Università degli Studi di Padova

Department of Information Engineering

Master Thesis in Automation Engineering

Adaptive non-Linear Model Predictive Control for UAV-UGV Coordination

Supervisor
Prof. Angelo Cenedese

Master Candidate
Daniele Barnabò

20 October 2022
Academic Year 2021/2022

Abstract

Unmanned Aerial Vehicles (UAV) have gained a great amount of popularity in the last years. Among the features that make them so successful are the cost, the portability, their airborne nature and their ability to perform Vertical Take-Off and Landing (VTOL). Since the different tasks that UAVs are given may require them to travel long distances, the small batteries that they are using can prove to be a problem if they have to return to the home-base to recharge. A solution can be sending a Unmanned Ground Vehicle (UGV) along with the UAVs, since they can have a much longer autonomy and provide a moving base.
In particular this thesis wants to improve the landing manoeuvre in such a way that the landing will be smooth and most accurate. The first part of this work is dedicated the implementation of the Non-linear Model Predictive Control (NMPC) as the baseline controller. This controller creates an optimal control that minimizes a given cost function along the chosen prediction horizon, while respecting the model constraints. For this work the non-linear version of the algorithm was chosen given the non-linearity of the model of the UAV in question, i.e. the quadrotor. One of the main features of this controller is the dependency on the internal model of the system, which gives it its name.
In the second part another complementary controller is added to the NMPC. Adaptive control is meant to deal with varying or unknown model parameters and for this reason $\mathcal{L}1$ Adaptive Control (L1AC) was chosen for the task. L1AC is a slight variation of the Model Reference Adaptive Control (MRAC), which tries to deal with errors at the input level, setting a new input counterpart to negate them.
The combination of the two controllers is meant to exploit each other's strength, while helping with the other's weakness, to create an overall stable and well performing algorithm that allows to account for the kinematics and dynamics of the quadrotor, while being extremely robust against the potential errors of the internal model or external disturbances.
The proposed method is then tested in various scenarios and validated through results on Matlab & Simulink.

Keywords: UAV, UGV, Quadrotor Landing, Nonlinear Model Predictive Control, $\mathcal{L}1$ Adaptive Control, Ground Effect

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 State of the art

Unmanned Aerial Vehicles (UAV) have gained attention lately and appear to have secured a place in our future [1]. Among them, quadrotors are a popular example thanks to their simplicity and ability to perform vertical take-off and landing (VTOL) [2].
UAVs can be used for multiple applications, from patrolling and mapping tasks [3][4] to carrying actual objects across various distances [5]. These tasks can also be performed in autonomy by the UAVs, but one of the problems that arise is the battery life of the drone, which typically does not last long since the UAV has to sustain its own weight [6].
Having a Unmanned Ground Vehicle (UGV) that moves along the UAVs could prove to be extremely helpful. In fact, the drones could have a moving base on which to descend once the batteries are empty or any kind of other problem occurs [7], instead of having to return to a fixed position like a home base [8]. In this type of operation, precision in the landing procedure plays an extremely important role since it could help fitting multiple quadrotors onto a relatively small surface. In an extreme case, good tracking precision could be used to make a UAV land in a charging device, situated on the UGV [9].
One way to achieve such precision is to utilize *Model Predictive Control* (MPC) [10] [11] [12], which exploits the knowledge of the system. MPC is a control method that predicts the future states of a system based on an internal model it is given. It then creates an optimal path according to some error measurements and a set of weights for these errors, which usually represent the difference between reference and actual state. This kind of control, which is also called *Receding Horizon Estimation and Control* and *Moving Horizon Optimal Control*, has been used to confront a variety of problems [13] [14] [15]. MPC has been successful and adopted in a variety of problems since it allows to easily include constraints in the controller formulation and in the last years it proved to be an

excellent tool both according to theory and in practice [16]. This kind of linear controller, however, cannot be used with most systems since usually they are non-linear because of kinematic/dynamic characteristics of the plant or due to its components (sensors, actuators, etc...). For this reason *Non-linear Model Predictive control* (NMPC) was introduced. It has the capability of capturing the non-linearities of the plant [17] and thanks to the recent development of embedded systems with computing capabilities it has become extremely popular in general and in particular with UAVs, due to the non-linearities that are intrinsic to the model.

NMPC has been used by a great number of researchers for quadrotor control [18] [19] [20], but this thesis contribution is the addition of *Adaptive Control*. Adaptive control covers a set of techniques that provide a systematic approach for automatic adjustment of controllers in real time, in order to achieve or maintain a desired level of control system performance when the parameters of the plant dynamic model are unknown and/or change in time [21]. This kind of control can be used either when the parameters of the system are constant and unknown or when they are time-varying and unpredictable[22]. Throughout the last six decades adaptive control has evolved and has gradually given results about its stability both in continuous and discrete time [23] [24], in systems that may be non-linear [25] or with bounded input [26].

In particular this thesis focuses on *L1 Adaptive Control* (L1AC), which is a variation of *Model Reference Adaptive Control* (MRAC). MRAC is based on the fact that the model of the plant is known and that correction of the parameter estimation can be performed by looking at the prediction error of the states. This creates a feedback control law that tries to give the plant output a performance similar to the reference output. This kind of control has been used extensively for quadrotor control [27] [28] and has seen successful results in the improvement over other control methods. L1AC differs from MRAC in the fact that the adaptive controller works at a higher frequency than the base controller. This is meant to reduce the oscillations that may occur otherwise. L1AC extends the state-predictor-based MRAC and handles the trade-off between performance and robustness with fast adaptation in a transparent and effective way [29]. Another way to define the L1AC is a MRAC with a low-pass filter acting on the control input [30]. This kind of approach, applied to quadrotor and octarotor control, has encountered a lot of success in the recent years [31] [32] [33].

1.2 Thesis structure

The thesis is organized in the following manner:

- In chapter 2 we provide the preliminary notions about the 3D pose of a rigid body using both the rotation matrices notation and the quaternions notation. Then, kinematics and the dynamics of the quadrotor are examined to derive its mathematical model.
- In chapter 3, a brief presentation of the model predictive control principles is presented. Then, the problem is formalized and all the assumptions used for this work are defined.
- In chapter 4 contains the controller formulation and the problem simulation. Cost functions, constraints and agent architectures and environment effect are defined. Then, adaptive control is introduced with the relative theory applied to the quadrotor model.
- Chapter 5 reports the results obtained. First, simple simulation results are presented to validate the controller and model. Then more interesting simulations are used to test the actual effectiveness of the architecture.
- Finally, in chapter 6 all the main results are summarized and suggestions for future works are given.

Chapter 2

Preliminary and agent model

2.1 Pose of a rigid body

A rigid body can be described by its position and orientation, which are always measured with respect to a reference frame and they define the pose of the object. If we call $\mathfrak{F}_w$ the reference or world frame found in the orthonormal base $\boldsymbol{x}\ \boldsymbol{y}\ \boldsymbol{z}$, then the pose of the object is fully described by the new body frame $\mathfrak{F}_b$, as shown in *Figure* 2.1. Here we can see that the position corresponds to

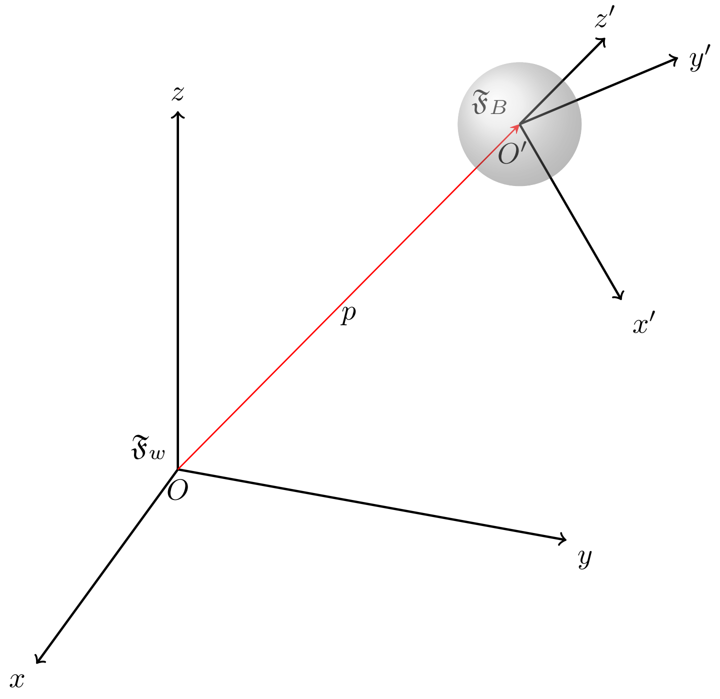


Figure 2.1: Position and orientation of a rigid body with respect to the world frame.

the coordinates of the body in the world frame $\mathfrak{F}_w$, but the orientation of the body needs a new tool to be described. To this end we can introduce *rotational matrices*. A rotational matrix $\boldsymbol{R}$ is a matrix whose columns are the coordinates of the body frame axis $\boldsymbol{x}'\ \boldsymbol{y}'\ \boldsymbol{z}'$ expressed with respect to the world frame $\mathfrak{F}_w$.

There are important properties that can be assigned to rotation matrices, among which we can find *orthogonality*. A matrix in general can be called *orthogonal*, if

$$\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_3 \tag{2.1.1}$$

where $\boldsymbol{I}_3$ represents the 3-by-3 identity matrix. From *Equation* 2.1.1 we can find, by multiplying on both sides by $\boldsymbol{R}^{-1}$, that

$$\boldsymbol{R}^T = \boldsymbol{R}^{-1} \tag{2.1.2}$$

This tells us that to find the inverse of a rotational matrix we only have to transpose it. In particular we say that if $\det(\boldsymbol{R}) = 1$ we are using the right-hand-rule, if $\det(\boldsymbol{R}) = -1$ the left-hand-rule. Although equivalent, generally the right-hand-rule is used and matrices whose $\det(\boldsymbol{R}) = 1$ represent the *Special Orthogonal* group $\mathbb{SO}(3)$, while those with $\det(\boldsymbol{R}) = -1$ can be interpreted as improper rotations, meaning that there is also a reflection along with the rotation.
As said before, a rotation matrix columns are the coordinates of the body frame expressed in the world frame. Each of these rotations can be expressed in a combination of rotations around the axis, also called *elementary rotations* and they are positive when in counter-clockwise direction around the axis. As an example we can look at a rotation around the x axis, by the angle α, from the frame $\mathfrak{F}_w$ to the new frame $\mathfrak{F}_{w'}$. Then $\boldsymbol{R}_x$ can be calculated in the following manner:

$$\boldsymbol{R}_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \tag{2.1.3}$$

We can notice that all points on the x axis do not change, while the rest change values on the remaining plane according to the value of the angle α. With the same process we can calculate the other two elementary rotations by the angles β and γ:

$$\boldsymbol{R}_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{bmatrix} \tag{2.1.4}$$

$$\boldsymbol{R}_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.1.5}$$

With quick look at the rotation matrices 2.1.3-2.1.5 we can check that the property 2.1.2 holds, as the inverse of a rotation is a rotation in the opposite

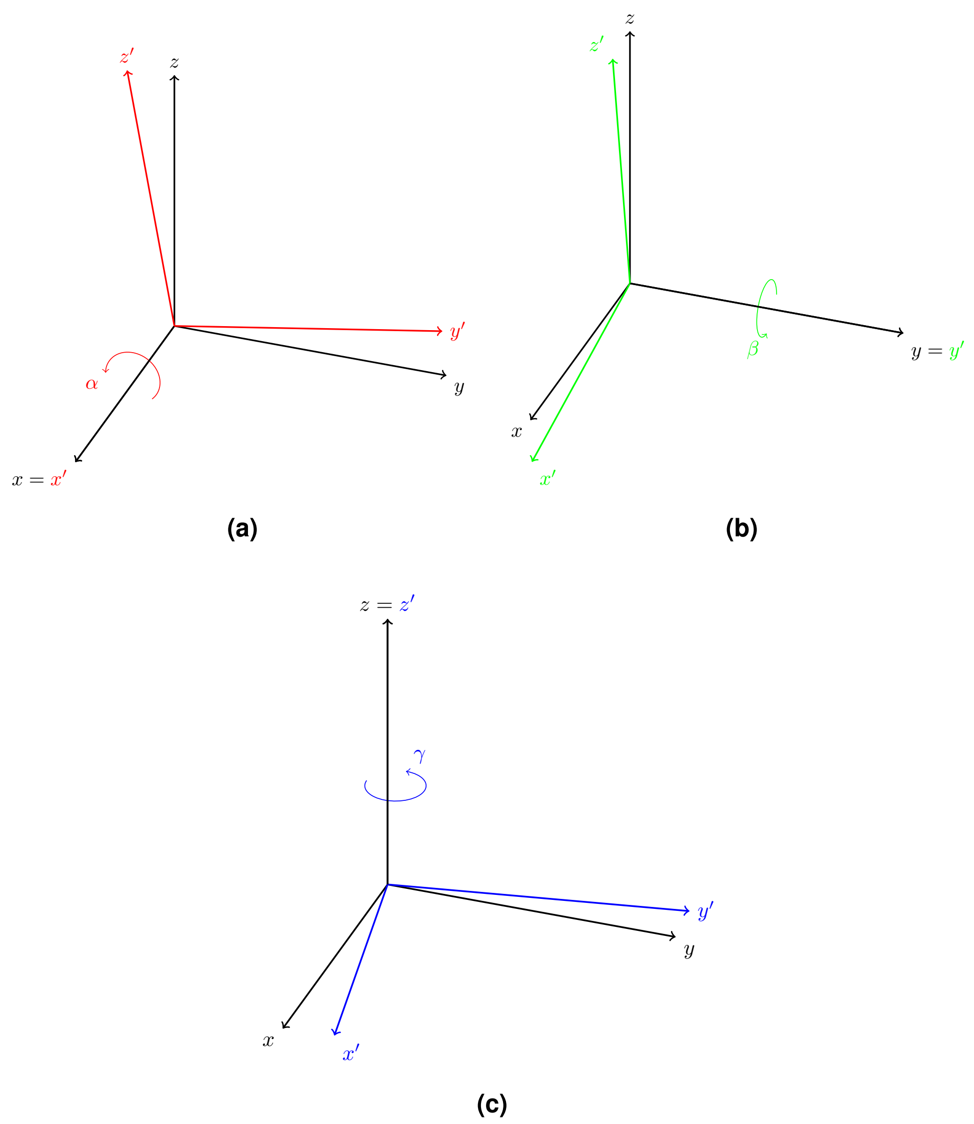


Figure 2.2: Elementary rotations about the coordinate axes. (a) rotation by an angle α about $\boldsymbol{x}$ axis, (b) rotation by an angle β about $\boldsymbol{y}$ axis and (c) rotation by an angle γ about $\boldsymbol{z}$ axis.

direction and

$$\boldsymbol{R}(-\theta) = \boldsymbol{R}^T(\theta) \tag{2.1.6}$$

Since the rotation matrix is used to align a new frame $\mathfrak{F}_{w'}$ with the reference frame $\mathfrak{F}_w$, it can also be used to express a point in the new frame p' in the reference frame p. This is accomplished by pre-multiplying p' by $\boldsymbol{R}$

$$\boldsymbol{p} = \boldsymbol{R}\boldsymbol{p}'. \tag{2.1.7}$$

Here we can only go from the new frame $\mathfrak{F}_{w'}$ to the reference frame $\mathfrak{F}_w$, however if we pre-multiply by $\boldsymbol{R}^{-1}$ on both sides we get

$$\boldsymbol{R}^{-1}\boldsymbol{p} = \boldsymbol{R}^{-1}\boldsymbol{R}\boldsymbol{p}' \tag{2.1.8}$$

$$\boldsymbol{p}' = \boldsymbol{R}^T\boldsymbol{p} \tag{2.1.9}$$

remembering from 2.1.2 that $\boldsymbol{R}^{-1} = \boldsymbol{R}^T$.
If more frames are considered for different rotations, it is possible to derive a total rotation starting from the single ones. Let's suppose that there are three frames $\mathfrak{F}_w0$, $\mathfrak{F}_w1$, $\mathfrak{F}_w2$ that have no translation between them, meaning they are placed on the same origin point $\boldsymbol{O}$, we can call the rotations to any frame i from another frame j, $\boldsymbol{R}^i_j$. Any point p^0, p^1 and p^2 are related by the following:

$$\boldsymbol{p}^0 = \boldsymbol{R}^0_1\boldsymbol{p}^1 \tag{2.1.10}$$

$$\boldsymbol{p}^1 = \boldsymbol{R}^1_2\boldsymbol{p}^2 \tag{2.1.11}$$

From these two relationships we can derive the third equation

$$\boldsymbol{p}^0 = \boldsymbol{R}^0_1\boldsymbol{R}^1_2\boldsymbol{p}^2 \tag{2.1.12}$$

$$\boldsymbol{R}^0_2 = \boldsymbol{R}^0_1\boldsymbol{R}^1_2 \tag{2.1.13}$$

In the same manner any rotation can be expressed as a composition of multiple rotation, which is usually the case since it is intuitive and easier to decompose rotations in elementary rotations as we will see in the next section. An important fact to keep in mind is that rotations in $\mathbb{SO}(3)$ are not commutative and that pre-multiplication and post-multiplication have different meanings and may give different results.

2.1.1 Euler angles

Rotation matrices work well and can define any rotation in $\mathbb{SO}(3)$, however they need a lot of elements. If we look at a generic rotation in $\mathbb{SO}(m)$, then $m(m-1)/2$ parameters are needed to fully characterize it.In the case of $\mathbb{SO}(2)$ this works since one angle is enough to describe each rotation, which is less than the four values of a 2-dimensional rotation matrix. In the case of 3-dimensional rotations each matrix is composed of nine elements which are dependent from each other. The condition of orthogonality 2.1.1 bounds these values with six constraints, which leaves the parameters actually needed to represent an $\mathbb{SO}(3)$ rotation to just three. The following theorem can then be brought to our attention

Theorem 2.1.1 (Euler's rotation theorem). *A generic rotation matrix can be obtained by composing a suitable sequence of three elementary rotations while guaranteeing that two successive rotations are not made about parallel axes.*

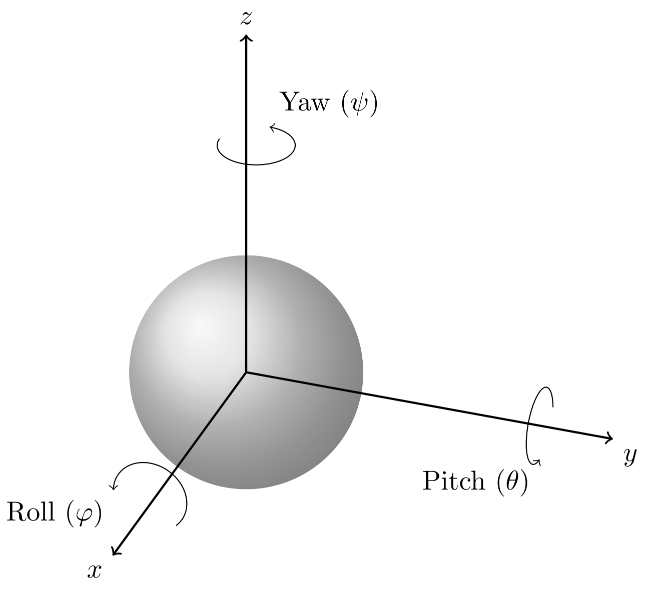


Figure 2.3: Representation of *Roll–Pitch–Yaw* angles along the respective $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ axis
.

The last part of the theorem gives us a condition that limits the total number of combinations of elementary rotations that can describe a generic rotation. In fact we have a free choice for the first one, while the second and third elementary rotations are limited to two possible axis. This gives us a total of 12 acceptable combinations, instead of 27 possible combinations. Since they are equivalent, from now on the only sequence that will be considered is x axis, y axis and z axis, which are called *Roll-Pitch-Yaw* angles respectively.

