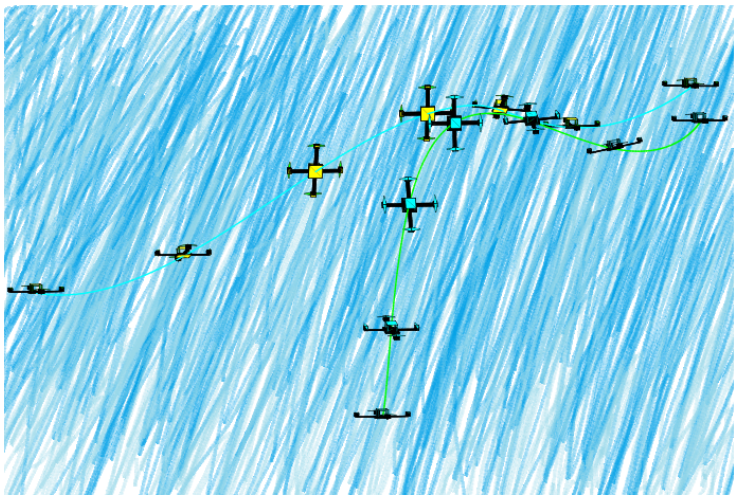
TÉCNICO LISBOA



# Model Predictive Control Strategies for Aggressive Parcel Relay Maneuvers Using Drones

**João Daniel Barata da Silva Pinto**

Thesis to obtain the Master of Science Degree in

## Electrical and Computer Engineering

Supervisors: Prof. Bruno João Nogueira Guerreiro
Prof. Rita Maria Mendes de Almeida Correia da Cunha

### Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Prof. Bruno João Nogueira Guerreiro
Member of the Committee: Prof. João Manuel Lage de Miranda Lemos

**September 2021**

## Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

## Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of Universidade de Lisboa.

# Acknowledgements

First and foremost, I would like to express my sincerest words of gratitude to my dissertation advisers, Prof. Bruno Guerreiro and Prof. Rita Cunha. I was not only privileged to work with proficient researchers in modelling and control of aerial vehicles, but also to be thoughtfully guided during the execution of this thesis. Furthermore, I am thankful for being given the opportunity to be a part of the research team of project REPLACE. I was involved in an innovative project that challenged me intellectually and had the chance to collaborate with exceptional people. Thank you, Tomás Bastos and Dr. Parth Thakar, for your suggestions and words of encouragement, and Gil Serrano, for providing any help when necessary. I would also like to acknowledge the contribution of Tiago Oliveira, who shared with me his experience of implementing and testing experimentally controllers for multirotors, and José Tojeira, who would not hesitate to provide assistance to conduct experiments.

In addition, I would like to thank my colleagues at DSOR lab, namely Bruno Ribeiro, Pedro Lima, and Thomas Berry, for their friendship and the socially-distanced fun we had together – I was always buzzing for those weekly paddle tennis matches.

A Master's thesis is only the tip of the iceberg, many years of meticulous work culminated in this document. Even though one never stops learning in this life, I am truly marvelled at how much my knowledge has expanded over the past few years. For this reason, I am immensely grateful to all professors and colleagues who shared their knowledge with me along this journey at Instituto Superior Técnico. A special thanks goes, once again, to Prof. Bruno Guerreiro, for his generosity and altruism, and to my dear friends, Duarte Correia and Renato Dias, with whom various nights were spent over work, for their sheer optimism, good spirit, and for always encouraging me to be the best version of myself. I could definitely not forget to mention the AeroTéc student group and, in particular, project UAV-ART, for all the memorable experiences, the extraordinary team spirit I shall forever cherish, the knowledge and skills I acquired as a member of this project, and for the friendships forged – Cheers, my great friends, João Canas and Luís Rodrigues!

My final acknowledgements go to my family – I am inestimably thankful for your support and for always being extremely kind to me. My very, very special thanks goes to my uncle Augusto, for his invaluable words of advice and for accompanying me throughout my academic life, and, of course, to my mother, for being an example of integrity and perseverance, a beacon in every moment of my life, for being with me through thick and thin.

This work was partially funded by the FCT projects CAPTURE (PTDC/EEI-AUT/1732/2020) and REPLACE (LISBOA-01-0145-FEDER-032107), which include Lisboa 2020 and PIDDAC funds, and also project LARSYS (UIDB/EEA/50009/2020).

# Resumo

Esta dissertação tem como objetivo o planeamento ótimo de manobras de estafeta para multirotores, de modo a trocarem uma encomenda em voo, e o desenho de controladores que permitam a realização destas manobras, tendo por base técnicas de controlo avançado, nomeadamente, controlo preditivo baseado em modelos. Começa-se por apresentar um método para planear manobras de estafeta agressivas, baseado na geração de trajetórias polinomiais com restrições na aceleração dos veículos, por forma a coordenar a atitude das aeronaves. Um problema de otimização inteira mista é responsável por gerar estas trajetórias para dois drones, no qual variáveis binárias asseguram que se evitam colisões entre os veículos. De seguida, é introduzido um controlador não-linear para segui-las e ação integral é considerada para mitigar as interações aerodinâmicas entre multirotores. Um controlador preditivo baseado em modelos é proposto para controlar a translação de um multirotor e a estabilidade exponencial do sistema em malha em fechada é comprovada. O primeiro controlador é validado ao realizar manobras de estafeta entre dois multirotores em ambiente de simulação, que tem em conta o software do piloto automático, e pela realização de testes experimentais preliminares; o segundo, pela realização de testes de simulação. São também apresentados resultados experimentais com vista à análise das interações aerodinâmicas entre dois veículos.

**Palavras-chave:** Planeamento de trajectórias, Manobras de estafeta, Controlo Preditivo baseado em modelos, Multirotores, Estabilidade

## Abstract

This dissertation addresses the planning of optimal relay manoeuvres using multirotors, in order to exchange a parcel during flight, and the design of controllers which are able to execute these manoeuvres, based on advanced control techniques, namely model predictive control. Firstly, a method for planning aggressive manoeuvres is presented, which uses polynomial trajectories with acceleration constraints, to coordinate the attitude of the vehicles during the package exchange. An optimisation problem to generate these trajectories for two vehicles is formulated as a mixed-integer program, where the integer constraints ensure inter-vehicle collision avoidance. Then, a non-linear controller is introduced to track such trajectories and integral action is incorporated to mitigate the effects of the aerodynamic interactions between multirotors. A model predictive controller for the translation control loop of a multirotor is proposed and conditions for exponential stability of the closed-loop system are determined. The first controller is validated by executing relay manoeuvres involving two multirotors in a simulation environment, which includes the autopilot software, and by means of preliminary experimental results; while the second is validated through simulations. Further experimental results tailored at analysing the aerodynamic interactions between two vehicles are also presented.



**Keywords:** Trajectory Planning, Relay Manoeuvres, Model Predictive Control, Multirotors, Stability

# Contents


Acknowledgements . . . . . v
Resumo . . . . . vii
Abstract . . . . . ix
List of Tables . . . . . xiii
List of Figures . . . . . xv
Nomenclature . . . . . xvii
Acronyms . . . . . xix

**1 Introduction** **1**
1.1 Context and Motivation . . . . . 1
1.2 Problem Statement . . . . . 2
1.3 Proposed Solution and Contributions . . . . . 3
1.4 Outline . . . . . 4
1.5 Notation . . . . . 5

**2 Related Work** **7**
2.1 Trajectory Planning . . . . . 7
2.2 Trajectory Tracking . . . . . 8
2.3 3-D Attitude Tracking Control . . . . . 9
2.4 Aerodynamics Modelling . . . . . 9

**3 Theoretical Background** **11**
3.1 Model Predictive Control . . . . . 11
3.1.1 Stability . . . . . 14

**4 Planning for Aggressive Manoeuvres** **19**
4.1 Single Quadrotor . . . . . 19
4.2 Planning for Two Quadrotors . . . . . 21
4.2.1 Computational Complexity and Numerical Algorithm . . . . . 22
4.2.2 Distance Measurement between Quadcopters . . . . . 23
4.2.3 Trajectory Generation . . . . . 25
4.3 Relay Manoeuvres Planning . . . . . 27

**5 Quadrotor Modelling and Control** **31**
5.1 Model . . . 31
5.2 Aerodynamic Interactions between Rotors . . . 32
5.3 Nonlinear Control on $\mathrm{SE}(3)$ . . . 33

**6 Model Predictive Control** **37**
6.1 MPC Regulator . . . 37
6.2 Stability Discussion . . . 42
6.3 Region of Attraction . . . 44

**7 Implementation, Results, and Discussion** **47**
7.1 Test Space and Off-board Control . . . 47
7.2 SITL Relay Planning Results . . . 51
7.3 Experimental Relay . . . 56
7.4 MPC Relay Results . . . 58
7.5 Aerodynamic Interactions . . . 64

**8 Conclusions** **69**
8.1 Future Work . . . 70

**Bibliography** **73**

**A Convex Optimisation** **78**
A.1 Mixed-Integer Programming . . . 80

**B Basic Stability Theory** **83**
B.1 Lyapunov Stability Theory . . . 84

**C NMPC Regulator Closed-Loop Stability** **87**

**D Mean Value Theorem for Vector Fields** **91**

**E Planning as a QP Problem** **93**

# List of Tables

4.1 Trajectory generation performance results excluding and including binary constraints. . . 26

# List of Figures


1.1 Illustration of two drones performing an acrobatic package exchange manoeuvre. . . . . 2
1.2 Illustration of the problem tackled in this dissertation. . . . . 3

4.1 Example of a polyhedron around a quadcopter. . . . . 23
4.2 Illustration of the distance between two polyhedrons. . . . . 24
4.3 Schematic representation of the established waypoints. . . . . 25
4.4 Position reference trajectories generation excluding and including binary constraints. . . . 26
4.5 Distance between reference trajectories excluding and including binary constraints. . . . . 26
4.6 Diagram illustrating the segments of the trajectory in order to reach the goal state $G$. . . . 27

5.1 Reference frames and rotor numbering. . . . . 31
5.2 Illustration of blade tip vortexes at different flight conditions. . . . . 33
5.3 Geometric tracking controller on $\mathrm{SE}(3)$ structure. . . . . 34

6.1 MPC controller coupled with a geometric geometric attitude controller. . . . . 41

7.1 Flying arena used to conduct experimental trials. . . . . 48
7.2 Multirotors considered to run SITL simulations and experimental tests. . . . . 48
7.3 Simplified diagram of the drone control code architecture. . . . . 49
7.4 Thrust vs throttle curves for the INTEL® Aero Ready to Fly and 3DR® Iris quadrotors. . . 50
7.5 Time axis representing the duration of the every phase of the SITL roll relay. . . . . 51
7.6 Schematic representation of the waypoints established for the SITL roll relay. . . . . 52
7.7 Position evolution of both vehicles during the SITL roll relay. . . . . 52
7.8 Velocity evolution of both vehicles during the SITL roll relay. . . . . 53
7.9 Orientation evolution of both vehicles during the SITL roll relay. . . . . 54
7.10 Angular velocity evolution of both vehicles during the SITL roll relay. . . . . 55
7.11 Throttle evolution of both vehicles during the SITL roll relay. . . . . 55
7.12 Distance between the quadcopters during the SITL roll relay. . . . . 55
7.13 Schematic representation of the waypoints established for the Experimental roll relay. . . 56
7.14 Illustration of the vehicles tracking the reference of the Experimental roll relay (xy plane). . 57
7.15 Illustration of the vehicles tracking the reference of the Experimental roll relay (yz plane). . 57
7.16 Orientation evolution of both vehicles during the Experimental roll relay. . . . . 58

7.17 Distance between the quadcopters during the Experimental roll relay. . . . . 58
7.18 Schematic representation of the waypoints established for the MPC roll relay. . . . . 59
7.19 Position evolution of both vehicles during the MPC roll relay. . . . . 59
7.20 Velocity evolution of both vehicles during the MPC roll relay. . . . . 60
7.21 Orientation evolution of both vehicles during the MPC roll relay. . . . . 61
7.22 Thrust evolution of both vehicles during the MPC roll relay. . . . . 62
7.23 Angular velocity evolution of both vehicles during the MPC roll relay. . . . . 62
7.24 Distance between the quadcopters during the MPC roll relay. . . . . 63
7.25 Illustration of the vehicles tracking the reference of *Phase 2* of the MPC roll relay. . . . . 63
7.26 Computation time during the MPC roll relay. . . . . 63
7.27 Position evolution of both vehicles in close proximity. . . . . 64
7.28 Position tracking error of both vehicles in close proximity. . . . . 65
7.29 Velocity evolution of both vehicles in close proximity. . . . . 65
7.30 Orientation evolution of both vehicles in close proximity. . . . . 66
7.31 Angular velocity of both vehicles in close proximity. . . . . 67

# Nomenclature

### Greek symbols

| | |
|---|---|
| $\boldsymbol{\omega}$ | Angular velocity vector. |
| $\boldsymbol{\sigma}$ | Vector of flat outputs. |
| $\boldsymbol{\tau}$ | Body torques vector. |
| $\phi$ | Roll Euler angle. |
| $\psi$ | Yaw Euler angle. |
| $\theta$ | Pitch Euler angle. |

### Roman symbols

| | |
|---|---|
| $\boldsymbol{R}$ | Rotation Matrix. |
| $\boldsymbol{u}$ | Control inputs vector. |
| $\boldsymbol{v}$ | Linear velocity vector. |
| $\boldsymbol{p}$ | Position vector. |
| $g$ | Earth's gravity. |
| $m$ | Quadrotor mass. |
| $T$ | Thrust. |
| $t$ | Time. |

### Subscripts

| | |
|---|---|
| $B$ | Body frame. |
| $des$ | Desired reference. |
| $T$ | Trajectory. |
| $x, y, z$ | Cartesian components. |

### Superscripts

| | |
|---|---|
| $T$ | Transpose. |

# Acronyms

**DP** Dynamic Programming

**ISR** Institute for Systems and Robotics

**LQR** Linear Quadratic Regulator

**MAV** Micro Aerial Vehicle

**MCS** Motion Capture System

**MIP** Mixed Integer Program

**MIQP** Mixed Integer Quadratic Program

**MPC** Model Predictive Control

**NMPC** Nonlinear Model Predictive Control

**OCP** Optimal Control Problem

**PD** Proportional Derivative

**PID** Proportional Integral Derivative

**QP** Quadratic Program

**SITL** Software in the Loop

**UAV** Unmanned Aerial Vehicle

**VRS** Vortex Ring State

**VTOL** Vertical Take-Off and Landing

# Chapter 1

# Introduction

In the past few years, the impressive technological developments in a vast array of fields of study, tied to the commitment of the scientific community to devise and build unmanned aerial vehicles (UAVs), triggered a steady, continuous growth of the application domain for these vehicles. Amongst the large list of applications suited for the use of drones, establishing a drone delivery service is definitely one of the hottest.

## 1.1 Context and Motivation

The versatility of drones, specially those which fall into the category of micro aerial vehicles (MAVs), soon caught the eye of the logistics and e-commerce *giants* [1] and of many other enterprises. Driven by the ambition to reduce the time and cost associated with parcel delivery, multiple small-scaled unmanned aircraft - fixed-wing, multirotors, tilt-rotors, and tilt-wings - have been developed. Numerous experimental tests, ranging from preliminary to several months long trials, have been conducted and some delivery services using UAVs began to be established in the 2010s.

Since the end of 2013, the international courier *DHL* together with *Wingcopter* successfully designed and validated, by means of challenging trials, three different fully autonomous vertical take-off and landing (VTOL) MAVs to primarily deliver medication to remote locations [2]. In 2016, *Zipline* launched a remarkably reliable fixed-wing drone delivery service aiming at ensuring the timely dispatch of blood products to hospitals in rural areas in Rwanda [3, 4]. By the end of the same year, *Amazon* accomplished the first airborne parcel delivery and, since then, further aircraft designs have been introduced [5]. Soon after, in 2017, *UPS* tested the concept of performing deliveries using human-piloted trucks and autonomous multirotors [6]. In the following years, as new legislation regulating the use of civilian MAVs in various countries was developed, more drone delivery services were established over many spots in the world.

From an academic point of view, automating parcel delivery resorting to drones presents itself as an excellent opportunity to further delve into the study of the interaction between aerial robots and the surrounding environment. In fact, load transportation using MAVs, in particular multirotors, for their me-

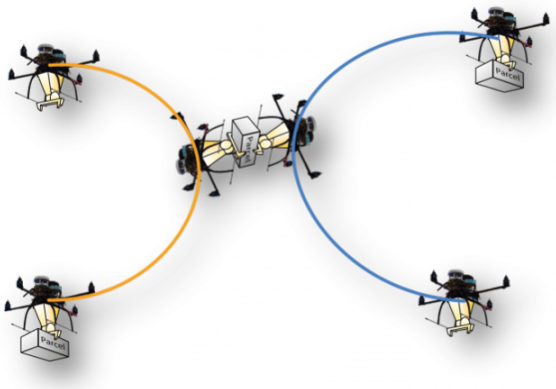


Figure 1.1: Illustration of two drones performing an acrobatic package exchange manoeuvre (from [12]).

chanical simplicity, compact size, ability to carry a diverse set of sensors and a significant payload, and high manoeuvrability, has been subject of much scrutiny [7]. Furthermore, one can find in the existing literature studies which address specifically the problem of designing adequate gripping mechanisms for aerial grasping [8, 9].

Despite the conducted research enabling the employment of multirotors for package delivery, there are usually stringent limitations on the range and endurance of a single vehicle. Acknowledging this fact, task scheduling and path planning strategies have been proposed for teams of cooperative autonomous drones and trucks [10]. In these approaches, the multirotors are deployed from a moving truck and are only responsible for handling the last leg of the delivery. There seems not to exist that many examples of drone-only delivery strategies in the literature, except for a few examples [11].

Project REPLACE aims at tackling the many scientific and technological challenges with regard to swift package delivery in urban environments using solely MAVs [12]. In order to overcome the short battery autonomy of a single drone, this project proposes the development of heterogeneous teams of autonomous quadrotors and VTOL aeroplanes which cooperate by performing in-flight parcel relay manoeuvres, thus enabling an effective, expeditious package delivery.

This dissertation was developed within the scope of project REPLACE and, as illustrated in Figure 1.1, the motivation behind this research effort is the design of optimal aggressive relay manoeuvre trajectories for quadrotors, in the sense that agile motion and high speed navigation is demanded from the aircraft, and control strategies which ensure a faithful tracking of these trajectories.

## 1.2 Problem Statement

The problem to be addressed is stated by assuming there are two quadrotors equipped with a gripping mechanism attached to their bottom. One of the vehicles is carrying a package, while the other is not. Given a *relay point*, that is, a point in a global frame where the package is to be transferred, the two rotorcraft must head to that point while, simultaneously, adjusting their relative orientation, in order to exchange a parcel, and avoiding collisions between them. Let Figure 1.2 illustrate this idea.

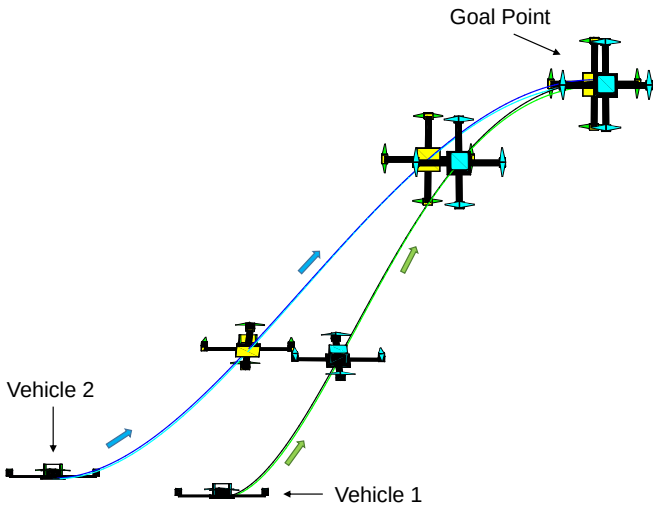



Figure 1.2: Illustration of the problem tackled in this dissertation.

Vehicle 1 and vehicle 2 start at given positions and move towards a goal point, a goal configuration in state-space, which must be reached by both aircraft at the same time instant. At this goal configuration, both vehicles must achieve a suitable orientation and angular velocities to ensure a successful parcel transfer. Ideally, both drones should be in an appropriate configuration for a sufficiently long time window to increase the chances of effectively exchanging a package between the two aircraft. The problem of controlling a mechanism responsible for holding and transferring packages between vehicles falls out the scope of this thesis.

In what regards the challenges associated with the aircraft being close proximity to relay a parcel during flight, beyond the real possibility of vehicles colliding with one another, the aerodynamic interactions between the aircraft are of particular concern. The turbulent downwash generated by the rotating propellers of one aircraft may affect negatively the performance of another aircraft which is subject to this airflow, leading to potential instability.

## 1.3 Proposed Solution and Contributions

The problem posed in the previous section is addressed by proposing a strategy for planning aggressive collision-free parcel relay manoeuvres for quadrotors, which ensures that the vehicles involved in the manoeuvre are in a suitable relative orientation to exchange a parcel during flight. This strategy relies on the generation of optimal polynomial trajectories with acceleration constraints to embed orientation constraints into the trajectory planning problem and integer constraints are considered to formulate convex collision avoidance constraints.

The delineated strategy for planning relay manoeuvres is validated through software-in-the-loop (SITL) simulations and experimental results involving a non-linear controller on the Special Euclidean Group $\mathrm{SE}(3)$ to track the optimal pre-planned relay trajectories. Considering this controller, the vehicles involved in the manoeuvre effectively follow these trajectories as the controller is designed taking into account the intrinsic geometry of the manifold where the quadrotor dynamics evolve, $\mathrm{SE}(3)$.

In order to run SITL simulations and experimental tests, a testing software environment, which is available in a public code repository [13], was developed to implement trajectory tracking controllers for multirotors equipped with the PX4 autopilot [14] leveraging on the features of the ROS middleware [15] and the MAVROS package [16]. Additional software was also developed to generate synthetic Motion Capture System (MCS) data with the objective of creating a simulation environment in Gazebo [17] which resembles closer an indoor flying arena featuring an MCS [18].

Moreover, a model predictive controller is introduced to handle simultaneously the planning of state and control trajectories. This controller is employed to control the translation dynamics of a multirotor, which are modelled as a triple integrator. A thorough stability analysis of the closed-loop system is provided in this dissertation and the conditions to establish exponential stability are also presented.

The MPC regulator for the translation control loop of the multirotor is then coupled with a geometric controller on the attitude manifold. Simulation results demonstrate that the combination of these two controllers enables the execution of aggressive manoeuvres, such as those which are required to exchange packages in mid-air, and in the sense that high linear and angular velocities are demanded from the rotorcraft.

In view of some considerations on rotor aerodynamics theory, the execution of an experimental trial in which two vehicles were commanded to hover one below the other, and the results obtained from conducting a preliminary experimental relay manoeuvre, the drones benefit from travelling at some speed as the negative effects of being in close proximity are mitigated.

Based on the work conducted for this dissertation, a paper regarding the strategy for planning parcel relay manoeuvres for multirotors was published in the proceedings of the 2021 International Conference on Unmanned Aircraft Systems (ICUAS'21) [19].

## 1.4 Outline

The remainder of this thesis is structured as follows. Chapter 2 presents the state-of-the art regarding trajectory design and control for quadrotors, model predictive control, and aerodynamic modelling. Chapter 3 provides the theoretical background on the necessary concepts to consider throughout this dissertation. In Chapter 4, the strategy for planning parcel relay manoeuvres is explained and polynomial trajectory generation algorithms are introduced. A model of the dynamics of the quadrotor is described and a non-linear trajectory tracking controller is presented in Chapter 5. Chapter 6 details a model predictive controller to control the translation of a quadrotor, including a stability analysis of the closed-loop system. In Chapter 7, the implementation and the setup used for testing controllers in SITL simulations and experimental trials are described; simulation and experimental results of two drones performing a relay manoeuvre are presented to validate the strategy for planning relay manoeuvres and the MPC controller proposed in Chapter 6; the experimental results of a test to analyse the aerodynamic interactions between two multirotors are also discussed. Finally, in Chapter 8, some concluding remarks are made and suggestions for future work are provided.

## 1.5 Notation

In this section, the mathematical notation used throughout this thesis is introduced. Here are some of the choices made:

- Vectors are represented by bold lowercase letters, $\boldsymbol{q}$ and matrices, by bold uppercase letters, $\boldsymbol{Q}$; scalars are represented by both plain, not bold, lower and uppercase letters, e.g., $q$, $Q$;
- The set of real numbers is denoted by $\mathbb{R}$, the subset of positive real numbers is denoted by $\mathbb{R}^+$, and the set of non-negative real numbers by $\mathbb{R}_0^+$; $\mathbb{R}^n$ denotes the $n$-dimensional Euclidean space, with norm $\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x} \cdot \boldsymbol{x}}$, where $\boldsymbol{a} \cdot \boldsymbol{b} \coloneqq \boldsymbol{a}^T \boldsymbol{b}$ denotes the inner product between two vectors; $\mathbb{R}^{n \times m}$ is the set of $n \times m$ matrices with real entries;
- The set of integer numbers is denoted by $\mathbb{Z}$ and $\mathbb{Z}^k$ is the set of vectors in $\mathbb{R}^k$ with only integer entries; the set of natural numbers is denoted by $\mathbb{N}$ and $\mathbb{N}_0 \coloneqq \mathbb{N} \cup \{0\}$ is the set of natural numbers including zero;
- The symbol $\boldsymbol{I}_n$ is used to denote the $n \times n$ identity matrix, and $\boldsymbol{0}_{m \times n}$, an $m \times n$ matrix whose entries are all equal to zero (when it is possible to deduce the dimensions of these matrices from context, the subscripts are omitted);
- The Special Euclidean Group in three dimensions $\mathrm{SE}(3)$ is defined as the semi-direct product of $\mathbb{R}^3$ and the Special Orthogonal Group in three dimensions $\mathrm{SO}(3) \coloneqq \{\boldsymbol{R} \in \mathbb{R}^{3 \times 3} \mid \boldsymbol{R}^T \boldsymbol{R} = \boldsymbol{I},\ \det(\boldsymbol{R}) = 1\}$;
- $\boldsymbol{S}(\cdot)$ is a map from $\mathbb{R}^3$ to $\mathfrak{so}(3)$, the space of three-by-three skew-symmetric matrices, defined by

$$\boldsymbol{S}(\boldsymbol{a}) = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix},\ \boldsymbol{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T. \tag{1.1}$$

  Note that $\mathfrak{so}(3)$ and $\mathbb{R}^3$ are isomorphic. This isomorphism is defined by the condition $\boldsymbol{S}(\boldsymbol{a})\boldsymbol{b} = \boldsymbol{a} \times \boldsymbol{b}$ for all $\boldsymbol{a},\ \boldsymbol{b} \in \mathbb{R}^3$.

# Chapter 2

# Related Work

This chapter includes the relevant work review on trajectory design and control of quadcopters, entering in further detail for aggressive manoeuvres and coordination of several aircraft. Several examples of MPC strategies in the literature applied to drones are presented. Furthermore, due to the proximity of the vehicles in the relay manoeuvre, a brief review the existing literature on aerodynamic modelling for quadrotors is provided.

## 2.1 Trajectory Planning

The subject of optimal trajectory design for quadrotors is an active topic of research. In the literature, there are several examples of polynomial and discrete trajectories, some including constraints for obstacle avoidance. While Model Predictive Control (MPC) is used for generating trajectories online, some authors take other approaches as addressed in this section.

*Mellinger et al.* have proposed various algorithms for generating trajectories for quadrotors. In [20], trajectories are defined as a sequence of segments each associated to a specific controller. The purpose of the composition of multiple trajectory segments is to enable aggressive manoeuvres such as flying through narrow gaps. Another algorithm formulates a Quadratic Program (QP) to obtain optimal polynomial trajectories that link established waypoints. These trajectories are optimal in the sense that the cost functional to be minimised is derived from the square norm of the snap [21] (fourth derivative of position). So as to allow trajectories to be of higher polynomial order, the authors used a basis of Legendre polynomials [22]. Moreover, collision avoidance constraints are introduced as integer constraints. The problem of generating trajectories for multiple heterogeneous drones is addressed using a centralised approach. At the end, it is acknowledged that this method is not scalable to a large team of quadcopters.

In the literature, mixed-integer linear programming (MILP) has been used for trajectory planning for single and multiple fixed-wing aircraft where integer constraints are for collision avoidance. Despite the linearity of the program enforcing the constraints relative to the dynamics to be linearised, or using other model approximations for ensuring linearity, various collision avoidance problems are successfully

solved in simulation [23]. *Schouwenaars et al.* established a MILP formulation for the problem trajectory planning using MPC techniques [24]. The authors in this study concluded that MPC can lead the aircraft to unsafe conditions.

An algorithm for generating discrete trajectories for a quadcopter fleet including avoidance constraints is presented in [25]. The optimisation problem formulated is non-convex and, therefore, Sequential Convex Programming (SCP) is employed to approximate non-convex constraints by convex ones. For the trajectory generation, a simplified dynamics model of the vehicle is considered, while *a posteriori* a feasibility check is performed on the interpolated trajectory. The experimental results reveal the algorithm running in real-time.

Some other studies, for instance [26], use a hybrid dynamic framework in order to decompose complex acrobatic flight into sequences of discrete manoeuvres. The safety of transition between modes is assured by means of reachable sets.

## 2.2 Trajectory Tracking

There is a myriad of tracking controllers developed, in particular for multirotors. One can find various studies which focus on designing controllers capable of tracking complex and intricate trajectories.

The nonlinear tracking controller introduced in [27] was directly developed on the Special Euclidean Group $\mathrm{SE}(3)$ and, therefore, singularities associated to other attitude representations of the quadcopter are avoided. One important contribution of this research is the definition of tracking errors which are employed in many other controllers, being the following one of those.

The work presented in [21] leverages the differential flatness of the quadrotor dynamics to develop a controller for tracking trajectories defined on the space of *flat* outputs. In fact, this tracking controller is very similar to the one proposed in [27], however it drops some of the somewhat stringent stability restrictions and feedforward terms that are difficult to implement in real vehicles. It is shown that this latter controller yields impressively good tracking performance.

A thorough comparison between linear and nonlinear MPC controllers is presented in [28]. It is concluded that, for the experiments conducted, the NMPC controller has an overall better tracking performance. In this article, an interesting point is also made on the two controllers discussed above, as they are not able to guarantee any state or input constraints.

As for other MPC controllers designed for trajectory tracking, the authors in [29] approach this problem by parametrising input and state trajectory so as to retain an infinite predictive horizon of the optimal control problem. In this solution, a reference trajectory is defined and tracked in the parameter space. Moreover, an iterative learning scheme is implemented, using a Kalman Filter, to estimate repeatable disturbances along a trajectory. It was demonstrated that the tracking performance improves over trials.

Finally, a prioritisation control inputs strategy to cope with motor saturation [30] in order to improve the tracking performance and the stabilisation of the aircraft is introduced. The experimental results for an aggressive trajectory reveal that the position tracking error is considerably reduced.

## 2.3 3-D Attitude Tracking Control

The problem of controlling the attitude of a quadcopter is of the uttermost importance, particularly in applications where the vehicle may undergo extreme attitude changes and high manoeuvrability is demanded. One of the challenges of this problem is how the attitude of the drone is represented. The attitude of any rigid-body is represented by a rotation matrix which belongs to Special Orthogonal Group $\mathrm{SO}(3)$. For this manifold being not Euclidean, the classical approach consists in linearising the nonlinear attitude dynamics around some condition (usually the hovering flight condition for quadcopters), and designing controllers based on a set of Euler angles to represent orientation [20, 31]. In addition to this strategy not allowing to perform agile and aggressive manoeuvres, Euler angles suffer from ambiguities in representing attitude and are kinematically singular.

In the modern literature unit quaternions are frequently adopted as a mean of parameterising rotation matrices [32, 33]. Quaternions are vectors which lie on the three dimensional unit sphere $\mathbb{S}^3$ and provide a global representation of the attitude of a rigid-body. However, the map from $\mathbb{S}^3$ to $\mathrm{SO}(3)$ is not unique, as there is always a pair of unit quaternions which correspond to the same physical attitude. Neglecting this issue may give rise to undesired phenomena such as *unwinding*, where the rigid-body rotates unnecessarily through a full rotation [34, 35].

There is also an increasing number of studies that motivated by the development of continuous and singularity-free control laws propose almost global asymptotically stable nonlinear attitude tracking controllers on the manifold of rotation matrices [21, 27, 36, 37]. One fundamental aspect, which is discussed in [34, 37], to be considered when tackling the problem of designing a controller on $\mathrm{SO}(3)$ for stabilising an desired rigid body attitude, is that is not possible to achieve global asymptotic stability with a continuous time-invariant feedback control law. Model predictive control (MPC) strategies for dynamical systems that evolve on $\mathrm{SO}(3)$ not only in the context of spacecraft attitude control [38], but also in the context of multirotor control [39] have also been proposed in the literature.

## 2.4 Aerodynamics Modelling

Most of the complexity of aerodynamic modelling for quadrotor vehicles is ignored. In fact, most often only the steady-state thrust and reaction torque of a rotor hovering in free space are modelled. Evidently, when the vehicle is out of these conditions secondary aerodynamic effects are introduced.

In [40], aerodynamic phenomena are studied for quadrotors. It is studied how the thrust produced by a propeller varies in many flight conditions, namely, proximity to horizontal surfaces (*ground effect*) and to other quadcopters. For the latter, the authors only investigate the effect for nearby vehicles in same horizontal plane. Some of these authors in [41] quantify empirically the effect of the downwash produced by the rotors of one drone on another flying below the first. The results from these experiments are used to define 'safe regions' around each vehicle in a team setting. A team of drones is also coordinated in [22] and constraints to disallow any drone from flying below the downwash of a similar-sized or larger quadcopter are included.

In [42], the downwash generated by a quadrotor is modelled using Momentum Theory with the objective of having a vehicle flying underneath another. The controller of the quadcopter incorporates an recursive Bayes filter in order to estimate the parameters of the downwash flow field. Simulation results reveal a drone hovering one meter below another successfully. Nevertheless, despite the satisfactory results, as the propeller diameter of the quadrotor is unknown and no other results are disclosed, these are somewhat inconclusive. Moreover, the authors acknowledge explicitly that the rotors of the drone below, in a real setting, would likely be put in Vortex Ring State [43], as it was not modelled. As a result, the quadrotor may become unstable.

# Chapter 3

# Theoretical Background

This chapter presents some basic concepts regarding model predictive control, that are necessary to consider throughout this dissertation. Furthermore, in Appendix A, an overview of convex optimisation is provided, and, in Appendix B, some core concepts regarding the stability of discrete-time systems are introduced.

## 3.1 Model Predictive Control

In order to design a Model Predictive controller for a deterministic dynamical system, it is necessary to establish its state-space model. Assuming that such model description is time-invariant and over continuous time,

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}), \tag{3.1}$$

where $\boldsymbol{f} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$.

As the infinite horizon LQ regulator (LQR) asymptotically stabilises the origin of a closed-loop system, one is suggested to obtain a control function $\boldsymbol{u}(\cdot)$ by minimising an objective function which measures the deviation of $\boldsymbol{x}(\cdot)$ and $\boldsymbol{u}(\cdot)$ from zero subject to the system dynamics (3.1). One selects a positive definite quadratic function $\ell : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}_0^+$ and sets the objective function $V_\infty(\cdot)$ as a functional of $\boldsymbol{u}(\cdot)$,

$$V_\infty\left(\boldsymbol{x}(0), \boldsymbol{u}(\cdot)\right) = \int_0^\infty \ell(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt. \tag{3.2}$$

The proposed optimisation problem to find $\boldsymbol{u}(\cdot)$ poses some challenge as the decision variable is a function of time and the time interval of interest $[0, \infty)$ gives rise to numerical issues. In this manner, the system dynamics is discretised,

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \tag{3.3}$$

the semi-infinite time interval is replaced by a finite time interval, and a terminal cost $V_f : \mathbb{X}_f \to \mathbb{R}_0^+$ is appended to the objective function so that the *cost-to-go* for the semi-infinite interval is approximated once the system enters an established terminal set $\mathbb{X}_f$.

Additionally, set membership constraints may be imposed on the current state, $\boldsymbol{x}(k) \in \mathbb{X} \subseteq \mathbb{R}^n$ and

on the current control input, $\boldsymbol{u}(k) \in \mathbb{U} \subseteq \mathbb{R}^m$. More generally, the two constraints are coupled into one $(\boldsymbol{x}(k), \boldsymbol{u}(k)) \in \mathbb{G}$, in which $\mathbb{G} \subseteq \mathbb{X} \times \mathbb{U}$. This allows for the inclusion of control constraints which are state dependent. The possible addition of these constraints is one of main advantages of Model Predictive Control (MPC).

