UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIEROS INDUSTRIALES VALENCIA

TRABAJO FIN DE MASTER EN INGENIERÍA INDUSTRIAL

DEVELOPMENT OF A QUATERNION-BASED QUADROTOR CONTROL SYSTEM BASED ON DISTURBANCE REJECTION

AUTOR: Alberto Castillo Frasquet

TUTOR: Pedro García Gil

Curso Académico: 2015-16

This work was partially supported by projects
PROMETEOII/2013/004, Conselleria de Educacion, and
TIN2014-56156-C4-4-P-AR, Ministerio de Economia y Competitividad

Development of a quaternion-based quadrotor control system based on disturbance rejection.

Castillo Frasquet, Alberto

July 7, 2016

Abstract

In the present document the use of quaternions as a tool for solving the attitude control problem for aerial vehicles is presented and discussed. Its main properties are shown and some of them are also demonstrated. Moreover, this document details how to solve an important problem involving quaternions, which is how to obtain a real-time quaternion measurement from low-cost sensors employing C++ as programming language. MatLab and C++ code is given through the document. Its performance is also shown including an experiment with real sensor data.

Furthermore, a quaternion-based attitude control system for aerial vehicles is developed. The proposal is made up of two interconnected controllers: an outer loop, which consists of a nonlinear quaternion-based controller, deals with the kinematic model; the inner loop is based on the Active Disturbance Rejection Control and deals with the dynamic model. The stability of each controller is proved and the global performance is illustrated through several simulations. This controller is particularized for quadrotors, but it can be easily generalized for any Remotely Piloted Aircraft Systems.

Index terms— quaternion, quadrotor, attitude, ADRC, Wahba, cascade

Contents

List of Figures

Part I
Main report

1 Introduction

This document corresponds to the final project of the following master: Master en Ingeniería Industrial. Imparted by Universidad Politécnica de Valencia during the years 2014-2016.

In the present document three important problems involving Remotely Piloted Aircraft Systems (RPAS) are treated. First, the use of quaternions as a tool for solving the attitude control problem is presented and discussed. Its main properties are shown and some of them are also demonstrated.

Second, a quaternion-based attitude control system for aerial vehicles is developed. The proposal is made up of two interconnected controllers: an outer loop, which consists of a nonlinear quaternion-based controller and it deals with the kinematic model; the inner loop, which is based on the Active Disturbance Rejection Control [23] deals with the dynamic model. The stability of each controller is proved and the global performance is illustrated through several simulations. This controller is particularized for quadrotors but it can be easily generalized for any other kind of RPAS.

Finally, this document also details how to solve an important problem involving quaternions. The problem is how to obtain a real-time quaternion measurement from low-cost sensors employing C++ as programming language. An algorithm written in MatLab and C++ is given, and its performance is also shown including some experiments with real sensor data. For this part, a quadrotor platform has been used in order to extract the sensor data. This quadrotor was developed in previous works by Pedro García Gil and its team [1–4].

1.1 Motivation and state of the art

The studies related to Remotely Piloted Aircraft Systems (RPAS, or also called UAVs) has become increasingly important in the last decades. Those vehicles are undoubtedly helpful when performing some kind of tasks. Specially, when an on-board pilot is unnecessary, or it is even dangerous. The RPAS area covers a wide number of different academic fields such as aerodynamics, control, signal processing or computer science. Any development on any of these fields could lead to a better performance of the whole system.

The European Union has allowed the use of RPAS systems in different sectors and applications such as: infrastructure inspection, atmospheric research, topography, management of risks and natural hazards, monitoring

of wind farms, movie filming, sport photography, environmental monitoring, and control of illegal hunting among others [5, 6].

The potential that this kind of vehicles offer has promoted the interest of the European Commission (EC). The EC has developed an strategy in order to integrate the RPAS into the European market. This integration has to safeguard the principles of security, privacy, reliability and public acceptance [7]. Furthermore, the EC contemplates the use of RPAS as an strategic sector to be developed in the program Horizon 2020 [8], which is a financial instrument for investment and innovation.

The final objective is the complete integration of the RPAS into the European airspace [5]. From this point of view, the security is a vital aspect and the RPAS control systems need to be improved as they need to deal with changing environments and with unforeseen situations. In general, the control systems need to be improved in order to guarantee some security standards and, in this sense, there is still some aspects that need to be solved.

The interest that the scientific community has in this field can be verified by the large number of papers and studies that have been published. Also, there are a large number of regular conferences and mono-graphic studies involving this area:

- 2015: 3rd Workshop on Research, Education and Development of Unmanned Aerial Systems. (http://eventegg.com/red-uas-2015/).
- 2015/2016: International Conference on Unmanned Aircraft Systems. (http://www.uasconferences.com/).
- UAV-g 2015 Conference: International Conference on Unmanned Aerial Vehicles in Geomatics. (www.uav-g-2015.ca/)

Specifically, there exists two important trending research topics:

- Development of new control strategies which are capable of rejecting unmodelled external disturbances.
- Development of new techniques of sensor data fusion in order to obtain robust and high quality measurements.

On the one hand, unmodelled external disturbances are really critical in this kind of vehicles. They use to fly under changing conditions such as wind or changes of weight. The on-board control needs to identify those situations and it needs to act as fast as possible in order to compensate for those disturbances. If the flight control system is not able to do that, the security requirements can not be warranted.

On the other hand, sensor fusion techniques are a powerful tool in order to improve the physical limitations of sensors (like random noise, incorrect measures, random biases, etc).

1.2 Objectives

As it has been mentioned. This work has three main objectives:

1. Describe and understand the use of quaternions as a mathematical tool for the RPAS attitude control.

2. To propose a new control strategy based on quaternions and Active Disturbance Rejection Control (ADRC). This control strategy will be able to deal with large external and unmodelled uncertainties.

3. To program a sensor fusion technique in order to get real-time quaternion measurements from low-cost sensors.

1.3 Results

At the end of the work it can be said that the three objectives have been properly completed. A new control scheme for quadrotors has been developed (objective 2). The main contribution is a cascade control with an external loop controlling the quadrotor kinematics taking into account the precise and simple kinematic model based on quaternions. The internal loop, designed by using the ADRC principles, considers the uncertain dynamic model of the quadrotor and rejects the external disturbances. Altogether, it allows a flight control of the quadrotor without singularities. The simulations results are very encouraging. Currently, the practical implementation of this control structure in a prototype is under development.

As a consequence of the development of this control strategy, a paper [9] has been accepted for the *International Conference on Information and Automation (ICIA)* and it will be published at the beginning of August, 2016.

Furthermore, the actual quadrotor platform has been provided with an algorithm which is able to provide quaternion measurements at a high frequency (objective 3). This algorithm receives as inputs the measurements of low-cost gyroscopes, magnetometers and accelerometers and provides a filtered and robust quaternion measurement.

1.4 Document structure

This document is structured as follows. Sections 2 and 3 corresponds to the objective 2. In those sections the quaternions are presented as an alternative to represent the attitude of a rigid body. Its main properties are also demonstrated. In Section 4 the new control scheme is developed. Some simulations are also included in order to show the control performance. This section corresponds to the objective 3. In section 5 the algorithm which provides quaternion measurements is described. MatLab and C++ code is also included.

2 Aerial vehicle attitude measure

The real situation where an automatically controlled aircraft is flying or moving through the environment can be abstracted and it can be seen from a pure mathematical point of view. It does not mind what kind of aerial vehicle we are referring to. This abstraction permits to formulate the control problem in a general way, and it is necessarily needed to develop controllers for aerial vehicles.

In this section, the mathematical principles which are necessary to understand the attitude control problem for aerial vehicles are presented.

2.1 The Inertial, the Body and the Reference frames

The first step is to create a coordinate system which is attached to the aerial vehicle. This frame is normally refereed as the *Body*, $\mathcal{B}$, frame. It moves and rotates with the aerial vehicle and, normally, all the onboard sensors (such as gyroscopes, accelerometers or magnetometers) express the information in $\mathcal{B}$. Furthermore, the origin (0, 0, 0) needs to be coincident with the body gravity center. Figure 1, shows an example of a *Body* frame attached to a quadrotor aerial vehicle

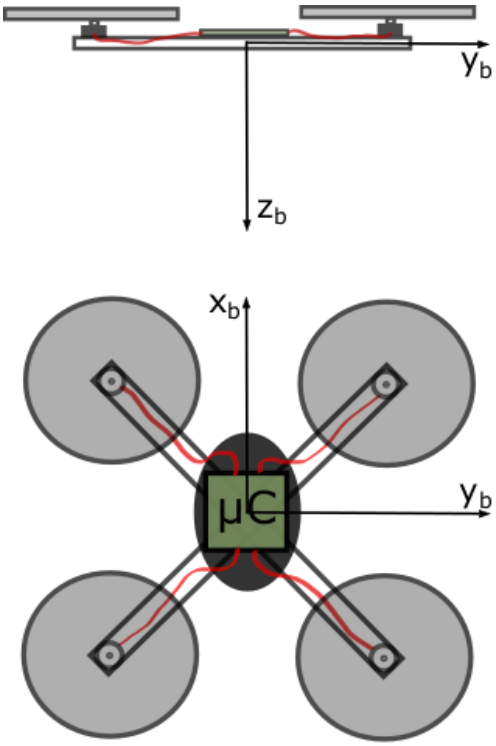

Figure 1: Body reference frame attached to a quadrotor aerial vehicle.

Since the aerial vehicle is moving and rotating in a fixed environment it is important to define a fixed coordinate frame also. Such frame is called the *Inertial*, $\mathcal{I}$, frame. The x-axis of $\mathcal{I}$ normally points to the north, the y-axis points to the east ant the z-axis points down. It is important to fix the axis with this directions because they are aligned with the principal earth magnitudes which are going to be measured (for example, the earth gravity points down, the earth magnetic field points up/down and north, the GPS measures are referred to north/south-east/west, etc). $\mathcal{I}$ can be placed in any arbitrary position, but it is normally placed in the take off or landing position.

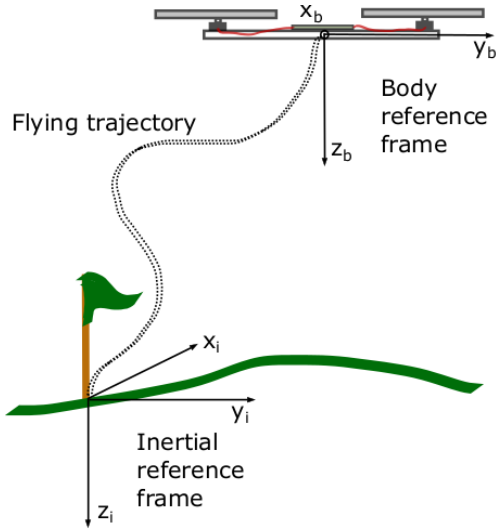

Figure 2: Inertial and Body references frames.

$\mathcal{B}$ and $\mathcal{I}$ defines the position and orientation of the aerial vehicle. However, since the aircraft is considered to be an autonomous (or semiautonomous) vehicle (i.e. it has an on-board control to perform autonomous flight or autonomous auto-stabilization), another coordinate frame needs to be defined. This coordinate frame is known as the *Reference*, $\mathcal{R}$, frame. The vehicle control algorithm will try to rotate and translate $\mathcal{B}$ in order to make it coincident with $\mathcal{R}$. The remote pilot will change the orientation and position of $\mathcal{R}$ via a remote control station, or via way-points, or via a trajectory generator, etc.

So from the point of view of the aerial control system, $\mathcal{B}$ has to follow an (apparently) randomly changing reference frame $\mathcal{R}$ (in reality the reference frame changes because the remote pilot changes it, but the on-board control does not know it). Furthermore the on-board control is able to compute its own position and orientation with respect to $\mathcal{I}$, and it receives as an input the position and orientation of $\mathcal{R}$ with respect to $\mathcal{I}$.

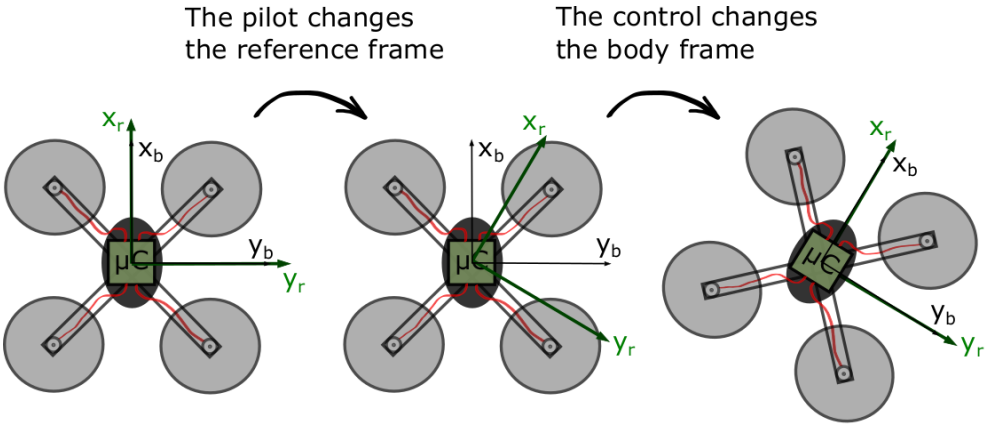

Figure 3: The on-board control forces the body frame to follow the reference frame.

This document treats only the attitude problem. So it is assumed that the control algorithm does not pay attention to the error in position and it only follows the orientation of the reference frame. However, the attitude control is the most important because, in many cases, the position control needs (beforehand) the attitude control to force position movements.

2.2 Rotation Matrix and Euler angles

Figure 4 shows two different coordinate systems, $(\vec{i}_0, \vec{j}_0, \vec{k}_0)$ and $(\vec{i}_1, \vec{j}_1, \vec{k}_1)$. So the first problem is how to represent the relative orientation between them. The rotation matrix, R, contains the information of the relative orientation between both frames. This matrix is constructed as follows.

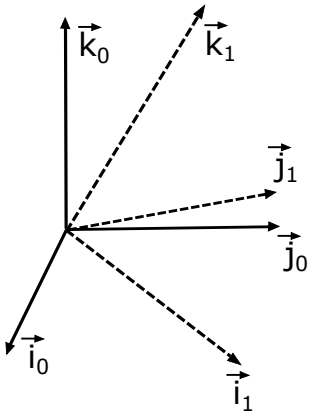

Figure 4: Rotated Cartesian coordinate frame.

$$R_0^1 = \begin{pmatrix} \vec{i}_1 \cdot \vec{i}_0 & \vec{i}_1 \cdot \vec{j}_0 & \vec{i}_1 \cdot \vec{k}_0 \\ \vec{j}_1 \cdot \vec{i}_0 & \vec{j}_1 \cdot \vec{j}_0 & \vec{j}_1 \cdot \vec{k}_0 \\ \vec{k}_1 \cdot \vec{i}_0 & \vec{k}_1 \cdot \vec{j}_0 & \vec{k}_1 \cdot \vec{k}_0 \end{pmatrix} = \begin{pmatrix} \vec{n}_1 & \vec{n}_2 & \vec{n}_3 \end{pmatrix} \tag{1}$$

where $(\cdot)$ denotes the dot product and $\vec{n}_1$, $\vec{n}_2$, $\vec{n}_3$ represents the vectors $\vec{i}_0$, $\vec{j}_0$, $\vec{k}_0$ as seen from the coordinate frame $(\vec{i}_1\, \vec{j}_1\, \vec{k}_1)$, respectively. R implicitly represents the relative orientation between both subsystems and it also has the following properties:

- If $v_0 = (v_{0,x}\, v_{0,y}\, v_{0,z})$ is a vector expressed in the frame $(\vec{i}_0\, \vec{j}_0\, \vec{k}_0)$ and $v_1 = (v_{1,x}\, v_{1,y}\, v_{1,z})$ is the same vector in $(\vec{i}_1\, \vec{j}_1\, \vec{k}_1)$, then $v_1 = Rv_0$.
- $det(R) = 1$
- $R^{-1} = R^T$

The composition of rotations can be stated from the first property. If R_0^1 is a rotation matrix that transforms the coordinates of v_0 from $(\vec{i}_0\, \vec{j}_0\, \vec{k}_0)$ to $(\vec{i}_1\, \vec{j}_1\, \vec{k}_1)$; and if R_1^2 transforms the coordinates of v_1 from $(\vec{i}_1\, \vec{j}_1\, \vec{k}_1)$ to $(\vec{i}_2\, \vec{j}_2\, \vec{k}_2)$, then $R_0^2 = R_1^2 R_0^1$ transforms v_0 from $(\vec{i}_0\, \vec{j}_0\, \vec{k}_0)$ to $(\vec{i}_2\, \vec{j}_2\, \vec{k}_2)$.

Proof. In fact $v_1 = R_0^1 v_0$ and $v_2 = R_1^2 v_1 = R_1^2 R_0^1 v_0 = R_0^2 v_0$. □

Although R contains the information about the relative orientation between two coordinate frames, it does not have an intuitive interpretation. For example, seeing at the following rotation matrix:

$$R = \begin{pmatrix} 0.7036 & 0.7036 & -0.0998 \\ -0.7071 & 0.7071 & 0 \\ 0.0706 & 0.0706 & 0.995 \end{pmatrix} \tag{2}$$

it is impossible to have an idea about the relative orientation between both coordinate frames.

This fact motivates to express R as a composition of three elementary rotations. An elementary rotation is a rotation that occur about one principal axis of the coordinate frame. For example, considering $(\vec{i}_0\, \vec{j}_0\, \vec{k}_0)$ as the reference frame, and rotating it along $\vec{k}_0$, the resulting reference frame $(\vec{i}_1\, \vec{j}_1\, \vec{k}_1)$ is related to $(\vec{i}_0\, \vec{j}_0\, \vec{k}_0)$ by an elementary rotation along $\vec{k}_0$.

Any spatial orientation can be reduced to three consecutive elementary rotations. For example, Figure 6 shows two consecutive elementary rotations, the first one along $\vec{k}_0$, and the second one along $\vec{j}_1$. Each elementary rotation defines an angle and, therefore, the spatial orientation between two coordinate frames can be reduced to three angles. Moreover, it can be seen that there is more than one possibility. For example, we can reach the same final orientation with a sequence $\vec{k}_0$, $\vec{j}_1$ and $\vec{i}_2$, or with another one (like $\vec{k}_0$,

$\vec{i}_1$ and $\vec{j}_2$). In fact, one can find at least twelve different sequences to arrive to the same final position. Each combination defines different angles with a different order.

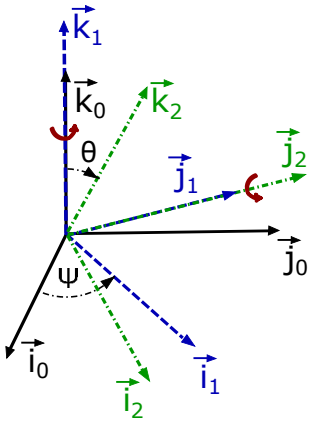

Figure 5: Two consecutive elementary rotations. The first along $\vec{k}_0$, and the second along $\vec{j}_1$.