This is the format that is commonly used in the aircraft field to describe the orientation of airborne vehicles and the corresponding angles are $\boldsymbol{\Phi} = [\varphi\ \theta\ \psi]^T$, which describe the rotation around the fixed frame attached to the center of mass of the aircraft *Figure* 2.3.
The rotation can be decomposed in am elementary rotation by the angle φ around the x axis (roll), an elementary rotation by the angle θ around the y axis (pitch) and an elementary rotation by the angle ψ around the z axis (yaw). Finally the whole rotation cane be calculated, with respect to the reference fixed frame, by pre-multiplication of the corresponding rotation matrices, giving

$$\begin{aligned}\boldsymbol{R}(\Phi) &= \boldsymbol{R}_z(\psi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_x(\varphi) \\ &= \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\varphi + c_\psi s_\theta s_\varphi & s_\psi s_\varphi + c_\psi s_\theta c_\varphi \\ s_\psi c_\theta & c_\psi c_\varphi + s_\psi s_\theta s_\varphi & -c_\psi s_\varphi + s_\psi s_\theta c_\varphi \\ -s_\theta & c_\theta s_\varphi & c_\theta c_\varphi \end{bmatrix}\end{aligned} \tag{2.1.14}$$

where c and s are abbreviations for cosine and sine respectively. It is possible to derive the triplets of angles $\boldsymbol{\Phi}$ from $\boldsymbol{R}$. Too that end, we start from

$$\boldsymbol{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} . \tag{2.1.15}$$

A solution can always be found for values of $\theta \neq \pm\frac{\pi}{2}$, or $c_\theta \neq 0$ and it is

$$\begin{aligned} \varphi &= \text{atan2}(r_{32}, r_{33}) \\ \theta &= \text{asin}(-r_{31}) \\ \psi &= \text{atan2}(r_{21}, r_{11}) \end{aligned} \tag{2.1.16}$$

if $\theta = \pm\frac{\pi}{2}$ we can only determine the quantity $\varphi \pm \psi$ and not their single values. This is commonly known as *gimbal lock* in literature and will be important in the next section.

2.2 Unit quaternion

Although Euler angles perform well in describing rotations, they suffer from the previously mentioned gimbal lock. To avoid these problems, a new tool can be used, which is called **Hamilton's quaternion**. A quaternion is hypercomplex number used to represent SO(3) rotations. It is composed of a real and a complex part, but on the contrary of normal classic complex numbers, the complex part is composed by three numbers

$$\mathfrak{q} = \underbrace{\eta}_{real} + \underbrace{i\epsilon_i + j\epsilon_j + k\epsilon_k}_{complex} = \eta + \boldsymbol{\epsilon} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{2.2.1}$$

where (i, j, k) create the relationship between the elements of the quaternion, which follow Hamilton's rules

$$\begin{aligned} & i^2 = j^2 = k^2 = ijk = -1 \\ & ij = k \quad jk = i \quad ki = j \\ & ji = -k \quad kj = -i \quad ik = -j \end{aligned} \tag{2.2.2}$$

Any rotation can be represented with a quaternion, which only uses one parameter more than Euler angles. The different rotations are encoded in the following manner

$$\mathfrak{q} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} = \begin{bmatrix} \cos\frac{\theta}{2} \\ \boldsymbol{e}\sin\frac{\theta}{2} \end{bmatrix} \tag{2.2.3}$$

where $\boldsymbol{e}$ represents the axis around which the rotation is performed and θ the angle of the rotation around said axis. The quaternion $\mathbf{q}$ is unitary, meaning that

$$\|\mathbf{q}\|^2 = \eta^2 + \boldsymbol{\epsilon}^T\boldsymbol{\epsilon} = \cos^2\left(\frac{\theta}{2}\right) + \sin^2\left(\frac{\theta}{2}\right) = 1. \tag{2.2.4}$$

A tool we have to compute the rotation matrix $\boldsymbol{R} \in \mathbb{SO}(3)$ from a quaternion is the **Rodrigues rotation formula**. This gives us an efficient method to interchange the different representation, starting from the unit vector $\boldsymbol{\omega} \in \mathbb{R}^3$ and the angle of the rotation θ. The rotation matrix $\boldsymbol{R}(\theta, \boldsymbol{\omega})$ is then given by

$$\boldsymbol{R}(\theta, \boldsymbol{\omega}) = \exp\left([\boldsymbol{\omega}]_x\right) = \boldsymbol{I}_3 + \frac{\sin\theta}{\theta}[\boldsymbol{\omega}]_x + \frac{1-\cos\theta}{\theta^2}[\boldsymbol{\omega}]_x^2 \tag{2.2.5}$$

where $[\boldsymbol{\omega}]_x$ is the skew-symmetric matrix for the vector $\boldsymbol{\omega}$ and is calculated in the following manner

$$[\boldsymbol{\omega}]_x = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{2.2.6}$$

To relate unit quaternion and rotation matrix directly we can use the following formula

$$\begin{aligned} \boldsymbol{R}(\mathbf{q}) &= \boldsymbol{I}_3 + 2\eta[\boldsymbol{\epsilon}]_x + 2[\boldsymbol{\epsilon}]_x^2 \\ &= \begin{bmatrix} \eta^2 + \epsilon_i^2 - \epsilon_j^2 - \epsilon_k^2 & 2(\epsilon_i\epsilon_j - \eta\epsilon_k) & 2(\epsilon_i\epsilon_k + \eta\epsilon_j) \\ 2(\epsilon_i\epsilon_j + \eta\epsilon_k) & \eta^2 - \epsilon_i^2 + \epsilon_j^2 - \epsilon_k^2 & 2(\epsilon_j\epsilon_k - \eta\epsilon_i) \\ 2(\epsilon_i\epsilon_k - \eta\epsilon_j) & 2(\epsilon_j\epsilon_k + \eta\epsilon_i) & \eta^2 - \epsilon_i^2 - \epsilon_j^2 + \epsilon_k^2 \end{bmatrix} \end{aligned} \tag{2.2.7}$$

where we combined the fact that $\theta = \|\boldsymbol{\omega}\|$ and the information in *Equation* (2.2.3). In the last equation we can notice that a quaternion $\mathbf{q}$ and its opposite $-\mathbf{q}$ represent the same rotation $\boldsymbol{R}(\mathbf{q})$. This property is called *double coverage*. The relationship that goes from quaternion to Euler angles can be derived substituting 2.2.7 into 2.1.16, obtaining

$$\begin{aligned} \varphi &= \operatorname{atan2}(2\epsilon_j\epsilon_k + 2\eta\epsilon_i, \eta^2 - \epsilon_i^2 - \epsilon_j^2 + \epsilon_k^2) \\ \theta &= \operatorname{asin}(2\eta\epsilon_j - 2\epsilon_i\epsilon_k) \\ \psi &= \operatorname{atan2}(2\epsilon_i\epsilon_j + 2\eta\epsilon_k, \eta^2 + \epsilon_i^2 - \epsilon_j^2 - \epsilon_k^2) \end{aligned} \tag{2.2.8}$$

2.2.1 Advantages of unit quaternion notation

Quaternions and Euler angles are both valid and commonly used methods to describe rotations in $\mathbb{R}^3$, however, as it was explained in the previous section, Euler angles suffer from the problem of the gimbal lock. Besides this, they need to work together with the respective rotation matrices, which contain nine parameters with six constraints that are due to the orthonormality that is implicit of the representation (i.e. $\boldsymbol{R}^T\boldsymbol{R} = \boldsymbol{I}_3$) and the additional constraint $\det \boldsymbol{R} = +1$ due to the simple rotation that does not involve reflections. Finally, due to numerical problems it is possible to have problems in both the orthonormality of the rotation matrix when performing matrix pre or post-multiplication as well as having problems the quaternion norm. The quaternion is again the more convenient choice since it is much easier to correct it and make it unitary, rather than correcting the orthogonality of the rotation matrix.

2.3 Agent model

At this point we want to give a definition of the quadrotor model and dynamics. From these we will define the controller in the later sections. The quadrotor model that will be used is the one described in [34]. In particular in this thesis the quadrotor model will have the parameters of the Crazyflie Nano Quadcopter, which is shown in *Figure* 2.4, a small quadrotor that is found in [35]. This was just one of the possible choices and did not have a special reason behind it, since our attention is elsewhere. Quadrotors are under-actuated

Figure 2.4: Picture of a Crazyflie Nano Quadcopter. This image was taken from their website.

by nature,since they are not able to apply instantaneous acceleration in any direction, but just vertically. To be fully actuated the quadrotor would have to be able to deal with all 6 *degrees of freedom* (DoFs), which are composed of 3 translational DoFs and 3 rotational DoFs, but only has 4 control inputs. The classic setup of a quadrotor is composed of a rigid cross-shaped frame with four brush-less DC (BLDC) motors, as shown in *Figure* 2.5. The rotors impart

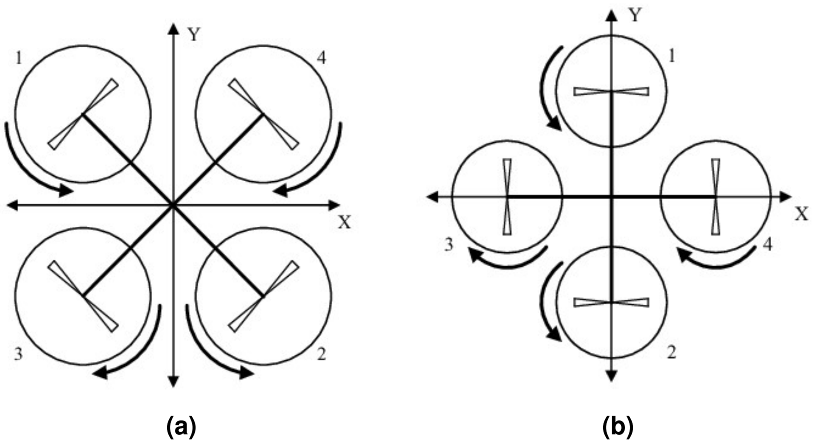


Figure 2.5: The main two configurations of quadrotor: (a) the cross-configuration, where the rotors point at a 45° angle with respect to the x and y axis and (b) the plus-configuration, where the rotors point in the same direction of the x and y axis.

the forces and moments necessary to move the quadrotor with agility and usually have one of two configurations. What differentiates them is the angle the reference axis make with the rotors position. In the first one, shown in *Figure* 2.5a, is called cross-configuration, where the axis make a 45° angle with the rotors position; the second one, in *Figure* 2.5b, is called plus-configuration and has the rotor and the axis pointing in the same direction. The main difference between the two is that in the first one has all rotors active at most times, making it more stable, while the latter is more common and less stable and is generally used for acrobatic tasks.
Quadrotors are highly unstable by nature. This is due to unstable dynamics that are caused by the variations in the quadrotor parameters. This can be associated with environmental reasons (e.g. the wind). Disturbances play an important role as well since there is a strong coupling between the DoFs and there is a lack of damping in the vehicle.

2.3.1 Dynamic Model

The dynamical model of the quadrotor is usually derived with one of two methods: the *Euler-Lagrange Formalism* or the *Newton-Euler Formalism*. The former a less intuitive energy based formulation that makes use of the *Euler-Lagrange Equation*; the latter instead, is a simpler to understand approach that

works directly on the forces and moments. They are both equally valid approaches, so we will use the *Newton-Euler Formalism*. In particular in the next paragraph the attitude model will be derived only using the rotation matrices first, to then move to quaternions.
We define as $\mathfrak{F}_w$ the inertial world frame and as $\mathfrak{F}_B$ the body frame of the quadrotor, centered in the center of mass (CoM) O_B of the platform. We then call $\boldsymbol{p} \in \mathbb{R}^3$ the position of the CoM O_B with respect to the world frame $\mathfrak{F}_w$. To describe the orientation of the quadrotor with respect to the world frame we use the rotation matrix $\boldsymbol{R} \in \mathbb{SO}(3)$. Now the pair $(\boldsymbol{p}, \boldsymbol{R}) \in \mathbb{R}^3 \times \mathbb{SO}(3)$ fully describe the pose of the quadrotor. The next pair necessary to describe the dynamics of the quadrotor is $(\boldsymbol{v}, \boldsymbol{\omega})$, where $\boldsymbol{v} \in \mathbb{R}^3$ is the velocity of the CoM in $\mathfrak{F}_w$ and $\boldsymbol{\omega} \in \mathbb{R}^3$ is the angular velocity of the platform in the inertial frame $\mathfrak{F}_w$. The kinematics of the quadrotor are then

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{2.3.1a}$$
$$\dot{\boldsymbol{R}} = \boldsymbol{R}[\boldsymbol{\omega}]_\times \tag{2.3.1b}$$

where the last equation is derived in *Section* (A.1).
Looking at the dynamics, the equations are derived from the forces and torques that are applied to the quadrotor from the single propellers. Each one rotates around its own axis with a spinning rate $\omega_i \in \mathbb{R}$, which can be either working in a *clockwise* manner (CW), or in a *counter-clockwise* one (CCW). In the first case the angular velocity of the propeller would be $-\omega_i \boldsymbol{e}_{z_i}$, while in the second one it would be $\omega_i \boldsymbol{e}_{z_i}$. These propellers are alternated in such a way that two of the same kind lay on opposite sided of the platform. If we call the control input $\Omega_i = \omega_i |\omega_i| \in \mathbb{R}$, then the forces created by each propeller are

$$\boldsymbol{f}_i = c_{f_i} \Omega_i \boldsymbol{e}_{z_i} \tag{2.3.2}$$

where $c_{f_i} \in \mathbb{R}^+$ is a constant parameter, meant to represent the thrust coefficient of the i-th propeller. The moments $\boldsymbol{\tau}_i^t \in \mathbb{R}^3$ associated to each propeller are generated from these forces and are found through the following equation

$$\boldsymbol{\tau}_i^t = \boldsymbol{p}_i \times \boldsymbol{f}_i = c_{f_i} \boldsymbol{p}_i \times \boldsymbol{e}_{z_i} \tag{2.3.3}$$

The rotation of each propeller causes a drag moment, called $\tau_i^d \in \mathbb{R}^3$, that points in the opposite direction with respect to the angular velocity. We can find its value in $\mathfrak{F}_B$ through the equation

$$\boldsymbol{\tau}_i^d = c_{\tau_i} \Omega_i \boldsymbol{e}_{z_i} \tag{2.3.4}$$

where $c_{\tau_i} \in \mathbb{R}$ is a constant parameter, meant to represent the drag coefficient of the i-th propeller. It is positive if the rotor is spinning in a CW direction, negative otherwise.
Looking at the the *Equations* 2.3.2-2.3.4, we can finally deduce the total force $\boldsymbol{f}_c \in \mathbb{R}^3$ and total torque $\boldsymbol{\tau}_c \in \mathbb{R}^3$ that act on the CoM of the quadrotor

$$\boldsymbol{f}_c = \sum_{i=1}^{4} \boldsymbol{f}_i = \sum_{i=1}^{4} c_{f_i} \Omega_i \boldsymbol{e}_{z_i} \tag{2.3.5a}$$

$$\boldsymbol{\tau}_c = \sum_{i=1}^{4} \boldsymbol{\tau}_i^t + \boldsymbol{\tau}_i^d = \sum_{i=1}^{4} (c_{f_i} \boldsymbol{p}_i \times \boldsymbol{e}_{z_i} + c_{\tau_i} \boldsymbol{e}_{z_i}) \Omega_i \tag{2.3.5b}$$

where the quantities are defined with respect to the body frame $\mathfrak{F}_B$. All this forces and moments can be observed in *Figure* 2.6.

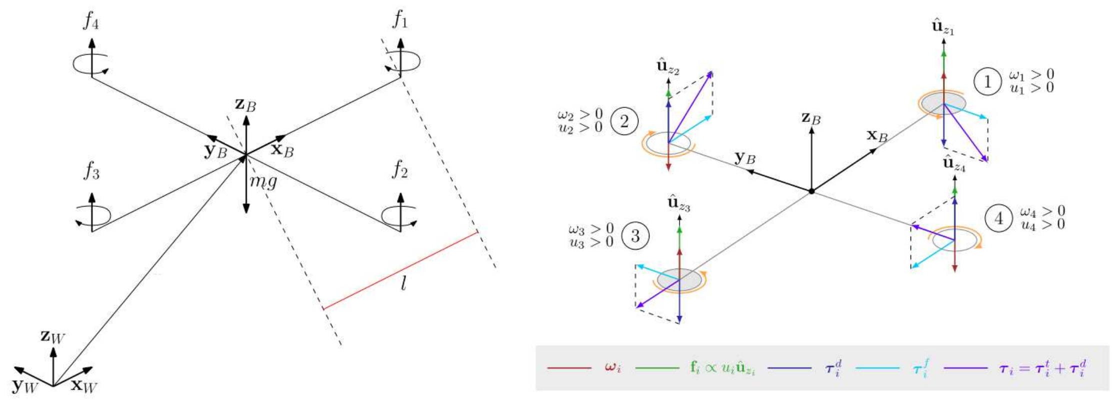


Figure 2.6: Schematic picture of a quadrotor, where the forces and moments in action are highlighted. Particular attention is given to the direction of all these components.

We can relate the force and torque we just found to the propeller input $\boldsymbol{\Omega} = [\Omega_1 \Omega_2 \Omega_3 \Omega_4]^T$ in the following manner

$$\boldsymbol{f}_c = \boldsymbol{F}\boldsymbol{\Omega} \tag{2.3.6a}$$

$$\boldsymbol{\tau}_c = \boldsymbol{M}\boldsymbol{\Omega} \tag{2.3.6b}$$

where l the length of the quadrotor arm, while $\boldsymbol{F} \in \mathbb{R}^{3\times 4}$ and $\boldsymbol{M} \in \mathbb{R}^{3\times 4}$ are the matrices defines as

$$\boldsymbol{F} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ c_{f_1} & c_{f_2} & c_{f_3} & c_{f_4} \end{bmatrix} \qquad \boldsymbol{M} = \begin{bmatrix} 0 & lc_{f_2} & 0 & -lc_{f_4} \\ -lc_{f_1} & 0 & lc_{f_3} & 0 \\ -|c_{\tau_1}| & |c_{\tau_2}| & -|c_{\tau_3}| & |c_{\tau_4}| \end{bmatrix} \tag{2.3.7}$$

This is a simpler model that does not take into consideration more complex effect that may happen in a more realistic flight simulation. Most of them would only cause minor perturbations in a robotic system, but an aircraft such as a quadrotor is more vulnerable to these effects. In this thesis these second order effects are neglected for simplicity sake, however other effects such as ground effect will be explored in the later sections. For now we will limit to following the system of *Newton-Euler* equations

$$m\ddot{\boldsymbol{p}} = -mg\boldsymbol{e}_3 + \boldsymbol{R}\boldsymbol{f}_c \tag{2.3.8a}$$

$$\boldsymbol{J}\dot{\boldsymbol{\omega}} = -\omega \times \boldsymbol{J}\omega + \boldsymbol{\tau}_c \tag{2.3.8b}$$

where $g > 0$, $m > 0$ and $\boldsymbol{J} \in \mathbb{R}^{3\times3}$ are, respectively, the gravitational acceleration and the drone total mass and inertia. $\boldsymbol{R}$ is the rotation matrix embedded in the drone pose and $\boldsymbol{e}_3 \in \mathbb{R}^3$ is the vertical direction, which corresponds to the third component of the world frame $\mathfrak{F}_w$.
If we then combine the kinematic equations (2.3.1b) and the dynamics equations (2.3.8), we get a full description of the quadrotor model

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{2.3.9a}$$

$$\dot{\boldsymbol{R}} = \boldsymbol{R}[\boldsymbol{\omega}]_\times \tag{2.3.9b}$$

$$\dot{\boldsymbol{v}} = -g\boldsymbol{e}_3 + m^{-1}\boldsymbol{R}\boldsymbol{f}_c \tag{2.3.9c}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(-\omega \times \boldsymbol{J}\omega + \boldsymbol{\tau}_c) \tag{2.3.9d}$$

Remembering that the only control inputs are $\boldsymbol{f}_c$ and $\boldsymbol{\tau}_c$, we can notice how the rotational dynamics 2.3.9d have an effect on the translational dynamics 2.3.9c through the rotational kinematics 2.3.9b. This kind of coupling can be better understood by looking at *Figure* 2.7.

