Event-Triggered Attitude Stabilization of a Quadcopter

DIOGO ALMEIDA

KTH Electrical Engineering

Master's Degree Project
Stockholm, Sweden 2014

XR-EE-RT 2014:011

Abstract

There are many possible ways to perform the attitude control of a quadcopter and, recently, the subject of event-triggered control has become relevant in the scientific community. This thesis deals with the analysis and implementation of a saturating attitude controller for a quadcopter system, together with the derivation of an event-triggering rule to work with it. Two distinct rules are presented, one that ensures the stability of the closed loop system, the other, a linearised version that does not. The way those were derived consists in the use of a Lyapunov based approach. The stability of the system when under these rules was verified experimentally.

Keywords: Non-linear control; Attitude stabilization; Event-triggering; Quadcopter

To my mother

Acknowledgements

I would like to thank all the friends that supported me during my study years, that culminated with this report, as well as my family for the constant support and encouragement.

Thanks to Dimos Dimarogonas for the supervision and for the suggestions made during the course of the project.

Special thanks to my friends at the Smart Mobility Lab for all the help in setting up the experiments and for the well spent moments. They are Francisco Martucci, João Pedro and Matteo Vanin.

Contents

Chapter 1

Introduction

In recent years, the topic of event-triggered control has been subject of interest by the part of the research community. Several different triggering strategies have been developed and implemented with encouraging results, allowing for the reduction of the number of computations required for the successful execution of control algorithms. In this thesis, a saturating attitude controller for a quadcopter is studied and implemented in a real system, and a triggering rule to work with it is derived.

1.1 Rigid-body attitude control

The subject of attitude control of a rigid-body is a well-known problem, that consists in sending the right command signals to a rigid-body system to make it adopt a desired orientation, see figure 1.1. It is a tractable problem, with well known kinematics and dynamics equations, with several admissible stabilizing control laws [4] [22].

A quadcopter helicopter can be thought of as a rigid body system, with the same six degrees of freedom and under-actuated dynamics ([15] [10], for example). This allows the application of known rigid-body control techniques to this kind of systems, or to help in the development of new ones.

1.1.1 Quadcopters

The concept of a quadcopter is not new. The first reported working system was developed in 1907 [36], and one of the most successful designs for early helicopter vehicles had precisely a quadcopter-like design [37]. It consists on a set of four arms making a cross shape, with rotors on each end. It is under-actuated, since it has six degrees of freedom (three spatial coordinates plus

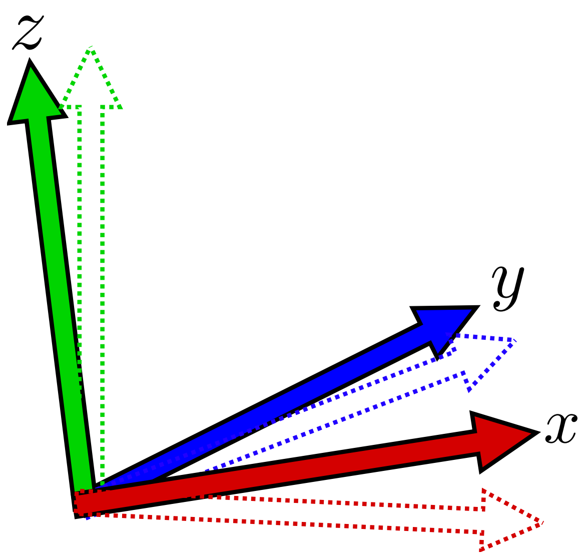

Figure 1.1: The attitude control problem consists in aligning a body frame (filled arrows) with a desired one (dashed)

three rotational angles) and only four actuators.

A quadcopter helicopter is controlled by changes in the rotation speeds of its four rotors. Creating a differential in those speeds allows to tilt the quadcopter frame, or to make it rotate around itself. Combining these movements, it is possible to navigate around three-dimensional environments.

These easy to understand concepts and high flight manoeuvrability capabilities make the quadcopter an interesting platform to study. From the attitude stabilization of this inherently unstable system [12][8] to flight in formation even over complex environments [24]. This became possible over the last decade, with the increasing availability of appropriate inertial measurement units. These, together with inboard control units, allow for attitude control algorithms to run and stabilize the system.

1.2 Event-Triggering Framework

Today's control systems are mostly digital, meaning that the control signal is not evolving continuously in time, but instead it needs to be computed at discrete times. Digital control has been extensively studied over the last century [1][2], but solid results were present only for time-triggered systems, where the control is updated at fixed time intervals. Nevertheless, it is known that event-triggered approaches may outperform time-triggered ones, although the analysis becomes more complex [6][18].

Event-triggered control is a control technique where the control signals are computed only at instants where the plant is considered to need attention. Usually, an event-triggered system has three main components, the controller, an event generator and the network fabric, an abstraction of the network that links the state sensing to the controller. Every time a measure of the plant state fulfils a certain condition, the event generator feeds back the state to the controller, that generates a new control signal. In between these instants, the signal may be kept constant or follow some kind of interpolating function [18].

The main purpose of this framework is to save network resources. This may mean that more bandwidth will be available for other tasks that share the network with the controller or that CPU usage can be minimized, and mostly allows for the development of control systems that operate in a resource aware manner. While the reduction of the number of computations is often the stated goal when developing this kind of systems, they can also shorten the inter-sampling time when the system requires so. This can be an advantage over the classical approach to digital control.

Some results for event-triggering control of dynamical systems have been derived in recent years, where the event-triggering and feedback rule are derived from a Control Lyapunov Function [23][27], while others create an event-triggering rule after defining the controller [25][17].

1.3 Objectives

In this thesis, the Fast and Saturating Attitude controller [28] is discussed, and an event-triggering rule to work with it is derived. Both the controller and the triggering rule are then implemented on a real APM:Copter [33] system, where the stabilizing properties can be verified. The triggering rule is compared with an heuristic approach, both in simulation and practical settings.

1.4 Thesis Outline

This report is organized as follows:

- In chapter 2, the necessary background for the project is given. This includes quaternion based attitude parametrization, a discussion of the proposed saturating controller and an introduction to event-triggering;
- Chapter 3 presents the event-triggering rule derivation, the main contribution of the thesis. Two rules are proposed, one providing analyti-

cal stability guaranties, the other offering a simplification that reduces significantly the computations that need to be carried out in between sampling times. Both are compared with an heuristic error-based rule;

- Chapter 4 describes the implementation of the attitude control system in a real quadcopter, together with an analysis of the results obtained;
- Finally, in chapter 6, conclusions and a discussion for future work are presented.

Chapter 2

Background

For the discussion of the studied controller, a quaternion parametrization of the attitude of a rigid body is used. The basics required to understand that are presented in this chapter, together with the principles of working of the attitude controller. Finally, an introduction to event-triggered control is given. During the remaining of the text, all the norms $\|.\|$ are the L_2 norm of a vector.

2.1 The quaternions

A quaternion is a mathematical object, originally aimed at expanding the set of the complex numbers, $\mathbb{C}$, into three dimensional space. This new system would then add a second imaginary number, j, to an existing complex number, such that $j^2 = -1$. The idea behind this is that from that starting point, the basic properties that make a closed algebraic set would be found, and the existing operations in the complex set could be applied here as well [32].

This proposed approach was not successful, but Hamilton (1805–1865) noted that, by considering a number of the form

$$q = ix + jy + kz + w, \tag{2.1}$$

where

$$\Re(q) = w \;\wedge\; \Im(q) = xi + jy + kz \tag{2.2}$$

are, respectively, the real and the imaginary parts of a quaternion, the extension could be made, albeit not forming a mathematical corpus. The numbers i, j and k obey the following rule:

$$i^2 = j^2 = k^2 = ijk = -1, \tag{2.3}$$

with $w \in \mathbb{R}$. The numbers i, j and k can be thought of as an orthonormal basis of $\mathbb{R}^3$. This makes for the imaginary part of a quaternion to be called the vector part,

$$\mathbf{q}_v = [q_1 \;\; q_2 \;\; q_3] \cdot [i \;\; j \;\; k]^\top , \tag{2.4}$$

with $\{q_1, \; q_2, \; q_3\} \in \mathbb{R}$, and $q_w \equiv w = q_4$. (2.1) can then be written as

$$\mathbf{q} = [\mathbf{q}_v \;\; q_w]^\top = [q_1 \;\; q_2 \;\; q_3 \;\; q_4]^\top , \tag{2.5}$$

with the basis $[i \;\; j \;\; k]^\top$ becoming implicit in the definition of the operations over quaternions.