Given an initial state $\boldsymbol{x}_0$, a control sequence $\boldsymbol{u}^S = \{\boldsymbol{u}(0), \boldsymbol{u}(1), \ldots, \boldsymbol{u}(N-1)\}$ that leads the state to the origin is obtained by solving the optimisation problem that follows. Only the first value of the sequence is applied to the plant and the next state is employed as the initial condition of the system dynamics,

$$\begin{aligned} \text{minimise} \quad & \sum_{k=0}^{N-1} \ell(\boldsymbol{x}(k), \boldsymbol{u}(k)) + V_f(\boldsymbol{x}(N)) \\ \text{s.t.} \quad & \boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)), \text{ for } k = 0, 1, \ldots, N-1, \\ & \boldsymbol{x}(0) = \boldsymbol{x}_0 \in \mathbb{X}, \\ & (\boldsymbol{x}(k), \boldsymbol{u}(k)) \in \mathbb{G}, \text{ for } k = 0, 1, \ldots, N-1, \\ & \boldsymbol{x}(N) \in \mathbb{X}_f \subseteq \mathbb{X}. \end{aligned} \tag{3.4}$$

The optimal cost for this problem is provided by the value function $V_N^0(\cdot)$. Given an initial state $\boldsymbol{x}_0$, $V_N^0(\boldsymbol{x}_0)$ denotes the value of the objective function of the optimal control problem (3.4) at the initial condition $\boldsymbol{x}_0$. In other words, it represents the *cost-to-go* from time 0 to time $N$ under the control inputs $\boldsymbol{u}(k),\ k = 0, 1, \ldots, N-1$, starting from state $\boldsymbol{x}_0$. For convenience, the optimal control problem (3.4) is denoted as $\mathbb{P}_N(\boldsymbol{x})$, where $N$ corresponds to the control horizon.

Dynamic Programming (DP), an optimisation technique developed by Richard Bellman in the 1950s, yields a sequence of optimal control laws $\boldsymbol{\kappa}^S = \{\boldsymbol{\kappa}_N(\cdot), \boldsymbol{\kappa}_{N-1}(\cdot), \ldots, \boldsymbol{\kappa}_1(\cdot)\}$, with $\boldsymbol{\kappa}_j : \mathcal{X}_j \to \mathbb{U}$, from solving the optimisation problem above [44]; the domain, the *feasibility set* $\mathcal{X}_j$ of each control law will be defined soon. Over the fixed time interval considered, $\boldsymbol{\kappa}_j(\cdot)$ is the optimal control law at time $N-j$, for $j = 1, 2, \ldots, N$, being $j$ the *time-to-go*. Furthermore, DP not only solves $\mathbb{P}_N(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}_N$, the domain of $V_N^0(\cdot)$, but also $\mathbb{P}_j(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{X}_j$, the domain of $V_j^0(\cdot)$, for $j = 0, 1, \cdots, N-1$. The symbol $\mathbb{P}_j(\boldsymbol{x})$ corresponds to an identical optimisation problem to (3.4) with control horizon $j$ and $V_j^0(\cdot)$, to the *cost-to-go* from time 0 to time $j$.

The DP equations are, for all $\boldsymbol{x} \in \mathcal{X}_j$,

$$V_j^0(\boldsymbol{x}) = \min_{\boldsymbol{u} \in \mathbb{U}} \left\{ \ell(\boldsymbol{x}, \boldsymbol{u}) + V_{j-1}^0\left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})\right) \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_{j-1} \right\}, \tag{3.5}$$

$$\boldsymbol{\kappa}_j(\boldsymbol{x}) = \arg\min_{\boldsymbol{u} \in \mathbb{U}} \left\{ \ell(\boldsymbol{x}, \boldsymbol{u}) + V_{j-1}^0\left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})\right) \mid \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_{j-1} \right\}, \tag{3.6}$$

with

$$\mathcal{X}_j \coloneqq \left\{ \boldsymbol{x} \in \mathbb{X} \mid \exists \boldsymbol{u} \in \mathbb{U} \text{ such that } \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_{j-1} \right\}, \tag{3.7}$$

for $j = 1, 2, \ldots, N$, with terminal conditions,

$$V_0^0(\boldsymbol{x}) = V_f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathcal{X}_0, \quad \mathcal{X}_0 = \mathbb{X}_f.$$

Starting from the terminal conditions, using (3.5), (3.6), and (3.7) recursively, progressing backwards in time, one obtains the value function $V_N^0(\cdot)$ and the implicit MPC control law $\boldsymbol{\kappa}_N(\cdot)$, however, when the state dimension is reasonably high, state or control constraints are included, or the system is non-linear, using the DP recursion to solve (3.4) is most often an intractable problem.

As stated in [45], MPC is best regarded as a practical means of implementing the DP solution; for a given state $\boldsymbol{x}$, it provides $V_N^0(\boldsymbol{x})$ and $\boldsymbol{\kappa}_N(\boldsymbol{x})$, the value of, respectively, the value function and the control law at *a point* $\boldsymbol{x}$. In addition, MPC can handle non-linear systems and satisfy hard constraints on the state and control inputs by swiftly solving an open-loop optimal problem and providing a finite control sequence, given the current state $\boldsymbol{x}(k)$. In any case, it is still worth examining the DP solution of $\mathbb{P}_N(\boldsymbol{x})$ for the insight it provides.

For instance, one important property, presented in the following proposition, that can be deduced from the DP equations regards the feasibility sets $\mathcal{X}_j$, the set of states in $\mathbb{X}$ which can be steered to $\mathbb{X}_f$ in $j$ steps by an *admissible* control sequence, that is, a control sequence that satisfies the constraints of $\mathbb{P}_j(\boldsymbol{x})$. Before presenting such proposition, let the definition of a control invariant set be introduced.

**Definition 3.1** (Control Invariant Set)
*A set $\mathcal{A}$ is control invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k))$, $\boldsymbol{u} \in \mathbb{U}$, if, for all $\boldsymbol{x}(0) \in \mathcal{A}$, there exists a $\boldsymbol{u} \in \mathbb{U}$ such that $\boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k)) \in \mathcal{A},\ \forall k \in \mathbb{N}_0$.*

**Proposition 3.1** (Nested Feasibility Sets, from [45])
*If $\mathcal{X}_0 \coloneqq \mathbb{X}_f$ is control invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}\left(\boldsymbol{x}(k), \boldsymbol{u}(k)\right)$, $\boldsymbol{u} \in \mathbb{U}$, and $0 \in \mathbb{X}_f$, then, for each $j \in \mathbb{N}_0$, the set $\mathcal{X}_j$ is also control invariant, $\mathcal{X}_j \supseteq \mathcal{X}_{j-1}$, and $0 \in \mathcal{X}_j$. In addition, the sets $\mathcal{X}_j$ and $\mathcal{X}_{j-1}$ are positive invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}\left(\ \boldsymbol{x}(k),\ \boldsymbol{\kappa}_j(\boldsymbol{x}(k))\ \right)$, for $j \in \mathbb{N}$.*

*Proof.* By assumption, $\mathcal{X}_0 \coloneqq \mathbb{X}_f$ is control invariant. According to (3.7),

$$\mathcal{X}_1 = \{\boldsymbol{x} \in \mathbb{X} \mid \exists \boldsymbol{u} \in \mathbb{U} \text{ such that } \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_0\}.$$

Since $\mathcal{X}_0$ is control invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}\left(\boldsymbol{x}(k), \boldsymbol{u}(k)\right)$, $\boldsymbol{u} \in \mathbb{U}$, for every $\boldsymbol{x} \in \mathcal{X}_0$, there exists a $\boldsymbol{u} \in \mathbb{U}$ such that $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \in \mathcal{X}_0$ so that $\boldsymbol{x} \in \mathcal{X}_1$. This implies that $\mathcal{X}_1 \supseteq \mathcal{X}_0$. Since for every $\boldsymbol{x} \in \mathcal{X}_1$, there exists a $\boldsymbol{u} \in \mathbb{U}$ such that $\boldsymbol{f}\left(\boldsymbol{x}(k), \boldsymbol{u}(k)\right) \in \mathcal{X}_0 \subseteq \mathcal{X}_1$, it follows that $\mathcal{X}_1$ is control invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}\left(\boldsymbol{x}(k), \boldsymbol{u}(k)\right)$, $\boldsymbol{u} \in \mathbb{U}$. By induction, if $\mathcal{X}_{j-1}$ is control invariant, $\mathcal{X}_j \supseteq \mathcal{X}_{j-1}$ and $\mathcal{X}_j$ is also control invariant, $\forall j \in \mathbb{N}$. Hence, $0 \in \mathcal{X}_j,\ \forall j \in \mathbb{N}_0$. Furthermore, it follows from (3.6) that $\kappa_j(\cdot)$ steers every $\boldsymbol{x} \in \mathcal{X}_j$ into $\mathcal{X}_{j-1} \subseteq \mathcal{X}_j$ and, thus, $\mathcal{X}_j$ is positive invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}\left(\ \boldsymbol{x}(k),\ \boldsymbol{\kappa}_j(\boldsymbol{x}(k))\ \right)$. Finally, as $\mathcal{X}_{j-1} \subseteq \mathcal{X}_j$, $\boldsymbol{\kappa}_j(\cdot)$ also steers $\boldsymbol{x} \in \mathcal{X}_{j-1}$ into $\mathcal{X}_{j-1}$, then $\mathcal{X}_{j-1}$ is positive invariant under this control law as well. ■

One relevant consequence from this proposition is that the terminal set $\mathbb{X}_f$ being control invariant ensures that the set $\mathcal{X}_N$, the set of initial conditions for which $\mathbb{P}_N(\boldsymbol{x})$ is feasible, is positive invariant. In this manner, if the optimal control problem is feasible for the initial condition, for all the successor states it is as well. This is a property of the uttermost importance and it is usually referred to as *recursive*

*feasibility*. Although this dissertation only focuses on nominal MPC controllers, one should bear in mind that in presence of uncertainty, in the form of additive disturbances, this remark might not hold true.

The set of admissible $(\boldsymbol{x}_0, \boldsymbol{u}^S)$, i.e., the set of $(\boldsymbol{x}_0, \boldsymbol{u}^S)$ for which the constraints of the optimisation problem (3.4) are satisfied, is denoted by

$$\begin{aligned}\mathbb{G}_N \coloneqq \{(\boldsymbol{x}_0, \boldsymbol{u}^S) \in \mathbb{R}^n \times \mathbb{R}^{Nm} \mid \\ (\ \boldsymbol{\beta}(k; \boldsymbol{x}_0, \boldsymbol{u}^S), \boldsymbol{u}(k)\ ) \in \mathbb{G},\ \forall k \in \{0, 1, \cdots, N-1\},\ \boldsymbol{\beta}(N; \boldsymbol{x}_0, \boldsymbol{u}^S) \in \mathbb{X}_f\},\end{aligned} \tag{3.8}$$

where $\boldsymbol{\beta}(k; \boldsymbol{x}_0, \boldsymbol{u}^S)$ denotes the solution of (3.3) at time $k$, if the initial state is $\boldsymbol{x}_0$ and the control sequence $\boldsymbol{u}^S$, and, thus, $\boldsymbol{x}(k) \coloneqq \boldsymbol{\beta}(k; \boldsymbol{x}_0, \boldsymbol{u}^S)$. This definition allows to define the set $\mathcal{X}_N$, the domain of the value function $V_N^0(\cdot)$, the region of attraction of a system in closed-loop with a MPC controller as follows.

**Definition 3.2** (Set of feasible initial states of a system coupled with a MPC controller)
*Let $\mathcal{X}_N$ denote the set of initial states which can be steered to $\mathbb{X}_f$ along the trajectories of $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k))$ in $N$ steps by an admissible control sequence $\boldsymbol{u}^S$, i.e., a control sequence that satisfies the control, state, and terminal constraints of the optimisation control problem (3.4). Moreover, $\mathcal{X}_N$ is the orthogonal projection of $\mathbb{G}_N \subset \mathbb{R}^n \times \mathbb{R}^{Nm}$ (3.8) onto $\mathbb{R}^n$,*

$$\mathcal{X}_N \coloneqq \left\{\boldsymbol{x}_0 \in \mathbb{R}^n \mid \exists \boldsymbol{u}^S \in \mathbb{R}^{Nm} \text{ such that } (\boldsymbol{x}_0, \boldsymbol{u}^S) \in \mathbb{G}_N\right\}.$$

The set $\mathcal{X}_N$ can be obtained analytically using the DP equations and may be computed if the system is linear and the constraints, including the terminal constraint, are polyhedral, and provided that the state dimension and the control horizon $N$ are suitably small. In most cases, the only alternative is to estimate the region of attraction by simulation.

### 3.1.1 Stability

In order to establish the asymptotic stability of the origin of the closed-loop system

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\ \boldsymbol{x}(k),\ \boldsymbol{\kappa}_N(\boldsymbol{x}(k))\ ),$$

where $\boldsymbol{\kappa}_N(\cdot)$ denotes the implicit MPC control law, the value function $V_N^0(\cdot)$ is selected to be a valid Lyapunov function for this system (*vide* Theorem B.1). In order to determine whether $V_N^0(\cdot)$ is a Lyapunov function, some assumptions must be satisfied.

**Assumption 3.1** (Properties of the system and cost functions)
*The vector field $\boldsymbol{f}(\cdot,\cdot)$, the stage cost $\ell(\cdot,\cdot)$, and the terminal cost $V_f(\cdot)$ are continuous functions. In addition, $\boldsymbol{f}(\boldsymbol{0},\boldsymbol{0}) = \boldsymbol{0}$, $\ell(\boldsymbol{0},\boldsymbol{0}) = 0$, $V_f(\boldsymbol{0}) = 0$.*

**Assumption 3.2** (Properties of constraint sets)
*The set $\mathbb{G} \subseteq \mathbb{X} \times \mathbb{U}$ is closed and $\mathbb{X}_f$ is compact. Should $\mathbb{U}$ be bounded, it is compact; else, the objective function of (3.4) is coercive. All these sets contain the origin within.*

**Assumption 3.3** (Stage and terminal costs)
*There are class $\mathcal{K}$ functions $\alpha_1(\cdot)$ and $\alpha_f(\cdot)$ which satisfy $\ell(\boldsymbol{x},\boldsymbol{u}) \geq \alpha_1(\|\boldsymbol{x}\|)$, $\forall \boldsymbol{x} \in \mathcal{X}_N$, $\forall \boldsymbol{u}$ such that $(\boldsymbol{x},\boldsymbol{u}) \in \mathbb{G}$, and $V_f(\boldsymbol{x}) \leq \alpha_f(\|\boldsymbol{x}\|)$, $\forall \boldsymbol{x} \in \mathbb{X}_f$.*

**Assumption 3.4** (Basic Stability Assumption)
*The terminal set $\mathbb{X}_f$ is control invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k),\boldsymbol{u}(k))$. For every $\boldsymbol{x} \in \mathbb{X}_f$, there is an $\boldsymbol{u} \in \mathbb{U}$ such that $\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \in \mathbb{X}_f$. Furthermore, there exists an $\boldsymbol{u}$, such that $(\boldsymbol{x},\boldsymbol{u}) \in \mathbb{G}$, which satisfies the following inequality, $V_f\left(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})\right) - V_f(\boldsymbol{x}) \leq -\ell(\boldsymbol{x},\boldsymbol{u})$, $\forall \boldsymbol{x} \in \mathbb{X}_f$.*

These assumptions ensure there exist suitable functions, defined over $\mathcal{X}_N$, which impose upper and lower bounds on the value function $V_N^0(\cdot)$ and the decrease of the value of this function along the trajectories of the closed-loop system. The lower bound on the value function follows from Assumption 3.3, where in which it is assumed there exists a $\mathcal{K}$ function $\alpha_1(\cdot)$ such that $\ell(\boldsymbol{x},\boldsymbol{u}) \geq \alpha_1(\|\boldsymbol{x}\|)$, $\forall \boldsymbol{x} \in \mathcal{X}_N$. The upper bound and the descent property of the value function is demonstrated using the following proposition.

**Proposition 3.2** (Monotonicity of the value function, from [45])
*Suppose Assumptions 3.1, 3.2, 3.3, and 3.4 are satisfied. Then,*

$$\begin{aligned} V_{j+1}^0(\boldsymbol{x}) &\leq V_j^0(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathcal{X}_j, \\ V_j^0(\boldsymbol{x}) &\leq V_f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{X}_f,\ \forall j \in \mathbb{N}_0. \end{aligned}$$

*Proof.* According to (3.5),

$$V_1^0(\boldsymbol{x}) = \min_{\boldsymbol{u} \in \mathbb{U}} \left\{ \ell(\boldsymbol{x},\boldsymbol{u}) + V_0^0(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})) \mid \boldsymbol{f}(\boldsymbol{x},\boldsymbol{u}) \in \mathcal{X}_0 \right\}.$$

Since Assumption 3.4 states that $\ell(\boldsymbol{x},\boldsymbol{u}) + V_f(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{u})) \leq V_f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{X}_f$, it follows that

$$V_1^0(\boldsymbol{x}) \leq V_f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{X}_f,$$

as, by definition, $V_0^0(\boldsymbol{x}) \coloneqq V_f(\boldsymbol{x})$ and $\mathcal{X}_0 \coloneqq \mathbb{X}_f$.

Assume that $j \in \mathbb{N}$, then, using the DP equations (3.5) and (3.6),

$$\begin{aligned} V_{j+1}^0(\boldsymbol{x}) - V_j^0(\boldsymbol{x}) = \ell(\boldsymbol{x},\boldsymbol{\kappa}_{j+1}(\boldsymbol{x})) + V_j^0(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\kappa}_{j+1}(\boldsymbol{x}))) + \\ - \ell(\boldsymbol{x},\boldsymbol{\kappa}_j(\boldsymbol{x})) - V_{j-1}^0(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\kappa}_j(\boldsymbol{x}))), \quad \forall \boldsymbol{x} \in \mathcal{X}_j \subseteq \mathcal{X}_{j+1}. \end{aligned}$$

Since $\boldsymbol{\kappa}_j(\cdot)$ may not be optimal for $\mathbb{P}_{j+1}(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathcal{X}_j \subseteq \mathcal{X}_{j+1}$,

$$\begin{aligned} V_{j+1}^0(\boldsymbol{x}) - V_j^0(\boldsymbol{x}) \leq \ell(\boldsymbol{x},\boldsymbol{\kappa}_j(\boldsymbol{x})) + V_j^0(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\kappa}_j(\boldsymbol{x}))) + \\ - \ell(\boldsymbol{x},\boldsymbol{\kappa}_j(\boldsymbol{x})) - V_{j-1}^0(\boldsymbol{f}(\boldsymbol{x},\boldsymbol{\kappa}_j(\boldsymbol{x}))), \quad \forall \boldsymbol{x} \in \mathcal{X}_j. \end{aligned}$$

The definition of $\mathcal{X}_j$ in (3.7) implies $\boldsymbol{x} \in \mathcal{X}_j \Rightarrow \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_j(\boldsymbol{x})) \in \mathcal{X}_{j-1}$, and, therefore, by assumption,

$$V_j^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_j(\boldsymbol{x}))) \leq V_{j-1}^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_j(\boldsymbol{x}))),\ \forall \boldsymbol{x} \in \mathcal{X}_j.$$

This allows to conclude that

$$V_{j+1}^0(\boldsymbol{x}) \leq V_j^0(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathcal{X}_j.$$

By induction, this inequality holds true for $j \in \mathbb{N}_0$.

As $\mathbb{X}_f$ is control invariant, the feasibility sets are nested, $\mathcal{X}_j \subseteq \mathcal{X}_{j+1},\ \forall j \in \mathbb{N}_0$, it follows that

$$V_j^0(\boldsymbol{x}) \leq V_f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{X}_f,\ \forall j \in \mathbb{N}_0.$$

■

This proposition reveals that the monotonicity of the value function establishes the descent property of $V_N^0(\cdot)$,

$$\begin{aligned} V_N^0(\boldsymbol{x}) &= \ell(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x})) + V_{N-1}^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x}))) \\ &= \ell(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x})) + V_N^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x}))) + \left[V_{N-1}^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x}))) - V_N^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x})))\right], \end{aligned}$$

rearranging the terms in the equation and using Proposition 3.2,

$$\begin{aligned} V_N^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x}))) &= V_N^0(\boldsymbol{x}) - \ell(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x})) + \left[V_N^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x}))) - V_{N-1}^0(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x})))\right] \\ &\leq V_N^0(\boldsymbol{x}) - \ell(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x})), \quad \forall \boldsymbol{x} \in \mathcal{X}_N. \end{aligned}$$

Proposition 3.2 also states that $V_N^0(\boldsymbol{x}) \leq V_f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{X}_f$. It follows from Assumption 3.3 that $V_N^0(\boldsymbol{x}) \leq \alpha_f(\|\boldsymbol{x}\|),\ \forall \boldsymbol{x} \in \mathbb{X}_f$, where $\alpha_f(\cdot)$ is a class $\mathcal{K}$ function. Through simple reasoning [45, Proposition 2.15], it can be demonstrated that Assumption 3.1 and 3.2 ensure that $V_N^0(\cdot)$ is locally bounded on $\mathcal{X}_N$ (i.e., locally bounded on every compact subset of $\mathcal{X}_N$). These same assumptions together with $\mathbb{X}_f$ being control invariant guarantee that the upper bound on $V_N^0(\cdot)$ in $\mathbb{X}_f$ can be extended to a similar bound on $V_N^0(\cdot)$ in $\mathcal{X}_N$. Hence, $V_f(\boldsymbol{x}) \leq \alpha_f(\|\boldsymbol{x}\|),\ \forall \boldsymbol{x} \in \mathbb{X}_f$, implies there exists a $\mathcal{K}$ function $\alpha(\cdot)$ such that $V_N^0(\boldsymbol{x}) \leq \alpha(\|\boldsymbol{x}\|),\ \forall \boldsymbol{x} \in \mathcal{X}_N$. This is stated in the next proposition.

**Proposition 3.3** (Extension for the upper bound to $\mathcal{X}_N$, from [45])
*Suppose Assumptions 3.1 and 3.2 are satisfied and that $\mathbb{X}_f$ is control invariant and contains the origin in its interior. Suppose, in addition, there exists a $\mathcal{K}$ function $\alpha_f(\cdot)$ such that $V_f(\boldsymbol{x}) \leq \alpha_f(\boldsymbol{x}),\ \forall \boldsymbol{x} \in \mathbb{X}_f$. Then, there exists a $\mathcal{K}$ function $\alpha(\cdot)$ such that*

$$V_N^0(\boldsymbol{x}) \leq \alpha(\|\boldsymbol{x}\|),\ \forall \boldsymbol{x} \in \mathcal{X}_N.$$

Although a proof of this proposition is not provided, the aforementioned assumptions ensure $V_N^0(\cdot)$ is continuous at zero and that $\mathcal{X}_N$ is closed – provided that $\mathbb{X}_f$ is closed, the DP equations ensure every

*feasibility set* $\mathcal{X}_j$ is also closed [45, Proposition 2.10(c)]; the imposition of $\mathbb{X}_f$ being control invariant guarantees that $\mathbb{X}_f \subseteq \mathcal{X}_N$, as a result of Proposition 3.1. Since $V_N^0(\cdot)$ is locally bounded on $\mathcal{X}_N$, Proposition 14 of [46] applies and, thus, Proposition 3.3 holds.

Finally, the asymptotic stability of the origin of the closed-loop system can be established by verifying Assumption 3.1 to 3.4, as stated in following theorem.

**Theorem 3.1** (Asymptotic Stability of the Origin, from [45])
*Suppose Assumptions 3.1, 3.2, 3.3, and 3.4 are satisfied. Then,*

- *There are class* $\mathcal{K}$ *functions* $\alpha_1(\cdot)$ *and* $\alpha_2(\cdot)$ *such that for all* $\boldsymbol{x} \in \mathcal{X}_N$ *impose, respectively, a lower and upper bounds on* $V_N^0(\cdot)$ *and ensure the decrease of the value of this function along trajectories of the closed-loop system,*

$$\alpha_1(\|\boldsymbol{x}\|) \leq V_N^0(\boldsymbol{x}) \leq \alpha_2(\|\boldsymbol{x}\|)$$
$$V_N^0\left(f(\boldsymbol{x}, \kappa_N(\boldsymbol{x}))\right) - V_N^0(\boldsymbol{x}) \leq -\alpha_1(\|\boldsymbol{x}\|);$$

- *The origin is asymptotically stable in* $\mathcal{X}_N$ *for* $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x}))$*;*

- *Exponential stability is established if there is a Lyapunov function in* $\mathcal{X}_N$ *for the closed-loop system with* $\alpha_i(\cdot) = c_i \|\cdot\|^a$*, where* $a, c_i \in \mathbb{R}_0^+,\ i = 1, 2$*.*

In Appendix C, this theorem is applied to verify if the origin of a deterministic time-invariant system in closed-loop with a NMPC controller is stable.

# Chapter 4

# Planning for Aggressive Manoeuvres

In this chapter, a strategy for planning reference trajectories for relay manoeuvres is presented. Based on the seminal works [21, 22], polynomial trajectory generation algorithms for one and two quadcopters are introduced. Inter-vehicle collision avoidance constraints are included in the trajectory generation problem as binary constraints in the same fashion as in [24]. The addition of these constraints enforces that reference trajectories are found by formulating mixed-integer quadratic programs (MIQP). A constraint on the linear acceleration of the vehicles is formulated to impose a desired direction of thrust along any point of the manoeuvre.

The work presented in this chapter led to a publication in the proceedings of the 2021 International Conference on Unmanned Aircraft Systems [19].

## 4.1 Single Quadrotor

As demonstrated in [21], the quadrotor dynamics are *differentially flat*. The choice of the position $\boldsymbol{p}$ and yaw angle $\psi$ of the vehicle as *flat outputs* allows the states and inputs of the system to be expressed algebraically in terms of these outputs and their derivatives. A dynamical model of a quadrotor is introduced in the next chapter, Chapter 5.

In this manner, piecewise polynomials of order $N$ over $M$ time intervals are selected to serve as basis functions for generating smooth trajectories $\boldsymbol{\sigma}(t)$ in the space of flat outputs,

$$\boldsymbol{\sigma}(t) = \begin{cases} \sum_{k=0}^{N} \boldsymbol{\sigma}_{k1} t^k & t_0 \leq t < t_1, \\ \sum_{k=0}^{N} \boldsymbol{\sigma}_{k2} t^k & t_1 \leq t < t_2, \\ \quad \vdots & \\ \sum_{k=0}^{N} \boldsymbol{\sigma}_{kM} t^k & t_{M-1} \leq t \leq t_M, \end{cases} \tag{4.1}$$

where the vector $\boldsymbol{\sigma}_{kn} = \begin{bmatrix} x_{kn} & y_{kn} & z_{kn} & \psi_{kn} \end{bmatrix}^T$ represents the set of constants associated with the polynomial of degree $k$ of the trajectory segment $\boldsymbol{\sigma}_n(t),\ n = 1, \ldots, M$. In a more compact form, (4.1)

can be written as

$$\boldsymbol{\sigma}(t) = \boldsymbol{\sigma}_n(t) = \boldsymbol{\Sigma}_n \boldsymbol{\eta}(t),\ t_{n-1} \leq t \leq t_n, \tag{4.2}$$

with $\boldsymbol{\Sigma}_n = \begin{bmatrix} \boldsymbol{\sigma}_{0n} & \boldsymbol{\sigma}_{1n} & \cdots & \boldsymbol{\sigma}_{Nn} \end{bmatrix}$ and $\boldsymbol{\eta}(t) = \begin{bmatrix} 1 & t & \cdots & t^N \end{bmatrix}^T,\ n = 0, \ldots, M.$

An optimisation problem is formulated in order to determine the polynomial coefficients $\sigma_{kn}$ which define the trajectory that passes through a set of waypoints $\boldsymbol{w}_n$ at times $t_n,\ n = 0, \ldots, M$. Since restricting the attitude of the vehicle is required during the parcel exchange manoeuvre, the desired orientation $\boldsymbol{R}_{des}$ is associated with each of the waypoints. As $\boldsymbol{z}_B$ points in the direction of thrust, the attitude of the vehicle, apart from the yaw angle, is determined by the direction of this axis. In this manner, from $\boldsymbol{R}_{des}$, the desired $\boldsymbol{z}_B$ is obtained,

$$\boldsymbol{z}_B(t_n) = \boldsymbol{R}_{des}(t_n)\boldsymbol{z}_W, \tag{4.3}$$

and the linear velocity dynamics equation presented in (5.3) can be used for formulating the following linear constraint on the linear acceleration of the vehicle which imposes a desired direction of thrust,

$$\left(\boldsymbol{I}_3 - \boldsymbol{z}_B(t_n)\boldsymbol{z}_B^T(t_n)\right)\left(\dot{\boldsymbol{v}}(t_n) + g\boldsymbol{z}_W\right) = \boldsymbol{0}, \tag{4.4}$$

where $\boldsymbol{I}_3$ corresponds to the three-by-three identity matrix. The cost function of the proposed optimisation problem is such that it minimises the integral of the squared norm of the snap (the fourth derivative of position) of the trajectory and the second derivative of the yaw angle,

$$\begin{aligned}
\text{minimise} \quad & \int_{t_0}^{t_M} \mu_p \left\| \frac{d^4\boldsymbol{p}}{dt^4} \right\|^2 + \mu_\psi \frac{d^2\psi}{dt^2}^2 \, dt \\
\text{s.t.} \quad & \boldsymbol{\sigma}(t_n) = \boldsymbol{w}_n,\ \text{for } n = 0, \ldots, M, \\
& \dot{\boldsymbol{\sigma}}(t_n) = \boldsymbol{v}_n,\ \text{for } n = 0, \ldots, M, \\
& \ddot{\boldsymbol{\sigma}}(t_M) = \boldsymbol{0} \text{ or free}, \\
& \text{Equation } (4.4),\ \text{for } n = 0, \ldots, M,
\end{aligned} \tag{4.5}$$

where $\mu_p,\ \mu_\psi \in \mathbb{R}$, $\boldsymbol{w}_n \in \mathbb{R}^3 \times (-\pi, \pi]$ and $\boldsymbol{v}_n \in \mathbb{R}^4$ correspond, respectively, to the position and yaw waypoint and to the desired linear velocity and yaw angle time derivative $\dot{\psi}$ at time $t_n$. The meaningfulness of this cost function lies in the fact that the input body torques $\boldsymbol{\tau}$ are algebraically related to the snap of the vehicle and the second derivative of the yaw angle [21].

So as to guarantee the smoothness of the generated trajectory, constraints are introduced to ensure the continuity of the first four derivatives of $\boldsymbol{p}$ and the first two derivatives of $\psi$. In particular, to achieve the desired orientation at each waypoint $\boldsymbol{w}_n$, it is necessary to impose the continuity of acceleration and jerk (the third derivative of position) to ensure the continuity of orientation and roll and pitch rate trajectories. However, the continuity of these position derivatives alone is not sufficient to make sure a quadrotor is able to follow accurately every single trajectory. Saturation and rate limit constraints may impede the execution of some manoeuvres.

Even though the objective function implicitly minimises the necessary input body torques $\boldsymbol{\tau}$ to track

the generated trajectory, one may also enforce bound constraints on the aforementioned derivatives of position to incorporate saturation and rate limit constraints into (4.5) with the objective of further ensuring the feasibility of trajectories. In the context of this work, it was opted not to include these constraints in (4.5) for the reasons discussed in Section 4.2.1.

In what concerns the determination of the coefficients $\sigma_{kn}$, the optimisation problem is readily solved by reformulating (4.5) as a quadratic program (QP),

$$\begin{aligned} \underset{\boldsymbol{c}}{\text{minimise}} \quad & \boldsymbol{c}^T \mathcal{H} \boldsymbol{c} + \boldsymbol{f}^T \boldsymbol{c} \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{c} = \boldsymbol{b}. \end{aligned} \tag{4.6}$$

The vector $\boldsymbol{c}$ contains $F = 4(N+1)M$ optimisation variables which correspond to the polynomial coefficients that characterise the whole trajectory. In Appendix E, one may find the algebraic manipulation in order to structure (4.5) as a QP and, thus, obtain the constants $\mathcal{H}$, $\boldsymbol{f}$, $\boldsymbol{A}$, and $\boldsymbol{b}$.

## 4.2 Planning for Two Quadrotors

In this section, the objective is to plan trajectories in a centralised fashion for two quadrotors while avoiding inter-vehicle collisions. The authors of [22, 24] have successfully formulated mixed-integer convex optimisation problems where the integer variables are employed in the collision and obstacle avoidance constraints.

Consider two vehicles each with position $\boldsymbol{p}_i(t_k) = \begin{bmatrix} x_i(t_k) & y_i(t_k) & z_i(t_k) \end{bmatrix}^T$, $i \in \{1, 2\}$. The inter-collision avoidance constraints are as follows,

$$\begin{aligned} x_1(t_k) - x_2(t_k) &\geq d_{x12} - b_{1k}V, \\ x_2(t_k) - x_1(t_k) &\geq d_{x21} - b_{2k}V, \\ y_1(t_k) - y_2(t_k) &\geq d_{y12} - b_{3k}V, \\ y_2(t_k) - y_1(t_k) &\geq d_{y21} - b_{4k}V, \\ z_1(t_k) - z_2(t_k) &\geq d_{z12} - b_{5k}V, \\ z_2(t_k) - z_1(t_k) &\geq d_{z21} - b_{6k}V, \\ \sum_{j=1}^{6} b_{jk} &\leq 5. \end{aligned} \tag{4.7}$$

The first six inequalities define two exclusion cuboid-shaped regions oriented with the inertial frame around each vehicle. In other words, around quadrotor $i$ a rectangular prism is considered within which the quadrotor $j$, $j \neq i$, cannot enter. The dimensions of these prisms are provided by the values specified by the $d_{xij}$, $d_{yij}$ and $d_{zij}$ constants. These constants should be large enough for the vehicle to be at any orientation and to remain inside the box.

As for the binary variables, these are denoted by $b_{jk}$, $1 \leq j \leq 6$ and are introduced as slack variables. The constant $V$ is a large positive number. In this manner, when $b_{jk} = 1$, the $j^{\text{th}}$ constraint is relaxed;

should be it zero, this constraint is enforced.

The final inequality expressed in (4.7) imposes that at least one of the constraints above is not relaxed. This last constraint is fundamental since if the binary variables were all equal to zero, it would be only necessary for one of the six constraints to be verified in order to ensure the vehicles were not in collision.

For the generation of trajectories for two quadrotors including the collision avoidance constraints, the vector $\boldsymbol{c}$ is augmented so as to include the polynomial coefficients of the two vehicles and the binary variables. The collision avoidance constraints are enforced at $K$ intermediate time steps, that is, at every time step except for those which are associated with a waypoint. It may be required the vehicles to be closer than permitted by the binary collision avoidance constraints to effectively exchange the parcel at a concrete waypoint.

In this manner, the cost function and constraints in the optimisation problem (4.6) are modified so as to account for the two vehicles. The new quadratic cost matrix $\widetilde{\mathcal{H}}$ is

$$\widetilde{\mathcal{H}} = \begin{bmatrix} \beta_1 \mathcal{H}_1 & \mathbf{0}_{F\times F} & \mathbf{0}_{F\times 6K} \\ \mathbf{0}_{F\times F} & \beta_2 \mathcal{H}_2 & \mathbf{0}_{F\times 6K} \\ \mathbf{0}_{6K\times F} & \mathbf{0}_{6K\times F} & \mathbf{0}_{6K\times 6K} \end{bmatrix}.$$

The constants $\beta_1$ and $\beta_2$ are weighting factors which allow to prioritise the trajectory generation of one vehicle over another [22]. If quadrotor $i$ has $\beta_i > \beta_j,\ j \neq i$, its trajectory will take a more direct path between the start and the end than the trajectory of quadrotor $j$. The dimensions of $\widetilde{\mathcal{H}}$ are $(2F + 6K) \times (2F + 6K)$. The linear terms vector $\widetilde{\boldsymbol{f}}$ has $2F + 6K$ entries equal to zero. The equality constraints correspond to the waypoint and smoothness constraints which account for $8(M + 1)$ and $36(M - 1)$ equations, respectively. Moreover, sixteen constraints may be imposed on the initial and final velocity of the vehicles, and another eight on the their final acceleration. At last, the collision avoidance constrains are expressed as $7K$ linear inequalities.

### 4.2.1 Computational Complexity and Numerical Algorithm

The mixed-integer quadratic program (MIQP) formulated for the generation of reference trajectories for two quadrotors with inter-vehicle collision avoidance constraints involves $2F+6K$ decision variables. Out of those $2F$ are continuous, and correspond to the polynomial coefficients of the trajectories, whereas $6K$ are binary, and are associated with the collision avoidance constraints.

The number of binary variables involved strongly determines the computational time. In the worst-case scenario, a branch-and-bound solver requires $2^{6K}$ quadratic programs to be solved [47]. The authors in [22] highlight the short amount of time required to obtain sub-optimal integer solutions. These are nevertheless, not at all times, adequate trajectories, as for a quadcopter to track them high thrust and body moments may be demanded from its rotors. To overcome this problem, one may consider imposing bounds on the derivatives of position for all time steps, however, the sheer amount of inequality constraints introduced would cause numerical issues to be more prone to arise.

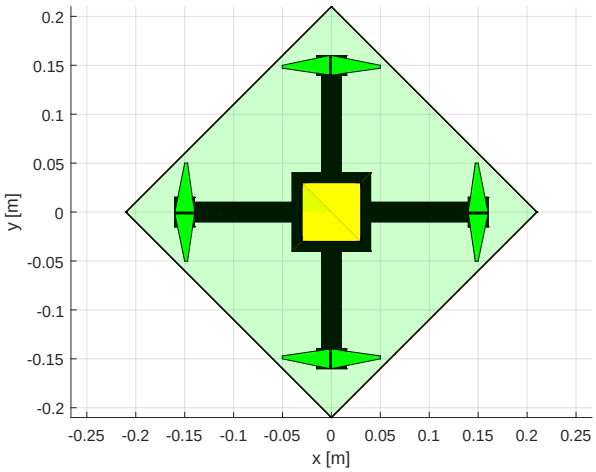



Figure 4.1: Example of a polyhedron around a quadcopter.

