triangle projected in the vertical plane

Moving on, Figure 4.1 shows the wind triangle projected on the vertical plane, where γ is the flight-path angle, γ_α is the air-mass-referenced flight-path angle and v_g is the UAV ground speed. From this figure, we can see that the pitch angle can be written as a function of the angle-of-attack and of γ_α. Therefore, we can set

$$F_\theta = \gamma_\alpha + \alpha,$$

and define the reference to $\dot{\theta}$ as

$$\dot{\theta}^r = K_{1,\theta}(F_\theta - \theta) + K_{2,\theta}\dot{F}_\theta,$$

with $K_{1,\theta} > 0$ and $K_{2,\theta} > 0$. As for the roll angle, we can set it so that there is no net side force acting on the UAV. For that, consider the schematics from Figure 4.2, where L is the lift force and χ is the course angle.

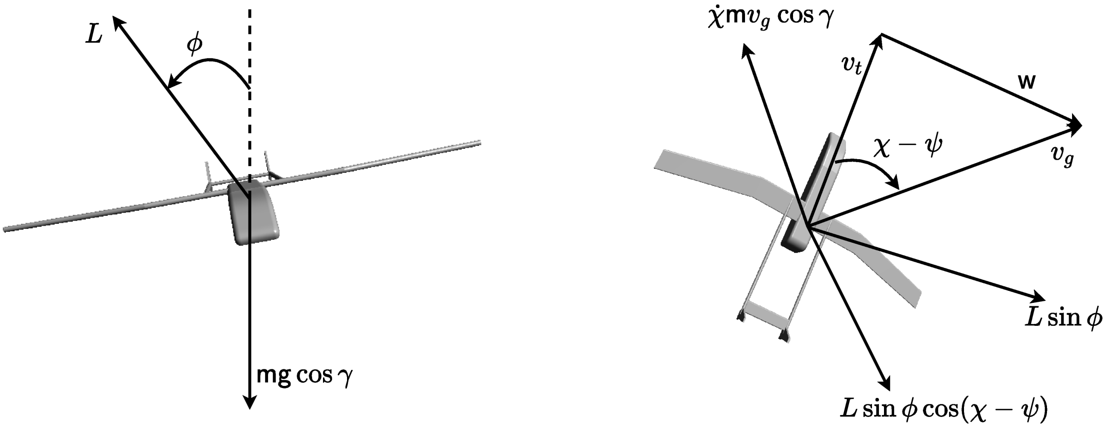

Figure 4.2: Free-body diagram of the UAV during a climbing coordinated turn.

Summing the forces in the horizontal plane, we obtain

$$L\sin\phi\cos(\chi - \psi) = \dot{\chi}\mathsf{m}v_g\cos\gamma, \tag{4.52}$$

and in the vertical plane

$$L\cos\phi = \mathsf{mg}\cos\gamma. \tag{4.53}$$

Dividing (4.52) by (4.53) we obtain the equation for a coordinated turn

$$\tan\phi = \frac{\dot{\chi}v_g}{\mathsf{g}\cos(\chi - \psi)}. \tag{4.54}$$

Moreover, in the absence of wind, we have $\chi = \psi$ and $v_t = v_g$. This way, (4.54) reduces to

$$\tan\phi = \frac{\dot{\psi}v_t}{\mathsf{g}}, \tag{4.55}$$

from which we can define the roll angle reference as

$$F_\phi = \arctan\left(\frac{\dot{\psi}v_t}{\mathsf{g}}\right),$$

and the reference to $\dot{\phi}$ as

$$\dot{\phi}^r = K_{1,\phi}(F_\phi - \phi) + K_{2,\phi}\dot{F}_\phi,$$

with $K_{1,\phi}$ and $K_{2,\phi} > 0$. Moreover, in Beard & McLain (2012), the authors show that (4.55) also holds in presence of wind. Therefore, we can define the reference trajectory for the first controller as

$$\boldsymbol{y}_0^r = \begin{bmatrix} F_{x_1} & F_{x_2} & F_{x_3} & \dot{\phi}^r & \dot{\theta}^r & \dot{\psi}^r \end{bmatrix}^T . \tag{4.56}$$

As for the second controller, we still need to define references for the aerodynamic angles. Let α_{sw} be the value of α at trimmed, steady-wing flight with null wind and sideslip. Then, we can set $\alpha^r = \alpha_{sw}$. Now, since flying at slip can be aerodynamically inefficient, as the lift-to-drag ratio is reduced. Then, we can define $\beta^r = 0$. This way, the reference vector for the second controller can be defined as

$$\boldsymbol{y}_0^r = \begin{bmatrix} v_t^r & \alpha^r & \beta^r & \dot{\phi}^r & \dot{\theta}^r & \dot{\psi}^r \end{bmatrix}^T . \tag{4.57}$$

With both (4.56) and (4.57) defined, we calculate $\boldsymbol{y}_0^r$ using the UAV position at $k = 0$ and integrate the vector field to obtain $\boldsymbol{y}_k^r$ for $k = 1, ..., n_p$.

5 Results

In this chapter, we present the results obtained with the proposed control and guidance strategies. We provide different simulation scenarios using an Aerosonde UAV in order to evaluate the proposed strategies. In the first two scenarios, we use two different target curves. In the third and fourth scenarios, we have added external disturbances and noisy sensor measurements to evaluate the robustness of the control strategies. Moreover, we also provide the simulation results for a Zagi UAV, so we can evaluate the proposed strategy in different UAVs.

The simulations were executed in MatLab using a simulation frequency of 1kHz and a control frequency of 100Hz. A time analysis is provided at the end of this chapter to evaluate the feasibility of practical implementation. Furthermore, we provide a video of the UAV performing the path following and obstacle avoidance tasks for each scenario. The video is available at https://youtu.be/_w8dxKzam8Q.

5.1 Aerosonde UAV

The Aerosonde is a small UAV designed by Insitu[1] to collect weather data over remote areas. Its aerodynamic coefficients can be found in Appendix A.1. The stall speed and the angle-of-attack limits are estimated using the lift curve of the UAV, presented in Figure 5.1.

In Figure 5.1 the blue line represents the dimensionless lift coefficient of the UAV, and

[1] https://www.insitu.com/

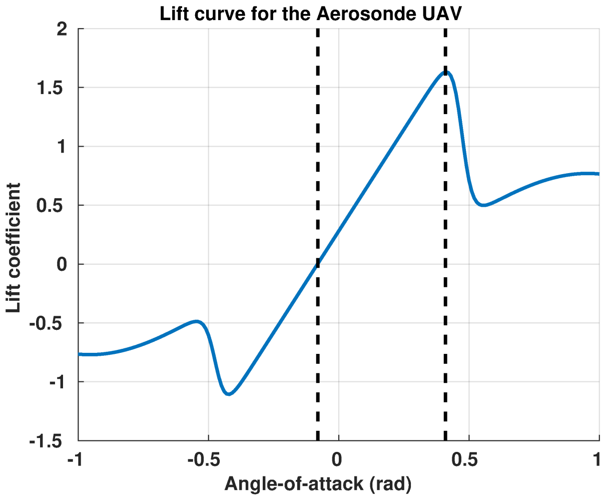

Figure 5.1: Lift curve, in blue, and α limits, in black, for the Aerosonde UAV.

the black dashed lines represent the upper and lower limits of the angle-of-attack. The lower limit corresponds to the point where the lift coefficient is zero, whereas the upper limit represents the point where an increase in the angle-of-attack is followed by a sudden reduction in lift, this point is also referred to as critical angle-of-attack. With the lift coefficient at the critical angle-of-attack ($L_{\max}$), it is possible to estimate the UAV stall speed by using

$$v_{stall} = \sqrt{\frac{2\mathsf{mg}}{\rho L_{\max} S}}.$$

For this UAV, we have that $v_{stall} = 15.25\text{m/s}$, $\alpha_{min} = -4.59\text{deg}$ and $\alpha_{\max} = 23.49\text{deg}$.

For the simulations, we have set a reference speed of $\vartheta = 35\text{m/s}$ and a perception range of $d_{in} = 200\text{m}$. For the trajectory gains, we have $K_{1,\psi} = K_{1,\phi} = 1.5$, $K_{1,\theta} = 1$, $K_{2,\psi} = 0.25$, $K_{2,\phi} = K_{2,\theta} = 1$, for both controllers. As for the MPC, we have $n_u = 2$, $n_p = 10$ with

$$\boldsymbol{Q} = \text{diag}\left(\begin{bmatrix}5 & 5 & 15 & 30 & 30 & 30 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}\right),$$
$$\boldsymbol{R} = \text{diag}\left(\begin{bmatrix}1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}\right),$$

for the first control strategy, and

$$\boldsymbol{Q} = \text{diag}\left(\begin{bmatrix}1 & 1 & 1 & 10 & 15 & 10 & 0 & 0 & 0 & 0 & 0 & 0\end{bmatrix}\right),$$
$$\boldsymbol{R} = \text{diag}\left(\begin{bmatrix}1 & 1 & 1 & 5 & 5 & 5\end{bmatrix}\right),$$

for the second control strategy. Moreover, $\boldsymbol{P}$ is given by the solution of the corresponding Algebraic Riccati Equation for each strategy. Furthermore, we have used the function *mpcActiveSetSolver* from MATLAB to solve the OCP.

As for the obstacles circulation, we have set the gain in (3.14) to $K_{G_i} = 2.5 \times 10^{-4}$, $\forall \mathcal{O}_i \in \mathcal{O}$. Finally, we consider that every obstacle has a corresponding cylinder with radius

$O_{i,r} = 50\text{m}$, thus, we set $\mathcal{O}_{i,R} = 75\text{m}$, $\forall \mathcal{O}_i \in \mathcal{O}$.

5.2 Noiseless scenarios

In this first case, we removed external disturbances and assumed that the UAV states are available to the controller. With that, we aim to test the effectiveness of the strategy. The target curve for the first scenario is defined implicitly by the functions

$$\begin{cases} \kappa_1 = x_3 - 100 + 50\frac{(x_1+2t)^2}{600^2}, \\ \kappa_2 = \frac{x_1^2}{600^2} + \frac{x_2^2}{400^2} - 1. \end{cases} \tag{5.1}$$

Also, for this curve, we have set the gains in (3.11) to $K_1 = 4.5 \times 10^{-5}$ and $K_2 = 3$. In this scenario, the curve is moving in the x_1 axis with a speed of -2m/s. Moreover, the obstacles move with the curve.

The target curve is represented by the black line in Figure 5.2. Furthermore, the obstacles are represented by the most internal cylinders, in black, the yellow cylinders give an extended $3D$ representation of the circulation curves $\mathcal{O}_{i,\kappa}$, and the green cylinders represent the perception distance d_{in}.

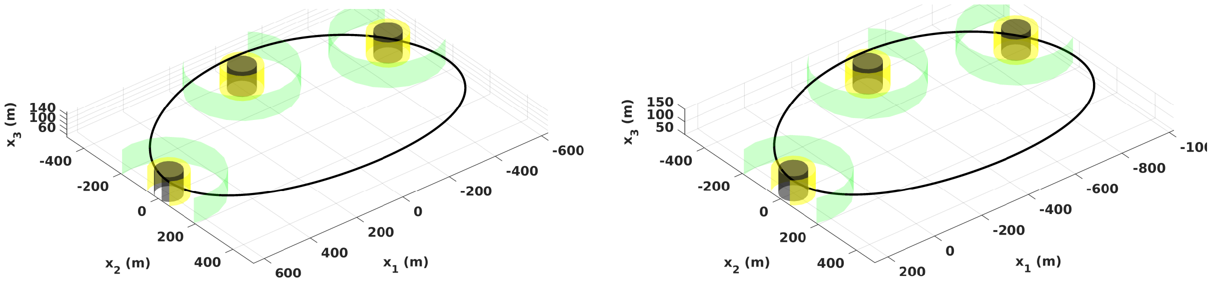

Figure 5.2: Target curve and obstacles represented at two time instants. On the left, we have the curve at $t = 0$s. On the right, we have the representation at $t = 200$s.

The UAV trajectory is shown in Figure 5.3. The obstacles are shown in the position they were at the time they were being circulated. Moreover, the UAV trajectory is represented by the blue lines when it is following the curve, by the red line when it is circulating the obstacle, and by the purple lines when it is in transition between behaviors. On the left, we have the results of the first control strategy, and on the right, the results of the second strategy. It can be seen that both strategies were efficient in leading the UAV to converge to and circulate the curve while avoiding the obstacles.