2.1.1 Operations with quaternions

The basic operations that can be done with quaternions are analogue to the ones found in $\mathbb{C}$, and their derivation can be done by taking the real and vector parts of the quaternion and using them as the real and imaginary parts of a complex number, together with (2.3). We then have

$$\begin{aligned} \mathbf{q} + \mathbf{p} &= [\mathbf{q}_v + \mathbf{p}_v \;\; q_w + p_w]^\top && \text{(addition)} \\ \mathbf{q} \times \mathbf{p} &= \left[\mathbf{q}_v^\top \mathbf{p}_v + \mathbf{q}_v p_w + q_w \mathbf{p}_v + q_w p_w\right]^\top && \text{(Multiplication)} \\ \overline{\mathbf{q}} &= [-\mathbf{q}_v \;\; q_w]^\top && \text{(Conjugate)} \end{aligned} \tag{2.6}$$

where the multiplication operation can be written in matrix form

$$\mathbf{p} \times \mathbf{q} = \underbrace{\begin{bmatrix} q_4 & q_3 & -q_2 & q_1 \\ -q_3 & q_4 & q_1 & q_2 \\ q_2 & -q_1 & q_4 & q_3 \\ -q_1 & -q_2 & -q_3 & q_4 \end{bmatrix}}_{\mathbf{W}(\mathbf{q})} \mathbf{p}. \tag{2.7}$$

2.1.2 Quaternion of rotation

Quaternions can be used to represent the attitude of a rigid body, in which case they are restricted to have unit norm [3]. This is achieved by codifying an axis and an angle of rotation around it, using the vector and scalar parts (2.2). The quaternion (2.5) becomes

$$\mathbf{q} = \left[\sin\left(\frac{\alpha}{2}\right) \mathbf{e} \;\; \cos\left(\frac{\alpha}{2}\right)\right]^\top \tag{2.8}$$

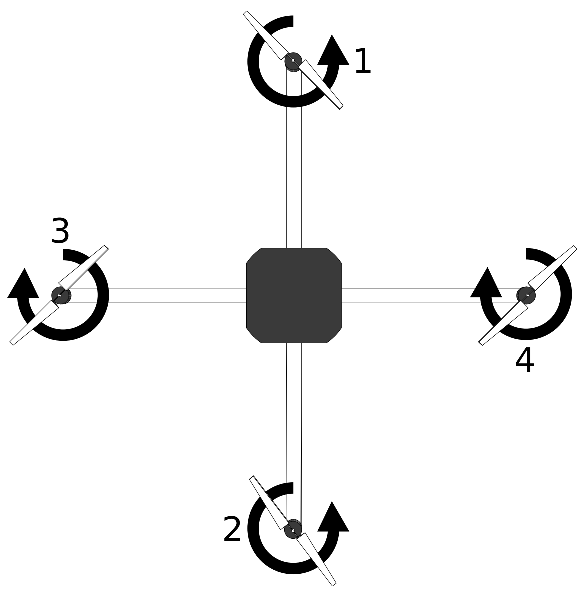

Figure 2.1: Quadcopter top view with the motors direction of rotation identified

where **e** is the axis of rotation. This representation has the advantage of not having singularities (where different attitudes give rise to the same representation). Note that the conjugate operation, $\overline{\mathbf{q}}$, represents the inverse rotation of (2.8), with $-\alpha$ as the angle of rotation. One caveat of this representation is that it is not unique, namely, $\mathbf{q}$ and $-\mathbf{q}$ represent the same orientation (easily verifiable from (2.8)). Still, there are workarounds this that avoid the unwinding effect, a behaviour that occurs when doing attitude control and the controller makes the system rotate 'the other way around' to achieve the equilibrium [22].

2.2 Quadcopter attitude control

A quadcopter is composed of a base frame (two perpendicular arms, forming a cross) with four motors, one at each end. At the center, the electronics to power the motors, the battery and the controller (plus other accessories) are located. Each motor is attached to a propeller and, when rotating, a lift force, the thrust, is produced.

At the same time, the propeller's rotation is counteracted by the air resistance, generating a drag force in the opposite direction of the propeller rotation movement. For this reason, motors along the same axis rotate in the opposite direction of motors in the other one (figure 2.1). This means that,

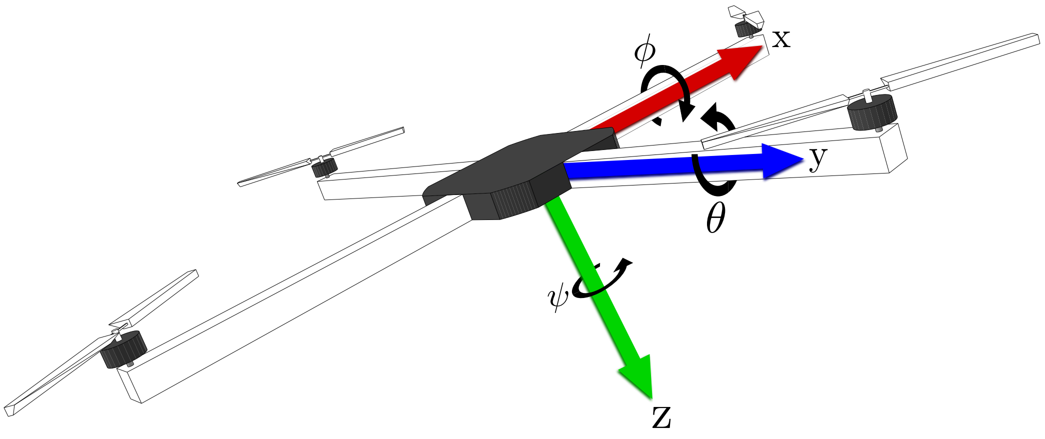

Figure 2.2: Body frame and Euler angles of the quadcopter

by keeping the sum of the rotation speeds of motors in the same axis equal to the sum in the other axis, the drag forces are cancelled and the vehicle will not rotate around itself.

The purpose of attitude control is to regulate the speed of each individual motor so that the quadcopter assumes a desired orientation with respect to a fixed reference frame. A body frame is defined, which has the same orientation as the quadcopter. Usually, it has its x and y axis along the body frame arms, see figure 2.2. The way one regulates the attitude of a quadcopter is by defining a set of three angles (the Euler angles), roll (ϕ), pitch (θ) and yaw (γ). This is convenient, since it is known that a rotation may be decomposed into three successive rotations of these angles, and they can be directly manipulated by adjusting the speed difference of motors along the same axis to change either the roll or the pitch (figure 2.3), and by using the drag forces, the yaw can be changed, see figure 2.4. For this reason, and even though quaternions will be used for representing the attitude, roll, pitch and yaw will be referenced along this report.

2.2.1 Attitude dynamics

To be able to do a controller project, the dynamics of the system need to be modelled. A commonly used model is to assume the quadcopter as a rigid body and model it using the Newton-Euler laws. That way, the only system dependent parameter is the inertia moments matrix, $\mathbf{J}$ [22]. The state variables are the attitude quaternion, $\mathbf{q}$ and the angular velocities along x,

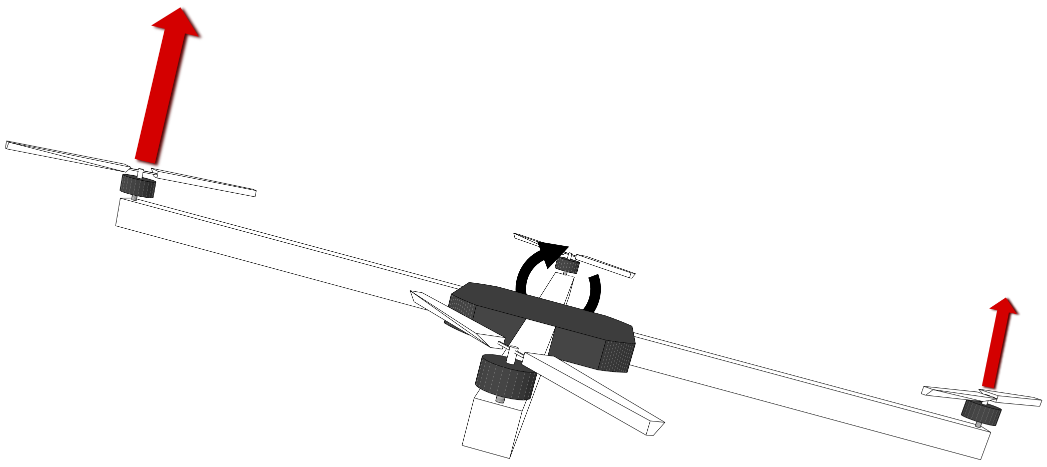

Figure 2.3: By changing the speed of motors along the same axis, roll and pitch can be adjusted

y and z in the body frame, $\boldsymbol{\omega} = [\omega_x \;\; \omega_y \;\; \omega_z]^\top$:

$$\left\{ \begin{array}{rcl} \dot{\mathbf{q}} & = & -\dfrac{1}{2}\mathbf{W_R}(\mathbf{q})\boldsymbol{\omega} \\ \\ \mathbf{J}\dot{\boldsymbol{\omega}} & = & \mathbf{J}\boldsymbol{\omega} \times \boldsymbol{\omega} + \boldsymbol{\tau} \end{array} \right. , \tag{2.9}$$

where $\mathbf{W_R}(\mathbf{q})$ is the same matrix as in (2.7), without the last column. $\boldsymbol{\tau} = [\tau_x \;\; \tau_y \;\; \tau_z]^\top$ are the control torques applied around x, y and z, respectively. They are generated by changing the motors speeds. A difference in the speed of the motors along the x axis will generate a torque along y (affecting the pitch), and a difference in the speed differential on the y axis will generate a torque around x (affecting the roll). The yaw torque comes from the drag forces, and is obtained by changing the difference in speed of motors in different axis. These torques are proportional to the thrust, T_i, generated by each motor, and to the drag forces in the case of the torque around z. Those in turn are proportional to the square of the rotation speeds of each motor, $\overline{\omega}_i$ for $i \in [1 \;\; 2 \;\; 3 \;\; 4]$ (numbered as in figure 2.1), referring to each of the motors [26], or can be modelled as proportional to the PWM signal sent to the motors speed controllers [27]. This later approach is favoured, since in the experimental setup a PWM signal is the final output of the control board.