The time complexity of this method for generating collision-free trajectories may render it unsuitable for planning long missions. As a remedy, one may weaken the optimality criteria of the mixed-integer program (MIP) solver [48] with the objective of reducing the time spent on solving the optimisation problem.

Additionally, as it is emphasised by the authors of [22], the choice of the spanning polynomial basis functions is of paramount importance. The matrix $\widetilde{\mathcal{H}}$ may become ill-conditioned if inadequate basis functions are selected. For the polynomial basis chosen, it was observed that numerical issues arise when attempting to generate long trajectories with high degree polynomials.

### 4.2.2 Distance Measurement between Quadcopters

Using the collision avoidance constraints described previously, the orientation of both vehicles is not considered. In this section, a method for computing the Euclidean distance between two polyhedrons with a given position and orientation relative to an inertial coordinate frame is introduced.

One starts by defining a polyhedron around each of the quadrotors, the coordinate frame attached to each of the vehicles is the respective body frame (*vide* Section 5.1); this polyhedron is a collection of planar constraints expressed in the body frame.

To this end, a coordinate frame is attached to each of the polyhedrons between which the distance is to be computed. The position and orientation of such coordinate frame relative to an inertial frame is specified by a position vector $\boldsymbol{p}$ and a rotation matrix $\boldsymbol{R}$, respectively.

The boundary of a polyhedron is obtained by the intersection of various planes. A plane is defined by $\hat{\boldsymbol{n}} \cdot z = d$, where $\hat{\boldsymbol{n}} \in \mathbb{R}^n$ is a unit vector perpendicular to the plane and $d \in \mathbb{R}$ represents the signed distance from the origin to the plane in the direction of vector $\hat{\boldsymbol{n}}$.

Let these constraints be defined as $\hat{\boldsymbol{n}}_B \cdot z = d_B$. The subscript $B$ indicates that the variables are expressed in the body frame. Each constraint is expressed in the inertial frame as $\hat{\boldsymbol{n}} \cdot z = d$ using the

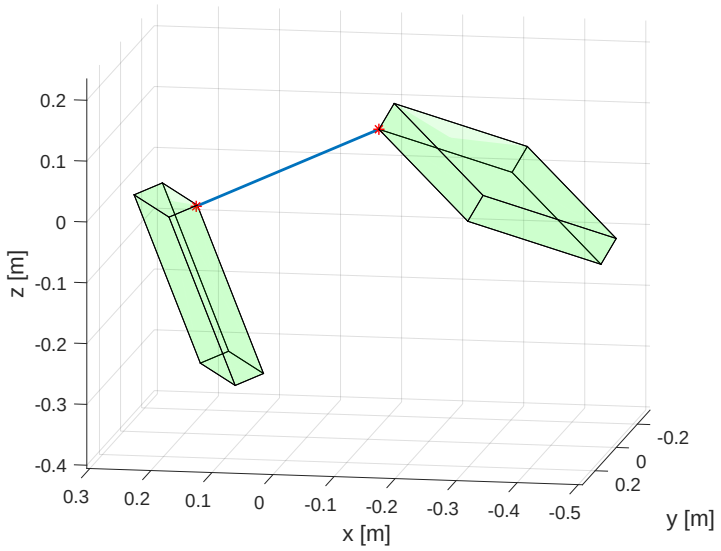



Figure 4.2: Illustration of the solution of the optimisation problem that allows to compute the distance between the two polyhedrons.

following equations,

$$\hat{\boldsymbol{n}} = \boldsymbol{R}\hat{\boldsymbol{n}}_B,$$
$$d = \hat{\boldsymbol{n}} \cdot (\boldsymbol{p} + \hat{\boldsymbol{n}} d_B)\,.$$

Being the polyhedrons expressed in the inertial frame, an optimisation problem is formulated so as to determine two points, one in each of the polyhedrons, which minimise the square of the $\ell_2$ norm between those points,

$$\begin{aligned} \underset{\boldsymbol{x}_1, \boldsymbol{x}_2}{\text{minimise}} \quad & \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 \\ \text{s.t.} \quad & \boldsymbol{x}_1 \in \mathcal{P}_1(\boldsymbol{p}_1, \boldsymbol{R}_1), \\ & \boldsymbol{x}_2 \in \mathcal{P}_2(\boldsymbol{p}_2, \boldsymbol{R}_2), \end{aligned} \tag{4.8}$$

where $\mathcal{P}_j(\boldsymbol{p}_j, \boldsymbol{R}_j)$ denotes the polyhedron wrapped around quadrotor $j$, located at position $\boldsymbol{p}_j$ with orientation $\boldsymbol{R}_j$. This problem is convex as the cost function is a semi-definite positive quadratic function and the constraints are linear inequalities. The optimal value of (4.8) establishes the squared distance function between two polyhedrons,

$$D(\mathcal{P}_1, \mathcal{P}_2) = \inf \left\{ \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2 \mid \boldsymbol{x}_1 \in \mathcal{P}_1, \boldsymbol{x}_2 \in \mathcal{P}_2 \right\}. \tag{4.9}$$

Despite not being possible to express (4.9) in closed-form, it can be deduced that $D(\cdot, \cdot)$ is non-convex both in position $\boldsymbol{p}_1$, $\boldsymbol{p}_2$ and orientation $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ of the polyhedrons, since polyhedra are shapes with flat faces and sharp corners. Be as it may, the squared distance function (4.9) is a handy analysis tool as it allows to compute with relative ease the Euclidean distance between any two polytopes living in the same dimension.

In Figure 4.1, an example of a polyhedron is depicted. It is a box which top and bottom faces are defined by the $\ell_1$ norm and its height is such that the quadcopter is within it. Note it is possible to define

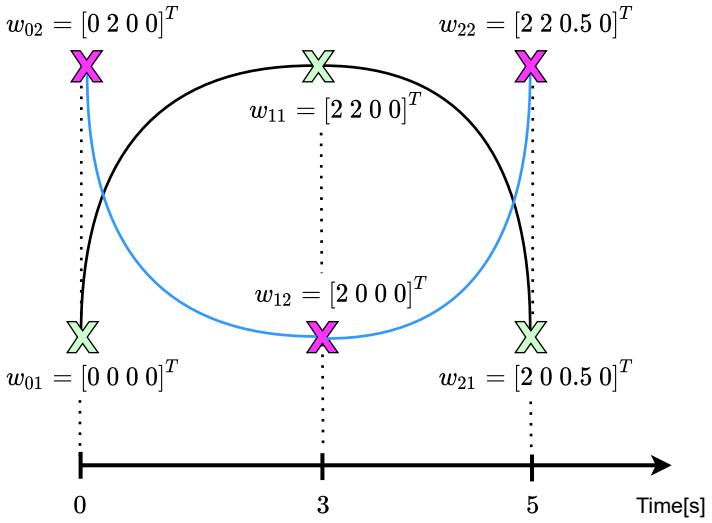



Figure 4.3: Schematic representation of the established waypoints.

(or to approximate) the quadrotor itself; its boundary can be expressed by a set of intersecting planes.

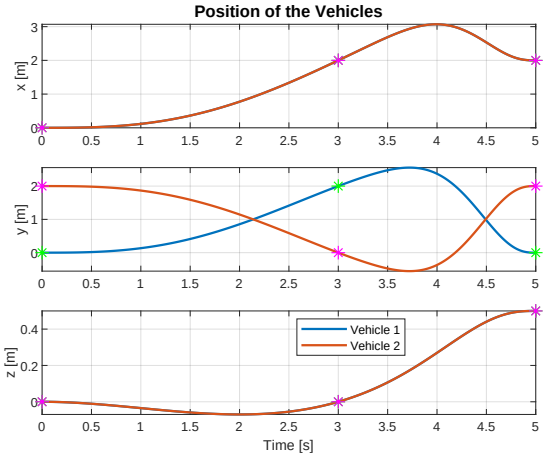
Finally, Figure 4.2 is an illustration of the solution of the optimisation problem (4.8). The red points correspond to the optimal solution $\{\boldsymbol{x}_1^*, \boldsymbol{x}_2^*\}$ and the length of the blue line linking those points is the square root of $D(\mathcal{P}_1, \mathcal{P}_2)$ evaluated at the optimum.

### 4.2.3 Trajectory Generation

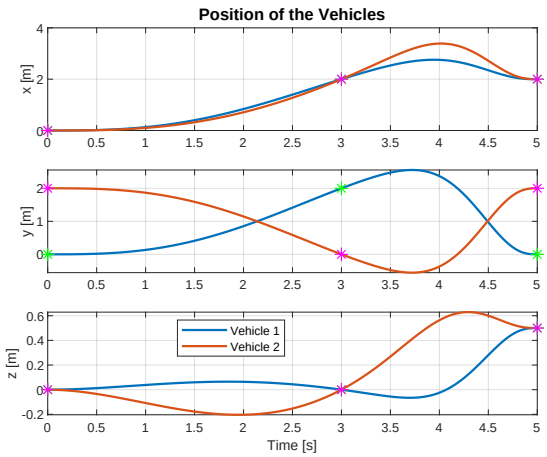
In this section, the proposed method for planning trajectories for two vehicles considering the binary collision avoidance constraints is evaluated. The presented trajectories were obtained from a computer with the operating system Windows 10 installed running MATLAB, equipped with an Intel Core i7-6700HQ CPU @2.60GHz, and 16,00GB of RAM, where the MIQPs were solved using the branch-and-bound solver provided by *CPLEX* [49].

For these experiment, the established waypoints $\boldsymbol{w}_{ij} = \begin{bmatrix} x_{ij} & y_{ij} & z_{ij} & \psi_{ij} \end{bmatrix}^T$ are schematically represented in Figure 4.3. The green crosses correspond to waypoints of the first vehicle, the magenta ones, to the second. The vector $\boldsymbol{w}_{ij}$ denotes the $i^{\text{th}}$ waypoint of vehicle $j$. In this diagram, a time scale is included so as to associate each waypoint with a time instant. The generated trajectories consider that both vehicles start tracking the reference trajectory with zero linear velocity and finish at hover.

The generated position reference trajectories excluding and including collision avoidance binary constraints are plotted in Figure 4.4(a) and 4.4(b), respectively. It is observed that the binary constraints enforce the vehicles to be further apart by performing a comparison between these figures. The trajectory in the $x$ and $z$ axes are modified in order to ensure the two vehicles are never in the same point throughout the trajectory. This result is further confirmed by computing the distance between the reference trajectories, employing (4.9). Note that the orientation reference can be obtained from the linear acceleration and yaw angle trajectories. Figure 4.5 reveals that, in fact, the binary constraints effectively enforces position references to be sufficiently distanced, so that potential collisions between quadrotors are averted.




(a) No collision avoidance constraints.




(b) Including collision avoidance constraints.

Figure 4.4: Position reference trajectories generation excluding and including binary constraints.

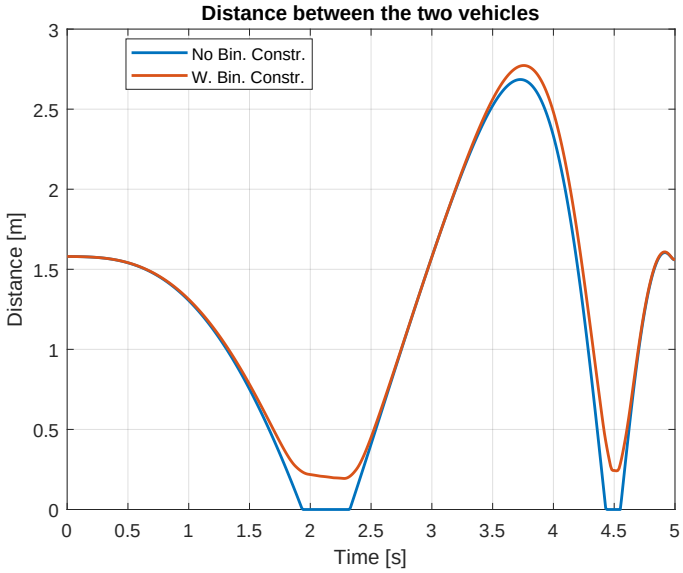



Figure 4.5: Distance between reference trajectories, computed using the square root of (4.9), excluding and including binary constraints.

Table 4.1: Trajectory generation performance results excluding and including binary constraints.

| Figure | $n_c$ | $n_b$ | Cost | Time [s] |
|---|---|---|---|---|
| 4.4(a) | 208 | 0 | 7.3334 | 0.1090 |
| 4.4(b) | 208 | 2988 | 7.8698 | 43.1090 |

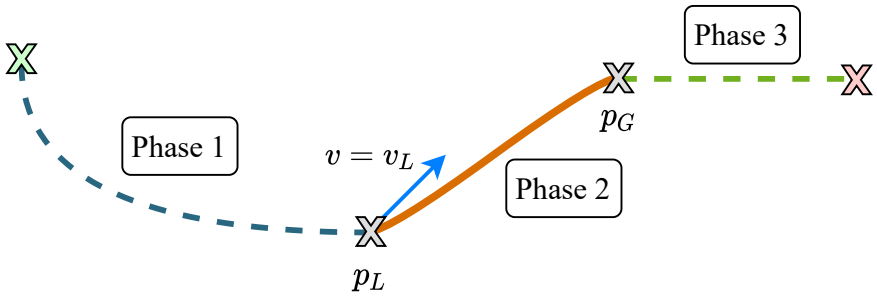



Figure 4.6: Diagram illustrating the segments of the trajectory in order to reach the goal state $G$.

Table 4.1 details the performance results regarding the generation of collision-free trajectories. Let $n_c = 2F$ and $n_b = 6K$ denote the number of continuous and binary variables involved in an optimisation problem, respectively. Notwithstanding the fact that, nowadays, there is a plethora of solvers which are able to achieve an optimal solution of a large convex QP in a fraction of a section, one can verify that the inclusion of the binary variables may increase the computational time significantly.

## 4.3 Relay Manoeuvres Planning

In spite of the addition of orientation constraints to the optimisation problem (4.5) ensuring the generated reference trajectories in the *flat outputs* space are such that the attitude references provided by the controller satisfy the constraints imposed, it does not necessarily imply the vehicles are capable to follow those accurately.

Since the quadcopters performing the manoeuvre must achieve some concrete configuration in state space, a strategy, based on the work published in [20], is devised so that the vehicles are driven to a goal state $G$. In the original article, it is proposed defining trajectories for a single quadrotor as a sequence of segments, each with an associated controller with the objective of reaching $G$.

It is established that the trajectory for each vehicle is composed of three segments, as illustrated in Figure 4.6. In order to generate these trajectory segments, $G$ is provided by specifying the desired position $\boldsymbol{p}_G$, velocity $\boldsymbol{v}_G$, attitude $\boldsymbol{\lambda}_G = \begin{bmatrix}\phi_G & \theta_G & \psi_G\end{bmatrix}^T$ parametrised by $Z - X - Y$ Euler angles, and an angular velocity $\boldsymbol{\omega}_G$ of zero. The reason for expressing the desired attitude using Euler angles rather than rotation matrices is explained soon.

- The segment *Phase 1*, represented as a blue dashed line, corresponds to the trajectory from an initial position (denoted by a green cross in Figure 4.6) to a launching position $\boldsymbol{p}_L$, from where the vehicle departs for the goal position $\boldsymbol{p}_G$ with a launch velocity $\boldsymbol{v}_L$, and with adequate roll, $\phi$, and pitch, $\theta$, angles regarding the desired trajectory;

- For *Phase 2*, depicted as a continuous orange line, the trajectory that is generated for this segment takes into account that the quadrotor must reach the desired goal position $\boldsymbol{p}_G$ with a pre-defined orientation parameterised by $\boldsymbol{\lambda}_G$. During this segment, the vehicles are to exchange the parcel;

- In the final segment, *Phase 3*, coloured in green, the vehicle is provided a trajectory from $\boldsymbol{p}_G$ until

a final position (denoted by a red cross in Figure 4.6). It may be imposed that the aircraft finishes at hover. Moreover, it should be ensured that the vehicles move apart from one another during this phase.

All segments of the relay manoeuvre are generated using the method described in the previous section, where collision avoidance constraints are enforced. Orientation and linear velocity constraints are imposed at every waypoint and the continuity of position and its derivatives is enforced between the three segments.

In order to obtain the launch point $\boldsymbol{p}_L$ and the launch velocity $\boldsymbol{v}_L$ necessary for *Phase 2*, the quadrotor dynamics coupled with an attitude control law are integrated backwards in time, starting from the desired state for the relay manoeuvre, $G$. As this backwards integration requires a control law, the thrust input $T$ is kept constant during this phase, whereas a simple proportional derivative (PD) control law computes the body torques input $\boldsymbol{\tau}$,

$$\boldsymbol{\tau} = \boldsymbol{K}_\lambda \boldsymbol{e}_\lambda - \boldsymbol{K}_\omega \boldsymbol{e}_\omega, \tag{4.10}$$

where $\boldsymbol{K}_\lambda$ and $\boldsymbol{K}_\omega$ are diagonal gain matrices. The attitude and angular velocity errors are denoted by $\boldsymbol{e}_\lambda$ and $\boldsymbol{e}_\omega$, respectively,

$$e_\lambda = \boldsymbol{\lambda}_{des} - \boldsymbol{\lambda}, \tag{4.11}$$

$$e_\omega = \boldsymbol{\omega}_{des} - \boldsymbol{\omega}. \tag{4.12}$$

The variable $\boldsymbol{\lambda} = \begin{bmatrix} \phi & \theta & \psi \end{bmatrix}^T$ represents the aircraft orientation parametrised by $Z - X - Y$ Euler angles and the time derivative of these angles is obtained from the angular velocity expressed in $\{\mathcal{B}\}$ by applying a linear transformation $\boldsymbol{Q}$, parametrised by the roll and pitch angles,

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & -\cos\phi\sin\theta \\ 0 & 1 & \sin\phi \\ \sin\theta & 0 & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \quad \Rightarrow \quad \boldsymbol{Q} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ \tan\phi\sin\theta & 1 & -\tan\phi\cos\theta \\ -\sin\theta/\cos\phi & 0 & \cos\theta/\cos\phi \end{bmatrix}. \tag{4.13}$$

Notice that $\boldsymbol{Q}$ is not globally defined, as for $\phi = \pm\pi/2$ a singularity is detected. If the desired attitude reference for *Phase 2* were parametrised by rotation matrices, one could attempt to use the orientation dynamics equation presented in (5.4) to determine $\boldsymbol{\omega}_{des}$, since

$$\boldsymbol{S}(\boldsymbol{\omega}_{des}) = \boldsymbol{R}_{des}^T \dot{\boldsymbol{R}}_{des},$$

however, it was verified that using finite-difference methods to find $\dot{\boldsymbol{R}}_{des}$ led to numerical errors. For the computation of the desired body rate $\boldsymbol{\omega}_{des}$, a first-order numerical differentiation method is employed, to approximate the time derivative of $\boldsymbol{\lambda}_{des}$, together with the expression (4.13). There exist, nonetheless, integration methods on the manifold of rotation matrices, $\mathrm{SO}(3)$ that shall be explored in the future.

The Euler angles reference $\boldsymbol{\lambda}_{des} : [0\ \mathcal{T}) \to \mathbb{R}^3$ for *Phase 2* is defined as the following map,

$$\boldsymbol{\lambda}_{des}(t) = \begin{cases} \begin{bmatrix} 0 & 0 & \psi_G \end{bmatrix}^T & 0 \leq t < t_b, \\ \\ \begin{bmatrix} \phi_G & \theta_G & \psi_G \end{bmatrix}^T & t_b \leq t < \mathcal{T}. \end{cases} \tag{4.14}$$

Provided that $\mathcal{T}$ denotes the duration of *Phase 2*, the choice of $t_b$ establishes the time window for which the vehicles are in a suitable orientation to exchange the parcel. As the dynamics are integrated starting from $G$ and the thrust input $T$ is constant, the position of gripping mechanism attached to vehicles should be taken into account to decide whether $t_b$ should be closer to zero or closer to $\mathcal{T}$. Despite the definition of $\boldsymbol{\lambda}_{des}(\cdot)$ in (4.14) imposing the roll and pitch angles to be equal to zero at the beginning of *Phase 2*, the attitude control law presented in (4.10) is able to track the desired Euler angles step reference, as long as the singularity at $\phi = \pm\pi/2$ is avoided.

The delineated strategy for determining the launch point and velocity $\boldsymbol{p}_L$ and $\boldsymbol{v}_L$, respectively, as demonstrated in Chapter 7, not only facilitates planning relay manoeuvres but also ensures the generation of reference trajectories which allow the parcel exchange between the two vehicles.

The three trajectory segments are obtained by formulating separate optimisation problems. This revealed to be necessary as, even though the solver achieved a solution for the relay manoeuvre reference, in some cases, one could verify that some constraints regarding the continuity of the trajectory were partially violated. Lowering the constraint violation tolerance would render the problem infeasible. Furthermore, having in mind the large number of binary variables which might be involved in the trajectory generation problem, splitting this problem into multiple ones is an advantage.

In order to perform the relay manoeuvre, it is necessary to establish the desired goal point $G_i$ for vehicle $i$. In other words, one should *a priori* picture the desired configuration of each vehicle when the parcel is to be transferred.

# Chapter 5

# Quadrotor Modelling and Control

A model of the dynamics of the quadcopter and a non-linear controller on the Special Euclidean Group $\mathrm{SE}(3)$, based upon the ones presented in [21, 27], are introduced. In this dissertation, this controller serves as a basis for tracking trajectories generated by various methods, including Model Predictive Control (MPC). As the aerodynamic interactions between multirotors should be taken into account to perform relay manoeuvres, a small overview of the relevant concepts from rotor aerodynamics is provided and integral action is considered in the proposed control law with the objective of mitigating the effects of these interactions.

## 5.1 Model

As shown in Figure 5.1, let $\mathcal{W}$ denote the world reference frame, defined by the axes $x_W = \begin{bmatrix}1 & 0 & 0\end{bmatrix}^T$, $y_W = \begin{bmatrix}0 & 1 & 0\end{bmatrix}^T$, and $z_W = \begin{bmatrix}0 & 0 & 1\end{bmatrix}^T$, and $\mathcal{B}$ denote the body frame attached to the vehicle with origin at the centre of mass. The axes $\boldsymbol{x}_B$ and $\boldsymbol{y}_B$ lie in the plane defined by the centre of the four rotors; $\boldsymbol{z}_B$ is perpendicular to this plane and points upwards, in the direction of total thrust. The propeller numbering convention is also included in this figure.

The configuration of the quadrotor is defined by the location of the centre of mass $\boldsymbol{p} = \begin{bmatrix}x & y & z\end{bmatrix}^T$

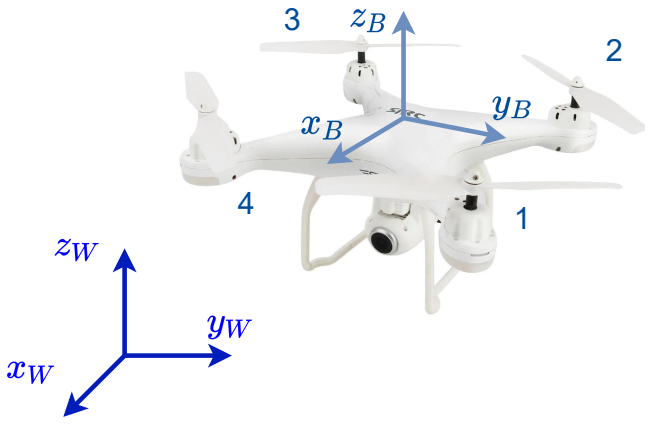



Figure 5.1: Reference frames and rotor numbering.

and its orientation with respect to $\mathcal{W}$, represented by a rotation matrix $\boldsymbol{R}$ from $\mathcal{B}$ to $\mathcal{W}$. Hence, the system evolves on $\mathrm{SE}(3)$.

The rotor with number $n$ has an angular speed $\omega_n$ and produces a force $F_n$ and a torque $\tau_n$ according to

$$F_n = k_F \omega_n^2,\ \tau_n = (-1)^{n+1} k_\tau \omega_n^2,$$

where $k_F$ and $k_\tau$ are positive constants. This model is for a rotor that is hovering in free space [50], no other aerodynamic phenomena is considered. It is defined that rotors 1 and 3 turn clockwise, whereas rotors 2 and 4 turn anticlockwise.

In this manner, the control input to the system $\boldsymbol{u}$ is comprised of the total thrust $T \in \mathbb{R}^+$ and the body torques $\boldsymbol{\tau} = \begin{bmatrix} \tau_x & \tau_y & \tau_z \end{bmatrix}^T \in \mathbb{R}^3$ produced by the four rotors, which can be expressed according to the rotor speeds $\omega_n$ as

$$\boldsymbol{u} = \boldsymbol{M} \begin{bmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F L & 0 & -k_F L \\ -k_F L & 0 & k_F L & 0 \\ k_\tau & -k_\tau & k_\tau & -k_\tau \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}, \tag{5.1}$$

where $L$ represents the distance from the axis of rotation of the rotors to the centre of mass of the vehicle, and $\boldsymbol{M}$ is a matrix which accounts for the position of the rotors expressed in $\mathcal{B}$. For the quadrotor represented in Figure 5.1,

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \sqrt{2}/2 & -\sqrt{2}/2 & 0 \\ 0 & \sqrt{2}/2 & \sqrt{2}/2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Let the linear velocity of the centre of mass in the inertial frame $\mathcal{W}$ be denoted by $\boldsymbol{v} = \begin{bmatrix} v_x & v_y & v_z \end{bmatrix}^T$ and the angular velocity of $\mathcal{B}$ relative to $\mathcal{W}$, expressed in $\mathcal{B}$, by $\boldsymbol{\omega} = \begin{bmatrix} p & q & r \end{bmatrix}^T$. The dynamics of the system are expressed by the following equations,

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \tag{5.2}$$

$$m\dot{\boldsymbol{v}} = -mg\boldsymbol{z}_W + T\boldsymbol{z}_B, \tag{5.3}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}\boldsymbol{S}(\boldsymbol{\omega}), \tag{5.4}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} + \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{J}\boldsymbol{\omega} = \boldsymbol{\tau}, \tag{5.5}$$

where $m \in \mathbb{R}^+$ and $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ represent the mass and the inertia tensor (about the frame $\mathcal{B}$) of the aircraft, respectively. The constant $g = 9.8\ \mathrm{m\,s^{-2}}$ denotes the Earth's gravity.

## 5.2 Aerodynamic Interactions between Rotors

As it is known from rotor aerodynamics, when an rotor is at hover, vortexes form at the blade tips of the propellers that generate a turbulent downwash. If a smaller, or similar-sized quadcopter were to hover

(a) OGE (Out of the Ground Effect) Hover (b) Forward Flight

Figure 5.2: Illustration of blade tip vortexes at different flight conditions.

below the first, the turbulent airflow coming from above would very likely put the rotors of this aircraft in a condition named Vortex Ring State (VRS). In this situation, the rotors stall and, therefore, enough thrust cannot be produced to keep the drone hovering. This condition is similarly named Settling with Power. This is also verified when a rotorcraft is descending vertically at a fast rate. The rotors are engulfed in their own turbulent downwash and, as consequence, the aircraft falls wobbling to the ground unless it 'escapes' to some laminar airflow [43]. However, as depicted in Figure 5.2, when a rotor moves in forward flight the downwash becomes increasingly less turbulent, up to a point the blade tip vortexes are outrun by the aircraft, leaving the rotor operating in undisturbed air.

In view of these considerations on rotor aerodynamics theory, it becomes clear that it is best the aircraft are travelling at some speed to overcome the possible negative effects of being in close proximity.

## 5.3 Nonlinear Control on $\mathrm{SE}(3)$

The overall controller structure is as follows. The outer loop of the controller is responsible for tracking a trajectory defined by the reference position $\boldsymbol{p}_T$, linear velocity $\boldsymbol{v}_T$, and linear acceleration $\boldsymbol{a}_T$. This loop computes the desired thrust vector $\boldsymbol{F}_{des}$ which is used by the inner loop, along with the reference yaw angle $\psi_T$ and its time derivative $\dot{\psi}_T$, and also the reference jerk $\boldsymbol{j}_T$ to define the desired body frame orientation $\boldsymbol{R}_{des}$ and angular velocity $\boldsymbol{\omega}_{des}$. In this manner, the three control moments $\boldsymbol{\tau}$ are computed. Figure 5.3 represents the described controller structure.

For tracking the reference trajectory the position and linear velocity errors are defined as

$$\boldsymbol{e}_p = \boldsymbol{p} - \boldsymbol{p}_T,\ \boldsymbol{e}_v = \boldsymbol{v} - \boldsymbol{v}_T,$$

which are used to define a proportional integral derivative (PID) control law, with the inclusion of the reference linear acceleration $\boldsymbol{a}_T$ as feedforward term,

$$\boldsymbol{F}_{des} = -\boldsymbol{K}_p\boldsymbol{e}_p - \boldsymbol{K}_i \int \boldsymbol{e}_p - \boldsymbol{K}_v\boldsymbol{e}_v + mg\boldsymbol{z}_W + m\boldsymbol{a}_T. \tag{5.6}$$

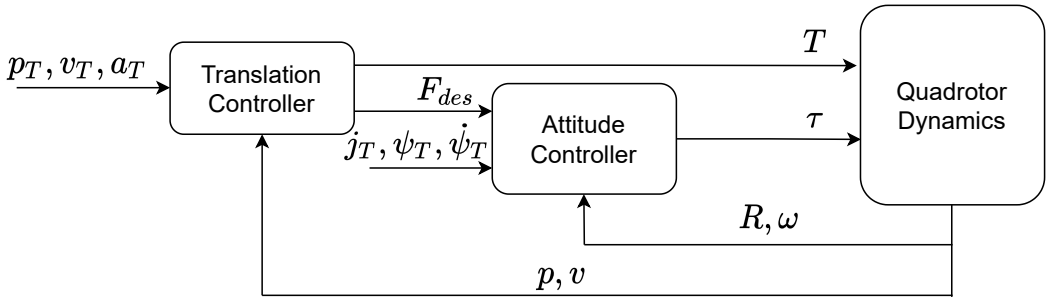



Figure 5.3: Geometric tracking controller on $\mathrm{SE}(3)$ structure proposed in [21].

The matrices $\boldsymbol{K}_p$ and $\boldsymbol{K}_v$ are positive definite and correspond to the position and velocity controller gains; $\boldsymbol{K}_i$ is also a positive definite matrix and denotes the integral gain. In [21], the integral term was not included, however, in the context of this work, it proved to be necessary to mitigate the effects of the aerodynamic interactions between multiple vehicles and to compensate modelling errors in thrust-vs-throttle curves. If the controller requests an input thrust above or below specified bounds, an anti-windup prevents the integral term from accumulating. Furthermore, the following logic is considered for the integral gain,

$$\boldsymbol{\alpha}_j^T \boldsymbol{K}_i = \begin{cases} \mathbf{0}_{1\times 3} & \boldsymbol{\alpha}_j^T |\boldsymbol{e}_p| > \varepsilon_j, \\ \boldsymbol{\alpha}_j^T \boldsymbol{K}_i' & \boldsymbol{\alpha}_j^T |\boldsymbol{e}_p| \leq \varepsilon_j, \end{cases} \tag{5.7}$$

where $\boldsymbol{\alpha}_j$ is a three-dimensional vector with all entries equal to 0, except for the $j^{\text{th}}$ which is equal to 1, and the constants $\varepsilon_j$ are positive numbers up to the choice of the designer. Additionally, the integral term is reset when large variations are observed in the position reference.

The control input $T$ is obtained from the projection of the desired thrust $\boldsymbol{F}_{des}$ onto the body frame axis $\boldsymbol{z}_B$,

$$T = \boldsymbol{F}_{des} \cdot \boldsymbol{z}_B. \tag{5.8}$$

So as to compute the other three inputs, the control torques $\boldsymbol{\tau}$, one begins by determining the desired orientation of the body frame $\boldsymbol{R}_{des}$. By assumption, $\boldsymbol{z}_B$ points in the direction of total thrust, and given that $\|\boldsymbol{F}_{des}\| \neq 0$, the desired $\boldsymbol{z}_B$ is

$$\boldsymbol{z}_{B,des} = \frac{\boldsymbol{F}_{des}}{\|\boldsymbol{F}_{des}\|}.$$

As the reference yaw angle $\psi_T$ is provided, by considering an intermediate yaw rotation from a frame $\mathcal{C}$ to $\mathcal{W}$,

$$\boldsymbol{x}_{C,des} = \begin{bmatrix} \cos\psi_T & \sin\psi_T & 0 \end{bmatrix}^T,$$

$\boldsymbol{x}_{B,des}$ and $\boldsymbol{y}_{B,des}$ are obtained by the following expressions, provided that $\boldsymbol{z}_{B,des} \times \boldsymbol{x}_{C,des} \neq 0$ (in other words, the two vectors are not collinear),

$$\boldsymbol{y}_{B,des} = \frac{\boldsymbol{z}_{B,des} \times \boldsymbol{x}_{C,des}}{\|\boldsymbol{z}_{B,des} \times \boldsymbol{x}_{C,des}\|},$$

$$\boldsymbol{x}_{B,des} = \boldsymbol{y}_{B,des} \times \boldsymbol{z}_{B,des}.$$

The desired body frame orientation is thus given by

$$\boldsymbol{R}_{des} = \begin{bmatrix} \boldsymbol{x}_{B,des} & \boldsymbol{y}_{B,des} & \boldsymbol{z}_{B,des} \end{bmatrix}. \tag{5.9}$$

The desired body angular rate $\boldsymbol{\omega}_{des}$ is determined taking the first derivative of linear velocity dynamics equation (5.3), projecting the resultant expression along $\boldsymbol{z}_B$, using the fact $\dot{T} = \boldsymbol{z}_B \cdot m\boldsymbol{j}_T$, and defining the vector $\boldsymbol{h}_\omega$ as such,

$$\boldsymbol{h}_{\omega,des} = (\boldsymbol{R}_{des}\boldsymbol{\omega}_{des}) \times \boldsymbol{z}_{B,des} = \frac{m}{T}\left(\boldsymbol{j}_T - (\boldsymbol{z}_{B,des} \cdot \boldsymbol{j}_T)\, \boldsymbol{z}_{B,des}\right). \tag{5.10}$$

This expression allows to obtain the components $p$ and $q$ of the desired angular rate,

$$p_{des} = -\boldsymbol{h}_{\omega,des} \cdot \boldsymbol{y}_{B,des},$$
$$q_{des} = \boldsymbol{h}_{\omega,des} \cdot \boldsymbol{x}_{B,des}.$$

The third component of the angular velocity $r$ is computed using the following expression,

$$r_{des} = \dot{\psi}_T \boldsymbol{z}_W \cdot \boldsymbol{z}_{B,des}. \tag{5.11}$$

Finally, the orientation error $\boldsymbol{e}_R$ and the angular velocity error $\boldsymbol{e}_\omega$ are defined as

$$\boldsymbol{e}_R = \frac{1}{2}\boldsymbol{S}^{-1}\left(\boldsymbol{R}_{des}^T\boldsymbol{R} - \boldsymbol{R}^T\boldsymbol{R}_{des}\right), \tag{5.12}$$

$$\boldsymbol{e}_\omega = \boldsymbol{\omega} - \boldsymbol{\omega}_{des}, \tag{5.13}$$

where $\boldsymbol{S}^{-1} : \mathfrak{so}(3) \mapsto \mathbb{R}^3$ corresponds to the inverse of the map defined in (1.1). The control moments to be input to the quadcopter are computed using the established attitude errors,

$$\boldsymbol{\tau} = -\boldsymbol{K}_R\boldsymbol{e}_R - \boldsymbol{K}_\omega\boldsymbol{e}_\omega, \tag{5.14}$$

where $\boldsymbol{K}_R$ and $\boldsymbol{K}_\omega$ are positive definite matrices representing the orientation and angular rates gains.

Proofs of stability and convergence for a similar controller are provided in [27]. It is demonstrated that the attitude dynamics is almost globally exponentially stable in $\mathrm{SO}(3) \times \mathbb{R}^3$. In fact, the continuity of the feedback control law (5.14) implies that the closed-loop vector field on $\mathrm{SO}(3) \times \mathbb{R}^3$ to stabilise a desired orientation and angular velocity must have more than one equilibrium, global asymptotic stability cannot be achieved [34, 37].

# Chapter 6

# Model Predictive Control

This chapter presents an MPC regulator to control the translation of a quadcopter, making it follow a desired reference trajectory, while respecting bounds on thrust and angular rates. The translation dynamics is modelled as a triple integrator, in order to gain access to the orientation and angular velocity of the vehicle. The MPC translation controller is coupled with the geometric attitude control controller on $\mathrm{SO}(3) \times \mathbb{R}^3$, proposed in [21], to ensure the rotorcraft is able to perform aggressive manoeuvres. The necessary conditions to establish the exponential stability of the MPC regulator coupled with the triple integrator are discussed. Finally, a strategy for obtaining the region of attraction of the regulator, using the Dynamic Programming (DP) equations, is presented, based upon [51].

Much of the notation used throughout this chapter is detailed in Section 3.1, where the theoretical concepts regarding MPC are introduced.