The convergence to the field can be seen in Figure 5.4. On top, we have the vector field orientation reference, in orange, and the UAV heading, in blue. On the bottom, we have the vector field velocity reference, in orange, and the UAV speed, in blue. From this figure, we can see that the second control strategy converges slightly better to the reference velocity than the first approach. This is also an effect of the conservative restriction used

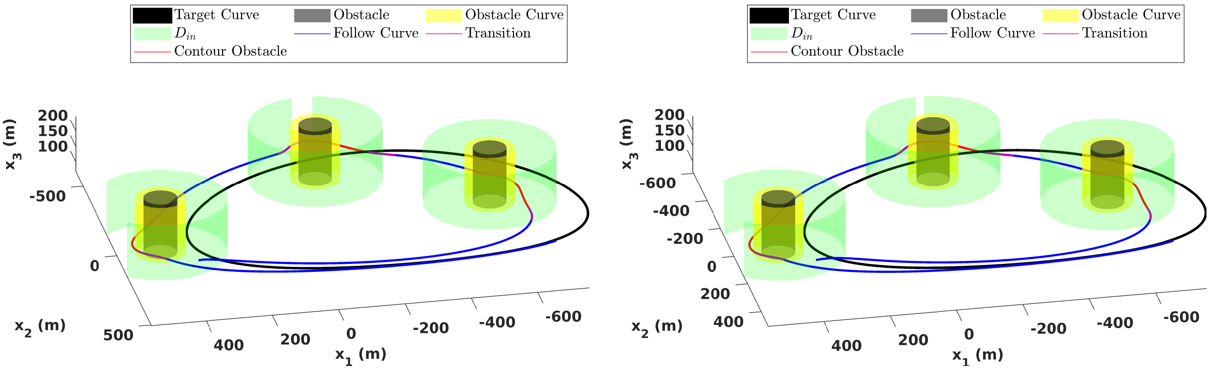

Figure 5.3: Trajectory executed by the UAV in the first scenario. In the left, the results for the first control strategy, on the right, the results for the second control strategy. The target curve is represented in black and the obstacles by the most internal cylinders.

in the first control strategy, when the UAV acceleration has to increase, or decrease, in order to keep the states inside the feasible solution set. However, both strategies were equally effective in tracking the heading reference. The convergence is also detailed in Figure 5.5, where each component of the reference velocities are represented by the orange lines, and the UAV speed is shown in blue.

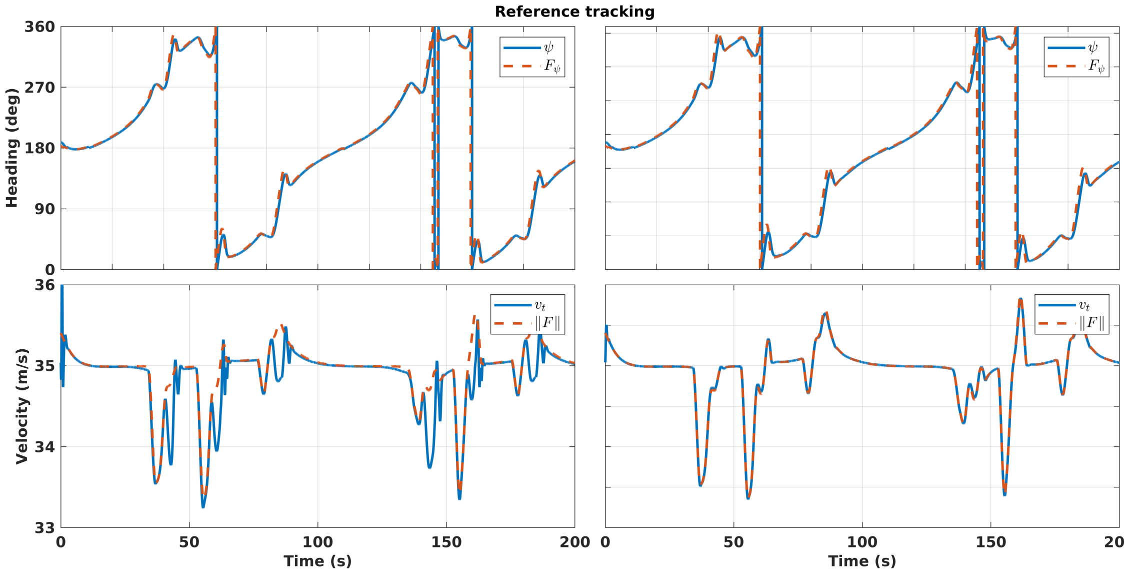

Figure 5.4: Heading tracking, on top, and velocity tracking, on bottom for the first scenario. The reference values are showed by the dashed lines, and the UAV states by the solid lines.

From both Figures 5.4 and 5.5, we can see that even though there is a small deviation in the x_3 component, the UAV still converges to the references. This deviation occurs when the derivative of ψ^r increases and the UAV has to perform a sharp turn. For the first controller, the mean absolute error was 3.15deg for the heading tracking and 0.10m/s for the velocity tracking, as for the second, we obtained 4.3deg and 0.003m/s.

Moreover, Figure 5.6 shows the applied control inputs and the corresponding bounds,

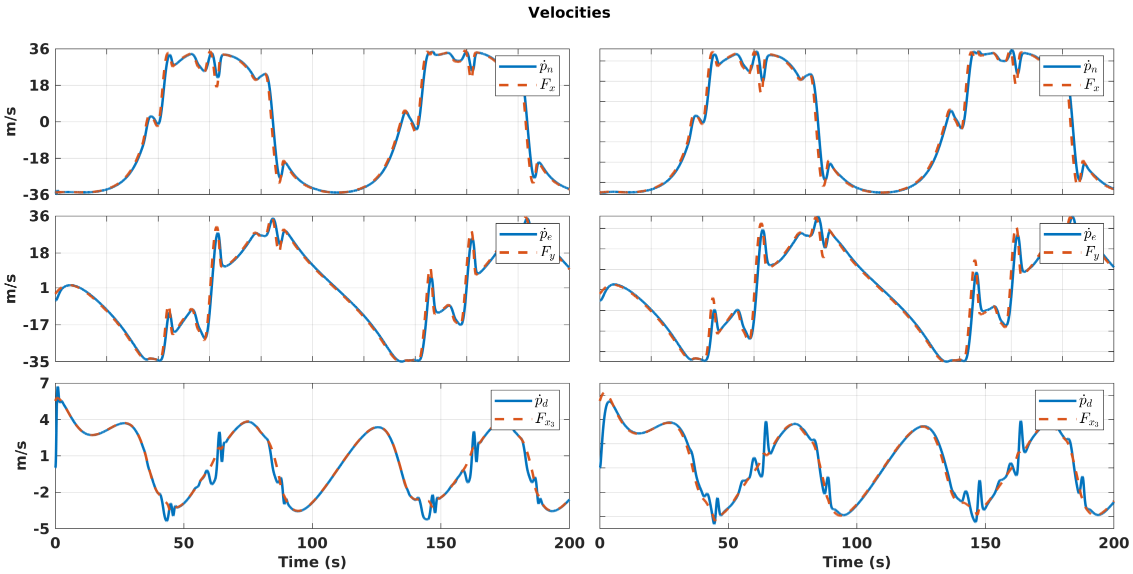

Figure 5.5: Vector field references, in orange, and UAV speed, in blue, for the first scenario. From top to bottom, we have: north, east and down velocities.

represented by the red, dashed lines. We can see that the ailerons and the rudder are the most demanded control surfaces due to constant changes in roll and yaw, especially when maneuvering around an obstacle. Also, the control inputs for both strategies are very similar, with a small difference at the beginning of the simulation.

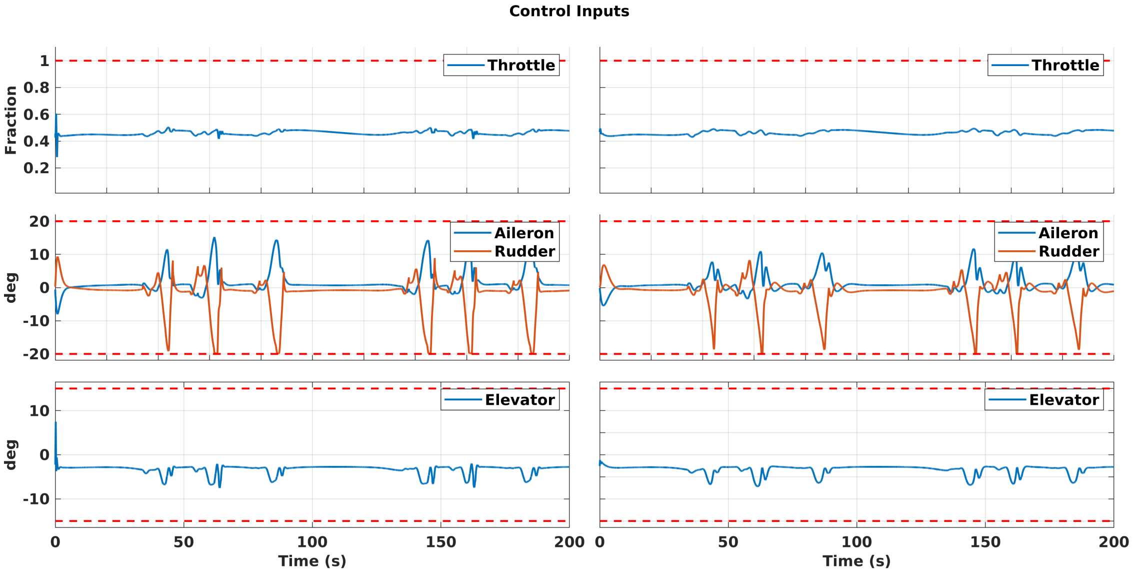

Figure 5.6: Applied control signals for the first scenario. From top to bottom, we have: the throttle fraction, aileron and rudder deflection and elevator deflection.

In the second scenario, we have selected the target curve defined implicitly by the functions

$$\begin{cases} \kappa_1 = & (x^2+y^2)^2 - 400^4 \\ & -600\cos\left(\frac{2\pi t}{400}\right)(y^3-3x^2y) \\ & +600\sin\left(\frac{2\pi t}{400}\right)(x^3-3xy^2), \\ \kappa_2 = & x_3 - 200 - \frac{x_1^2+x_2^2}{10^4}. \end{cases} \tag{5.2}$$

With the gains in (3.11) set to $K_1 = 0.5 \times 10^{-15}$, $K_2 = 50$ and $K_p = 3.5 \times 10^{-3}$. The target curve and the obstacles are represented in Figure 5.7. In this scenario, the curve is rotating around the x_3 axis, with angular speed of 0.15deg/s. Once again, the obstacles move with the curve.

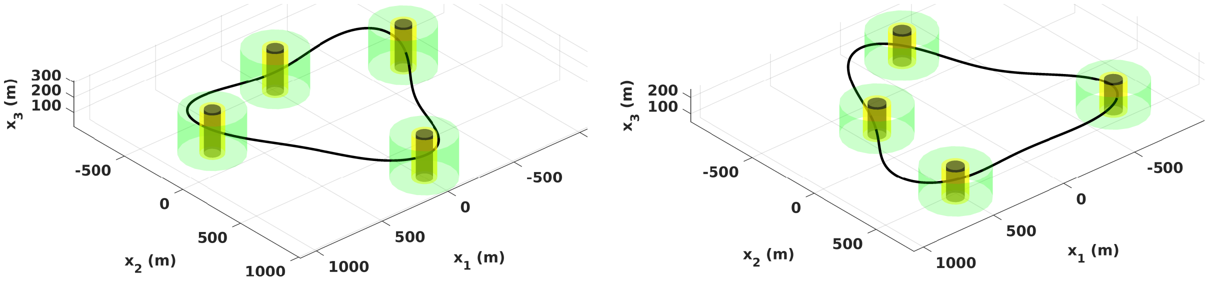

Figure 5.7: Target curve and obstacles represented at two time instants. On the left, we have the curve at $t = 0$s. On the right, we have the representation at $t = 200$s.

Figure 5.8 shows the UAV trajectory, for both strategies. In this scenario, we can see the effect of the *GetCircDir*() function, when one of the obstacles is circulated in a different direction in relation to the others. Moreover, note that the two obstacles at the top are the same, but they were circulated at different times.

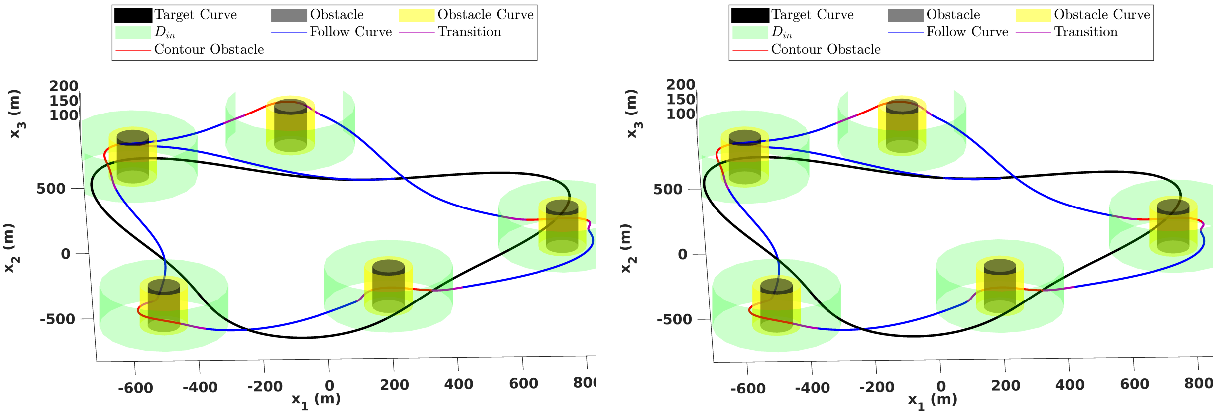

Figure 5.8: Trajectory executed by the UAV in the second scenario. In the left, the results for the first control strategy, on the right, the results for the second control strategy. The target curve is represented in black and the obstacles by the most internal cylinders.