In general for aerial vehicles, the sequence is chosen in the following order: $\vec{k}_0$, $\vec{j}_1$ and $\vec{i}_2$, defining the angles ψ (yaw), θ (pitch), ϕ (roll). Those angles are known as the Euler angles. In Figure 6 it can be seen the first two elementary rotations (the first one about $\vec{k}_0$ and the second one about $\vec{j}_1$), the last one would be about $\vec{i}_2$. Taking the definition of R, it can be seen that:

$$R_0^1 = \begin{pmatrix} \vec{i}_1\cdot\vec{i}_0 & \vec{i}_1\cdot\vec{j}_0 & \vec{i}_1\cdot\vec{k}_0 \\ \vec{j}_1\cdot\vec{i}_0 & \vec{j}_1\cdot\vec{j}_0 & \vec{j}_1\cdot\vec{k}_0 \\ \vec{k}_1\cdot\vec{i}_0 & \vec{k}_1\cdot\vec{j}_0 & \vec{k}_1\cdot\vec{k}_0 \end{pmatrix} = \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3}$$

$$R_1^2 = \begin{pmatrix} \vec{i}_2\cdot\vec{i}_1 & \vec{i}_2\cdot\vec{j}_1 & \vec{i}_2\cdot\vec{k}_1 \\ \vec{j}_2\cdot\vec{i}_1 & \vec{j}_2\cdot\vec{j}_1 & \vec{j}_2\cdot\vec{k}_1 \\ \vec{k}_2\cdot\vec{i}_1 & \vec{k}_2\cdot\vec{j}_1 & \vec{k}_2\cdot\vec{k}_1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \tag{4}$$

$$R_2^3 = \begin{pmatrix} \vec{i}_3\cdot\vec{i}_2 & \vec{i}_3\cdot\vec{j}_2 & \vec{i}_3\cdot\vec{k}_2 \\ \vec{j}_3\cdot\vec{i}_2 & \vec{j}_3\cdot\vec{j}_2 & \vec{j}_3\cdot\vec{k}_2 \\ \vec{k}_3\cdot\vec{i}_2 & \vec{k}_3\cdot\vec{j}_2 & \vec{k}_3\cdot\vec{k}_2 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \tag{5}$$

so if we consider $(\vec{i}_0\,\vec{j}_0\,\vec{k}_0)$ to be the Inertial frame, $\mathcal{I}$, and considering $(\vec{i}_3\,\vec{j}_3\,\vec{k}_3)$ to be the Body frame, $\mathcal{B}$, we can find two intermediate coordinate frames, v_1, v_2, that permits to get the global rotation as a succession of three elementary rotations ($\mathcal{I} \to v_1 \to v_2 \to \mathcal{B}$). The total rotation matrix, $R_{\mathcal{I}}^{\mathcal{B}}$, can be constructed as:

$$R_{\mathcal{I}}^{\mathcal{B}} = R_{v_2}^{\mathcal{B}} R_{v_1}^{v_2} R_{\mathcal{I}}^{v_1} \tag{6}$$

with (3)-(6) the rotation matrix, $R_{\mathcal{I}}^{\mathcal{B}}$, can be expressed in terms of ψ, θ and ϕ:

$$R_{\mathcal{I}}^{\mathcal{B}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \begin{pmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$R_{\mathcal{I}}^{\mathcal{B}} = \begin{pmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \sin\phi\sin\theta\sin\psi + \cos\phi\cos\psi & \cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{pmatrix} \tag{7}$$

Equation (7) gives the rotation matrix assuming that we know the angles ψ, θ, ϕ. Moreover, given ψ, θ, ϕ we can only obtain one (and only one) rotation matrix. But this not happen with the inverse problem. The inverse problem is to extract ψ, θ, ϕ from R and, as we are going to see, it does not have a unique solution.

In fact, we have $n_{1,x} = \cos\theta\cos\psi$, $n_{1,y} = \cos\theta\sin\psi$, $n_{1,z} = -\sin\theta$, $n_{2,z} = \sin\phi\cos\theta$ and $n_{3,z} = \cos\phi\cos\theta$. Therefore ψ, θ, ϕ can be reconstructed as follows:

$$\psi = atan\left(\frac{n_{y,1}}{n_{x,1}}\right) \quad \theta = atan\left(\frac{-n_{z,1}}{\sqrt{n_{x,1}^2 + n_{y,1}^2}}\right) \quad \phi = atan\left(\frac{n_{z,2}}{n_{z,3}}\right) \tag{8}$$

And this problem, if $\cos\theta \neq 0$ has two solutions: ψ, θ, ϕ and $\psi+180$, $\theta-180$, $\phi+180$. Moreover, if $\cos\theta = 0 \leftrightarrow \theta = \pm\pi/2$ then:

$$R_{\mathcal{I}}^{\mathcal{B}} = \begin{pmatrix} 0 & \sin(\phi-\psi) & \cos(\phi-\psi) \\ 0 & \cos(\phi-\psi) & -\sin(\phi-\psi) \\ \pm 1 & 0 & 0 \end{pmatrix} \tag{9}$$

which means that only $\phi - \psi$ can be observed. So ψ, θ, ϕ can not be reconstructed at $\theta = \pm\pi/2$. This is known as the Euler singularity and it is very important since it will constitute a mathematical singularity in the kinematic model if we use Euler angles to represent it.

2.3 Quaternions

Quaternions where first proposed by Hamilton in 1844 [35] as an extension of the imaginary numbers. He realized that complex numbers where a powerful tool to manage rotations in 2D. Therefore, he supposed that a generalization of the complex numbers would also result in a good tool to manage rotations in 3D.

2.3.1 Complex numbers to manage rotations in 2D

One of the most popular formulas in mathematics is the Euler's formula:

$$e^{\theta i} = \cos(\theta) + i\sin(\theta) \tag{10}$$

being $\theta \in \mathbb{R}$ and i the imaginary unit. The above expression can be proved as follows:

Proof. The power series expansions of i are:

$$\begin{aligned} i^0 = 1, \quad i^1 = i, \quad i^2 = -1, \quad i^3 = -i, \\ i^4 = 1, \quad i^5 = i, \quad i^6 = -1, \quad i^7 = -i, \end{aligned}$$

therefore, one can state:

$$\begin{aligned} e^{\theta i} &= 1 + i\theta + \frac{(i\theta)^2}{2!} + \frac{(i\theta)^3}{3!} + \frac{(i\theta)^4}{4!} + \frac{(i\theta)^5}{5!} + \frac{(i\theta)^6}{6!} + \frac{(i\theta)^7}{7!} + ... \\ &= 1 + i\theta - \frac{\theta^2}{2!} - \frac{i\theta^3}{3!} + \frac{\theta^4}{4!} + \frac{i\theta^5}{5!} - \frac{\theta^6}{6!} - \frac{i\theta^7}{7!} + ... \\ &= \left(1 - \frac{\theta^2}{2!} + \frac{\theta^4}{4!} - \frac{\theta^6}{6!} + ...\right) + i\left(\theta - \frac{\theta^3}{3!} + \frac{\theta^5}{5!} - \frac{\theta^7}{7!} + ...\right) \\ &= \cos(\theta) + i\sin(\theta) \end{aligned}$$

□

Equation (10) it is important for three reasons:

1. because it formally express the relationship between the exponential of a pure imaginary number and the trigonometric functions.

2. because assuming (10) any complex number can be expressed in a polar form.

   Proof.

   $$z = a + bi = |z|\cos(\theta) + i|z|\sin(\theta) = |z|e^{\theta i} \tag{11}$$

   being θ the angle that $a + bi$ form in the complex plane. □

3. because with (10) the exponential of a general complex number can be derived in an easy way.

   Proof.

   $$e^z = e^{a+bi} = e^a e^{bi} = e^a\left(\cos(b) + i\sin(b)\right) \tag{12}$$

   □

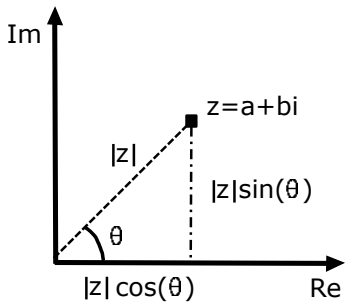

Figure 6: Imaginary number in the complex plane.

The complex exponential defined by (12) also preserve some interesting properties which are inherited from the real exponential. For example:

$$\frac{d}{dt}\left(e^{zt}\right) = ze^{zt} \tag{13}$$

Proof. (13) can be directly proved using the power expansion of the exponential. But here we will prove it in a different way. First lets prove that

$$\frac{d}{dt}\left(e^{it}\right) = ie^{it} \tag{14}$$

in fact, (14) can be developed as:

$$\begin{aligned}\frac{d}{dt}\left(e^{it}\right) &= \frac{d}{dt}\left(\cos(t) + i\sin(t)\right)\\ &= -\sin(t) + i\cos(t)\\ &= i(\cos(t) + i\sin(t))\\ &= ie^{it}\end{aligned}$$

therefore

$$\begin{aligned}\frac{d}{dt}\left(e^{zt}\right) = \frac{d}{dt}\left(e^{(a+bi)t}\right) = \frac{d}{dt}\left(e^{at}e^{bti}\right) =\\ &= ae^{at}e^{bti} + bie^{at}e^{bti}\\ &= (a+bi)e^{(a+bi)t}\\ &= ze^{zt}\end{aligned}$$

□

A vector in $\mathbb{R}^2$, $v = (v_1, v_2)$ can be considered as a complex number $v = v_1 + iv_2 = |v|e^{\theta i}$. If v is rotated an angle $\triangle\theta$, then the new coordinates, $\bar{v}$, are given by:

$$\bar{v} = |v|e^{(\theta+\triangle\theta)i} = |v|e^{\theta i}e^{\triangle\theta i} = ve^{\triangle\theta i} = vq \tag{15}$$

where

$$q = e^{\triangle\theta i} = \cos(\theta) + i\sin(\theta) \tag{16}$$

is the complex exponential of $\triangle\theta i$ and it can be defined as the rotation operator. Equation (15) provides an easy way to perform a rotation of a vector in $\mathbb{R}^2$, because the rotation is reduced to a single multiplication of two complex numbers. Now lets show how this approach can be generalized to manage rotations in 3D.

2.3.2 Definition of quaternion: sum and product of quaternions

This idea can be extended to 3-dimensional space by adding two imaginary numbers to the system. The general expression of a quaternion is given by:

$$q = q_0 + q_1 i + q_2 j + q_3 k \tag{17}$$

whose algebra is given by the Hamilton's famous expression

$$i^2 = j^2 = k^2 = ijk = -1 \tag{18}$$

which imply that:

$$\begin{array}{lll} ij = k, & jk = i, & ki = j, \\ ji = -k, & kj = -i, & ik = -j \end{array} \tag{19}$$

Note that with (18)-(19), a quaternion can be seen as an hipercomplex number where the "real" part is the complex number $q_0 + q_1 i$ and where the "imaginary" part is another complex number $q_2 + q_3 i$:

$$\begin{aligned} q &= q_0 + q_1 i + (q_2 + q_3 i)j \\ &= q_0 + q_1 i + q_2 j + q_3 ij \\ &= q_0 + q_1 i + q_2 j + q_3 k \end{aligned}$$

The product rule of two quaternions is given by (18):

$$\begin{aligned} qp &= (q_0 + q_1 i + q_2 j + q_3 k)(p_0 + p_1 i + p_2 j + p_3 k) \\ &= q_0 p_0 - (q_1 p_1 + q_2 p_2 + q_3 p_3) + p_0(q_1 i + q_2 j + q_3 k) + q_0(p_1 i + p_2 j + p_3 k) + \\ &+ (q_2 p_3 - q_3 p_2)i + (q_3 p_1 - q_1 p_3)j + (q_1 p_2 - q_2 p_1)k \end{aligned}$$

We can define a quaternion as a vector of $\mathbb{R}^4$ as follows:

$$q = q_0 + q_1 i + q_2 j + q_3 k = (q_0,\ q_1,\ q_2,\ q_3)^T = (q_0,\ \bar{q})^T \tag{20}$$

with $\bar{q} = (q_1,\ q_2,\ q_3)^T$. Then, the sum and multiplication of two quaternions can be expressed as:

$$q + p = \begin{pmatrix} q_0 + p_0 \\ \bar{q} + \bar{p} \end{pmatrix} \tag{21}$$

$$qp = \begin{pmatrix} q_0 p_0 - \bar{q}^T \bar{p} \\ q_0 \bar{p} + p_0 \bar{q} + S(\bar{q})\bar{p} \end{pmatrix} \tag{22}$$

where

$$S(\bar{q}) = \begin{pmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{pmatrix} \tag{23}$$

2.3.3 Quaternion conjugate, norm and inverse

The conjugate of the quaternion (20) is defined as:

$$q^* = q_0 - q_1 i - q_2 j - q_3 k = (q_0,\ -\bar{q}) \tag{24}$$

and it has the following properties:

1. $(q^*)^* = q$

   Proof. $(q^*)^* = (q_0,\ -\bar{q})^* = (q_0,\ \bar{q}) = q$ □

2. $q + q^* = 2q_0$

   Proof. $q + q^* = (q_0,\ \bar{q}) + (q_0,\ -\bar{q}) = (2q_0,\ 0) = 2q_0$ □

3. $q^* q = q q^*$

   Proof.

$$\begin{aligned} q^* q &= (q_0,\ \bar{q})(q_0,\ -\bar{q}) \\ &= \begin{pmatrix} q_0^2 + \bar{q}^T \bar{q} \\ q_0 \bar{q} - q_0 \bar{q} + S(\bar{q})\bar{q} \end{pmatrix} \\ &= \begin{pmatrix} q_0^2 + q_1^2 + q_2^2 + q_3^2 \\ 0 \end{pmatrix} \\ &= q q^* \end{aligned}$$

   □

4. $(pq)^* = q^* p^*$

   Proof.

$$(pq)^* = \begin{pmatrix} q_0 p_0 - \bar{p}^T \bar{q} \\ -p_0 \bar{q} - q_0 \bar{p} - S(\bar{p})\bar{q} \end{pmatrix} \tag{25}$$

$$\begin{aligned} q^* p^* &= \begin{pmatrix} q_0 p_0 - \bar{p}^T \bar{q} \\ -p_0 \bar{q} - q_0 \bar{p} - S(-\bar{q})\bar{p} \end{pmatrix} \\ &= \begin{pmatrix} q_0 p_0 - \bar{p}^T \bar{q} \\ -p_0 \bar{q} - q_0 \bar{p} - S(\bar{p})\bar{q} \end{pmatrix} \qquad (26) \\ &= (pq)^* \end{aligned}$$

   □

The norm of a quaternion is defined as $|q| = \sqrt{qq^*} = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2}$ and it has the following property:

1. $|qp|^2 = |q|^2|p|^2$

   Proof.

   $$|qp|^2 = (qp)(qp)^* = qpp^*q^* = q|p|^2q^* = qq^*|p|^2 = |q|^2|p|^2 \tag{27}$$

   □

The quaternion inverse is defined as:

$$q^{-1} = \frac{q^*}{|q|^2} \tag{28}$$

and it has the following property:

1. $qq^{-1} = q^{-1}q = 1$

   Proof.

   $$qq^{-1} = \frac{qq^*}{|q|^2} = \frac{|q|^2}{|q|^2} = 1$$
   $$q^{-1}q = \frac{q^*q}{|q|^2} = \frac{|q|^2}{|q|^2} = 1$$

   □

2.3.4 Quaternion exponential

The exponential function of a real number is defined as:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{29}$$

with $x \in \mathbb{R}$.

Equation (29) also defines the exponential for complex numbers or quaternions. As it has been mentioned in section 2.3.1, the Euler's identity is proved using (29), and with the Euler identity one can state the general exponential function of any complex number (equation (12)). Here an additional step will be made and we will find the general form of the quaternion exponential. Then we will see how the complex exponential and the real exponential are particular cases of the quaternion exponential.

In order to get the expression of the quaternion exponential, one could directly substitute x by a quaternion $q = q_0 + q_1 i + q_2 j + q_3 k$ in (29) and should operate with the rules defined by (18). But if this is difficult even if

we consider x to be just complex number, trying to directly substitute x by q and operate this is just a non-viable way.

Instead, let us take a different approach. It is known that if x is substituted by qt in (29), then it is verified that

$$\frac{d}{dt}\left(e^{qt}\right) = qe^{qt} \tag{30}$$

so $p(t) = e^{qt}$ is the solution of the following quaternion differential equation

$$\begin{cases} \dot{p} = qp \\ p(0) = (1,\ 0,\ 0,\ 0) \end{cases} \tag{31}$$

and, therefore, the quaternion exponential must take the following form:

$$p(t) = e^{qt} = e^{q_0 t} \begin{pmatrix} \cos(|\bar{q}|t) \\ \dfrac{q_1}{|\bar{q}|}\sin(|\bar{q}|t) \\ \dfrac{q_2}{|\bar{q}|}\sin(|\bar{q}|t) \\ \dfrac{q_3}{|\bar{q}|}\sin(|\bar{q}|t) \end{pmatrix} \tag{32}$$

Proof. If $p(t) = (p_1(t),\ p_2(t),\ p_3(t),\ p_4(t))^T$, then $\dot{p}(t) = (\dot{p}_1(t),\ \dot{p}_2(t),\ \dot{p}_3(t),\ \dot{p}_4(t))^T$. Applying the Laplace transform to $\dot{p}(t)$ we have that

$$\begin{aligned} \mathcal{L}\left\{\dot{p}(t)\right\}(s) &= (sp_1(s) - 1,\ sp_2(s),\ sp_3(s),\ sp_4(s))^T \\ &= sP(s) - \mathbf{1} \end{aligned}$$

with $\mathbf{1} = (1,\ 0,\ 0,\ 0)^T$. The quaternion product qp given by (22) can also be written as:

$$qp = \begin{pmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 - q_1p_3 + q_2p_0 + q_3p_1 \\ q_0p_3 + q_1p_2 - q_2p_1 + q_3p_0 \end{pmatrix}$$

it can be seen that each element of qp is linear on p_i, therefore

$$\mathcal{L}\left\{qp\right\}(s) = qP(s)$$

so applying the Laplace transform to (31) leads to

$$\begin{aligned} \mathcal{L}\left\{\dot{p}(t)\right\}(s) &= \mathcal{L}\left\{qp\right\}(s) \\ sP(s) - \mathbf{1} &= qP(s) \\ (s\mathbf{1} - q)P(s) &= \mathbf{1} \\ P(s) &= (s\mathbf{1} - q)^{-1} \end{aligned}$$

the quaternion inverse is given by (28). It is easy to see that

$$\begin{aligned} P(s) = (s\mathbf{1} - q)^{-1} &= (s - q_0 - q_1 i - q_2 j - q_3 k)^{-1} \\ &= \frac{s - q_0 + q_1 i + q_2 j + q_3 k}{(s-q_0)^2 + q_1^2 + q_2^2 + q_3^2} \\ &= \frac{s - q_0 + q_1 i + q_2 j + q_3 k}{(s-q_0)^2 + |\bar{q}|^2} \\ &= \begin{pmatrix} \dfrac{s - q_0}{(s-q_0)^2 + |\bar{q}|^2} \\ \dfrac{q_1}{|\bar{q}|} \cdot \dfrac{|\bar{q}|}{(s-q_0)^2 + |\bar{q}|^2} \\ \dfrac{q_2}{|\bar{q}|} \cdot \dfrac{|\bar{q}|}{(s-q_0)^2 + |\bar{q}|^2} \\ \dfrac{q_3}{|\bar{q}|} \cdot \dfrac{|\bar{q}|}{(s-q_0)^2 + |\bar{q}|^2} \end{pmatrix} \end{aligned}$$

applying the inverse Laplace transformation results in the equation (32):

$$e^{qt} = \mathcal{L}^{-1}\left\{P(s)\right\} = e^{q_0 t} \begin{pmatrix} \cos(|\bar{q}|t) \\ \dfrac{q_1}{|\bar{q}|}\sin(|\bar{q}|t) \\ \dfrac{q_2}{|\bar{q}|}\sin(|\bar{q}|t) \\ \dfrac{q_3}{|\bar{q}|}\sin(|\bar{q}|t) \end{pmatrix}$$

□

If we do $u = {}^{1}/_{|\bar{q}|}\,(q_1 i + q_2 j + q_3 k)$, the quaternion exponential can be expressed as:

$$e^{q} = e^{q_0}\left(\cos(|\bar{q}|) + u\sin(|\bar{q}|)\right) \tag{33}$$

which is a beautiful generalization of the complex exponential (12). It can be seen that for the particular case when $q_2 = q_3 = 0$, then (33) leads to (12). Moreover, if $q = u$ is considered as a pure unit quaternion ($q_0 = 0,\ |q| = 1$), then it is obtained the generalized Euler identity:

$$e^{\theta u} = (\cos(\theta) + u\sin(\theta)) \tag{34}$$

with $\theta \in \mathbb{R}$.