The total thrust of the quadcopter (that allows it to hover and have a translational movement) is simply the sum of each individual motor thrust.

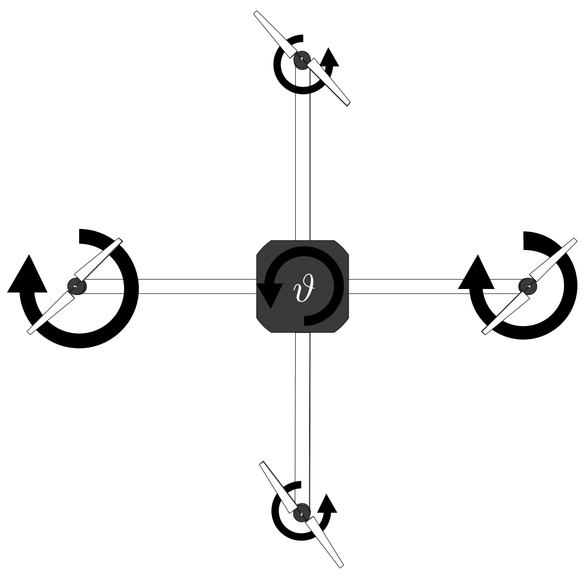

Figure 2.4: By changing the difference between motor speeds on different axis, an angular movement around z is produced

Summarizing this discussion, we have

$$\begin{cases} T & = & c_T \sum_{i=1}^{4} u_i \\ \tau_x & = & dc_T \left(u_3 - u_4\right) \\ \tau_y & = & dc_T \left(u_2 - u_1\right) \\ \tau_z & = & c_D \left(u_3 + u_4 - u_1 - u_2\right) \end{cases}, \tag{2.10}$$

where c_T $(N.s^{-1})$ and c_D $(N.m.s^{-1})$ are coefficients that relate the PWM signal to the generated thrust or Drag forces, respectively, and d is the distance of each motor to the center of mass of the quad, assumed to be the center of the base frame. u_i represents the PWM width in seconds, for the motor i. (2.10) forms a set of linear equations,

$$\begin{bmatrix} T \\ \boldsymbol{\tau} \end{bmatrix} = \underbrace{\begin{bmatrix} c_T & c_T & c_T & c_T \\ 0 & 0 & dc_T & -dc_T \\ dc_T & -dc_T & 0 & 0 \\ -c_D & -c_D & c_D & c_D \end{bmatrix}}_{\Gamma} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}. \tag{2.11}$$

To obtain the PWM values to be sent to the motors, one just needs to invert Γ and multiply it by the Thrust and torques vector.

2.2.2 Control approaches

A common approach to the attitude control of a quadcopter is to continuously measure an error term for each of the Euler angles and feed them to a PID control algorithm [15] [10] [13]. This has advantages in terms of simplicity and its an effective way of stabilizing the attitude, that can be tuned experimentally in a way that has few to none dependences on the system model.

Other controllers have been developed, as a LQ controller [13], and some that take into account the non-linearities of the model, such as PD^2 [15], backstepping [31] [16] or other Lyapunov based approaches [12]. The motivations for each implementation are diverse, but the general idea is to include more information about the system in the design process, so that the performance of the control (in settling time or smoothness, for instance) is better.

2.2.3 Proposed controller

The controller that was studied and implemented in the project was firstly introduced as a thrust direction controller [29], meaning that its only purpose was to align the thrust vector of the quadcopter, see figure 2.5, regulating the yaw rate to be zero at any given time.

The controller was later expanded to cover the full attitude of the system [28].

A fundamental result from non-linear control theory that is heavily used here is the Lyapunov Stability Theorem:

Theorem 1 (Lyapunov Stability)**.** *Let* $\dot{\mathbf{x}} = f(\mathbf{x})$*, with* $f(\mathbf{0}) = \mathbf{0}$*,* $\mathbf{x} \in \mathbb{R}^n$ *and* $0 \in \Omega \in \mathbb{R}^n$*. If there exists a* $\mathbb{C}^1$ *function* $\mathbf{V} : \Omega \to \mathbb{R}$ *such that*

1. $\mathbf{V}(\mathbf{0}) = 0$
2. $\mathbf{V}(\mathbf{x}) > 0$ *for all* $\mathbf{x} \in \Omega$*,* $\mathbf{x} \neq 0$
3. $\dot{\mathbf{V}}(\mathbf{x}) \leq 0$ *for all* $\mathbf{x} \in \Omega$

then $\mathbf{x} = 0$ *is locally stable.*

Proof. Proof can be found in any reference textbook, such as [9]. □

The controller implements an energy shaping technique [7], where the system is viewed as an "energy-transformation device". Here, stability is achieved by forcing the closed loop system to adopt a desired energy function. This allows for the transient behaviour of the system to be considered in the

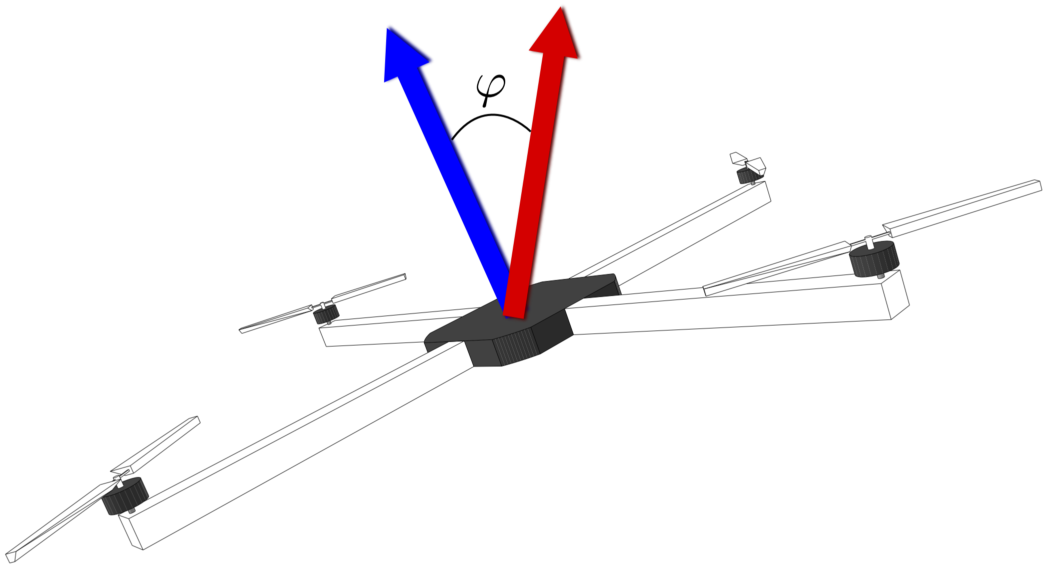

Figure 2.5: Thrust vector displacement

design process, for instance, by adopting as a Lyapunov function the energy of the system, that is null at the desired equilibrium, plus an artificial term, that ensures that the system adopts a desirable behaviour. This is the approach taken by the authors of [29] [28].

The added artificial term allows for problem specific solutions to be included, and in this case, a saturating behaviour was implemented. The controller design process requires the definition of the maximum control torques that will be available for use, favouring their exploitation whenever possible, by saturating the control output. Furthermore, it is given priority to the alignment of the thrust vector, since that is the fundamental operation for successful positioning of the quadcopter. Therefore, the maximum available torques are defined over the xy axis and separately for the z axis

$$\boldsymbol{\tau} = \left[\boldsymbol{\tau}_{xy} \;\; \tau_z\right]^\top \;,\; \boldsymbol{\tau}_{xy} = \left[\tau_x \;\; \tau_y\right]^\top \;. \tag{2.12}$$

The quaternion notation is particularly convenient to exploit these torques, since its angle and axis of rotation interpretation may be directly applied to the notion that we want to control the thrust vector orientation. Firstly, the attitude error is introduced:

$$\mathbf{q} = \overline{\mathbf{q}}_b \times \mathbf{q}_d = \left[q_1 \;\; q_2 \;\; q_3 \;\; q_4\right]^\top , \tag{2.13}$$

where $\mathbf{q}_b$ is the current orientation and $\mathbf{q}_d$ the desired one. The error $\mathbf{q}$ may be decomposed in the sequence of two other displacements, one of the thrust axis, $\mathbf{q}_{xy} = \left[q_x \;\; q_y \;\; 0 \;\; q_p\right]^\top$ and the remaining yaw error, $\mathbf{q}_z = \left[0 \;\; 0 \;\; q_z \;\; q_w\right]^\top$.