## 6.1 MPC Regulator

In Section 3.1, the problem of driving a state of a system to the origin using an MPC controller was addressed. The OCP presented in (3.4), given an initial condition $x_0$, provides a finite control sequence which leads the state to zero. Choosing a coordinate transformation to measure the deviation of $\boldsymbol{x}(\cdot)$ and $\boldsymbol{u}(\cdot)$ from a desired reference $(\boldsymbol{x}_{ref}, \boldsymbol{u}_{ref})$, the same optimisation problem can be employed to track such reference. In this fashion, the following deviation variables are defined,

$$\widetilde{\boldsymbol{x}}(k) = \boldsymbol{x}(k) - \boldsymbol{x}_{ref}(k), \tag{6.1}$$

$$\widetilde{\boldsymbol{u}}(k) = \boldsymbol{u}(k) - \boldsymbol{u}_{ref}(k). \tag{6.2}$$

Leveraging the fact that translation dynamics can be modelled as a triple integrator, an MPC regulator is presented to control the translation of a quadrotor. The translation dynamics of the quadrotor,

expressed in the inertial frame $\mathcal{W}$, can be modelled as a triple integrator,

$$\dot{\boldsymbol{p}} = \boldsymbol{v}, \tag{6.3}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a}, \tag{6.4}$$

$$\dot{\boldsymbol{a}} = \boldsymbol{u}, \tag{6.5}$$

with $\boldsymbol{a} = \begin{bmatrix} a_x & a_y & a_z \end{bmatrix}^T$ being the linear acceleration of the vehicle, driven by the control input $\boldsymbol{u} \in \mathbb{R}^3$, which can be related to the model presented in the previous chapter by defining

$$\boldsymbol{u} = \frac{1}{m}\left(\dot{T}\boldsymbol{z}_B + T\boldsymbol{S}(\boldsymbol{R}\boldsymbol{\omega})\boldsymbol{z}_B\right). \tag{6.6}$$

To start designing the MPC regulator, one starts by discretising the triple integrator dynamics. Considering a sampling time $T_s$, and assuming that the input $\boldsymbol{u}$ is constant between sampling instances,

$$\begin{aligned}
\boldsymbol{p}(k+1) &= \boldsymbol{p}(k) + T_s\boldsymbol{v}(k) + \frac{T_s^2}{2}\boldsymbol{a}(k) + \frac{T_s^3}{6}\boldsymbol{u}(k),\\
\boldsymbol{v}(k+1) &= \boldsymbol{v}(k) + T_s\boldsymbol{a}(k) + \frac{T_s^2}{2}\boldsymbol{u}(k),\\
\boldsymbol{a}(k+1) &= \boldsymbol{a}(k) + T_s\boldsymbol{u}(k).
\end{aligned}$$

Thus, the discretised dynamics are expressed as $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$, where $\boldsymbol{x} = \begin{bmatrix} \boldsymbol{p}^T & \boldsymbol{v}^T & \boldsymbol{a}^T \end{bmatrix}^T$ and the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are readily obtained from the equations above.

Assuming that state and control input references evolve according to the triple integrator dynamics,

$$\boldsymbol{x}_{ref}(k+1) = \boldsymbol{A}\boldsymbol{x}_{ref}(k) + \boldsymbol{B}\boldsymbol{u}_{ref}(k), \tag{6.7}$$

the dynamic model of the deviation variables is

$$\begin{aligned}
\widetilde{\boldsymbol{x}}(k+1) &= \boldsymbol{x}(k+1) - \boldsymbol{x}_{ref}(k+1)\\
&= \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k) - (\boldsymbol{A}\boldsymbol{x}_{ref}(k) + \boldsymbol{B}\boldsymbol{u}_{ref}(k))\\
\widetilde{\boldsymbol{x}}(k+1) &= \boldsymbol{A}\widetilde{\boldsymbol{x}}(k) + \boldsymbol{B}\widetilde{\boldsymbol{u}}(k).
\end{aligned}$$

One very important thing to note, if the reference trajectory $(\boldsymbol{x}_{ref}, \boldsymbol{u}_{ref})$ corresponds to a constant setpoint $(\boldsymbol{x}_{sp}, \boldsymbol{u}_{sp})$, it must also satisfy (6.7). For the triple integrator dynamics, the only *reachable* setpoints are those which correspond to the drone being at hover, at some position $\boldsymbol{p}_{sp}$, and, therefore, a constant setpoint $(\boldsymbol{x}_{sp}, \boldsymbol{u}_{sp})$ must satisfy $\boldsymbol{x}_{sp} = \begin{bmatrix} \boldsymbol{p}_{sp}^T & \boldsymbol{0}_{1\times 3} & \boldsymbol{0}_{1\times 3} \end{bmatrix}^T$ and $\boldsymbol{u}_{sp} = \boldsymbol{0}_{3\times 1}$ such that $(\boldsymbol{x}_{sp}, \boldsymbol{u}_{sp}) \in \mathbb{G}$. Additionally, the polynomial trajectories generated by the methods proposed in Chapter 4 also satisfy the triple integrator dynamics.

In this manner, the optimisation that follows is formulated to drive $\widetilde{\boldsymbol{x}}(\cdot)$ and $\widetilde{\boldsymbol{u}}(\cdot)$ to the origin,

$$\begin{aligned}
\underset{\widetilde{\boldsymbol{x}},\widetilde{\boldsymbol{u}}}{\text{minimise}} \quad & \sum_{k=0}^{N-1} \ell(\widetilde{\boldsymbol{x}}(k),\widetilde{\boldsymbol{u}}(k)) + V_N(\widetilde{\boldsymbol{x}}(N)) \\
\text{s.t.} \quad & \widetilde{\boldsymbol{x}}(k+1) = \boldsymbol{A}\widetilde{\boldsymbol{x}}(k) + \boldsymbol{B}\widetilde{\boldsymbol{u}}(k), \text{ for } k = 0,1,\ldots,N-1, \\
& \widetilde{\boldsymbol{x}}(0) = \widetilde{\boldsymbol{x}}_0 \in \mathbb{X}, \\
& (\widetilde{\boldsymbol{x}}(k),\widetilde{\boldsymbol{u}}(k)) \in \mathbb{G}, \text{ for } k = 0,1,\ldots,N-1, \\
& \widetilde{\boldsymbol{x}}(N) \in \mathbb{X}_f \subseteq \mathbb{X},
\end{aligned} \tag{6.8}$$

where $\widetilde{\boldsymbol{x}}_0 = \boldsymbol{x}_0 - \boldsymbol{x}_{ref}(0)$, $\boldsymbol{x}_0$ being the initial condition of the system $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$.

In what concerns state and input constraints, the maximum thrust and angular rates the vehicle can achieve should be taken into account. Given the maximum value of thrust $T_{max} \in \mathbb{R}^+$ the rotating propellers can produce, according to the linear velocity dynamics equation (5.3),

$$a_x^2 + a_y^2 + (a_z + g)^2 \leq \left(\frac{T_{max}}{m}\right)^2. \tag{6.9}$$

Although this inequality constraint could be added to the optimisation problem without causing any harm, as it is convex, in the spirit of determining analytically the region of attraction of the controller, the boundary of this sphere that limits the linear acceleration the vehicle can achieve is approximated by a polyhedron. Parameterising this boundary using spherical coordinates,

$$\begin{aligned}
a_x &= R\sin\alpha\cos\beta, \\
a_y &= R\sin\alpha\sin\beta, \\
a_z &= R\cos\alpha - g,
\end{aligned}$$

with $R = T_{max}/m$, $0 \leq \alpha \leq \pi$, and $0 \leq \beta < 2\pi$, the edge of the sphere is sampled with ease. Computing the convex hull of the sampled points, one can write a set of linear inequalities that approximate (6.9),

$$\mathbb{A} \coloneqq \{\boldsymbol{a} \in \mathbb{R}^3 \mid \boldsymbol{N}\boldsymbol{a} \leq \boldsymbol{\nu}\}. \tag{6.10}$$

The roll and pitch rates can be expressed by a dot product between $\boldsymbol{u}$ and the basis vectors that define the body frame $\mathcal{B}$,

$$p = -\frac{m}{T}\boldsymbol{u} \cdot \boldsymbol{y}_B, \tag{6.11}$$

$$q = \frac{m}{T}\boldsymbol{u} \cdot \boldsymbol{x}_B. \tag{6.12}$$

To bound roll and pitch rates, constraints are imposed on the input variable $\boldsymbol{u}$ to limit the norm of this vector. Without any loss of generality, it is assumed that $p_{max} = q_{max} = \Omega \in \mathbb{R}_0^+$ and, therefore,

$$\|\boldsymbol{u}\| \leq \frac{\widetilde{T}}{m}\Omega, \tag{6.13}$$

where $\widetilde{T} \in \mathbb{R}^+$ such that $\widetilde{T} \leq T_{max}$ is up to the choice of the designer. As the angular rates a quadrotor can achieve depend on the value of thrust, one cannot fully ensure that $|p| \leq \Omega$ and $|q| \leq \Omega$. The choice of $\widetilde{T}$ indicates, in a sense, how willing is the designer to the angular rate inputs surpassing the maximum achievable value. Selecting $\widetilde{T} = T_{max}$, for $T < T_{max}$ the quadrotor might rotate with a higher angular velocity than $\Omega$. Similar to (6.9), (6.13) represents a sphere, but with radius $(\widetilde{T}/m)\Omega$. Using again spherical coordinates, points are sampled from the edge of sphere and the convex hull of those points is computed, thus the set of admissible control inputs is a polyhedron,

$$\mathbb{U} \coloneqq \{\boldsymbol{u} \in \mathbb{R}^m \mid \boldsymbol{\Xi u} \leq \boldsymbol{\xi}\}. \tag{6.14}$$

Additional bound constraints are enforced on the linear velocity to ensure the vehicle does not reach velocities for which the aerodynamic drag force due to the fuselage of the aircraft is no longer negligible; constraints on position may also be imposed. These constraints are expressed as the following set,

$$\mathbb{B} \coloneqq \{(\boldsymbol{p}, \boldsymbol{v}) \in \mathbb{R}^3 \times \mathbb{R}^3 \mid \boldsymbol{\Lambda p} \leq \boldsymbol{\lambda},\ \boldsymbol{Mv} \leq \boldsymbol{\mu}\}, \tag{6.15}$$

and, hence, the set of admissible states $\mathbb{X}$ is obtained from (6.10) and (6.15),

$$\mathbb{X} \coloneqq \mathbb{A} \times \mathbb{B}, \tag{6.16}$$

where $\times$ denotes the Cartesian product, $\mathbb{A} \times \mathbb{B} \coloneqq \{(\boldsymbol{a}, \boldsymbol{b}) \mid \boldsymbol{a} \in \mathbb{A},\ \boldsymbol{b} \in \mathbb{B}\}$.

Finally, the stage and final cost functions, $\ell(\cdot, \cdot)$ and $V_f(\cdot)$, respectively, are selected as follows

$$\begin{aligned} \ell(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}) &= \widetilde{\boldsymbol{x}}^T \boldsymbol{Q} \widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{u}}^T \boldsymbol{R} \widetilde{\boldsymbol{u}}, \\ V_f(\widetilde{\boldsymbol{x}}) &= \widetilde{\boldsymbol{x}}^T \boldsymbol{P} \widetilde{\boldsymbol{x}}, \end{aligned}$$

where $\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{P}$ are positive definite matrices. The final cost $V_f(\cdot)$ is chosen as the value function of the infinite horizon unconstrained problem and, thus, $\boldsymbol{P}$ corresponds to the solution to the discrete algebraic Riccati Equation,

$$\boldsymbol{P} = \boldsymbol{Q} + \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} - \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{B} \left( \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R} \right)^{-1} \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{A}. \tag{6.17}$$

The advantage of using the infinite horizon LQR value function comes from the fact that for the terminal states $\widetilde{\boldsymbol{x}}(N)$ that lie in $\mathbb{X}_f \subseteq \mathbb{X}$, the terminal control law $\boldsymbol{\kappa}_f(\widetilde{\boldsymbol{x}}) = \boldsymbol{K} \widetilde{\boldsymbol{x}}$, with

$$\boldsymbol{K} = - \left( \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{B} + \boldsymbol{R} \right)^{-1} \boldsymbol{B}^T \boldsymbol{P} \boldsymbol{A}, \tag{6.18}$$

makes the error between the state and the reference trajectory $\widetilde{\boldsymbol{x}}(\cdot)$ asymptotically converge to zero. In what regards the choice of the terminal set, one should have in mind that the controller $\boldsymbol{\kappa}_f(\cdot)$ does not necessarily satisfy the control and state constraints.

In order to obtain a polytopic terminal set, one may compute the maximal invariant constraint admissible set for $\widetilde{\boldsymbol{x}}(k+1) = (\boldsymbol{A} + \boldsymbol{BK}) \widetilde{\boldsymbol{x}}(k)$, that is, $\mathbb{X}_f \coloneqq \{\widetilde{\boldsymbol{x}} \in \mathbb{X} \mid \boldsymbol{K} \widetilde{\boldsymbol{x}} \in \mathbb{U}\}$. The constraint set $\mathbb{G} \coloneqq \mathbb{X} \times \mathbb{U}$

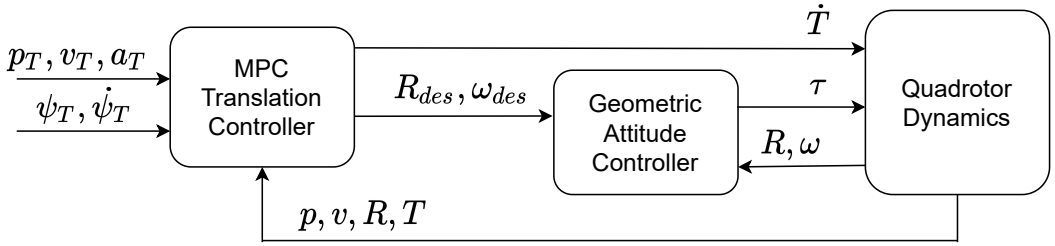



Figure 6.1: Model predictive translation controller coupled with the geometric attitude controller on $\mathrm{SO}(3) \times \mathbb{R}^3$ proposed in [21] combined architecture.

is expressed as the following polytope,

$$\mathbb{G} \coloneqq \{(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}) \in \mathbb{R}^n \times \mathbb{R}^m \mid \boldsymbol{G}\widetilde{\boldsymbol{x}} + \boldsymbol{H}\widetilde{\boldsymbol{u}} \leq \boldsymbol{\psi} - \boldsymbol{G}\boldsymbol{x}_{ref} - \boldsymbol{H}\boldsymbol{u}_{ref}\}\,, \tag{6.19}$$

with $n = 9$ and $m = 3$ for the triple integrator dynamics. The matrix $\boldsymbol{G}$ is obtained from constructing a block diagonal matrix with matrices $\boldsymbol{\Lambda}$, $\boldsymbol{M}$, and $\boldsymbol{N}$, defined in (6.10) and (6.15); the matrix $\boldsymbol{H}$ is obtained from $\boldsymbol{\Xi}$, defined in (6.14); the vector $\boldsymbol{\psi}$ is obtained by stacking the vectors $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\boldsymbol{\nu}$, and $\boldsymbol{\xi}$, defined in (6.10), (6.14), and (6.15). Keep in mind that the constraints (6.9) and (6.13) could be added to the OCP in their non-linear form, the constraint set would still be convex, but no longer polyhedral.

Provided that $\widetilde{\boldsymbol{u}} = \boldsymbol{K}\widetilde{\boldsymbol{x}}$, according to (6.19), the terminal set is defined as,

$$\mathbb{X}_f \coloneqq \{\widetilde{\boldsymbol{x}} \in \mathbb{R}^n \mid (\boldsymbol{G} + \boldsymbol{H}\boldsymbol{K})\widetilde{\boldsymbol{x}} \leq \boldsymbol{\psi} - \boldsymbol{G}\boldsymbol{x}_{ref} - \boldsymbol{H}\boldsymbol{u}_{ref}\}\,. \tag{6.20}$$

Alternatively, the terminal set might be a sublevel set of $V_f(\cdot)$, that is $\mathbb{X}_f \coloneqq \{\boldsymbol{x} \in \mathbb{X} \mid V_f(\boldsymbol{x}) \leq \alpha\}$, with $\alpha \in \mathbb{R}^+$. This fashion of defining the terminal set allows for selecting an ellipsoidal subset of the maximum constraint invariant set, which is convex, but not polyhedral. When choosing the value for $\alpha$, one should ensure that Assumption 3.4 is verified, there should exist an $\boldsymbol{u}$ such that $(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{G}$, for any $\boldsymbol{x} \in \mathbb{X}_f$.

As the presented MPC regulator stands, it is able to plan position, velocity, acceleration, and jerk (the control input of the triple integrator) trajectories regarding the thrust and angular velocity limitations of a quadrotor. From these trajectories and a yaw angle trajectory, one can obtain thrust, $T$; the first time derivative of this force, $\dot{T}$; the body frame $z$ axis, the direction of thrust, $\boldsymbol{z}_B$; and angular rates, $\boldsymbol{\omega}$ to control the quadcopter.

Given $\boldsymbol{u}$ and the linear acceleration vector $\boldsymbol{a}$, one can find $\boldsymbol{\omega}$ such that (6.6) is satisfied. From the linear velocity dynamics equation (5.3), it is verified that

$$T = m\left\|\boldsymbol{a} + g\boldsymbol{z}_W\right\|, \tag{6.21}$$

$$\boldsymbol{z}_B = \frac{\boldsymbol{a} + g\boldsymbol{z}_W}{\left\|\boldsymbol{a} + g\boldsymbol{z}_W\right\|}. \tag{6.22}$$

Taking the time derivative of $T$ and using the definition of $\boldsymbol{z}_B$,

$$\dot{T} = m\left(\boldsymbol{z}_B^T\dot{\boldsymbol{a}}\right) = m(\boldsymbol{z}_B \cdot \boldsymbol{u}). \tag{6.23}$$

Rearranging (6.6), a system of linear equations is established to determine $\boldsymbol{\omega}$,

$$\begin{aligned} \dot{T}\boldsymbol{z}_B - m\boldsymbol{u} &= T\boldsymbol{S}(\boldsymbol{z}_B)\boldsymbol{R}\boldsymbol{\omega} \\ &= T(p\boldsymbol{y}_B - q\boldsymbol{x}_B). \end{aligned} \tag{6.24}$$

To compute $\boldsymbol{x}_B$ and $\boldsymbol{y}_B$, in addition to the desired direction of thrust $\boldsymbol{z}_B$, a yaw angle $\psi$ must be provided. Given this angle, it is straightforward to determine the roll and pitch rates, $p$ and $q$, respectively. Using (5.11), one may obtain the desired yaw rate, $r$ from the first time derivative of $\psi$.

In order to control the translation of the vehicle, $\dot{T}$ should be provided as input, since it is a function of $\boldsymbol{u}$. Alternatively, one may compute thrust inputs from $\dot{T}$ using a numerical integration method; to control the attitude of the vehicle, $\boldsymbol{\omega}$ is given as input.

However, for aggressive manoeuvring, the angular velocity dynamics, expressed in (5.5), should be regarded to fully model the attitude dynamics. Hence, the MPC regulator for the triple integrator dynamics is coupled with the attitude control law introduced in (5.14) to determine torque inputs $\boldsymbol{\tau}$ from the orientation and angular velocity velocities defined in (5.12) and (5.13) to stabilise the attitude dynamics.

Finally, in Figure 6.1, a block diagram of the combined control architecture of the MPC translation regulator is presented, considering the triple integrator dynamics, and the attitude control law on $\mathrm{SO}(3) \times \mathbb{R}^3$ proposed in [21] is presented. The block 'Quadrotor dynamics' corresponds to the equations (5.2), (5.3), (5.4), and (5.5). The equation (6.23) is also considered to provide $\boldsymbol{u}$ as input to the vehicle. Given a state-space reference trajectory $(\boldsymbol{p}_T, \boldsymbol{v}_T, \boldsymbol{a}_T)$ for the triple integrator and $(\psi_T, \dot{\psi}_T)$ trajectories, the MPC regulator provides $\left\{\dot{T}(k), \boldsymbol{R}(k), \boldsymbol{\omega}(k)\right\}$, $k = 0, 1, \ldots, N$, with $N$ being the considered control horizon, at every iteration. The first elements of the sequence $\{\boldsymbol{R}(\cdot), \boldsymbol{\omega}(\cdot)\}$, $\boldsymbol{R}_{des} \coloneqq \boldsymbol{R}(0)$ and $\boldsymbol{\omega}_{des} \coloneqq \boldsymbol{\omega}(0)$, are provided as virtual inputs to the attitude controller. The expressions (5.12), (5.13), and (5.14) allow to obtain a torque input $\boldsymbol{\tau}$ to stabilise the attitude of the vehicle. Similarly, the first element of the sequence $\left\{\dot{T}(k)\right\}$ is given as input to the quadrotor to control its translational dynamics. Along with the current position and linear velocity of the aircraft to set the initial condition of the OCP, the current orientation and thrust are provided to the translation controller to obtain the linear acceleration initial condition, using the linear velocity dynamics equation (5.3).

In Section 7.4, the simulation results of two quadrotors coupled with the controller proposed are presented and analysed.

## 6.2 Stability Discussion

In this section, the stability of the triple integrator dynamics in closed-loop with the MPC regulator (6.8) is established in this section. Even though stability proofs for the full quadrotor dynamics in closed-loop with the MPC translation controller and the geometric attitude controller fall out the scope of this thesis, the conclusions drawn in this section are extremely important, so as to ensure mishaps associated with the potential infeasibility of the MPC regulator do not take place.

The set of admissible states $\mathbb{X}$, defined in (6.16), corresponds to intersection of closed half-spaces and, thus, this set is closed; the set of admissible control inputs $\mathbb{U}$, defined in (6.14), is a bounded polyhedron and, hence, $\mathbb{U}$ is compact. Given the definition of $\mathbb{G}$ in (6.19), provided that all entries of vector $\boldsymbol{\psi}$ are greater or equal to zero, both sets, $\mathbb{X}$ and $\mathbb{U}$, contain the origin within if and only if the reference trajectory satisfies the state and control constraints,

$$\boldsymbol{G}\boldsymbol{x}_{ref}(k) + \boldsymbol{H}\boldsymbol{u}_{ref}(k) \leq \boldsymbol{\psi}, \text{ for } k = 0, 1, \ldots, N-1. \tag{6.25}$$

Since $\ell(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}) = \widetilde{\boldsymbol{x}}^T \boldsymbol{Q}\widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{u}}^T \boldsymbol{R}\widetilde{\boldsymbol{u}}$, with $\boldsymbol{Q}$ and $\boldsymbol{R}$ positive definite, there is a constant $c_1 \in \mathbb{R}^+$ for which the value function of (6.8) verifies $V_N^0(\widetilde{\boldsymbol{x}}) \geq \ell(\widetilde{\boldsymbol{x}}, \widetilde{\boldsymbol{u}}) \geq c_1 \left\|\widetilde{\boldsymbol{x}}\right\|^2$, $\forall \widetilde{\boldsymbol{x}} \in \mathcal{X}_N$, and there exists $c_f \in \mathbb{R}^+$ which sets an upper bound on the terminal cost, $V_f(\widetilde{\boldsymbol{x}}) \leq c_f \left\|\widetilde{\boldsymbol{x}}\right\|^2$, $\forall \widetilde{\boldsymbol{x}} \in \mathbb{X}_f$, with $c_f$ equal to the largest eigenvalue of $\boldsymbol{P}$.

It can be verified that $V_f(\cdot)$ is a Lyapunov function for the system in closed-loop with the terminal control law $\widetilde{\boldsymbol{u}} = \boldsymbol{K}\widetilde{\boldsymbol{x}}$, with $\boldsymbol{K}$ defined as in (6.18), since it follows that the value of this function decreases asymptotically along the trajectories of the closed-loop system $\widetilde{\boldsymbol{x}}(k+1) = \boldsymbol{\mathcal{A}}\widetilde{\boldsymbol{x}}(k)$, with $\boldsymbol{\mathcal{A}} = \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}$, for all $\widetilde{\boldsymbol{x}} \in \mathbb{X}_f$,

$$\begin{aligned} V_f(\boldsymbol{\mathcal{A}}\widetilde{\boldsymbol{x}}) - V_f(\widetilde{\boldsymbol{x}}) &\leq -\ell(\widetilde{\boldsymbol{x}}, \boldsymbol{K}\widetilde{\boldsymbol{x}}) \\ \widetilde{\boldsymbol{x}}^T \boldsymbol{\mathcal{A}}^T \boldsymbol{P} \boldsymbol{\mathcal{A}}\widetilde{\boldsymbol{x}} + \widetilde{\boldsymbol{x}}^T \boldsymbol{\mathcal{Q}}\widetilde{\boldsymbol{x}} - \widetilde{\boldsymbol{x}}^T \boldsymbol{P}\widetilde{\boldsymbol{x}} &\leq 0. \end{aligned}$$

To verify this inequality holds true, note that the discrete algebraic Riccati equation (6.17) is equivalent to the Lyapunov equation, $\boldsymbol{\mathcal{A}}^T \boldsymbol{P} \boldsymbol{\mathcal{A}} - \boldsymbol{P} = \boldsymbol{\mathcal{Q}}$, where $\boldsymbol{\mathcal{Q}} = \boldsymbol{Q} + \boldsymbol{K}^T \boldsymbol{R} \boldsymbol{K}$.

Furthermore, the terminal control law $\widetilde{\boldsymbol{u}} = \boldsymbol{K}\widetilde{\boldsymbol{x}}$ also ensures the terminal constraint set $\mathbb{X}_f$ is control invariant. This controller drives the state asymptotically to the origin and, as $\mathbb{X}_f$ was chosen to be the maximal invariant constraint admissible set, $\mathbb{X}_f \coloneqq \{\widetilde{\boldsymbol{x}} \in \mathbb{X} \mid \boldsymbol{K}\widetilde{\boldsymbol{x}} \in \mathbb{U}\}$, the state and control constraints are satisfied. As a consequence of Proposition 3.1, $\mathbb{X}_f$ being control invariant ensures $\mathcal{X}_N$ is positive invariant. Since $\mathbb{X}$ is closed and $\mathbb{U}$ is compact, $\mathbb{X}_f$ is closed. Similarly to $\mathbb{X}$ and $\mathbb{U}$, $\mathbb{X}_f$ contains the origin within if and only if the terminal $\boldsymbol{x}_{ref}$ and $\boldsymbol{u}_{ref}$ satisfy the state and control constraints,

$$\boldsymbol{G}\boldsymbol{x}_{ref}(N) + \boldsymbol{H}\boldsymbol{u}_{ref}(N) \leq \boldsymbol{\psi}. \tag{6.26}$$

From this analysis, one can conclude that Assumptions 3.1, 3.2, 3.3, and 3.4 are satisfied, with the condition that (6.25) and (6.26) hold. From Proposition 3.2, one verifies that the value function $V_N^0(\cdot)$ decreases exponentially along the trajectories of the closed-loop system,

$$V_N^0\left(\boldsymbol{A}\widetilde{\boldsymbol{x}} + \boldsymbol{B}\boldsymbol{\kappa}_N(\widetilde{\boldsymbol{x}})\right) - V_N^0(\widetilde{\boldsymbol{x}}) \leq -c_1 \left\|\widetilde{\boldsymbol{x}}\right\|^2, \ \forall \widetilde{\boldsymbol{x}} \in \mathcal{X}_N, \tag{6.27}$$

and that the value function has an upper bounded in $\mathbb{X}_f$,

$$V_N^0(\widetilde{\boldsymbol{x}}) \leq c_f \left\|\widetilde{\boldsymbol{x}}\right\|^2, \ \forall \widetilde{\boldsymbol{x}} \in \mathbb{X}_f. \tag{6.28}$$

Provided that (6.26) holds, according to Proposition 3.3, there is a constant $c_2 \in \mathbb{R}^+$ for which

$$V_N^0(\widetilde{\boldsymbol{x}}) \leq c_2 \left\|\widetilde{\boldsymbol{x}}\right\|^2, \ \forall \widetilde{\boldsymbol{x}} \in \mathcal{X}_N. \tag{6.29}$$

Hence, by Theorem 3.1, the origin is exponentially stable for $\widetilde{\boldsymbol{x}}(k+1) = \boldsymbol{A}\widetilde{\boldsymbol{x}}(k) + \boldsymbol{B}\boldsymbol{\kappa}_N(\widetilde{\boldsymbol{x}}(k))$ in $\mathcal{X}_N$, provided that the reference trajectory satisfies the state and control constraints.

## 6.3 Region of Attraction

When the reference trajectory $(\boldsymbol{x}_{ref}, \boldsymbol{u}_{ref})$ is time-varying, as the horizon recedes over this trajectory, the terminal and constraint sets, $\mathbb{X}_f$ and $\mathbb{G}$, respectively, change at every iteration the OCP is solved to obtain a control sequence, which means that the region of attraction of the MPC controller varies over time. This can easily be verified by revisiting Definition 3.2, which provides a concrete definition of what the region of attraction of an MPC controller is.

Despite $\mathbb{X}_f$ and $\mathbb{G}$ being time-varying, the necessary properties of these sets to ensure *recursive feasibility* and closed-loop stability still hold, in nominal conditions and assuming that the reference trajectory satisfies the state and control constraints. The constraint set $\mathbb{G}$, defined in (6.19), is always a closed set, for being the intersection of closed half-spaces; the terminal set $\mathbb{X}_f$ is also the intersection of closed half-spaces, thus, closed; the terminal control law $\boldsymbol{\kappa}(\widetilde{\boldsymbol{x}}) = \boldsymbol{K}\widetilde{\boldsymbol{x}}$ ensures this set is positive invariant. In this manner, if the initial condition of the system $\boldsymbol{x}(0)$ lies in the region of attraction, computed for the initial reference state and control input, $\boldsymbol{x}_{ref}(0)$ and $\boldsymbol{u}_{ref}(0)$, respectively, the successor states will remain in the region of attraction of the MPC controller, though its description changes over time.

In order to compute $\mathcal{X}_N$ using the definition of feasibility set (3.7), it is necessary to compute all feasibility sets $\mathcal{X}_j, \ j = 1, 2, \ldots, N$. Starting from $\mathcal{X}_0 \coloneqq \mathbb{X}_f$, one determines $\mathcal{X}_1$, the set of states in $\mathbb{X}$ which in 1 step are driven to $\mathbb{X}_f$. Provided the set $\mathcal{X}_1$, $\mathcal{X}_2$ is the set of states in $\mathbb{X}$ which in 1 step are driven to $\mathcal{X}_1$, is obtained, and so on, progressing in this fashion, until $\mathcal{X}_N$ is computed. Provided that the system dynamics are linear and the constraint and terminal sets, $\mathbb{G} \coloneqq \mathbb{X} \times \mathbb{U}$ and $\mathbb{X}_f$, respectively, are polyhedral, all feasibility sets are also polyhedral.

Let $\mathbb{A}_j$ denote the set of $(\boldsymbol{x}, \boldsymbol{u})$ which lie in $\mathbb{G} \coloneqq \{(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^n \times \mathbb{R}^m \mid \boldsymbol{E}\boldsymbol{x} + \boldsymbol{F}\boldsymbol{u} \leq \boldsymbol{\xi}\}$ and, when propagated by the system dynamics $\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k)$, the successor state lies in $\mathcal{X}_{j-1}, \ \forall j \in \mathbb{N}$,

$$\mathbb{A}_j \coloneqq \{(\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{R}^n \times \mathbb{R}^m \mid \boldsymbol{G}\boldsymbol{x} + \boldsymbol{H}\boldsymbol{u} \leq \boldsymbol{\psi}\}. \tag{6.30}$$

The projection of $\mathbb{A}_j$ onto $\mathbb{R}^n$ is the feasibility set $\mathcal{X}_j$,

$$\mathcal{X}_j \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n \mid \exists \boldsymbol{u} \in \mathbb{R}^m \text{ such that } (\boldsymbol{x}, \boldsymbol{u}) \in \mathbb{A}_j\}, \ \forall j \in \mathbb{N}. \tag{6.31}$$

In [51], algorithms based on the Fourier-Motzkin elimination method were presented to project an

polyhedral convex set onto a subspace and to express $\mathcal{X}_j$ as

$$\mathcal{X}_j \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{P}\boldsymbol{x} \leq \boldsymbol{\gamma}\}, \tag{6.32}$$

Firstly assuming that $m = 1$, Algorithm 1 introduces the procedure to eliminate the variable $\boldsymbol{u} \in \mathbb{R}$ from $\mathbb{A}_j \subset \mathbb{R}^n \times \mathbb{R}$ to obtain $\mathcal{X}_j \subset \mathbb{R}^n$ (6.32). In the sequel, the $i^{\text{th}}$ row of a matrix $\boldsymbol{F} \in \mathbb{R}^{n \times m}$ is denoted by $\boldsymbol{F}^i$.

**Algorithm 1** Projection of the polytope $\mathbb{A}_j$ onto $\mathbb{R}^n$ with $m = 1$ (from [51])

1: **procedure** POLYTOPEPROJECTION1($\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{\psi}$)
2: Identify the following subsets of $\{1, 2, \cdots, s\}$:
$I^0 \coloneqq \{i \mid \boldsymbol{H}^i = 0\}$,
$I^+ \coloneqq \{i \mid \boldsymbol{H}^i > 0\}$,
$I^- \coloneqq \{i \mid \boldsymbol{H}^i < 0\}$.
3: Let $\boldsymbol{C} = \begin{bmatrix}\boldsymbol{G} & \boldsymbol{\psi}\end{bmatrix}$,
4: Form the matrix $\begin{bmatrix}\boldsymbol{P} & \boldsymbol{\gamma}\end{bmatrix}$, whose rows are the elements of the sets:
$\{\boldsymbol{C}^i \mid i \in I^0\}$,
$\{\boldsymbol{H}^i\boldsymbol{C}^j - \boldsymbol{H}^j\boldsymbol{C}^i \mid i \in I^+,\ j \in I^-\}$.
5: **return** $\begin{bmatrix}\boldsymbol{P} & \boldsymbol{\gamma}\end{bmatrix}$
6: **end procedure**

When the sets $I^+$ and $I^-$ are empty, it is evident that the rows of $\left[\boldsymbol{P} \quad \boldsymbol{\gamma}\right]$ are the elements of $\{\boldsymbol{C}^i \mid i \in I^0\}$. If these sets are not empty, $\left[\boldsymbol{P} \quad \boldsymbol{\gamma}\right]$ also contains rows which can be expressed as $(\boldsymbol{H}^+ \otimes \boldsymbol{C}^- - \boldsymbol{H}^- \otimes \boldsymbol{C}^+)$, where $\boldsymbol{H}^+ \coloneqq \{\boldsymbol{H}^i \mid i \in I^+\}$, $\boldsymbol{C}^+ \coloneqq \{\boldsymbol{C}^i \mid i \in I^+\}$, $\boldsymbol{H}^- \coloneqq \{\boldsymbol{H}^i \mid i \in I^-\}$, and $\boldsymbol{C}^- \coloneqq \{\boldsymbol{C}^i \mid i \in I^-\}$; the symbol $\otimes$ represents the Kronecker product. The proof of this algorithm is thoroughly presented in [51].

If $m > 1$, the algorithm just presented can be repeated $m$ times, thus eliminating one component of the vector $\boldsymbol{u} \in \mathbb{R}^m$ at a time. Given set $\mathbb{A}_j^k$, for $k \in \{m-1, \cdots, 0\}$, let $\boldsymbol{x}_k = (\boldsymbol{x}, u_1, \cdots, u_k) \in \mathbb{R}^{n+k}$, $\mathbb{A}_j^m \coloneqq \mathbb{A}_j$, and

$$\mathbb{A}_j^k = \left\{\boldsymbol{x}_k \in \mathbb{R}^{n+k} \mid \exists u_{k+1} \text{ such that } (\boldsymbol{x}_k, u_{k+1}) \in \mathbb{A}_j^{k+1}\right\}. \tag{6.33}$$

From these definitions, $\mathbb{A}_j^0 \coloneqq \mathcal{X}_j$, with $\mathcal{X}_j$ expressed as in (6.31).

Algorithm 2 presents the steps to be followed to obtain $\mathcal{X}_j$ from $\mathbb{A}_j$, when $m > 1$. In line 3 of this algorithm, all columns of matrix $\left[\boldsymbol{G} \quad \boldsymbol{H}\right]$ except for the last are kept in matrix $\boldsymbol{G}_m$. In this manner, according to (6.33), for $k = m - 1$, given that $\mathbb{A}_j^{k+1} \coloneqq \mathbb{A}_j$,

$$\mathbb{A}_j^k \coloneqq \left\{\boldsymbol{x}_k \in \mathbb{R}^{n+k} \mid \exists u_{k+1} \text{ such that } \boldsymbol{G}_{k+1}\boldsymbol{x}_k + \boldsymbol{H}_{k+1}u_{k+1} \leq \boldsymbol{\psi}\right\}.$$

In the first iteration of the *while* loop in lines 5-11, $u_{k+1}$ is eliminated using Algorithm 1 to determine $\boldsymbol{P}_k$ and $\boldsymbol{\gamma}_k$ such that

$$\mathbb{A}_j^k \coloneqq \left\{\boldsymbol{x}_k \in \mathbb{R}^{n+k} \mid \boldsymbol{P}_k\boldsymbol{x}_k \leq \boldsymbol{\gamma}_k\right\}.$$

In the following iterations, the remaining entries of the vector $\boldsymbol{u}$ are also eliminated. The algorithm ends when $\boldsymbol{P} \coloneqq \boldsymbol{P}_0$ and $\boldsymbol{\gamma} \coloneqq \boldsymbol{\gamma}_0$ are obtained.