2.3.5 Quaternion as a rotator operator

If we have an arbitrary vector in the space, $\mathbf{x}$, and we rotate it an angle θ about an axis $\mathbf{u}$, the new coordinates of the rotated vector are:

$$\mathbf{x}' = \mathbf{x}_{\parallel} + \mathbf{x}_{\perp}\cos\theta + (\mathbf{u} \times \mathbf{x})\sin\theta \tag{35}$$

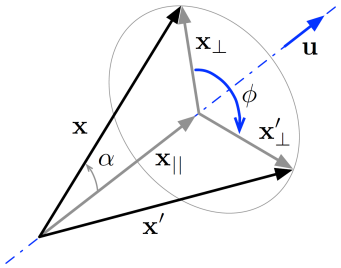

Figure 7: Rotation of a vector about an arbitrary axis. (Extrated from [36]).

where $\mathbf{x}_\parallel$ and $\mathbf{x}_\perp$ are the components of $\mathbf{x}$ which are parallel and perpendicular to $\mathbf{u}$, respectively.

Proof. The vector $\mathbf{x}$ can be decomposed into a part $\mathbf{x}_\parallel$ parallel to $\mathbf{u}$ and a part $\mathbf{x}_\perp$ perpendicular to $\mathbf{u}$.

$$\mathbf{x} = \mathbf{x}_\parallel + \mathbf{x}_\perp$$

where

$$\mathbf{x}_\parallel = \mathbf{u}(\parallel x \parallel \cos\theta) = \mathbf{u}\mathbf{u}^T\mathbf{x}$$
$$\mathbf{x}_\perp = \mathbf{u} \times \mathbf{x}$$

obviously, the parallel part does not rotate. So the rotation only changes $\mathbf{x}_\perp$. If we create a new orthogonal basis (e_1, e_2) being

$$e_1 = \mathbf{x}_\perp$$
$$e_2 = \mathbf{u} \times \mathbf{x}_\perp = \mathbf{u} \times \mathbf{x}$$

then the rotated vector can be expressed as

$$\mathbf{x}'_\perp = e_1 \cos\theta + e_2 \sin\theta = (\mathbf{x}_\perp)\cos\theta + (\mathbf{u} \times \mathbf{x})\sin\theta$$

and therefore

$$\mathbf{x}' = \mathbf{x}_\parallel + \mathbf{x}'_\perp = \mathbf{x}_\parallel + \mathbf{x}_\perp \cos\theta + (\mathbf{u} \times \mathbf{x})\sin\theta \tag{36}$$

□

The same operation can be performed using quaternions and the generalized Euler identity (34). Let us express the unitary axis u as $u_1 i + u_2 j + u_3 k$, and the rotation angle as θ. Then, we can build the rotation quaternion:

$$q = e^{u(\theta/2)} = \cos(\theta/2) + u\sin(\theta/2) \tag{37}$$

and if we express the vector $\mathbf{x}$ as a pure quaternion, $\mathbf{x} = x_1 i + x_2 j + x_3 k$, then

$$\mathbf{x}' = q\mathbf{x}q^* \tag{38}$$

Proof. One just need to operate (37)-(38) and it will be found that

$$\mathbf{x}' = q\mathbf{x}q^* = \mathbf{x}_{\parallel} + \mathbf{x}_{\perp}\cos\theta + (\mathbf{u}\times\mathbf{x})\sin\theta$$

□

So given a vector v we can rotate it about an axis just by applying (38), which is simply a quaternion multiplication (involving scalar multiplications). So the quaternion encodes the same information as the rotation matrix.

Equation (38) performs the rotation of the vector. That means that if we want to rotate a single vector, v, an angle θ about an axis, u, we can build the rotation quaternion as $q = e^{u(\theta/2)}$ and then we can apply (38). But sometimes is better to codify with the quaternion the rotation of the whole reference frame and not a single vector. The difference can be seen in Figure 8, where the quaternion q represents a rotation of $\theta = 90°$ about $\vec{k}_0$. If the quaternion represents a rotation of the coordinate frame, then the vector "sees" the opposite rotation. And if we want to represent the vector in the other reference frame, the equation (38) needs to be changed by

$$v' = q^* v q \tag{39}$$

If we have a vector v_0 expressed in the coordinate frame $(\vec{i}_0, \vec{j}_0, \vec{k}_0)$, and if such coordinate frame is rotated by q_0 leading to a second coordinate frame $(\vec{i}_1, \vec{j}_1, \vec{k}_1)$, then the coordinates of v_0 expressed in $(\vec{i}_1, \vec{j}_1, \vec{k}_1)$ are:

$$v_1 = q_0^* v_0 q_0$$

therefore, if $(\vec{i}_1, \vec{j}_1, \vec{k}_1)$ is rotated again by q_1 leading to $(\vec{i}_2, \vec{j}_2, \vec{k}_2)$, then the coordinates of v_0 expressed in $(\vec{i}_2, \vec{j}_2, \vec{k}_2)$ are:

$$v_2 = q_1^* v_1 q_1 = q_1^* q_0^* v_0 q_0 q_1 = (q_0 q_1)^* v_0 (q_0 q_1) \tag{40}$$

the quaternion $q = (q_0 q_1)$ represents the global rotation, and it encodes information of the axis and the angle in order to go from $(\vec{i}_0, \vec{j}_0, \vec{k}_0)$ to $(\vec{i}_2, \vec{j}_2, \vec{k}_2)$. Equation (40) represents the quaternion version of the composition of rotations.

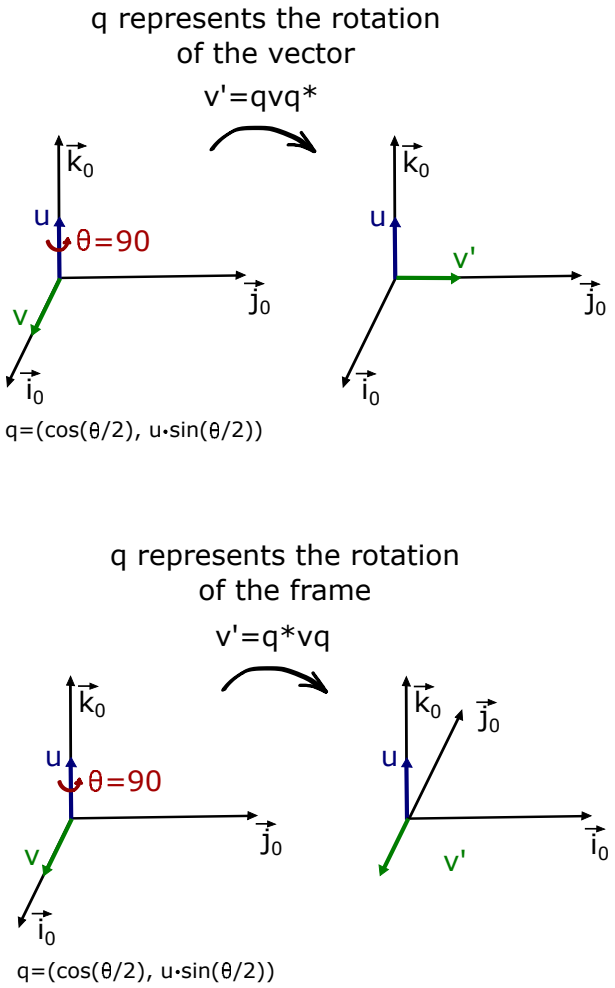

Figure 8: Quaternion as rotation operator. Rotating the vector or the frame.

3 Derivation of the quadrotor attitude kinematic model

In this section we derive the attitude kinematics. The kinematics relates how the body angular velocity change the attitude measure. As it has been mentioned in Section 2, an aerial vehicle can be interpreted as a coordinate frame, $\mathcal{B}$, which is rotated with respect to the inertial frame, $\mathcal{I}$. This relative orientation can be described by the Euler's angles or by a single quaternion. The question now is: if we know the instant angular velocity, Ω, of $\mathcal{B}$, can we know the rate of change (d/dt) of the Euler's angles or the quaternion?. This relationship corresponds to the kinematic model of the aerial vehicle and, in this section, we will derive this expression for each case.

3.1 The Euler-based kinematic model and its singularities

In Section 2 it was shown that between $\mathcal{B}$ and $\mathcal{I}$ there were two intermediate frames: v_1 and v_2. Those intermediate frames exist because the Euler angles codify three consecutive rotations. The first one, ψ, transforms $\mathcal{I}$ to v_1 ($\mathcal{R} \rightarrow v_1$). The second one, θ, transforms $v_1 \rightarrow v_2$ and the last one, ϕ, transforms $v_2 \rightarrow \mathcal{B}$.

The rotation matrix of each frame with respect to its previous is given by (3)-(5), and the total rotation matrix is given by (7). The relationship between the Euler angles and the angular velocity, Ω, is not evident because these quantities are defined in different frames. Ω is defined in $\mathcal{B}$, ϕ is defined in v_2, θ is defined in v_1 and ψ is defined in $\mathcal{I}$.

The aforementioned relationship is given by:

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \sin(\phi)\sec(\theta) & \cos(\phi)\sec(\theta) \end{pmatrix} \cdot \Omega \tag{41}$$

Proof. To prove the above expression we just need to express all the magnitudes in $\mathcal{B}$.

$$\begin{aligned} \Omega &= \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + R^{\mathcal{B}}_{v_2}(\phi) \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + R^{\mathcal{B}}_{v_2}(\phi) R^{v_2}_{v_1}(\theta) \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \\ &= \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \\ &+ \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{pmatrix} \begin{pmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix} \end{aligned}$$

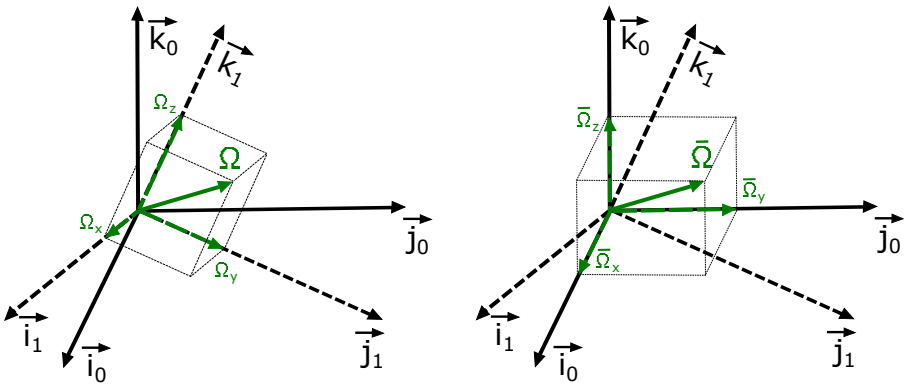

Figure 9: The body frame, $\mathcal{B} = (i_1, j_1, k_1)$, rotates with respect to $\mathcal{I} = (i_0, j_0, k_0)$ with its own angular velocity Ω.

operating we get:

$$\Omega = \begin{pmatrix} 1 & 0 & -\sin(\theta) \\ 0 & \cos(\phi) & \sin(\phi)\cos(\theta) \\ 0 & -\sin(\phi) & \cos(\phi)\cos(\theta) \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\theta} \end{pmatrix}$$

and, inverting the matrix one gets (41). □

Note that (41) is not defined in $\theta = \pi/2$. The singularity remains in the kinematic model. So one needs to be careful with the singularity if the control is based on the Euler-based kinematic model.

3.2 The quaternion-based kinematic model

If we have two coordinate frames, $\mathcal{B} = (i_1, j_1, k_1)$ and $\mathcal{I} = (i_0, j_0, k_0)$, as depicted in Figure 9. The relative orientation between them can be defined by a rotation θ about an axis u. This relative orientation can be codified by a quaternion, q, by means of the quaternion exponential as it has been mentioned in section 2.3.4:

$$q = e^{u\frac{\theta}{2}} = \cos\left(\frac{\theta}{2}\right) + u\sin\left(\frac{\theta}{2}\right)$$

If $\mathcal{B}$ is rotating with an angular velocity $\bar{\Omega} = \Omega_x i + \Omega_y j + \Omega_z k$ (expressed in $\mathcal{I}$), then, the following expression holds:

$$\dot{q} = \frac{1}{2}\bar{\Omega}q \tag{42}$$

Proof. At time t the relative orientation is described by $q(t)$. After some time $\triangle t$ we have $q(t + \triangle t)$. This extra rotation is about the instantaneous

axis $\hat{\Omega} = \bar{\Omega}/\|\bar{\Omega}\|$ through an angle $\triangle\theta = \|\bar{\Omega}\|\triangle t$. This change of orientation can be also described by a quaternion.

$$\triangle q = \cos\left(\frac{\triangle\theta}{2}\right) + \hat{\Omega}\sin\left(\frac{\triangle\theta}{2}\right) \tag{43}$$

so, the rotation at $t + \triangle t$ is described by a quaternion sequence: $q(t), \triangle q$. Therefore:

$$q(t + \triangle t) = \triangle q q(t) \tag{44}$$

So at this point we are able to apply the definition of the time derivative

$$\dot{q}(t) = \lim_{\triangle t \to 0} \frac{q(t + \triangle t) - q(t)}{\triangle t} \tag{45}$$

being

$$\begin{aligned} q(t + \triangle t) - q(t) &= \left(\cos\frac{\|\bar{\Omega}\|\triangle t}{2} + \hat{\Omega}\sin\frac{\|\bar{\Omega}\|\triangle t}{2}\right) q(t) - q(t) \\ &= -2\sin^2\frac{\|\bar{\Omega}\|\triangle t}{4} q(t) + \hat{\Omega}\sin\frac{\|\bar{\Omega}\|\triangle t}{2} \end{aligned} \tag{46}$$

the first term is of higher order than the second one, so it goes to zero. Then we have:

$$\begin{aligned} \dot{q}(t) &= \lim_{\triangle t \to 0} \frac{q(t + \triangle t) - q(t)}{\triangle t} \\ &= \hat{\Omega} \lim_{\triangle t \to 0} \frac{\sin(\|\bar{\Omega}\|\triangle t/2)}{\triangle t} q(t) \\ &= \hat{\Omega}\frac{\|\bar{\Omega}\|}{2}\cos\left(\frac{\|\bar{\Omega}\|t}{2}\right)\Big|_{t=0} q(t) \\ &= \frac{1}{2}\bar{\Omega}(t)q(t) \end{aligned} \tag{47}$$

□

Normally, the angular velocity is expressed in $\mathcal{B}$ and not in $\mathcal{I}$. In that case we can do $\bar{\Omega} = q\Omega q^*$, and substituting in (47) we have

$$\dot{q} = \frac{1}{2} q\Omega \tag{48}$$

4 Quadrotor attitude control

The quadrotor model can be developed using either Euler angles or quaternions. Both approaches have been widely studied in the literature over the last decades. For example, a detailed description of the Euler-based model and how it is derived from basic physical laws can be found in [12]. However, it is known that the Euler-based model is strongly non-linear and it has a singularity in $\pi/2$. For those reasons, the quaternion approach has become increasingly important as it eliminates the singularity and notably reduces the non-linearities. Many quaternion-based controllers have been proposed for quadrotors (see [11, 15, 17–22]). That is because quaternions are an excelent tool for managing 3-D space rotations and they have a direct application in quadrotor control. Quaternions were first proposed by Hamilton in 1843 and, since then, its algebra and properties has been extensively developed [13, 14, 27].

The quadrotor model can be interpreted as a cascade connection between two subsystems: the dynamic one and the kinematic one. The dynamics depends on physical parameters and describes how the control inputs (and other disturbances) change the body angular velocity. The kinematics do not depend on any physical parameter and simply relates the attitude variation with the angular velocity [16]. This decomposition suggests that a natural way to perform control is to develop a cascade control so that one control loop is designed for each subsystem. The cascade control approach has been explicitly [11] or implicitly [18, 20, 21] mentioned in the literature and it is possible as far as the angular velocity and the attitude quaternion are available measurements.

Three important problems in the quadrotor control are: parameter identification, non-linearities and external disturbances. Several control schemes have been proposed to solve these problems, but only a few treat them altogether. For example, a quaternion-based non-linear control law is proposed in [18] demonstrating exponential convergence, but it is assumed that the model is perfectly known and it does not take into account any model mismatch. In [15], the model uncertainties and external disturbances are considered as an input equivalent disturbance and are compensated, but small error angles are assumed. In [22], an extended observer is designed to estimate and compensate for a class of time-varying external disturbances and it is proved to be convergent, but perfect model matching is also assumed. In [21], a control solution with disturbance rejection where the disturbances and model mismatches are estimated by a Nonlinear Disturbance Observer (NDOB) is proposed. Other interesting studies have been also developed. For instance, in [16], many control schemes based on Sliding Mode Control (SMC) and Extended State Observer (ESO) (among others) are proposed for different aerial vehicles. In this document, a simple controller that is capable of solving the three aforementioned problems is proposed. This control

scheme is based on the properties of quaternions to manage space rotations and in some theoretical results of the Active Disturbance Rejection Control (ADRC) to deal with external disturbances, non-linearities and model uncertainties.