A thrust displacement and yaw error angles, represented in figures 2.5 and 2.4, respectively, are defined as

$$\varphi = 2\arccos(q_p)\ ,\ \vartheta = 2\arccos(q_w). \tag{2.14}$$

The control objective is simply:

$$\begin{cases} \mathbf{q} &= [0\ \ 0\ \ 0\ \pm 1]^\top \Leftrightarrow \mathbf{q}_{xy} = \mathbf{q}_z = [0\ \ 0\ \ 0\ \ 1]^\top \Leftrightarrow \varphi = \vartheta = 0 \\ \boldsymbol{\omega} &= [0\ \ 0\ \ 0]^\top \end{cases} . \tag{2.15}$$

The dynamics of the system are determined by (2.9), and after some mathematical manipulation we can obtain the dynamics of the error angles,

$$\dot{\varphi} = -\mathbf{e}_\varphi^\top \boldsymbol{\omega} \tag{2.16}$$

$$\dot{\vartheta} = -\frac{q_z}{\sqrt{1-q_w^2}}\left(\frac{\sqrt{1-q_p^2}}{q_p}\mathbf{e}_\perp^\top + \mathbf{e}_z^\top\right)\cdot\boldsymbol{\omega}. \tag{2.17}$$

This means that a varying ϑ has components around the z axis and also in the orthogonal direction to $\dot{\varphi}$, with the basis vectors and extensive derivation presented in [28].

Energy shaping of the closed loop system

The control law is derived from a specified desired close loop energy function. This consists of the kinetic energy of the system summed to an artificial potential energy, where the problem specific terms will appear. This energy function will be used as the Lyapunov Function of the closed loop system:

$$\mathbf{V}(\mathbf{x}) = E_{\text{rot}}(\boldsymbol{\omega}) + E_{\text{pot}}(\mathbf{q}) = \frac{1}{2}\boldsymbol{\omega}^\top \mathbf{J}\boldsymbol{\omega} + E_{\text{pot}}(\mathbf{q}), \tag{2.18}$$

where $\mathbf{x} = [\mathbf{q}\ \ \boldsymbol{\omega}]^\top$ is the state vector representing the attitude error of the quadcopter. The time derivative of (2.18) is

$$\dot{\mathbf{V}}(\mathbf{x}) = \boldsymbol{\omega}^\top\boldsymbol{\tau} + \frac{\partial E_{\text{pot}}(\mathbf{q})}{\partial \mathbf{q}}\dot{\mathbf{q}} = \boldsymbol{\omega}^\top\boldsymbol{\tau} - \mathbf{T}(\mathbf{q})^\top\boldsymbol{\omega}, \tag{2.19}$$

with

$$\mathbf{T}(\mathbf{q}) = -\frac{1}{2}\frac{\partial E_{\text{pot}}(\mathbf{q})}{\partial \mathbf{q}}\mathbf{W}_{\mathbf{R}}(\mathbf{q}). \tag{2.20}$$

Both (2.19) and (2.20) come from (2.9). By defining the control law

$$\boldsymbol{\tau} = \mathbf{T}(\mathbf{q}) - \mathbf{D}(\mathbf{x})\boldsymbol{\omega}, \tag{2.21}$$

with $\mathbf{D}(\mathbf{x}) \geq 0$ being called a damping matrix, (2.19) becomes

$$\dot{\mathbf{V}}(\mathbf{x}) = -\boldsymbol{\omega}^\top \mathbf{D}(\mathbf{x})\boldsymbol{\omega} \leq 0. \tag{2.22}$$

The term $\mathbf{T}(\mathbf{q})$ may be thought of as an artificial torque field that is directed towards the desired attitude, with the matrix $\mathbf{D}(\mathbf{x})$ ensuring that asymptotical stability is achieved. The definition of these two quantities will determine the closed loop system properties.

Artificial potential energy

The artificial potential energy, $E_{\text{pot}}(\mathbf{q})$ needs to be designed such that it is definite positive with a minimum at the desired equilibrium. Furthermore, its time derivative, $-\mathbf{T}(\mathbf{q})^\top\boldsymbol{\omega}$, should be defined in a way that allows for a good response to deviations from the equilibrium. The saturating controller decomposes $\mathbf{T}(\mathbf{q})$ into four components that act on the two error angles along the appropriate basis vectors:

$$\begin{aligned}
\mathbf{T}(\mathbf{q})^\vartheta_\varphi &= -\cos^3\left(\frac{\varphi}{2}\right)\sin\left(\frac{\varphi}{2}\right)\cdot c_\vartheta \int_0^\vartheta \Lambda^{\vartheta_u}_{\vartheta_l}(\xi)\,\mathrm{d}\xi \cdot \mathbf{e}_\varphi && (2.23)\\
\mathbf{T}(\mathbf{q})^\vartheta_\perp &= \frac{q_z}{\sqrt{1-q_w^2}}\cos^3\left(\frac{\varphi}{2}\right)\sin\left(\frac{\varphi}{2}\right)\cdot c_\vartheta \Lambda^{\vartheta_u}_{\vartheta_l}(\vartheta)\cdot \mathbf{e}_\perp && (2.24)\\
\mathbf{T}(\mathbf{q})^\vartheta_z &= \frac{q_z}{\sqrt{1-q_w^2}}\cos^4\left(\frac{\varphi}{2}\right)\cdot c_\vartheta \Lambda^{\vartheta_u}_{\vartheta_l}(\vartheta)\cdot \mathbf{e}_z && (2.25)
\end{aligned}$$

with the global artificial torques vector being given by the sum of the individual ones,

$$\mathbf{T}(\mathbf{q}) = \mathbf{T}(\mathbf{q})^\varphi_\varphi + \mathbf{T}(\mathbf{q})^\vartheta_\varphi + \mathbf{T}(\mathbf{q})^\vartheta_\perp + \mathbf{T}(\mathbf{q})^\vartheta_z. \tag{2.26}$$

The function $\Lambda^{\xi_u}_{\xi_l}(\xi)$ is a linear function of ξ until $\xi = \xi_l$, remains constant for $\xi \leq \xi_u$ and vanishes to zero when ξ reaches π, see figure 2.6.

Damping matrix

To ensure the stability of the controlled system, a damping term is added to the artificial torques and from that we obtain the control torques to be sent to the quadcopter, (2.21). It suffices for the damping matrix $\mathbf{D}(\mathbf{x})$ to be positive semi-definite for (2.22) to be fulfilled. It is noted in [7], though, that this may not always be good for the performance of the closed loop system.

For instance, if we are far away from the equilibrium and the errors are evolving in its direction, it is undesirable to damp the movement along the

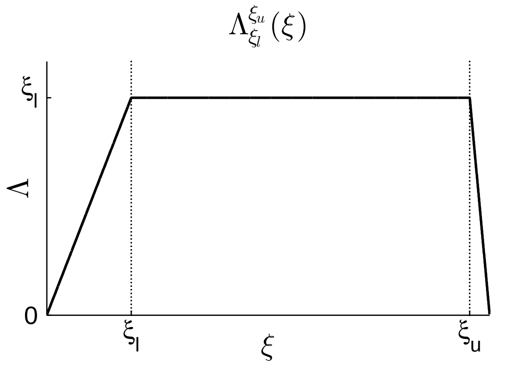

Figure 2.6: The function $\Lambda_{\xi_l}^{\xi_u}(\xi)$. It consists in a regular saturation, together with a vanishing to zero behaviour that prevents singularities in the control.

axis that affect the control variables, since that would make for slower convergence. Moreover, we would like to saturate the control torques in this situation, to ensure faster convergence.

The damping matrix is given by

$$\mathbf{D}(\mathbf{x}) = \kappa_{xy}(\mathbf{x}) \left(d_\varphi(\mathbf{x})\mathbf{e}_\varphi \mathbf{e}_\varphi^\top + d_\perp \mathbf{e}_\perp \mathbf{e}_\perp^\top \right) + \kappa_z(\mathbf{x}) d_z(\mathbf{x}) \mathbf{e}_z \mathbf{e}_z^\top, \tag{2.27}$$

where $\kappa_{xy}(\mathbf{x})$, $\kappa_z(\mathbf{x})$, $d_\varphi(\mathbf{x})$ and $d_z(\mathbf{x})$ are variable gains that ensure $\mathbf{D}(\mathbf{x}) \geq 0$ and that this desirable behaviour is obtained.