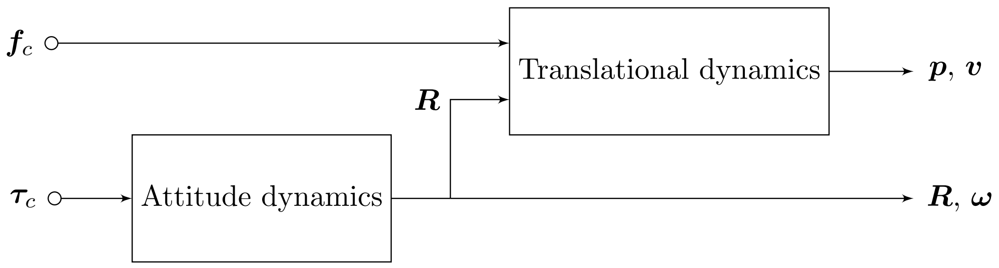


Figure 2.7: Representation of the quadrotor dynamic model with a block scheme, where the coupling through the rotation matrix between control force and the control torque is made clear.

Since, as it was explained before, quaternions can perform better than rotation matrices, the equations in 2.3.9 can be updated to

$$\dot{\boldsymbol{p}} = \boldsymbol{v} \tag{2.3.10a}$$

$$\dot{\mathfrak{q}} = \frac{1}{2}\mathfrak{q} \circ \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} = \frac{1}{2}\boldsymbol{M}(\mathfrak{q}) \begin{bmatrix} 0 \\ \boldsymbol{\omega} \end{bmatrix} \tag{2.3.10b}$$

$$\dot{\boldsymbol{v}} = -g\boldsymbol{e}_3 + m^{-1}\boldsymbol{R}(\mathfrak{q})\boldsymbol{f}_c \tag{2.3.10c}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(-\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{\tau}_c) \tag{2.3.10d}$$

where the values of the rotation matrices were replaced by quaternions where possible or derived from them in the translational kinematics. Partial proof of the equation 2.3.10b can be found in *Section* (A.2), where the same assumptions were made as in the previous equations.

Chapter 3

Model Predictive Control and Problem formulation

3.0.1 Model Predictive Control

Let us first introduce the concept of *Infinite horizon optimal control problem*. The generic continuous time system in question is the one described by the following dynamic equation

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t)) \tag{3.0.1}$$

in which $\boldsymbol{x} \in \mathbb{R}^n$ is the state and $\boldsymbol{u} \in \mathbb{R}^m$ is the input. Our aim is finding $\boldsymbol{u}(t)$ with $t \in [0, +\infty)$ that minimizes the following cost function defined over an infinite horizon with as initial condition $\boldsymbol{x}(0) = \tilde{\boldsymbol{x}}_0$

$$J_\infty(\boldsymbol{x}, \boldsymbol{u}) = \int_0^\infty V(\boldsymbol{x}(t), \boldsymbol{u}(t))\, dt \tag{3.0.2}$$

Although the problem is not easy to solve in general, it is possible to prove that if $V(\cdot)$ is positive definite and both $\boldsymbol{f}(\cdot)$ and $V(\cdot)$ are regular enough, then $\boldsymbol{u}^*(t)$ with $t \in [0, +\infty]$, the input that minimizes (3.0.2), stabilizes the origin of the system (3.0.1) for initial conditions in a neighborhood of $\tilde{\boldsymbol{x}}_0$ [16]. In particular, we want to minimize the following quantity

$$\min_{\boldsymbol{u}(\cdot)} \quad J_\infty(\boldsymbol{x}, \boldsymbol{u}) = \int_0^\infty \boldsymbol{x}^T(t)\boldsymbol{Q}\boldsymbol{x}(t) + \boldsymbol{u}^T(t)\boldsymbol{R}\boldsymbol{u}(t)\, dt \tag{3.0.3a}$$

subject to

$$\begin{aligned} &\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t) && (\textit{model dynamics}) \\ &\boldsymbol{x}(0) = \tilde{\boldsymbol{x}}_0 && (\textit{initial condition}) \end{aligned}$$

where $\boldsymbol{Q}$ and $\boldsymbol{R}$ are chosen cost weights, which follow the conditions $\boldsymbol{Q} \geq 0$ and $\boldsymbol{R} > 0$, then the solution is guaranteed and is obtained in closed form through the *Theorem* (3.0.1).

Theorem 3.0.1 (Solution of the infinite horizon LQ optimal control problem). *Let $\boldsymbol{Q}^{\frac{1}{2}}$ be a matrix such that $\boldsymbol{Q} = (\boldsymbol{Q}^{\frac{1}{2}})^T\boldsymbol{Q}^{\frac{1}{2}}$. It holds that:*

1. *the Algebraic Riccati Equation (ARE),*

$$\boldsymbol{A}^T\boldsymbol{P} + \boldsymbol{P}\boldsymbol{A} - \boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P} + \boldsymbol{Q} = \boldsymbol{0} \tag{3.0.4}$$

   has a unique positive semi-definite solution $\boldsymbol{P}_\infty$. Moreover, if $(\boldsymbol{A}, \boldsymbol{Q}^{\frac{1}{2}})$ is observable, then $\boldsymbol{P}_\infty$ is positive definite.

2. *the state feedback control law:*

$$\boldsymbol{u}(t) = -\boldsymbol{K}\boldsymbol{x}(t) \qquad \text{with} \qquad \boldsymbol{K} = \boldsymbol{R}^{-1}\boldsymbol{B}^T\boldsymbol{P}_\infty \tag{3.0.5}$$

   minimise the infinite horizon quadratic cost function (3.0.3a)*, and makes the closed–loop system asymptotically stable.*

if and only if $(\boldsymbol{A}, \boldsymbol{B})$ is stabilizable and $(\boldsymbol{A}, \boldsymbol{Q}^{\frac{1}{2}})$ is detectable ([36]).

Let us remember that in (3.0.3a) we are considering a special case where the model of the system is linear and the cost function is a linear-quadratic one, but in general, when $V(\cdot)$ and $\boldsymbol{f}$ assume a generic form, the optimal input that minimizes (3.0.2) cannot be found in close form. Another issue is that this kind of problem is not suitable to be solved numerically, in the first place because the solution space is infinite dimensional and in the second place because the time interval of interest is infinite as well. Moreover, when it is possible, the computed control input is open loop and represents a solution obtained using the exact model defined by $\boldsymbol{f}(\cdot)$, which is, in many cases, an inexact representation of the actual system. Indeed, we are looking for a closed loop system that is robust to model errors.

For this reason, we will simplify the problem with an approximate numerical solution. The first assumption is to address the problem to discrete time models. Although most systems have a continuous dynamical model by nature, it is preferred to use discrete dynamical models as the controller will certainly be implemented on a digital device and this makes it more feasible to compute the input to be applied to the system. Then we can discretize the model described in (3.0.1) with the proper technique and the sample time T_s, to finally get

$$\boldsymbol{x}(k+1) = \boldsymbol{f}_k(\boldsymbol{x}(k), \boldsymbol{u}(k)) \tag{3.0.6}$$

where, with a slight abuse of notation, it holds that $k = hT_s,\ h \in \mathbb{N}$.
The infinite horizon was the second problem, but we can simplify it by considering a finite time interval $[0, T]$. Since we address the problem as a discrete

time system, this time interval can be sampled obtaining a finite set of samples of length N, referred to as *control horizon*. The optimal problem then changes to the new form

Problem 3.0.1. *Find $\boldsymbol{u}^*(0)\dots\boldsymbol{u}^*(N-1)$ that minimizes the following optimization problem*

$$\begin{aligned}
&\min_{\boldsymbol{u}(\cdot)} && J = \sum_{k=0}^{k-1} V(\boldsymbol{x}(k), \boldsymbol{u}(k)) + V(\boldsymbol{x}(N)) && \\
&\textit{subject to} && && \\
& && \boldsymbol{x}(k+1) = \boldsymbol{f}_k(\boldsymbol{x}(k), \boldsymbol{u}(k)) && \textit{(model dynamics)} \\
& && \boldsymbol{x}(0) = \tilde{\boldsymbol{x}}(t) && \textit{(initial condition)} \\
& && \boldsymbol{x}(k) \in \mathcal{X} && \textit{(state constraint)} \\
& && \boldsymbol{u}(k) \in \mathcal{U} && \textit{(input constraint)}
\end{aligned}$$

where $J(\cdot)$ represents the cost function, $\boldsymbol{x}(\cdot)$ and $\boldsymbol{u}(\cdot)$ are the system state and manipulated variable (system control) respectively, $\mathcal{X}$ and $\mathcal{U}$ are the states space and input space respectively.

We can notice that the formulation of constraints are embedded in the optimisation problem, since the set $\mathcal{X}$ represents the possible states that the system can assume and the set $\mathcal{U}$ the possible inputs.
Model Predictive Control is an approach based on the iterative solution of the *Optimal Control Problem* (OPC) (3.0.1) at each time instant t. The cost function takes into account the evolution of the system over the next N instances of the horizon and by minimising this function we get a series of inputs $\boldsymbol{u}^*(0)\dots\boldsymbol{u}^*(N-1)$ that is linked to the minimum cost index. If we want to have close-loop control, we need to take as input only the first element of the series, namely $\boldsymbol{u}^*(0)$, so that each iteration the system evolves producing a new state, which is measured, and a new control problem is constructed. The main idea behind the MPC concept is described in *Figure* 3.1.

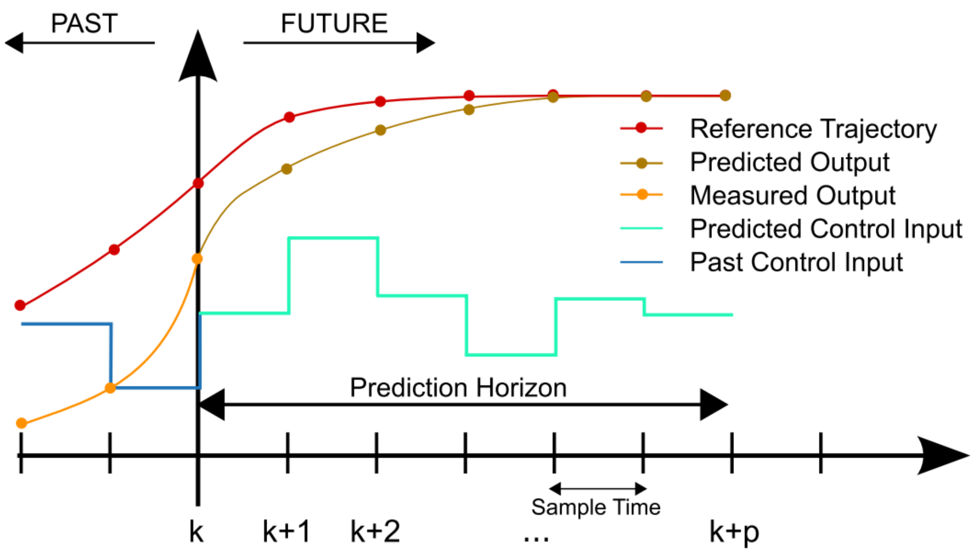


Figure 3.1: Principle of MPC.

3.1 Problem formulation

The objective of this section is to provide an idea of the UAV-UGV coordination problem we want to solve, along with the assumptions we are going to use to design our controller and the overall approach for the problem solution. The UGV has to be tracked by the quadrotor, of which there a complete definition of the dynamics that it has to obey in its movement. The UGV instead, is not modeled but will simply move on a flat plane. It was explained in *Section* (2.2) why quaternions are preferable over rotation matrices and for this reason only the model described by the *Equations* (2.3.10a) - (2.3.10d) will be used in the following paragraphs. To fully characterize the quadrotor, we want to have a definition of its position in the reference frame $\boldsymbol{p} = [x\ y\ z]^T$, its attitude through the quaternion $\mathbf{q} = [q_1\ q_2\ q_3\ q_4]^T$, its velocity along the frame axis $\boldsymbol{v} = [v_x\ v_y\ v_z]^T$ and the angular velocity of its body $\boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]^T$. At this point we can define the state vector $\boldsymbol{\xi} \in \mathbb{R}^{13}$, i.e.

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{p} \\ \mathbf{q} \\ \boldsymbol{v} \\ \boldsymbol{\omega} \end{bmatrix}. \tag{3.1.1}$$

Looking at the problem on the controller side, it was decided that the output would consist of the thrust values of the single rotors $[T_0\ T_1\ T_2\ T_3]^T$, which are related to the total thrust and total torque that were presented in (2.3.6a)-(2.3.6b) through the following equations

$$f_c = \begin{bmatrix} 0 \\ 0 \\ \Sigma T_i \end{bmatrix}. \tag{3.1.2}$$

In (3.1.2) we can see that all the thrust generated by the propellers is directed along the z axis in the body frame, as we expected since the quadrotor is under-actuated.

$$\boldsymbol{\tau}_c = \boldsymbol{P} \begin{bmatrix} T_0 \\ T_1 \\ T_2 \\ T_3 \end{bmatrix}. \tag{3.1.3}$$

where the $\boldsymbol{P}$ matrix is built according to the quadrotor shape and following its rotor drag coefficients

$$\boldsymbol{P} = \begin{bmatrix} -d_{x_0} & -d_{x_1} & d_{x_2} & d_{x_3} \\ d_{y_0} & -d_{y_1} & -d_{y_2} & d_{y_3} \\ -c_\tau & c_\tau & -c_\tau & c_\tau \end{bmatrix} \tag{3.1.4}$$

We derive the other value needed for the dynamics $\boldsymbol{\tau}_c$, directly in the body frame. At this point we can define as input vector the rotors thrusts $u_c = [T_0\ T_1\ T_2\ T_3]^T$ and get the following system

$$\dot{\boldsymbol{\xi}}(t) = \boldsymbol{f}(\boldsymbol{\xi}(t), \boldsymbol{u}_c(t)) \tag{3.1.5}$$

where $\boldsymbol{f}(\cdot,\cdot)$ comes from the *Equations* (2.3.10a) - (2.3.10d).

In this work we assume that there are already precise measurements of the states of the quadrotor, since the focus is not on estimating the quadrotor state. For this reason the quadrotor has access to the state error $\tilde{\boldsymbol{\xi}}$, which is defined as

$$\tilde{\boldsymbol{\xi}} = \begin{bmatrix} \tilde{p} \\ \tilde{v} \\ \tilde{\mathfrak{q}} \\ \tilde{\omega} \end{bmatrix} \in \mathbb{R}^{13}. \tag{3.1.6}$$

It is common practice to have multiple layers of control when working with quadrotors because of their under-actuation. Generally there is a high level position and speed controller, which generates thrust and reference values for a lower level attitude controller. Then, the torques generated by the attitude controller along with the thrusts generated by the position controller are converted to propeller speeds. In this work though, the goal of the controller is to provide the rotors thrust $[T_0\ T_1\ T_2\ T_3]$, which will be tracked by a low-level controller setting the motor speeds. Here, the motor control is not considered, since it required a high rate, difficult to reliably achieve with onboard real-time optimization, to stabilize and get vibration-free flight. For the quadrotor low-level control, we refer to [37].

Now that we have defined the the general approach to the controller design, we can define better how each controller is going to have a role in it. The NMPC is needed to provide a stable and accurate baseline controller, that will be able to track the reference trajectory. In fact this problem is formulated as a leader-follower problem, where the leader, the UGV, is moving along its path and does not have access to any information about the quadrotor, which has to track the UGV to the best of its capabilities. To this end we will make use of the NMPC with the addition of a variable weight set that can adjust itself according to the position error. This is so that the NMPC can work both when the quadrotor is far from the reference trajectory, as well as when it needs to make small adjustments in close proximity to the UGV and should not care too much about attitude error any longer. For this reason the cost function weights seen in (3.0.3a) will change slightly and have the following formulation at each iteration

$$\boldsymbol{Q}(e_p) = (1 - e^{-||e_p||_2^2})\boldsymbol{Q}_{far} + e^{-||e_p||_2^2}\boldsymbol{Q}_{close} \tag{3.1.7}$$

where $\boldsymbol{Q}_{far}$ and $\boldsymbol{Q}_{close}$ are preset weights corresponding to a big position error e_p and a small one respectively. These parameters fine-tuned manually starting from $\boldsymbol{Q}_{far}$, with the objective of having a stable performance and would not give too much weight to the position error. The second set of parameters $\boldsymbol{Q}_{close}$ was chosen to increase the weight of the position error after the quadrotor got closer to the reference position, so that it could make ulterior adjustments without caring too much about the other state errors.

L1AC is meant deal with the errors in the model due to unexpected effects or errors in the model parameters and in this work we decided to use its ability to reject the disturbances. The simulated disturbance was the ground effect. Ground effect is an aerodynamic phenomena which reduces the induced drag of aircraft and thereby increases its lift-to-drag ratio. It is the lift increase generated by rotors when an aircraft is close to a surface. The ground effect has been studied for years and an effective ground effect model for helicopter rotor has been presented by Cheeseman and Bennett [38]. It is shown that the ratio of the thrust in ground effect (IGE) to that outside of ground effect (OGE) for a single rotor operating at constant power, is a function of the rotor radius and the vertical distance from propeller disk to the ground. In practice the formula used in the simulation was the following

$$\frac{T_{IGE}}{T_{OGE}} = \frac{1}{1 - \left(\frac{r}{4z_r}\right)^2} \tag{3.1.8}$$

Note that T_{IGE} and T_{OGE} are in the case of constant power, and T_{IGE} is always not less than T_{OGE}. However the empirical formula 3.1.8 is obtained from a single rotor. It may be unsuitable for quadrotor. The four rotors stand close to each other for quadrotor, and there may exist some unpredictable airflow influences between them. In [39] a correction coefficient is introduced into the previous equation to model the ground effect of the quadrotor, getting the following equation

$$\frac{T_{input}}{T_{output}} = 1 - \rho \left(\frac{r}{4z_r}\right)^2 \tag{3.1.9}$$

According to their tests the correct value for ρ was 8.6. Although the result is based on a different quadrotor, the value was kept the same during the simulations of this thesis. Although this equation would usually be considered valid for values of $0.5 < \frac{z_r}{r} < 2$, we decided to stop at 0.8, which corresponds roughly to a ratio of 6 between T_{input} and T_{output}, so that the output thrust T_{output} would not diverge. In fact, if we look at the *Figure* 3.2, we can notice that it diverges for values of z_r that are too low.

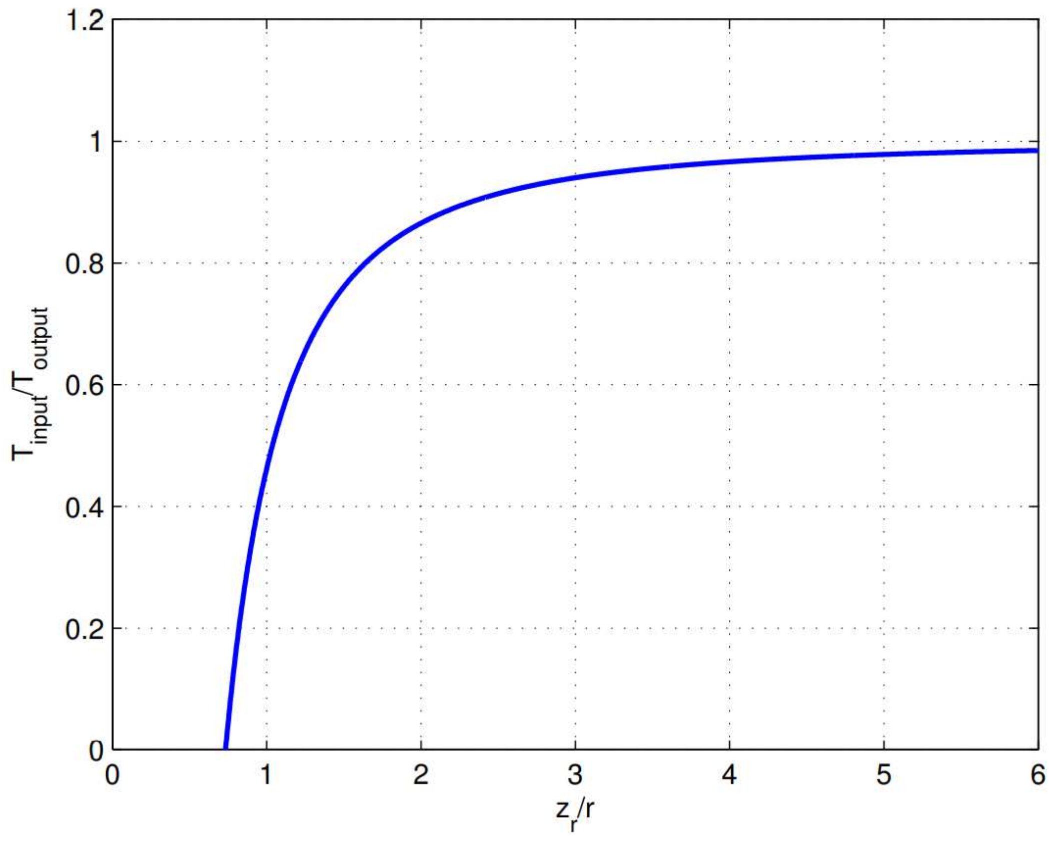


Figure 3.2: Graph representing the ratio T_{input}/T_{output}, as the height of the quadrotor changes.