The ADRC was firstly proposed by Han [23, 24] with the aim of solving the main disadvantages of traditional PID. It is mainly composed of three elements: a Tracking-Differentiator (TD), an Extended State Observer (ESO) and a state-feedback control law. As it was proposed, the ESO and the feedback control law could be non-linear functions. However, important theoretical analyses have been developed for the particular case when the ESO is a reduced-order linear ESO (RLESO), and when the state feedback is also linear [25]. For that particular case, the input-output stability can be analyzed by the Xue and Huang's theorem [26]. In this document, a RLESO-based linear ADRC (LADRC) is proposed to control the quadrotor dynamic system.

This section is structured as follows. In Section 4.1, the whole quadrotor model is introduced and discussed. The proposed control scheme is developed in Section 4.2. Some simulations are presented in Section 4.3 in order to illustrate the closed-loop control performance.

4.1 Quadrotor Model

The quadrotor model can be seen as the interconnection of two subsystems: the dynamic one and the kinematic one. In this document, for control purposes, both subsystems are treated separately in order to develop a specific controller for each one of them.

4.1.1 Kinematic Model

A quaternion $q \in \mathbb{R}^4$ is composed of a scalar term $q_0 \in \mathbb{R}$, a vector part $\bar{q} \in \mathbb{R}^3$, and it is defined by $q = (q_0, q_1, q_2, q_3)^T = (q_0, \bar{q})^T$. As it has been mentioned in Section 2.3.5, it is well known that the orientation of two reference frames can be expressed as a single rotation θ along a unitary axis $u \in \mathbb{R}^3$. The quaternion encodes information about such rotation. Specifically, it can be expressed as

$$q = \begin{pmatrix} q_0 \\ \bar{q} \end{pmatrix} = \begin{pmatrix} \cos(\theta/2) \\ \sin(\theta/2) \cdot u \end{pmatrix} = e^{u(\theta/2)} \tag{49}$$

From (49), it follows that $\|q\| = 1$ and hence q is referred to as a unit quaternion. Furthermore, it can be seen that $-1 \leq q_0 \leq 1$, however it is certified that $(q_0, \bar{q})$ and $(-q_0, -\bar{q})$ correspond to the same physical point. So q_0 is chosen to be always positive ($0 \leq q_0 \leq 1$). In what follows, q_i^j denotes the quaternion containing the rotation (θ, u) that, applied to the reference frame i, makes it coincident with the frame j.

Let us denote the inertial and body-fixed reference frames by $\mathcal{I}$ and $\mathcal{B}$, respectively. In Section 3.2 it has been demonstrated that the quadrotor kinematics can be written as:

$$\dot{q}_{\mathcal{I}}^{\mathcal{B}} = \frac{1}{2} q_{\mathcal{I}}^{\mathcal{B}} \otimes w_b \tag{50}$$

where $w_b = (0,\, \Omega_b)^T \in \mathbb{R}^4$ is a quaternion associated to the body angular velocity, Ω_b, and $\otimes$ denotes the quaternion product rule which it has been also defined in Section 2.3.2.

The quaternion product in (50) can be expressed in the following matrix form:

$$q \otimes w = \begin{pmatrix} -\bar{q}^T \\ q_0 \cdot I_{3x3} + S(\bar{q}) \end{pmatrix} \cdot \Omega \tag{51}$$

where $S(\cdot)$ is the skew-symmetric operator

$$S(\bar{q}) = \begin{pmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{pmatrix} \tag{52}$$

Equation (50) represents the quadrotor kinematic model with respect to the fixed inertial reference frame, $\mathcal{I}$. However, in order to develop a more general kinematic model, in this document is assumed that the desired orientation of the vehicle is given by a feasible trajectory $q_{\mathcal{I}}^{\mathcal{R}}(t)$ of a virtual desired reference frame $\mathcal{R}$, which also must satisfy.

$$\dot{q}_{\mathcal{I}}^{\mathcal{R}} = \frac{1}{2} q_{\mathcal{I}}^{\mathcal{R}} \otimes w_d \tag{53}$$

where $w_d = (0,\, \Omega_d)^T$, and $\Omega_d(t)$ is the angular velocity of the desired reference frame.

The relative orientation error between $\mathcal{B}$ and $\mathcal{R}$ can be obtained as a composition of rotations [14, 27]:

$$q_{\mathcal{R}}^{\mathcal{B}} = (q_{\mathcal{I}}^{\mathcal{R}})^* \otimes q_{\mathcal{I}}^{\mathcal{B}} \tag{54}$$

where $(\cdot)^*$ denotes the quaternion conjugate. Equation (54) gives $q_{\mathcal{R}}^{\mathcal{B}}$ which is a measure of the quadrotor attitude, $\mathcal{B}$, with respect to the desired reference frame $\mathcal{R}$.

It can be proved, [19], that using (50), (53), (54) and the derivative product rule, the following differential equation holds:

$$\dot{q}_{\mathcal{R}}^{\mathcal{B}} = \frac{1}{2} q_{\mathcal{R}}^{\mathcal{B}} \otimes \tilde{w} \tag{55}$$

with $\tilde{w} = (0,\, \tilde{\Omega}) = w_b - (q_{\mathcal{R}}^{\mathcal{B}})^* \otimes w_d \otimes q_{\mathcal{R}}^{\mathcal{B}}$.

Equation (55) represents the complete kinematic model of the quadrotor. It should be mentioned that in the particular case when the desired reference is fixed ($w_d = 0$), the model (53) is null and the models (50) and (55) are equivalent.

In what follows, in order to simplify notation, $q_{\mathcal{R}}^{\mathcal{B}}$, $q_{\mathcal{I}}^{\mathcal{B}}$ and $q_{\mathcal{I}}^{\mathcal{R}}$ are respectively referenced as q_e, q_b and q_d, denoting the error, body and desired attitudes.

4.1.2 Dynamic Model

The quadrotor rotational dynamics is described by the Newton's second law, which is given in its simplest form by

$$\dot{\Omega}_b = I_{cm}^{-1} \cdot \tau(t) \tag{56}$$

where $\Omega_b \in \mathbb{R}^3$ is the angular velocity, $\tau \in \mathbb{R}^3$ is the total torque acting on the vehicle and I_{cm} is the inertia matrix, which is considered to be diagonal.

It is well known that, for a quadrotor, $\tau(t)$ can be expressed as the sum of the torques produced by: the propellers, the Coriolis term, and the gyroscopics and aerodynamics effects [15, 17, 18]. According to this decomposition $\tau(t)$ can be expressed as:

$$\tau(t) = M \cdot u - \Omega_b \times I_{cm}\Omega_b - G_a(t) - \tau_{aero}(t) + p(t) \tag{57}$$

where $M = diag\{m_{11}, m_{22}, m_{33}\}$ is a matrix relating the torque with the motors speed, $u \in \mathbb{R}^3$, and, G_a and τ_{aero} are the gyroscopic and aerodynamic torques, respectively. The term $p(t)$ represents other unknown external disturbances (like wind gusts) and unmodeled dynamics.

The terms G_a and τ_{aero} are usually expressed as:

$$G_a = \textstyle\sum_{i=1}^{4} I_r(\Omega_b \times e_z)(-1)^{i+1} w_i \tag{58}$$

$$\tau_{aero}(t) = 1/2 \begin{pmatrix} \rho C_x W_x^2 & \rho C_y W_y^2 & \rho C_z W_z^2 \end{pmatrix}^T \tag{59}$$

where $I_r \in \mathbb{R}^{3x3}$ is the rotors inertia matrix, $e_z = (0,\ 0,\ 1)^T$, $w_i \in \mathbb{R}^3$ is the rotors angular velocity and ρ, C_i, $W_i, \in \mathbb{R}$, $i = x, y, z$ respectively are the air density, aerodynamic coefficients and the quadrotor velocity with respect to the air.

Although (57)-(59) provide a complete description of the dynamics, it should be remarked that none of its terms is accurately known. For example, the matrix M and τ_{aero} depend on aerodynamic coefficients that are very difficult to model. The term G_a depends on some rotors parameters which also are unknown. The external disturbances, $p(t)$, are obviously unknown. Even the Coriolis term is uncertain because it depends on the inertia matrix which is uncertain too.

In the light of the arguments raised above, it is of dubious interest to develop control laws which rely on the accurate knowledge of these parameters. Instead, let us use (56)-(57) to rewrite the quadrotor dynamic model as

$$\dot{\Omega}_b(t) = d(t) + f(\Omega, t) + Bu(t) \tag{60}$$

where $d(t) = I_{cm}^{-1} p(t)$ and $B = I_{cm}^{-1} M = diag\{b_1, b_2, b_3\}$. $f(\Omega, t)$ is a lumped term that contains the rest of the unknown dynamics:

$$f(\Omega, t) = -I_{cm}^{-1} \left[\Omega_b \times I_{cm}\Omega_b + G_a(t) + \tau_{aero}(t)\right] \tag{61}$$

The dynamic control should drive the quadrotor (60) to follow a given reference trajectory, $\Omega_r(t)$. Therefore, for control design purposes it is helpful to define the error model as $x_1 = \Omega_b - \Omega_r$. Differentiating x_1 leads to:

$$\dot{x}_1 = \tilde{d}(t) + f(\Omega, t) + Bu(t) \tag{62}$$

where $\dot{\Omega}_r$ is considered as an unknown disturbance and thus $\tilde{d}(t) = d(t) - \dot{\Omega}_r$.

The model (62), which retains the information of the certainly known dynamics, will be used for control design purposes.

4.2 Proposed Control Scheme

The proposed control scheme (as depicted in Figure 10) consists of a cascade connection between two independent controllers, which are developed next.

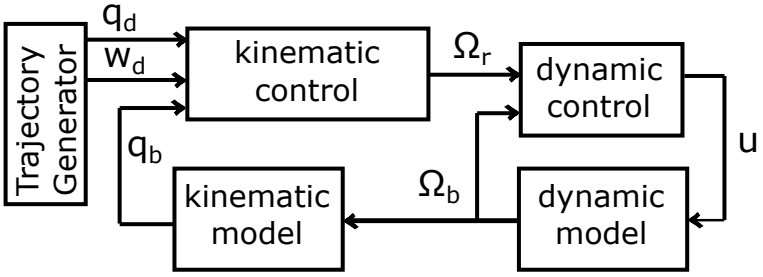

Figure 10: Proposed control scheme

4.2.1 Kinematic Loop

The outer loop consists of a non-linear quaternion-based control law that solves the general attitude kinematic control problem. It takes q_d, w_d, q_b as inputs, and generates a reference angular velocity, Ω_r, for the dynamic control system. The pure attitude kinematic control problem neglects the system dynamics, that is, it is assumed that Ω_b tracks Ω_r instantly ($\Omega_b = \Omega_r$). This problem can be stated as:

Problem 1: *Given a solid ($\mathcal{B}$), in an arbitrary initial orientation, which can be rotated with an angular velocity Ω_b satisfying (50), and a given a desired reference frame ($\mathcal{R}$) which freely rotates satisfying (53); find a control law $\Omega_b(q_b, q_d, \Omega_d)$ such that the error between them (55) is driven to $q_e = (1,\, 0,\, 0,\, 0)$.*

Theorem 1. If measures q_b, q_d, and Ω_d (or $\dot{q}_d$) are available, the following control law solves the Problem 1 and the attitude error, q_e, is exponentially convergent to $(1,\, 0,\, 0,\, 0)$:

$$\begin{aligned} w_{d'} &= (q_e)^* \otimes w_d \otimes q_e = (0,\, \Omega_{d'})^T \\ \Omega_b &= -k_k \bar{q}_e + \Omega_{d'} \end{aligned} \tag{63}$$

Proof. As $w_b = (0,\, \Omega_b)^T$ and $w_{d'} = (q_e)^* \otimes w_d \otimes q_e = (0,\, \Omega_{d'})^T$, then $w_b - w_{d'} = (0,\, \Omega_b - \Omega_{d'})^T = (0,\, \tilde{\Omega})^T$. Therefore $\tilde{\Omega} = \Omega_b - \Omega_{d'}$. In order to to demonstrate that the system (55) is convergent to $(1, 0, 0, 0)^T$ the following Lyapunov candidate function is considered:

$$V = (1 - q_{e,0})^2 + (\bar{q}_e)^T \bar{q}_e \geq 0 \tag{64}$$

Where $V = 0$ if, and only if, $q_{e,0} = 1$ and $\bar{q}_e = 0$. With the quaternion product matrix representation (51), and model (55) its time-derivative takes the form:

$$\begin{aligned} \dot{V} &= (1 - q_{e,0}) (\bar{q}_e)^T \tilde{\Omega} + (\bar{q}_e)^T (q_{e,0} I_3 + S(\bar{q}_e)) \tilde{\Omega} \\ &= (\bar{q}_e)^T \tilde{\Omega} \end{aligned}$$

Taking $\tilde{\Omega} = -k_k \bar{q}_e$:

$$\dot{V} = -k_k (\bar{q}_e)^T \bar{q}_e \leq 0 \tag{65}$$

So under $\tilde{\Omega} = -k_k \bar{q}_e \Leftrightarrow \Omega_b = -k_k \bar{q}_e + \Omega_{d'}$, it can be concluded that the system is asymptotically stable. Now, lets prove the exponential convergence. Since $0 \leq q_{e,0} \leq 1$ and $\|q\| = 1$, V and $\dot{V}$ can be expressed as:

$$V = \quad 2(1 - q_{e,0}) \leq 2 \tag{66}$$

$$\dot{V} = \quad -k_k (1 - q_{e,0}^2) \tag{67}$$

with equations (66)-(67) the Lyapunov function can be written in the form:

$$\dot{V} = -k_k V + k_k \frac{V^2}{4} = -k_k V + g(V)$$

$g(V)$ is upper bounded by $(k_k/2)V$ in the region $V \leq 2$, so:

$$\begin{aligned} \dot{V} = -k_k V + g(V) &\leq -k_k V + \frac{k_k}{2} V \\ &\leq -\frac{k_k}{2} V \end{aligned}$$

Therefore

$$V(t) \leq V(0) \cdot e^{-\left(\frac{k_k}{2} \cdot t\right)} \tag{68}$$

and the exponential convergence is demonstrated.

□

Remark 1. The control action produced by (63) will be used as the reference input for the dynamic control system. In order to make this connection achievable, the dynamic loop needs to be faster (high bandwidth) than the kinematic loop. So control parameters needs to be chosen for that purpose. If this condition is satisfied, the dynamic loop will force $\Omega_b \to \Omega_r$ sufficiently fast in order to assure $\Omega_b \approx \Omega_r$.

4.2.2 Dynamic loop

The inner loop consists of a complete LADRC structure that takes the output of the kinematic controller (Ω_r) and the angular velocity measurement (Ω_b) as inputs, and generates the motors speed, u. The controller consists of a a RLESO, a new proposed TD valid for first-order systems and a linear feedback (Figure 11). It is proved next that the LADRC drives (62) to zero rejecting external disturbances and model uncertainties.

Let us define $\bar{B} = diag\{\bar{b_1}, \bar{b_2}, \bar{b_3}\}$ as an estimation of B. Since $B, \bar{B}$ are diagonal, the system (62) can be decoupled by each axis. Therefore the design of the LADRC can be reduced to a single axis, the results being equal for the other two. For the axis $i = x, y, z$ and including the unknown disturbances as an extended state, the model (62) becomes:

$$\begin{cases} \dot{x}_{1,i} = x_{2,i} + \bar{b}_i u_i \\ \dot{x}_{2,i} = h_i(\Omega_{b,i}, u_i, t) \\ y_{m,i} = x_{1,i} = \Omega_{b,i} - \Omega_{r,i} \end{cases} \tag{69}$$

where $x_{1,i} \in \mathbb{R}$ and $x_{2,i} = \tilde{d}_i(t) + f_i(\Omega_{b,i}, t) + (b_i - \bar{b}_i)u_i \in \mathbb{R}$ being $h_i(\Omega_{b,i}, u_i, t)$ its time derivative.

Since $y_{m,i} = x_{1,i}$ is available, only the estimation of $x_{2,i}$ is needed. So the following RLESO [26] will be used:

$$\begin{cases} \dot{z}_{2,i} = -\beta z_{2,i} - \beta^2 x_{1,i} - \beta \bar{b}_i u_i \\ \hat{x}_{2,i} = z_{2,i} + \beta x_{1,i} \end{cases} \tag{70}$$

whose characteristic polynomial is $(s + \beta)$ being $\beta > 0$ the bandwidth of the RLESO. The initial condition must be chosen to force $\hat{x}_{2,i}(0) = 0$, so: $z_{2,i}(0) = -\beta x_{1,i}(0)$.

The control law is designed to track the following exponential-convergent system:

$$\dot{x}^*_{1,i} = -k_d x^*_{1,i} \tag{71}$$

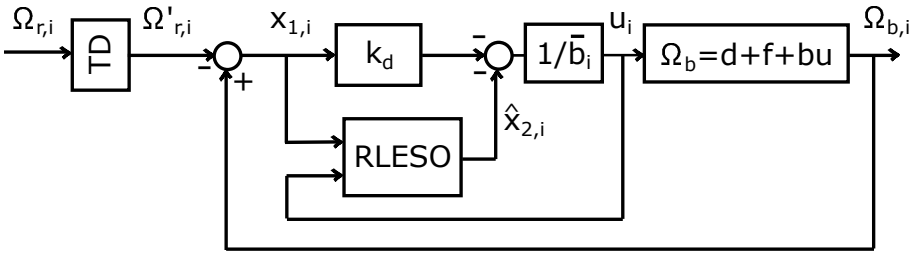

Figure 11: Block diagram of the LADRC structure.

being $k_d > 0$ the feedback control gain.

For that purpose u is chosen to carry out the feedback linearization:

$$u_i = \frac{-k_d x_{1,i} - \hat{x}_{2,i}}{\bar{b}_i} \tag{72}$$

The main purpose of the TD is to act as a filter for the inputs in order to reduce the input noise and to avoid overshoots in the control actions [23]. For this work, the proposed TD is based on a system of the form $\dot{x} = u$, $|u| < r$ where u needs to be chosen such $x(t) \to x_{ref}(t)$ being x_{ref} the TD input. For that purpose the following TD system is proposed:

$$\begin{pmatrix} \dot{v}_1 \\ \dot{v}_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & -\beta \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + \begin{pmatrix} 1 & 0 \\ 0 & \beta \end{pmatrix} \begin{pmatrix} u_{TD} \\ \dot{\Omega}_{r,i} \end{pmatrix} \tag{73}$$

where $v_1 = \Omega'_{r,i}$, $v_2 = \dot{\Omega}'_{r,i}$ and the control action is chosen as $u_{TD} = \xi(-k_d(v_1 - \Omega_{r,i}) + v_2,\, r)$, being $\xi(x_1, x_2)$ a saturation function which returns x_1 if $|x_1| < |x_2|$ or $|x_2| sign(x_1)$ if $|x_1| \geq |x_2|$.