The convergence for the second scenario is shown in Figure 5.10, and it can be seen in detail in Figure 5.9. As in the first scenario, the second controller converges to the speed references slightly better than the first. In this scenario, the first controller had a mean absolute error of 3.46deg for the heading tracking and 0.11m/s for the velocity tracking, as for the second, we obtained 4.91deg and 0.006m/s.

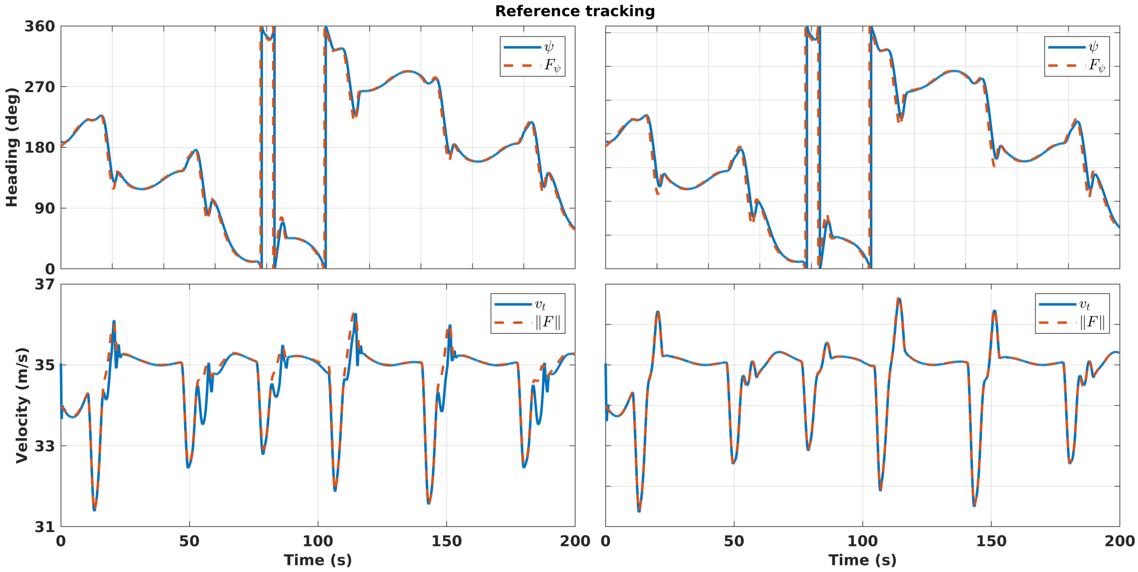

Figure 5.9: Heading tracking, on top, and velocity tracking, on bottom for the second scenario. The reference values are showed by the dashed lines, and the UAV states by the solid lines.

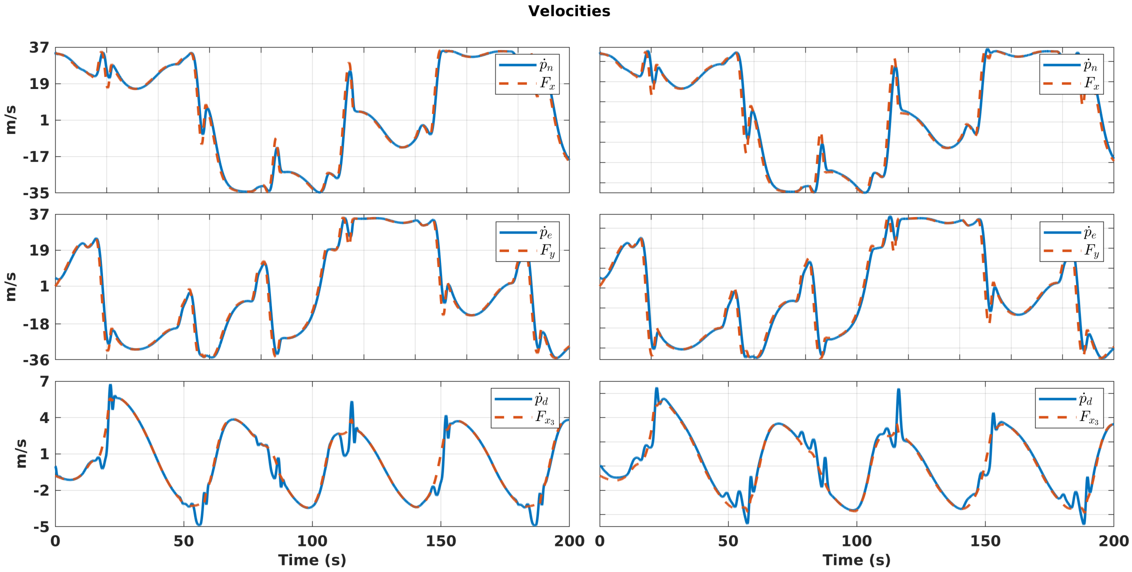

Figure 5.10: Vector field references, in orange, and UAV speed, in blue for the second scenario. From top to bottom, we have: north, east and down velocities.

Figure 5.11 shows the control inputs. We can see that this scenario demands more from the rudder, as the UAV has to change its orientation more often. As in the first case, both controllers produce almost the same control inputs.

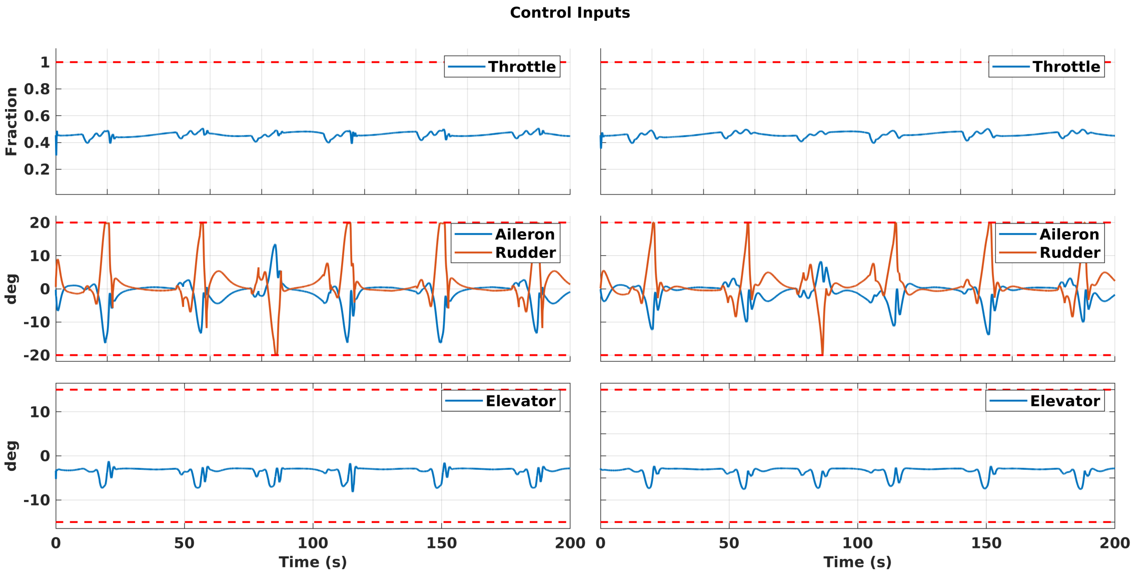

Figure 5.11: Applied control signals for the second scenario. From top to bottom, we have: the throttle fraction, aileron and rudder deflection and elevator deflection.

From these scenarios, we can see that the guidance strategy is effective in leading the UAV to converge to and circulate the target curve while avoiding the obstacles in the path. Furthermore, the proposed control laws efficiently drove the UAV to the vector field references, while respecting the state and control constraints.

5.3 Realistic scenarios

In order to provide more realistic results, we performed simulations in the presence of disturbances originated from sensor measurements and atmospheric disturbances. For the atmospheric disturbances, we based on Beard & McLain (2012) to consider the following Dryden model for gust wind

$$H_u(s) = \sigma_u \sqrt{\frac{2\vartheta}{L_u}} \frac{1}{s + \frac{\vartheta}{L_u}}, \tag{5.3}$$

$$H_v(s) = \sigma_v \sqrt{\frac{3\vartheta}{L_v}} \frac{s + \frac{\vartheta}{\sqrt{3}L_v}}{\left(s + \frac{\vartheta}{L_v}\right)^2}, \tag{5.4}$$

$$H_w(s) = \sigma_w \sqrt{\frac{3\vartheta}{L_w}} \frac{s + \frac{\vartheta}{\sqrt{3}L_w}}{\left(s + \frac{\vartheta}{L_w}\right)^2}, \tag{5.5}$$

which represent filters, L_u, L_v and L_w are spatial wavelengths, σ_u, σ_v and σ_w represent the turbulence intensities along the UAV body, frame and ϑ is the reference speed. The wind gust components are obtained by passing white noise through the filters (5.3)-(5.5). For this work, we have set $L_u = L_v = 200$, $L_w = 50$, $\sigma_u = \sigma_v = 1.06$, $\sigma_w = 0.7$, which are the parameters for low altitude, light turbulence wind (Beard & McLain, 2012). One should have in mind that the differences in the u axis from the v and w axis comes

Furthermore, we have set the steady wind to

$$\mathsf{w}_s = \begin{bmatrix} 3\text{m/s} & 1\text{m/s} & -0.5\text{m/s} \end{bmatrix}^T .$$

As for the sensor measurements, in this work we assume that the following sensors are available: *i*) a Global Positioning System (GPS), providing the position of the UAV with respect to the inertial frame along axes x_1 and x_2; *ii*) a barometer, providing the position of the UAV with respect to the inertial frame along axis x_3; *iii*) an accelerometer, providing the UAV accelerations in the body frame; *iv*) a gyroscope, providing the UAV angular rates in the body frame.

With these sensors, we have implemented an EKF following the guidelines presented in Chapter 2. For that, we have used the accelerometer and gyroscope measurements as inputs in the prediction step, and the ground truth data from the GPS and barometer in the correction step. For the covariance matrices, please refer to Appendix B.2.

Table 5.1 provides the parameters of sensors used in the experiment, based on Beard & McLain (2012) and Rego & Raffo (2019). The noises have a Gaussian distribution truncated within three times the standard deviation.

Table 5.1: Sensors parameters

Sensor	Noise Bounds	Sampling Time
Accelerometer	$\pm\ 7.5\times10^{-3}\mathsf{g}$	10ms
Gyroscope	± 0.43deg/s	10ms
GPS	± 0.15m	120ms
Barometer	± 0.6m	120ms

In the third scenario, we have used the curve defined by (5.1). The UAV trajectory is shown in Figure 5.12. Note that from the $3D$ trajectory, the UAV trajectory is very similar to the noise-free scenario.

Figure 5.13 shows the convergence to the vector field references. Observe that there is a small deviation from the references, when compared to the noiseless scenarios. Moreover, from Figure 5.14, we can note that the down component of the speed is the most affected by the disturbances. For the first controller, the mean absolute error was 5.41deg for the heading tracking and 0.17m/s for the velocity tracking, as for the second, we obtained 4.95deg and 0.16m/s. Note that for the second strategy, the average speed error increased more, indicating a greater sensitivity to disturbances.

From Figures 5.13 and 5.14, we can observe that, even in the presence of disturbances, the UAV was able to converge to the vector field references. Moreover, it should be noted that the references provided by the guidance strategy are almost equal to the ones obtained in the noiseless scenarios, thus showing the robustness of the strategy against noise.

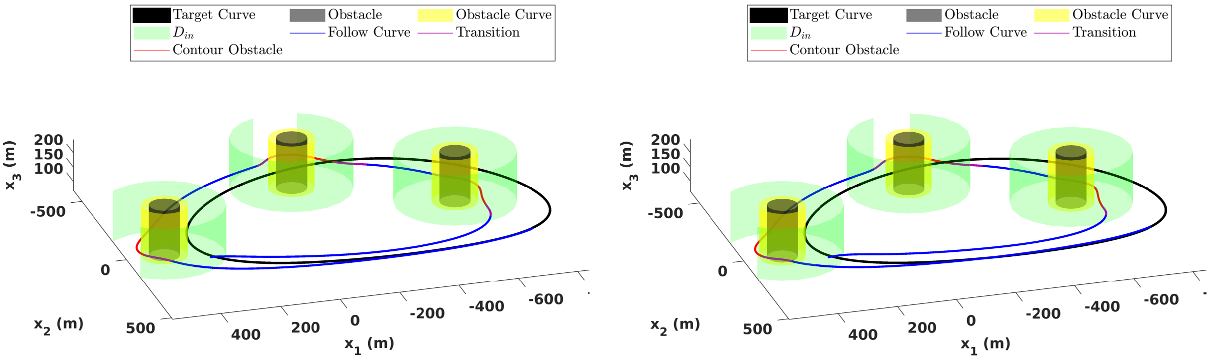

Figure 5.12: Trajectory executed by the UAV in the third scenario. In the left, the results for the first control strategy, on the right, the results for the second control strategy. The target curve is represented in black and the obstacles by the most internal cylinders.