Chapter 4

UAV Control via NMPC and L1AC

In this chapter we will further look into the *Nonlinear Model Predictive Control* (NMPC) and *L1 Adaptive Control* (L1AC) schemes used for the UGV tracking. Our aim is to provide all the tools and arguments for the construction of the cost function, define the input space along with the agent states, provide a formulation of the L1AC and the controllers schemes.

4.1 Runge-Kutta Integration

The first step consists in showing how the agent dynamics and kinematics update in a discrete manner. Because of the strong non-linearity describing the model of the agent, classical discretization techniques cannot be applied. For this reason, the *Runge-Kutta* integration method was chosen as the integration method. It is an effective and widely used method for solving the initial value problems of differential equations. With it we can construct an accurate high-order numerical method for the functions themselves without the need for the high-order derivatives of the functions. Consider first-order initial-value problem

$$\begin{cases} y' &= f(x,y), \qquad a \le x \le b \\ y(a) &= y_0 \end{cases} \tag{4.1.1}$$

To derive the *Runge–Kutta* method, we divide the interval $[a,b]$ into N sub-intervals as $[x_n, x_{n+1}]$ $n = 0, 1, \dots, N-1$, integrating $y' = f(x,y)$ over $[x_n, x_{n+1}]$ and utilizing the *mean value theorem* for integrals. We obtain

$$y(x_{n+1}) - y(x_n) = \int_{x_n}^{x_{n+1}} f(x, y(x))\, dx = hf(\varepsilon, y(\varepsilon)) \tag{4.1.2}$$

where $h = x_{n+1} - x_n$, $\varepsilon \in [x_n, x_{n+1}]$, i.e.

$$y(x_{n+1}) = y(x_n) + hf(\varepsilon, y(\varepsilon)) \tag{4.1.3}$$

If we approximate $f(\varepsilon, y(\varepsilon))$ by the linear combination values $f(\varepsilon_1, y(\varepsilon_1))$, $f(\varepsilon_2, y(\varepsilon_2)), \dots, f(\varepsilon_m, y(\varepsilon_m))$ of $f(x, y(x))$ on the interval $[x_n, x_{n+1}]$, then we arrive at the general form of *Runge-Kutta* method

$$y_{n+1} = y_n + h \sum_{i=1}^{m} c_i f(\varepsilon_i, y(\varepsilon_i)) \tag{4.1.4}$$

Different values for the parameters m, c_i and ξ_i allow us to obtain different forms of *Runge-Kutta* computation formula; in this way it is possible to obtain higher order *Runge–Kutta* computation formula by choosing suitable values of parameters. The most widely used *Runge–Kutta* formula is

$$\begin{cases} y_{n+1} & = y_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right) \\ k_1 & = hf(x_n, y_n) \\ k_2 & = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1) \\ k_3 & = hf(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2) \\ k_4 & = hf(x_n + h, y_n + k_3) \end{cases} \tag{4.1.5}$$

which is also called *four-order Runge–Kutta* method due to the need of four values of function in each step iteration.
We will only consider the discrete-time model describing the agent's dynamics, obtained by applying the *Runge-Kutta* method to the equation (3.1.5). Therefore we have that

$$\boldsymbol{\xi}(k+1) = \boldsymbol{f}_k(\boldsymbol{\xi}(k), \boldsymbol{u}(k)). \tag{4.1.6}$$

Please note that from now on we will write $\boldsymbol{u}(\cdot)$ without the subscript c to indicate the control input.

4.2 UAV NMPC Controller

The quadrotor has the task of landing on the UGV, which in practice translates to moving along with the UGV standing directly on top of it. In order to achieve this goal in a realistic and satisfactory manner it needs to be able to both bring the quadrotor close to the platform when it is far away and to accurately make small corrections when it is close to the target destination. Since we have access to the full state of the quadrotor, we can confront it with the desired state and minimize the difference. However, this operation has to be performed while obeying the constraints that are set for the input, for the model and following the dynamics of the latter. Thus, the optimization problem can be formulated as

$$\boldsymbol{u}^*(\cdot) = \underset{\boldsymbol{u}(\cdot)}{\operatorname{argmin}} \quad J = \sum_{k=0}^{N-1} \boldsymbol{e}^T(k)\boldsymbol{Q}\boldsymbol{e}(k) + \boldsymbol{u}^T(k)\boldsymbol{R}\boldsymbol{u}(k) \tag{4.2.1a}$$

$$\text{subject to} \quad \boldsymbol{\xi}(k+1) = \boldsymbol{f}_k(\boldsymbol{\xi}(k), \boldsymbol{u}(k)) \qquad (\textit{model dynamics}) \tag{4.2.1b}$$

$$\boldsymbol{\xi}(0) = \boldsymbol{\xi}(t) \qquad (\textit{initial condition}) \tag{4.2.1c}$$

$$\boldsymbol{u}(k) \in \mathcal{U} \qquad (\textit{input constraint}) \tag{4.2.1d}$$

where the first term $\boldsymbol{e}^T(k)\boldsymbol{Q}\boldsymbol{e}(k)$ is due to the state error in the cost function and the second term $\boldsymbol{u}^T(k)\boldsymbol{R}\boldsymbol{u}(k)$ is due to the input amplitude.

4.2.1 Definition of the cost function

In the formulation of the cost function we used in (4.2.1a) we did not define properly what $\boldsymbol{e}$ was. First it must be said that this is the difference of the state error with respect to the reference state error. Obviously the latter one is supposed to be always zero, so this value really consists only in the state error and the reasons behind the use of this more convoluted form of state error will be explained next. To do that we have to decompose $\boldsymbol{e}$ in its single elements, to get the following equations

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{e_p} \\ \boldsymbol{e_\mathfrak{q}} \\ \boldsymbol{e_v} \\ \boldsymbol{e_\omega} \end{bmatrix} \in \mathbb{R}^{13} \tag{4.2.2}$$

where the values of $\boldsymbol{e_p}$, $\boldsymbol{e_v}$ and $\boldsymbol{e_\omega}$ are respectively

$$\boldsymbol{e_p} = \begin{bmatrix} (p_x - p_{ref_x}) - 0 \\ (p_y - p_{ref_y}) - 0 \\ (p_z - p_{ref_z}) - 0 \end{bmatrix} \in \mathbb{R}^3 \tag{4.2.3}$$

$$\boldsymbol{e_v} = \begin{bmatrix} (v_x - v_{ref_x}) - 0 \\ (v_y - v_{ref_y}) - 0 \\ (v_z - v_{ref_z}) - 0 \end{bmatrix} \in \mathbb{R}^3 \tag{4.2.4}$$

$$\boldsymbol{e_\omega} = \begin{bmatrix} (\omega_x - \omega_{ref_x}) - 0 \\ (\omega_y - \omega_{ref_y}) - 0 \\ (\omega_z - \omega_{ref_z}) - 0 \end{bmatrix} \in \mathbb{R}^3 \tag{4.2.5}$$

which are all simple differences of the state elements from the corresponding reference signals.
The quaternion error $\boldsymbol{e}_{\mathbf{q}}$ however, is formulated in a different manner. The goal is to minimize the pitch, roll and yaw angle error of the quadrotor, but there are multiple ways to choose the type of distance we want to use for this task. In particular we may opt for a simpler distance measure between the quaternion $\mathbf{q}$ and the reference quaternion $\mathbf{q}_{ref}$, which would be the so-called *Frobenius* or *chordal* metric. In this case the aim is to compute the length of the minimum curve connecting the two quaternions, but the curve does not have to belong to the sphere $\mathbb{SO}^3$. This would correspond to taking the difference between the quaternions in the following manner

$$d_F^2(\mathbf{q}, \mathbf{q}_{ref}) = \|\mathbf{q} - \mathbf{q}_{ref}\|_F^2 \tag{4.2.6}$$

The other and more significant option is the *Riemaniann geodesic* metric, which calculates the length of the minimum curve, belonging to the sphere $\mathbb{SO}^3$, that connects the two quaternions. However, determining this measure is not trivial and requires solving an integral in three variables. The equivalent quaternion based version of this distance is the so-called "Deviation from the identity metric" does not imply the integral with its complexity for the solution and is defined as

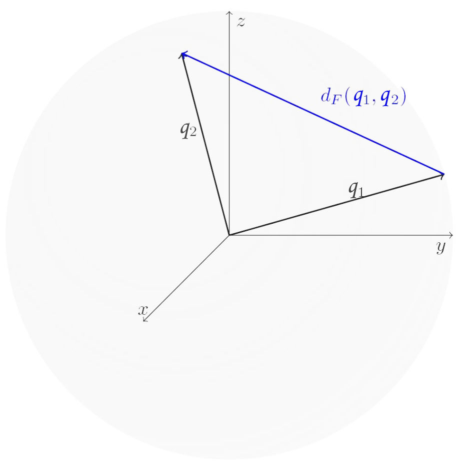
$$d_I^2(\mathbf{q}, \mathbf{q}_{ref}) = \|\mathbb{1}_{\mathbf{q}} - \mathbf{q}' \circ \mathbf{q}_{ref}\|_F^2 \tag{4.2.7}$$

where $\mathbb{1}_{\mathbf{q}}$ represents the quaternion corresponding to the identity matrix rotation, namely $\mathbb{1}_{\mathbf{q}} = [1, 0, 0, 0]^T$ and $\mathbf{q}'$ is the conjugate of the quaternion $\mathbf{q}$.
In *Figure* 4.1 we can see the difference between the two metrics. In the first case 4.1a, the *chordal* metric shows an approximation of the distance between the two quaternion rotations, that creates a chord connecting the two. In the second case 4.1b, the *Riemaniann geodesic* metric displays a perfect representation of the distance between quaternion 1 and quaternion 2, with an arc that spans across the sphere.

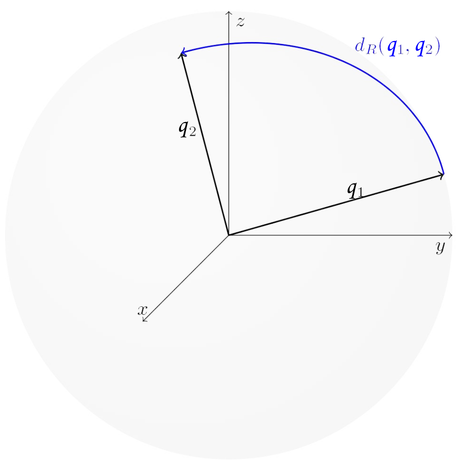
With the previous formulation, at each NMPC iteration we can define the orientation error as

$$\boldsymbol{e}_{\mathbf{q}} = d_I^2(\mathbf{q}, \mathbf{q}_{ref}) \in \mathbb{R}^4 \tag{4.2.8}$$

which is the last term needed for the state derived part of the cost function seen in (4.2.2).
The cost function has to optimize the input to minimize the state derived errors and the input, however it also has to follow the model dynamics as reported in (4.2.1b). For the most part, constraints on the states are not set, since they could



(a)



(b)

Figure 4.1: Chordal (a) and Riemaniann (b) geodesic metrics representation. The chordal metric on $\mathbb{S}^2 \in \mathbb{R}^3$ is measured directly in $\mathbb{R}^3$, while the geodesic metric is measured along $\mathbb{S}^2$.

make the problem unsolvable for the NMPC, but rather included in the state error which is minimized according to the model. The only exception is a bound on the height of the quadrotor, for obvious reasons, where the formulation is simply

$$\boldsymbol{p}_z > 0 \tag{4.2.9}$$

The input, however was given some constraints. This is due to the fact that it would not make the problem unfeasible and those constraints corresponded to the maximum thrust that the quadrotor of choice could generate. The constraints were set according to values found in [40], which was 2.6 and represented the relationship between the weight of the quadrotor and its maximum thrust, which was split equally in four parts for the maximum thrust of each propeller, namely

$$\forall i = 0, 1, 2, 3 \qquad \max|T_i| = 2.6\,\frac{mg}{4} \tag{4.2.10}$$

setting in this way the bounds on the input

$$\boldsymbol{u}_{min} \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{max} \tag{4.2.11}$$

Another extremely important feature of the NMPC is the prediction horizon over which it analyzes the state evolution and the corresponding state errors. In fact the length of this horizon has to be big enough so that it can see far into the future. However there is a drawback to increasing the horizon indefinitely, since the computing power of the computer is limited. For each instance in the horizon the computer has to find the minimum of the cost function and if the computing time for this minimization along with the other tasks that the processor has to keep up with have to stay below the sampling time if we want the algorithm to be applicable in real time. The number of instances that have to be explored depends directly on the time horizon we desire and the sampling time, since

$$N = \frac{T_h}{T_s} \tag{4.2.12}$$

where N is the number of instances, T_h the period of the whole horizon and T_s the sampling time of the NMPC. In particular we have that T_s cannot be too large, or the controller will not be able to keep up with the system, and T_h has to be long enough so that it can see the relatively long-term advantages that certain input choices may bring.

To help with the effectiveness of the NMPC the reference signal was modified according to the movement of the UGV. In fact the reference velocity was not simply set to zero, but estimated from the reference positions. The reference velocity passed to the NMPC controller was then

$$\hat{\boldsymbol{v}}_{ref}(k) = \frac{\boldsymbol{p}_{ref}(k) - \boldsymbol{p}_{ref}(k-1)}{T_s} \tag{4.2.13}$$

where $\boldsymbol{p}_{ref}(k-1)$ is the previous reference position, delayed by the sampling time T_s. Once the velocity $\hat{\boldsymbol{v}}_{ref}(k)$ was estimated, the reference position was also updated according to the new reference velocity. Starting from the reference position at the k-th instant, the horizon of the reference positions was set to

$$\hat{\boldsymbol{p}}_{ref}(k+j) = \boldsymbol{p}_{ref}(k) + j \cdot \hat{\boldsymbol{v}}_{ref}(k) \tag{4.2.14}$$

4.2.2 NMPC controller architecture

Looking at *Figure* 4.2 we can get a clear and complete idea of how the NMPC controller works and what its position would be in real system. We can see that there is a reference signal $\boldsymbol{\xi}_{ref}$ that goes through a velocity estimator. The estimated velocity is the used to correct part of the reference signal and obtain $\boldsymbol{\xi}'_{ref}$, which is then fed along with the estimated state $\hat{\boldsymbol{\xi}}$ to the NMPC controller. At this point the optimization process we saw in *Equation* (4.2.1d) is followed to produce the first input u, which is then applied to the motor controller. This determines what the propellers thrust $\boldsymbol{T}$ will be. According to the quadrotor dynamics, which include phenomena like the ground effect, a new state $\boldsymbol{\xi}$ is produced according to the equations (3.1.2) and (3.1.3) that relate the input thrust to the more classic model formulation seen in (2.3.9), and it will be estimated by the sensors to start another cycle.

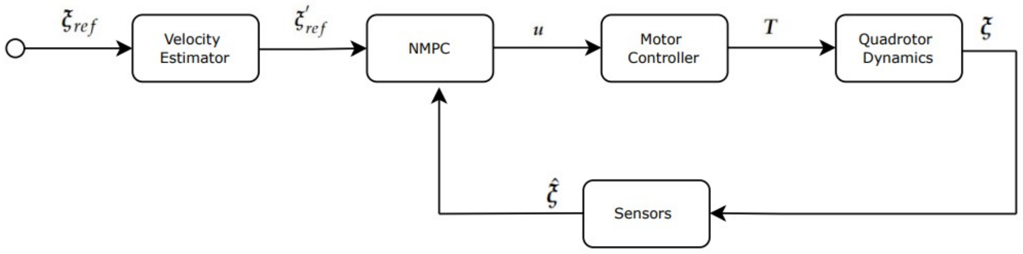


Figure 4.2: NMPC control architecture.

4.3 L1AC Controller

The purpose of this thesis is to complement the control performance of the NMPC with a supplementary L1AC. This interaction between has been experimented and has proven its capabilities with satisfactory results in the past years both with with NMPC [41] and other kinds of MPC [42] [43]. The L1AC is meant to compensate for the model errors and for external disturbances, which in the case of this work were due to either the ground effect and changes in weight.

L1AC uses the nonlinear reference model presented in (2.3.10) and estimates both matched and unmatched uncertainties using a piece-wise constant adaptation law. Like in [41], we want to account for the uncertainties directly at the rotor thrust level. We can rewrite the dynamics to account for both matched and unmatched uncertainties as

$$\dot{\boldsymbol{v}} = -g\boldsymbol{e}_3 + \boldsymbol{R}(q)\frac{\boldsymbol{T}}{m} + \boldsymbol{R}(q)\frac{\boldsymbol{\zeta}}{m} - \boldsymbol{C}_d\boldsymbol{v} \tag{4.3.1}$$

$$\dot{\boldsymbol{\omega}} = \boldsymbol{J}^{-1}(-\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{\tau}_c + \boldsymbol{\xi}) \tag{4.3.2}$$

where $\boldsymbol{T} = f_c$ represents the total thrust of the propellers, $\boldsymbol{\zeta} = [\zeta_x, \zeta_y, \zeta_z]^T$ is the uncertainty appearing in the thrust and $\boldsymbol{\xi} = [\xi_x, \xi_y, \xi_z]^T$ is the uncertainty appearing in the torque. Since a quadrotor is an under-actuated system, capable of providing linear acceleration only along its body z-axis, the unmatched uncertainties defined as $\boldsymbol{\sigma}_{um} = [\zeta_x, \zeta_y]^T$ appear purely in the X and Y linear directions. This can be thought of as existing in the null space of the controllability matrix and therefore cannot be compensated for directly. Then, what remains are the matched uncertainties $\boldsymbol{\sigma}_m = [\zeta_z, \xi_x, \xi_y, \xi_z]^T$ which can be compensated for directly.