Therefore, the TD receives as inputs the non-processed $\Omega_{r,i}$, $\dot{\Omega}_{r,i}$ and provides a filtered version $\Omega'_{r,i}$, $\dot{\Omega}'_{r,i}$ satisfying that $\Omega'_{r,i} \to \Omega_{r,i}$ and $|u_{TD}| \leq r$. It is important to note that since $\Omega_{r,i}$ may be non-continuous, it can be no differentiable and $\dot{\Omega}_{r,i}$ may no exist so (73) needs to be implemented in its discrete version, where $\dot{\Omega}_{r,i}$ can be always computed as $\dot{\Omega}_{r,i}(k) \approx \frac{1}{h}(\Omega_{r,i}(k) - \Omega_{r,i}(k-1))$ being h the time period.

For the considered LADRC structure, the Xue and Huang's theorem assures the closed-loop stability if three assumption are fulfilled. Now, these assumptions are analyzed for the quadrotor dynamic system considered.

A1: The discontinous points of $\tilde{d}(t)$ represented as $\{t_i\}_1^\infty$, $t_i < t_{i+1}$, are all first class and there exist positive constants w_1, w_2, w_3 such that:

$$\begin{cases} \sup\limits_{\forall t} |\tilde{d}(t)| \leq w_1 \\ \sup\limits_{\forall t \neq \{t_i\}_1^\infty} |\dot{\tilde{d}}(t)| \leq w_2 \\ \inf\limits_{\forall i} \{t_{i+1} - t_i\} \geq w_3 \end{cases}$$

A2: Ω_b, B, B^{-1} are bounded.

A3: The model error mismatch satisfies, for $i = x, y, z.$:

$$^{b_i}/_{\bar{b}_i} \in [w_4,\, w_5] \subset (0,\, \infty)$$

If A1-A3 holds, then:

Theorem 2. (Xue and Huang [26]). For an initial condition $|x_{1,i}(0)| \leq \tilde{\rho}_0$, there exist positives w^*, $\tilde{\rho}$, η_i^*, $i = 1, 2, 3$ which are dependent on $\tilde{\rho}_0$, k_d, w_j, $j = 1, 2, 3, 4, 5$ and the boundedness of Ω_b, such that the closed-loop system composed of (69)-(72) satisfies $\forall \beta \in [w^*,\, \infty]$

$$\begin{cases} \sup\limits_{\forall t} \|x_{1,i}(t)\| \leq \tilde{\rho} \\ \sup\limits_{\forall t} \left\|x_{1,i}(t) - x_{1,i}^*(t)\right\| \leq \eta_1^* \max\left\{\dfrac{ln(\beta)}{\beta},\, \dfrac{1}{\beta}\right\} \\ \|\hat{x}_{2,i}(t) - x_{2,i}(t)\| \leq \dfrac{\eta_2^*}{\beta} \;\forall t \in \left[t_j + \eta_3^* \max\left\{\frac{ln(\beta)}{\beta}, 0\right\}, t_{j+1}\right) \end{cases} \tag{74}$$

Remark 2. Note that A1-A2 always hold in most quadrotor platforms. Furthermore, since $b_i > 0$ by definition, choosing $\bar{b}_i > 0$ assures A3 and, therefore, the stability is guaranteed. The error between the system (69) and the model reference (71) can be reduced by increasing β or k_d.

Remark 3. The produced control action, u_i, can be limited by smoothing the input reference, Ω_r'. This can be done through the parameter r.

4.3 Simulation Results

To get a better insight into the proposal some simulations are presented. The non-linear quadrotor model has been introduced accordingly to (60) with $B = diag(13.5, 13.5, 7)$. The disturbance term, $f(\Omega_b, t)$, has been introduced as in (61) where all the "unknown" coefficients has been approximately chosen to represent a real quadrotor.

4.3.1 Inner control loop

For the LADRC design an estimation of B is needed. This estimation has been set to $\bar{B} = diag(1, 1, 1)$ which represents a considerable modeling error. The feedback gain (71)-(72) is taken as $k_d = 5$, and the observer (70) bandwith $\beta = 50$. The TD (73) limitation parameter has been chosen to be $r = 1$.

Figure 12 shows a comparison between the system (62) response, $x_{1,i}$, and the reference model (71), $x_{1,i}^*$, when the system initial state is $x_{1,i} = x_{1,i}^* = -1$.

In Figure 13, the TD (73) performance can be seen. The control system will receive Ω_r' which is a smoothed version of the step Ω_r. Therefore, overshoots and high values in the control actions will be avoided.

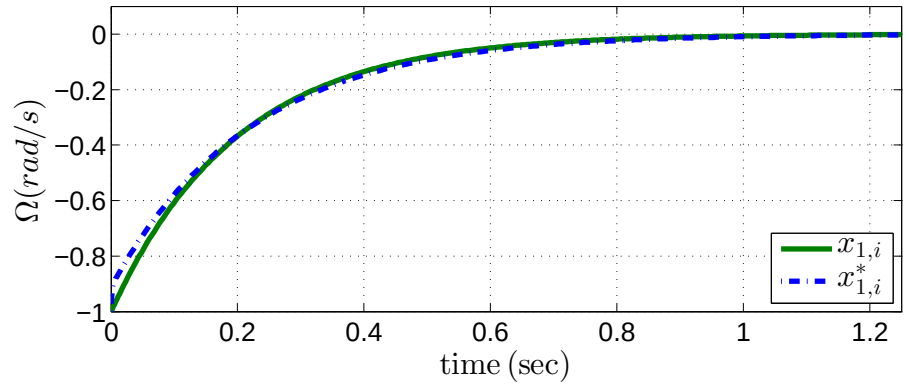

Figure 12: Comparison between system response and reference model.

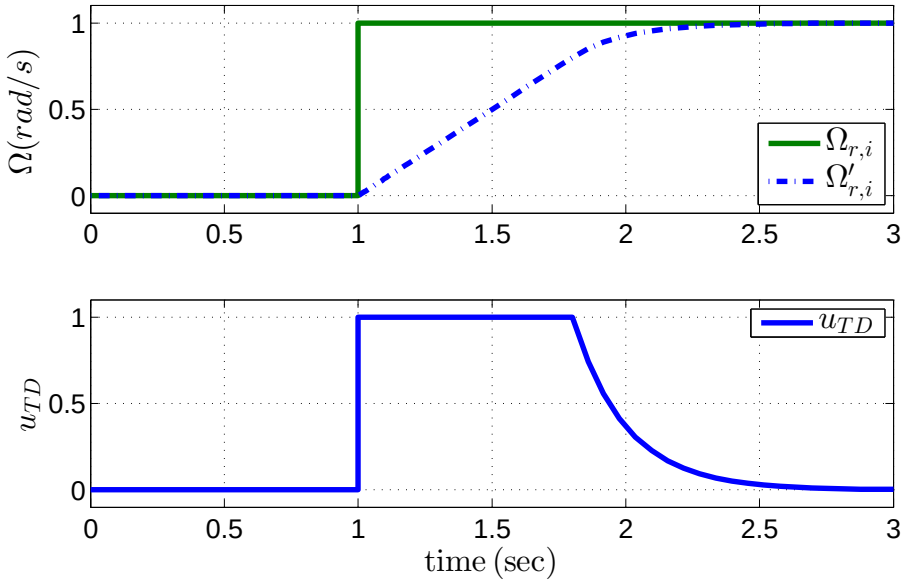

Figure 13: TD performance when a step is introduced to the system.

The whole dynamic system step response can be seen in Figure 14 where the step reference value is $\Omega_r = (1.25,\ 1,\ 0.75)^T$ (rad/sec). It is important to remark that three LADRC has been simulated in parallel to control the three axis ($i = x, y, z$) of the quadrotor. As it is shown, the effect of the TD on Ω_b is present and, as a consequence, the control actions are well limited as depicted in the middle plot. The third plot shows the total disturbance in each axis and its estimation.

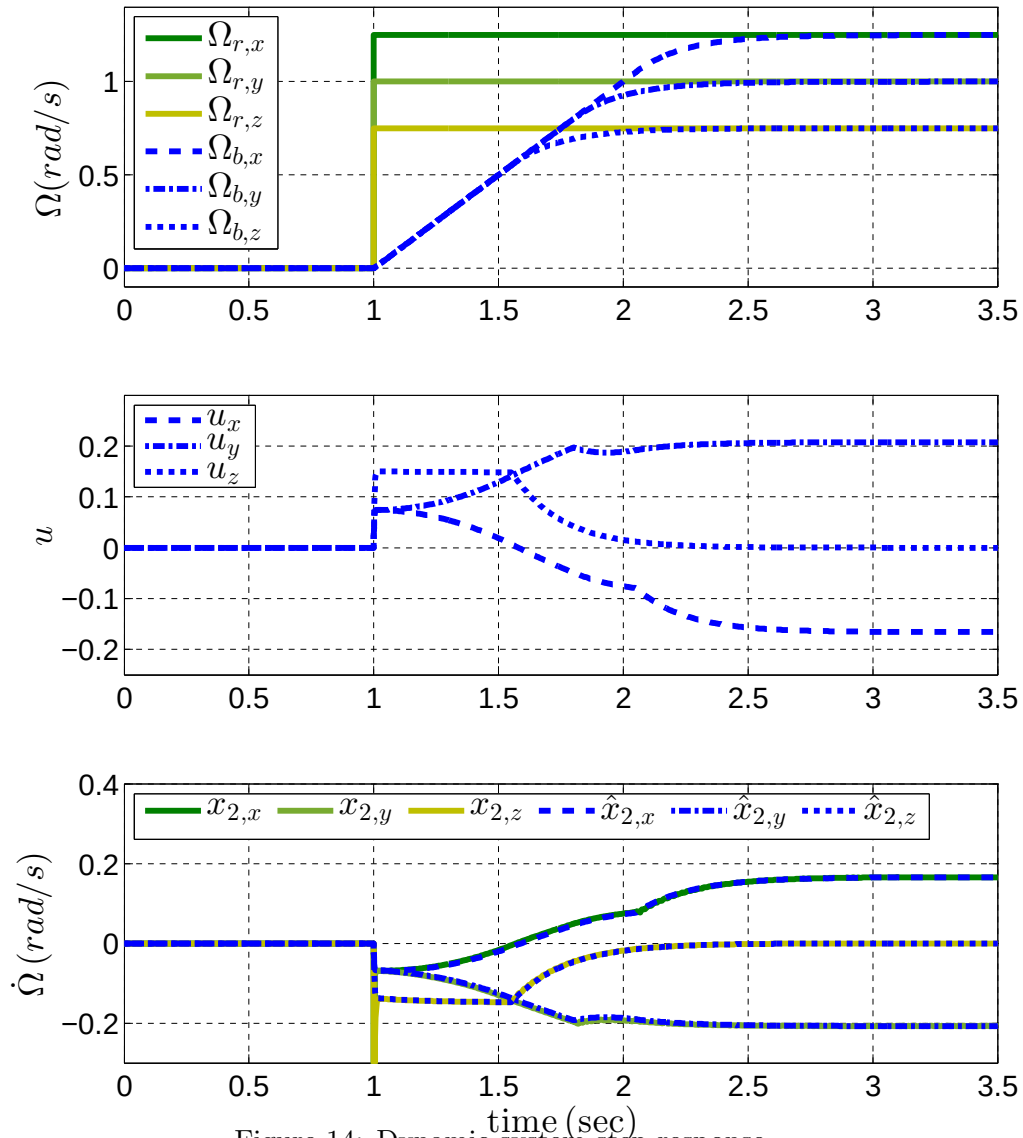

Figure 14: Dynamic system step response.

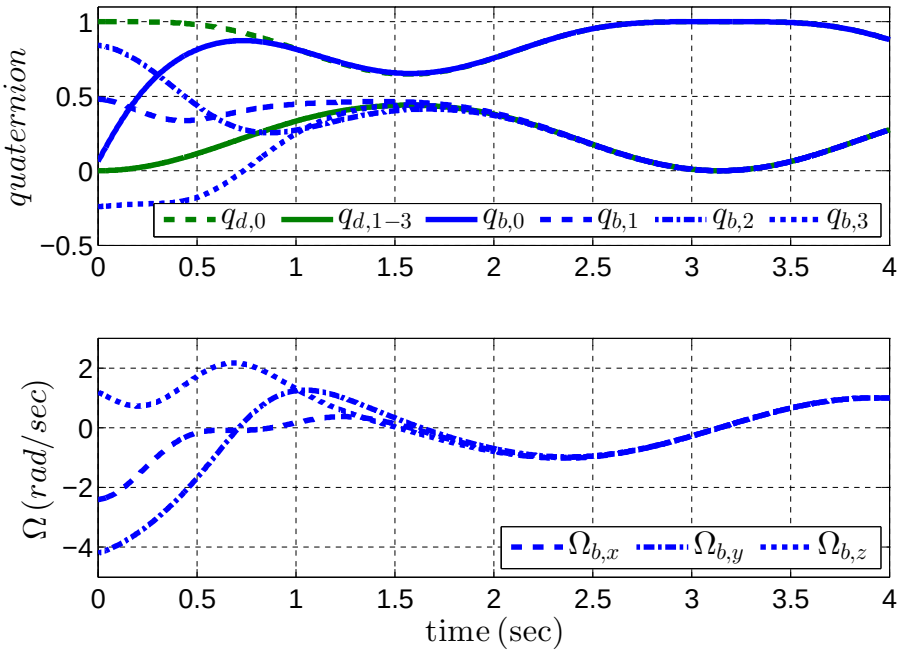

Figure 15: Simulation of the pure kinematic control loop under (63).

4.3.2 Outer control loop

The pure kinematic closed-loop system has been simulated under (63) with $k_k = 5$. As it has been said, this ignores the system dynamics and it is assumed that $\Omega_b = \Omega_r$. The results can be seen in Figure 15 where the trajectory has been generated doing $\Omega_d = \sin(2t)(1,\ 1,\ 1)^T$ and the initial conditions have been set to $q_d(0) = (1,\ 0,\ 0,\ 0)$, $q_b(0) = (0.06,\ 0.48,\ 0.84,\ -0.24)$, which corresponds to an initial attitude error of $\theta = 173.12°$.

Figure 16 shows the same simulation as in Figure 15 but the dynamics has been considered. That means that the LADRC dynamic loop has been incorporated to drive the angular velocity of the quadrotor, Ω_b, in order to follow Ω_r. For this simulation, the LADRC parameters have been set to: $k_d = 5$, $\beta = 50$, $r = 10$. It can be seen how the assumption $\Omega_r \approx \Omega_d$ approximately holds.

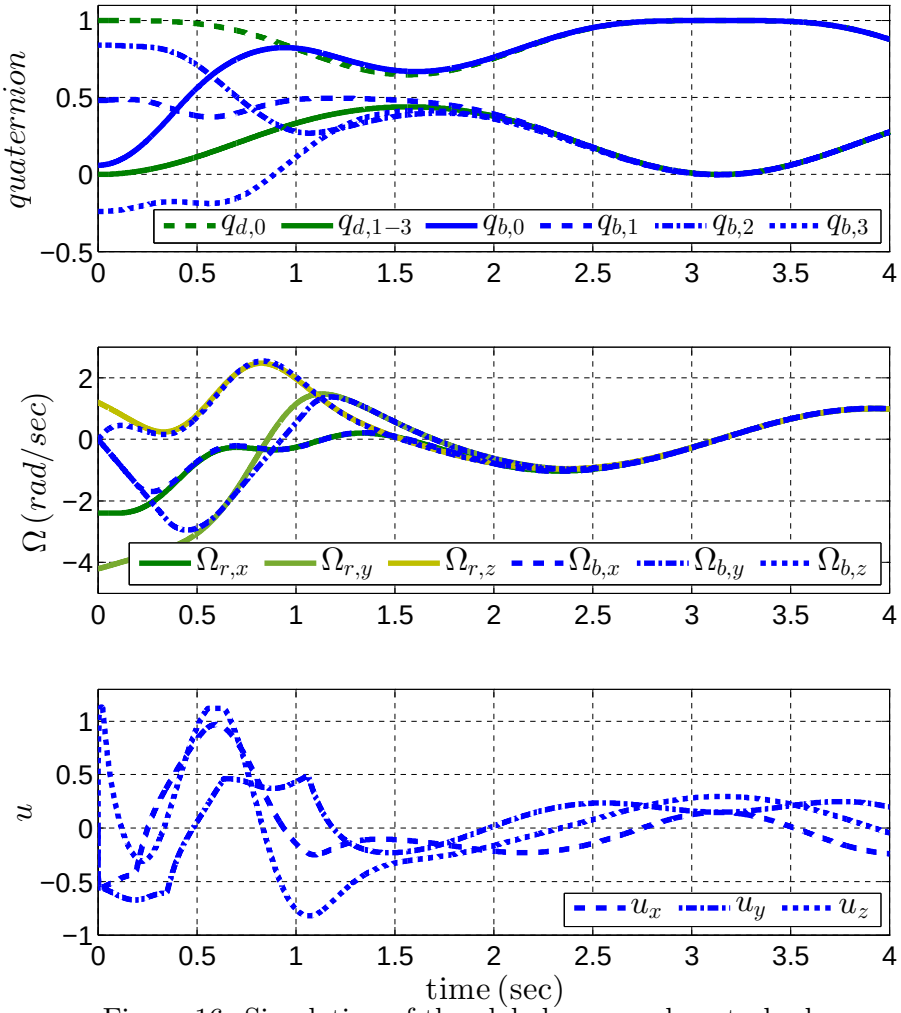

Figure 16: Simulation of the global proposed control scheme.

5 Obtaining quaternion measurements from sensor observations

This section aims to solve an important problem involving quaternions, which is how to obtain a real-time quaternion measurement from low-cost sensors and employing C++ as programming language. This work is particularized for a quadrotor, but it can be generalized to any Remotely Piloted Aircraft Systems (RPAS). The only restriction is that the RPAS needs to be equipped with a 3-axis accelerometer, a 3-axis gyroscope, and a 3-axis magnetometer.

Getting an accurate quaternion measurement is a problem which can be divided into two steps. The first one is to get the attitude quaternion from of the earth gravity and the earth magnetic field projections. This problem is referred as the problem of attitude determination from vector observations (also known as Wahba's problem) [29, 30]. The second step is to improve the quaternion measure by filtering techniques. For that purpose, it is very common to design observers based on the kinematic model (48) which greatly enhance the quality of the quaternion measure. Figure 17 shows a block diagram of the whole estimation process.

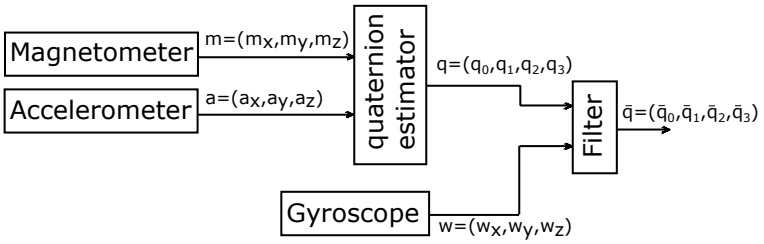

Figure 17: Quaternion estimation form sensor data. Block diagram.