2.3 Event-Triggered control

Implementing a control design is something that is easy to do nowadays, with the large availability of inexpensive micro controllers. The possibilities of discrete time control have been explored for a long time, and solid results have been established for systems where the control is updated periodically [1][2].

This classic framework implements what can be thought of as a time-triggered control technique. The control signal is updated at fixed instants t_i separated by a period Δ_t,

$$t_{i+1} = t_i + \Delta_t. \tag{2.28}$$

In between sampling instants, the control signal may be kept constant (zero order hold), vary according to some polynomial (higher order holds) or be set to zero (impulse hold)[18]. When implementing non-linear controllers in a discrete setting, the most common approach is to try to have a sampling period as small as possible, so that the continuous closed loop system is

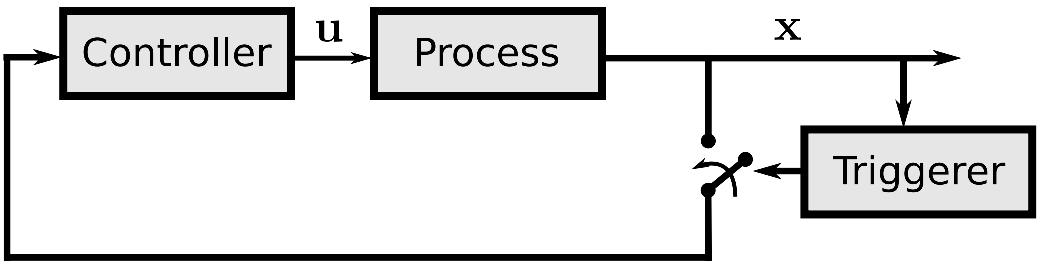

Figure 2.7: An event-triggered feedback loop

emulated. The lower the sampling period, the closer the results are to the desired ones.

The requirement of small sampling periods may not be possible to satisfy, or it can be an undesirable request. The go to example when introducing event-triggered control is a sensor network that measures certain properties of the environment and actuate upon it [17] [19] [6]. If the sensors are periodically broadcasting their measurements over the network, and the controller is updating its output value for every time instant, network bandwidth is being consumed, together with CPU capacity, in what may be a wasteful effort.

A possible solution to this problem has come in the form of event-triggered control. With this technique, the system state is measured and plugged in an evaluation function. This function has the task of indicating if the system requires attention, triggering a sampling instant, or not (figure 2.7).

Event-triggered control assumes that the system is actuated discretely,

$$\dot{x}(t) = f(x(t), u(t_i)), \tag{2.29}$$

with the instants t_i being given not as in (2.28), but as a result of a triggering function output. A possible approach to the problem may be by defining a region B inside which the control is not updated, resorting to periodic sampling outside of it,

$$B = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x} - \mathbf{x}_d\| < \delta\}. \tag{2.30}$$

A common approach to the problem is to define the state error [34] [25] [30],

$$e(t) = x(t) - x(t_i), \tag{2.31}$$

and update the state every time the norm of this error grows above a certain threshold,

$$|e(t)| > \delta_e. \tag{2.32}$$

This has the advantage of only allowing the system to function in open loop over limited regions around the state during the last sampling time. Furthermore, both approaches allow for a trade-off between performance of the control task and number of sampling instants.

Lyapunov based approaches [17][23][21] are also very common. The methods used, as well as the results achieved, are varied, but one goal remains constant: the events must be generated in a way that preserves $\dot{V}(\mathbf{x}) \leq 0$. That way, stability of the system while controlled in an event-triggered manner is ensured.

In a very recent work [27], an event-triggered attitude controller has been introduced for the attitude stabilisation of a quadcopter. The technique used for the design was introduced in [23]. It assumes an asymptotic stabilizing feedback law and the corresponding Control Lyapunov Function (CLF). From the CLF, using a formula derived in the cited work, a triggering function is defined that ensures the stability of the controlled system. Practical work was carried out and experimental results were presented.

While functional, the fact that the control rule is constructed so that the event-triggering formula can be applied does not make problem specific solutions, as presented in section 2.2.3 of this chapter, possible. However, it was the first time that an event-triggered attitude controller got applied to the problem of quadcopter attitude stabilisation.

Chapter 3

Event-Triggering Rule

One of the goals of this project is to implement the saturating controller presented in chapter 2 in a real system, and to develop and implement an event-triggering rule that works with it.

When introducing the event-triggering framework, an event-triggered attitude controller for a quadcopter [27] was briefly discussed. The main novelty of the work presented there is that the control is designed with event-triggering in mind. This is not the case, so a rule needs to be derived specifically for the saturating controller. Ideally, the properties of the controller, namely the thrust axis alignment priority and the saturating behaviour, should be preserved.

The approach taken for the rule derivation is that the Lyapunov Function derivative (2.19) should be kept smaller than zero at all times. This ought to be enough to preserve the stability properties of the controller.

Two rules were derived. The first completely ensures that $\dot{V}(\mathbf{x})$ is never positive. The second one is an approximation that, through a linearisation, manages to depend only on the state error in between sampling times.

3.1 Problem statement

Given the Lyapunov Function (2.18), with time derivative (2.19), we want to find a control update rule such that (2.19) does not become positive. Discrete time is assumed, with a sequence of sampling instants

$$t_1, t_2, t_3, t_4, \dots \tag{3.1}$$

where the control signal $\boldsymbol{\tau}$ is updated:

$$\boldsymbol{\tau}(t) = \boldsymbol{\tau}(t_k) \text{ , for } t \in [t_k, t_{k+1}]. \tag{3.2}$$

During this discussion, the time index will be dropped for continuous variables, and the subscript k will be used to denote the value of some variable at time $t = t_k$.

Finally, the state error is given by

$$\mathbf{e} = \mathbf{x} - \mathbf{x}_k = \begin{bmatrix} \mathbf{q} - \mathbf{q}_k \\ \boldsymbol{\omega} - \boldsymbol{\omega}_k \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{q}} \\ \hat{\boldsymbol{\omega}} \end{bmatrix}. \tag{3.3}$$

With the control (3.2), equation (2.22) becomes

$$\dot{V}(\mathbf{x}) = \boldsymbol{\omega}^\top \boldsymbol{\tau}_k - \mathbf{T}(\mathbf{q})^\top \boldsymbol{\omega}, \tag{3.4}$$

and, bearing in mind that by keeping the control constant,

$$\boldsymbol{\tau}_k = \mathbf{T}(\mathbf{q}_k) - \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k, \tag{3.5}$$

we get

$$\begin{aligned} \dot{V}(\mathbf{x}) &= \boldsymbol{\omega}^\top \boldsymbol{\tau}_k - \mathbf{T}(\mathbf{q})^\top \boldsymbol{\omega} = \\ &= \boldsymbol{\omega}^\top [\mathbf{T}(\mathbf{q}_k) - \mathbf{T}(\mathbf{q})] - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \end{aligned}. \tag{3.6}$$

Notice that the last term is the Lyapunov Derivative of the system evaluated at $t = t_k$, and as such,

$$-\boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k = \dot{V}(\mathbf{x}_k) \leq 0. \tag{3.7}$$

We now want to find triggering rules for the system, using (3.6) as a starting point.

3.2 Rule derivation

The main challenge when evaluating (3.6) is the term $\mathbf{T}(\mathbf{q})$. All the other terms are either kept constant between sampling instants or direct measures of the system state. Ideally, we would like to find an expression that is related to (3.6) and is simpler to compute, since it is desirable that the triggering function can be computed faster than the control law, in order to free resources in between triggering instants.

The trivial solution is to compute (3.6) directly, and generate a triggering instant everytime it gets positive. Since the state is not measured continuously, it may be desirable to be more conservative than that. This can be achieved by making

$$\boldsymbol{\omega}^\top [\mathbf{T}(\mathbf{q}_k) - \mathbf{T}(\mathbf{q})] - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \leq \alpha \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k, \tag{3.8}$$

with α being a constant such that $\alpha \in]0,1[$. The triggering rule then works by monitoring the left-hand side of (3.8) and comparing it to the constant positive semi-definite term on the right-hand side. The smaller the α, the more conservative the rule becomes. This can be adjusted experimentally.