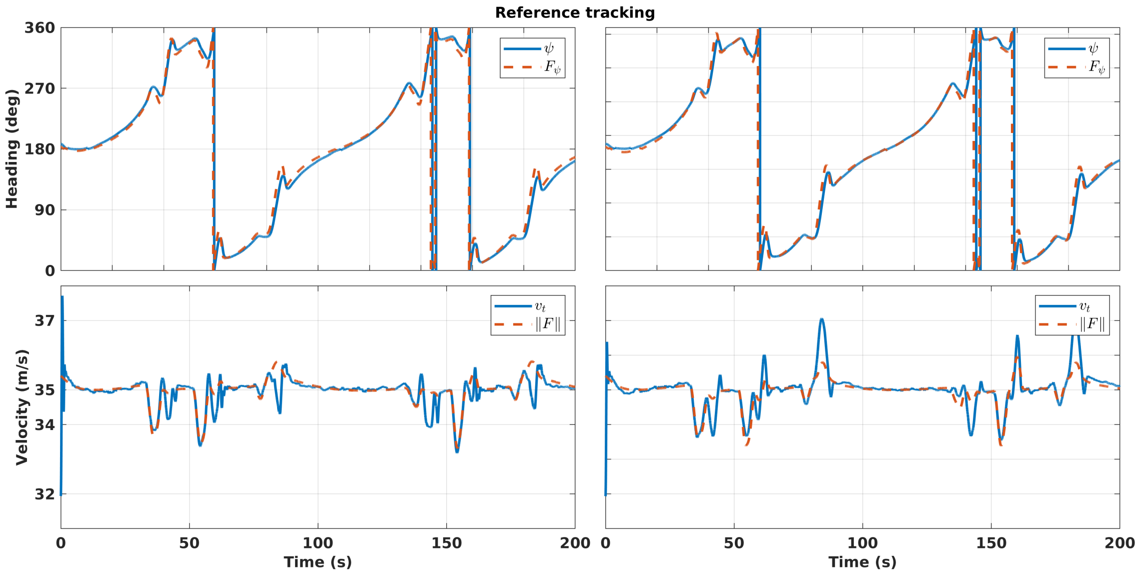

Figure 5.13: Heading tracking, on top, and velocity tracking, on bottom for the third scenario. The reference values are showed by the dashed lines, and the UAV states by the solid lines.

The control inputs for this scenario are depicted in Figure 5.15. Here, we can note the effects of the applied disturbances as a small chattering behavior, which can be easily seen in the aileron and rudder signals. Moreover, the chattering is slightly reduced when using the first control strategy.

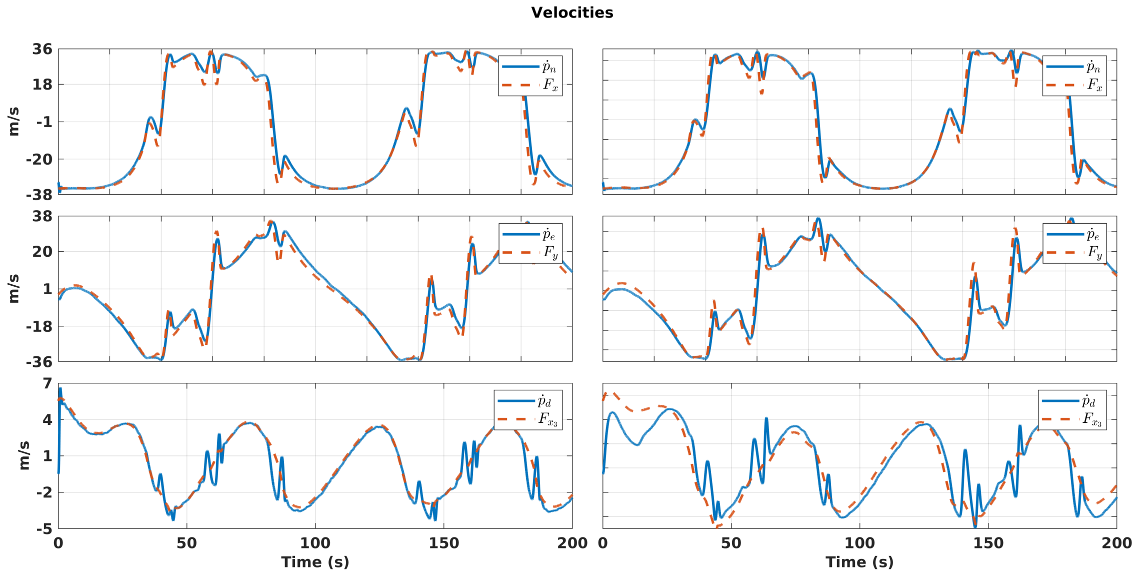

Figure 5.14: Vector field references, in orange, and UAV speed, in blue for the third scenario. From top to bottom, we have: north, east and down velocities.

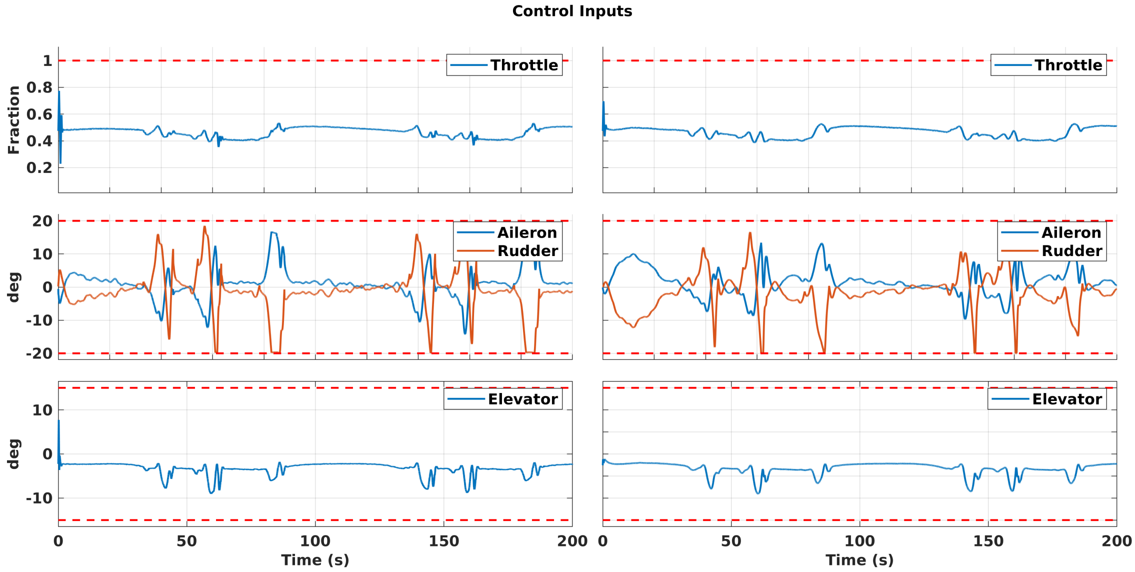

Figure 5.15: Applied control signals for the third scenario. From top to bottom, we have: the throttle fraction, aileron and rudder deflection and elevator deflection.

In the last scenario, we have used the target curve defined in (5.2). The UAV trajectory can be seen on Figure 5.16. From these two scenarios, we can observe that the disturbances have not affected the avoidance behavior, either the direction from which the obstacle circulation happens.

Similarly to the corresponding noiseless scenario, the UAV is able to converge to the references, as shown in Figures 5.17 and 5.18. For the first controller, the mean absolute error was 5.81deg for the heading tracking and 0.20m/s for the velocity tracking, as for the

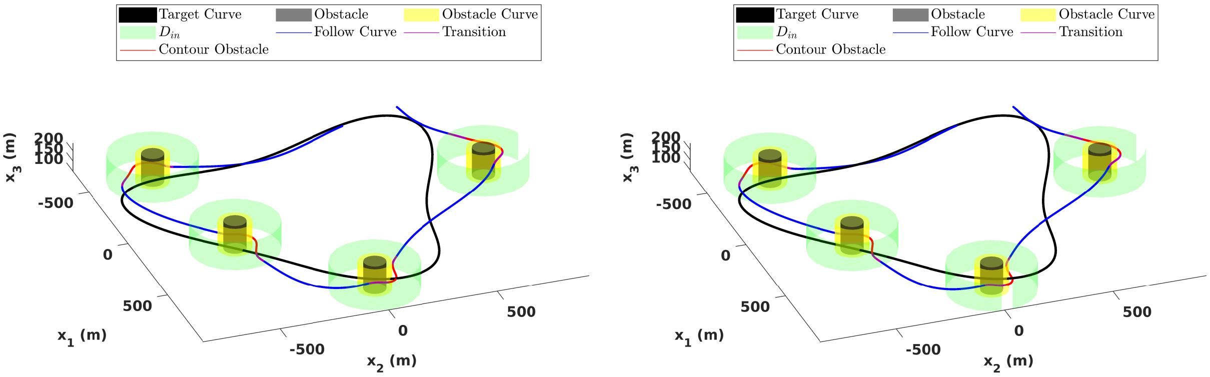

Figure 5.16: Trajectory executed by the UAV in the fourth scenario. In the left, the results for the first control strategy, on the right, the results for the second control strategy. The target curve is represented in black and the obstacles by the most internal cylinders.

second, we obtained 4.98deg and 0.18m/s.

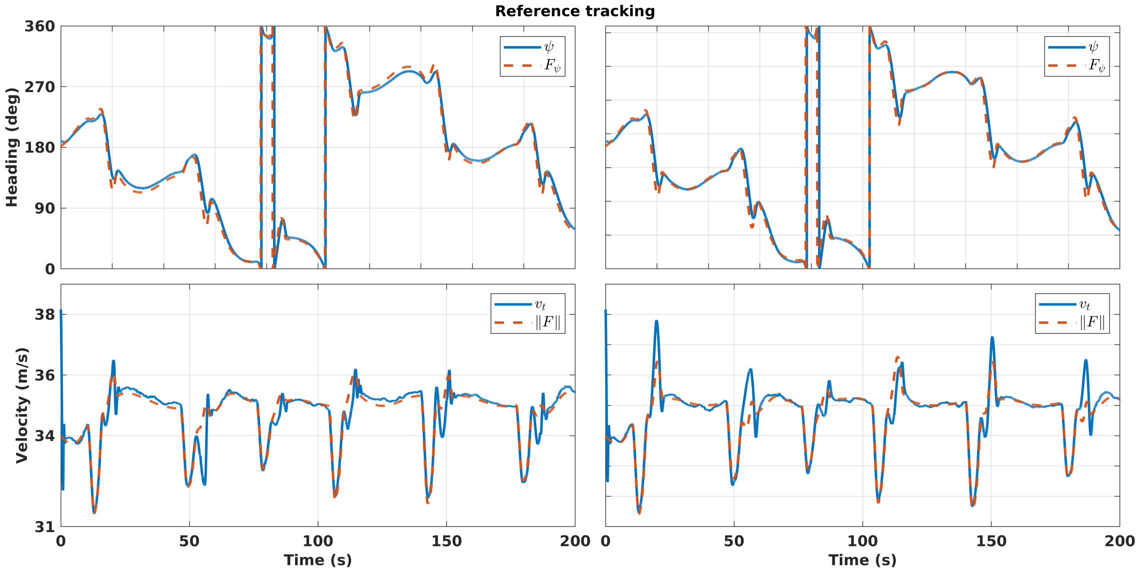

Figure 5.17: Heading tracking, on top, and velocity tracking, on bottom for the fourth scenario. The reference values are showed by the dashed lines, and the UAV states by the solid lines.

Figure 5.19 shows the control inputs. In this case, we can see that the rudder is the most demanded control surface, and the optimization problem kept it inside the boundaries.

From these two scenarios, it can be seen that despite the disturbances, the UAV is still able to converge to the references. The noise effect can be seen in the control inputs, as a small chattering, especially in the ailerons and elevators. Moreover, we can see that the first control strategy has a better behavior when dealing with the disturbances.

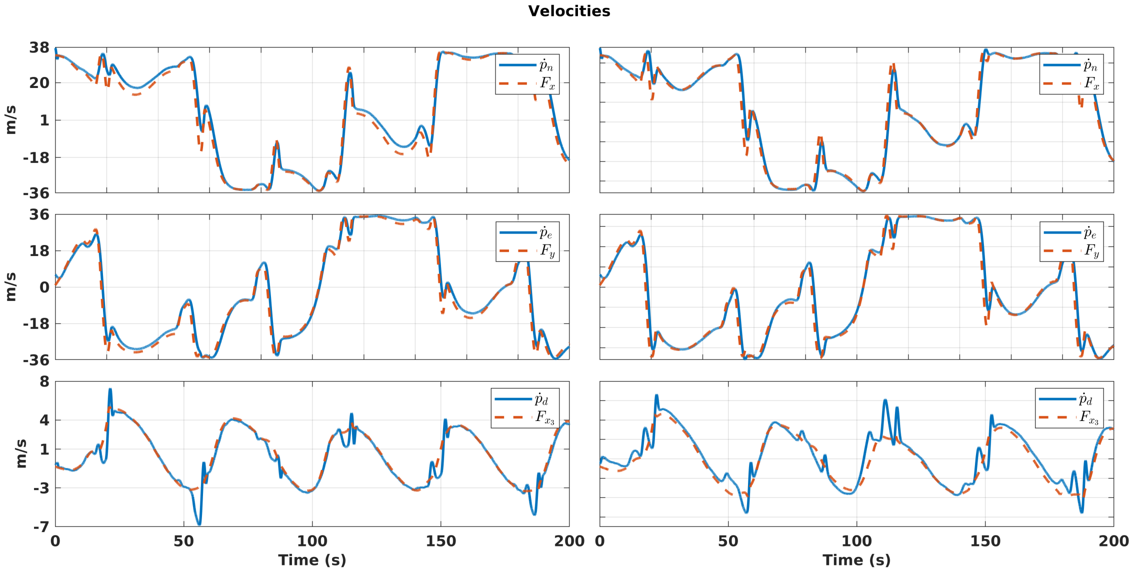

Figure 5.18: Vector field references, in orange, and UAV speed, in blue for the fourth scenario. From top to bottom, we have: north, east and down velocities.