It is important to remark that the magnetometer needs to be calibrated in the three axis and also needs to be placed far enough from potential magnetic disturbances (motors, high currents,...), which could degrade the local earth magnetic field.

This Section is structured as follows. First, as a 3-axis calibrated magnetometer is needed, in Section 5.1 an easy and accurate technique to carry out the calibration with MatLab is explained. In Section 80 the Wahba's problem of attitude determination from vector observations is presented. Also, the Davenport's solution (also known as Davenport's q-Method) is explained. With the Davenport's q-method we will be able to estimate the quaternion measure from earth gravity and magnetic field projections. In Section 5.3, a non-linear observer based on the kinematic model is presented as a way of filtering the quaternion measure and estimating the gyroscope biases. In Section 5.4, a MatLab and C++ code implementing the Dav-

enport's q-Method and the non-linear observer is given. The C++ code is based on the *Eigen library* which greatly simplifies working with matrices and quaternions. Finally, in section 5.5, some experimental results are presented in order to show the performance of the whole quaternion estimation algorithm. Computational costs are also analyzed.

5.1 3-D magnetometer calibration

The magnetic field measures, as well as acceleration, needs to be normalized to fit an unitary sphere centered at zero. By default, the magnetometer raw measurements fit a 3-D ellipsoid (as the one depicted in Figure 18). Additionally, the ellipsoid is not centered at zero and could be also rotated.

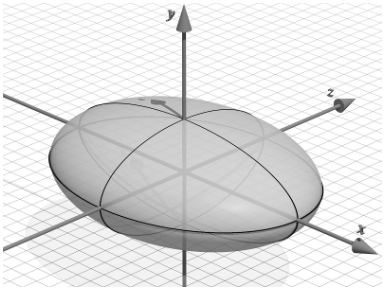

Figure 18: Ellipsoid.

If $m^{raw} = (m_x^{raw}, m_y^{raw}, m_z^{raw})^T$ is the magnetic raw data coming from the sensor, and $m = (m_x, m_y, m_z)^T$ is the calibrated data which fits a unit sphere, the relationship between them is given by:

$$m = E \cdot R^T \cdot (m^{raw} - offset) \tag{75}$$

where the $offset$ is the position of the ellipsoid center, R is the rotation matrix associated its principal axes, and $E = diag(e_1, e_2, e_3)$ is a scaling factor in each axis.

In order to perform the calibration we need to find the values of E, R and the $offset$. Here, for that purpose, a post-processing technique will be used. That is, a big set of raw data points will be extracted from the aerial vehicle via a *.txt* file and those points will be used to get the parameters by least squares approximation.

In order to get an accurate calibration, the measurements needs to be taken in all the possible 3-D orientations. So the vehicle needs to be rotated in all directions when the data is being sampled. Figure 19 shows an example of magnetic data acquisition.

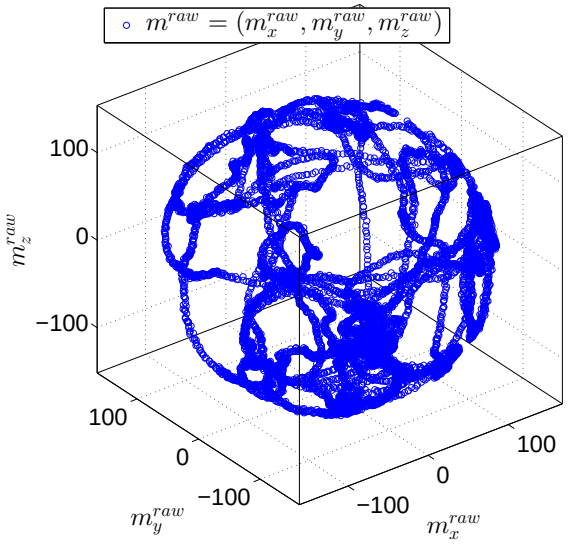

Figure 19: 3-D raw magnetic data acquisition.

5.1.1 MatLab code to get the calibration parameters

Here it is shown how to get the calibration parameters using MatLab and its *ellipsoid_fit* function [28].

```
%% Load magnet data. Saved as .txt file
load_Magnet_calibration_Data
% Loads 3 vectors: mx_raw, my_raw, mz_raw.
%                  = 3-axis raw magnetometer reads.

%% Least squares ellipsoid fitting
[offset, radio, R, v, chi2] = ellipsoid_fit([mx_raw my_raw
    mz_raw]); % "help ellipsoid_fit" for more details

offset  % Ellipsoid center.
R  % Rotation matrix. We need to change the order of columns in
    R in order to get the higher values in the diagonal.
E = diag([1/radio(1) 1/radio(2) 1/radio(3)])  % Scaling factor.
```

As an example, with the acquired data in Figure 19, and running the MatLab code shown above, the obtained parameters are:

$$offset = \begin{pmatrix} 0.4045 & -0.5655 & 0.0772 \end{pmatrix}^T \tag{76}$$

$$R = \begin{pmatrix} -0.9550 & 0.2965 & -0.0109 \\ 0.2966 & 0.9529 & -0.0641 \\ 0.0086 & 0.0644 & 0.9979 \end{pmatrix} \tag{77}$$

$$radio = \begin{pmatrix} 165.75 & 163.69 & 148.84 \end{pmatrix}^T \tag{78}$$

$$E = \begin{pmatrix} 0.006 & 0 & 0 \\ 0 & 0.0061 & 0 \\ 0 & 0 & 0.0067 \end{pmatrix} \tag{79}$$

where the *radio* is the magnitude of the three ellipsoid axes.

5.1.2 Applying calibration parameters to magnet raw data

The following MatLab code (which is easy to implement in C++) applies the calibration given by (75).

```
%% Load magnet data. Saved as .txt file
load_Magnet_calibration_Data
% Loads 3 vectors: mx_raw, my_raw, mz_raw.
%                  = 3-axis raw magnetometer reads.

%% Correction parameters
offset = [0.4045 -0.5655 0.0772];
R = [-0.9550 0.2965 -0.0109;...
      0.2966 0.9529 -0.0641;...
      0.0086 0.0644  0.9979];
E = diag([0.006 0.0061 0.0067]);

%% Correcting data
for i=1:1:size(mx_raw,1) % The calibration is applied to all
    points
    % Centering the ellipsoid at zero.
    m(1,i) = mx_raw(i) - offset(1);
    m(2,i) = my_raw(i) - offset(2);
    m(3,i) = mz_raw(i) - offset(3);

    % Applying the inverse rotation.
    m(:,i) = R'*m(:,i);

    % Escaling axes to get a sphere.
    m(:,i) = E*m(:,i);
end
```

The results can be seen in Figure 20, where all the points m^{raw} have been transformed to fit a unitary sphere.

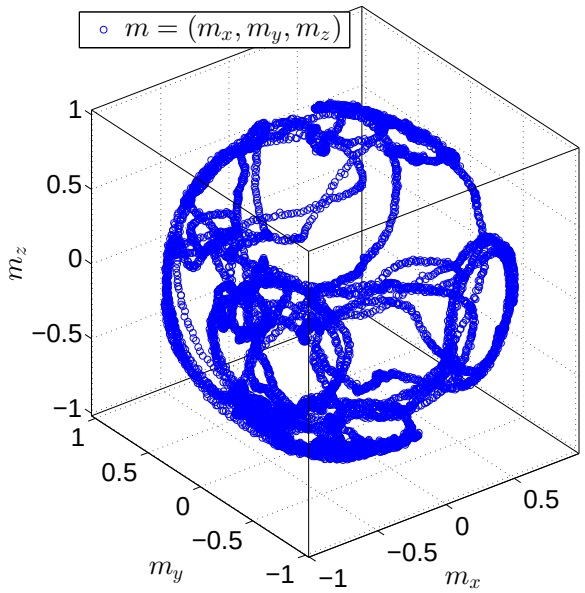

Figure 20: 3-D calibrated magnetic data.

5.2 Wahba's problem

The unit vectors of the earth gravity, $a = (a_x, a_y, a_z)$, and the calibrated magnetic field, $m = (m_x, m_y, m_z)$, contain information about the body attitude. The question is if there is any way to determine the complete quadrotor attitude using these measurements. In fact, this is a famous problem which was proposed in 1965 by Grace Wahba [29]. The Wahba's problem states that if we have a set of i measurable vectors (for example the gravity and the earth magnetic field, or others) and, b_i is the measurement of the vector i in the body frame, and r_i is the expected value in some reference frame; the problem is to find an orthogonal matrix R, satisfying $det(R) = +1$, that minimizes the following loss function:

$$L(R) = \frac{1}{2} \sum_i a_i |b_i - Rr_i|^2 \tag{80}$$

where $a_i > 0$ are weighting factors.

Note that R is the rotation matrix representing the body attitude with respect to the reference frame, and $|b_i - Rr_i|$ is the error between the measured vector and the expected value. For example, the expected value of the earth gravity in the inertial frame, $\mathcal{I}$, is $a = (0,\ 0,\ 1)^T$. If we measure $a = (0.707, 0.707, 0)^T$ is because the body frame, $\mathcal{B}$, is in a different orientation. So we need to find the matrix R that makes $R \cdot (0,\ 0,\ 1)^T \approx (0.707, 0.707, 0)^T$. The loss function (80) can also be written as:

$$L(R) = \lambda_0 - tr(RB^T) \tag{81}$$

with

$$\lambda_0 = \sum_i a_i \tag{82}$$

and

$$B = \sum_i a_i b_i r_i^T \tag{83}$$

so finding the minimum of $L(R)$ is equivalent to find the maximum of $tr(RB^T)$.

Problem (80)-(83) is a general problem and, depending on the aircraft hardware, different kind of vectors b_i, r_i can be used. It is known that, at least, two vectors are needed to solve it. So, for the quadrotor case, b_1 and b_2 needs to be chosen as $b_1 = a$, $b_2 = m$. The expected values of a and b in the inertial reference frame are $r_1 = (0,\ 0,\ 1)^T$, $r_2 = (0.401,\ 0,\ 0.916)^T$

5.2.1 Solution: Davenport's q-Method

Many solutions have been proposed to Wahba's problem [30], but Davenport's solution is the unique which is useful due to its computational efficiency.

He rewrited the rotation matrix in terms of a unit quaternion:

$$q = \begin{pmatrix} cos(\theta/2) \\ \bar{\mathbf{r}} \cdot sin(\theta/2) \end{pmatrix} = \begin{pmatrix} q_0 \\ \mathbf{q}_v \end{pmatrix} = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^T \tag{84}$$

$$R(q) = (q_3^2 - |\mathbf{q}_v|^2)I + 2\mathbf{q}_v\mathbf{q}_v^T - 2q_3[\mathbf{q}_v\times] \tag{85}$$

With such representation, as $R(q)$ is a homogeneous quadratic function of q, the trace can be written as:

$$tr(RB^T) = q^T K q \tag{86}$$

being

$$K = \begin{pmatrix} S - Itr(B) & z \\ z^T & tr(B) \end{pmatrix} \tag{87}$$

$$S = B + B^T \tag{88}$$

$$z = \sum_i a_i b_i \times r_i = \begin{pmatrix} B_{23} - B_{32} \\ B_{31} - B_{13} \\ B_{12} - B_{21} \end{pmatrix} \tag{89}$$

The quaternion representing the optimal attitude which minimizes L(R) is the eigenvector of K corresponding to the largest eigenvalue. That is, the solution of:

$$Kq_{opt} = \lambda_{max} q_{opt} \tag{90}$$

So finding the attitude which best fits with the measurements a, m is reduced to find an eigenvector of a matrix. Unless many forms have been proposed [30] to solve (90) in a recursive way (easy to program), here, in section 5, it will be directly solved by means of *Eigen C++* library.

5.3 Improving measurement: Including gyroscopes

Equation (90) gives the optimal quaternion measure according to the information contained in a and m. However, during flight, the motors generates vibrations and, therefore, a and m are normally measurements with high levels of noise. Thus, q would be also corrupted by noise.

In order to improve the quality of this measure, the gyroscopes information can be used to filter it. But gyros are neither a perfect measure. They are corrupted by biases that could cause estimation errors if they are not taken into account.

For those reasons here it is proposed a non-linear observer based on the quaternion kinematic model (48) which uses the quaternion measure generated by (90) and the gyroscopes in order to estimate the "true" quaternion attitude and the gyros biases.

5.3.1 Propagation of the quaternion-based kinematic model

As it has been mentioned, the quaternion-based kinematic model is given by the following equation:

$$\dot{q(t)} = \frac{1}{2} q(t) w(t) \tag{91}$$

where $q = (q_0\ q_1\ q_2\ q_3)^T$ is the attitude quaternion, $w = (0,\ \Omega) = (0,\ \Omega_x,\ \Omega_y,\ \Omega_z)^T$ is the quaternion associated with the angular velocity, Ω. The quaternion product rule can is given by:

$$qp = \begin{pmatrix} q_0p_0 - q_1p_1 - q_2p_2 - q_3p_3 \\ q_0p_1 + q_1p_0 + q_2p_3 - q_3p_2 \\ q_0p_2 - q_1p_3 + q_2p_0 + q_3p_1 \\ q_0p_3 + q_1p_2 - q_2p_1 + q_3p_0 \end{pmatrix} \tag{92}$$

Equation (91) can be used to propagate the attitude quaternion. For that purpose, it is important to know what is the solution $q(t)$ of (91). In order to find the solution, let us state the following three lemmas:

Lemma 1. (Quaternion exponential (32)) The time-invariant quaternion differential equation $\dot{q} = aq$, with $a = (a_0\ a_1\ a_2\ a_3)^T = cte$ and the initial condition $q(0) = (1\ 0\ 0\ 0)^T$, has as unique solution $q(t) = e^{at}$, being e^{at} the quaternion exponential function defined by:

$$e^{at} = \begin{pmatrix} e^{a_0 t}\cos(\bar{a}t) \\ e^{a_0 t}\dfrac{a_1}{\bar{a}}\sin(\bar{a}t) \\ e^{a_0 t}\dfrac{a_2}{\bar{a}}\sin(\bar{a}t) \\ e^{a_0 t}\dfrac{a_3}{\bar{a}}\sin(\bar{a}t) \end{pmatrix} \tag{93}$$

$$\bar{a} = \sqrt{a_1^2 + a_2^2 + a_3^2} \tag{94}$$

Lemma 2. If $\Phi(t)$ is the solution of the time-variant differential equation $\dot{q} = qa(t)$ with the initial condition $(1\ 0\ 0\ 0)^T$; then $q(t) = q_0\Phi(t)$ is the solution of $\dot{q} = qa(t)$ with the initial condition q_0.

Lemma 3. For the kinematic model (91), and considering $w(t) = w = cte$, the exact solution is given by $q(t) = q(t_0) \cdot e^{\frac{1}{2}\int_{t_0}^{t} w dt} = q(t_0) \cdot e^{\frac{1}{2}w(t-t_0)}$.

Proof. The proofs of the three Lemmas can be found in [32]. □

Note that the gyroscopes provide measures of Ω with a given period, T. So for $t \in [t_k,\ t_{k+1})$, the equation (91) can be approximated as:

$$\dot{q} \approx q \cdot w_k \tag{95}$$

being w_k constant in $t \in [t_k,\ t_{k+1})$. Therefore, by Lemmas 1-3, the following recursive propagation law for the attitude quaternion can be proposed:

$$q_{k+1} \approx q_k \cdot e^{\frac{1}{2}w_k T} \tag{96}$$

Remark 1. Note that the equality is not strictly certain because the gyroscopes only provide discrete information which is retained via a Zero Order Hold (ZOH). So, during $t \in (t_k,\ t_{k+1})$ it is assumed that $w_k \approx w(t)$.

Remark 2. If we had exact measures of w_k we could estimate the quaternion measures only with the gyroscopes. But the gyroscopes have biases and, therefore, the recursive law (96) is not convergent. For that purpose we need to construct an observer.

5.3.2 Non-linear observer

In [33,34] a non-linear observer based on (91) which is capable of estimating the gyro biases (and it is proved to be exponentially convergent) was proposed. In this document this observer will be employed in order to filter the measure provided by (90).

Let us define q as the unit quaternion measure coming from (90), $\hat{q}$ as the unit observed quaternion, and $\tilde{q} = (\hat{q})^* \cdot q = (\tilde{q}_0\ \tilde{\mathbf{q}})^T$ as the error between them. The angular velocity measured by the gyroscopes, Ω_g is corrupted by biases. So it can be written as:

$$\Omega_g = \Omega + \delta_\Omega \tag{97}$$

being Ω the true angular velocity and δ_Ω the true bias which is assumed to be constant.

Therefore, the observed angular velocity, $\hat{\Omega}$, is defined as:

$$\hat{\Omega} = \Omega_g - \hat{\delta}_\Omega \tag{98}$$

where $\hat{\delta}_\Omega$ is the observed bias.

With the aforementioned definitions, the observer can be written as:

$$\dot{\hat{q}} = \frac{1}{2} q \cdot (\tilde{q})^* \cdot \begin{pmatrix} 0 \\ \hat{\Omega} + k\,\mathrm{sign}(\tilde{q}_0)\tilde{\mathbf{q}} \end{pmatrix} \cdot \tilde{q} \tag{99}$$

$$\dot{\hat{\delta}}_{\Omega} = -\frac{1}{2}\mathrm{sign}(\tilde{q}_0)\tilde{\mathbf{q}} \tag{100}$$

with $k > 0$.

If measures of q and Ω_g are available at the same instant, and considering the Lemmas 1-3, the following approximate recursive law can be stated in order to implement the observer (99)-(100):

$$\hat{q}_k = \hat{q}_{k-1} \cdot e^{\frac{T}{2} r_{k-1}} \tag{101}$$

$$\hat{\delta}_{k,\Omega} = \hat{\delta}_{k-1,\Omega} - \frac{T}{2}\mathrm{sign}(\tilde{q}_{k-1,0})\tilde{\mathbf{q}}_{k-1} \tag{102}$$

$$\tilde{q}_k = (\hat{q}_k)^* \cdot q_k \tag{103}$$

$$\hat{\Omega}_k = \Omega_{k,g} - \hat{\delta}_{k,\Omega} \tag{104}$$

$$r_k = (\tilde{q}_k)^* \cdot \begin{pmatrix} 0 \\ \hat{\Omega}_k + k \cdot \mathrm{sign}(\tilde{q}_{k,0})\tilde{\mathbf{q}}_k \end{pmatrix} \cdot \tilde{q}_k \tag{105}$$

5.4 Programming the quaternion estimator

In this section, a code example for the quaternion estimator described in Sections 80-5.3 is given. This code is first given in MatLab language in order to get a first insight, then a C++ version employing *Eigen library* is also given. For the C++ code, computational costs are also analyzed.