Another way to fulfil the inequality (3.8) would be to upper bound it by a term that is simpler to compute. At the same time, this technique will be necessarily more conservative than (3.8). We first note that, using (2.14),

$$\begin{aligned} \sin\left(\tfrac{\varphi}{2}\right) &= \sqrt{1-q_p^2} \\ \cos^3\left(\tfrac{\varphi}{2}\right) &= q_p^3 \\ \cos^4\left(\tfrac{\varphi}{2}\right) &= q_p^4 \end{aligned} , \tag{3.9}$$

and thus the artificial torques vector becomes

$$\mathbf{T}(\mathbf{q}) = \begin{bmatrix} \left(\dfrac{c_\varphi \Lambda_{\varphi_l}^{\varphi_u}(\varphi)}{\sqrt{1-q_p^2}} - q_p^3 c_\vartheta \displaystyle\int_0^\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\epsilon)d\epsilon\right) q_x + \dfrac{q_z q_p^3 c_\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta) q_y}{\sqrt{1-q_w^2}} \\ \left(\dfrac{c_\varphi \Lambda_{\varphi_l}^{\varphi_u}(\varphi)}{\sqrt{1-q_p^2}} - q_p^3 c_\vartheta \displaystyle\int_0^\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\epsilon)d\epsilon\right) q_y - \dfrac{q_z q_p^3 c_\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta) q_x}{\sqrt{1-q_w^2}} \\ \dfrac{q_z q_p^4 c_\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta)}{\sqrt{1-q_w^2}} \end{bmatrix} . \tag{3.10}$$

We can upper bound (3.6) as

$$\begin{aligned} &\boldsymbol{\omega}^\top \left[\mathbf{T}(\mathbf{q}_k) - \mathbf{T}(\mathbf{q})\right] - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \leq \\ &\leq \boldsymbol{\omega}^\top \mathbf{T}(\mathbf{q}_k) + \|\boldsymbol{\omega}\|\|\mathbf{T}(\mathbf{q})\| - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k, \end{aligned} . \tag{3.11}$$

and $\|\mathbf{T}(\mathbf{q})\|$ can be upper bounded as well, by a simpler term, $\|\mathbf{T}^\star(\mathbf{q})\|$, and then the triggering rule becomes

$$\begin{aligned} &\boldsymbol{\omega}^\top \mathbf{T}(\mathbf{q}_k) + \|\boldsymbol{\omega}\|\,\|\mathbf{T}^\star(\mathbf{q})\| - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \leq 0 \Leftrightarrow \\ &\boldsymbol{\omega}^\top \mathbf{T}(\mathbf{q}_k) + \|\boldsymbol{\omega}\|\,\|\mathbf{T}^\star(\mathbf{q})\| - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \leq \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \end{aligned} . \tag{3.12}$$

Corollary 1. *Given the system* $\dot{\mathbf{x}} = f(\mathbf{x})$ *with*

1. $\mathbf{x} = \begin{bmatrix} \mathbf{q} \\ \boldsymbol{\omega} \end{bmatrix}$
2. $f(\mathbf{x}) = \begin{bmatrix} -\dfrac{1}{2}\mathbf{W}_{\mathbf{R}}(\mathbf{q})\boldsymbol{\omega} \\ \\ \boldsymbol{\omega} \times \boldsymbol{\omega} + \mathbf{J}^{-1}\boldsymbol{\tau} \end{bmatrix}$
3. *Control law* $\boldsymbol{\tau} = \mathbf{T}(\mathbf{q}) - \mathbf{D}(\mathbf{x})\boldsymbol{\omega}$

The triggering rule $\boldsymbol{\omega}^\top \mathbf{T}(\mathbf{q}_k) + \|\boldsymbol{\omega}\|\,\|\mathbf{T}^\star(\mathbf{q})\| - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \leq \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k$ *ensures almost global asymptotic stability of the closed loop system.*

Proof. The inequality $\boldsymbol{\omega}^\top[\mathbf{T}(\mathbf{q}_k) - \mathbf{T}(\mathbf{q})] - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \leq \boldsymbol{\omega}^\top \mathbf{T}(\mathbf{q}_k) + \|\boldsymbol{\omega}\|\,\|\mathbf{T}^\star(\mathbf{q})\| - \hat{\boldsymbol{\omega}}^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k$ is true if $-\boldsymbol{\omega}^\top \mathbf{T}(\mathbf{q}) \leq \|\boldsymbol{\omega}\|\,\|\mathbf{T}^\star(\mathbf{q})\|$. Since, by construction, $\|\mathbf{T}^\star(\mathbf{q})\| \geq \|\mathbf{T}(\mathbf{q})\|$, the triggering rule upper bounds the time derivative of the Lyapunov function of the system, and the stability properties found in [28] are preserved.

□

Ideally, $\mathbf{T}^\star(\mathbf{q})$ should be as simple as possible, while remaining an upper bound to $\mathbf{T}(\mathbf{q})$, but with the drawback that the simpler it becomes, the chances of becoming an overly conservative bound increases. With

$$\mathbf{T}^\star(\mathbf{q}) = \begin{bmatrix} \dfrac{\|q_x\| c_\varphi \Lambda_{\varphi_l}^{\varphi_u}(\varphi)}{\sqrt{1-q_p^2}} + \|q_x q_p^3\| c_\vartheta \vartheta \vartheta_l + \dfrac{\|q_z q_p^3 q_y\| c_\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta)}{\sqrt{1-q_w^2}} \\ \dfrac{\|q_y\| c_\varphi \Lambda_{\varphi_l}^{\varphi_u}(\varphi)}{\sqrt{1-q_p^2}} + \|q_y q_p^3\| c_\vartheta \vartheta \vartheta_l + \dfrac{\|q_z q_p^3 q_x\| c_\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta)}{\sqrt{1-q_w^2}} \\ \dfrac{\|q_z\| q_p^4 c_\vartheta \Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta)}{\sqrt{1-q_w^2}} \end{bmatrix}, \tag{3.13}$$

we get a term that upper bounds $\mathbf{T}(\mathbf{q})$. This follows from having $0 \leq \varphi \leq \pi$, q_x, q_y, q_z, q_p and q_w with absolute value smaller than one and $\Lambda_{\xi_l}^{\xi_u}(\xi) \leq \xi$. A major drawback of this approach is that it still requires extra computations besides the measurement of the state to be carried out. Even considering that there are a lot of repeated terms in (3.13), it is not clear that it offers advantages over the direct computation of (3.10). Making $\mathbf{T}^\star(\mathbf{q})$ simpler by removing the absolute values of the quaternion elements will make the triggering rule to be extremely conservative, since their values will be less than one for most of the time and thus will reduce $\|\mathbf{T}^\star(\mathbf{q})\|$ significantly.

Remark 1. *An alternative approach is to find a first-order approximation to* (3.10) *and use this for the triggering rule. The linearisation of* $\mathbf{T}(\mathbf{q})$ *around* $\mathbf{q}_k$ *is given by*

$$\mathbf{T}(\mathbf{q}) \simeq \mathbf{T}(\mathbf{q}_k) + \nabla\mathbf{T}(\hat{\mathbf{q}}), \tag{3.14}$$

where

$$\nabla\mathbf{T}(\hat{\mathbf{q}}) = \frac{\partial T}{\partial q_x}\cdot \hat{q}_x + \frac{\partial T}{\partial q_y}\cdot \hat{q}_y + \frac{\partial T}{\partial q_z}\cdot \hat{q}_z + \frac{\partial T}{\partial q_p}\cdot \hat{q}_p + \frac{\partial T}{\partial q_w}\cdot \hat{q}_w. \tag{3.15}$$

Inserting (3.14) *into* (3.6), *we get*

$$\dot{V}(\mathbf{x}) \simeq -\boldsymbol{\omega}^\top \nabla\mathbf{T}(\hat{\mathbf{q}}) - \hat{\boldsymbol{\omega}}(t)^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k - \boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k, \tag{3.16}$$

where every 'difficult' term is computed only during triggering instants. $\nabla\mathbf{T}(\hat{\mathbf{q}})$ *is defined in appendix A*

Using this approximation to $\dot{V}(\mathbf{x})$, we can define a new triggering rule:

$$-\boldsymbol{\omega}^\top \nabla\mathbf{T}(\hat{\mathbf{q}}) - \hat{\boldsymbol{\omega}}(t)^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k \le \alpha\boldsymbol{\omega}_k^\top \mathbf{D}(\mathbf{x}_k)\boldsymbol{\omega}_k, \tag{3.17}$$

with $\alpha \in]0,1[$ taking a slightly different role than in (3.8). Make it closer to 1 and the chances that the approximation done to $\mathbf{T}(\mathbf{q})$ is far away from its true value are larger. Make it close to zero and the results may be too conservative to be useful.

To benchmark these rules, a simple error based rule is proposed. Similar to (2.32), we measure the state error (3.3) and update the control whenever it changes more than a certain amount,

$$\|\mathbf{e}\| > \delta_e. \tag{3.18}$$

3.3 Simulation results

To be able to assert the performance of the proposed rules, we propose to compare them to the simulation results presented in [28]. This allows to see if the behaviour of the controlled system remains acceptable. The simulations assume a quadcopter with inertia moments $J_x = 8.5 \times 10^{-3}$ kg.m^2 and $J_z = 14 \times 10^{-3}$ kg.m^2, maximum torques $\overline{\boldsymbol{\tau}}_{xy} = 0.15$ N.m and $\overline{\tau}_z = 0.03$ N.m and the controller parameters are the ones given in table 3.1.