We can now consider the reduced the state variable $\boldsymbol{z} = [\boldsymbol{v}, \boldsymbol{\omega}]$. Then its derivative e can be broken up as a function of the nominal and uncertain dynamic behavior as follows

$$\dot{\boldsymbol{z}} = \boldsymbol{f}(\boldsymbol{R}(\mathbf{q})) + \boldsymbol{g}(\boldsymbol{R}(\mathbf{q}))(\boldsymbol{u}_{\mathcal{L}1} + \boldsymbol{\sigma}_m) + \boldsymbol{g}(\boldsymbol{R}(\mathbf{q}))^{\perp}\boldsymbol{\sigma}_{um} \tag{4.3.3}$$

where $f(\boldsymbol{R}(\mathbf{q}))$ is the desired dynamics

$$f(\boldsymbol{R}(\mathbf{q})) = \begin{bmatrix} \boldsymbol{g} + \frac{T_{NMPC}}{m}\boldsymbol{e}_z^B - \boldsymbol{C}_d\boldsymbol{v} \\ \boldsymbol{J}^{-1}(-\boldsymbol{\omega} \times \boldsymbol{J}\boldsymbol{\omega} + \boldsymbol{\tau}_c) \end{bmatrix} \tag{4.3.4}$$

In the second part of *Equation* (4.3.3) we define $g(\boldsymbol{R}(\mathbf{q}))$ as the uncertainty in the matched component of the dynamics, while $g(\boldsymbol{R}(\mathbf{q}))^{\perp}$ is the unmatched component.

$$g(\boldsymbol{R}(\mathbf{q})) = \begin{bmatrix} \frac{\boldsymbol{e}_z^B}{m}, \frac{\boldsymbol{e}_z^B}{m}, \frac{\boldsymbol{e}_z^B}{m} \\ \boldsymbol{J}^{-1}\boldsymbol{P} \end{bmatrix} \qquad g(\boldsymbol{R}(\mathbf{q}))^{\perp} = \begin{bmatrix} \frac{\boldsymbol{e}_x^B}{m}, \frac{\boldsymbol{e}_y^B}{m} \\ \mathbf{0}_{3\times1}, \mathbf{0}_{3\times1} \end{bmatrix} \tag{4.3.5}$$

Let $\boldsymbol{u}_{\mathcal{L}1}$ be the adaptive control input which can act as a standalone controller, or complement the NMPC signal via addition. Define the $\mathcal{L}1$ observer as

$$\dot{\hat{\boldsymbol{z}}} = f(\boldsymbol{R}(\mathbf{q})) + g(\boldsymbol{R}(\mathbf{q}))(\boldsymbol{u}_{\mathcal{L}1} + \hat{\sigma}_m) + g(\boldsymbol{R}(\mathbf{q}))^{\perp}\hat{\sigma}_{um} + \boldsymbol{A}_s\tilde{\boldsymbol{z}} \tag{4.3.6}$$

where $\tilde{z} = \hat{z} - z$ and noting that z is the state obtained from an estimator and $\hat{z}$ is the state predicted from the $\mathcal{L}1$ observer. Then we can get the derivative of $\tilde{z}$ in the following manner

$$\dot{\tilde{\boldsymbol{z}}} = g(\boldsymbol{R}(\mathbf{q}))(\hat{\sigma}_m - \sigma_m)) + g(\boldsymbol{R}(\mathbf{q}))^{\perp}(\hat{\sigma}_{um} - \sigma_{um}) + \boldsymbol{A}_s\tilde{\boldsymbol{z}} \tag{4.3.7}$$

To find the close form solution of this problem we need to discretize the *Equation* (4.3.7). First we remember that the analytical solution for the continuous model is

$$\tilde{z}(t) = e^{\boldsymbol{A}_s t}\tilde{z}(0) + \int_0^t e^{\boldsymbol{A}_s(t-\tau)}\boldsymbol{G}[\hat{\sigma}(\tau) - \sigma(\tau)]d\tau \tag{4.3.8}$$

where

$$\boldsymbol{G}(t) = \left[g(\boldsymbol{R}(\mathbf{q}(t))), g(\boldsymbol{R}(\mathbf{q}(t)))^{\perp}\right] \qquad \hat{\sigma}(\tau) = \begin{bmatrix} \hat{\sigma}_m(\tau) \\ \hat{\sigma}_{um}(\tau) \end{bmatrix} \tag{4.3.9}$$

From now on we will omit the time dependence of $\boldsymbol{G}$. At this point we want to proceed with the discretization. First we define $\tilde{z}(k) = \tilde{z}(kT_s)$ and get the following equations

$$\begin{aligned}
\tilde{z}(k) &= e^{\boldsymbol{A}_s kT_s}\tilde{z}(0) + \int_0^{kT_s} e^{\boldsymbol{A}_s(kT_s-\tau)}\boldsymbol{G}[\hat{\sigma}(\tau) - \sigma(\tau)]d\tau \\
\tilde{z}(k+1) &= e^{\boldsymbol{A}_s(k+1)T_s}\tilde{z}(0) + \int_0^{(k+1)T_s} e^{\boldsymbol{A}_s((k+1)T_s-\tau)}\boldsymbol{G}[\hat{\sigma}(\tau) - \sigma(\tau)]d\tau \\
\tilde{z}(k+1) &= e^{\boldsymbol{A}_s T_s}\tilde{z}(k) + \int_{kT_s}^{(k+1)T_s} e^{\boldsymbol{A}_s((k+1)T_s-\tau)}\boldsymbol{G}[\hat{\sigma}(\tau) - \sigma(\tau)]d\tau
\end{aligned} \tag{4.3.10}$$

If at this point we substitute $v(\tau) = (k+1)T_s - \tau$ and assume the signal

$[\hat{\sigma}(\tau) - \sigma(\tau)]$ to be constant during the interval T_s, we can say that

$$\begin{aligned}
\tilde{z}(k+1) &= e^{\boldsymbol{A}_s T_s}\tilde{z}(k) - \Big(\int_{v(kT_s)}^{v((k+1)T_s)} e^{\boldsymbol{A}_s v} dv\Big)\boldsymbol{G}[\hat{\sigma}(k) - \sigma(k)] \\
\tilde{z}(k+1) &= e^{\boldsymbol{A}_s T_s}\tilde{z}(k) - \Big(\int_{T_s}^{0} e^{\boldsymbol{A}_s v} dv\Big)\boldsymbol{G}[\hat{\sigma}(k) - \sigma(k)] \\
\tilde{z}(k+1) &= e^{\boldsymbol{A}_s T_s}\tilde{z}(k) + \boldsymbol{A}_s^{-1}\Big(e^{\boldsymbol{A}_s T_s} - \mathbb{I}\Big)\boldsymbol{G}[\hat{\sigma}(k) - \sigma(k)]
\end{aligned} \tag{4.3.11}$$

At this point we are going to ignore the terms $-\boldsymbol{A}_s^{-1}\Big(e^{\boldsymbol{A}_s T_s} - \mathbb{I}\Big)\boldsymbol{G}\sigma(k)$ along with $\tilde{z}(k+1)$. The motivation for this decision is that it can be proven that the control will be stable and that all inputs, states and state estimation errors will remain bounded. The proof can be found in [44], but being extremely convoluted and out of the scope of this work, it will not be included in this thesis. With that being said, if we invert the revised equation, we get

$$\boldsymbol{A}_s^{-1}\Big(e^{\boldsymbol{A}_s T_s} - \mathbb{I}\Big)\boldsymbol{G}\hat{\sigma}(k) = -e^{\boldsymbol{A}_s T_s}\tilde{z}(k) \tag{4.3.12}$$

If we define $\boldsymbol{\Phi} = \boldsymbol{A}_s^{-1}\Big(e^{\boldsymbol{A}_s T_s} - \mathbb{I}\Big)$ and $\boldsymbol{\mu} = e^{\boldsymbol{A}_s T_s}\tilde{z}(k)$ we can write

$$\boldsymbol{\Phi}\boldsymbol{G}\hat{\sigma}(k) = -\boldsymbol{\mu} \tag{4.3.13}$$

$$\begin{bmatrix} \hat{\sigma}_m(k) \\ \hat{\sigma}_{um}(k) \end{bmatrix} = \hat{\sigma}(k) = -\boldsymbol{G}^{-1}\boldsymbol{\Phi}^{-1}\boldsymbol{\mu} \tag{4.3.14}$$

where *Equation* 4.3.14 is the same that is reported in the article [41]. At this point the new input is set to

$$\boldsymbol{u}_{\mathcal{L}1} = -\boldsymbol{C}(s)\hat{\sigma}_m \tag{4.3.15}$$

where $\boldsymbol{C}(s)$ is a low-pass filter. In practice, we implement the control law in discrete time as

$$\boldsymbol{u}_{\mathcal{L}1,k} = \boldsymbol{u}_{\mathcal{L}1,k-1}e^{-\omega_{co}T_s} - \hat{\sigma}_{m,k}(1 - e^{-\omega_{co}T_s}) \tag{4.3.16}$$

where *co* is the cut-off frequency of the strictly proper first order filter. Finally, the discrete time $\mathcal{L}1$ observer can be propagated forward in time via

$$\hat{z}(k+1) = \hat{z}(k) + [f_k + g_k(\boldsymbol{u}_{\mathcal{L}1,k} + \hat{\sigma}_{m,k}) + g_k^{\perp}\hat{\sigma}_{um,k} + \boldsymbol{A}_s\tilde{z}_k]\boldsymbol{T}_s \tag{4.3.17}$$

4.3.1 Complete controller architecture

In *Figure* 4.3 we can see where the L1AC stands and understand better the kind of role it plays. The reference signal does not change, since it is only needed for the NMPC controller. The L1AC needs as input the state estimation $\hat{\boldsymbol{\xi}}$ as well as the NMPC output $\boldsymbol{u}_{NMPC}$. This information is going to be used to estimate the disturbances starting from the reduced sate estimation error $\tilde{z}_k$ and after passing through a low-pass filter, an additional input $\boldsymbol{u}_{\mathcal{L}1}$ will be summed to $\boldsymbol{u}_{NMPC}$. At this point the total input $\boldsymbol{u}$ will behave like in the previous case depicted in *Figure* 4.2 and try to compensate the ground effect embedded in the quadrotor dynamics as well as providing an optimal state evolution.

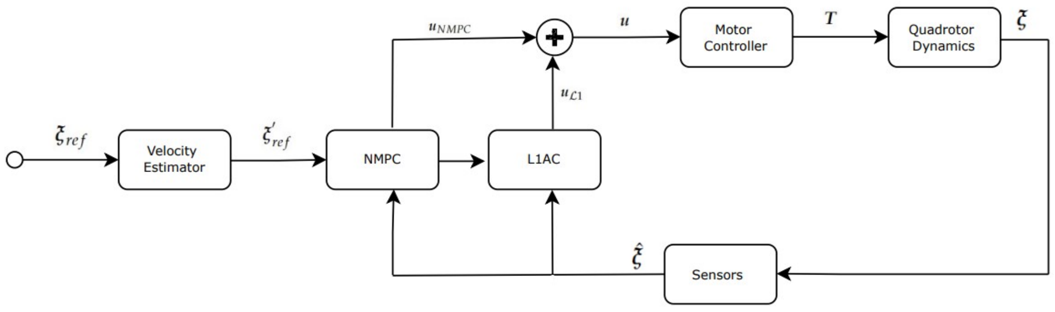


Figure 4.3: Architecture of the combination of both controllers.

Chapter 5

Simulation and results

In this chapter we will test and validate the proposed NMPC and L1AC combination for the UGV tracking problem that was described in *Chapter* 4. In particular we will first run the simulations with just the NMPC, to then add the L1AC and compare the performance improvement the L1AC actually produces, to see whether this control correct the ground effect. To further test it, we will add model imperfections and external disturbances in a further section. In order to validate the algorithm we set up a simulation environment in Matlab & Simulink.

5.1 Simulation setup

Before starting to tackle scenarios that characterise the study carried out in this thesis, it is fair to describe the general setup and the tools used. Furthermore, it is interesting to validate the model derived in *Section* 2.3, and for this purpose a simple hovering problem is studied. This is useful to test the NMPC for individual control and make some preliminary considerations on the cooperative algorithm.
Regarding the implementation of the NMPC, it was very useful to use **MATMPC**. This tool aims at providing an easy-to-use NMPC implementation. The optimal control problem (OCP) that should be solved is transcribed by multiple shooting and the resulting nonlinear program is solved by Sequential Quadratic Programming (SQP) method. Moreover, this tool supports fixed step (explicit/implicit) Runge-Kutta integrator for multiple shooting, which was presented in the beginning of *Section* 2.3. The derivatives that are needed to perform optimization are obtained by CasADi, the state-of-the-art automatic/algorithmic differentiation toolbox. For further information, please refer to the Matlab and MATMPC website [45].

To start, we recall that quadrotor control is accomplished by applying the input $\boldsymbol{u}_c = [T_0, T_1, T_2, T_3]^T$, while the system output, which in this work also corresponds to the state measured by the on-board sensors, is $\boldsymbol{\xi}$ as defined in (3.1.1). The input $\boldsymbol{u}_c$ is then connected to the state by the minimization of the cost function

$$\begin{aligned}
\boldsymbol{u}^*(\cdot) = \underset{\boldsymbol{u}(\cdot)}{\operatorname{argmin}} \quad & \sum_{k=0}^{k-1} V(\hat{\boldsymbol{\xi}}(k), \boldsymbol{u}(k)) & \\
\text{subject to} \quad & \boldsymbol{\xi}(k+1) = \boldsymbol{f}_k(\boldsymbol{\xi}(k), \boldsymbol{u}(k)) & (\textit{model dynamics}) \\
& \boldsymbol{\xi}(0) = \tilde{\boldsymbol{\xi}}(t) & (\textit{initial condition}) \\
& \boldsymbol{u}_{min} \leq \boldsymbol{u}(k) \leq \boldsymbol{u}_{max} & (\textit{input constraint})
\end{aligned}$$

where the function $V(\cdot, \cdot)$ is given by

$$V(\boldsymbol{\xi}(k), \boldsymbol{u}(k)) = \sum_{k=0}^{N-1} \boldsymbol{e}^T(k)\boldsymbol{Q}\boldsymbol{e}(k) + \boldsymbol{u}^T(k)\boldsymbol{R}\boldsymbol{u}(k) \tag{5.1.2}$$

and the diagonal matrix$\boldsymbol{Q}$ can be decomposed in all the coefficients for the different errors

$$\boldsymbol{Q} = diag([c_p, c_{\mathbf{q}}, c_v, c_\omega]) \tag{5.1.3}$$

Some simulations are shown below with the purpose of validating the model and simulink scheme used for this thesis. Throughout the thesis, as an example, the Crazyflie platform data is used to implement the model related to agents. The values of the model parameters are available, but are listed below anyway in the *Table* 5.1.

Agent parameters	
m [kg]	0.027
l [m]	0.042
r [m]	0.023
J_x [kg·m^2]	$2.4 \cdot 10^{-5}$
J_y [kg·m^2]	$2.4 \cdot 10^{-5}$
J_z [kg·m^2]	$3.2 \cdot 10^{-5}$
c_τ [m]	$3.7 \cdot 10^{-3}$
C_d [kg/s]	$4 \cdot 10^{-4}$

Table 5.1: Crazyflie quadrotor parameters.

Here m represents the mass of the quadrotor, l and r the arm length from center to rotor and propeller radius respectively, $[J_x, J_y, J_z]$ are the inertial values that compose the corresponding diagonal matrix, c_τ is the rotor drag torque constant, used in (3.1.4), and C_d is the wind friction coefficient.

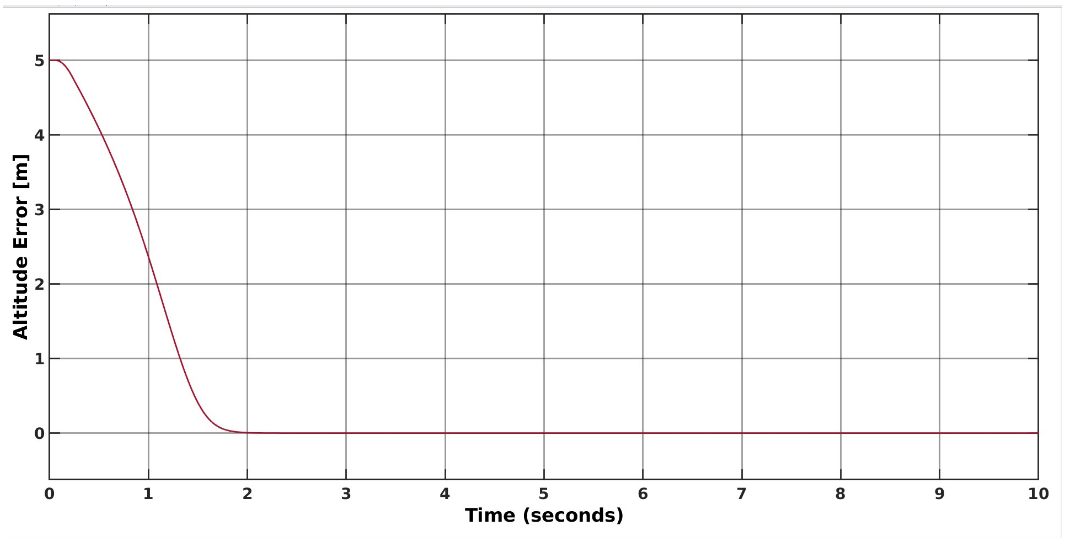
Next we present the NMPC setup that was used for the hovering task.

Initial condition	
$\boldsymbol{p}(0)$ [m]	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
$\boldsymbol{v}(0)$ [m/s]	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
$\mathbf{q}(0)$	$\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^T$
$\boldsymbol{\omega}(0)$ [rad/s]	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
Reference	
$\boldsymbol{p}^d$ [m]	$\begin{bmatrix}0 & 0 & 5\end{bmatrix}^T$
$\mathbf{q}^d$	$\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$
NMPC parameters	
Ts [s]	0.01
N	100
c_{p_f}	$\begin{bmatrix}7 & 7 & 1\end{bmatrix}^T$
c_{p_c}	$\begin{bmatrix}1000 & 1000 & 100\end{bmatrix}^T$
$c_{\mathbf{q}_f}$	$\begin{bmatrix}20 & 20 & 20 & 20\end{bmatrix}^T$
$c_{\mathbf{q}_c}$	$\begin{bmatrix}20 & 20 & 20 & 20\end{bmatrix}^T$
c_{v_f}	$\begin{bmatrix}2 & 2 & 3\end{bmatrix}^T$
c_{v_c}	$\begin{bmatrix}1 & 1 & 1\end{bmatrix}^T$
c_{ω_f}	$\begin{bmatrix}7 & 7 & 7\end{bmatrix}^T$
c_{ω_c}	$\begin{bmatrix}3 & 3 & 3\end{bmatrix}^T$
$T_{i,min}$ [N]	-0.172
$T_{i,max}$ [N]	0.172

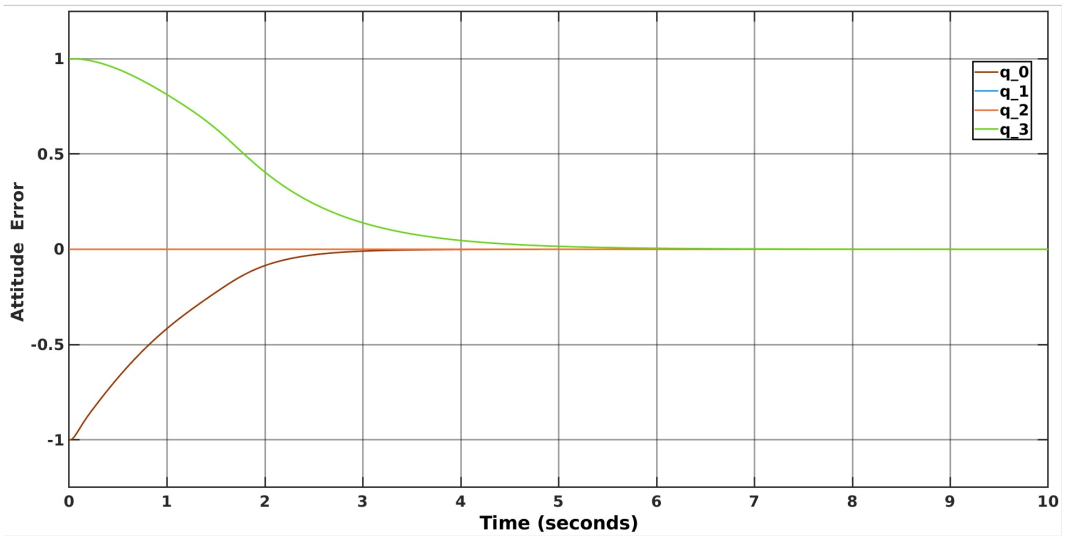
Table 5.2: Numerical simulation parameters.

where the far and close values for Q change based on the position error of the quadrotor, according to *Equation* 3.1.7. $T_{i,min}$ and $T_{i,max}$ represent the minimum and maximum thrusts that the i-th propeller can exert.
In the MPC, the tuning of parameters assumes a significant role, in particular because by acting on them it is possible to get different responses based on the chosen parameters. With regard to the sample time T_S and prediction horizon N, the computing time does not represent a limitation in Simulink as the simulation is not real time, and the computation can be as long as needed. The sampling

time was kept at the more classical value of $T_S = 0.01$ s, while the prediction horizon was set to $N = 100$ to cover a whole second.In the testing this proved to be the smallest value for the correct functioning of the algorithm. The NMPC parameters shown in the *Table* 5.2 are the result of several simulations in search of a good system response, providing stability for the quadrotor and bringing the state error to zero.



(a) Altitude error



(b) Attitude error

Figure 5.1: NMPC control test for quadrotor hovering.