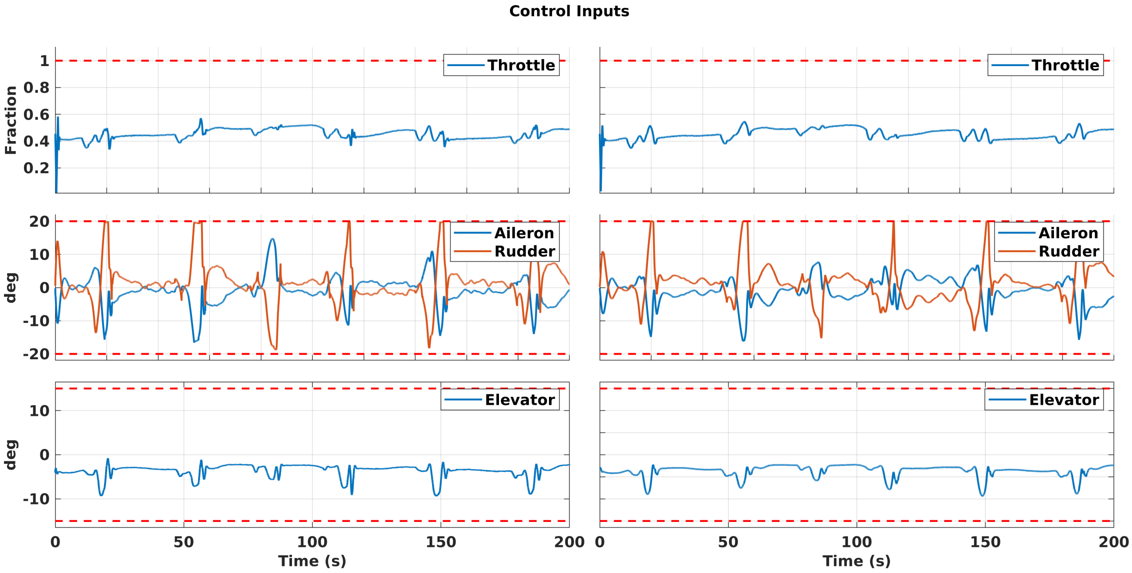

Figure 5.19: Applied control signals for the fourth scenario. From top to bottom, we have: the throttle fraction, aileron and rudder deflection and elevator deflection.

5.4 Zagi UAV

In order to test the efficiency of the strategy in different UAVs, we have simulated a scenario using a Zagi UAV. Different from the Aerosonde, the Zagi does not have ailerons, elevators, and rudder. Instead, it just has elevons, which are responsible for pitching, rolling, and yawing the UAV. The Zagi UAV can be seen on Figure 5.20.

Despite having different control surfaces, we can relate the elevons to the aileron-elevator

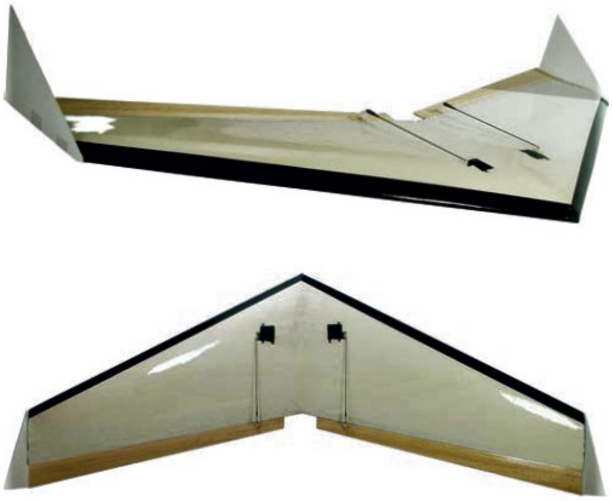

Figure 5.20: The Zagi UAV (*Source:* https://zagi.com/product/thl/).

signals with

$$\begin{bmatrix} \delta_e \\ \delta_a \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \delta_{er} \\ \delta_{el} \end{bmatrix}, \tag{5.6}$$

where δ_{er} is the right elevon deflection, and δ_{el} is the left elevon deflection. Therefore, we can use the same model for the forces and moments derived in Chapter 2. Also, by removing the rudder from the input vector, we obtain a relationship between the virtual control inputs, and the UAV control surfaces as follows

$$\bar{\boldsymbol{u}} = \boldsymbol{M}_{\boldsymbol{u}}^* \boldsymbol{u}^*, \tag{5.7}$$

with

$$\boldsymbol{M}_{\boldsymbol{u}}^* = \frac{1}{2}\rho v_t^2 S \begin{bmatrix} 0 & C_{X\delta_e} & \frac{S_{prop} C_{prop}}{v_t^2 S} k_{motor}^2 \hat{\delta}_t \\ C_{Y\delta_a} & 0 & 0 \\ 0 & C_{Z\delta_e} & 0 \\ bC_{l\delta_a} & 0 & -\frac{4k_{T_p} k_\omega^2 \hat{\delta}_t}{\rho v_t^2 S} \\ 0 & cC_{m\delta_e} & 0 \\ bC_{n\delta_a} & 0 & 0 \end{bmatrix}.$$

$$\bar{\boldsymbol{u}} = \begin{bmatrix} f_{x\boldsymbol{u}} & f_{y\boldsymbol{u}} & f_{z\boldsymbol{u}} & m_{\boldsymbol{u}} & l_{\boldsymbol{u}} & n_{\boldsymbol{u}} \end{bmatrix}^T,$$

$$\boldsymbol{u}^* = \begin{bmatrix} \delta_a & \delta_e & \delta_t \end{bmatrix}^T,$$

For this UAV, we have used the target curve defined in (5.1) and set the reference speed

$\vartheta = 15\text{m/s}$. Moreover, we set the steady wind to

$$\mathsf{w}_s = \begin{bmatrix} 0.5\text{m/s} & 1.5\text{m/s} & 0.25\text{m/s} \end{bmatrix}^T.$$

For the first controller, we set the gains to

$$\begin{aligned} \boldsymbol{Q} &= \operatorname{diag}\left(\begin{bmatrix} 0.1 & 0.1 & 0.1 & 5 & 5 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right), \\ \boldsymbol{R} &= \operatorname{diag}\left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}\right). \end{aligned}$$

As for the second controller, we set

$$\begin{aligned} \boldsymbol{Q} &= \operatorname{diag}\left(\begin{bmatrix} 1 & 1 & 1 & 10 & 10 & 10 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right), \\ \boldsymbol{R} &= \operatorname{diag}\left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}\right). \end{aligned}$$

The UAV trajectory can be observed in Figure 5.21. Note that the trajectory is similar to the one obtained with the Aerosonde UAV, with the difference that the Zagi takes longer to complete a loop around the target curve.

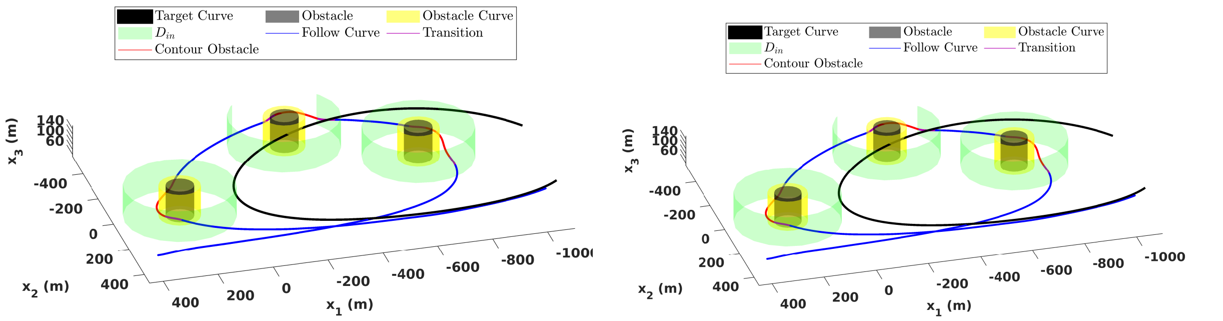

Figure 5.21: Trajectory executed by the UAV in the Zagi UAV scenario. In the left, the results for the first control strategy, on the right, the results for the second control strategy. The target curve is represented in black and the obstacles by the most internal cylinders.

The UAV convergence is presented in Figures 5.22 and 5.23. We can note that the UAV converges to both vector field references, with a small chattering in the speed due to the unmeasured disturbances. However, it can be noted that the UAV has some difficulty following the down velocity component reference. This is due to the lack of control surfaces. As the Zagi has only two deflection surfaces, it can only control two angles simultaneously. For the first controller, the mean absolute error was 6.87deg for the heading tracking and 0.089m/s for the velocity tracking, as for the second, we obtained 2.11deg and 0.055m/s.

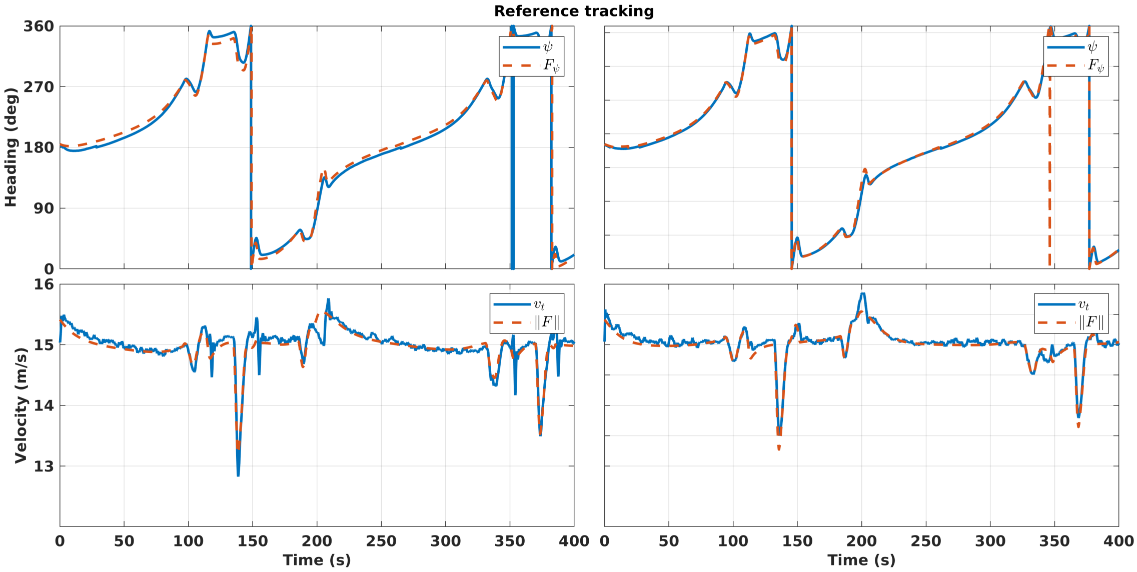

Figure 5.22: Heading tracking, on top, and velocity tracking, on bottom for the Zagi UAV scenario. The reference values are showed by the dashed lines, and the UAV states by the solid lines.

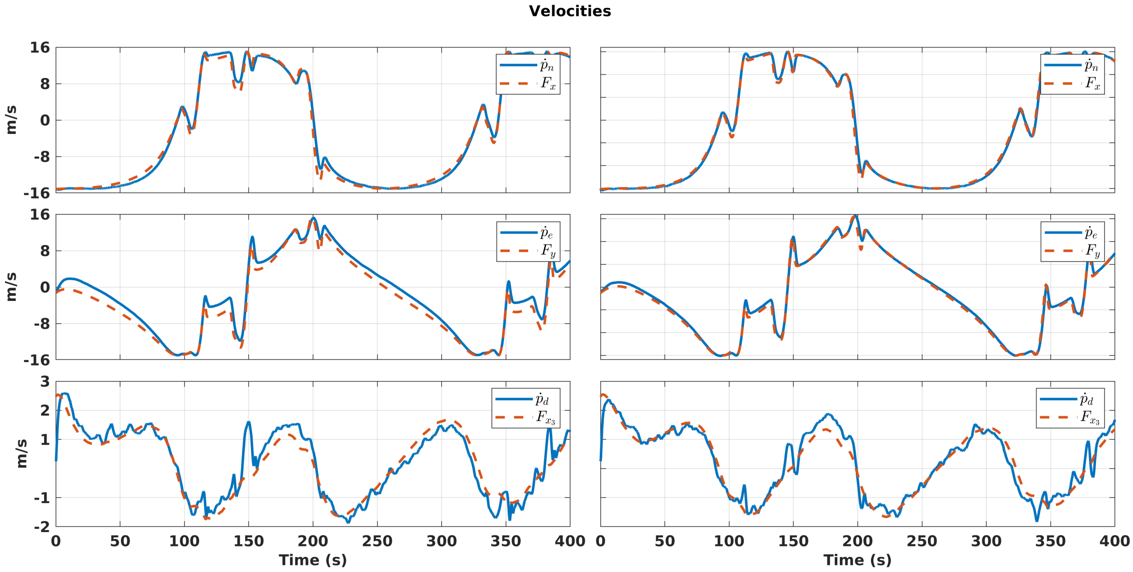

Figure 5.23: Vector field references, in orange, and UAV speed, in blue for the Zagi UAV scenario. From top to bottom, we have: north, east and down velocities.