The starting conditions are $\varphi_0 = 170\frac{\pi}{180}$ rad and $\vartheta_0 = 10\frac{pi}{180}$ rad, with zero angular velocities around x and y and $\omega_z = 1.7$ rad/s. Those allow to see the various regions of operation in action, as well as the prioritization of the thrust vector alignment at work.

Parameter	Value
φ_l	$10\pi/180$ rad
ϑ_l	$15\pi/180$ rad
c_φ	0.817 Nm/rad
c_ϑ	0.109 Nm/rad
$\upsilon_{\varphi_{\max}}$	1.425 rad/s
$\upsilon_{\vartheta_{\max}}$	0.624 rad/s
δ_φ	0.1999 Nms/rad
δ_z	0.0961 Nms/rad
$\Delta_\varphi = \Delta_\vartheta$	$5\pi/180$ rad
$\varphi_u = \vartheta_u$	$175\pi/180$ rad
$r_\varphi = r_\vartheta$	0.75
$\upsilon_\varphi = \upsilon_\vartheta$	0.1 rad/s

Table 3.1: Controller parameters used in [28]

The simulation results for the continuous case are depicted in figure 3.1. In the first moments ($t < 0.5$s), the control torques are saturated, allowing for a fast rate of convergence, followed by a switch in the torques signal to make for the decelerating phase, finally ending with the proportional control torques when close to the equilibrium, around $t = 1s$. Furthermore, the compensation of the yaw error only happens once the thrust axis is aligned. The time derivative of the Lyapunov function is, as expected, smaller or equal to zero. A successful triggering strategy should not allow for this derivative to be larger than zero, and ought to preserve the stability properties.

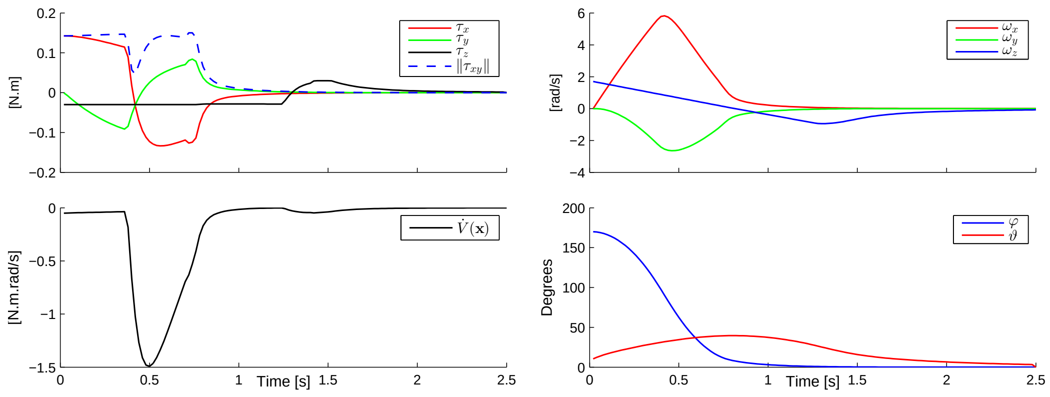

Figure 3.1: Simulation results for the continuous case

3.3.1 Triggering results

To simulate the rules, two set of tests were done. One with a frequency $f = 100\text{Hz}$, the other with $f = 1000\text{Hz}$. The motive for this is to allow for some empirical understanding of how well do the rules scale with the frequency of the state measurements. Besides the results using (3.8), all the depicted results were run at $f = 100\text{Hz}$.

First of all, computing directly (3.8) and enforcing that inequality was discarded completely, since it did not offer any kind of desirable robustness properties. This is due to the fact that, even for extremely low α values, the simulated behaviour showed severe overshooting of the system response (figure 3.2). This simulation was run at $f = 1000\text{Hz}$, since it did not offer results acceptable enough to be compared with the remaining ones. Clearly, it is not enough for $\dot{V}(\mathbf{x})$ to be kept non-positive, since it happens that the switching strategy discussed previously on this report occurs precisely when $\dot{V}(\mathbf{x})$ has its smallest values (as noticed in figures 3.1, 3.3, 3.4 and, by omission, in figure 3.2). For the controller to behave as well as intended, more conservative Lyapunov based rules are needed, or an altogether different approach.

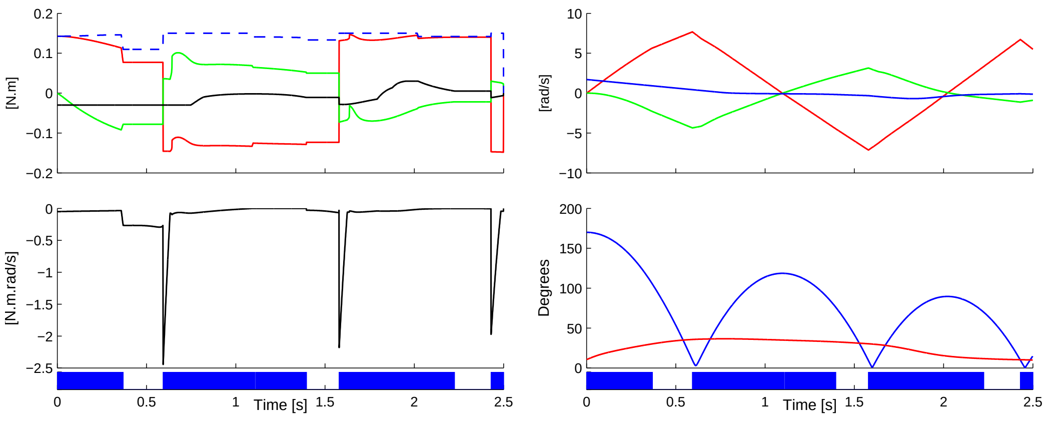

Figure 3.2: Simulation results for the strategy (3.8), with $\alpha = 0$

Rule (3.12) presented the most conservative results, as expected. During great part of the transient the controller was being triggered, with only two discrete sections where no triggering occurred. There were 204 triggering instants for $f = 100\text{Hz}$, and 2019 for $f = 1000\text{Hz}$.

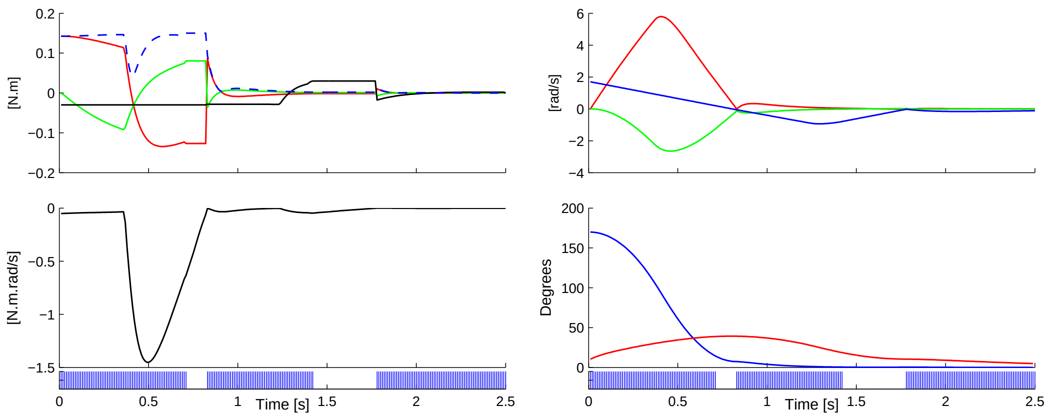

Figure 3.3: Simulation results for rule (3.12)

For the remaining rules, the approach taken when selecting the respective parameters was that the largest value that would not create unwanted behaviour (oscillatory movement or $\dot{V}(\mathbf{x}) > 0$, for instance) would be the one chosen.

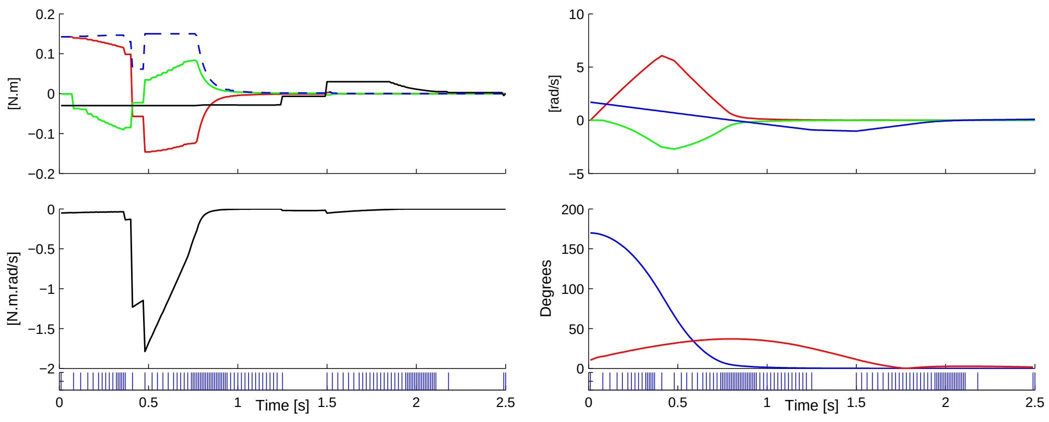

Figure 3.4: Simulation results for rule (3.17) with $\alpha = 0.1$

Rule (3.17) performed as in figure (3.4). It is clearly less conservative than (3.12), which was to be expected. With the chosen parameter $\alpha = 0.1$, $\dot{V}(\mathbf{x})$ never became negative and the simulated results were similar to the continuous case, particularly in settling time and transient behaviour of the error angles. There were 103 triggering instants for $f = 100$Hz, and 152 for $f = 1000$Hz.