Using the Fourier-Motzkin elimination method, the feasibility sets $\mathcal{X}_j \coloneqq \{\boldsymbol{x} \in \mathbb{R}^n \mid \boldsymbol{P}\boldsymbol{x} \leq \boldsymbol{\gamma}\}$ may many include redundant constraints. The boundary of each of the feasibility sets is the intersection of hyperplanes $\boldsymbol{P}^i\boldsymbol{x} = \boldsymbol{\gamma}^i,\ i = 1, 2, \ldots, K$, with $K$ being the number of rows of matrix $\boldsymbol{P}$. Assuming that $\boldsymbol{\gamma}^i > 0$, the rows of $\boldsymbol{P}$ are re-scaled such that $\left(\boldsymbol{P}^i/\boldsymbol{\gamma}^i\right)\boldsymbol{x} = \widetilde{\boldsymbol{P}}^i\boldsymbol{x} = 1$. Let $\widetilde{\boldsymbol{P}} \coloneqq \left\{\widetilde{\boldsymbol{P}}^i \mid i = 1, 2, \ldots, K\right\}$, the convex hull of the rows of $\widetilde{\boldsymbol{P}}$ defines which of these are, in fact, necessary to define the boundary of the feasibility sets.

As stated in Section 3.1.1, obtaining $\mathcal{X}_N$ using the DP equations is possible if state dimension and the control horizon $N$ are suitably small. In the case of the quadrotor translation dynamics (modelled as a triple integrator), the state dimension is $n = 9$. On top of that, keep in mind that high number of linear constraints, defined by the sets $\mathbb{G}$ and $\mathbb{X}_f$, may render the method discussed in this section intractable. As an alternative, one may estimate the $\mathcal{X}_N$ by simulation.

A practical measure to ensure the state of the system lies in the region of attraction is to start at the origin. In the particular case of the quadrotor, it should start tracking a desired reference trajectory with an error of zero, because if the reference trajectory satisfies the state and control constraints, the origin is within $\mathcal{X}_N$.

**Algorithm 2** Projection of the polytope $\mathbb{A}_j$ onto $\mathbb{R}^n$ with $m > 1$ (from [51])

**procedure** POLYTOPEPROJECTION($\boldsymbol{G}, \boldsymbol{H}, \boldsymbol{\psi}$)
  Let $\boldsymbol{H}_m$ denote the $m^{\text{th}}$ column of $\boldsymbol{H}$,
  $\begin{bmatrix}\boldsymbol{G}_m & \boldsymbol{H}_m\end{bmatrix} \leftarrow \begin{bmatrix}\boldsymbol{G} & \boldsymbol{H}\end{bmatrix}$
  $k \leftarrow m - 1$
  **while** $k \geq 0$ **do**
    $\begin{bmatrix}\boldsymbol{P}_k & \boldsymbol{\gamma}_k\end{bmatrix} \leftarrow \text{polytopeProjection1}(\boldsymbol{G}_{k+1}, \boldsymbol{H}_{k+1}, \boldsymbol{\psi})$
    Let $\boldsymbol{H}_k$ be a vector and $\boldsymbol{G}_k$, a matrix such that
      $\begin{bmatrix}\boldsymbol{G}_k & \boldsymbol{H}_k\end{bmatrix} \leftarrow \boldsymbol{P}_k$
    $\boldsymbol{\psi} \leftarrow \boldsymbol{\gamma}_k$
    $k \leftarrow k - 1$
  **end while**
  **return** $\begin{bmatrix}\boldsymbol{P}_0 & \boldsymbol{\gamma}_0\end{bmatrix}$
**end procedure**

# Chapter 7

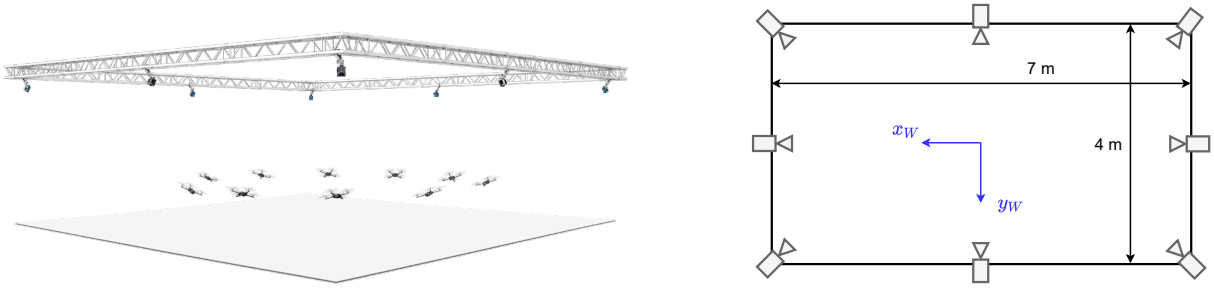
# Implementation, Results, and Discussion

This chapter is structured as follows:

- In order to test tracking controllers for quadrotors, equipped with the PX4 autopilot [14], in realistic software-in-the-loop (SITL) simulations and experimental trials, in Section 7.1, a testing software environment [13, 18] is briefly described. The considered test space to run experiments is also detailed in this section;
- In the following three sections, Section 7.2 (SITL Relay Planning Results), Section 7.3 (Experimental Relay Results), and Section 7.4 (MPC Planning Results), the proposed strategy for planning relay manoeuvres, presented in Section 4.3, and the model predictive controller, presented in Section 6.1, are validated by analysing the simulation (Sections 7.2 and 7.4) and experimental results (Section 7.3) of one manoeuvre:
  - **Roll relay**: The two vehicles vary their roll angle until the desired orientation $\boldsymbol{\lambda}_G$ is achieved, while simultaneously shortening the lateral distance between them.
- Finally, in Section 7.5, the results of an experiment in which two INTEL® Aero Ready to Fly quadrotors were commanded to hover, one below the other, are also presented to analyse the aerodynamic interactions between two vehicles and to motivate the need to perform aggressive relay manoeuvres.

## 7.1 Test Space and Off-board Control

The test space for running experiments was an indoor flying arena featuring an OptiTrack Motion Capture System [52], as depicted in Figure 7.1(a). The arena is about $2.5\,\mathrm{m}$ high and the MCS consists of eight cameras and a central computer which is responsible for processing the vision data acquired and




(a) Illustration of a flying arena featuring an MCS (from [52]).

(b) Arena dimensions and cameras positioning.

Figure 7.1: Flying arena used to conduct experimental trials.

(a) INTEL® Aero Ready to Fly.

(b) 3DR® Iris.

Figure 7.2: Multirotors considered to run SITL simulations and experimental tests.

providing high frequency measurements of position and orientation of multirotors flying inside the arena. The inertial frame is placed on the floor, at the centre of the arena, as illustrated in Figure 7.1(b).

The INTEL® Aero Ready to Fly quadrotor, represented in Figure 7.2(a), equipped with the PX4 autopilot was employed to run the experiments. These vehicles also include an on-board computer with an Ubuntu distribution installed, running the Robot Operating System (ROS) [15], which communicates with the autopilot via a built-in serial connection, using the MAVLink protocol [53]. A connection between an external computer, serving as a ground control station (GCS), and the on-board computer was established through a Wi-Fi link. Furthermore, the ROS package, MAVROS [16], was used to forward and interpret MAVLink messages. The bridge between the serial port and the Wi-Fi network is handled by a dedicated program developed by INTEL® named MAVLink Router [54].

In order to control an aircraft from an off-board computer and to perform manoeuvres such as position hold and landing resorting to the PX4 autopilot, reliable position, velocity, and orientation estimates are required. The autopilot receives position and orientation measurements from the MCS using the MAVROS package. From these orientation measurements, it is necessary to consider yaw angle values, as the on-board magnetometer cannot provide suitable readings indoors. An Extended Kalman Filter (EKF), included in the autopilot, fuses MCS measurements with Inertial Measurement Unit (IMU) data to obtain the required estimates. In the context of this dissertation, the problem of tuning an EKF is not addressed.

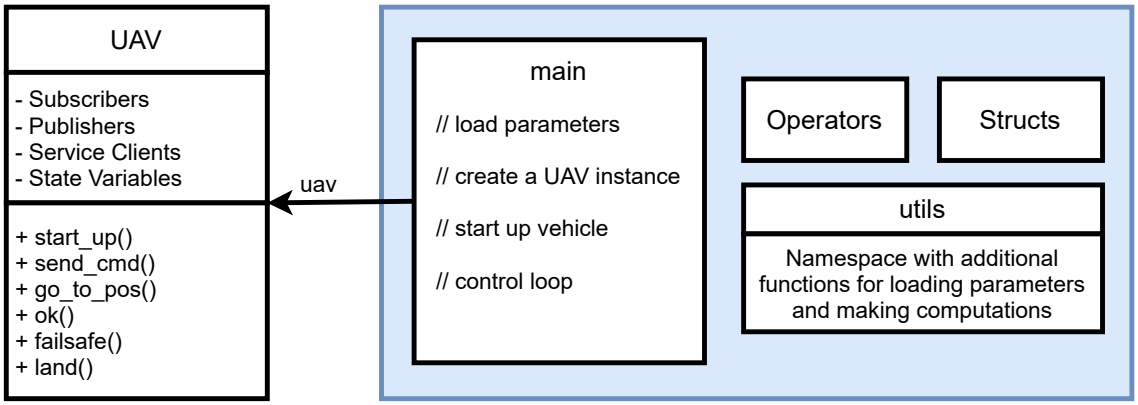


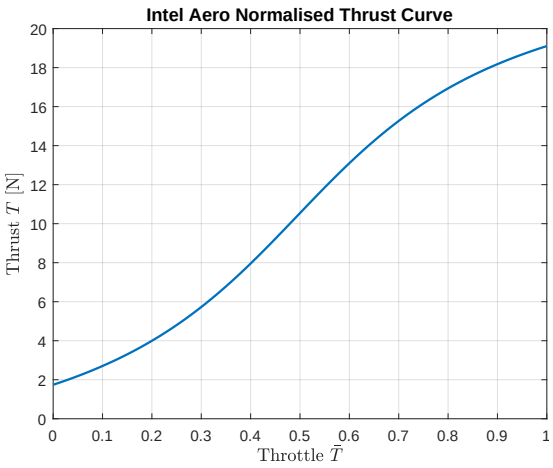

Figure 7.3: Simplified diagram of the drone control code architecture.

Moreover, a testing software environment was developed in C++ to efficiently implement tracking controllers for quadrotors, using the features of MAVROS, being controlled by a single GCS [13]. In Figure 7.3, a diagram is presented in order to represent the code structure. The class 'UAV' sets all publishers and subscribers necessary to publish commands to control the vehicle (for instance, to command the aircraft to move to a specific position waypoint and hover, to provide thrust and angular rate inputs to the motors, or to land) and to access the value of the state variables. Class member functions to handle user input commands were also implemented to provide fail-safe commands, such as hovering and landing, in emergency circumstances. Additional structures were created to organise code related to input commands bounding and normalisation. Despite the ROS package 'tf2' [55] being used to perform computations involving matrices, more operators were defined. Finally, the desired controller to be tested should be defined in the *main* function.

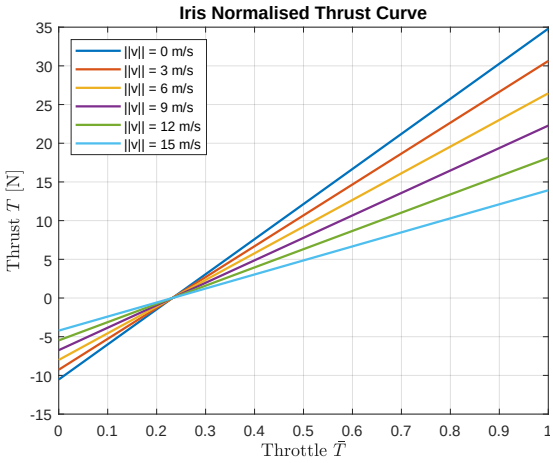
In order to control multiple vehicles simultaneously, multiple MAVROS instances and of the controller to be experimented are created from a ROS *launch* file. Having this in mind, as it was required to perform tests with two vehicles, a couple of simple ROS *services* were implemented to ensure both aircraft begin tracking the desired trajectory simultaneously. After taking off, each of the vehicles is commanded to reach a certain position waypoint before following the trajectory. Whenever any of the quadrotors reaches its own setpoint, it periodically ascertains whether its peer has similarly reached its own. Once a positive response is obtained, the vehicle which posed the question requests the other to start tracking the trajectory together.

As PX4 supports Software in the Loop (SITL) simulations, the developed code may also be used to test controllers in, for instance, the Gazebo Simulator [17] as a physical engine before flying in the arena, which is a great advantage. The multirotor considered in these simulations is the 3DR® Iris quadrotor, depicted in Figure 7.2(b). This vehicle is not identical to the INTEL® Aero Ready to Fly quadrotor since its mass, including the battery pack used, is $1.35\,\mathrm{kg}$ compared to the $1.52\,\mathrm{kg}$ of the latter and their thrust curves are different.

To control the vehicles in thrust, the desired throttle $\bar{T} \in [0, 1]$ must be input to the autopilot and, therefore, functions that relate thrust, $T$ to throttle, $\bar{T}$ are necessary. For the INTEL® Aero Ready to Fly quadrotor, other Institute for Systems and Robotics (ISR) researchers acquired experimental points




(a) INTEL® Aero Ready to Fly.




(b) 3DR® Iris.

Figure 7.4: Thrust vs throttle curves for the INTEL® Aero Ready to Fly and 3DR® Iris quadrotors.

$(\bar{T}, T)$ by measuring the static thrust generated, i.e., thrust generated when the airspeed is zero, by the four propellers, varying the throttle input given to the autopilot. The $\arctan(\cdot)$ function was selected to describe the evolution of thrust with throttle,

$$T = a \arctan\left(b\bar{T} + c\right) + d, \tag{7.1}$$

with $a = 8.84$, $b = 2.995$, $c = -1.478$, and $d = 10.37$. This curve is plotted in Figure 7.4(a). In what regards the 3DR® Iris quadrotor, a linear relationship between the two variables was considered,

$$f(\boldsymbol{v})T = k\bar{T} + h, \tag{7.2}$$

where $k = 45.34$ and $h = 1.52g - 0.561k$ ($g$ denotes the Earth's gravity). Since the rotor model in Gazebo appears to take into account the airspeed of the vehicle to compute the amount of thrust produced by a rotating propeller, the function $f : \mathbb{R}^3 \to \mathbb{R}$ incorporates the linear velocity of the vehicle into the model, while wind velocity is not considered, $f(\boldsymbol{v}) = 1/(1 - \|\boldsymbol{v}\|/25)$. As observed in Figure 7.4(b), for $\bar{T} \leq 0.23$, the model assumes negative thrust is generated by the rotors, however, it was verified that, through extensive simulations in Gazebo, for intermediate throttle values, the corresponding thrust magnitude is correctly computed from the proposed linear model.

Comparing the two static thrust curves, for maximum throttle, that is, $\bar{T} = 1$, the INTEL® Aero Ready to Fly is capable of producing around 19 N of thrust, whereas the 3DR® Iris, about 35 N. The thrust-to-weight ratio, the maximum thrust divided by the weight, of each vehicle is, respectively, $1.44$ and $2.35$. Although the inertia tensor of each is not known, this figure of merit suggests the 3DR® Iris is more manoeuvrable. For this reason, the same thrust bounds were imposed for both multirotors. Some parameters of the simulated aircraft, the 3DR® Iris, namely the rotor drag coefficient and the inner-loop attitude controller gains, were additionally tweaked so as to achieve a similar performance to the real vehicle.

In order to control the vehicles in torque, it would similarly be necessary to determine the normalised

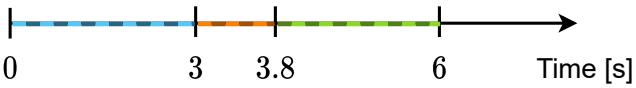



Figure 7.5: Time axis representing the duration of the every phase of the SITL roll relay.

curve $\bar{\boldsymbol{\tau}} \mapsto \boldsymbol{\tau}$. According to (5.1), it would be required to make the rotors rotate in various speeds and measure the torques generated along the three axis of the body frame $\mathcal{B}$. As the rotating propellers can produce negative moments, $\bar{\tau}_x, \bar{\tau}_y, \bar{\tau}_z \in [-1, 1]$. Due to the intricacy of conducting an experiment to acquire the necessary data to, then, determine the normalised curves, it was decided that the inner-loop of the autopilot would handle the computation of the desired torques to stabilise the attitude of the vehicle. Additionally, it is sensible to rely on this autopilot loop as it operates in hard real-time and benefits from running in a micro-controller at very high frequency.

Finally, to make the simulated environment in Gazebo closer to the flying arena, a simple program to generate MCS synthetic data was created as the original PX4 SITL setup relies on a Global Navigation Satellite System (GNSS) plug-in from the Gazebo simulator to estimate the state of the vehicle. This program merely consists in adding white Gaussian noise to the pose of the vehicle obtained from Gazebo and forwarding it to the autopilot [18].

## 7.2 SITL Relay Planning Results

The relay manoeuvre planning strategy, presented in Section 4.3, is assessed by analysing the results of two vehicles, coupled with the tracking controller of Section 5.3, performing the *Roll relay* manoeuvre. These results were obtained from Software in the Loop (SITL) simulations conducted in the Gazebo simulator [17], involving the PX4 autopilot [14], where the vehicles were controlled from an off-board computer, as explained in Section 7.1. The tracking controller introduced in Section 5.3 was implemented using the features of the software architecture developed (*vide* Section 7.1).

The simulation results presented in this section were obtained from a computer with the operating system Ubuntu 16.04 LTS (Xenial Release) installed, equipped with an Intel Core i7-8750H CPU @2.20GHz and 16,00GB of RAM. As in experimental trials, the desired throttle and orientation are provided as inputs to the autopilot. The off-board controller gains were set to: $\boldsymbol{K}_p = 4.5\boldsymbol{I}$, $\boldsymbol{K}_v = 4.3\boldsymbol{I}$, and $\boldsymbol{K}_i = 0.5\boldsymbol{I}$.

The time duration of each phase of the manoeuvre is depicted in Figure 7.5. Throughout this section, the colour scheme adopted in Section 4.3 is considered: *Phase 1* is associated with the colour blue; *Phase 2*, with the colour orange; and *Phase 3*, with the colour green.

In Figure 7.6, a schematic representation of the established waypoints for the roll manoeuvre is provided. Let $\boldsymbol{p}_{Sj}$, $\boldsymbol{p}_{Gj}$, $\boldsymbol{p}_{Fj} \in \mathbb{R}^3$ denote the start, goal, and final positions of vehicle $j$, respectively. The trajectory generated for *Phase 2* is planned taking into account that one vehicle should have a roll manoeuvre of $60°$ and the other of $-60°$. The velocity of each quadcopter at the goal point $\boldsymbol{v}_{Gj}$ was carefully chosen to ensure proper trajectory tracking. When discussing the results, this point is further developed. Both vehicles start with linear velocity and acceleration equal to zero and finish at hover.

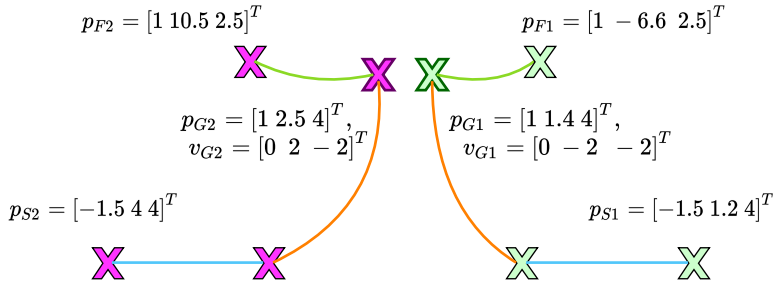
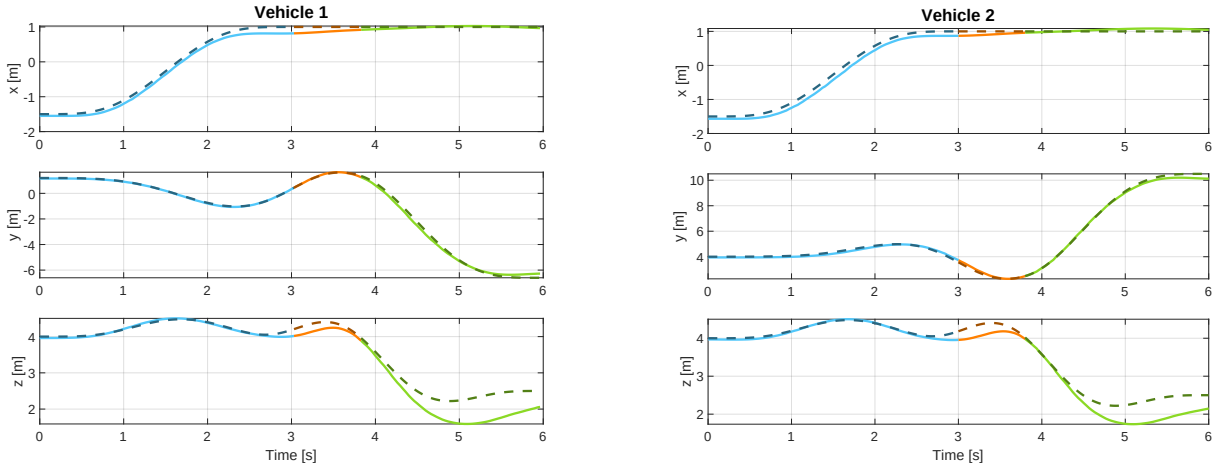



Figure 7.6: Schematic representation of the waypoints established for the SITL roll relay.




(a) Position response of the first vehicle

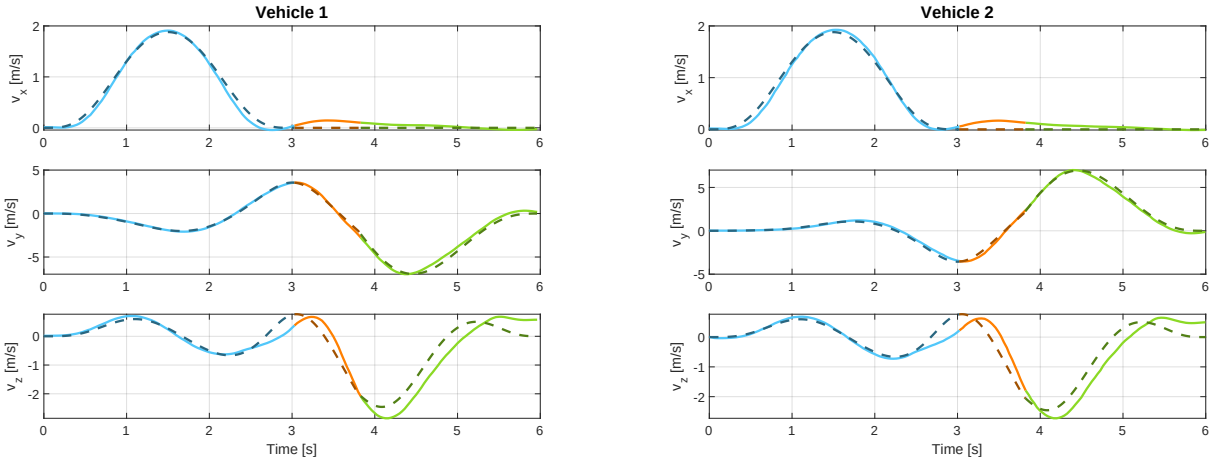
(b) Position response of the second vehicle

Figure 7.7: Position evolution of both vehicles during the SITL roll relay.

Figures 7.7 and 7.8 present the position and velocity plots of both vehicles, respectively. In these plots, the continuous lines in lighter colours correspond to the observed trajectories; the dashed ones in darker colours, to the references. Figure 7.9 depicts the orientation input, reference, and its actual evolution over time. As the quadrotor dynamics is *differentially flat*, from the reference trajectory in the space of *flat outputs*, the reference trajectories for all state variables and inputs can be algebraically obtained. The inputs are represented using dash-dot lines in bright shades. The throttle input and reference for both vehicles is plotted in Figure 7.11. Finally, in Figure 7.10, the plots of the observed and reference angular rates are depicted.

Firstly, analysing Figures 7.7(a) and 7.7(b), it is visible both vehicles struggle in tracking the position reference trajectories for *Phase 2* and *Phase 3*, in particular, the tracking error in the $z$ axis is noticeable.

Even though both aircraft reach the desired goal position $\boldsymbol{p}_{Gj}$ with linear velocity close to $\boldsymbol{v}_{Gj}$, vehicle 1 missed the launch position $\boldsymbol{p}_{L1}$ by $18\,\text{cm}$ in $x$ and $19\,\text{cm}$ in $z$; vehicle 2 missed $\boldsymbol{p}_{L2}$ by $13\,\text{cm}$ in $x$ and $22\,\text{cm}$ in $z$. Examining Figures 7.8(a) and 7.8(b), it is observed that vehicle 1 failed in achieving the desired launch velocity $\boldsymbol{v}_{L1}$ by an error of $0.4\,\text{m}\,\text{s}^{-1}$ in the $z$ axis and, similarly, vehicle 2 missed $\boldsymbol{v}_{L2}$ by $0.5\,\text{m}\,\text{s}^{-1}$ also in $z$, despite the linear velocity reference demanding both vehicles to reach a fairly low velocity of $0.76\,\text{m}\,\text{s}^{-1}$ in $z$, at the launch point. However, in the $y$ axis, the absolute value of linear velocity reference, between $t = 2.3\,\text{s}$ and $t = 3\,\text{s}$, increases considerably. Between these two time instants, the velocity reference in this component varies, roughly in linear fashion, for vehicle 1 between $-0.15\,\text{m}\,\text{s}^{-1}$




(a) Velocity response of the first vehicle

(b) Velocity response of the second vehicle

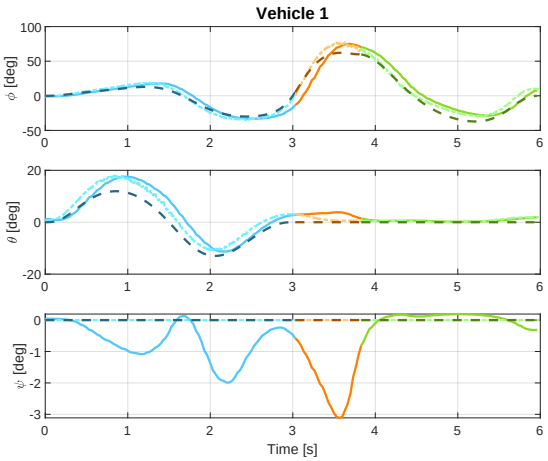
Figure 7.8: Velocity evolution of both vehicles during the SITL roll relay.

and $3.5\,\mathrm{m\,s^{-1}}$; for vehicle 2 between $-0.18\,\mathrm{m\,s^{-1}}$ and $-3.6\,\mathrm{m\,s^{-1}}$. For this reason, in the considered time interval, the motors reach the maximum thrust that can be produced by the rotating propellers, as evinced in Figures 7.11(a) and 7.11(b). Recall from Section 7.1 that the motors saturate before achieving the maximum possible throttle, $\bar{T} = 1$, since the thrust limits of the INTEL® Aero Ready to Fly quadrotor were considered to run this SITL simulation. Furthermore, the saturated thrust input corresponds to a branch of an hyperbola in throttle, as the thrust-vs-throttle curve employed (7.2) takes into account that for a given value of throttle, thrust and the airspeed absolute value are negatively proportional.

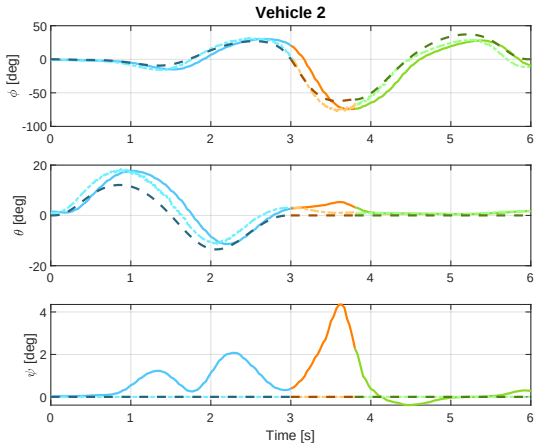
The tracking error observed in the $z$ component of the position and linear velocity references for *Phase 3* of the relay manoeuvre is again explained by the necessary linear acceleration the vehicles must achieve to ensure an accurate tracking of these references. From Figures 7.11(a) and 7.11(b), one verifies that, between $t = 3.81\,\mathrm{s}$ and $t = 5.33\,\mathrm{s}$, the rotors of both vehicles were in saturation. At the end of the manoeuvre, vehicle 1 was $37\,\mathrm{cm}$ off the desired final position in $z$ and vehicle 2, $44\,\mathrm{cm}$. Although the quadrotors finished following the reference trajectories with linear velocity in $x$ and $y$ close to zero, at the final time instant, vehicle 1 was moving up with $0.50\,\mathrm{m\,s^{-1}}$ in $z$ and vehicle 2, with $0.57\,\mathrm{m\,s^{-1}}$.

In what concerns following the orientation input, during all phases of the manoeuvre, the tracking performance is good (check Figures 7.9(a) and 7.9(b)), however, none of the vehicles achieved a roll angle equal to zero, $\phi_{Lj} = 0$, at the launch point. At this point, vehicle 1 missed the desired orientation by $-13°$ in roll and vehicle 2, by $20°$. At the goal point, vehicle 1 reached a roll angle of $\phi_{G1} = 70.5°$; vehicle 2, $\phi_{G2} = -73.8°$. Despite the quadrotors not achieving the established orientation at the goal point (vehicle 1 should roll $60°$ and vehicle 2, $-60°$), one observes that the roll and pitch inputs to the autopilot, during *Phase 2*, converge to the orientation reference trajectory. The reference trajectories for this manoeuvre were planned taking into account that both vehicles should finish at hover, however, vehicle 1 exhibited roll and pitch angles $(\phi_{F1}, \theta_{F1}) = (7.9°, 1.6°)$ and vehicle 2, $(\phi_{F2}, \theta_{F2}) = (-8.3°, 1.8°)$. This observed error in the final attitude of the vehicles is not major problem though, since the autopilot can drive the roll and pitch angles of the aircraft to zero by choosing a suitable flight mode.

The angular velocity plots are in Figures 7.10(a) and 7.10(b). It is seen that none of the vehicles




(a) Orientation response of the first vehicle




(b) Orientation response of the second vehicle

Figure 7.9: Orientation evolution of both vehicles during the SITL roll relay.

exceeds the maximum angular rates they can achieve. Both quadrotors reach a maximum roll rate of, in absolute value, $|p_j| = 3.6\,\mathrm{rad\,s^{-1}}$. Furthermore, during *Phase 2*, they progressively decrease their roll rate until the goal point is reached; throughout *Phase 3*, one can observe that both aircraft vary their roll rate very gently, which is positive. At the end of this phase, the roll rate of vehicle 1 was $p_{F1} = 0.74\,\mathrm{rad\,s^{-1}}$ and, of vehicle 2, $p_{F2} = -0.57\,\mathrm{rad\,s^{-1}}$, while the angular velocity reference trajectory demanded both vehicle to end with roll and pitch rates equal to zero. Finally, it is also worth highlighting that, even though angular rate inputs are not provided to the autopilot, the angular rates evolution of the aircraft is very similar to the reference trajectory.

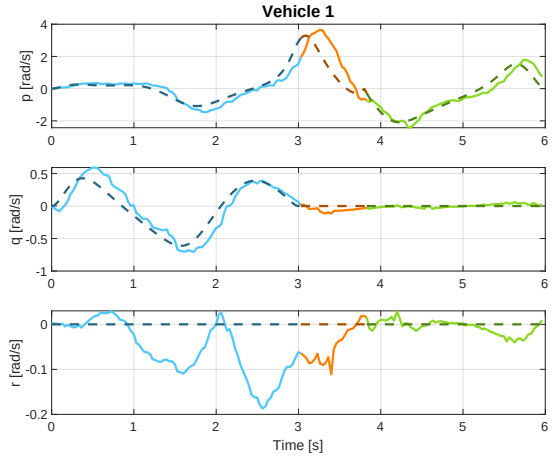
Lastly, in Figure 7.12, using the squared distance function between two polyhedrons (4.9), the distance between the vehicles along the manoeuvre was computed. The dimensions of the considered parallelipiped were $50 \times 50 \times 15$ cm, so that an INTEL® Aero Ready to Fly quadrotor could fit inside it. Note that the pose data of both vehicles was synchronised and interpolated to compute this distance. The reference polyhedron distance was obtained from position, linear acceleration, and yaw angle trajectories. In spite of the observed tracking errors in the previous plots, the distance between the two quadrotors evolves in accordance with the reference. The minimum distance between the two is $30\,\mathrm{cm}$, at $t = 3.5\,\mathrm{s}$. At this time instant, it is verified that both vehicles are in a suitable relative orientation to exchange a parcel.

A video of the manoeuvre is available at [56].

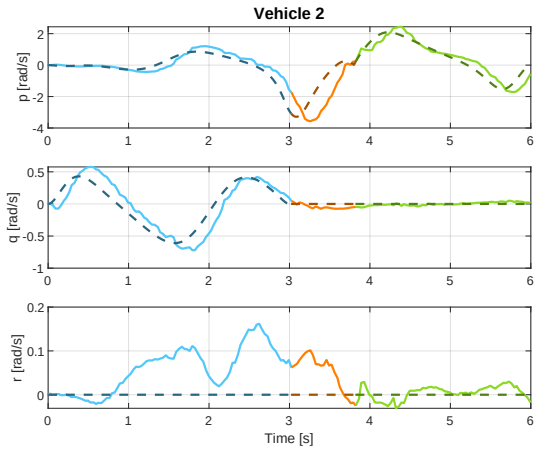
## Remarks

The results of this SITL simulation and other experiments that were conducted, but not as thoroughly presented and analysed in this document due to space limitations, allow to draw some significant conclusions about the presented relay manoeuvre strategy introduced in Section 4.3.

The underlying problem across all experiments is thrust and angular rates limitations. In the results presented in this section, the available thrust hinders in the accurate following of the reference trajectory. Achieving the necessary launch velocity $\boldsymbol{v}_{Lj}$ to reach the goal state $G_j$ might not be possible if $\boldsymbol{v}_{Gj}$ is

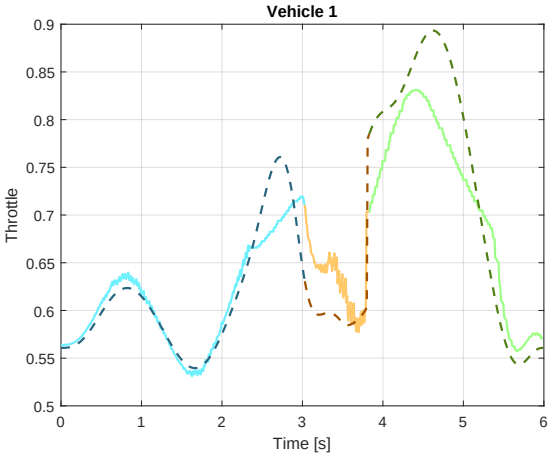



(a) Angular velocity response of the first vehicle

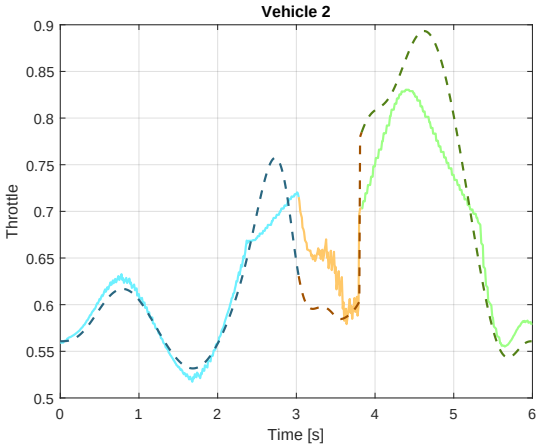



(b) Angular velocity response of the second vehicle

Figure 7.10: Angular velocity evolution of both vehicles during the SITL roll relay.




(a) Throttle response of the first vehicle




(b) Throttle response of the second vehicle

Figure 7.11: Throttle evolution of both vehicles during the SITL roll relay.

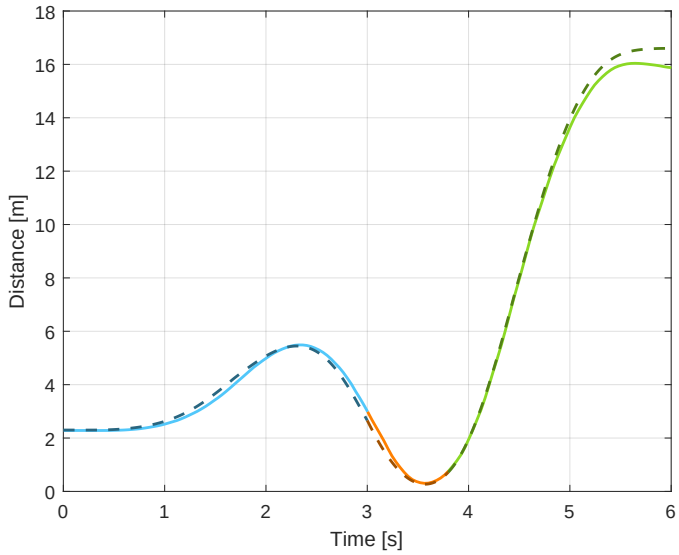



Figure 7.12: Distance between the quadcopters during the SITL roll relay, computed using the square root of (4.9).