5.4.1 MatLab code

The following MatLab code gets the quaternion measure, q, by the Davenport's q-method (described in Section 80). The measures a, m are supposed to be loaded from a file, and m is also supposed to be calibrated in the three axis as it has been shown in Section 5.1.

```
load_data % Load observation data.
% Gives a(i,:)=(ax(i),ay(i),az(i))
%       m(i,:)=(mx(i),my(i),mz(i))
%       w(i,:)=(wx(i),wy(i),wz(i))

%% Initializations
b1 = [0 0 1]';
%expected aceleration in the Inertial reference frame
b2 = [0.4 0 0.916]';
%expected normalized magnetic field in the Inertial reference
    frame
a1 = 1; a2 = 1;
%weighing factors

```

```
%% Get quaternion measure from vector observations
q = zeros(size(t,1),4);
for i=1:1:size(t,1)
    % observation vectors
    r1 = a(i,:)';
    r2 = m(i,:)';
    % quaternion measure
    q(i,:) = Davqmethod(r1,r2,b1,b2,a1,a2);
end
```

```
% Gives the quaternion measure from accelerometer and
    magnetometer observations.
%
% inputs: ri,bi,ai; i=1,2.
% return: observed quaternion

function [ qobs ] = Davqmethod(r1,r2,b1,b2,a1,a2)

    B = a1*b1*r1' + a2*b2*r2'; % getting B matrix

    % assembling K matrix
    S = B + B';
    z = [B(2,3)-B(3,2);B(3,1)-B(1,3);B(1,2)-B(2,1)];
    K = [S - eye(3,3)*trace(B) z;z' trace(B)];

    % getting the maximum eigenvalue and its associated
        eigenvector
    [V,D] = eigs(K);
    for j=1:1:4
        landa(j) = D(j,j);
    end
    for j=1:1:4
        if(D(j,j) == max(landa))
            q(1) = V(4,j);
            q(2) = -V(1,j); % We want the conjugated attitude
            q(3) = -V(2,j); % (from inertial to body)
            q(4) = -V(3,j);
        end
    end

   % q and -q represent the same physical attitude.
   % We chose 0<= q <= 1
    if(q(1)<0)
        q = -q;
    end

    qobs = q;

end
```

Next, this code is extended in order to incorporate the observer described in Section 4. For that purpose the following functions need to be defined: quaternion multiplication (*qmult*), quaternion conjugate (*qconj*), quaternion

exponential (*qexp*):

```
% Quaternion multiplication. r = q*p
function [r] = qmult(q,p)
    r(1) = q(1)*p(1) - q(2)*p(2) - q(3)*p(3) - q(4)*p(4);
    r(2) = q(1)*p(2) + q(2)*p(1) + q(3)*p(4) - q(4)*p(3);
    r(3) = q(1)*p(3) - q(2)*p(4) + q(3)*p(1) + q(4)*p(2);
    r(4) = q(1)*p(4) + q(2)*p(3) - q(3)*p(2) + q(4)*p(1);
end
```

```
% Quaternion conjugate. conj_q = (q)*
function [conj_q] = qconj(q)
    conj_q = [q(1) -q(2) -q(3) -q(4)];
end
```

```
% Quaternion exponential.
function [eq] = qexp(q)
    sigma = q(1);
    w1    = q(2);
    w2    = q(3);
    w3    = q(4);
    exp_q = zeros(1,4);

    a = sqrt(w1^2 + w2^2 + w3^2);
    exp_q(1) = exp(sigma)*cos(a);

    if(a~=0)
        exp_q(2) = exp(sigma)*(w1/a)*sin(a);
        exp_q(3) = exp(sigma)*(w2/a)*sin(a);
        exp_q(4) = exp(sigma)*(w3/a)*sin(a);
    else
        exp_q(2) = 0;
        exp_q(3) = 0;
        exp_q(4) = 0;
    end

    eq = exp_q;
end
```

With these functions, the complete MatLab code implementing the Davenport's q-method and the observer is:

```
load_data % Load observation data.
% Gives a(i,:)=(ax(i),ay(i),az(i))
%       m(i,:)=(mx(i),my(i),mz(i))
%       w(i,:)=(wx(i),wy(i),wz(i))
%       T = sensors period

%% Initializations
b1 = [0 0 1]'; %expected aceleration in the Inertial reference
    frame
b2 = [0.4 0 0.916]'; %expected normalized magnetic field in the
    Inertial reference frame
a1 = 1; a2 = 1; %weighing factors
```

```
k = 4; %observer gain.

hat_q(1,:) = [1 0 0 0]; %initial value of hat_q
hat_b(1,:) = [0 0 0]; %initial value of hat_b
r = [0 0 0 0]'; %initial value of r
tilde_q = [1 0 0 0]; tilde_q_o = tilde_q(1); b_tilde_q = tilde_q
    (2:4); %initial value of tilde_q
q(1,:) = [1 0 0 0]; %initial value q

%% Get quaternion measure from vector observations
for i=2:1:size(t,1)
    % observation vectors
    r1 = a(i,:)';
    r2 = m(i,:)';
    Omega_g = gyr(i,:);

    % quaternion measure
    q(i,:) = Davqmethod(r1,r2,b1,b2,a1,a2);

    % Observer
    hat_q(i,:) = qmult(hat_q(i-1,:),qexp(T/2*r));
         %observed quaternion
    hat_b(i,:) = hat_b(i-1,:) - T/2*sign(tilde_q_o)*b_tilde_q;
         %observed gyro bias

    tilde_q = qmult(qconj(hat_q(i,:)),q(i,:));
    tilde_q_o = tilde_q(1);
    b_tilde_q = tilde_q(2:4);
    hat_Omega = Omega_g - hat_b(i,:);
    r = qmult(qconj(tilde_q),qmult([0 [hat_Omega + k*sign(
        tilde_q_o)*[b_tilde_q]]],tilde_q));
end
```

5.4.2 Eigen library

Eigen [31] is an open-source library which is very interesting if we need to work with matrices in C++. It supports all matrix sizes, from fixed-size to dynamic-size. It also supports all kind of numeric types such as: integers, complex or quaternions.

Furthermore, it has a very large number of functions for matrix decomposition or geometry features. It has been developed for several years, so it is fast and reliable. The main benefit is that you can operate with matrices in C++ as in MatLab.

It can be included in any project as a header. In [31] all the information, examples and tutorial are given. In order to program the algorithm in C++ we will use the Eigen facilities.

5.4.3 C++ code

This is the C++ code that reads the sensors and computes the quaternion attitude measure.

```
/**     Initializations     */
    static Vector3f b1(0,0,1),          //global frame expected
        aceleration
                    b2(0.4,0,0.916);    //global frame expected
                        magnetic field

    static float    a1 = 1,     //weigh for accelerometer
                    a2 = 1;     //weigh for magnetometer

    static float    k = 4;      //non-linear observer gain


    /**     Sensors reading     */
    gyrx = (IMU_MPU->Get_gyrx());     // rad/s
    gyry = (IMU_MPU->Get_gyry());
    gyrz = (IMU_MPU->Get_gyrz());

    ax = (IMU_MPU->Get_accx()) - X_ACEL_OFFSET; // g/g
    ay = (IMU_MPU->Get_accy()) - Y_ACEL_OFFSET;
    az = (IMU_MPU->Get_accz());

    if(Magnet->NewMeas() == true){
        r2 = Magnet->GetMeasure();
    }

    /**     Assembling vectors  */
    r1 << ax, ay, az;
    r2 << magX, magY, magZ;
    w_gyr << gyrx, gyry, gyrz;

    /**  Davenports q-method    */
    q_meas = Dav_qmethod(r1,r2,b1,b2,a1,a2);

    /**  Non-linear observer    */
    aux_q = q_obs;
    aux_b = bias_obs;

    q_obs = aux_q*qexp(vect2q(Ts/2*q2vect(a)));
    bias_obs = aux_b - Ts/2*epsilon_err*sign(eta_err);

    w = w_gyr - bias_obs;
    q_err = quat_error(q_meas,q_obs);
    eta_err = q_err.w();
    epsilon_err = q_err.vec();

    w_quat.w()   = 0;
    w_quat.vec() = w + k*epsilon_err*sign(eta_err);
    a = q_err*w_quat*q_err.inverse();

```

```
48      if(q_obs.w() < 0){
49          q_att.w() = -q_obs.w();
50          q_att.vec() = -q_obs.vec();
51      }
52
53      else{
54          q_att = q_obs;
55      }
56
57      /** q_att = quaternion attitude */
```

In what follows, all the functions that have been used in the previous code are developed.

```
1  Vector4f EQ::q2vect(Quaternionf q){
2      Vector4f quat;
3      Vector3f q_v;
4
5      quat(0) = q.w();
6      q_v = q.vec();
7
8      quat(1) = q_v(0);
9      quat(2) = q_v(1);
10     quat(3) = q_v(2);
11
12     return quat;
13 }
14
15 Quaternionf EQ::vect2q(Vector4f v){
16     Quaternionf q;
17     Vector3f v_3;
18
19     q.w() = v(0);
20     v_3(0) = v(1);
21     v_3(1) = v(2);
22     v_3(2) = v(3);
23     q.vec() = v_3;
24
25     return q;
26 }
```

```
1  Quaternionf EQ::qexp(Quaternionf q,char flag){
2
3      Quaternionf exp_q;
4      Vector3f vect, exp_v;
5      float sigma, exp_sigma, w, sinw;
6
7      sigma = q.w();
8      vect  = q.vec();
9
10     w = vect.norm();
11     sinw = sin(w);
12     exp_sigma = exp(sigma)*cos(w);
13
```

```
    if(w != 0){
        exp_v(0) = exp(sigma)*(vect(0)/w)*sinw;
        exp_v(1) = exp(sigma)*(vect(1)/w)*sinw;
        exp_v(2) = exp(sigma)*(vect(2)/w)*sinw;
    }

    else{
        exp_v(0) = 0;
        exp_v(1) = 0;
        exp_v(2) = 0;
    }

    exp_q.w() = exp_sigma;
    exp_q.vec() = exp_v;

    return exp_q;

}
```

```
Quaternionf EQ::Dav_qmethod(Vector3f r1,Vector3f r2,Vector3f b1,
    Vector3f b2,float a1,float a2){

    static Vector3f Z;
    static Vector4f eigenvalues, eig_vector;
    static Matrix3f B, S;
    static Matrix4f K, eigenvectors;
    static float eig_max;

    B = a1*b1*r1.transpose() + a2*b2*r2.transpose();
    S = B + B.transpose();
    Z << (B(1,2)-B(2,1)),  (B(2,0)-B(0,2)),  (B(0,1)-B(1,0));

    K << (S - (B.trace()*MatrixXf::Identity(3,3))),   Z      ,
                 Z.transpose()                     , B.trace();

    SelfAdjointEigenSolver<Matrix4f> eigensolver(K);
    if (eigensolver.info() != Success){
        abort();
        cout << "Eigenvalue error!." << endl;
    }

    eigenvalues  = eigensolver.eigenvalues();
    eigenvectors = eigensolver.eigenvectors();

    eig_max = eigenvalues.maxCoeff();

    int j=0;
    while (j<4){
        if(eigenvalues(j) == eig_max){
            eig_vector(0) = eigenvectors(3,j);
            eig_vector(1) = -eigenvectors(0,j);
            eig_vector(2) = -eigenvectors(1,j);
            eig_vector(3) = -eigenvectors(2,j);
        }
```

```
35          j++;
36      }
37
38      if(eig_vector(0) < 0){
39          eig_vector(0) = -eig_vector(0);
40          eig_vector(1) = -eig_vector(1);
41          eig_vector(2) = -eig_vector(2);
42          eig_vector(3) = -eig_vector(3);
43      }
44
45      return vect2q(eig_vector);
46
47  }
```

```
1   Quaternionf EQ::quat_error(Quaternionf q1,Quaternionf q2){
2       Vector4f q_error, auxq1;
3       Matrix4f K;
4       Quaternionf err;
5
6       auxq1 << q1.vec(),q1.w();
7
8       K << q2.w()*MatrixXf::Identity(3,3) - crossp_mat(q2.vec()),
            -q2.vec(),
9                                                   q2.vec().transpose(),
                                                            q2.w();
10
11
12      q_error = K*auxq1;
13
14      err.w() = q_error(3);
15      err.vec()(0) = q_error(0);
16      err.vec()(1) = q_error(1);
17      err.vec()(2) = q_error(2);
18
19      return err;
20
21  }
```

```
1   Matrix3f EQ::crossp_mat(Vector3f v){
2
3       Matrix3f aux;
4
5       aux <<      0, -v(2),  v(1),
6              v(2),      0, -v(0),
7             -v(1),  v(0),      0;
8
9       return aux;
10
11  }
```

5.5 Results, experimental validation

In order to illustrate (and validate) the quaternion estimator described above, an experiment has been done rotating the quadrotor around a known axis. The sensors raw data has been saved in a *.txt* file. The Matlab codes described in Section 5 will be applied to this raw data in order to see the quaternion estimator performance.

First of all, it is important to define clearly which are the quadrotor axis. These axis are defined by the IMU orientation, so it is important to orientate precisely the IMU in order to fit with the desired quadrotor axis (body frame, $\mathcal{B}$). Figure 21 shows the quadrotor x-y axis according to the data that will be presented next. The z-axis is pointing down in order to form an orthogonal Cartesian coordinate system.

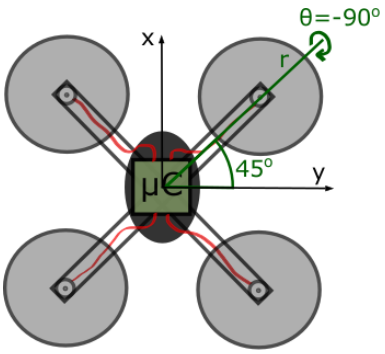

Figure 21: Quadrotor axis in a X-controlled platform.

The following experiment has been done: The quadrotor has been rotated $\theta \approx -90^\circ$ around the axis r (defined in Figure 21). This axis is given by its unitary vector $u = \sqrt{2}/2 \cdot (1,\, 1,\, 0) \approx (0.707,\, 0.707,\, 0)$. Therefore the expected quaternion at the final orientation is:

$$q = \begin{pmatrix} \cos(\theta/2) \\ u \sin(\theta/2) \end{pmatrix} \approx \begin{pmatrix} 0.707 \\ -0.5 \\ -0.5 \\ 0 \end{pmatrix} \tag{106}$$

the expected quaternion (106) will be used to validate the results.

Figure 22 shows the raw sensor data that has been extracted after performing the experiment. The first plot depicts the 3-axis accelerometer measurements. The middle plot shows the uncalibrated magnetometer reads, and the third plot shows the gyroscopes readings. As the rotation has been performed around the r axis, which is rotated 45°, we have that $\Omega_x \approx \Omega_y$ and $\Omega_z \approx 0$.

The first step is to apply (75) with the obtained parameters (76)-(79) to the magnetometer raw measurements in order to get the calibrated data

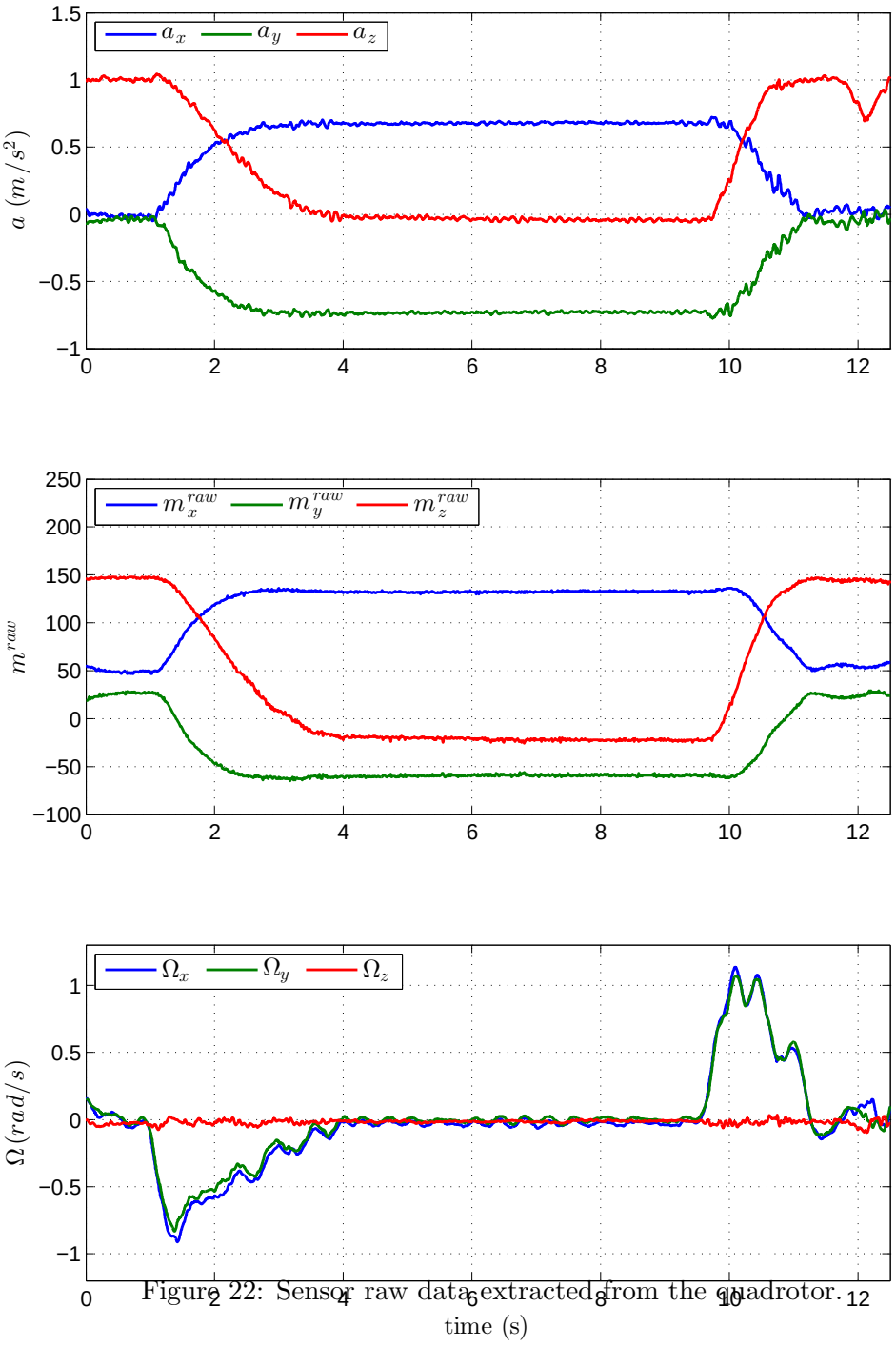

Figure 22: Sensor raw data extracted from the quadrotor.

which is valid for the quaternion estimator algorithm. This can be seen in Figure 23, where the first plot depicts the raw measurements and the second plot represents the calibrated data.