Figure 5.24 shows the control inputs. Note that for this UAV, the throttle command is the most demanded. Also, notice that the elevons signals are almost symmetrical. This happens when the demand for rotational torque around the i_{body} axis exceeds the demand for torque in the j_{body} axis. In this situation, the elevons are acting as ailerons, producing rolling moment.

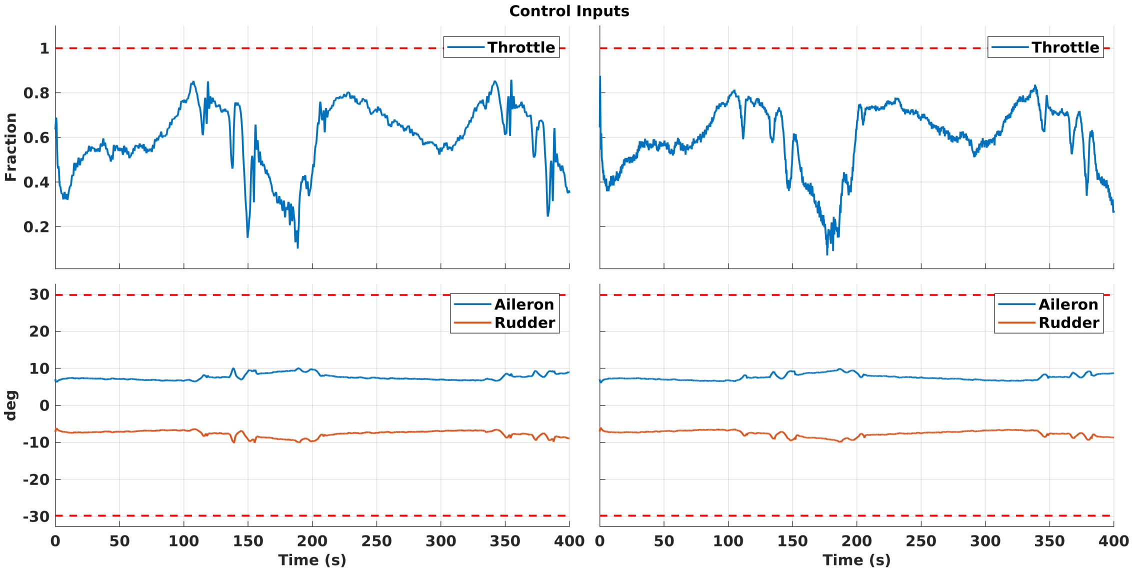

Figure 5.24: Applied control signals for the Zagi UAV scenario. From top to bottom, we have: the throttle fraction, left elevon and right elevon deflection.

5.5 Computational Effort

In this section, we assess the computational effort of the proposed control strategies, in order to evaluate the feasibility of a practical implementation. We have tested both strategies in an embedded hardware NVIDIA Jetson Nano Quad-core ARM A57 @1.43GHz 4 GB LPDDR4. Moreover, in order to generate a compatible software with the target hardware, we have used the MATLAB Coder Toolbox® to convert the entire simulation to the C++ language.

We have performed 2×10^4 executions of the control loop for each strategy. Table 5.2 shows the results for both strategies. In the first column, we have the maximum time, in seconds, taken to perform a control loop. We can note that the maximum execution time was smaller than the controller sampling time (10ms). Therefore, given that the control sampling time is greater than the maximum time, we conclude that the proposed strategy is feasible for practical implementation.

	Max Time (ms)	Mean Time (ms)	Standard Deviation (ms)
First Strategy	3.58	0.43	0.031
Second Strategy	3.25	0.31	0.021

Table 5.2: Timing analysis for both control strategies.

5.6 Comments on the results

Regarding the vector field strategy, it was shown that the strategy is able to produce a reference path that leads the UAV to converge to and circulate the target curves while avoiding the obstacles. Moreover, by using the proposed switching strategy, we were able to achieve continuity between the vector field behaviors, and therefore, we avoid abrupt changes in the references.

As for the control strategies, the FL+MPC scheme efficiently drove the UAV to the references provided by the vector field. In the first approach, the controller showed to be smoother than the second strategy. This happens because of the output vector selection. In the first approach, we were able to directly use the vector field references for the linear velocities in the NED frame. However, if we were to directly map these reference velocities into the UAV true speed and aerodynamic angles we could obtain undesirable references for α. To overcome this issue, we have set the reference to this angle as a fixed value. In addition, we have set the reference of β to zero, which in the presence of wind requires a bigger control effort. Because of that, the controller is more aggressive than the one used in the first approach.

On the other hand, the first approach uses a conservative constraint on the UAV linear velocities, which could lead to infeasibility of the solution. In the simulations, the OCP became unfeasible when the UAV speed was below 25m/s, which corresponds to an increase of approximately 60% when comparing to the UAV stall speed. This issue is solved in the second approach, by using the UAV true speed as output. Nonetheless, both strategies efficiently solved the control problem.

In addition, the computational effort analysis using an embedded computational system showed that both strategies can run faster than the controller sampling time. This is possible because the higher level vector field guidance controller is the one responsible for generating safe reference velocities, thus, we were able to remove the collision constraints from the MPC formulation. With that, together with the fact that the formulated MPC is linear thanks to the application of a feedback linearization scheme, we achieved reasonable execution time (less than 10ms), making it amenable to implementation in real systems.

6
Conclusion

This work dealt with the problem of guiding and controlling a fixed-wing UAV to converge to and circulate a desired curve, while avoiding obstacles in the path. The solution was divided into two steps: guidance and control. The guidance strategy acts at a higher level, providing a collision-free reference path for the UAV to converge to and circulate a desired curve. For that, we developed a vector-field strategy that switches between two behaviors: *i*) converge to and circulate the curve; and *ii*) circulate the nearest obstacle. Regarding the control strategy, we used the generalized forces and moments that originated from the UAV control surfaces as virtual control inputs. With that, we were able to apply a Feedback Linearization controller to transform the nonlinear UAV model into a linear one. This linear model is then used in an Incremental MPC framework which is responsible for driving the UAV states to the vector field references.

Simulations were performed in MatLab using the full 6DOF UAV model to show the efficiency of the proposed strategy. In addition, we also provide simulation results in the presence of external disturbances and noisy sensor measurements. In all scenarios, both strategies were able to guide the UAV to the reference curve and avoid the obstacles. Furthermore, a computational effort analysis showed that the proposed strategy is feasible for practical implementation.

From the simulation results, we could observe that each strategy has its own advantages and issues. In the first control strategy, the controller is less aggressive, however, since we are using a conservative restriction, the feasible solutions set is reduced. The second controller overcomes this issue by using the UAV true speed as output for the system.

Nonetheless, it is more sensitive to disturbances.

6.1 Future works

Regarding the guidance strategy, we intend to extend it to a more general class of obstacle avoidance problems. One future contribution would be to remove the cylinders around the obstacles, thus allowing the avoidance to occur in any direction. Also, we intend to deal with the case where the obstacles may move closer to each other. With that, the strategy could be extended to the guidance of multiple UAVs, by representing each UAV as an obstacle for the remaining other UAVs. Moreover, a study on new vector field strategies is already under development, in which we intend to use parametric curves.

As for the control problem, we intend to develop a stability proof for the proposed controllers. Furthermore, we intend to implement a tube-based strategy to improve the robustness of the strategy. Moreover, given that the execution time is feasible for practical implementation, we intend to implement it in a real platform. Finally, the online detection of the obstacles by real sensors would be a future contribution to the project.

Bibliography

Akbaripour, H. & Masehian, E. (2017). Semi-lazy probabilistic roadmap: a parameter-tuned, resilient and robust path planning method for manipulator robots. *The International Journal of Advanced Manufacturing Technology*, 89(5-8), 1401–1430.

Ali, R. & Ragb, H. K. (2019). Fused deep convolutional neural networks based on voting approach for efficient object classification. In *2019 IEEE National Aerospace and Electronics Conference (NAECON)* (pp. 335–339).

Alotaibi, E. T., Alqefari, S. S., & Koubaa, A. (2019). Lsar: Multi-uav collaboration for search and rescue missions. *IEEE Access*, 7, 55817–55832.

Andersen, T. S. & Kristiansen, R. (2017). Path-following in three dimensions using quaternions for a fixed-wing uav. In *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)* (pp. 1117–1122).: IEEE.

Beard, R. W., Ferrin, J., & Humpherys, J. (2014). Fixed wing uav path following in wind with input constraints. *IEEE Transactions on Control Systems Technology*, 22(6), 2103–2117.

Beard, R. W. & McLain, T. W. (2012). *Small unmanned aircraft: Theory and practice*. Princeton university press.

Bohlin, R. & Kavraki, L. E. (2000). Path planning using lazy prm. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1 (pp. 521–528).: IEEE.

Borrelli, F., Bemporad, A., & Morari, M. (2017). *Predictive control for linear and hybrid systems*. Cambridge University Press.

Choset, H. M., Hutchinson, S., Lynch, K. M., Kantor, G., Burgard, W., Kavraki, L. E., Thrun, S., & Arkin, R. C. (2005). *Principles of robot motion: theory, algorithms, and implementation*. MIT press.

Deng, J., Becerra, V., & Stobart, R. (2009). Input constraints handling in an mpc/feedback linearization scheme. *International Journal of Applied Mathematics and Computer Science*, 19(2), 219–232.

Dubins, L. E. (1957). On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3), 497–516.

Espinoza, T., Dzul, A., Lozano, R., & Parada, P. (2014). Backstepping-sliding mode controllers applied to a fixed-wing uav. *Journal of Intelligent & Robotic Systems*, 73(1-4), 67–79.

Ge, J., Liu, L., Dong, X., & Tian, W. (2020). Trajectory planning of fixed-wing uav using kinodynamic rrt* algorithm. In *2020 10th International Conference on Information Science and Technology (ICIST)* (pp. 44–49).: IEEE.

Gonçalves, V. M., de Araújo Pimenta, L. C., Maia, C. A., Dutra, B. C. O., & Pereira, G. A. S. (2010). Vector fields for robot navigation along time-varying curves in n-dimensions. *IEEE Transactions on Robotics*, 26(4), 647–659.

Hagen, I. B., Kufoalor, D. K. M., Brekke, E. F., & Johansen, T. A. (2018). Mpc-based collision avoidance strategy for existing marine vessel guidance systems. In *2018 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 7618–7623).: IEEE.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2), 100–107.

Ismail, A., Tuyishimire, E., & Bagula, A. (2018). Generating dubins path for fixed wing uavs in search missions. In *International Symposium on Ubiquitous Networking* (pp. 347–358).: Springer.

Jahn, A. & de Araújo Pimenta, L. C. (2016). Sampling based path planning and vector fields for curve tracking by uavs. In *2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR)* (pp. 223–228).: IEEE.

Jesus, T. A., de Araújo Pimenta, L. C., Tôrres, L. A. B., & Mendes, E. M. A. M. (2013). On the coordination of constrained fixed-wing unmanned aerial vehicles. *Journal of Control, Automation and Electrical Systems*, 24(5), 585–600.

Jung, D. & Tsiotras, P. (2008). Bank-to-turn control for a small uav using backstepping and parameter adaptation. *IFAC Proceedings Volumes*, 41(2), 4406–4411.

Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems [j]. *Journal of basic Engineering*, 82(1), 35–45.

Kang, Y. & Hedrick, J. K. (2009). Linear tracking for a fixed-wing uav using nonlinear model predictive control. *IEEE Transactions on Control Systems Technology*, 17(5), 1202–1210.

Karaman, S. & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7), 846–894.

Khalil, H. K. & Grizzle, J. W. (2002). *Nonlinear systems*, volume 3. Prentice hall Upper Saddle River, NJ.

Kirk, D. E. (2004). *Optimal control theory: an introduction.* Courier Corporation.

LaValle, S. M. (1998). Rapidly-exploring random trees: A new tool for path planning.

Lee, D. & Shim, D. H. (2016). Path planner based on bidirectional spline-rrt$*$ for fixed-wing uavs. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 77–86).: IEEE.

Limón, D., Alvarado, I., Alamo, T., & Camacho, E. F. (2008). Mpc for tracking piecewise constant references for constrained linear systems. *Automatica*, 44(9), 2382–2387.