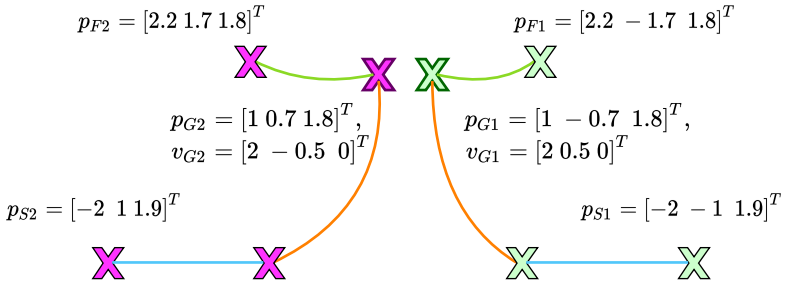



Figure 7.13: Schematic representation of the waypoints established for the Experimental roll relay.

not adequately selected and, while integrating the quadrotor dynamics backwards in time, not pondering the orientation reference the vehicle should track.

In what regards the choice of $\boldsymbol{v}_{Gj}$, in the experiment above, for instance, it was established both quadrotors should achieve $-2\,\mathrm{m\,s^{-1}}$ in the $z$ axis. Although it is not ideal the vehicles having literally to descend to perform the parcel relay, it is necessary to ensure a low launch velocity in the $z$ component. However, imposing the velocity in the $z$ axis to be negative at $G_j$ is, in a sense, a double-edged sword since the vehicles request excessive thrust to avoid falling down and to finish the manoeuvre at hover.

As mentioned before, the orientation reference selected for the backwards integration is also extremely important. Given that this reference is a step in roll and pitch angles (check (4.14)), the choice of the parameter $t_b$, which defines the time window in which the vehicles are in a suitable orientation to perform the package transfer, also affects the value of the launch velocity. Choosing a desired orientation at $G_j$ that deviates significantly from the hover condition (that is, high roll and/or pitch angles) and setting $t_b$ close to zero will demand that the aircraft achieve a large momentum. One should also note that selecting a suitable value of thrust $T$ to be kept constant during the backwards integration is relevant, as the higher this constant value is, the higher the launch velocity will be required. As a rule of thumb, setting this constant thrust value as enough to hover is sufficient.

Finally, the initial position of each of the quadrotors $\boldsymbol{p}_{Sj}$ should also be carefully chosen to ensure large values of linear acceleration are not required to reach the necessary launch velocity successfully.

## 7.3 Experimental Relay

In this section, the relay manoeuvre planning strategy, presented in Section 4.3, is evaluated, similarly to the previous section, by examining the results of two vehicles coupled with the tracking controller of Section 5.3, performing the *Roll relay* manoeuvre. However, the results disclosed in this section were obtained from an experimental test, considering the test space described in Section 7.1. The specifications of the off-board computer to control the vehicles are identical to which was employed to run SITL simulations. The off-board controller gains were set to: $\boldsymbol{K}_p = 4.5\boldsymbol{I}$, $\boldsymbol{K}_v = 4.3\boldsymbol{I}$, and $\boldsymbol{K}_i = 0.5\boldsymbol{I}$.

The time duration of each phase of the manoeuvre matches what is depicted in Figure 7.5, but the final time instant of the manoeuvre is set to $t = 5.2\,\mathrm{s}$. In Figure 7.13, a schematic representation of the established waypoints for the roll manoeuvre is provided. The trajectory generated for *Phase 2* is planned taking into account that one vehicle should have a roll manoeuvre of $15^\circ$ and the other of $-15^\circ$.

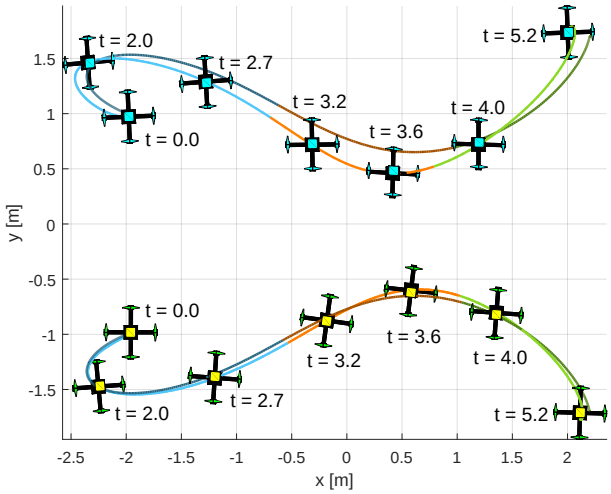



Figure 7.14: Composite illustration of the vehicles tracking the reference of the Experimental roll relay ($xy$ plane).

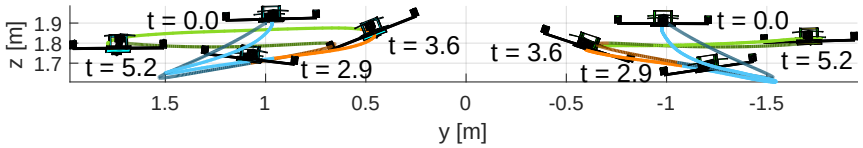


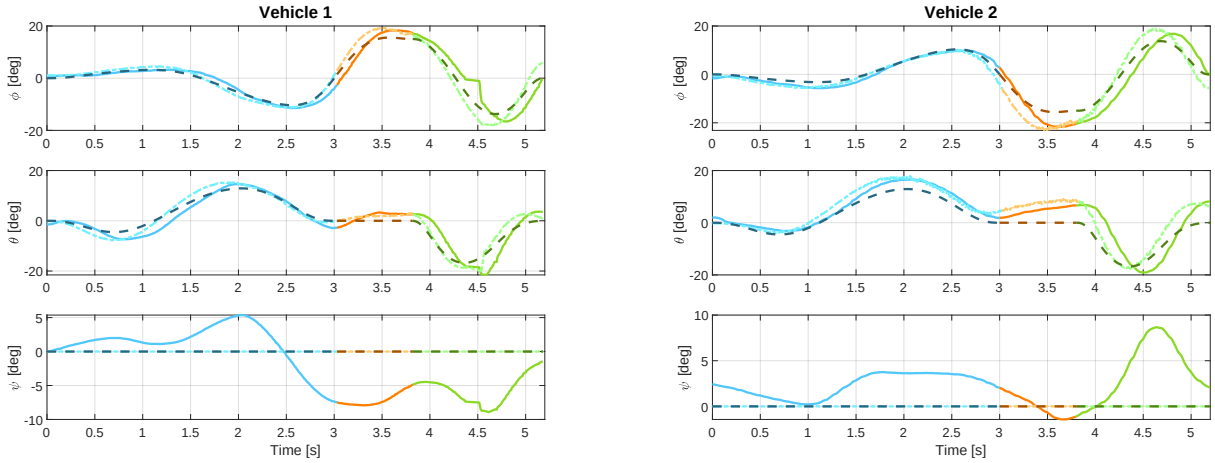

Figure 7.15: Composite illustration of the vehicles tracking the reference of the Experimental roll relay ($yz$ plane).

It was not possible to request a greater roll angle variation due to the dimensions the flying arena.

In this section, the same colour scheme of the previous section is adopted. In Figures 7.14 and 7.15, the position reference trajectories of both vehicles together with the described trajectories during this manoeuvre are presented. In first figure, a top view of the manoeuvre is depicted ($xy$ plane perspective), while, in the second, a frontal view ($yz$ plane perspective). Along the described trajectories, which are depicted in lighter shades, drawings of the vehicles are included at specific time instants, taking into account their actual orientation during the experiment.

Although the aircraft start tracking the reference trajectory simultaneously, as it can be verified by watching a video captured during the experiment [57], the quadrotors are misaligned in $x$ and $z$ during *Phase 2* of the relay manoeuvre. Furthermore, one observes that vehicle 1 (the yellow quadcopter) tracked its reference trajectory more accurately than vehicle 2 (the blue quadcopter). At a first glance, one might suspect that the aerodynamic interactions between the aircraft are the underlying cause of the observed misalignment or that the controller gains of vehicle 2 require better tuning. A thorough analysis of various logs with flight data evinced, nevertheless, that vehicle 1 consistently broadcasts more frequently information on its pose than vehicle 2 does. This problem demands investigating further the internal configurations of the latter vehicle.

Figure 7.16 depicts the orientation input, reference, and its actual evolution over time. The orientation input, during all phases of the manoeuvre, is followed extremely satisfactorily (check Figures 7.16(a) and 7.16(b)). Despite the quadrotors not achieving the established orientation at the goal point (vehicle 1 should roll $15^\circ$ and vehicle 2, $-15^\circ$), one observes that, during *Phase 2*, the actual roll angle of




(a) Orientation response of the first vehicle

(b) Orientation response of the second vehicle

Figure 7.16: Orientation evolution of both vehicles during the Experimental roll relay.

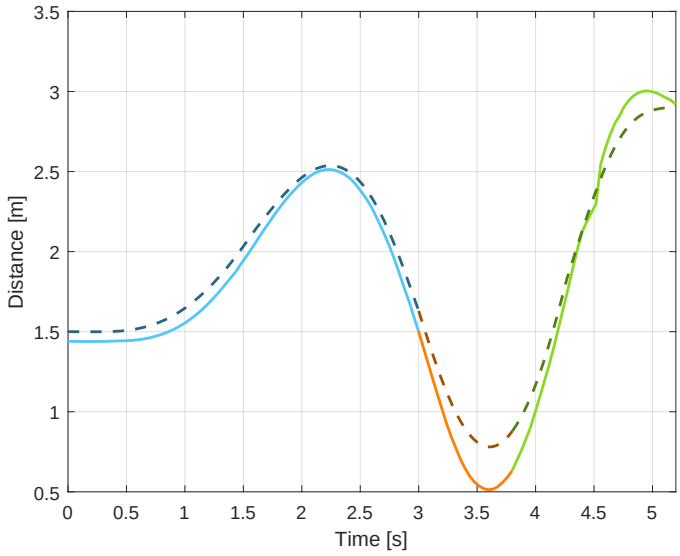



Figure 7.17: Distance between the quadcopters during the Experimental roll relay, computed using the square root of (4.9).

the vehicles does not deviate significantly from the orientation reference trajectory, however, it is also noticeable that the roll and pitch angles error and, in particular, the pitch error relative to the orientation reference of vehicle 2 is larger of vehicle 1, for the position tracking error observed.

Using the squared distance function between two polyhedrons, introduced in (4.9), the distance between the two aircraft can be computed taking into account their dimensions and orientation is plotted in Figure 7.17.

## 7.4 MPC Relay Results

The MPC regulator, presented in Section 6.1, is assessed by analysing the results of the vehicles performing the *Roll relay* manoeuvre. While this regulator provides thrust inputs $T$ to control the translation of the rotorcraft, a geometric attitude controller on $\mathrm{SO}(3) \times \mathbb{R}^3$, given a desired direction of thrust $\boldsymbol{z}_B$ and

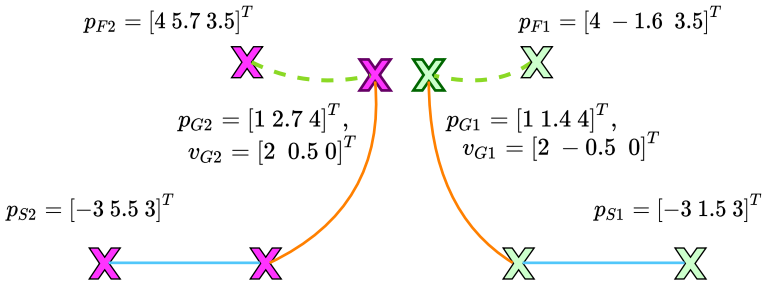
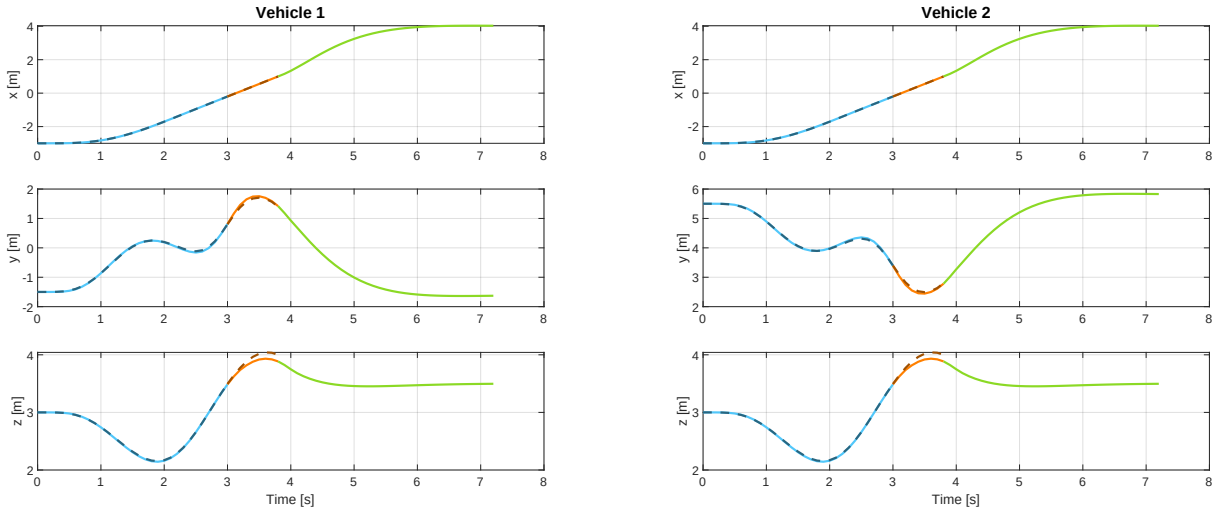



Figure 7.18: Schematic representation of the waypoints established for the MPC roll relay.




(a) Position response of the first vehicle

(b) Position response of the second vehicle

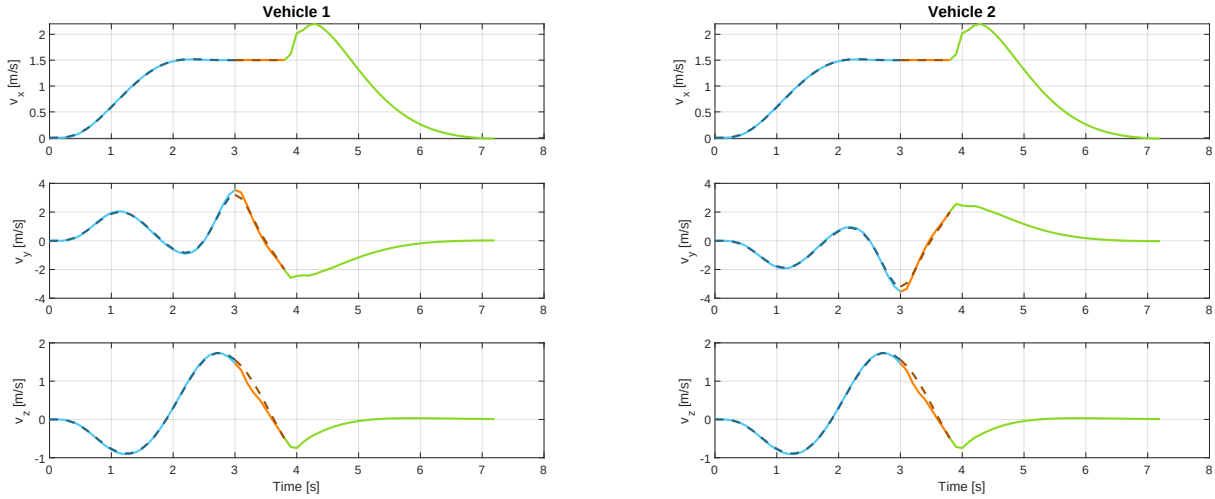
Figure 7.19: Position evolution of both vehicles during the MPC roll relay.

an angular velocity $\boldsymbol{\omega}$ (computed by the regulator and given as virtual inputs to this controller), and a yaw angle $\psi$, outputs a vector of input body torques $\boldsymbol{\tau}$ to stabilise the attitude dynamics of the aircraft.

The reference trajectories were obtained using the methods introduced in Chapter 4. The generated reference jerk trajectory was provided as control input reference $\boldsymbol{u}_{ref}(\cdot)$ to the MPC regulator to improve trajectory tracking. During *Phase 1* and *Phase 2* of the manoeuvre, both vehicles follow a pre-planned trajectory, while in *Phase 3*, each of the rotorcraft is commanded to reach a position setpoint and hover.

In Figure 7.18, a schematic representation of the established waypoints for this manoeuvre is provided. Let $\boldsymbol{p}_{Sj}$, $\boldsymbol{p}_{Gj}$, $\boldsymbol{p}_{Fj} \in \mathbb{R}^3$ denote the start, goal, and final positions of vehicle $j$, respectively. The trajectory generated for *Phase 2* is planned taking into account that one vehicle should have a roll manoeuvre of $45°$ and the other of $-45°$. A different reference trajectory from the one considered for the SITL experiment, presented in Section 7.2, was selected to ensure closed-loop stability. The time duration of *Phase 1* and *Phase 2* matches what is presented in the time axis of Figure 7.5, on the other hand, a fixed time duration is not set for *Phase 3*. For this last phase, the MPC controller is responsible for planning state and control inputs trajectories online, without a baseline time-varying trajectory.

A sampling time of $T_s = 0.1\,\mathrm{s}$ and a control horizon of $N = 6$ was considered to run this simulation. The reasoning behind choosing these values is discussed soon. The vehicle considered for obtaining the results presented in this section was an INTEL® Aero Ready to Fly quadrotor.




(a) Velocity response of the first vehicle

(b) Velocity response of the second vehicle

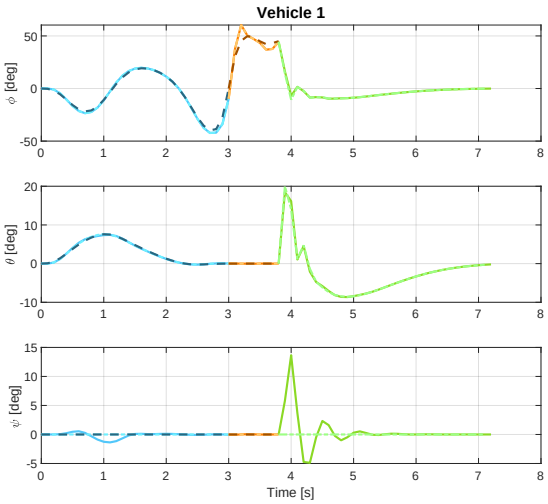
Figure 7.20: Velocity evolution of both vehicles during the MPC roll relay.

The simulation results presented in this section from a computer with the operating system Windows 10 installed running MATLAB, equipped with an Intel Core i7-6700HQ CPU @2.60GHz, and 16,00GB of RAM, where the OCPs were solved using the QP solver provided by *CPLEX* [49]. The dynamics of the quadrotor system, expressed by the equations (5.2), (5.3), (5.4), and (5.5), were propagated using *CasADi* [58], which is a general-purpose toolbox for numerical optimisation, algorithmic differentiation, and for solving optimal control problems. It interfaces with various solvers, namely *CVODES* (from the SUNDIALS suite [59]) which is dedicated to solve systems of ordinary differential equations (ODEs).

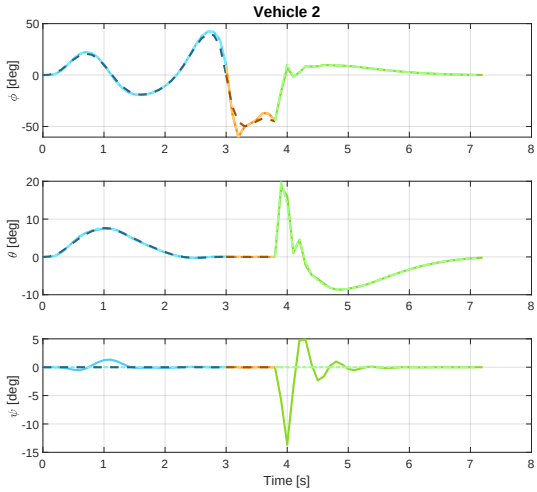
Figures 7.19 and 7.20 present the position and velocity plots of both vehicles, respectively. Figure 7.21 depicts the desired orientation computed by the attitude controller, the orientation reference trajectory, and its actual evolution over time. The throttle input and reference trajectory for both vehicles is plotted in Figure 7.22. In Figure 7.23, the plots of the observed and reference angular rates are depicted. Similarly to the previous section, in the following plots, the continuous lines in lighter colours correspond to the actual trajectories of the vehicles; the dashed lines in darker colours, to the reference trajectories. The reference trajectories for each state variable and input were obtained from the *flat outputs*. The inputs variables are represented using dash-dot lines in bright shades. The colour scheme introduced in Section 4.3 is again considered.

In what regards the tracking of the position references, from the plots in Figures 7.19(a) and 7.19(b), one observes that these references were followed adequately, however, during *Phase 2*, the vehicles were not able to fully drive the tracking error in the $z$ axis to zero. In the time interval between $t = 3.5\,\mathrm{s}$ and $t = 3.8\,\mathrm{s}$, both aircraft were off the desired altitude by roughly $10\,\mathrm{cm}$. The plots of linear velocity in Figures 7.20(a) and 7.20(b) reveal that the vehicles were able to track the reference trajectories, however, both missed the desired launch velocity $\boldsymbol{v}_{Lj}$ by around $0.3\,\mathrm{m\,s^{-1}}$ in the $y$ axis.

The orientation and angular velocity plots, in Figures 7.21(a) and 7.21(b) and Figures 7.23(a) and 7.23(b), reveal some issues which explain the observed altitude tracking error in the position plots of *Phase 2*. During this phase, although the roll angle $\phi(\cdot)$ of the vehicles follow precisely the desired




(a) Orientation response of the first vehicle




(b) Orientation response of the second vehicle

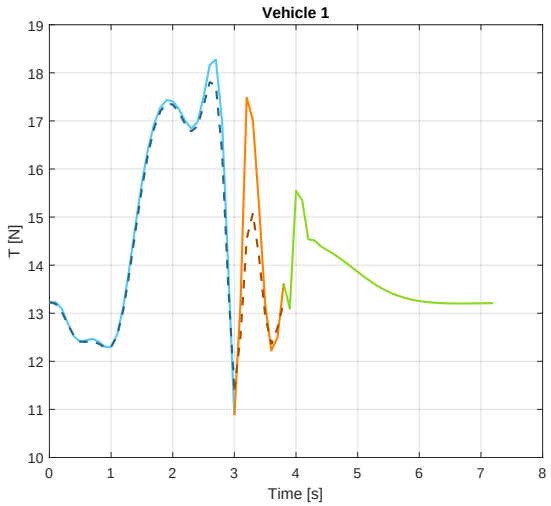
Figure 7.21: Orientation evolution of both vehicles during the MPC roll relay.

orientation computed by the attitude controller (note that the actual roll is depicted using a continuous orange line, while the desired orientation is depicted using a dash-dot line in bright orange; the pre-planned reference trajectory is represented by a dashed line in dark orange), some overshoot is noticeable.

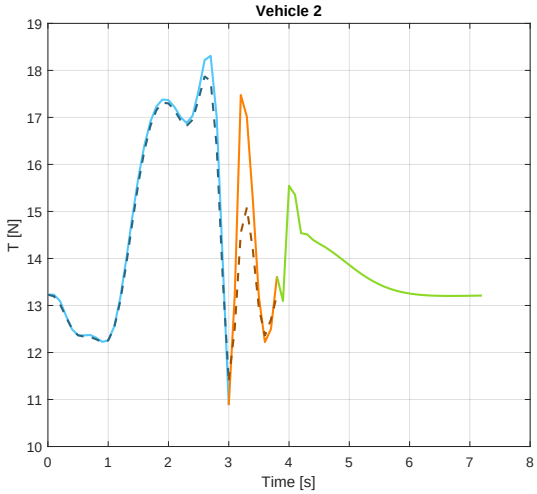
Looking over the angular velocity plots, right at the beginning of *Phase 2*, a spike is observed in the roll rate $p(\cdot)$ of both vehicles. At $t = 3\,\text{s}$, the pre-planned reference trajectory demands the quadcopters achieve $|p_j| = 6\,\text{rad}\,\text{s}^{-1}$, while the MPC regulator requests a desired roll rate of $|p_{j,des}| = 8.5\,\text{rad}\,\text{s}^{-1}$. As a result, at the next time instant, $t = 3.1\,\text{s}$, the rotorcraft reach $|p_j| = 10\,\text{rad}\,\text{s}^{-1}$, which is clearly excessive. This is a consequence of the control constraints not taking into account that the value of pitch and roll rates depend on the value of thrust (check (6.11) and (6.12)). In any case, in subsequent time instances, the roll rate, $p$ converges to zero and the roll angle, $\phi$ to the desired reference.

This problem associated with the tracking of attitude references also reflects in the thrust evolution along *Phase 2*. As evinced in Figures 7.22(a) and 7.22(b), the actual thrust exceeds significantly the reference trajectory. The pre-planned reference trajectory, at $t = 3.2\,\text{s}$ demands the quadcopters achieve a thrust of $T = 14.6\,\text{N}$, however the aircraft achieve $T = 17.5\,\text{N}$. Despite the excessive thrust achieved by the quadrotors at the beginning of *Phase 2*, once the attitude, i.e., orientation and angular velocity, converge to the desired reference, the thrust evolution matches the pre-planned trajectory.

Using the squared distance function between two polyhedrons, introduced in (4.9), the distance between the two aircraft can be computed taking into account their dimensions and orientation. The plot in Figure 7.24 reveals that the vehicles never collide during the manoeuvre and, during *Phase 2*, get close enough to one another to exchange a parcel. In this figure, the dashed line corresponds to the distance between polyhedrons computed using the pre-planned trajectory. At $t = 3.5\,\text{s}$, the vehicles reach a minimum distance of $22\,\text{cm}$. Combining this plot with the data presented in the plots of the orientation evolution (*vide* Figures 7.21(a) and 7.21(b)), there is a time window around this time instant in which the two vehicles are in a suitable relative orientation to perform the package relay. This further confirmed by

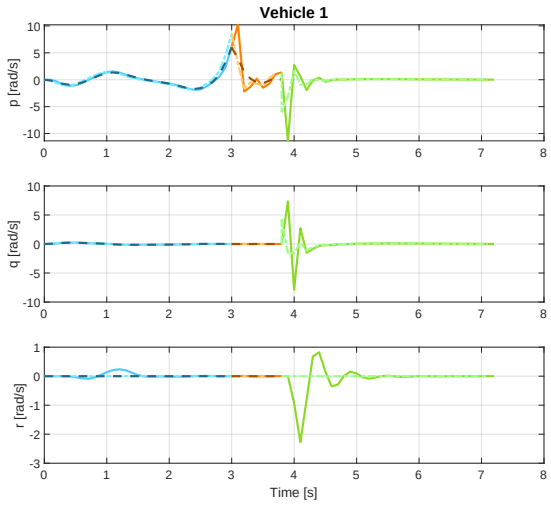



(a) Thrust response of the first vehicle

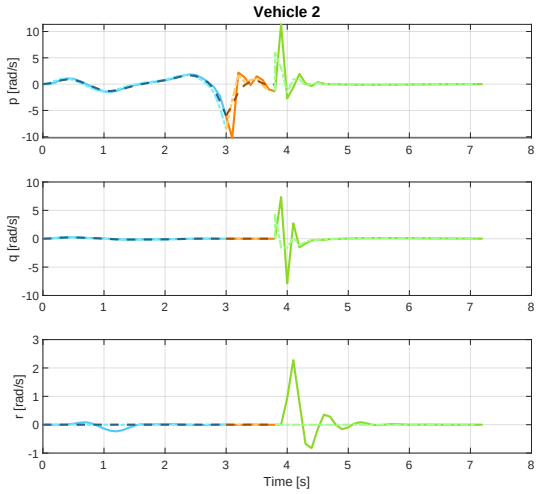



(b) Thrust response of the second vehicle

Figure 7.22: Thrust evolution of both vehicles during the MPC roll relay.




(a) Angular velocity response of the first vehicle




(b) Angular velocity response of the second vehicle

Figure 7.23: Angular velocity evolution of both vehicles during the MPC roll relay.

examining Figure 7.25 which presents the the position reference trajectories of both vehicles together with the described trajectories during *Phase 2*. Note that the quadrotors are not drawn up to scale.

Briefly going through the plots of *Phase 3* of the manoeuvre, both position and velocity very gently converge to the desired setpoint for each vehicle (vehicle 1 is driven to $\boldsymbol{p}_{F1}$ and vehicle 2, to $\boldsymbol{p}_{F2}$; both aircraft finish the manoeuvre with a linear velocity equal to zero) as demonstrated in Figures 7.19 and 7.20. Examining the attitude plots, in Figures 7.21 and 7.23, it is seen that the orientation varies somewhat sharply, particularly when comparing the orientation plots of *Phase 3* with the same plots of *Phase 1* and *Phase 2*; similarly, at the beginning of *Phase 3*, despite the desired roll and pitch rates being within reasonable bounds, $|p_{des}| \leq 5\,\mathrm{rad\,s^{-1}}$, $|q_{des}| \leq 5\,\mathrm{rad\,s^{-1}}$, the rotorcraft achieve higher roll and pitch rates than desirable. This suggests that the gains of the attitude control law should be better tuned. Finally, it is observed that the vehicle reaches the final setpoint at $t = 7\,\mathrm{s}$.

Finally, the time the *CPLEX* QP solver took to obtain a solution to the OCP to plan online trajectories

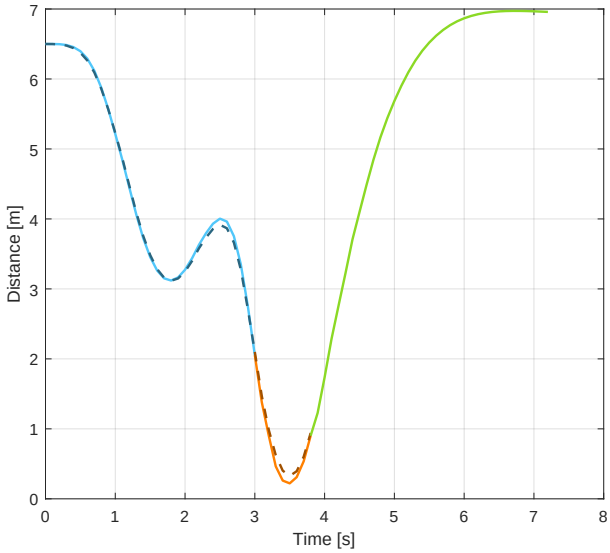



Figure 7.24: Distance between the quadcopters computed during the MPC roll relay, using the square root of (4.9).

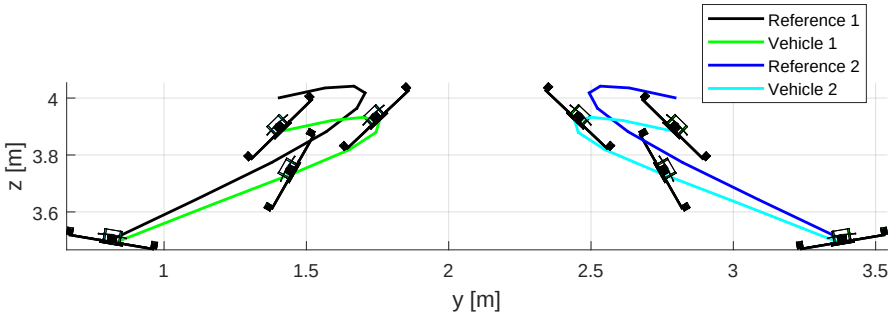
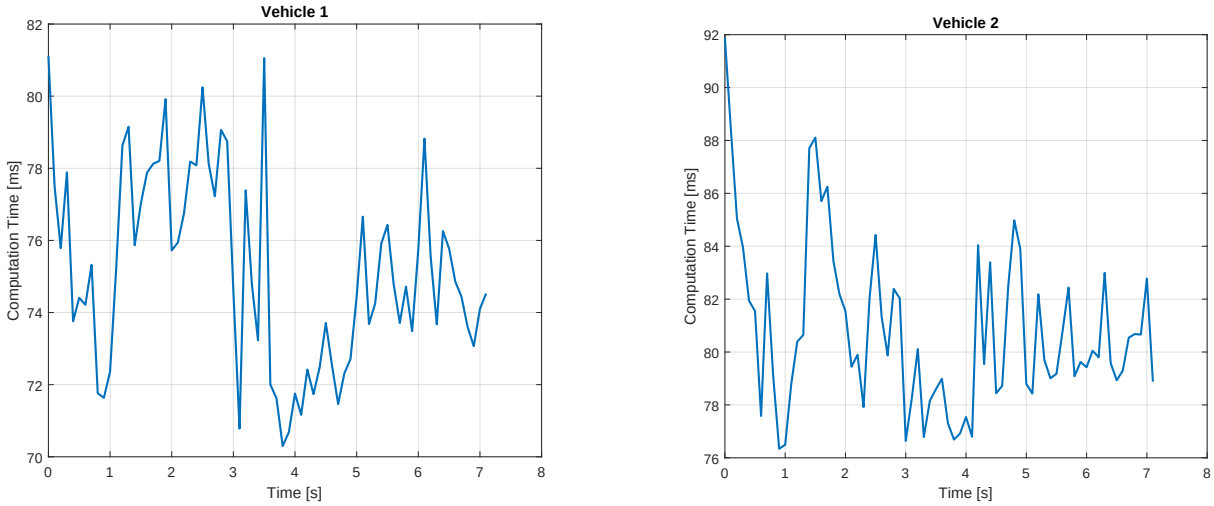


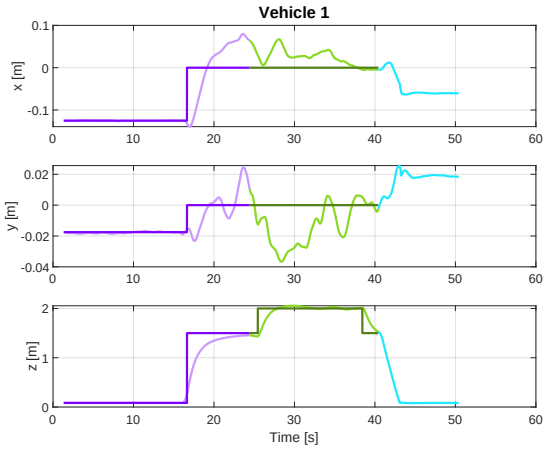

Figure 7.25: Composite illustration of the vehicles tracking the reference of *Phase 2* of the MPC roll relay.



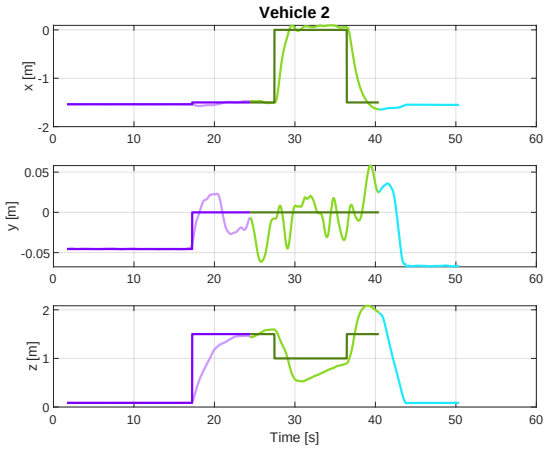

(a) Computation time of the first vehicle

(b) Computation time of the second vehicle

Figure 7.26: Computation time during the MPC roll relay.




(a) Position response of the first vehicle.




(b) Position response of the second vehicle.

Figure 7.27: Position evolution of both vehicles in close proximity while hovering one below the other.

for each of the vehicles (in other words, the time that the MPC regulator took to provide a solution) is depicted in Figures 7.26(a) and 7.26(b). It is observed the MPC regulator controlling each of the vehicles is able to compute a solution in less than the sampling time $T_s = 100\,\text{ms}$. However, there is not a great margin between the observed computation times and $T_s$. For instance, at the initial iteration, the MPC controller of vehicle 2 took almost $92\,\text{ms}$ to provide a solution. In addition, despite having a higher rate attitude controller inner loop, it would be extremely desirable to have the MPC controller running at a higher frequency than $10\,\text{Hz}$ for this kind of manoeuvres.

The many linear constraints defining the admissible states and control inputs significantly increase the computation time of the MPC regulator. Furthermore, using dedicated solvers to solve efficiently optimal control problems should be considered.

## 7.5 Aerodynamic Interactions

Here, the aerodynamic interactions between two quadcopters, two INTEL® Aero Ready to Fly quadrotors, are assessed by analysing the results of an experiment in which the vehicles are commanded to hover, one below the other, 1 metre apart. As discussed in Section 5.2, the turbulent downwash produced by the rotating propellers of the aircraft above affects negatively the performance of the rotorcraft below, potentially destabilising it. The results presented in this section demonstrate that the vehicle hovering below a turbulent airflow struggles visibly to reach a desired altitude constant reference, despite the rotorcraft being considerably apart from one another (about four times the propeller diameter of the rotorcraft involved in this experiment).

The aim of presenting the following experimental results is to make evident that including integral action to a translation control law, with the objective of counteracting the effect of the aerodynamic interactions between the rotorcraft, is not enough to make viable exchanging a parcel during flight in this manner, thus motivating the need to perform aggressive parcel relay manoeuvres.