This first simulation where the task was simple hovering was a success and the results is reported in *Figure* 5.1. It is possible to see that the control is effective, in fact both altitude and attitude errors converge to zero, thus the quadrotor stabilizes at the desired position. From the graphs in 5.2 it can be seen that the altitude steady state error is about zero and that there is no overshoot for both. Instead, from the graph 5.2 we can observe that the input after an

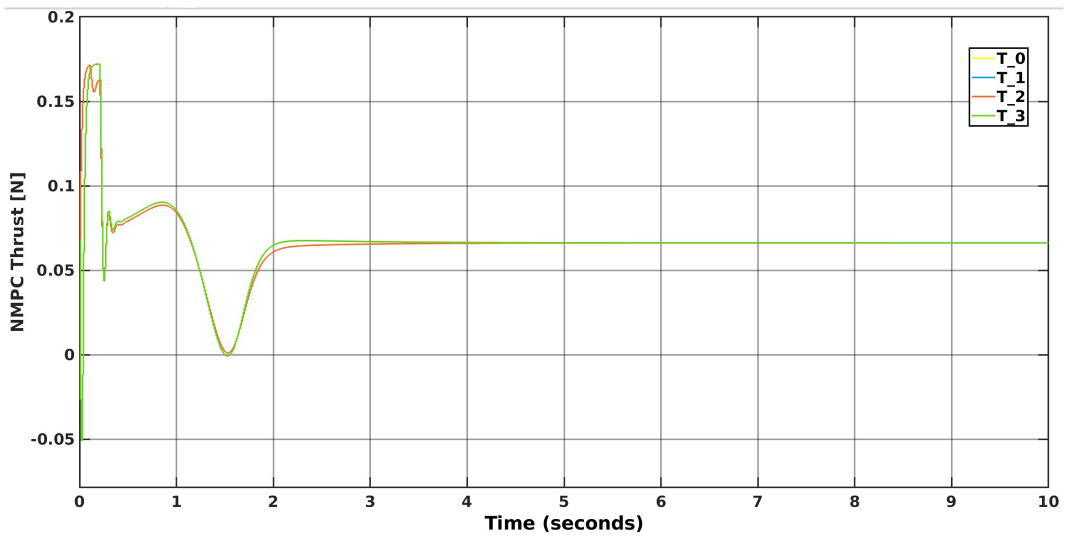


Figure 5.2: Thrust force exerted by the propellers during the NMPC regulated hovering task.

initial agitated phase, finds stability and converges. We remember that the goal of this test is to validate the used model and the simulation scheme and therefore we have not gone to depth in search of improved results since we don't want to assess controller performance at this stage.

5.2 Landing Simulation with Ground Effect

Here are some results about the ability of the quadrotor to perform a landing manoeuvre on a UGV. To simulate this event a reference state was created, which was meant to represent the ideal position for the landing, slightly above the UGV platform. The UGV would also be moving at constant speed in a chosen direction (which was chosen to be in the x-axis direction for simplicity reasons) for a total time of 10 s. This would test the ability of the algorithm to land the quadrotor without the need to stop the other vehicle.
One of the biggest challenges with the ground effect is that it creates a disturbance that varies in space and is therefore hard to compensate. On a positive note though, this effect pushes the aircraft up rather then down, making overshoots less likely.
While the parameters of the NMPC were left the same as in the hovering case, the parameters of the L1AC were added in this simulation. They included the adaptation gain A_s and the cutoff frequency co, which were necessary for the adaptation law reported in *Equations* (4.3.6) and (4.3.16) respectively. A_s had to be an Hurwitz matrix, so for simplicity reasons it was kept to an identity matrix multiplied by a constant. It was then fine tuned manually. The parameter co instead was chosen as the highest value for the low-pass filter cutoff frequency that allowed for a relatively smooth controller output. All the new parameters

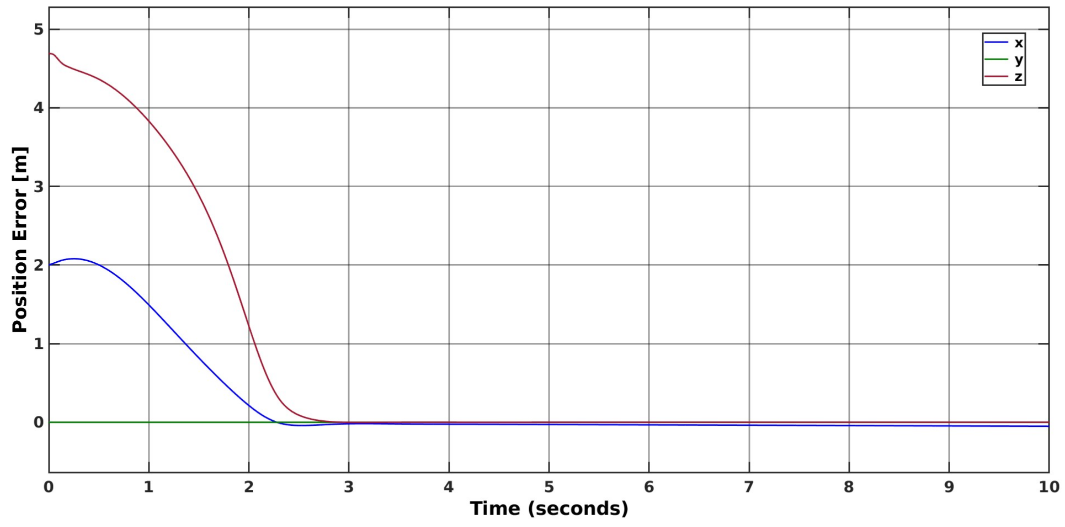
used in the simulations are the following

Initial condition	
$\boldsymbol{p}(0)$ [m]	$\begin{bmatrix}2 & 0 & 5\end{bmatrix}^T$
$\boldsymbol{v}(0)$ [m/s]	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
$\mathbf{q}(0)$	$\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$
$\boldsymbol{\omega}(0)$ [rad/s]	$\begin{bmatrix}0 & 0 & 0\end{bmatrix}^T$
Reference	
$\boldsymbol{p}^d(0)$ [m]	$\begin{bmatrix}0 & 0 & 5\end{bmatrix}^T$
$\mathbf{q}^d$	$\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$
$\boldsymbol{v}^d$ [m/s]	$\begin{bmatrix}-0.5 & 0 & 0\end{bmatrix}^T$
L1AC parameters	
A_s	$-1 \cdot \mathbb{1}_6$
co	10

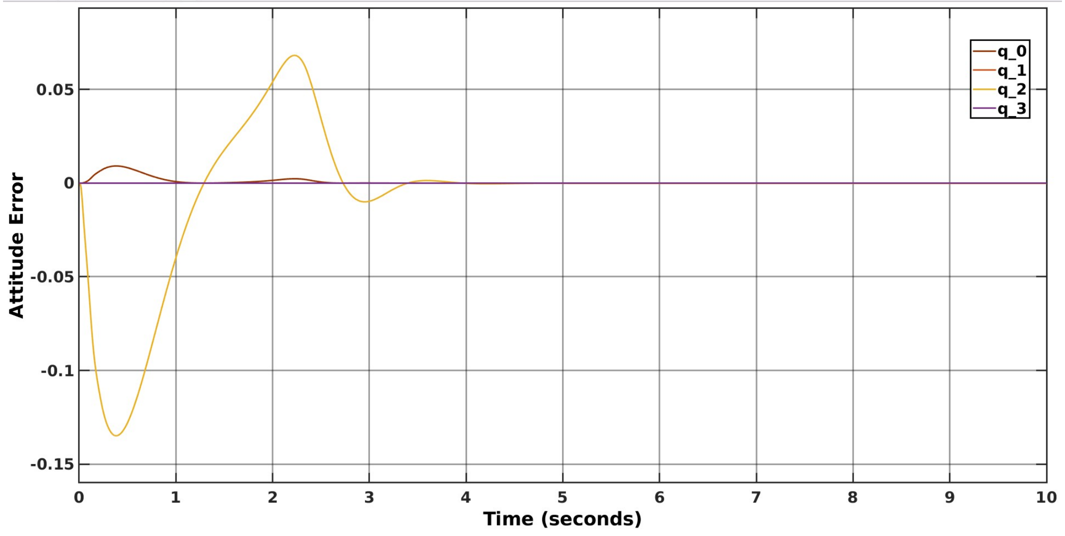
Table 5.3: Numerical simulation parameters for landing manoeuvre.

To assess the effectiveness of the L1AC the simulation was first run simply implementing the NMPC. At this point the ground effect was added and the performance of the NMPC was tested again in such condition. Finally the L1AC was added to the control scheme, with as expected result a restoration of the initial performance of the NMPC without disturbances thanks to the ability of the L1AC to compensate for external disturbances and model errors.
In the Simulink the ground effect block was set after the saturation block that limited the sum of the controller inputs according to the constraints in (4.2.11). This decision was motivated by the fact that this phenomenon, although generating thrust, does not depend on the capabilities of the quadrotor.
In *Figure* 5.3 we can see the performance of the NMPC alone, when the ground effect in not effecting the quadrotor performance and having the L1AC still inactive. This simulation is meant to give us an idea of the optimal performance for the NMPC. Here we can see in *Figure* 5.3a that the position error converges quickly to zero and that there is no overshoot. This is especially important in this task, since an overshoot in a landing manoeuvre could bring to a crash of the quadrotor.
Next, in *Figure* 5.3b, we can appreciate how the attitude error stay close to zero at all times. In fact the UAV is set in same orientation as the reference in the beginning and changes it only slightly to increase change its position, since it is

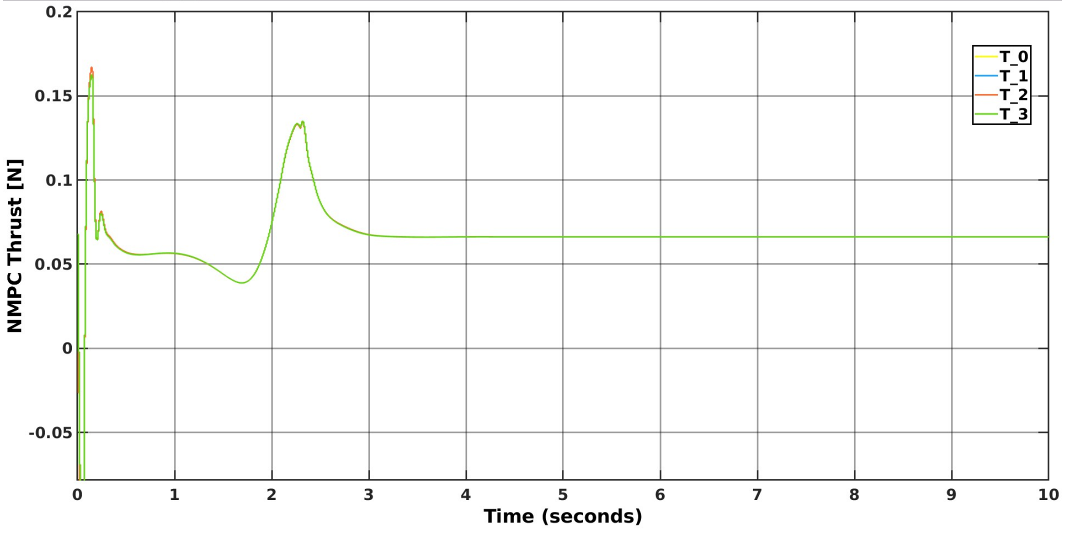
under-actuated.



(a) Position error



(b) Attitude error



(c) Thrust force

Figure 5.3: NMPC test for quadrotor landing without ground effect.

In *Figure* 5.3c we see the individual thrusts exerted by each individual propeller. The NMPC output acts strongly in the beginning to accelerate the platform, but then stabilizes around the steady state, moving along with the UGV. This can be considered a successful performance in tracking the UGV and would allow a smooth landing.
Next we focus on the two other cases. In the first one the only change is the addition of the ground effect. As mentioned before in *Section* 3.1 this effect amplifies the input thrust and pushes the quadrotor upwards. The NMPC is extremely precise in this case as well, having a performance that would look nearly identical to the previous case, especially thanks to the varying weights implemented using (3.1.7).
The only notable difference is in the steady-state of the altitude error, since it is in the direction the ground effect is acting upon. Here, although the error is still small, it amounts to about nine times what it was before the ground effect was added. Finally the simulation was run a third time, adding the L1AC to compensate for the ground effect. The results showed a perfect correction of the disturbance and brought the altitude steady-state error down to the initial values, making the performances virtually the same. These results can be observed in *Figure* 5.5, where the correction that L1AC achieves is extremely clear.
In the simulations of the next sections the ground effect will always be present to create a more realistic environment.

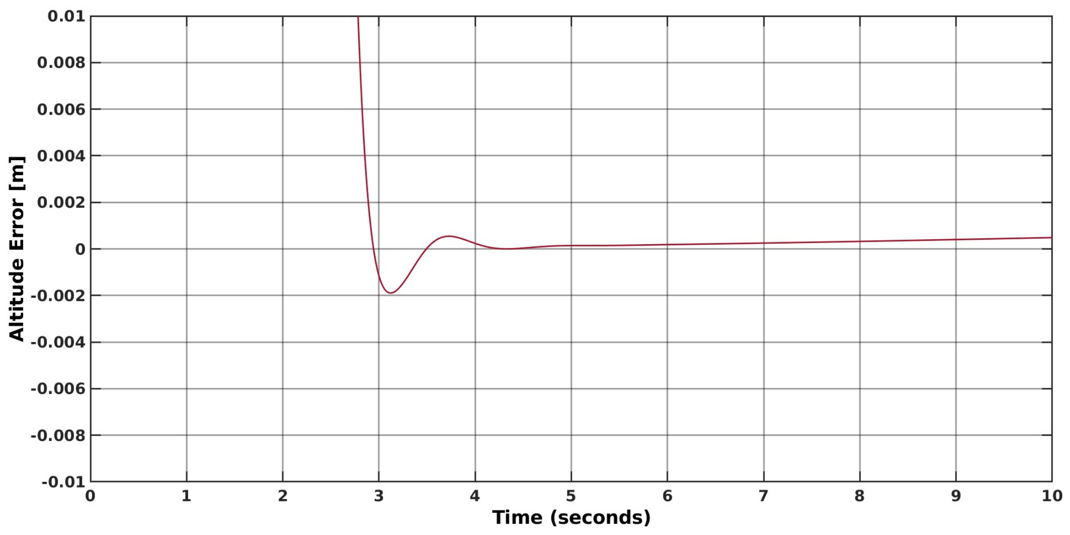
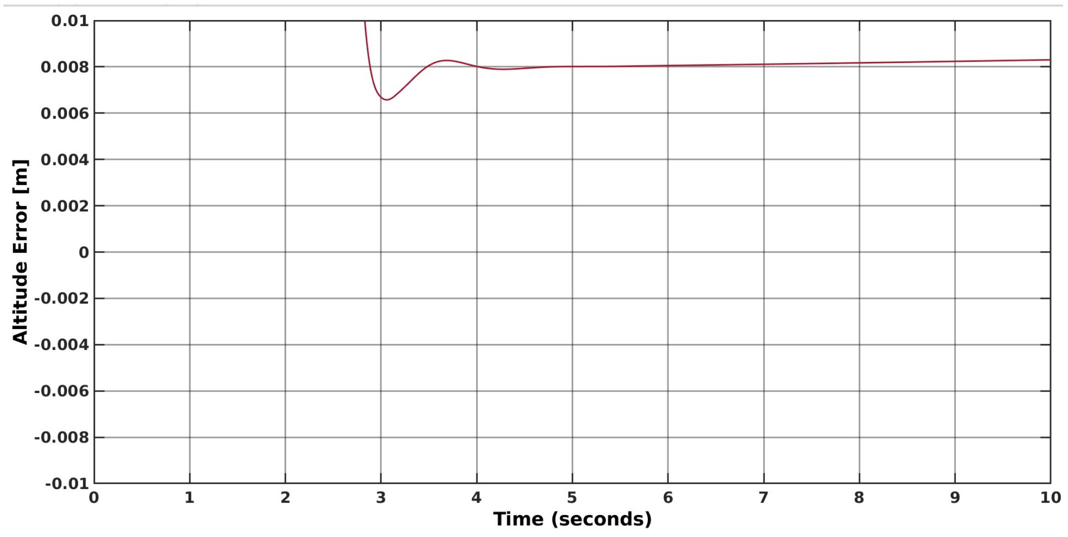
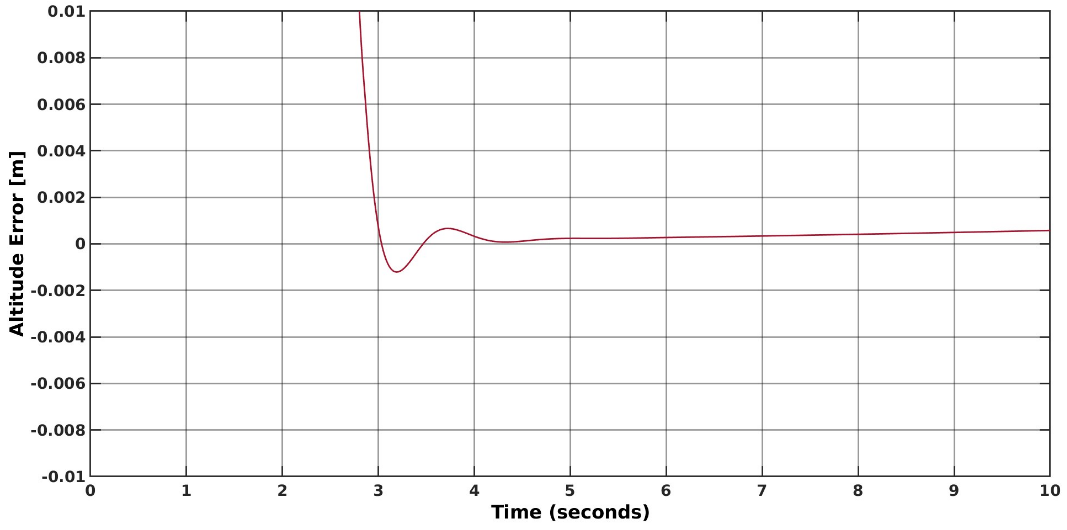


Figure 5.4: NMPC without ground effect.



(a) NMPC with ground effect

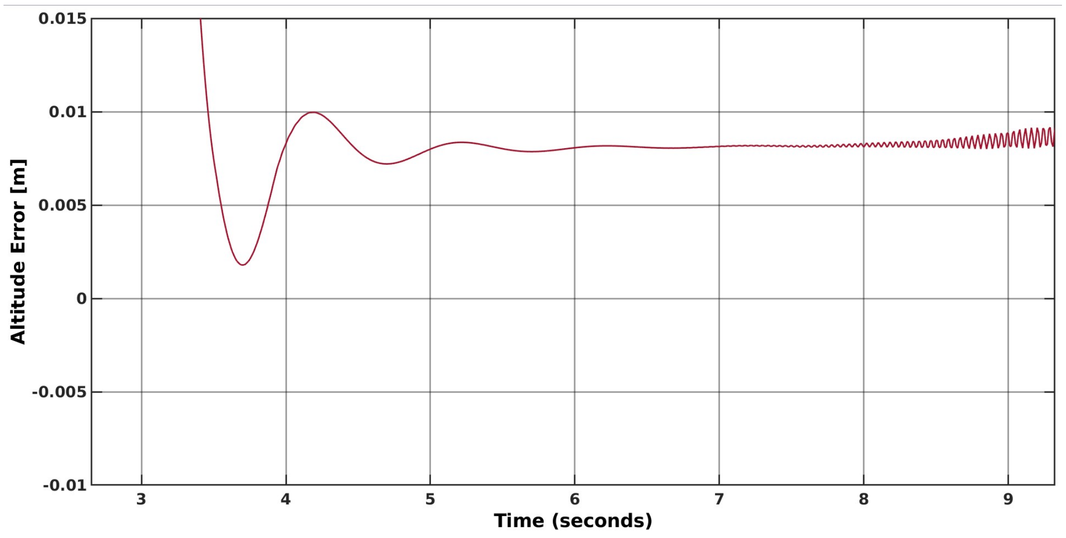


(b) NMPC with ground effect and L1AC

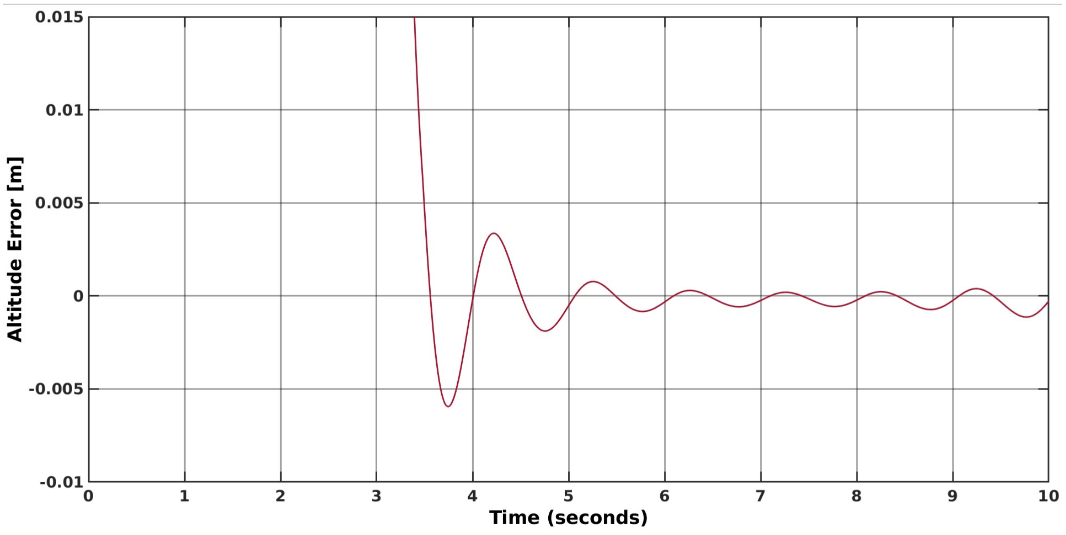
Figure 5.5: Juxtaposition of the altitude errors for the cases of NMPC and L1AC when the ground effect is considered.