Lu, L., Zong, C., Lei, X., Chen, B., & Zhao, P. (2016). Fixed-wing uav path planning in a dynamic environment via dynamic rrt algorithm. In *Mechanism and Machine Science* (pp. 271–282). Springer.

Lugo-Cárdenas, I., Flores, G., Salazar, S., & Lozano, R. (2014). Dubins path generation for a fixed wing uav. In *2014 International conference on unmanned aircraft systems (ICUAS)* (pp. 339–346).: IEEE.

Mammarella, M. & Capello, E. (2018). A robust mpc-based autopilot for mini uavs. In *2018 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 1227–1235).: IEEE.

Mathisen, S. H., Fossen, T. I., & Johansen, T. A. (2015). Non-linear model predictive control for guidance of a fixed-wing uav in precision deep stall landing. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 356–365).: IEEE.

Mayne, D. Q., Rawlings, J. B., Rao, C. V., & Scokaert, P. O. (2000). Constrained model predictive control: Stability and optimality. *Automatica*, 36(6), 789–814.

McLain, T., Beard, R. W., & Owen, M. (2014). Implementing dubins airplane paths on fixed-wing uavs.

Novák, A., Sedláčková, A. N., Bugaj, M., Kandera, B., & Lusiak, T. (2020). Use of unmanned aerial vehicles in aircraft maintenance. *Transportation Research Procedia*, 51, 160–170.

Oettershagen, P., Melzer, A., Leutenegger, S., Alexis, K., & Siegwart, R. (2014). Explicit model predictive control and l 1-navigation strategies for fixed-wing uav path tracking. In *22nd Mediterranean Conference on Control and Automation* (pp. 1159–1165).: IEEE.

Olavo, J. L. G., Thums, G. D., Jesus, T. A., De Araujo Pimenta, L. C., Torres, L. A. B., & Palhares, R. M. (2018). Robust guidance strategy for target circulation by controlled uav. *IEEE Transactions on Aerospace and Electronic Systems*, 54(3), 1415–1431.

Pereira, G. A. S., Choudhury, S., & Scherer, S. (2016). A framework for optimal repairing of vector field-based motion plans. In *2016 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 261–266).: IEEE.

Pereira, L. A. A., de Araújo Pimenta, L. C., & Raffo, G. V. (2020). Mpc based feedback-linearization strategy of a fixed-wing uav. *Anais da Sociedade Brasileira de Automática*, 2(1).

Pester, A. & Schrittesser, M. (2019). Object detection with raspberry Pi3 and movidius neural network stick. In *2019 5th Experiment International Conference (exp.at'19)* (pp. 326–330).

Quintero, S. A., Copp, D. A., & Hespanha, J. P. (2015). Robust uav coordination for target tracking using output-feedback model predictive control with moving horizon estimation. In *2015 American Control Conference (ACC)* (pp. 3758–3764).: IEEE.

Rego, B. S. & Raffo, G. V. (2019). Suspended load path tracking control using a tilt-rotor uav based on zonotopic state estimation. *Journal of the Franklin Institute*, 356(4), 1695–1729.

Ren, W. & Atkins, E. (2005). Nonlinear trajectory tracking for fixed wing uavs via backstepping and parameter adaptation. In *AIAA Guidance, Navigation, and Control Conference and Exhibit* (pp. 6196).

Rezende, A. M. C., Gonçalves, V. M., Raffo, G. V., & de Araújo Pimenta, L. C. (2018). Robust fixed-wing uav guidance with circulating artificial vector fields. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 5892–5899).: IEEE.

Rimon, E. & Koditschek, D. E. (1992). Exact robot navigation using artificial potential functions. *Departmental Papers (ESE)*, (pp. 323).

Rossiter, J. A. (2003). *Model-based predictive control: a practical approach*. CRC press.

Sartori, D., Quagliotti, F., Rutherford, M. J., & Valavanis, K. P. (2013). Design and development of a backstepping controller autopilot for fixed-wing uavs. *Journal of Aircraft: AIAA*.

Schnelle, F. & Eberhard, P. (2015). Constraint mapping in a feedback linearization/mpc scheme for trajectory tracking of underactuated multibody systems. *IFAC-PapersOnLine*, 48(23), 446–451.

Sepulchre, R., Jankovic, M., & Kokotovic, P. V. (2012). *Constructive nonlinear control*. Springer Science & Business Media.

Shakhatreh, H., Sawalmeh, A. H., Al-Fuqaha, A., Dou, Z., Almaita, E., Khalil, I., Othman, N. S., Khreishah, A., & Guizani, M. (2019). Unmanned aerial vehicles (uavs): A survey on civil applications and key research challenges. *Ieee Access*, 7, 48572–48634.

Sherstjuk, V., Zharikova, M., & Sokol, I. (2018). Forest fire-fighting monitoring system based on uav team and remote sensing. In *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)* (pp. 663–668).: IEEE.

Simon, D., Löfberg, J., & Glad, T. (2013). Nonlinear model predictive control using feedback linearization and local inner convex constraint approximations. In *2013 European Control Conference (ECC)* (pp. 2056–2061).: IEEE.

Song, X. & Hu, S. (2017). 2d path planning with dubins-path-based a∗ algorithm for a fixed-wing uav. In *2017 3rd IEEE international conference on control science and systems engineering (ICCSSE)* (pp. 69–73).: IEEE.

Stevens, B. L., Lewis, F. L., & Johnson, E. N. (2015). *Aircraft control and simulation: dynamics, controls design, and autonomous systems*. John Wiley & Sons.

Suomi, I. & Vihma, T. (2018). Wind gust measurement techniques—from traditional anemometry to new possibilities. *Sensors*, 18(4), 1300.

Váňa, P., Neto, A. A., Faigl, J., & Macharet, D. G. (2020). Minimal 3d dubins path with bounded curvature and pitch angle. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 8497–8503).: IEEE.

Yang, K., Kang, Y., & Sukkarieh, S. (2013). Adaptive nonlinear model predictive path-following control for a fixed-wing unmanned aerial vehicle. *International Journal of Control, Automation and Systems*, 11(1), 65–74.

Yang, L., Qi, J., Xiao, J., & Yong, X. (2014). A literature review of uav 3d path planning. In *Proceeding of the 11th World Congress on Intelligent Control and Automation* (pp. 2376–2381).: IEEE.

Yao, W., Lin, B., & Cao, M. (2019). Integrated path following and collision avoidance using a composite vector field. In *2019 IEEE 58th Conference on Decision and Control (CDC)* (pp. 250–255).: IEEE.

Yuan, C., Liu, Z., & Zhang, Y. (2015). Uav-based forest fire detection and tracking using image processing techniques. In *2015 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 639–643).: IEEE.

Zhang, S., Xu, T., Cheng, H., & Liang, F. (2020). Collision avoidance of fixed-wing uavs in dynamic environments based on spline-rrt and velocity obstacle. In *2020 International Conference on Unmanned Aircraft Systems (ICUAS)* (pp. 48–58).: IEEE.

Zhao, W. & Go, T. (2010). 3-d formulation of formation flight based on model predictive control with collision avoidance scheme. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition* (pp. 493).

Zhao, W. & Go, T. H. (2014). Quadcopter formation flight control combining mpc and robust feedback linearization. *Journal of the Franklin Institute*, 351(3), 1335–1355.

Zogopoulos-Papaliakos, G. & Kyriakopoulos, K. J. (2020). A flight envelope determination and protection system for fixed-wing uavs. In *2020 IEEE International Conference on Robotics and Automation (ICRA)* (pp. 9599–9605).: IEEE.

A UAV parameters

A.1 Aerodynamic coefficients

In this section, we detail the aerodynamic coefficients presented in Section 2.1. Each coefficient represents the gain of a respective state or control into a determined force or moment. For instance, $C_{L_{\delta_e}}$ represents the gain from the elevators to the Lift force. The parameters for both UAV were taken from Beard & McLain (2012).

Aerosonde UAV

Parameter	Value	Forces Coefficients	Value	Moments Coefficients	value
m	13.5kg	C_{D_0}	0.03	C_{l_0}	0
J_{xx}	$0.8244\text{kg}\cdot\text{m}^2$	C_{D_α}	0.3	C_{l_β}	-0.12
J_{yy}	$1.135\text{kg}\cdot\text{m}^2$	C_{D_q}	0	C_{l_r}	0.14
J_{zz}	$1.759\text{kg}\cdot\text{m}^2$	C_{D_p}	0.0437	C_{l_p}	-0.26
J_{xz}	$0.1204\text{kg}\cdot\text{m}^2$	$C_{D_{\delta_e}}$	0	$C_{l_{\delta_a}}$	0.08
S	0.55m^2	C_{L_0}	0.28	$C_{l_{\delta_r}}$	0.105
b	2.8956m	C_{L_α}	3.45	C_{m_0}	-0.02338
c	0.18994m	C_{L_q}	0	C_{m_α}	-0.38
S_{prop}	0.2027m^2	$C_{L_{\delta_e}}$	-0.36	C_{m_q}	-3.6
ρ	$1.2682\text{kg}/\text{m}^3$	C_{Y_0}	0	$C_{m_{\delta_e}}$	-0.5
k_{motor}	80	C_{Y_β}	-0.98	C_{n_0}	0
k_{T_p}	0	C_{Y_r}	0	C_{n_β}	0.25
k_Ω	0	C_{Y_p}	0	C_{n_r}	-0.35
e	0.9	$C_{Y_{\delta_a}}$	0	C_{n_p}	0.022
C_{prop}	1	$C_{Y_{\delta_r}}$	-0.17	$C_{n_{\delta_r}}$	-0.032
α_0	0.4712			$C_{n_{\delta_a}}$	0.06
M	50				

ϵ	0.1592				

Table A.1: Aerodynamic coefficients and parameters for the Aerosonde UAV.

Zagi UAV

Parameter	Value	Forces Coefficients	Value	Moments Coefficients	value
m	1.56kg	C_{D_0}	0.01631	C_{l_0}	0
J_{xx}	$0.1147\text{kg} \cdot \text{m}^2$	C_{D_α}	0.2108	C_{l_β}	-0.02854
J_{yy}	$0.0576\text{kg} \cdot \text{m}^2$	C_{D_q}	0	C_{l_r}	0.03066
J_{zz}	$0.1712\text{kg} \cdot \text{m}^2$	C_{D_p}	0.0254	C_{l_p}	-0.3209
J_{xz}	$0.0015\text{kg} \cdot \text{m}^2$	$C_{D_{\delta_e}}$	0.3045	$C_{l_{\delta_a}}$	0.1682
S	0.2589m^2	C_{L_0}	0.09167	C_{m_0}	-0.02338
b	1.4224m	C_{L_α}	3.5016	C_{m_α}	-0.5675
c	0.3302m	C_{L_q}	2.8932	C_{m_q}	-1.3990
S_{prop}	0.0314m^2	$C_{L_{\delta_e}}$	0.2724	$C_{m_{\delta_e}}$	-0.3254
ρ	1.2682kg/m^3	C_{Y_0}	0	C_{n_0}	0
k_{motor}	20	C_{Y_β}	-0.07359	C_{n_β}	-0.00040
k_{T_p}	0	C_{Y_r}	0	C_{n_r}	-0.00434
k_Ω	0	C_{Y_p}	0	C_{n_p}	-0.01297
e	0.9	$C_{Y_{\delta_a}}$	0	$C_{n_{\delta_a}}$	-0.00328
C_{prop}	1				
α_0	0.4712				
M	50				
ϵ	0.1592				

Table A.2: Aerodynamic coefficients and parameters for the Zagi UAV.

It should be noted that though the Zagi UAV has elevons instead of ailerons and elevators, the coefficients are expressed in terms of the latter.

B

Control matrices

In this chapter, we present the matrices used in the development of the control strategies proposed in this work.