The controller introduced in Section 5.3 was considered to conduct this experiment. In order to

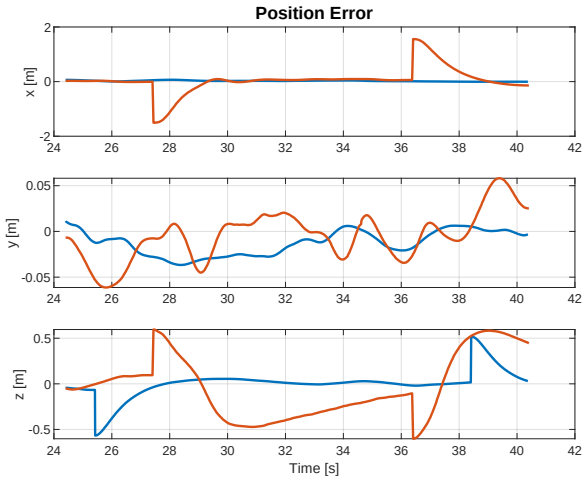
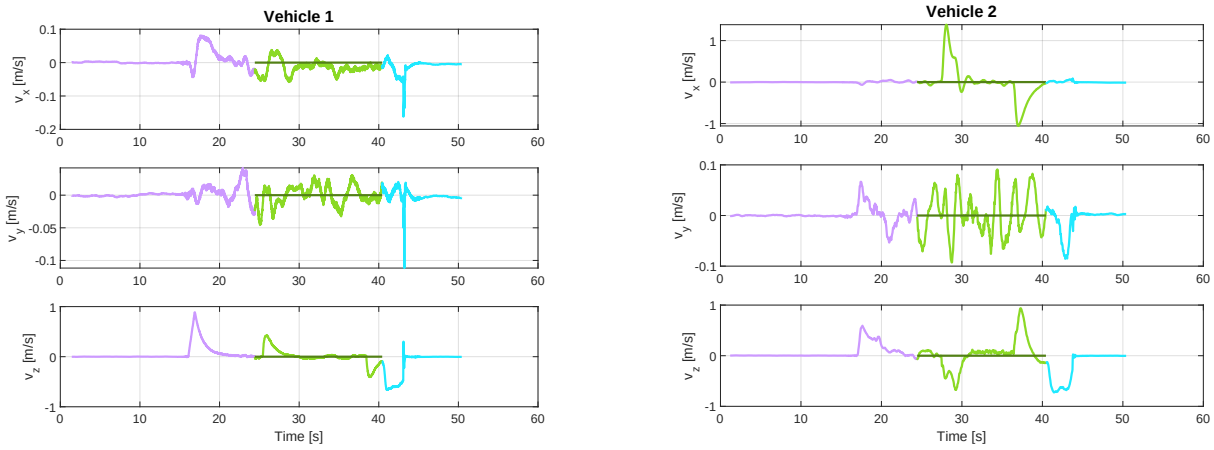



Figure 7.28: Position tracking error of both vehicles in close proximity while hovering one below the other, during off-board mode (the tracking error of the first vehicle is plotted in blue, of the second, in red).



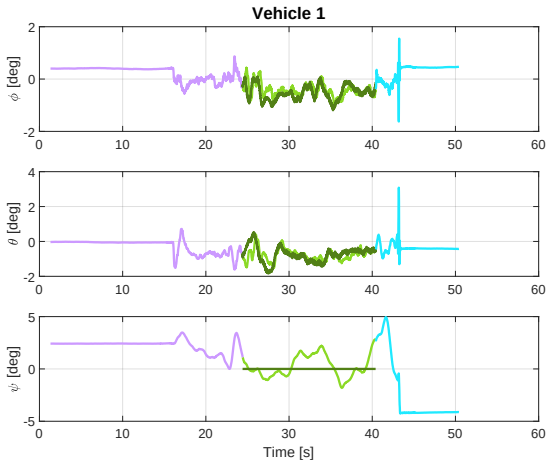

(a) Velocity response of the first vehicle.

(b) Velocity response of the second vehicle.

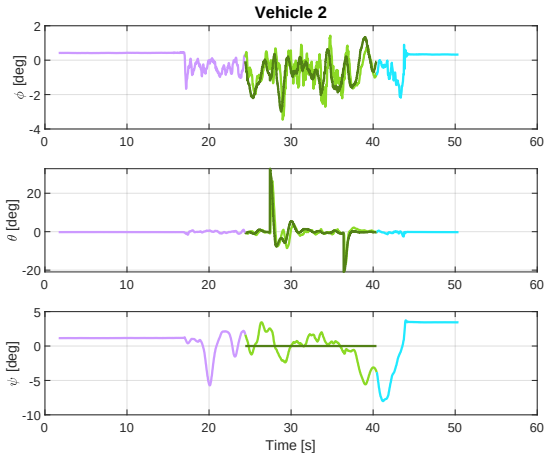
Figure 7.29: Velocity evolution of both vehicles in close proximity while hovering one below the other.

obtain the desired thrust vector $\boldsymbol{F}_{des}$ at every time instant, position $\boldsymbol{p}_T$, linear velocity $\boldsymbol{v}_T$, and linear acceleration $\boldsymbol{a}_T$ references where provided to the controller (5.6). From the yaw angle trajectory $\psi_T$ and $\boldsymbol{F}_{des}$, the desired orientation $\boldsymbol{R}_{des}$ is readily determined and given as input to the PX4 autopilot. It was chosen not to provide input torques to the autopilot for the reasons discussed in Section 7.1. One should note it is not recommended computing the desired angular velocity $\boldsymbol{\omega}_{des}$ from jerk $\boldsymbol{j}_T$ and the yaw angle time derivative $\dot{\psi}_T$ trajectories, using (5.10) and (5.11), and employing it as an input variable to the system, because no feedback control law is considered to obtain it.

The experiment is divided into three phases: takeoff, off-board control, and landing. The first and last phases were conveniently handled to the autopilot by setting suitable flight modes. In the plots to be presented, the trajectories associated with taking off are depicted in purple; the trajectories when the vehicle is receiving commands from an off-board computer, in green; the trajectories corresponding to the landing manoeuvre, in cyan. The observed trajectories are in lighter colours; the references, in darker shades. It was established that vehicle 1 is the aircraft hovering above vehicle 2.




(a) Orientation response of the first vehicle.




(b) Orientation response of the second vehicle.

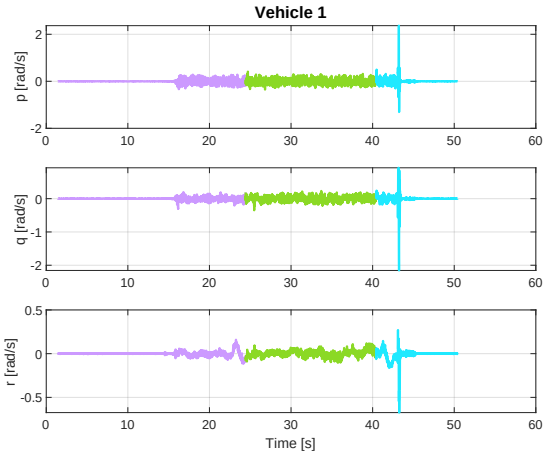
Figure 7.30: Orientation evolution of both vehicles in close proximity while hovering one below the other.

Figures 7.27(a) and 7.27(b) present the position plots of both vehicles. As soon as vehicle 2 is commanded to hover below vehicle 1, at $z = 1\,\text{m}$ (check the trajectories depicted in green in the plots of Figure 7.27(b)), descends somewhat sharply from $z = 1.5\,\text{m}$ to $z = 0.53\,\text{m}$. The integral term in the translation control law (5.6) successfully reduces the observed altitude offset from $47\,\text{cm}$ to $10\,\text{cm}$, however, when the vehicle is commanded to return to $z = 1.5\,\text{m}$, considerable overshoot in $z$ ($58\,\text{cm}$ to be exact) is observed due to the lack of better anti-windup strategies. In this experiment, the controller failed to reset this term, which demonstrates how costly this issue could have been. The position tracking error of each vehicle relative to their respective position reference trajectory, during off-board mode, is depicted in Figure 7.28 for further comparison.

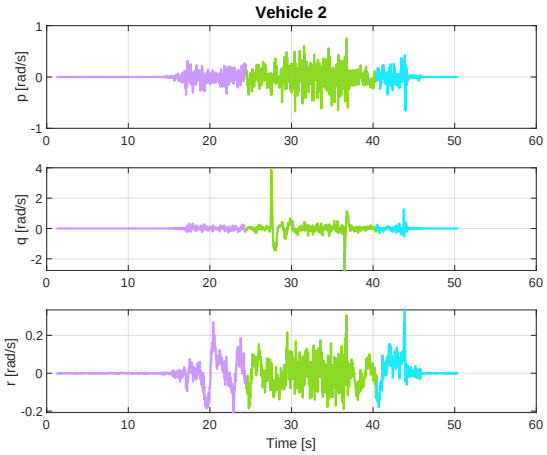
In Figures 7.30(a) and 7.30(b), the reference and observed orientation trajectories are depicted. Focusing again on vehicle 2, although the amplitude of the oscillations in roll $\phi(\cdot)$ seems not significant, it is observed that this vehicle needs to adjust its orientation more tightly when heading into the incoming airflow, when compared to the roll evolution of vehicle 1 (in Figure 7.30(a)). While the roll angle of vehicle 1 barely drifts from zero, the roll angle reference of vehicle 2 (depicted in dark green), oscillates between positive and negative values to counteract the effects of the downwash coming from above. On the other hand, the evolution of the pitch angle $\theta(\cdot)$ of each of the vehicles is slowly-varying around zero. In this experiment, the aerodynamic interactions between the two vehicles had no particular effect on the evolution of this angle.

It is also relevant to make an inspection of the velocity graphs presented in Figures 7.29(a) and 7.29(b). The variations of the roll angle of vehicle 2 reflect in the oscillations observed in the linear velocity in the $y$ axis, $v_y(\cdot)$. This plot ($2^{\text{nd}}$ tile of Figure 7.29(b)) provides further evidence that this vehicle had a tendency to move sideways, shaking lightly and quickly.

Examining the angular velocity plots, in Figures 7.31(a) and 7.31(b), it is verified that vehicle 2 wobbles increasingly as it gets closer to the other rotorcraft. Observe the pitch and yaw rates plots, $p(\cdot)$ and $r(\cdot)$, respectively, depicted in green of Figure 7.31(b) and compare with the plots of the same variables of vehicle 1 in Figure 7.31(a).




(a) Angular velocity response of the first vehicle.




(b) Angular velocity response of the second vehicle.

Figure 7.31: Angular velocity of both vehicles in close proximity while hovering one below the other.

To complement the analysis of the presented results, the reader is recommended to watch a video recorded during the execution of this experiment [60].

# Chapter 8

# Conclusions

The goal of this dissertation was to devise planning and control strategies, with an emphasis on model predictive control techniques, so that two drones can safely and effectively exchange a parcel during flight. The problem at hands was firstly tackled by proposing a method for generating reference trajectories suitable for parcel relay manoeuvres between quadrotors, then a non-linear tracking controller was considered to track those references and, finally, a model predictive controller was described to plan trajectories in online fashion.

The delineated strategy for planning relay manoeuvres, presented in Chapter 4, relies on the generation of optimal polynomial trajectories with acceleration constraints in order to coordinate the attitude of the vehicles during the package exchange. The problem is formulated as a mixed-integer quadratic program (MIQP), where the integer constraints ensure collision avoidance. Despite the planned trajectories being optimal in the sense that they minimise the necessary input body torques to track them, it would have been desirable to impose bound constraints on the derivatives of position to incorporate saturation and rate limit constraints into the trajectory generation problem. However, the sheer amount of additional inequality constraints that would have to be introduced would cause numerical issues to be more prone to arise.

A non-linear tracking controller was introduced and implemented in Chapter 5. In order to test this tracking controller, or any other, for quadrotors in software-in-the-loop simulations and experimental trials, software was developed using C++ to efficiently implement tracking controllers for quadrotors, using the features of MAVROS, being controlled by a single ground control station. Moreover, an experiment was conducted to analyse the aerodynamic interactions between two drones hovering on top of one another and to motivate the need to perform aggressive parcel relay manoeuvres. Although the obtained results demonstrate that the addition of integral action to the considered controller mitigates the effects the aerodynamic interactions between the vehicles, it was observed that the rotorcraft which was subject to the turbulent downwash had flight performance degradation.

The model predictive controller, described in Chapter 6, consists in an MPC regulator, to control the translation of a quadrotor, coupled with an attitude control law on $\mathrm{SO}(3) \times \mathbb{R}^3$ to stabilise the attitude dynamics. The fact that the translation dynamics of the vehicle was modelled as a triple integrator allows

to obtain control input sequences which satisfy certain bounds on thrust and angular velocity. A stability analysis of the triple integrator dynamics in closed-loop with the MPC regulator is presented and, the exponential stability of the origin of the closed-loop is established, under nominal conditions. Furthermore, based on the Dynamic Programming (DP) equations, algorithms are presented to determine analytically the region of attraction of the regulator.

Software-in-the-loop simulations were also conducted in the Gazebo simulator, involving the PX4 autopilot, where the vehicles were controlled by the tracking controller, presented in Chapter 5. The quadrotors were commanded to track reference trajectories obtained by relay manoeuvre planning algorithm. The vehicles exhibited high manoeuvrability and the obtained results indicate it is possible to exchange a package during flight, however, it was observed that thrust limitations hinder in the accurate following of reference trajectories. The preliminary experimental results to validate the planning of relay manoeuvres provided good indicators, in the sense that the vehicles were able to track the reference trajectories satisfactorily, however, problems regarding the synchronisation of the aircraft were detected, due to problems related to the configuration of the vehicles.

Finally, the performance of the model predictive controller, introduced in Chapter 6, was assessed by commanding the vehicles to perform a relay manoeuvre. The aircraft revealed good tracking performance and the results demonstrate that a parcel can be exchanged in mid-air between the two vehicles. Nevertheless, the aircraft achieved somewhat high roll and pitch rates at some time instants, along the manoeuvre.

## 8.1 Future Work

There are various paths that can be explored in terms of future work.

To further ensure the feasibility of the reference trajectories for performing aggressive manoeuvres, in the sense that the planned trajectories satisfy thrust and angular rates limitations, additional constraints should be introduced in the proposed methods for generating polynomial trajectories. Although binary slack variables allow to formulate convex collision-avoidance constraints, alternative strategies should be sought in order to generate trajectories which ensure not only inter-vehicle collisions are avoided, but also that take into account the actuation limits of the vehicles.

In what regards the planning of relay manoeuvres, the proposed backwards integration strategy to find the *launch* point of the manoeuvre is fundamental to guarantee the vehicles achieve a suitable relative orientation to exchange a parcel during flight. Possible lines of research include assessing the robustness of this strategy under disturbances, namely the aerodynamic interactions between the two rotorcraft and its performance when the vehicle dynamics are coupled with different controller types.

As for the proposed MPC regulator, software-in-the-loop (SITL) simulations should be run with the objective of evaluating its performance in more realistic conditions and to take further conclusions regarding its stability and feasibility. In addition, since the translational dynamics of a multirotor can be modelled as a triple integrator, the optimal control problem to plan online trajectories was cast as an convex optimisation problem, however, the proposed bound constraints on jerk cannot fully ensure that

the angular rates inputs provided to the vehicles do not surpass the maximum achievable value. One possible solution to this problem is considering an adaptive $\widetilde{T}$ (check (6.13)), for instance, the actual value of thrust, $T$ on the current iteration may be employed to set an adequate value for $\widetilde{T}$, in the next iteration.

Finally, it is worth coupling an actual mechanism to the vehicles, leveraging on the contributions of the work conducted by researchers of project REPLACE [9], and pursuing sensor-based cooperative control strategies so that the drones involved in the manoeuvre can reach a consensus on the best trajectories to ensure a successful parcel exchange.

# Bibliography


[1] R. D'Andrea. Guest Editorial - Can Drones Deliver? *IEEE Transactions on Automation Science and Engineering*, 11(3):647–648, 2014.

[2] Deutsche Post DHL Group — DHL Parcelcopter. [Online] Available: `https://www.dpdhl.com/en/media-relations/specials/dhl-parcelcopter.html`.

[3] A. Baker. The American Drones Saving Lives in Rwanda. *Time Magazine*, December 3, 2018. [Online] Available: `https://time.com/rwanda-drones-zipline/`.

[4] M. Giles. Zipline launches the world's fastest commercial delivery drone. *MIT Technology Review*, April 3, 2018. [Online] Available: `https://www.technologyreview.com/2018/04/03/144144/zipline-launches-the-worlds-fastest-commercial-delivery-drone/`.

[5] Amazon — Amazon Prime Air. [Online] Available: `https://www.amazon.com/Amazon-Prime-Air/b?node=8037720011`.

[6] S. Burns. Drone meets delivery truck. *UPS Longitudes*, February 22, 2017. [Online] Available: `https://www.ups.com/us/en/services/knowledge-center/article.page?kid=cd18bdc2`.

[7] D. K. Villa, A. S. Brandao, and M. Sarcinelli-Filho. A Survey on Load Transportation Using Multirotor UAVs. *Journal of Intelligent & Robotic Systems*, pages 1–30, 2019.

[8] D. Mellinger, Q. Lindsey, M. Shomin, and V. Kumar. Design, modeling, estimation and control for aerial grasping and manipulation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2668–2673. IEEE, 2011.

[9] F. Azevedo. Automatic Grasping System for Multi-drone Parcel Delivery. Master's thesis, Instituto Superior Técnico, Lisbon, Portugal, July 2019.

[10] N. Mathew, S. L. Smith, and S. L. Waslander. Planning Paths for Package Delivery in Heterogeneous Multirobot Teams. *IEEE Transactions on Automation Science and Engineering*, 12(4): 1298–1308, 2015.

[11] I. Hong, M. Kuby, and A. T. Murray. A range-restricted recharging station coverage model for drone delivery service planning. *Transportation Research Part C: Emerging Technologies*, 90:198–212, 2018.

[12] Project REPLACE. [Online] Available: http://replace.isr.tecnico.ulisboa.pt.

[13] Quad_Ctrl_MAVROS — Bitbucket Repository. [Online] Available: https://bitbucket.org/jspinto/quad_ctrl_mavros/src/master/, .

[14] PX4 Autopilot — Open Source Autopilot for Drones. [Online] Available: https://px4.io/.

[15] Robot Operating System. [Online] Available: https://www.ros.org/.

[16] MAVROS — ROS Wiki. [Online] Available: http://wiki.ros.org/mavros, .

[17] Gazebo Simulator. [Online] Available: http://gazebosim.org/.

[18] PX4_SITL_Gazebo_Mocap — Bitbucket Repository. [Online] Available: https://bitbucket.org/jspinto/px4_sitl_gazebo_mocap/src/master/.

[19] J. Pinto, B. J. Guerreiro, and R. Cunha. Planning Parcel Relay Manoeuvres for Quadrotors. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 137–145. IEEE, 2021.

[20] D. Mellinger, N. Michael, and V. Kumar. Trajectory Generation and Control for Precise Aggressive Maneuvers with Quadrotors. In *Int, Symposium on Experimental Robotics*, 2010.

[21] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 2520–2525, 2011.

[22] D. Mellinger, A. Kushleyev, and V. Kumar. Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 477–483, 2012.

[23] A. Richards and J. P. How. Aircraft Trajectory Planning With Collision Avoidance Using Mixed Integer Linear Programming. In *Proceedings of the American Control Conference (IEEE Cat. No. CH37301)*, volume 3, pages 1936–1941, Anchorage, AK, 2002. IEEE.

[24] T. Schouwenaars, B. De Moor, E. Feron, and J. How. Mixed Integer Programming for Multi-Vehicle Path Planning. In *2001 European control conference (ECC)*, pages 2603–2608, Porto, Portugal, 2001. IEEE.

[25] F. Augugliaro, A. P. Schoellig, and R. D'Andrea. Generation of collision-free trajectories for a quadrocopter fleet: A sequential convex programming approach. *IEEE International Conference on Intelligent Robots and Systems*, pages 1917–1922, 2012.

[26] J. H. Gillula, H. Huang, M. P. Vitus, and C. J. Tomlin. Design of guaranteed safe maneuvers using reachable sets: Autonomous quadrotor aerobatics in theory and practice. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 1649–1654, 2010.

[27] T. Lee, M. Leok, and N. McClamroch. Geometric Tracking Control of a Quadrotor UAV on SE(3). *IEEE Conference on Decision and Control*, pages 5420–5425, 2010.

[28] M. Kamel, M. Burri, and R. Siegwart. Linear vs Nonlinear MPC for Trajectory Tracking Applied to Rotary Wing Micro Aerial Vehicles. *IFAC-PapersOnLine*, 50(1):3463–3469, 2017.

[29] C. Sferrazza, M. Muehlebach, and R. D'Andrea. Trajectory tracking and iterative learning on an unmanned aerial vehicle using parametrized model predictive control. *2017 IEEE 56th Annual Conference on Decision and Control, CDC 2017*, pages 5186–5192, 2018.

[30] M. Faessler, D. Falanga, and D. Scaramuzza. Thrust Mixing, Saturation, and Body-Rate Control for Accurate Aggressive Quadrotor Flight. *IEEE Robotics and Automation Letters*, 2(2):476–482, 2017.

[31] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC press, Florida, 1994.

[32] A. Tayebi and S. McGilvray. Attitude Stabilization of a VTOL Quadrotor Aircraft. *IEEE Transactions on control systems technology*, 14(3):562–571, 2006.

[33] D. Brescianini, M. Hehn, and R. D'Andrea. Nonlinear Quadrocopter Attitude Control. Technical report, Eidgenössische Technische Hochschule Zürich (ETH Zürich), Departement Maschinenbau und Verfahrenstechnik, 2013.

[34] S. P. Bhat and D. S. Bernstein. A topological obstruction to continuous global stabilization of rotational motion and the unwinding phenomenon. *Systems & Control Letters*, 39(1):63–70, 2000.

[35] C. G. Mayhew, R. G. Sanfelice, and A. R. Teel. On quaternion-based attitude control and the unwinding phenomenon. In *Proceedings of the 2011 American Control Conference*, pages 299–304. IEEE, 2011.

[36] T. Lee. Exponential stability of an attitude tracking control system on SO (3) for large-angle rotational maneuvers. *Systems & Control Letters*, 61(1):231–237, 2012.

[37] N. Chaturvedi, A. K. Sanyal, and N. McClamroch. Rigid-Body Attitude Control. *IEEE Control Systems Magazine*, 31(3):30–51, 2011.

[38] U. V. Kalabić, R. Gupta, S. Di Cairano, A. M. Bloch, and I. V. Kolmanovsky. MPC on manifolds with an application to the control of spacecraft attitude on SO(3). *Automatica*, 76:293–300, 2017.

[39] M. Kamel, K. Alexis, M. Achtelik, and R. Siegwart. Fast Nonlinear Model Predictive Control for Multicopter Attitude Tracking on SO(3). In *2015 IEEE Conference on Control and Applications (CCA)*, pages 1160–1166. IEEE, 2015.

[40] C. Powers, D. Mellinger, A. Kushleyev, B. Kothmann, and V. Kumar. Influence of Aerodynamics and Proximity Effects in Quadrotor Flight. In *Experimental Robotics*, pages 289–302. Springer, 2013.

[41] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar. The GRASP Multiple Micro-UAV Test Bed. *IEEE Robotics & Automation Magazine*, 17(3):56–65, 2010.

[42] N. Sydney, B. Smyth, and D. A. Paley. Dynamic Control of Autonomous Quadrotor Flight in an Estimated Wind Field. In *52nd IEEE Conference on Decision and Control*, pages 3609–3616, Florence, Italy, 2014. IEEE.

[43] J. Leishman. *Principles of Helicopter Aerodynamics*. Cambridge University Press, 2nd edition, 2006.

[44] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[45] J. B. Rawlings, D. Q. Mayne, and M. M. Diehl. *Model predictive control: Theory, Computation, and Design*. Nob Hill Publishing, Santa Barbara, CA, 2nd edition, 2019.

[46] J. B. Rawlings and M. J. Risbeck. On the equivalence between statements with $\epsilon$-$\delta$ and k-functions. Technical report, University of Wisconsin-Madison, 2017.

[47] L. A. Wolsey and G. L. Nemhauser. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1st edition, 1988.

[48] International Business Machines Corporation. *IBM ILOG CPLEX Optimization Studio CPLEX User's Manual V12.8*, 2017.

[49] International Business Machines Corporation. *IBM ILOG CPLEX Optimization Studio Getting Started with CPLEX for MATLAB V12.8*, 2017.

[50] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics and Automation Magazine*, 19(3):20–32, 2012.

[51] S. Keerthi and E. Gilbert. Computation of Minimum-Time Feedback Control Laws for Discrete-Time Systems with State-Control Constraints. *IEEE Transactions on Automatic Control*, 32(5):432–435, 1987.

[52] OptiTrack — Motion Capture Systems. [Online] Available: `https://optitrack.com/`.

[53] MAVLink Developer Guide. [Online] Available: `https://mavlink.io/en/`, .

[54] INTEL® MavLink Router — GitHub Repository. [Online] Available: `https://github.com/mavlink-router/mavlink-router`, .

[55] tf2 — ROS Wiki. [Online] Available: `http://wiki.ros.org/tf2`.

[56] Quadrotor Roll Relay. [Online] Available: `https://youtu.be/1Ny10kDg6hY`.

[57] Preliminary Experimental Relay Manoeuvre (ISR Flying Arena). [Online] Available: `https://youtu.be/TcO0HD8QKcM`.

[58] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. CasADi – A software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1): 1–36, 2019.

[59] A. C. Hindmarsh, P. N. Brown, K. E. Grant, S. L. Lee, R. Serban, D. E. Shumaker, and C. S. Woodward. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Transactions on Mathematical Software (TOMS)*, 31(3):363–396, 2005.

[60] Quadrotors in close proximity while hovering. [Online] Available: `https://www.youtube.com/watch?v=9CqhjFE8CFY`, .

[61] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, 9th edition, 2004.

[62] J. Nocedal and S. Wright. *Numerical Optimization*. Springer Science & Business Media, New York, 2nd edition, 2006.

[63] H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, 3rd edition, 2002.

[64] N. Bof, R. Carli, and L. Schenato. Lyapunov Theory for Discrete Time Systems. Technical report, Università degli studi di Padova, Dipartimento di Ingegneria dell'Informazione, 2017.

# Appendix A

# Convex Optimisation

An optimisation problem is a mathematical object of the following form

$$\begin{array}{ll} \text{minimise} & f(\boldsymbol{x}) \\ \text{s.t.} & h_i(\boldsymbol{x}) = 0, \text{ for } i = 1, \ldots, p, \\ & g_i(\boldsymbol{x}) \leq 0, \text{ for } i = 1, \ldots, m. \end{array} \tag{A.1}$$

The vector $\boldsymbol{x} \in \mathbb{R}^n$ is the optimisation variable; the function $f : \mathbb{R}^n \to \mathbb{R}$ is the cost or objective function to be minimised; the functions $h_1, \ldots, h_p, g_1, \ldots, g_m : \mathbb{R}^n \to \mathbb{R}$ are constraint functions. Solving the problem (A.1) means finding a global minimiser, a vector $\boldsymbol{x}^*$ which has the smallest objective value among all vectors that satisfy the constraints.

In this appendix, the class of convex optimisation is addressed, based on the concepts presented in Chapters 1 to 4 of [61] and Chapters 1 and 2 of [62]. The importance of studying this class of optimisation problems lies in the fact that any minimiser is globally optimal. Furthermore, there are reliable and efficient methods for solving most problems of this class. On the other hand, finding a global minimiser for non-convex problems is usually an arduous task. Solving them usually involves some compromise. In broad terms, either one stops seeking a global minimiser in favour of quickly solving the optimisation problem, settling with finding local minima, or relaxes the requirement of finding a solution in a short amount of time so that a global minimiser is attained.

An optimisation problem is convex if and only if the cost function and the set formed by the constraints are convex. In the same fashion as in [61], it is considered that a convex optimisation problem has a constraint set specified by inequalities involving convex functions, $g_i(\cdot)$, and affine equality constraints, $h_i(\boldsymbol{x}) = \boldsymbol{a}_i^T \boldsymbol{x} + b_i$. These impositions ensure the feasible set is always convex. Let the following definitions demonstrate this statement is true.

**Definition A.1** (Convex Set)
*A set $S \subseteq \mathbb{R}^n$ is convex if the line segment between any two points in $S$ lies in $S$, that is, for $\boldsymbol{x}_1, \boldsymbol{x}_2 \in S$ and $0 \leq \alpha \leq 1$,*

$$\alpha \boldsymbol{x}_1 + (1 - \alpha)\boldsymbol{x}_2 \in S.$$

**Definition A.2** (Convex Function)

*A function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is convex if its domain $\boldsymbol{dom}\ \varphi \subseteq \mathbb{R}^n$ is a convex set, and for all $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{dom}\ \varphi$ and $0 \leq \alpha \leq 1$,*

$$\varphi(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha\varphi(\boldsymbol{x}) + (1-\alpha)\varphi(\boldsymbol{y}).$$

Geometrically interpreting this definition, it is deduced that a function $\varphi : \mathbb{R}^n \to \mathbb{R}$ is convex if, for all $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{dom}\ \varphi$ the line segment between $(\boldsymbol{x}, \varphi(\boldsymbol{x}))$ and $(\boldsymbol{y}, \varphi(\boldsymbol{y}))$ lies above its graph, and if $\boldsymbol{dom}\ \varphi$ is a convex set. Note that, in the context of Definition A.2, it is implicitly assumed that a convex function $\varphi(\cdot)$ is extended to all of $\mathbb{R}^n$ by defining its value to be $\infty$ outside its domain,

$$\tilde{\varphi}(\boldsymbol{x}) = \begin{cases} \varphi(\boldsymbol{x}) & \boldsymbol{x} \in \boldsymbol{dom}\ \varphi, \\ \infty & \boldsymbol{x} \notin \boldsymbol{dom}\ \varphi. \end{cases}$$

This simplifies the notation as there is not the necessity of describing explicitly the domain of the original function $\varphi(\boldsymbol{x})$, or add the qualifier 'for all $\boldsymbol{x} \in \boldsymbol{dom}\ \varphi$' every time one refers to it.

In order to verify whether the constraint set is indeed convex, one starts by recognising that the intersection of convex sets is convex. As the equality constraints are considered to be affine and, therefore, convex, it is required to examine the zero-sublevel sets of $g_i(\cdot),\ i = 1, \ldots, p$.

Let the $\alpha$-sublevel set of a function $\varphi : \mathbb{R}^n \to \mathbb{R}$ be denoted by $\mathrm{lev}_\alpha \varphi := \{\boldsymbol{x} \in \boldsymbol{dom}\ \varphi \mid \varphi(\boldsymbol{x}) \leq \alpha\}$. If $\boldsymbol{x}, \boldsymbol{y} \in \mathrm{lev}_\alpha \varphi$, then $\varphi(\boldsymbol{x}) \leq \alpha$ and $\varphi(\boldsymbol{y}) \leq \alpha$. If $\varphi(\cdot)$ is convex, it follows from Definition A.2 that $\varphi(\beta\boldsymbol{x} + (1-\beta)\boldsymbol{y}) \leq \alpha$, for $0 \leq \beta \leq 1$. Hence, Definition A.1 ensures that the sublevel sets of $\varphi(\cdot)$ are convex. By the same reasoning so are the zero-sublevel sets of $g_i(\cdot),\ i = 1, \ldots, p$.

In addition to Definition A.2, there are two important properties of convex functions. From the first-order convexity condition, as stated in Corollary A.1, local information about a convex function establishes a global lower bound on such function. The second order condition permits to evaluate whether a function is convex by inspecting its second derivative.

**Theorem A.1** (First-Order Convexity Condition)

*Suppose $\varphi : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and convex. If and only if its domain $\boldsymbol{dom}\ \varphi$ is also convex, then*

$$\varphi(\boldsymbol{y}) \geq \varphi(\boldsymbol{x}) + \nabla\varphi(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x})$$

*holds for all $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{dom}\ \varphi$.*

**Corollary A.1** (Global Minimiser of a Convex Function)

*Let $\varphi : \mathbb{R}^n \to \mathbb{R}$ be a continuously differentiable and convex function. It follows from Theorem A.1 that if $\nabla\varphi(\boldsymbol{x}^*) = 0$, then, for all $\boldsymbol{x} \in \boldsymbol{dom}\ \varphi$, $\varphi(\boldsymbol{x}) \geq \varphi(\boldsymbol{x}^*)$. This implies that $\boldsymbol{x}^*$ is a global minimiser of $\varphi(\cdot)$.*

**Theorem A.2** (Second-Order Convexity Condition)

*Suppose $\varphi : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable. Then, $\varphi(\cdot)$ is convex if and only if $\boldsymbol{dom}\ \varphi$ is convex and its Hessian matrix is positive semi-definite, that is, $\nabla^2\varphi(\boldsymbol{x}) \succcurlyeq 0$, for all $\boldsymbol{x} \in \boldsymbol{dom}\ \varphi$.*

Even though the previous definitions and theorems are sufficient for determining if a function is

convex, it is more practical to establish a set of elementary convex functions, and some convexity-preserving operations so as to build and recognise more intricate convex functions with greater ease.

Some elementary convex functions on $\mathbb{R}^n$ are, for example,

- From Definition A.2, it is evident that any affine function $\varphi : \mathbb{R}^n \to \mathbb{R}$, $\varphi(\boldsymbol{x}) = \boldsymbol{s}^T\boldsymbol{x} + r$ is convex;
- Any norm is convex as $\|\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}\| \leq \|\alpha\boldsymbol{x}\| + \|(1-\alpha)\boldsymbol{y}\| \leq \alpha\|\boldsymbol{x}\| + (1-\alpha)\|\boldsymbol{y}\|$, for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$;
- A quadratic form $\varphi : \mathbb{R}^n \to \mathbb{R}$, $\varphi(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}^T\boldsymbol{x} + c$ is convex if and only if $\boldsymbol{A} \succcurlyeq 0$, as a consequence of Theorem A.2.

The following set of theorems presents some operations preserve convexity which allow to recognise a myriad of convex functions.

**Theorem A.3** (Non-negative weighted sums of convex functions)
*Let $\varphi_i : \mathbb{R}^n \to \mathbb{R}$ be convex functions and $\alpha_i \geq 0$ for $i = 1, \ldots, n$, then $\varphi(\cdot) = \alpha_1\varphi_1(\cdot) + \cdots \alpha_n\varphi_n(\cdot)$ is convex.*

**Theorem A.4** (Composition of a convex function with an affine mapping)
*Suppose $\varphi : V \to \mathbb{R}$ and $\boldsymbol{\xi} : W \to V$, $\boldsymbol{\xi}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{b}$ is an affine map between two vector spaces $V$ and $W$. Then, if $\varphi(\cdot)$ is convex, so is $(\varphi \circ \boldsymbol{\xi})(\cdot)$.*

**Theorem A.5** (Composition of a convex functions)
*Suppose $\varphi : \mathbb{R}^m \to \mathbb{R}$ is a convex function and $\boldsymbol{\xi} : \mathbb{R}^n \to \mathbb{R}^m$, a convex map. Then, if $\varphi(\cdot)$ is non-decreasing in each argument, the function $(\varphi \circ \boldsymbol{\xi})(\cdot)$ is convex.*

In order to evaluate the usefulness of these theorems, consider the following simple example. Let $\varphi : \mathbb{R}^N \to \mathbb{R}$,

$$\varphi(x_1, \cdots, x_N) = \sum_{j=1}^{N} \left( x_j - \frac{x_1 + \cdots + x_N}{N} \right)^2$$

Within the parent class of convex optimisation problems, there are various sub-classes, characterised by particular forms of the objective and constraint functions. For each of these sub-classes, there are efficient and reliable algorithms for solving them. After identifying that an optimisation problem is convex, one should inspect it and select the adequate solver for such problem.

## A.1 Mixed-Integer Programming

In this section, the class of Mixed-Integer Programming (MIP) is summarily presented. In the scope of this thesis, the interest in the inclusion of integer variables, in particular, binary variables into an optimisation problem lies in the possibility of formulating convex collision avoidance constraints in trajectory planning algorithms, as discussed in Chapter 4.

As for mixed-integer programs the optimisation variable $\boldsymbol{x}$ lives in $\mathbb{R}^n \times \mathbb{Z}^k$, in comparison to the optimisation problem in (A.1), the main difference is the addition of a constraint imposing that the last $k$ variables to be optimised are integers,

$$\begin{aligned} &\text{minimise} && f(\boldsymbol{x}) \\ &\text{s.t.} && h_i(\boldsymbol{x}) = 0, \text{ for } i = 1, \dots, p, \\ & && g_i(\boldsymbol{x}) \leq 0, \text{ for } i = 1, \dots, m, \\ & && \boldsymbol{x}_i \in \mathbb{Z}, \text{ for } i = n+1, \dots, n+k. \end{aligned} \tag{A.2}$$

In order to ensure a MIP is convex, it is required to grasp how it is solved. Instead of optimising $f(\cdot)$ over the whole feasible set $\mathcal{S}$, the problem is solved by optimising over smaller sets, and finding over which of those sets the best solution is obtained. The division of the feasible set is most often done recursively and schematically represented by a tree diagram. At each node of the tree, the integer constraints are *relaxed*, that is, the last constraint of (A.2) is dropped, and the partition is done by adding linear constraints to the optimisation problem [47]. Hence, ignoring the integer constraint of (A.2), the problem should be convex so as to quickly obtain a global minimiser at every node.