5.3 Landing Simulation at High Speed

This section is dedicated at the exploration of the possibility of pushing the quadrotor to faster speeds. This test was meant to check whether the control, in particular the base NMPC, would work at different reference velocities than the preset one, while keeping the control law the same. The reference speed was increased from 0.5 m/s to 1.5 m/s and the simulation was run in the same manner of the one in *Section* 5.2.



(a) NMPC with high reference speed



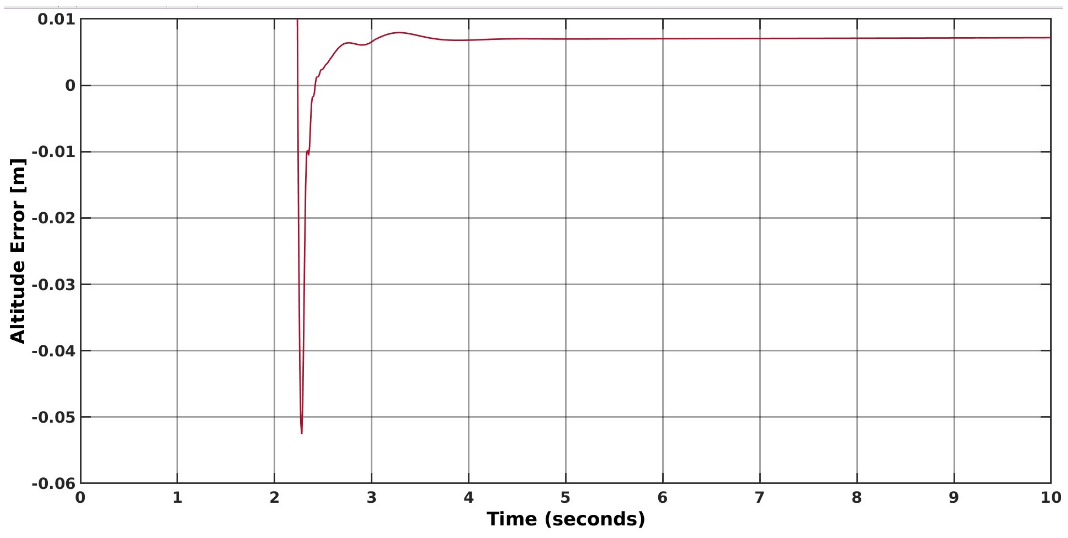
(b) L1AC with high reference speed

Figure 5.6: Altitude error of the quadrotor with high reference speed.

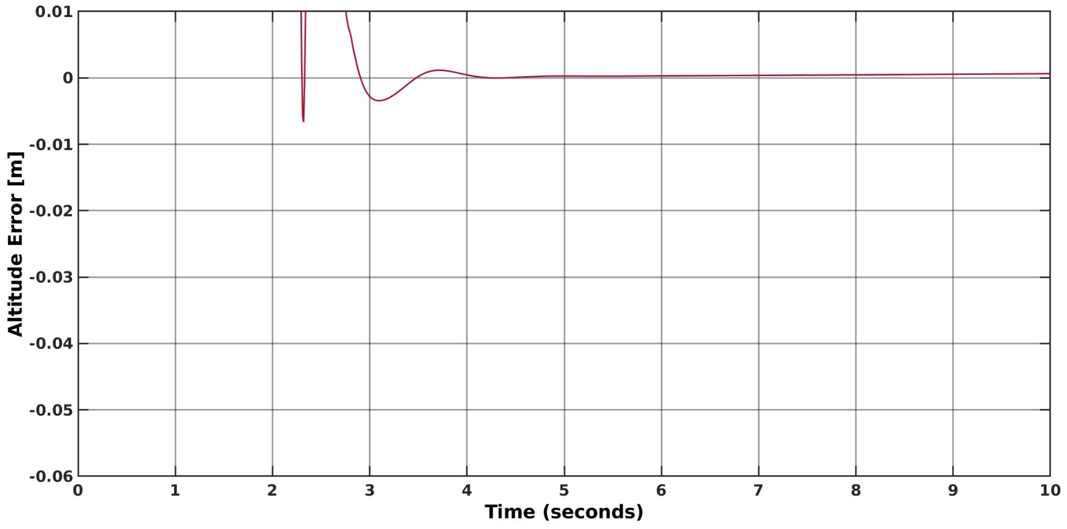
Looking at the results we can see in *Figure* 5.6a-5.6b that the increased velocity caused some oscillations in both cases. Like in the previous cases, it is clear that although the NMPC was able to follow the reference trajectory in a satisfactory manner. The L1AC was able to bring the altitude error to zero and kept oscillating with an amplitude similar to the NMPC alone. The only downside to the L1AC in this case is a slight overshoot, mostly due to the aggressive behaviour necessary to track such a reference state and the fact that it was able to neutralize the ground effect, bringing the quadrotor closer to the platform.

5.4 Landing Simulation with Additional Weight

Next we want to test our algorithm when the quadrotor is carrying additional weight. This is not an unlikely case, since quadrotors may have hardware attached to them that was not taken into account when setting the internal model parameters. Hence, this is also a test to the robustness of the L1AC to model parameter errors.
To accomplish this task the weight of the quadrotor was increased to twice the original weight. This increase in weight was chosen since it would have significant effect on the algorithm performance, while remaining reasonable, given that the quadrotor maximum thrust was able support a maximum of 2.6 times the original weight.



(a) NMPC with additional weight



(b) L1AC with additional weight

Figure 5.7: Performance of the NMPC and L1AC when the weight of the quadrotor is greater than expected.

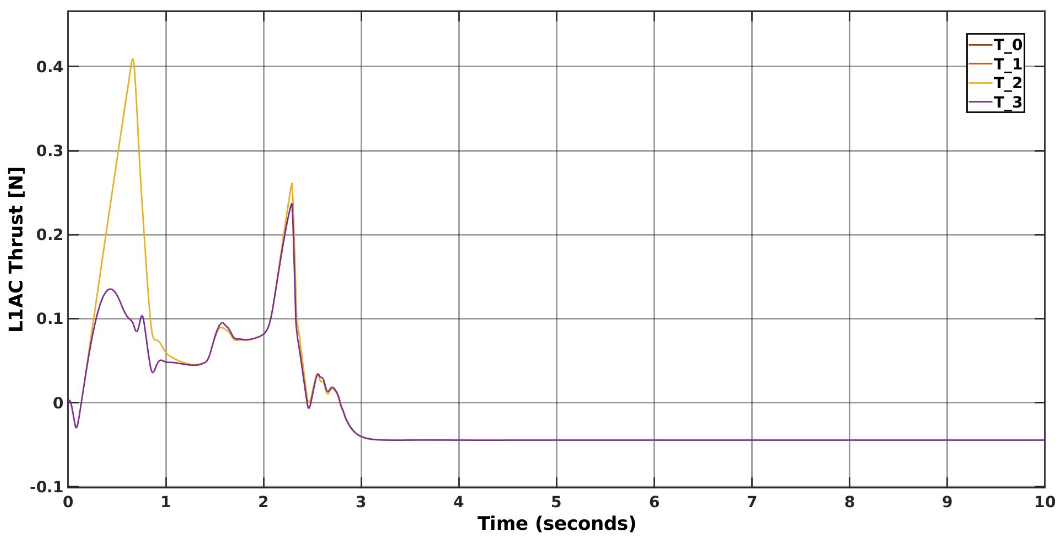


Figure 5.8: L1AC thrust with additional weight.

Looking at the results of the simulations in *Figure* 5.7b we can see the performance of the NMPC alone. The algorithm is again able to compensate for the additional weight thanks to its robustness, however we can notice that it overshot the reference altitude by considerable amount. This can be problematic especially if we are aiming for precise landing and may cause damage to the quadrotor on impact.
On the other side, in *Figure* 5.7b the L1AC is able to follow the reference altitude much better, although there is some instability in the beginning, the overshoot is much smaller than the previous case, by a significant amount.
Finally, in *Figure* 5.8, we can have a look at the L1AC input thrust during the landing. It is clear how in the beginning the controller compensates for the additional weight up to around 2.5 seconds. After that point it starts compensating with negative thrust since the ground effect has surpassed the effect of the extra weight. At this point we see the input stabilize.

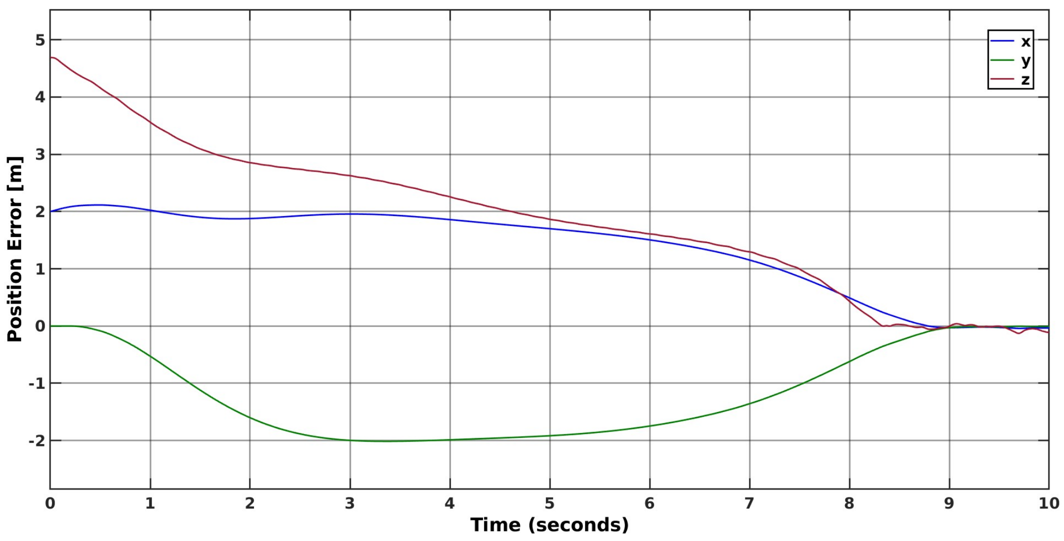
5.5 Landing Simulation with Broken Rotor

The last simulation is a test of an emergency situation, where one propeller of the quadrotor is partially broken. In this case it was not sensible to test a completely broken propeller, because of the mechanics that regulate L1AC. In fact, in the event of a completely broken propeller, the L1AC would try to increase the broken propeller thrust without any success. In this case we simulated the inconvenience as a loss of a certain percentage of the selected rotor thrust. This was achieved in the following manner

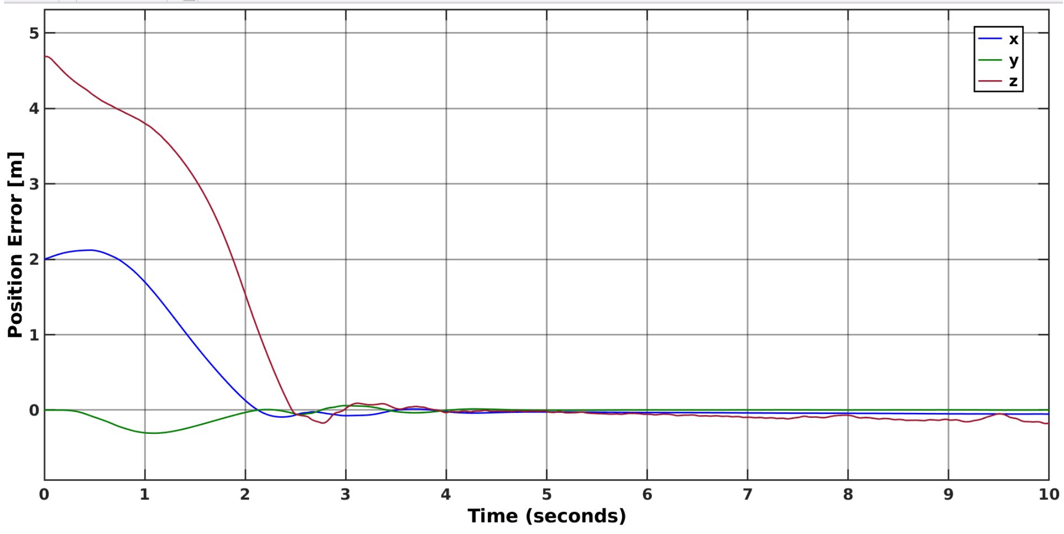
$$\boldsymbol{T}_{break} = diag([1,1,1,0.5])[T_0, T_1, T_2, T_3] \tag{5.5.1}$$

giving the 4-th rotor a deficit of 50% of its input thrust. This caused great problems in the control of the aircraft along with an unstable behaviour when compared with the other tests, although being an emergency scenario, it was expected.

The results are depicted in *Figure* 5.9. In particular we can observe the extremely long time it took the NMPC to bring the quadrotor to the UGV in *Figure* 5.9a. The NMPC proved again its robustness and avoided crashing the UAV. In *Figure* 5.9b we can instead appreciate how the L1AC is able to bring the quadrotor down in a much quicker time, drastically improving the performance of the NMPC alone.



(a) NMPC with broken propeller



(b) L1AC with broken propeller

Figure 5.9: Performance of the NMPC and L1AC during emergency landing with broken propeller.

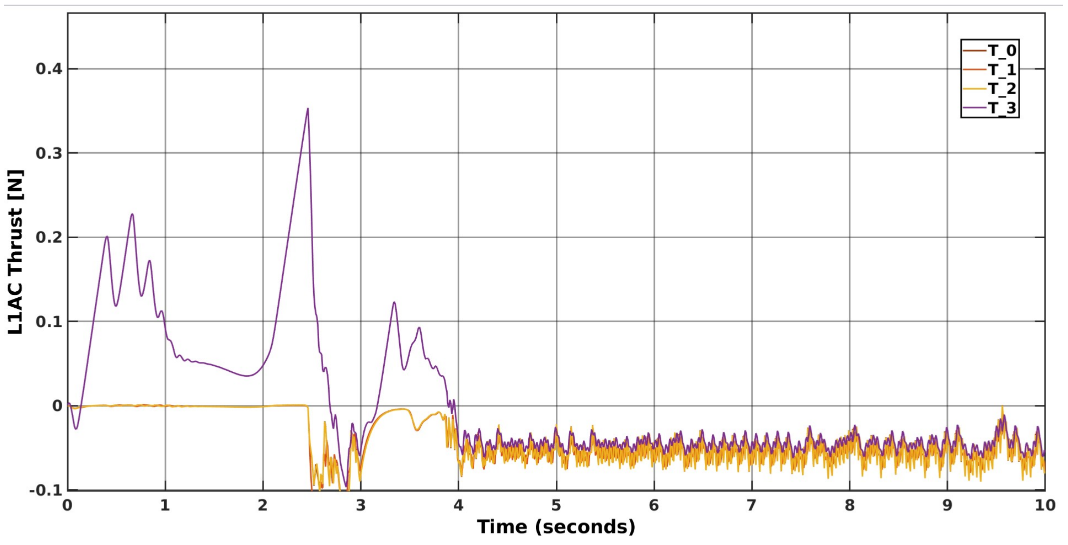


Figure 5.10: Input thrust of the L1AC for quadrotor with broken propeller.

In *Figure* 5.10 the L1AC input is represented. We can clearly see how the controller tries to compensate for the damaged propeller and exert a stronger thrust on the 4-th rotor only.

Chapter 6

Conclusion

In this thesis the control of a quadrotor via non-linear MPC and L1 Adaptive Control have been studied, with special attention for the landing manoeuvre and the challenges that are associated with it. The aircraft was tasked with performing such landing on a moving platform situated on a UGV, which had the role of the leader in the leader-follower relationship between the two unmanned vehicles.
Although NMPC is a form of optimal control and can achieve great robustness, it is heavily dependent on the internal model of the quadrotor. The L1AC, on the other hand, can be used to compensate the model deficiencies that may occur by compensating them at the input via an observer estimation. Moreover, the L1AC does not require heavy calculations, in contrast with the NMPC, since it does not perform an optimization procedure, but instead makes use of a closed form solution. For these reasons the coupling between the two control methods works well and the already well-performing NMPC can be improved with the L1AC.
The control methods were tested in different scenarios, to validate the improvements that the L1AC could add to the NMPC. To that end each test was performed first with the NMPC, to appreciate its performance. Then a second simulation was run with the addition of the adaptive control and the results were compared to evaluate the actual change in converging speed or steady-state error. Rejection of the ground effect, which is caused by the propellers of a quadrotor when in close vicinity to the ground, as well as different speeds of the UGV during the landing manoeuvre. Next the control was tested against errors in the model parameters of the quadrotor, which are essential for the NMPC. Finally a test of an emergency situation was run, to see how a damaged propeller could influence the results.
This thesis provides a few starting points for future works:

- We tested and validated control methods based on NMPC and the corresponding optimization procedure on Matlab & Simulink, where the time constraints that a real-time control were not an issue. A future work could try to find a way to implement these techniques in real time and obeying the relative constraints, see [41], and validate them in another real-time environment such as ROS Gazebo and then in the laboratory.

- This thesis assumed that the UGV was simply moving at constant velocity and that the quadrotor should try to land without any help from the leader. A future work could combine the actions of the two vehicles, since the NMPC gives us the tools to do so.
- In this work a single quadrotor was simulated, but the problem could be extended to a more complex case where multiple quadrotors have to coordinate their landing and take off from the UGV

Appendix A

Appendix

A.1 Time derivative of rotation matrices

In this section will explain how to obtain the time derivative of a rotation matrix $\boldsymbol{R}_{W,b} \in \mathbb{SO}(3)$, which represents the orientation of the body fixed frame $\mathfrak{F}_b$ with respect to the world inertial frame $\mathfrak{F}_W$ when a angular rate $\boldsymbol{\omega}$ is applied. We recommend [46] for more details. Considering a point P in the 3D space, its coordinates, expressed in an inertial frame, are denoted with the vector $\boldsymbol{p} \in \mathbb{R}^3$. Assuming that P possesses an angular rate $\boldsymbol{\omega} \in \mathbb{R}^3$, that has axis $\boldsymbol{\omega}/\|\boldsymbol{\omega}\| \in \mathbb{S}^2$ passing though the origin of the inertial frame and magnitude $\|\boldsymbol{\omega}\| > 0$.

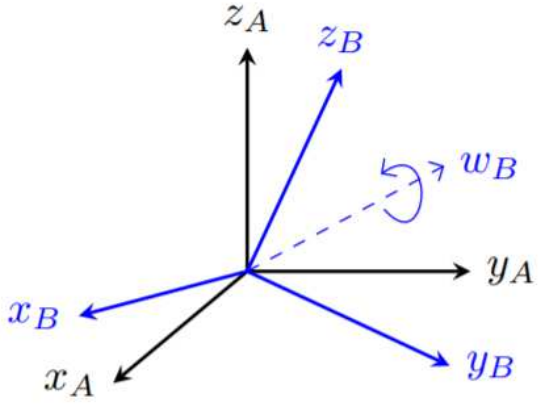


Figure A.1: Frame A is fixed while frame B is rotating.