B.1 Feedback Linearization matrices

In this section, we seek to present the feedback linearization matrices $\boldsymbol{\Lambda}(\boldsymbol{x})$ and $\boldsymbol{\lambda}(\boldsymbol{x})$ for both controllers. For both cases, we have that $f(\boldsymbol{x})$ is defined as

$$f(\boldsymbol{x}) = \begin{pmatrix} u_r c_\psi c_\theta + v_r \left(c_\psi s_\theta s_\phi - s_\psi c_\phi\right) + w_r \left(c_\psi s_\theta c_\phi + s_\psi s_\psi\right) + \mathsf{w}_n \\ u_r c_\theta s_\psi + v_r \left(c_\psi c_\phi + s_\psi s_\theta s_\phi\right) + w_r \left(s_\psi s_\theta c_\phi - c_\psi s_\phi\right) + \mathsf{w}_e \\ -u_r s_\theta + v_r c_\theta s_\phi + w_r c_\theta c_\phi + \mathsf{w}_d \\ v_r r - w_r q - \mathsf{g} s_\theta + \frac{\mathbf{f}_{xs}}{\mathsf{m}} - c_\psi c_\theta \dot{\mathsf{w}}_n - s_\psi c_\theta \dot{\mathsf{w}}_e + s_\theta \dot{\mathsf{w}}_d \\ w_r p - u_r r + \mathsf{g} s_\phi c_\theta + \frac{\mathbf{f}_{ys}}{\mathsf{m}} - \left(c_\psi s_\theta s_\phi - s_\psi c_\phi\right) \dot{\mathsf{w}}_n - \left(c_\psi c_\phi + s_\psi s_\theta s_\phi\right) \dot{\mathsf{w}}_e - c_\theta s_\phi \dot{\mathsf{w}}_d \\ u_r q - v_r p + \mathsf{g} c_\phi c_\theta + \frac{\mathbf{f}_{zs}}{\mathsf{m}} - \left(s_\psi s_\phi + c_\psi s_\theta c_\phi\right) \mathsf{w}_n - \left(s_\psi s_\theta c_\phi - c_\psi s_\phi\right) \mathsf{w}_e - c_\theta c_\phi \mathsf{w}_d \\ p + q s_\phi \tan\theta + c_\phi \tan\theta \\ q c_\phi - r s_\phi \\ \frac{q s_\phi}{c_\theta} + \frac{r c_\phi}{c_\theta} \\ \frac{J_{xz}\left(J_{xx} - J_{yy} + J_{zz}\right)pq - \left(J_{zz}\left(J_{zz} - J_{yy}\right) + J_{xz}^2\right)qr + J_{zz}\mathbf{l}_s + J_{xz}\mathbf{l}_s}{J_{xx}J_{zz} - J_{xz}^2} \\ \frac{\left(J_{zz} - J_{xx}\right)pr - J_{xz}\left(p^2 - r^2\right) + \mathbf{m}_s}{J_{yy}} \\ \frac{J_{xz}\left(J_{xx} - J_{yy} + J_{zz}\right)qr - \left(J_{xx}\left(J_{xx} - J_{yy}\right) + J_{xz}^2\right)pq + J_{xz}\mathbf{l}_s + J_{xx}\mathbf{n}_s}{J_{xx}J_{zz} - J_{xz}^2} \end{pmatrix} \tag{B.1}$$

Moreover, the vectors $\boldsymbol{g}_i$ are given by

$$
\begin{aligned}
\boldsymbol{g}_1 &= \begin{bmatrix} 0 & 0 & 0 & \mathsf{m}^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \\
\boldsymbol{g}_2 &= \begin{bmatrix} 0 & 0 & 0 & 0 & \mathsf{m}^{-1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \\
\boldsymbol{g}_3 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & \mathsf{m}^{-1} & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T, \\
\boldsymbol{g}_4 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{J_{zz}}{J_{xx}J_{zz}-J_{xz}^2} & 0 & \frac{J_{xz}}{J_{xx}J_{zz}-J_{xz}^2} \end{bmatrix}^T, \\
\boldsymbol{g}_5 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{J_{yy}} & 0 \end{bmatrix}^T, \\
\boldsymbol{g}_6 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{J_{xz}}{J_{xx}J_{zz}-J_{xz}^2} & 0 & \frac{J_{xx}}{J_{xx}J_{zz}-J_{xz}^2} \end{bmatrix}^T.
\end{aligned}
$$

First controller

For the first controller, we have that $\boldsymbol{\Lambda}(\boldsymbol{x})$ is given by

$$
\boldsymbol{\Lambda}(\boldsymbol{x}) = \begin{bmatrix}
\frac{c_\psi c_\theta}{\mathsf{m}} & \frac{c_\psi s_\theta s_\phi - s_\psi c_\phi}{\mathsf{m}} & \frac{s_\psi s_\phi + c_\psi s_\theta c_\phi}{\mathsf{m}} & 0 & 0 & 0 \\
\frac{s_\psi c_\theta}{\mathsf{m}} & \frac{c_\psi c_\phi + s_\psi s_\theta s_\phi}{\mathsf{m}} & \frac{s_\psi s_\theta c_\phi - c_\psi s_\phi}{\mathsf{m}} & 0 & 0 & 0 \\
\frac{-s_\theta}{\mathsf{m}} & \frac{c_\theta s_\phi}{\mathsf{m}} & \frac{c_\theta c_\phi}{\mathsf{m}} & 0 & 0 & 0 \\
0.0 & 0.0 & 0.0 & \frac{-s_\phi J_{xz}}{J_{xx}J_{zz}-J_{xz}^2} & \frac{c_\phi}{J_{yy}} & \frac{-s_\phi J_{xx}}{J_{xx}J_{zz}-J_{xz}^2} \\
0.0 & 0.0 & 0.0 & \frac{-s_\phi J_{xz}}{J_{xx}J_{zz}-J_{xz}^2} & \frac{c_\phi}{J_{yy}} & \frac{-s_\phi J_{xx}}{J_{xx}J_{zz}-J_{xz}^2} \\
0.0 & 0.0 & 0.0 & \frac{c_\phi J_{xz}}{c_\theta\left(J_{xx}J_{zz}-J_{xz}^2\right)} & \frac{s_\phi}{J_{yy}c_\theta} & \frac{c_\phi J_{xx}}{c_\theta\left(J_{xx}J_{zz}-J_{xz}^2\right)}
\end{bmatrix}. \tag{B.2}
$$

As for $\boldsymbol{\lambda}(\boldsymbol{x})$ we have

$$
\begin{bmatrix}
\mathsf{mg}s_\theta - f_{x\mathbf{s}} \\
-\mathsf{mg}s_\phi c_\theta - f_{y\mathbf{s}} \\
-\mathsf{mg}c_\phi c_\theta - f_{z\mathbf{s}} \\
\lambda_4 \\
\lambda_5 \\
\lambda_6
\end{bmatrix}, \tag{B.3}
$$

where λ_4, λ_5 and λ_6 are given by

$$
\begin{aligned}
\lambda_4 =& \frac{(-2qrJ_{xx}c_\phi^2 - J_{xx}s_\phi(q-r)(q+r)c_\phi + r(J_{xx}-J_{yy}+J_{zz})q - l_{\mathbf{s}})c_\theta)}{c_\theta} + \\
& \frac{\left(2rqc_\phi^3 + s_\phi(q^2-r^2)c_\phi^2 + q^2 s_\phi\right) J_{xz}s_\theta}{c_\theta}, \qquad (B.4) \\
\lambda_5 =& \frac{\left(-J_{xz}r^2 + p(J_{xx}+J_{yy}-J_{zz})r + J_{xz}p^2 - m_{\mathbf{s}}\right) c_\theta}{c_\theta} + \\
& \frac{J_{yy}s_\theta \left((q^2-r^2)c_\phi^3 - 2qrs_\phi c_\phi^2 + (-q^2+2r^2)c_\phi + 2qrs_\phi\right)}{c_\theta}, \qquad (B.5) \\
\lambda_6 =& \frac{(2qrJ_{xz}c_\phi^2 + s_\phi J_{xz}(q-r)(q+r)c_\phi - p(J_{xx}-J_{yy}+J_{zz})q - n_{\mathbf{s}})c_\theta)}{c_\theta} + \\
& \frac{-2s_\theta J_{zz}\left(qrc_\theta^3 + 0.5s_\phi c_\phi^2(q^r - r^2) + 0.5q^2 s_\phi\right)}{c_\theta}. \qquad (B.6)
\end{aligned}
$$

Second controller

For the second controller only the first block of $\boldsymbol{\Lambda}(\boldsymbol{x})$ changes, thus, we have

$$\boldsymbol{\Lambda}(\boldsymbol{x}) = \begin{bmatrix} \frac{-w_r}{(u_r^2+w_r^2)\mathsf{m}} & 0.0 & \frac{u_r}{(u_r^2+w_r^2)\mathsf{m}} & 0.0 & 0.0 & 0.0 \\ \frac{-v_r u_r}{v_t^3 \mathsf{m} c_\beta} & \frac{u_r^2+w_r^2}{v_t^3 \mathsf{m} c_\beta} & \frac{v_r w_r}{v_t^3 \mathsf{m} c_\beta} & 0.0 & 0.0 & 0.0 \\ \frac{u_r}{\mathsf{m} v_t} & \frac{v_r}{\mathsf{m} v_t} & \frac{w_r}{\mathsf{m} v_t} & 0.0 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & \frac{-s_\phi J_{xz}}{J_{xx}J_{zz}-J_{xz}^2} & \frac{c_\phi}{J_{yy}} & \frac{-s_\phi J_{xx}}{J_{xx}J_{zz}-J_{xz}^2} \\ 0.0 & 0.0 & 0.0 & \frac{-s_\phi J_{xz}}{J_{xx}J_{zz}-J_{xz}^2} & \frac{c_\phi}{J_{yy}} & \frac{-s_\phi J_{xx}}{J_{xx}J_{zz}-J_{xz}^2} \\ 0.0 & 0.0 & 0.0 & \frac{c_\phi J_{xz}}{c_\theta\left(J_{xx}J_{zz}-J_{xz}^2\right)} & \frac{s_\phi}{J_{yy}c_\theta} & \frac{c_\phi J_{xx}}{c_\theta\left(J_{xx}J_{zz}-J_{xz}^2\right)} \end{bmatrix}. \tag{B.7}$$

For the vector $\boldsymbol{\lambda}$ the same logic applies

$$\boldsymbol{\lambda}(\boldsymbol{x}) = \begin{bmatrix} \mathsf{mg}s_\theta + \mathsf{m}qw_r - \mathsf{m}rv_r - f_{x\mathsf{s}} \\ -\mathsf{mg}s_\phi c_\theta - \mathsf{m}pw + r + \mathsf{m}ru_r - f_{y\mathsf{s}} \\ -\mathsf{mg}c_\phi c_\theta + \mathsf{m}pv_r - \mathsf{mg}qu_r - f_{z\mathsf{s}} \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \end{bmatrix}, \tag{B.8}$$

where λ_4, λ_5 and λ_6 are given by (B.4)-(B.6).

B.2 EKF matrices

In this section we present the covariance matrix used in the EKF filter applied in this work. For the process covariance matrix $\boldsymbol{\Gamma}_p$, we have

$$\begin{aligned} \boldsymbol{\Gamma}_p &= \begin{bmatrix} \boldsymbol{\Gamma}_{p_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Gamma}_{p_2} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Gamma}_{p_3} \end{bmatrix}, \\ \boldsymbol{\Gamma}_{p_1} &= \operatorname{diag}\left(\begin{bmatrix} 0 & 0 & 0 \end{bmatrix}\right), \\ \boldsymbol{\Gamma}_{p_2} &= \operatorname{diag}\left(\begin{bmatrix} 6.25 \times 10^{-6}\mathrm{g}^2 & 6.25 \times 10^{-6}\mathrm{g}^2 & 6.25 \times 10^{-6}\mathrm{g}^2 \end{bmatrix}\right), \\ \boldsymbol{\Gamma}_{p_3} &= \operatorname{diag}\left(\begin{bmatrix} 6.76 \times 10^{-6} & 6.76 \times 10^{-6} & 6.76 \times 10^{-6} \end{bmatrix}\right). \end{aligned}$$

For the observation noise covariance matrix, we have

$$\boldsymbol{\Gamma}_o = \operatorname{diag}\left(\begin{bmatrix} 0.05 & 0.05 & 0.2 \end{bmatrix}\right).$$

Finally, the covariance matrix $\mathbf{\Gamma}_e$ is obtained by exhaustively iterating the EKF algorithm until the desired converge is obtained. In this work, we iterated 10^4 times to obtain

$$\mathbf{\Gamma}_e = \begin{bmatrix} 7.49e-3 & -9.17e-4 & 3.33e-3 & 4.12e-3 & -2.70e-3 & -7.75e-4 & -5.35e-5 & -1.16e-4 & -1.70e-6 \\ -9.17e-4 & 1.04e-2 & 1.00e-3 & 1.44e-3 & 7.88e-3 & -9.46e-4 & 1.97e-4 & -4.19e-5 & 3.77e-5 \\ 3.33e-3 & 1.00e-3 & 2.35e-2 & 4.38e-3 & 1.27e-3 & 2.59e-3 & -2.81e-6 & -3.32e-4 & -3.82e-5 \\ 4.12e-3 & 1.44e-3 & 4.38e-3 & 4.35e-3 & 1.35e-3 & -1.29e-3 & 1.89e-5 & -1.67e-4 & -3.92e-5 \\ -2.70e-3 & 7.88e-3 & 1.27e-3 & 1.35e-3 & 5.70e-2 & -4.90e-3 & 8.80e-4 & -5.16e-5 & -1.46e-3 \\ -7.75e-4 & -9.46e-4 & 2.59e-3 & -1.29e-3 & -4.90e-3 & 3.37e-3 & -7.12e-5 & 4.95e-5 & 1.35e-4 \\ -5.35e-5 & 1.97e-4 & -2.81e-6 & 1.89e-5 & 8.80e-4 & -7.12e-5 & 3.12e-5 & -6.57e-7 & -1.68e-5 \\ -1.16e-4 & -4.19e-5 & -3.32e-4 & -1.67e-4 & -5.16e-5 & 4.95e-5 & -6.57e-7 & 1.48e-5 & 1.53e-6 \\ -1.70e-6 & 3.77e-5 & -3.82e-5 & -3.92e-5 & -1.46e-3 & 1.35e-4 & -1.68e-5 & 1.53e-6 & 5.03e-5 \end{bmatrix}$$