# Appendix B

# Basic Stability Theory

In this appendix, some basic stability properties of discrete systems are introduced and some Lyapunov related theorems for these systems are similarly presented. This summary on stability theory was based on the analysis of Appendix B of [45], Chapter 4 of [63], which is a general reference for stability theory of continuous time systems, and, finally, a technical report on Lyapunov theory for discrete systems [64]. The concepts in this section establish a framework for studying the stability of any deterministic discrete time-invariant system.

Consider the time-invariant autonomous system,

$$\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k)), \tag{B.1}$$

where $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^n$ is a continuous map. An equilibrium point of (B.1) is a point $\bar{\boldsymbol{x}} \in \mathbb{R}^n$ which satisfies $f(\bar{\boldsymbol{x}}) = \bar{\boldsymbol{x}}$, for all $k$. In the same manner as in Appendix B of [45], the conditions to characterise the convergence to a closed, positive invariant set $\mathcal{A}$ are presented. The establishment of more general conditions may be necessary for studying the stability of a system in closed-loop with a MPC controller.

**Definition B.1** (Positive Invariant Set)
*A set $\mathcal{A}$ is positive invariant for $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k))$ if $\boldsymbol{x}(0) \in \mathcal{A} \Rightarrow \boldsymbol{x}(k) \in \mathcal{A},\ \forall k \in \mathbb{N}_0$.*

Notice that for the particular case $\mathcal{A} \coloneqq \{\bar{\boldsymbol{x}}\}$, this definition ensures that if $\boldsymbol{x} \in \mathcal{A}$, $\boldsymbol{f}(\boldsymbol{x}) = \bar{\boldsymbol{x}}$.

In order to introduce some basic concepts of stability, the distance between set $\mathcal{A}$ and a point $\boldsymbol{x}$ is defined as

$$\|\boldsymbol{x}\|_{\mathcal{A}} \coloneqq \inf_{\boldsymbol{z} \in \mathcal{A}} \|\boldsymbol{x} - \boldsymbol{z}\| .$$

**Definition B.2** (Basic Stability Concepts)
*The closed, positive invariant set $\mathcal{A}$ is*

- *locally stable if, for each $\varepsilon > 0$, there exists a $\delta = \delta(\varepsilon)$ such that*

$$\|\boldsymbol{x}(0)\|_{\mathcal{A}} < \delta \Rightarrow \|\boldsymbol{x}(k)\|_{\mathcal{A}} < \varepsilon,\ \forall k \in \mathbb{N}_0;$$

- *unstable in case it is not locally stable;*

- *locally attractive if there exists* $\eta > 0$ *such that*

$$\|\boldsymbol{x}(0)\|_{\mathcal{A}} < \eta \Rightarrow \lim_{k\to\infty} \|\boldsymbol{x}(k)\|_{\mathcal{A}} = 0;$$

- *globally attractive if*

$$\lim_{k\to\infty} \|\boldsymbol{x}(k)\|_{\mathcal{A}} = 0,\ \forall \boldsymbol{x}(0) \in \mathbb{R}^n;$$

- *asymptotically stable if locally stable and locally attractive;*

- *globally asymptotically stable if locally stable and globally attractive;*

- *exponentially stable if there exist* $\eta > 0$, $c > 0$, *and* $\gamma \in (0, 1)$ *such that*

$$\|\boldsymbol{x}(0)\|_{\mathcal{A}} < \eta \Rightarrow \|\boldsymbol{x}(k)\|_{\mathcal{A}} \leq c \|\boldsymbol{x}(0)\|_{\mathcal{A}} \gamma^k,\ \forall k \in \mathbb{N}_0.$$

## B.1 Lyapunov Stability Theory

The Lyapunov theory provides sufficient conditions for establishing the stability and asymptotic stability of a system [63]. The core idea is to define a positive definite function, a continuously differentiable function that is zero at zero and positive everywhere else, which value decreases along the trajectories of the system. In this fashion, as time tends to infinity, the state approaches the desired set or equilibrium. Let the next theorem define rigorously what a Lyapunov function is. Before introducing such theorem, it is necessary to take into account the following definition.

**Definition B.3** (Class $\mathcal{K}$ function)
*A continuous function* $\alpha : [0, a) \to [0, \infty)$ *is said to belong to class* $\mathcal{K}$ *if it is strictly increasing and* $\alpha(0) = 0$.

**Theorem B.1** (Lyapunov Function and Asymptotic Stability, from [45])
*Suppose* $X \subseteq \mathbb{R}^n$ *is positive invariant, the set* $\mathcal{A} \subseteq X$ *is closed and also positive invariant for (B.1), and* $\boldsymbol{f}(\cdot)$ *is locally bounded. A function* $V : X \to \mathbb{R}_0^+$ *is a Lyapunov function in* $X$ *for the system (B.1) and set* $\mathcal{A}$ *if there exist functions* $\alpha_1,\ \alpha_2 \in \mathcal{K}$, *and a positive definite function* $\alpha_3$ *such that for any* $\boldsymbol{x} \in X$

$$\begin{aligned} \alpha_1(\|\boldsymbol{x}\|_{\mathcal{A}}) \leq V(\boldsymbol{x}) \leq \alpha_2(\|\boldsymbol{x}\|_{\mathcal{A}}) \\ V(\boldsymbol{f}(\boldsymbol{x})) - V(\boldsymbol{x}) \leq -\alpha_3(\|\boldsymbol{x}\|_{\mathcal{A}}), \end{aligned}$$

- *Then,* $\mathcal{A}$ *is asymptotically stable in* $X$ *for* $\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k))$;

- *Exponential stability is established if* $\alpha_i(\cdot) = a_i \|\cdot\|_{\mathcal{A}}^{\sigma},\ a_i,\ \sigma > 0$.

Consider now the linear time-invariant system,

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k),\ \boldsymbol{A} \in \mathbb{R}^{n\times n}. \tag{B.2}$$

In the sequel it is assumed that $\boldsymbol{A}$ is full rank, and, therefore, $\boldsymbol{x} = 0$ is the unique equilibrium point of (B.2). The following theorem explains how to characterise the stability of the origin.

**Theorem B.2** (Stability Properties of the Origin, from [64])
*The equilibrium point $\boldsymbol{x} = 0$ of the linear time-invariant system (B.2) is*

- *stable if and only if all the eigenvalues of $\boldsymbol{A}$ satisfy $|\lambda_i| \leq 1$, and for every eigenvalue which $|\lambda_i| = 1$ its algebraic and geometric multiplicity match;*
- *globally asymptotically stable if and only if all the eigenvalues of $\boldsymbol{A}$ are such that $|\lambda_i| < 1$.*

A matrix $\boldsymbol{A}$ for which all eigenvalues satisfy $|\lambda_i| < 1$ is called a *Schur matrix*, and, hence, the origin of (B.2) is asymptotically stable if and only if $\boldsymbol{A}$ is Schur.

The asymptotic stability of the origin may also be verified by employing the Lyapunov stability theory. To this end, consider the following quadratic Lyapunov candidate

$$V(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x},$$

where $\boldsymbol{P}$ is a real symmetric positive definite matrix. The discrete derivative of $V(\cdot)$ along the trajectories of (B.2) is given by

$$V(\boldsymbol{A}\boldsymbol{x}) - V(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} \boldsymbol{x} - \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} = \boldsymbol{x}^T (\boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} - \boldsymbol{P}) \boldsymbol{x} = -\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x}$$

where $\boldsymbol{Q}$ is a symmetric matrix defined by the Lyapunov Equation

$$\boldsymbol{A}^T \boldsymbol{P} \boldsymbol{A} - \boldsymbol{P} = -\boldsymbol{Q}. \tag{B.3}$$

If $\boldsymbol{Q}$ is positive definite, according to Theorem B.1, the origin is asymptotically stable, since $\lambda_{min} \|\boldsymbol{x}\|^2 \leq \boldsymbol{x}^T \boldsymbol{P} \boldsymbol{x} \leq \lambda_{max} \|\boldsymbol{x}\|^2$. In this fashion, selecting a real symmetric positive definite matrix $\boldsymbol{Q}$, and solving (B.3) for $\boldsymbol{P}$, if this matrix is symmetric and positive definite, one can establish the asymptotic stability of the origin, as stated in the following theorem.

**Theorem B.3** (Lyapunov Function for Linear Systems, from [64])
*A matrix $\boldsymbol{A}$ is Schur if and only if, for any positive definite matrix $\boldsymbol{Q}$ there exists a positive definite symmetric matrix $\boldsymbol{P}$ that satisfies (B.3). Moreover, if $\boldsymbol{A}$ is Schur, then $\boldsymbol{P}$ is the unique solution of (B.3).*

The proof of this last theorem is provided in [64].

# Appendix C

# NMPC Regulator Closed-Loop Stability

In order to use the assumptions listed in Section 3.1.1 to ascertain whether the origin of a deterministic time-invariant system in closed-loop with a NMPC controller is stable, an example based on one provided in [45] is presented in this appendix.

The system to be controlled is described by $\boldsymbol{x}(k+1) = \boldsymbol{f}(\boldsymbol{x}(k), \boldsymbol{u}(k))$, where it is assumed that $\boldsymbol{f}(\cdot)$ is twice continuously differentiable. It is considered that the system is subject to state and control constraints $\boldsymbol{x} \in \mathbb{X},\ \boldsymbol{u} \in \mathbb{U}$. The set $\mathbb{X}$ is closed and $\mathbb{U}$ is compact. Both sets contain the origin.

The stage cost $\ell(\cdot)$ of the objective function is selected to be a positive definite quadratic,

$$\ell(\boldsymbol{x}, \boldsymbol{u}) = \frac{1}{2}\left[\boldsymbol{x}^T \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^T \boldsymbol{R} \boldsymbol{u}\right], \tag{C.1}$$

where $\boldsymbol{Q}, \boldsymbol{R} \succ 0$.

So as to ensure the origin is asymptotically stable, the terminal cost function $V_f(\cdot)$ and the terminal set $\mathbb{X}_f$ have to be defined. The terminal cost function is obtained by linearising the non-linear system at the origin. The linearised system is defined as

$$\boldsymbol{x}(k+1) = \boldsymbol{A}\boldsymbol{x}(k) + \boldsymbol{B}\boldsymbol{u}(k), \tag{C.2}$$

with

$$\boldsymbol{A} \coloneqq \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{(0,0)},\ \boldsymbol{B} \coloneqq \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{(0,0)}.$$

Assume that the pair $(\boldsymbol{A}, \boldsymbol{B})$ is stabilisable and select a linear state feedback terminal control law $\boldsymbol{\kappa}_f(\boldsymbol{x}) = \boldsymbol{K}\boldsymbol{x}$, for example the infinite horizon LQR controller, such that the origin is globally exponentially stable (GES). A discrete linear system is stabilisable if there exists a gain matrix $\boldsymbol{K}$ which ensures the origin is stable (check Theorem B.2).

The terminal cost function is defined as a Lyapunov function,

$$V_f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x},$$

where the matrix $\boldsymbol{P}$ is obtained by solving the Lyapunov Equation,

$$\widetilde{\boldsymbol{A}}^T\boldsymbol{P}\widetilde{\boldsymbol{A}} + \mu\widetilde{\boldsymbol{Q}} = \boldsymbol{P}, \tag{C.3}$$

where $\widetilde{\boldsymbol{A}} \coloneqq \boldsymbol{A} + \boldsymbol{B}\boldsymbol{K}$. The matrix $\widetilde{\boldsymbol{Q}} \coloneqq \boldsymbol{Q} + \boldsymbol{K}^T\boldsymbol{R}\boldsymbol{K}$ is positive definite, and comes from the stage cost $\ell(\boldsymbol{x}, \boldsymbol{K}\boldsymbol{x}) = (1/2)\boldsymbol{x}^T\widetilde{\boldsymbol{Q}}\boldsymbol{x}$. The reason for which the constant $\mu > 1$ was introduced will become clear soon. Notice that given $\widetilde{\boldsymbol{Q}} \succ 0$, there exists an unique $\boldsymbol{P} \succ 0$.

It can be verified that $V_f(\cdot)$ is a global Lyapunov function for the linearised system (C.2) under the terminal control law $\boldsymbol{\kappa}_f(\cdot)$, since it follows from (C.3) that the value of this function decreases along the trajectories of the closed-loop system for all $\boldsymbol{x} \in \mathbb{R}^n$,

$$\begin{aligned} V_f(\widetilde{\boldsymbol{A}}\boldsymbol{x}) - V_f(\boldsymbol{x}) &\leq -\mu \cdot \ell(\boldsymbol{x}, \boldsymbol{K}\boldsymbol{x}) \\ \frac{1}{2}\boldsymbol{x}^T\widetilde{\boldsymbol{A}}^T\boldsymbol{P}\widetilde{\boldsymbol{A}}\boldsymbol{x} + \frac{\mu}{2}\boldsymbol{x}^T\widetilde{\boldsymbol{Q}}\boldsymbol{x} - \frac{1}{2}\boldsymbol{x}^T\boldsymbol{P}\boldsymbol{x} &= 0. \end{aligned} \tag{C.4}$$

At this point, if it were the asymptotic stability of the origin of the linearised system to be established, there would be no need to define a terminal constraint. The result above leads to the conclusion that the terminal control law $\boldsymbol{\kappa}_f(\boldsymbol{x}) = \boldsymbol{K}\boldsymbol{x}$ would drive the state of the system to zero starting from any point in $\mathbb{R}^n$. However, for the non-linear system the same conclusion cannot be drawn. It has to be demonstrated that $V_f(\cdot)$ is a local Lyapunov Function in some neighbourhood of the origin. In this manner, the terminal set $\mathbb{X}_f$ has to be defined.

For being more practical, the problem of defining $\mathbb{X}_f$ and demonstrating the asymptotic stability of the origin is established as showing there is an $a \in \mathbb{R}^+$ for which

$$V_f(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{K}\boldsymbol{x})) + \frac{1}{2}\boldsymbol{x}^T\widetilde{\boldsymbol{Q}}\boldsymbol{x} - V_f(\boldsymbol{x}) \leq 0, \ \forall \boldsymbol{x} \in \mathrm{lev}_a V_f. \tag{C.5}$$

Combining the previous inequality (C.5) with (C.4), it is observed that satisfying the next inequality ensures the satisfaction of the one above,

$$V_f(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{K}\boldsymbol{x})) - V_f(\widetilde{\boldsymbol{A}}\boldsymbol{x}) \leq \frac{\mu - 1}{2}\boldsymbol{x}^T\widetilde{\boldsymbol{Q}}\boldsymbol{x}, \ \forall \boldsymbol{x} \in \mathrm{lev}_a V_f. \tag{C.6}$$

Let $\boldsymbol{e} : \mathbb{R}^n \to \mathbb{R}^n$ be defined as

$$\boldsymbol{e}(\boldsymbol{x}) = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{K}\boldsymbol{x}) - \widetilde{\boldsymbol{A}}\boldsymbol{x} = \boldsymbol{w}(\boldsymbol{x}) - \boldsymbol{y}(\boldsymbol{x}). \tag{C.7}$$

Using this definition, the left slide of the inequality (C.6) is expressed as a function of $\boldsymbol{e}(\cdot)$,

$$\begin{aligned} V_f(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{K}\boldsymbol{x})) - V_f(\widetilde{\boldsymbol{A}}\boldsymbol{x}) &= \frac{1}{2}\left[\boldsymbol{w}^T\boldsymbol{P}\boldsymbol{w} - \boldsymbol{y}^T\boldsymbol{P}\boldsymbol{y}\right] \\ &= \frac{1}{2}\boldsymbol{e}^T\boldsymbol{P}\boldsymbol{e} + \boldsymbol{y}^T\boldsymbol{P}\boldsymbol{e}. \end{aligned} \tag{C.8}$$

From (C.7), $\boldsymbol{e}(\boldsymbol{0}) = \boldsymbol{0}$, and, provided that $\boldsymbol{u} = \boldsymbol{K}\boldsymbol{x}$,

$$\left.\frac{d\boldsymbol{e}}{d\boldsymbol{x}}\right|_{x=0} = \left[\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} + \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\frac{\partial \boldsymbol{u}}{\partial \boldsymbol{x}}\right]_{(0,0)} - \widetilde{\boldsymbol{A}} = \boldsymbol{0}_{1\times n}. \tag{C.9}$$

Since $\boldsymbol{f}(\cdot)$ is $C^2$, for any $\delta > 0$, there exists, according to Extreme Value Theorem, $c_\delta > 0$ such that $\|e_i''(\boldsymbol{x})\| \leq c_\delta$, for all $\boldsymbol{x}$ within a closed ball $\mathcal{B}_\delta$ with radius $\delta$. Using Preposition D.1,

$$\begin{aligned} |e_i(\boldsymbol{x})| &= \left| e_i(\boldsymbol{0}) + e_i'(\boldsymbol{0})\boldsymbol{x} + \int_0^1 (1-s)\boldsymbol{x}^T e_i''(s\boldsymbol{x})\boldsymbol{x}\, ds \right| \\ &\leq \int_0^1 \left|(1-s)\boldsymbol{x}^T e_i''(s\boldsymbol{x})\boldsymbol{x}\right|\, ds \\ &\leq \int_0^1 (1-s)c_\delta \left\|\boldsymbol{x}\right\|^2\, ds \\ &\leq \frac{1}{2}c_\delta \left\|\boldsymbol{x}\right\|^2. \end{aligned}$$

As each component of $\boldsymbol{e}(\cdot)$ is bounded by a positive definite quadratic, one concludes from (C.8) that there is an $\varepsilon \in (0, \delta]$ for which (C.6), and, consequently, (C.5) are satisfied for all $\boldsymbol{x} \in \mathcal{B}_\varepsilon$.

Furthermore, it is verified from (C.3), that there exists a $c_1 \in \mathbb{R}^+$ such that $V_f(\boldsymbol{x}) \geq \ell(\boldsymbol{x}, \boldsymbol{K}\boldsymbol{x}) \geq c_1 \left\|\boldsymbol{x}\right\|^2$, $\forall \boldsymbol{x} \in \mathbb{R}^n$. Since $\boldsymbol{x} \in \text{lev}_a V_f$, it follows that $\|\boldsymbol{x}\| \leq \sqrt{a/c_1}$. If $a$ is chosen such that $\varepsilon = \sqrt{a/c_1}$, the condition $\boldsymbol{x} \in \text{lev}_a V_f$ implies that $\|\boldsymbol{x}\| \leq \varepsilon \leq \delta$ and, therefore, the inequality (C.5) is satisfied.

In this manner, one demonstrates there is an $a > 0$ for which $\mathbb{X}_f \coloneqq \text{lev}_a V_f$ is compact and contains the origin in its interior, as any sublevel set of a positive definite function is an ellipsoid with the origin as its centre. In addition, there is a terminal control law $\boldsymbol{\kappa}_f(\boldsymbol{x}) = \boldsymbol{K}\boldsymbol{x}$ which ensures that Assumption 3.4 is satisfied, $V_f(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_f(\boldsymbol{x})) - V_f(\boldsymbol{x}) \leq -\ell(\boldsymbol{x}, \boldsymbol{\kappa}_f(\boldsymbol{x})) \leq c_2 \left\|\boldsymbol{x}\right\|^2$, $\forall \boldsymbol{x} \in \mathbb{X}_f$, where $c_2 \in \mathbb{R}^+$.

Furthermore, since $\ell(\boldsymbol{x}, \boldsymbol{u}) = (1/2)\left[\boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} + \boldsymbol{u}^T\boldsymbol{R}\boldsymbol{u}\right]$, with $\boldsymbol{Q}$ and $\boldsymbol{R}$ positive definite, $V_N^0(\boldsymbol{x}) \geq \ell(\boldsymbol{x}, \boldsymbol{u}) \geq c_1 \left\|\boldsymbol{x}\right\|^2$, $\forall \boldsymbol{x} \in \mathcal{X}_N$, and there exists a $c_f \in \mathbb{R}^+$ which sets an upper bound on the terminal cost, $V_f(\boldsymbol{x}) \leq c_f \left\|\boldsymbol{x}\right\|^2$, $\forall \boldsymbol{x} \in \mathbb{X}_f$. Hence, Assumption 3.3 is satisfied.

In summary, Assumptions 3.1, 3.2, 3.3, and 3.4 are satisfied and there are $\mathcal{K}$ functions and constants $c_1, c_f \in \mathbb{R}^+$ which ensure that $V_N^0(\boldsymbol{x}) \geq c_1 \left\|\boldsymbol{x}\right\|^2$, $\forall \boldsymbol{x} \in \mathcal{X}_N$; $V_N^0(\boldsymbol{x}) \leq c_f \left\|\boldsymbol{x}\right\|^2$, $\forall \boldsymbol{x} \in \mathbb{X}_f$; $V_N^0\left(\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{\kappa}_N(\boldsymbol{x}))\right) - V_N^0(\boldsymbol{x}) \leq -c_1 \left\|\boldsymbol{x}\right\|^2$, $\forall \boldsymbol{x} \in \mathcal{X}_N$.

By Theorem 3.1, the origin is exponentially stable for $\boldsymbol{x}(k+1) = f(\boldsymbol{x}(k), \kappa_N(\boldsymbol{x}(k))$ in $\mathcal{X}_N$.

# Appendix D

# Mean Value Theorem for Vector Fields

The following proposition is presented in [45, Proposition A.11] for scalar fields. In this appendix, it is generalised for vector fields.

**Proposition D.1** (Mean Value Theorem for Vector Fields)
*Suppose that vector field $\boldsymbol{h} : \mathbb{R}^n \to \mathbb{R}^m$ has continuous partial derivatives of order two at each point $\boldsymbol{x}$ of $\mathbb{R}^n$. Then for any $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$,*

$$h_i(\boldsymbol{y}) = h_i(\boldsymbol{x}) + h_i'(\boldsymbol{\xi}(0))(\boldsymbol{y}-\boldsymbol{x}) + \int_0^1 (1-s)(\boldsymbol{y}-\boldsymbol{x})^T h_i''\left(\boldsymbol{\xi}(s)\right)(\boldsymbol{y}-\boldsymbol{x})\, ds, \text{ for } i = 1, \ldots, m,$$

*where the map $\boldsymbol{\xi} : \mathbb{R} \to \mathbb{R}^n$ is provided by $\boldsymbol{\xi}(s) = \boldsymbol{x} + s(\boldsymbol{y}-\boldsymbol{x})$ and $h_i : \mathbb{R}^n \to \mathbb{R}$ denotes the $i^{th}$ component of vector field $\boldsymbol{h}(\cdot)$.*

*Proof.* Consider the map $\boldsymbol{g}(s) = \boldsymbol{h}(\boldsymbol{\xi}(s))$. Then,

$$\frac{d}{ds}\left[(1-s)\boldsymbol{g}'(s) + \boldsymbol{g}(s)\right] = \boldsymbol{g}''(s)(1-s).$$

Integrating this expression from 0 to 1,

$$\int_0^1 \boldsymbol{g}''(s)(1-s)\, ds = \left[(1-s)\boldsymbol{g}'(s) + \boldsymbol{g}(s)\right]_0^1 = \boldsymbol{g}(1) - \boldsymbol{g}(0) - \boldsymbol{g}'(0). \tag{D.1}$$

The first and second derivatives of $\boldsymbol{g}(\cdot)$ are computed as follows, note that given the definition of $\boldsymbol{\xi}(\cdot)$, $\boldsymbol{\xi}''(s) = \boldsymbol{0}_{n\times 1},\ \forall s \in \mathbb{R}$,

$$\boldsymbol{g}'(s) = \frac{d\boldsymbol{h}}{d\boldsymbol{\xi}}\frac{d\boldsymbol{\xi}}{ds} = \boldsymbol{h}'\left(\boldsymbol{\xi}(s)\right)(\boldsymbol{y}-\boldsymbol{x}),$$

$$\begin{aligned} g_i''(s) &= \frac{d}{ds}\left[\frac{dh_i}{d\boldsymbol{\xi}}\frac{d\boldsymbol{\xi}}{ds}\right] = \left[\frac{d^2h_i}{d\boldsymbol{\xi}d\boldsymbol{\xi}^T}\frac{d\boldsymbol{\xi}}{ds}\right]\cdot\frac{d\boldsymbol{\xi}}{ds} + \frac{dh_i}{d\boldsymbol{\xi}}\frac{d^2\boldsymbol{\xi}}{ds^2} \\ &= (\boldsymbol{y}-\boldsymbol{x})^T h_i''\left(\boldsymbol{\xi}(s)\right)(\boldsymbol{y}-\boldsymbol{x}). \end{aligned}$$

In this manner, when substituting the expression obtained for the first and second derivatives in (D.1), and replacing $\boldsymbol{g}(s)$ with $\boldsymbol{h}(\boldsymbol{\xi}(s))$, the proof is complete. ■

# Appendix E

# Planning as a QP Problem

In this appendix, it is demonstrated that the established optimisation problem (4.5) for generating polynomial trajectories for a single quadcopter can be structured as a QP.

First and foremost, the cost function $f$ is split into a sum of cost functions $f_n$. The cost function $f_n$ is associated with the cost of the segment $\boldsymbol{\sigma}_n(\cdot)$ of the trajectory,

$$\begin{aligned} f &= \int_{t_0}^{t_M} \mu_p \left\| \frac{d^4\boldsymbol{p}}{dt^4} \right\|^2 + \mu_\psi \frac{d^2\psi}{dt^2}^2 \, dt \\ &= \sum_{n=1}^{M} \int_{t_{n-1}}^{t_n} \mu_p \left\| \frac{d^4\boldsymbol{p}_n}{dt^4} \right\|^2 + \mu_\psi \frac{d^2\psi_n}{dt^2}^2 \, dt \\ &= \sum_{n=1}^{M} \mu_p \int_{t_{n-1}}^{t_n} \frac{d^4x_n}{dt^4}^2 + \frac{d^4y_n}{dt^4}^2 + \frac{d^4z_n}{dt^4}^2 \, dt + \mu_\psi \int_{t_{n-1}}^{t_n} \frac{d^2\psi_n}{dt^2}^2 \, dt \\ &= \sum_{n=1}^{M} \mu_p (f_{nx} + f_{ny} + f_{nz}) + \mu_\psi f_{n\psi} \ . \end{aligned} \tag{E.1}$$

Since the flat outputs were developed as polynomial expressions, let $h_k = k(k-1)(k-2)(k-3)$,

$$\begin{aligned} f_{nx} &= \int_{t_{n-1}}^{t_n} \left( \frac{d^4 \sum_{j=0}^{N} x_{jn} t^j}{dt^4} \right)^2 dt \\ &= \int_{t_{n-1}}^{t_n} \left( \sum_{j=0}^{N} \frac{d^4 x_{jn} t^j}{dt^4} \right)^2 dt \\ &= \int_{t_{n-1}}^{t_n} \left( \sum_{j=4}^{N} h_j x_{jn} t^{j-4} \right)^2 dt \\ &= \int_{t_{n-1}}^{t_n} \sum_{j=4}^{N} \left( h_j x_{jn} t^{j-4} \right)^2 dt + 2 \int_{t_{n-1}}^{t_n} \sum_{j=4}^{N} \sum_{\ell=j+1}^{N} h_j x_{jn} h_\ell x_{\ell n} t^{j+\ell-8} dt \\ &= \sum_{j=4}^{N} h_j^2 x_{jn}^2 \left. \frac{t^{2j-7}}{2j-7} \right|_{t_{n-1}}^{t_n} + 2 \sum_{j=4}^{N} \sum_{\ell=j+1}^{N} h_j x_{jn} h_\ell x_{\ell n} \left. \frac{t^{j+\ell-7}}{j+\ell-7} \right|_{t_{n-1}}^{t_n} \ . \end{aligned} \tag{E.2}$$

For the terms $f_{ny}$ and $f_{nz}$ the algebraic manipulation is identical, the expansion of the term $f_{n\psi}$ is different from the obtained for the other three terms. Proceeding in the same fashion as in (E.2), let

$g_k = k(k-1)$,

$$
\begin{aligned}
f_{n\psi} &= \int_{t_{n-1}}^{t_n} \left( \frac{d^2 \sum_{j=0}^{N} \psi_{jn} t^j}{dt^2} \right)^2 dt \\
&= \int_{t_{n-1}}^{t_n} \left( \sum_{j=0}^{N} \frac{d^2 \psi_{jn} t^j}{dt^2} \right)^2 dt \\
&= \int_{t_{n-1}}^{t_n} \left( \sum_{j=2}^{N} g_j \psi_{jn} t^{j-2} \right)^2 dt \\
&= \int_{t_{n-1}}^{t_n} \sum_{j=2}^{N} \left( g_j \psi_{jn} t^{j-2} \right)^2 dt + 2 \int_{t_{n-1}}^{t_n} \sum_{j=2}^{N} \sum_{\ell=j+1}^{N} g_j \psi_{jn} g_\ell \psi_{\ell n} t^{j+\ell-4} dt \\
&= \sum_{j=2}^{N} g_j^2 \psi_{jn}^2 \left. \frac{t^{2j-3}}{2j-3} \right|_{t_{n-1}}^{t_n} + 2 \sum_{j=2}^{N} \sum_{\ell=j+1}^{N} g_j \psi_{jn} g_\ell \psi_{\ell n} \left. \frac{t^{j+\ell-3}}{j+\ell-3} \right|_{t_{n-1}}^{t_n} .
\end{aligned}
\tag{E.3}
$$

The final expressions in (E.2) and (E.3) confirm that the cost function $f$ is a quadratic $c^T \mathcal{H} c + \boldsymbol{w}^T c$, with $\boldsymbol{w}$ a vector of $F$ zeros, as these expressions are only composed of quadratic terms.

So as to build the matrix $\mathcal{H}$, the $F = 4(N+1)M$ polynomial coefficients of the trajectory are stacked in vector $c$. This vector can be seen as the composition of four other vectors which include the polynomial coefficients, ordered by interval number and degree, for each of the flat outputs $x,\ y,\ z,\ \psi$. The first $(N+1)M$ entries correspond to the coefficients of $x$, the next $(N+1)M$ entries correspond to the coefficients of $y$, and so on.

In order to write the expressions (E.2) and (E.3) in matrix form, the matrices $\boldsymbol{H}_n$ and $\boldsymbol{H}_n^\psi$ are defined. Both have dimensions $(N+1) \times (N+1)$ and the subscript $n$ indicates that these matrices are related to the $n^{\text{th}}$ interval of the trajectory. The constant $T_n$ denotes the duration of the trajectory segment $n$, that is, $T_n = t_n - t_{n-1}$.

$$
\boldsymbol{H}_n = \left[ \begin{array}{c|cccc}
\multicolumn{5}{c}{\mathbf{0}_{4\times(N+1)}} \\
\hline
 & h_4^2 T_n & 2h_4 h_5 T_n^2/2 & 2h_4 h_6 T_n^3/3 & \cdots & 2h_4 h_N T_n^{N-3}/(N-3) \\
 & 0 & h_5^2 T_n^3/3 & 2h_5 h_6 T_n^4/4 & \cdots & 2h_5 h_N T_n^{N-2}/(N-2) \\
\mathbf{0}_{(N-3)\times 4} & 0 & 0 & h_6^2 T_n^5/5 & \cdots & 2h_6 h_N T_n^{N-1}/(N-1) \\
 & \vdots & \vdots & & \ddots & \vdots \\
 & 0 & 0 & 0 & & h_N^2 T_n^{2N-7}/(2N-7)
\end{array} \right]
$$

$$\boldsymbol{H}_n^{\psi} = \left[\begin{array}{c:cccc} \multicolumn{5}{c}{\mathbf{0}_{2\times(N+1)}} \\ \hdashline \multirow{4}{*}{$\mathbf{0}_{(N-1)\times 2}$} & g_2^2 T_n & 2g_2 g_3 T_n^2/2 & 2g_2 g_4 T_n^3/3 & \cdots & 2g_2 g_N T_n^{N-1}/(N-1) \\ & 0 & g_3^2 T_n^3/3 & 2g_3 g_4 T_n^4/4 & \cdots & 2g_3 g_N T_n^N/N \\ & 0 & 0 & g_4^2 T_n^5/5 & \cdots & 2g_4 g_N T_n^{N+1}/(N+1) \\ & \vdots & \vdots & & \ddots & \vdots \\ & 0 & 0 & 0 & & g_N^2 T_n^{2N-3}/(2N-3) \end{array}\right].$$

The expression achieved in (E.1) motivates the definition of the matrices $\boldsymbol{H}$ and $\boldsymbol{H}^{\psi}$ from the 'template' matrices $\boldsymbol{H}_n$ and $\boldsymbol{H}_n^{\psi}$. These two new matrices are $(N+1)M \times (N+1)M$ diagonal block matrices,

$$\boldsymbol{H} = \mu_p \begin{bmatrix} \boldsymbol{H}_1 & & & \\ & \boldsymbol{H}_2 & & \\ & & \ddots & \\ & & & \boldsymbol{H}_M \end{bmatrix} \qquad \boldsymbol{H}^{\psi} = \mu_{\psi} \begin{bmatrix} \boldsymbol{H}_1^{\psi} & & & \\ & \boldsymbol{H}_2^{\psi} & & \\ & & \ddots & \\ & & & \boldsymbol{H}_M^{\psi} \end{bmatrix}.$$

As the sum of all $f_{nx}$, $f_{ny}$, and $f_{nz}$ can be obtained from $\boldsymbol{H}$, and $\boldsymbol{H}^{\psi}$ provides the sum of all $f_{n\psi}$, the matrix $\mathcal{H}$ is such that given the vector of the coefficients $\boldsymbol{c}$ sums the summations of the terms associated with each of the flat outputs for all trajectory segments,

$$\mathcal{H} = \begin{bmatrix} \boldsymbol{H} & & & \\ & \boldsymbol{H} & & \\ & & \boldsymbol{H} & \\ & & & \boldsymbol{H}^{\psi} \end{bmatrix}.$$

This matrix has dimensions $F \times F$, as expected when provided the dimensions of $\boldsymbol{c}$.

The next step in order to demonstrate the optimisation problem may be reformulated as a QP is to ensure that the equality constraints are linear in $\boldsymbol{c}$.

The waypoint constraints are linear in terms of the polynomial coefficients as

$$\boldsymbol{\sigma}(t_n) = \sum_{j=0}^{N} \boldsymbol{\sigma}_{jn} t_n^j = \boldsymbol{w}_n,$$

since the flat outputs are developed using polynomial functions.

The smoothness constraints in (4.5) are also expressed as a linear map. The continuity of trajectory between the segments $\boldsymbol{\sigma}_n$ and $\boldsymbol{\sigma}_{n+1}$ at $t_n$ requirement is written as a set of affine constraints,

$$\boldsymbol{\sigma}_n(t_n) = \boldsymbol{\sigma}_{n+1}(t_n) \Leftrightarrow \sum_{j=0}^{N} \left(\boldsymbol{\sigma}_{jn} - \boldsymbol{\sigma}_{j(n+1)}\right) t_n^j = \mathbf{0}_{4\times 1}.$$

Similarly, the constraints imposing the continuity of the $m^{\text{th}}$ derivative of the flat outputs is linear,

$$\left.\frac{d^m \boldsymbol{\sigma}_n}{dt^m}\right|_{t=t_n} = \left.\frac{d^m \boldsymbol{\sigma}_{n+1}}{dt^m}\right|_{t=t_n} \Leftrightarrow \sum_{j=m}^{N} \left(\boldsymbol{\sigma}_{jn} - \boldsymbol{\sigma}_{j(n+1)}\right) t_n^{j-m} \prod_{\ell=0}^{m-1} (j-\ell) = \mathbf{0}_{4\times 1}.$$

The constraints imposing a desired linear velocity at $t = t_n$ are also linear in the polynomial coefficients $\sigma_{jn}$,

$$\left.\frac{d\boldsymbol{\sigma}}{dt}\right|_{t=t_n} = \sum_{j=1}^{N} j\boldsymbol{\sigma}_{jn} t_n^{j-1} = \boldsymbol{v}_n,$$

so are those which regard the aircraft finishing at hover flight,

$$\left.\frac{d^2\boldsymbol{\sigma}}{dt^2}\right|_{t=t_M} = \sum_{j=1}^{N} j(j-1)\boldsymbol{\sigma}_{jM} t_M^{j-2} = \mathbf{0}_{4\times 1}.$$

Hence, the constraints in (4.5) can be rewritten as $\boldsymbol{Ac} = \boldsymbol{b}$. The matrix $\boldsymbol{A}$ has dimensions $(22M - 14) \times F$. Out of the $22M - 14$ constraints, $4(M+1)$ account for the waypoint constraints, $15(M-1)$ correspond to the continuity constraints on the position and on its four first derivatives (velocity, acceleration, jerk, and snap), $3(M-1)$ are on the yaw angle and on its two first derivatives. Another eight constraints may be imposed on the initial and final velocity, and another four which require that the acceleration at the end of the trajectory is zero.

Finally, the convexity of the problem (4.6) is confirmed as $\mathcal{H} \succcurlyeq 0$. One starts by examining the $\boldsymbol{H}_n$ matrix and evaluating its eigenvalues. If the first four rows and columns are removed from this matrix, one can verify that the resulting sub-matrix has positive eigenvalues using the Sylvester's Criterion. As the deleted rows and columns are filled with zeros, $\boldsymbol{H}_n$ has four zero eigenvalues, and the remaining $(N-4)$ are positive. By the same reasoning, all of the $N$ eigenvalues of $\boldsymbol{H}_n^{\psi}$, except for two which are equal to zero, are positive. Since $\mathcal{H}$ is composed of these matrices and zero block matrices, it is concluded that all of its eigenvalues are non-negative.