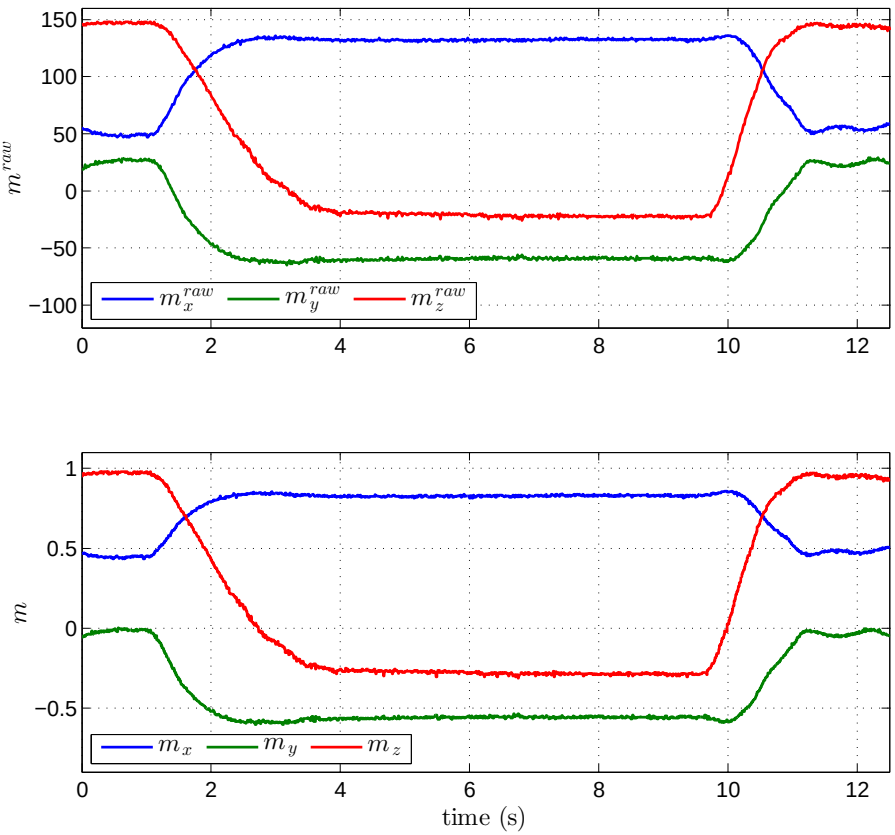

Figure 23: Magnetic (raw and calibrated) measurements.

At this point, the algorithm described in Sections 80-5.3 can be applied to estimate the attitude quaternion. Figure 24 shows the quaternion measure given by the Davenport's q-method, q, Once the measure q has been filtered by the observer, we get $\hat{q}$. q and $\hat{q}$ has been calculated with the previous data and running the MatLab code presented in section 5. It can be seen how the quaternion measure provided by the Davenport's q-method is noisy due to the fact that the inputs a and m are also noisy. However, the observed quaternion, $\hat{q}$, is a clean measure free of noise. Note that $\hat{q} \approx (0.707\ -0.5\ -0.5\ 0)^T$ which is the expected quaternion (106) from the experiment, so the algorithm is validated.

The observer also estimates the gyroscope biases, $\hat{\delta}_\Omega$, which are shown in Figure 25.

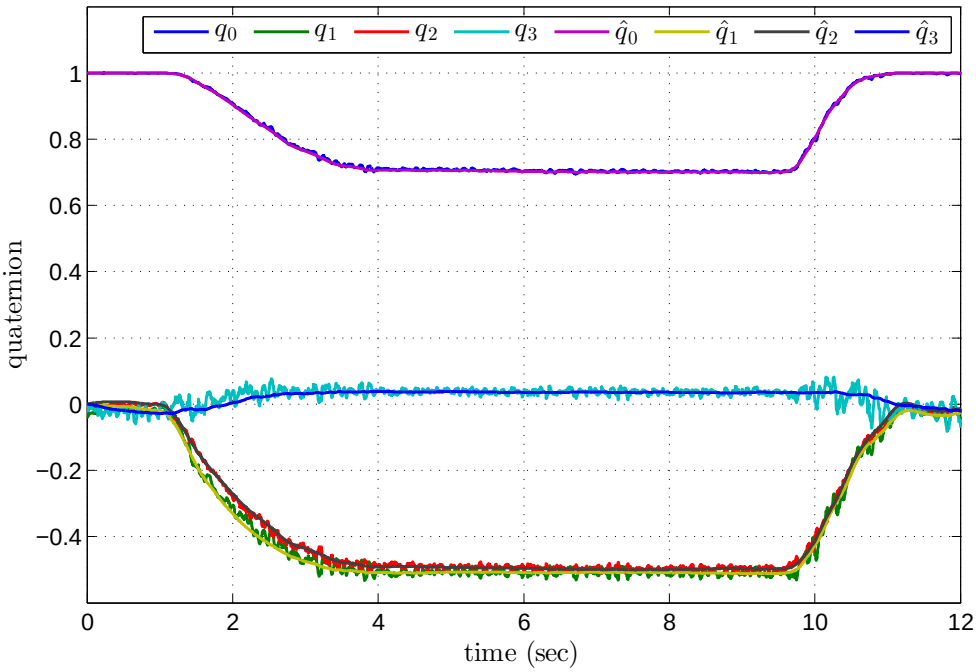

Figure 24: Quaternion measure and observed quaternion.

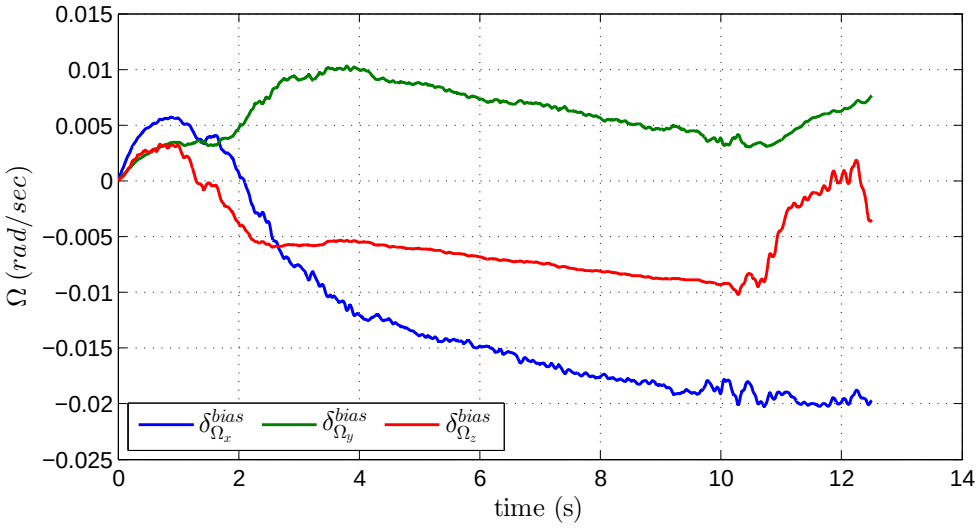

Figure 25: Gyroscope biases estimation.

6 Conclusions

In the present document, the properties of quaternions to manage 3-D space rotations have been analyzed. Quaternions have been demonstrated to be a very powerful tool for that purpose as they avoid the Euler singularity.

Furthermore, the combination between the use of quaternions and the ADRC principles lead to a very simple RPAS flight control which is valid in the full attitude range. As it has been shown, the ADRC philosophy shows that there is no need to have an accurate mathematical model of the aircraft. Also, high external disturbances (like wind gusts) will be rapidly compensated by the ADRC.

In this document it has been also shown that an accurate quaternion measure can be obtained with low-costs sensors. Choosing the appropriate algorithm is crucial for that purpose.

References

[1] Ródenas, L., Sanz, R., Albiol, P., Castillo, A., Verdú, D., & García, P. (2014). Plataforma para la implementación y validación de algoritmos de control de tiempo real en mini-helicópteros de varios rotores. Jornadas de Automática.

[2] Castillo Frasquet, A. (2015). Desarrollo integral de un QuadRotor: Diseño de un algoritmo de control para la posición xy basado en señales GPS.

[3] Albiol Graullera, P. (2015). Desarrollo integral de un QuadRotor: Diseño e implementación de los sistemas de telemetría e interfaz gráfica (HMI) para la monitorización y control mediante entorno Qt.

[4] VerdŤorres, D. (2015). Desarrollo integral de un quadrotor: Control de la orientación basado en una IMU de bajo coste y control de altura mediante un sensor barométrico.

[5] Ruiz Domínguez, F. (2013). La importancia de los RPAS/UAS para la Unión Europea.

[6] Marshall, D. M., Barnhart, R. K., Shappee, E., & Most, M. T. (Eds.). (2015).Introduction to unmanned aircraft systems. Crc Press.

[7] Ec.europa.eu. (2016). Remotely Piloted Aircraft Systems (RPAS) - European Commission. [online] Disponible en: http://ec.europa.eu/growth/sectors/aeronautics/rpas/index_en.html [Accessed 8 Apr. 2016].

[8] Commission, Flying New Way: A boost for European Creativity and Innovation, página 6, 2013. Folleto distribuido por la EC.

[9] A.Castillo, R.Sanz, P.Garcia and P.Albertos. "A Quaternion-based and Active Disturbance Rejection Attitude Control for Quadrotor" 2016 International Conference on Information and Automation (ICIA).

[10] Xia, Yuanqing, and Mengyin Fu. Compound control methodology for flight vehicles. Vol. 438. Berlin, Germany, Heidelberg: Springer, 2013.

[11] Fresk, Emil, and George Nikolakopoulos. "Full quaternion based attitude control for a quadrotor." 2013 European Control Conference (ECC), July. 2013.

[12] Beard, Randal. "Quadrotor Dynamics and Control Rev 0.1." (2008).

[13] Kuipers, J. B. (1999). Quaternions and rotation sequences (Vol. 66). Princeton: Princeton university press.

[14] Jia, Yan-Bin. "Quaternions and rotations." Com S 477 (2008): 577.

[15] Liu, Hao, and Xiafu Wang. "Quaternion-based robust attitude control for quadrotors." Unmanned Aircraft Systems (ICUAS), 2015 International Conference on. IEEE, 2015.

[16] Xia, Yuanqing, and Mengyin Fu. Compound control methodology for flight vehicles. Vol. 438. Berlin, Germany, Heidelberg: Springer, 2013.

[17] Arellano-Muro, Carlos A., et al. "Quaternion-based trajectory tracking robust control for a quadrotor." System of Systems Engineering Conference (SoSE), 2015 10th. IEEE, 2015.

[18] Tayebi, Abdelhamid, and Stephen McGilvray. "Attitude stabilization of a VTOL quadrotor aircraft." IEEE Transactions on control systems technology 14.3 (2006): 562-571.

[19] Tayebi, Abdelhamid. "Unit quaternion-based output feedback for the attitude tracking problem." Automatic Control, IEEE Transactions on 53.6 (2008): 1516-1520.

[20] Guerrero-Castellanos, J. F., et al. "Bounded attitude control of rigid bodies: Real-time experimentation to a quadrotor mini-helicopter." Control Engineering Practice 19.8 (2011): 790-797.

[21] Zhang, Yanjun, and Lu Wang. "Anti-disturbance control methodology for attitude tracking of an UAV." 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2015.

[22] Zhang, Rongting, Quan Quan, and K-Y. Cai. "Attitude control of a quadrotor aircraft subject to a class of time-varying disturbances." Control Theory and Applications, IET 5.9 (2011): 1140-1146.

[23] Han, Jingqing. "From PID to active disturbance rejection control." Industrial Electronics, IEEE transactions on 56.3 (2009): 900-906.

[24] Han, J. "Control theory: Is it a theory of model or control?." Systems Science and Mathematical Sciences 9.4 (1989): 328-335.

[25] Huang, Yi, and Wenchao Xue. "Active disturbance rejection control: methodology and theoretical analysis." ISA transactions 53.4 (2014): 963-976.

[26] Xue, Wenchao, and Yi Huang. "On performance analysis of ADRC for nonlinear uncertain systems with unknown dynamics and discontinuous disturbances." Control Conference (CCC), 2013 32nd Chinese. IEEE, 2013.

[27] Großekatthöfer, K., and Z. Yoon. "Introduction into quaternions for spacecraft attitude representation." TU Berlin (2012).

[28] fit, E. (2016). Ellipsoid fit - File Exchange - MATLAB Central. [online] Mathworks.com. Available at: http://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit [Accessed 24 May 2016].

[29] Wahba, G. (1965). A least squares estimate of satellite attitude. SIAM review, 7(3), 409-409.

[30] Markley, F. L., & Mortari, D. (2000). Quaternion attitude estimation using vector observations. Journal of the Astronautical Sciences, 48(2), 359-380.

[31] Eigen.tuxfamily.org. (2016). Eigen. [online] Available at: http://eigen.tuxfamily.org/ [Accessed 25 May 2016].

[32] Gupta, S. (1998, March). Linear quaternion equations with application to spacecraft attitude propagation. In Aerospace Conference, 1998 IEEE (Vol. 1, pp. 69-76). IEEE.

[33] Vik, B., Shiriaev, A., & Fossen, T. I. (1999). Nonlinear observer design for integration of DGPS and INS. In New Directions in nonlinear observer design (pp. 135-159). Springer London.

[34] Thienel, J., & Sanner, R. M. (2003). A coupled nonlinear spacecraft attitude controller and observer with an unknown constant gyro bias and gyro noise. Automatic Control, IEEE Transactions on, 48(11), 2011-2015.

[35] Hamilton, W. R. (1844). Ii. on quaternions; or on a new system of imaginaries in algebra. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 25(163), 10-13.

[36] Sola, J. (2012). Quaternion kinematics for the error-state KF. Laboratoire d'Analyse et d'Architecture des Systemes-Centre national de la recherche scientifique (LAAS-CNRS), Toulouse, France, Tech. Rep.

Part II
Budget

7 Budget

7.1 Introduction

The present document aims to estimate the total costs associated with the project: *development of a quaternion-based quadrotor control system based on disturbance rejection*. The project has been divided into three stages:

1. Study of the State of the Art. (Code: ***SSA***)
2. Development of the Control Strategy. (Code: ***DCS***)
3. Programming the Quaternion Estimator. (Code: ***PQE***)

In what follows, the costs associated with each stage are going to be calculated. For that purpose, all the developed activities will be identified and they will be listed. Then, all the costs associated with each activity will be estimated.

7.2 Summary of activities

SSA		
Activity	*Code*	*Time (h)*
Study of the state of the art	U1	50
	TOTAL	**50**

DCS		
Activity	*Code*	*Time (h)*
Theoretical Development	U2	80
Simulations	U3	50
Meetings/Travels	U4	60
	TOTAL	**190**

PQE		
Activity	*Code*	*Time (h)*
Testing the quadrotor platform	U5	10
Programing the algorithm	U6	30
Validating the algorithm	U7	20
	TOTAL	**60**

7.3 Costs of labour force

Labour-cost		
Job category	*Code*	*Salary (€/h)*
Industrial Engineer	MII	13
Thecnical Engineer	TE	8

7.4 Costs of equipments

Equipments Costs			
Equipment	*Code*	*Cost (€)*	*units*
Laptop	LPTP	1000	1
MatLab license	MTLB	500	1
Access to IEEE papers	IEEE	2200	1
Quadrotor platform	QDRT	4579,27	1

7.5 Other costs

Other costs			
Equipment	*Code*	*Cost (€)*	*units*
Flight to IEEE ICIA conference	FLGHT	737,15	1
Diets	DTS	80	1
Lodging	LDGNG	250	1
Conference Inscription	INSCR	520	1
Transport	TRNSP	50	1

7.6 Unit costs

U1	Study of the state of the art				
Code	*Ud*	*Description*	*Price (€/Ud)*	*Efficiency (Ud)*	*Cost (€)*
LPTP	units	Laptop	1000	0,1667	166,7
IEEE	units	Access to IEEE papers	2200	0,75	1650
MII	h	Industrial Engineer	13	50	650
				TOTAL	**2466,7**

U2	Theoretical Development				
Code	*Ud*	*Description*	*Price (€/Ud)*	*Efficiency (Ud)*	*Cost (€)*
LPTP	units	Laptop	1000	0,2667	266,67 €
MTLB	units	MatLab license	500	0,6154	307,69 €
IEEE	units	Access to IEEE papers	2200	0,25	550,00 €
MII	h	Industrial Engineer	13	80	1.040,00 €
				TOTAL	**2.164,36 €**

U3	Simulations				
Code	*Ud*	*Description*	*Price (€/Ud)*	*Efficiency (Ud)*	*Cost (€)*
LPTP	units	Laptop	1000	0,1667	166,67 €
MTLB	units	MatLab license	500	0,3846	192,31 €
MII	h	Industrial Engineer	13	50	650,00 €
				TOTAL	**1.008,97 €**

U4	Meetings/Travels				
Code	*Ud*	*Description*	*Price (€/Ud)*	*Efficiency (Ud)*	*Cost (€)*
LPTP	units	Laptop	1000	0,2	200,00 €
MII	h	Industrial Engineer	13	60	780,00 €
FLGHT	units	Flight to IEEE ICIA conference	737,15	1	737,15 €
DTS	units	Diets	80	1	80,00 €
LDGNG	units	Lodging	250	1	250,00 €
INSCR	units	Conference Inscription	520	1	520,00 €
TRNSP	units	Transport	50	1	50,00 €
				TOTAL	**2.617,15 €**

U5	Testing the quadrotor platform				
Code	*Ud*	*Description*	*Price (€/Ud)*	*Efficiency (Ud)*	*Cost (€)*
LPTP	units	Laptop	1000	0,0333	33,33 €
MII	h	Industrial Engineer	13	10	130,00 €
QDRT	units	Quadrotor platform	4579,27	0,1667	763,21 €
				TOTAL	**926,55 €**

U6	Programing the algorithm				
Code	*Ud*	*Description*	*Price (€/Ud)*	*Efficiency (Ud)*	*Cost (€)*
LPTP	units	Laptop	1000	0,1	100,00 €
MII	h	Industrial Engineer	13	30	390,00 €
QDRT	units	Quadrotor platform	4579,27	0,5	2.289,64 €
				TOTAL	**2.779,64 €**

U7	Validating the algorithm				
Code	*Ud*	*Description*	*Price (€/Ud)*	*Efficiency (Ud)*	*Cost (€)*
LPTP	units	Laptop	1000	0,0667	66,67 €
MII	h	Industrial Engineer	13	20	260,00 €
QDRT	units	Quadrotor platform	4579,27	0,3333	1.526,42 €
				TOTAL	**1.853,09 €**

7.7 Total costs

TOTAL COSTS			
Activity	*Description*		*Price (€/Ud)*
U1	Study of the state of the art		*2.466,70 €*
U2	Theoretical Development		*2.164,36 €*
U3	Simulations		*1.008,97 €*
U4	Meetings/Travels		*2.617,15 €*
U5	Testing the quadrotor platform		*926,55 €*
U6	Programing the algorithm		*2.779,64 €*
U7	Validating the algorithm		*1.853,09 €*
		TOTAL	**13.816,46 €**
	Direct costs	0,01	*138,16 €*
	Indirect costs	0,02	*276,33 €*
		TOTAL	**14.230,95 €**