The position vector $\boldsymbol{p} \in \mathbb{R}^3$ can then be derived as

$$\dot{\boldsymbol{p}} = \boldsymbol{\omega} \times \boldsymbol{p}. \tag{A.1.1}$$

Consider now, $\boldsymbol{e}_{w,i} \in \mathbb{S}^2$, $i = 1, 2, 3$ the vectors representing the canonical vectors $\boldsymbol{e}_i \in \mathbb{S}^2$, axes of frame $\mathfrak{F}_b$, w.r.t. the inertial frame $\mathfrak{F}_W$, namely $\boldsymbol{e}_{w,i} = \boldsymbol{R}_{W,b}\boldsymbol{e}_i$. Then, if we call $\boldsymbol{\omega}_w \in \mathbb{R}^3$ the value of the angular rate $\boldsymbol{\omega}$ expressed in the inertial frame. Then

$$\dot{\boldsymbol{e}}_{w,i} = \boldsymbol{\omega}_w \times \boldsymbol{e}_{w,i}. \tag{A.1.2}$$

By calling $\boldsymbol{e}_{w,i}$, where $i = 1, 2, 3$, the i-th column of $\boldsymbol{R}_{W,b}$ and recalling the properties of the skew-symmetric matrices, we find that

$$\dot{\boldsymbol{R}}_{W,b} = [\boldsymbol{\omega}_w]_\times \boldsymbol{R}_{W,b} \tag{A.1.3}$$

$\dot{\boldsymbol{R}}_{W,b}$ is the time derivative of the rotation matrix that expresses $\mathfrak{F}_b$ with respect to the frame $\mathfrak{F}_W$, assuming the angular rate is expressed in $\mathfrak{F}_W$ instead. Note that (A.1.3) still holds even if both frames are not inertial, since this fact was not exploited in this reasoning.

A.2 Time derivative of unit quaternions

The following proposition is taken from [47], which we recommend for more details.

Proposition A.2.1. *Given the unit quaternion $\mathfrak{q}_i \in \mathbb{S}^3$ representing the orientation of $\mathfrak{F}_i$ w.r.t. $\mathfrak{F}_W$, its time derivative has expression*

$$\dot{\mathfrak{q}}_i = \frac{1}{2}\boldsymbol{\omega}_w^+ \circ \mathfrak{q}_i \tag{A.2.1}$$

where $\boldsymbol{\omega}_w^+ = [0\ \boldsymbol{\omega}_w^T]^T \in \mathbb{R}^4$, with $\boldsymbol{\omega}_w \in \mathbb{R}^3$ an angular rate expressed in $\mathfrak{F}_W$.

Proof. Let $\boldsymbol{x}_0^+ \in \mathbb{R}^4$ be any given vector (quaternion with zero scalar part) fixed at time t_0 and $\boldsymbol{x}_t^+ \in \mathbb{R}^4$ the same vector at time t. The two can be related through the unit quaternion $\mathfrak{q}_t$,

$$\boldsymbol{x}_t^+ = \mathfrak{q}_t \circ \boldsymbol{x}_0^+ \circ \mathfrak{q}_t^{-1}. \tag{A.2.2}$$

Differentiating (A.2.2), it results

$$\dot{\boldsymbol{x}}_t^+ = \dot{\mathfrak{q}}_t \circ \boldsymbol{x}_0^+ \circ \mathfrak{q}_t^{-1} + \mathfrak{q}_t \circ \boldsymbol{x}_0^+ \circ \dot{\mathfrak{q}}_t^{-1}. \tag{A.2.3}$$

From equations (A.2.2) and (A.2.3) it holds

$$\dot{\boldsymbol{x}}_t^+ = \dot{\mathfrak{q}}_t \circ \mathfrak{q}_t^{-1} \circ \boldsymbol{x}_0^+ + \boldsymbol{x}_0^+ \circ \mathfrak{q}_t \circ \dot{\mathfrak{q}}_t^{-1}. \tag{A.2.4}$$

Since the norm of quaternion is unit $\mathfrak{q}_t \circ \mathfrak{q}_t^{-1} = 1$, we have

$$\frac{d}{dt}\mathfrak{q}_t \circ \mathfrak{q}_t^{-1} = \dot{\mathfrak{q}}_t \circ \mathfrak{q}_t^{-1} + \mathfrak{q}_t \circ \dot{\mathfrak{q}}_t^{-1} = 0. \tag{A.2.5}$$

It follows from equations (A.2.4) and (A.2.5) that

$$\dot{\boldsymbol{x}}_t^+ = \dot{\mathbf{q}}_t \circ \mathbf{q}_t^{-1} \circ \boldsymbol{x}_0^+ - \boldsymbol{x}_0^+ \circ \dot{\mathbf{q}}_t \circ \mathbf{q}_t^{-1}. \tag{A.2.6}$$

Define $\boldsymbol{p}^+ = \dot{\mathbf{q}}_t \circ \mathbf{q}_t^{-1}$, then, denoting with $\eta(\mathbf{q})$ and $\epsilon(\mathbf{q})$ the scalar and imaginary parts of the quaternion $\mathbf{q} \in \mathbb{S}^3$, we have

$$\begin{aligned}\eta(\boldsymbol{p}^+) &= \eta(\dot{\mathbf{q}}_t)\eta(\mathbf{q}_t^{-1}) - \epsilon(\dot{\mathbf{q}}_t)^T\epsilon(\mathbf{q}_t^{-1}) \\ &= \eta(\dot{\mathbf{q}}_t)\eta(\mathbf{q}_t) - \epsilon(\dot{\mathbf{q}}_t)^T\epsilon(\mathbf{q}_t) = 0,\end{aligned} \tag{A.2.7}$$

because the norm of $\mathbf{q}_t$ is unit and $\eta(\boldsymbol{p}^+) = \frac{1}{2}\frac{d}{dt}\|\mathbf{q}_t\|$. Then, $\eta(\boldsymbol{p}^+)$ is a pure quaternion, and being also $\boldsymbol{x}_t^+$ a pure quaternion, it holds that

$$\dot{\boldsymbol{x}}_t^+ = \boldsymbol{p}_t^+ \circ \boldsymbol{x}_t^+ - \boldsymbol{x}_t^+ \circ \boldsymbol{p}_t^+ = \begin{bmatrix} 0 \\ \\ 2(\boldsymbol{p}_t \times \boldsymbol{x}_t) \end{bmatrix}. \tag{A.2.8}$$

On the other and, $\dot{\boldsymbol{x}}_t \in \mathbb{R}^3$ has expression

$$\dot{\boldsymbol{x}}_t = \boldsymbol{\omega} \times \boldsymbol{x}_t \tag{A.2.9}$$

being the time derivative of a vector with fixed length. Putting together equations (A.2.8) and (A.2.9) it follows that

$$\boldsymbol{\omega}_t^+ = 2\boldsymbol{p}_t^+ = 2\dot{\mathbf{q}}_t \circ \mathbf{q}_t^{-1} \tag{A.2.10}$$

thus,

$$\dot{\mathbf{q}}_t = \frac{1}{2}\boldsymbol{\omega}_t^+ \circ \mathbf{q}_t. \tag{A.2.11}$$

□

Bibliography

[1] G.D. Goh, S. Agarwala, G.L. Goh, V. Dikshit, S.L. Sing, and W.Y. Yeong. Additive manufacturing in unmanned aerial vehicles (uavs): Challenges and potential. *Aerospace Science and Technology*, 63:140–151, 2017. ISSN 1270-9638. doi: https://doi.org/10.1016/j.ast.2016.12.019. URL `https://www.sciencedirect.com/science/article/pii/S127096381630503X`. (Cited at page 1)

[2] Guillaume J.J. Ducard and Mike Allenspach. Review of designs and flight control techniques of hybrid and convertible vtol uavs. *Aerospace Science and Technology*, 118:107035, 2021. ISSN 1270-9638. doi: https://doi.org/10.1016/j.ast.2021.107035. URL `https://www.sciencedirect.com/science/article/pii/S1270963821005459`. (Cited at page 1)

[3] Sophie Jordan, Julian Moore, Sierra Hovet, John Box, Jason Perry, Kevin Kirsche, Dexter Lewis, and Zion Tsz Ho Tse. State-of-the-art technologies for uav inspections. *IET Radar, Sonar & Navigation*, 12(2):151–164, 2018. doi: https://doi.org/10.1049/iet-rsn.2017.0251. URL `https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/iet-rsn.2017.0251`. (Cited at page 1)

[4] Yuncheng Lu, Zhucun Xue, Gui-Song Xia, and Liangpei Zhang. A survey on vision-based uav navigation. *Geo-spatial Information Science*, 21(1):21–32, 2018. doi: https://doi.org/10.1080/10095020.2017.1420509. (Cited at page 1)

[5] Christopher M. Korpela, Todd W. Danko, and Paul Y. Oh. Mm-uav: Mobile manipulating unmanned aerial vehicle. *Journal of Intelligent and Robotic Systems*, 65:93–101, 2012. doi: https://doi.org/10.1007/s10846-011-9591-3. (Cited at page 1)

[6] Boris Galkin, Jacek Kibilda, and Luiz A. DaSilva. Uavs as mobile infrastructure: Addressing battery lifetime. *IEEE Communications Magazine*, 57 (6):132–137, 2019. doi: 10.1109/MCOM.2019.1800545. (Cited at page 1)

[7] Marcos Felipe Santos Rabelo, Alexandre Santos Brandão, and Mário Sarcinelli-Filho. Landing a uav on static or moving platforms using a formation controller. *IEEE Systems Journal*, 15(1):37–45, 2021. doi: 10.1109/JSYST.2020.2975139. (Cited at page 1)

[8] Thien Hoang Nguyen, Muqing Cao, Thien-Minh Nguyen, and Lihua Xie. Post-mission autonomous return and precision landing of uav. In *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1747–1752, 2018. doi: 10.1109/ICARCV.2018.8581117. (Cited at page 1)

[9] Sepehr Seyedi, Yasin Yazicioğlu, and Derya Aksaray. Persistent surveillance with energy-constrained uavs and mobile charging stations. *IFAC-PapersOnLine*, 52(20):193–198, 2019. ISSN 2405-8963. doi: https://doi.org/10.1016/j.ifacol.2019.12.157. URL `https://www.sciencedirect.com/science/article/pii/S2405896319320087`. 8th IFAC Workshop on Distributed Estimation and Control in Networked Systems NECSYS 2019. (Cited at page 1)

[10] Maidul Islam, Mohamed Okasha, and Erwin Sulaeman. A model predictive control (mpc) approach on unit quaternion orientation based quadrotor for trajectory tracking. *International Journal of Control, Automation and Systems*, 17:2819–2832, 2019. doi: https://doi.org/10.1007/s12555-018-0860-9. (Cited at page 1)

[11] Abolfazl Eskandarpour and Inna Sharf. A constrained error-based mpc for path following of quadrotor with stability analysis. *Nonlinear Dynamics*, 99: 899–918, 2020. doi: https://doi.org/10.1007/s11071-019-04859-0. (Cited at page 1)

[12] Tong Lv, Yanhua Yang, and Li Chai. Extended state observer based mpc for a quadrotor helicopter subject to wind disturbances. In *2019 Chinese Control Conference (CCC)*, pages 8206–8211, 2019. doi: https://doi.org/10.23919/ChiCC.2019.8865370. (Cited at page 1)

[13] Nicola Lissandrini, Giulia Michieletto, Riccardo Antonello, Marta Galvan, Alberto Franco, and Angelo Cenedese. Cooperative optimization of uavs formation visual tracking. *Robotics*, 8(3), 2019. ISSN 2218-6581. doi: 10.3390/robotics8030052. URL `https://www.mdpi.com/2218-6581/8/3/52`. (Cited at page 1)

[14] Elio Tuci, Muhanad HM Alkilabi, and Otar Akanyeti. Cooperative object transport in multi-robot systems: A review of the state-of-the-art. *Frontiers in Robotics and AI*, 5:59, 2018. (Cited at page 1)

[15] Zhi Feng, Guoqiang Hu, Yajuan Sun, and Jeffrey Soon. An overview of collaborative robotic manipulation in multi-robot systems. *Annual Reviews in Control*, 49:113–127, 2020. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2020.02.002. URL `https://www.sciencedirect.com/science/article/pii/S1367578820300043`. (Cited at page 1)

[16] J.B. Rawlings, D.Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017. ISBN 9780975937730. URL `https://books.google.it/books?id=MrJctAEACAAJ`. (Cited at pages 2 and 19)

[17] Michael A. Henson. Nonlinear model predictive control: current status and future directions. *Computers Chemical Engineering*, 23(2):187–202, 1998. ISSN 0098-1354. doi: https://doi.org/10.1016/S0098-1354(98)00260-9. URL `https://www.sciencedirect.com/science/article/pii/S0098135498002609`. (Cited at page 2)

[18] Andrea Zanelli, Greg Horn, Gianluca Frison, and Moritz Diehl. Nonlinear model predictive control of a human-sized quadrotor. In *2018 European Control Conference (ECC)*, pages 1542–1547, 2018. doi: 10.23919/ECC.2018.8550530. (Cited at page 2)

[19] Bárbara Barros Carlos, Tommaso Sartor, Andrea Zanelli, Gianluca Frison, Wolfram Burgard, Moritz Diehl, and Giuseppe Oriolo. An efficient real-time nmpc for quadrotor position control under communication time-delay. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 982–989, 2020. doi: 10.1109/ICARCV50220.2020.9305513. (Cited at page 2)

[20] Dong Wang, Quan Pan, Yang Shi, Jinwen Hu, and Chunhui Zhao. Efficient nonlinear model predictive control for quadrotor trajectory tracking: Algorithms and experiment. *IEEE Transactions on Cybernetics*, 51(10):5057–5068, 2021. doi: 10.1109/TCYB.2020.3043361. (Cited at page 2)

[21] Ioan Doré Landau, Rogelio Lozano, Mohammed M'Saad, and Alireza Karimi. *Adaptive control: algorithms, analysis and applications*. Springer Science & Business Media, 2011. (Cited at page 2)

[22] Anuradha M Annaswamy and Alexander L Fradkov. A historical perspective of adaptive control and learning. *Annual Reviews in Control*, 52:18–41, 2021. (Cited at page 2)

[23] AP Morgan and KS Narendra. On the uniform asymptotic stability of certain linear nonautonomous differential equations. *SIAM Journal on Control and Optimization*, 15(1):5–24, 1977. (Cited at page 2)

[24] Kumpati S Narendra and Anuradha M Annaswamy. Persistent excitation in adaptive systems. *International Journal of Control*, 45(1):127–160, 1987. (Cited at page 2)

[25] Changyun Wen and David J Hill. Adaptive linear control of nonlinear systems. *IEEE transactions on automatic control*, 35(11):1253–1257, 1990. (Cited at page 2)

[26] Joseph E Gaudio, Anuradha M Annaswamy, and Eugene Lavretsky. Adaptive control of hypersonic vehicles in the presence of rate limits. In *2018 AIAA Guidance, Navigation, and Control Conference*, page 0846, 2018. (Cited at page 2)

[27] Zachary T Dydek, Anuradha M Annaswamy, and Eugene Lavretsky. Adaptive control of quadrotor uavs: A design trade study with flight evaluations. *IEEE Transactions on control systems technology*, 21(4):1400–1406, 2012. (Cited at page 2)

[28] Alia Abdul Ghaffar and Tom Richardson. Model reference adaptive control and lqr control for quadrotor with parametric uncertainties. *International Journal of Mechanical and Mechatronics Engineering*, 9(2):244–250, 2015. (Cited at page 2)

[29] Paul De Monte and Boris Lohmann. Position trajectory tracking of a quadrotor helicopter based on l1 adaptive control. In *2013 European Control Conference (ECC)*, pages 3346–3353. IEEE, 2013. (Cited at page 2)

[30] Saeid Jafari, Petros Ioannou, and Lael E Rudd. What is l1 adaptive control. In *AIAA guidance, navigation, and control (GNC) Conference*, page 4513, 2013. (Cited at page 2)

[31] Hossein Beikzadeh and Guangjun Liu. Trajectory tracking of quadrotor flying manipulators using l1 adaptive control. *Journal of the Franklin Institute*, 355(14):6239–6261, 2018. (Cited at page 2)

[32] Randy Beard, Chengyu Cao, and Naira Hovakimyan. An l1 adaptive pitch controller for miniature air vehicles. In *AIAA guidance, navigation, and control conference and exhibit*, page 6777, 2006. (Cited at page 2)

[33] Irene Gregory, Chengyu Cao, Enric Xargay, Naira Hovakimyan, and Xiaotian Zou. L1 adaptive control design for nasa airstar flight test vehicle. In *AIAA guidance, navigation, and control conference*, page 5738, 2009. (Cited at page 2)

[34] Xiaodong Zhang, Xiaoli Li, Kang Wang, and Yanjun Lu. A survey of modelling and identification of quadrotor robot. In *Abstract and Applied Analysis*, volume 2014. Hindawi, 2014. (Cited at page 12)

[35] Benoit Landry et al. *Planning and control for quadrotor flight through cluttered environments*. PhD thesis, Massachusetts Institute of Technology, 2015. (Cited at page 12)

[36] Ettore Fornasini and Giovanni Marchesini. *Appunti di teoria dei sistemi*. Libreria progetto, 2011. (Cited at page 20)

[37] Robert Mahony, Vijay Kumar, and Peter Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *IEEE Robotics Automation Magazine*, 19(3):20–32, 2012. doi: 10.1109/MRA.2012.2206474. (Cited at page 24)

[38] IC Cheeseman and WE Bennett. The effect of the ground on a helicopter rotor in forward flight. 1955. (Cited at page 24)

[39] Li Danjun, Zhou Yan, Shi Zongying, and Lu Geng. Autonomous landing of quadrotor based on ground effect modelling. In *2015 34th Chinese control conference (CCC)*, pages 5647–5652. IEEE, 2015. (Cited at page 25)

[40] Guanya Shi, Wolfgang Hönig, Yisong Yue, and Soon-Jo Chung. Neural-swarm: Decentralized close-proximity multirotor control using learned interactions. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3241–3247. IEEE, 2020. (Cited at page 31)

[41] Drew Hanover, Philipp Foehn, Sihao Sun, Elia Kaufmann, and Davide Scaramuzza. Performance, precision, and payloads: Adaptive nonlinear mpc for quadrotors. *IEEE Robotics and Automation Letters*, 7(2):690–697, 2021. (Cited at pages 33, 35 and 52)

[42] Jintasit Pravitra, Kasey A Ackerman, Chengyu Cao, Naira Hovakimyan, and Evangelos A Theodorou. 1-adaptive mppi architecture for robust and agile control of multirotors. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7661–7666. IEEE, 2020. (Cited at page 33)

[43] Mingcheng Liu, Fubiao Zhang, and Shuaipeng Lang. The quadrotor position control based on mpc with adaptation. In *2021 40th Chinese Control Conference (CCC)*, pages 2639–2644. IEEE, 2021. (Cited at page 33)

[44] Enric Xargay, Naira Hovakimyan, and Chengyu Cao. L 1 adaptive controller for multi-input multi-output systems in the presence of nonlinear

unmatched uncertainties. In *Proceedings of the 2010 American control conference*, pages 874–879. IEEE, 2010. (Cited at page 35)

[45] Yutao Chen, Mattia Bruschetta, Enrico Picotti, and Alessandro Beghi. Matmpc-a matlab based toolbox for real-time nonlinear model predictive control. In *2019 18th European Control Conference (ECC)*, pages 3365–3370. IEEE, 2019. URL `https://github.com/chenyutao36/MATMPC`. (Cited at page 37)

[46] Shiyu Zhao. Time derivative of rotation matrices: A tutorial. *arXiv preprint arXiv:1609.06088*, 2016. (Cited at page 53)

[47] Basile Graf. Quaternions and dynamics. *arXiv preprint arXiv:0811.2889*, 2008. (Cited at page 54)