Rule	$f = 100$Hz	$f = 1000$Hz
(3.12)	204 (82%)	2019 (81%)
(3.17)	103 (41%)	152 (6%)
(3.18)	99 (40%)	145 (6%)
Baseline	250 (100%)	2500 (100%)

Table 3.2: Number of triggering instants per rule

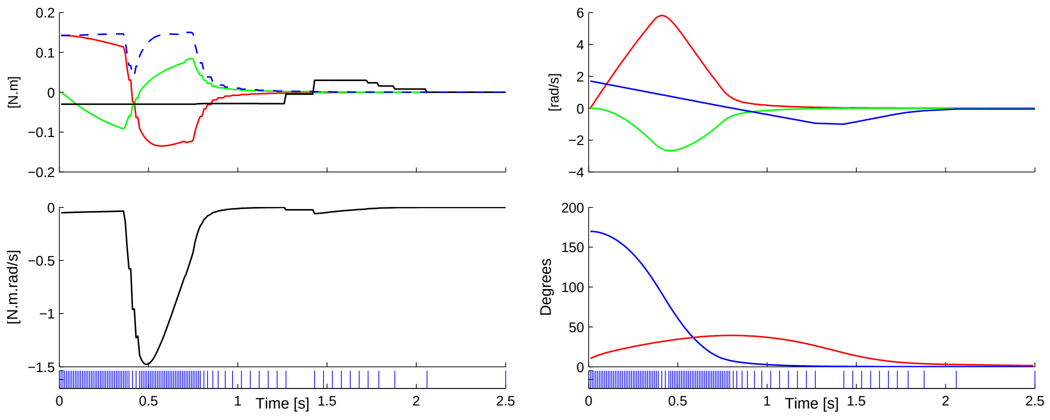

Figure 3.5: Simulation results for rule (3.18) with $\delta_e = 10\%$

Finally, the heuristic rule (3.18) (figure (3.5)) triggers almost every time the state is changing rapidly, while the triggering instants become less frequent as the system comes to the equilibrium. There were 99 triggering instants for the lowest frequency, and 145 for the highest.

The results were summarized in table 3.2. It is clear that rules (3.17) and (3.18) provide the best results, and none of them ensure the stability of the closed loop system. Rule (3.17) was built by linearising $\mathbf{T}(\mathbf{q})$ around the state at $t = t_k$, which encodes some information about the way the system's Lyapunov function is changing, while rule (3.18) simply triggers when the state norm changes more than a specified amount. This makes it much simpler to implement, and provides good simulated results, but it is worth noting that the way the triggering is occurring is fundamentally different for the two rules, as can be seen in figures 3.4 and 3.5.

Chapter 4

Implementation

The saturating controller was implemented in the open-source APM:Copter platform [33] . The main challenges faced were the computational limitations of the on-board controller, together with the PWM-to-thrust relationships that would change quickly as the battery dropped its charge. It was verified that the controller is very sensitive to errors in this relationship, as should be expected, since the definition of a maximum available torque is one of the main features of the controller.

The original controller, together with the three triggering strategies discussed previously, was implemented. This chapter gives some details on the experimental platform, and presents the experimental results.

4.1 The hardware

The controller was implemented in an adapted C++ language called Wiring, since the embedded device that runs the controller is an Arduino-compatible board. This board is based on the Arduino Mega version of the platform, and uses the same ATmega1280 8-bit processor, with a clock running at 16MHz.

Figure 4.1: The controller board, with the inertial estimation board on top

On top of the controller board there is a second board.(figure 4.1) This includes 3 gyroscopes and 3 accelerometers, which allow the estimation of the attitude. There is also the option of adding a 3 axis magnetometer for full 9-degree of freedom coverage.

The system is powered by a 3-cell 2200mAh battery, connected to the board and to four electronic speed controllers (ESCs). These are responsible for regulating the current that is sent to the brushless DC motors, one for each arm. Connected to each DC motor is a 10x4.5cm propeller. The board, following the control algorithm, provides the reference signals sent to the ESCs, in the form of a PWM signal. Finally, communication is done via a pair of Motes, that together with a conversion board emulates the radio signal that the firmware is expecting to receive over the analog input port from the board. This allows for reference signals to be sent from a computer running a positioning control software, for instance.

4.2 Firmware

The APM project provides all the source code for the original firmware in a GitHub repository. It includes many features that were not useful for this project, such as waypoint navigation over GPS and sonar based altitude control, as well as a PID attitude controller and a full Attitude and Heading Reference System (AHRS), that implements a Direction Cosine Matrix algorithm [20] to estimate the attitude from the gyroscopes, using the remaining sensors for drift correction. The included AHRS performed well enough, so it was kept for the practical work. The remaining features were disabled.

The main challenge in implementing the saturating controller was in achieving a satisfactory sampling rate. The maximum frequency the AHRS can run is slightly above 120Hz, which allowed for the original PID attitude controller to run at 100Hz. The saturating controller requires a series of extensive computations, for obtaining both the artificial torques (2.20) or the damping matrix (2.27). Furthermore, when applying the triggering strategy (3.17), the triggering instants imply the computation of $\nabla \mathbf{T}(\hat{\mathbf{q}})$, that is also computationally expensive. To ensure that every cycle fulfils the sampling period, the highest frequency that could be achieved was 50Hz, which is not enough to verify the results in table 3.2.

This frequency was enough, though, to verify that the controller is working and that the triggering rules maintain the stability of the system.

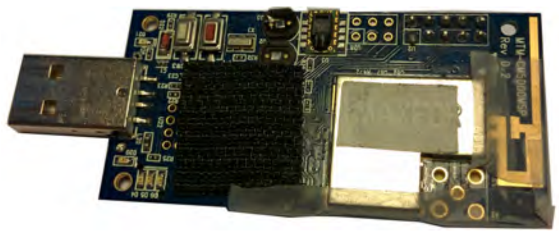

Figure 4.2: The communication mote

4.3 Motor thrust and inertia moments

The saturating controller makes use of the inertia moments of the system when computing the switching instants between saturating and non saturating behaviour, and as such it is required to have an estimate of those values.

An analytical method was used for that purpose, using basic physical concepts [35]. By abstracting the quadcopter components to be shaped as uniformly dense cylinders, parallelepiped and discs (almost flat cylinders), a basic estimate can be attained.

The quadcopter was assumed to be composed of four different elements,

1. Four motors, shaped as cylinders;
2. Four disc shaped propellers;
3. A cross section composed by four parallelepiped arms;
4. A central parallelepiped box, that includes the controller, battery and electronics.

For uniformly densely figures, the inertia moments of each individual element are simple to compute, and by assuming that the quadcopter is symmetric, only the inertia moments around x, y and z need to be computed.

A parallelepiped with mass M, width W align with the y axis, height H and length L along x has moments given by:

$$\begin{aligned} J_x &= M\left(\frac{W^2}{12} + \frac{H^2}{12} + D^2\right) \\ J_y &= M\left(\frac{L^2}{12} + \frac{H^2}{12} + D^2\right) \\ J_z &= M\left(\frac{L^2}{12} + \frac{W}{12}\right) \end{aligned} \tag{4.1}$$

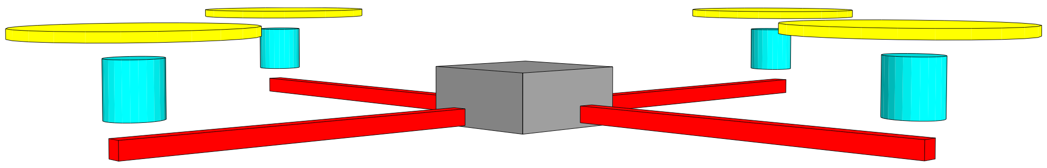
Figure 4.3: Simplified quadcopter model, with the considered geometric figures in different colours

A cylinder with its height H along z, radius R and mass M has the following moments:

$$\begin{aligned} J_x &= M\left(\frac{R^2}{4}+\frac{H^2}{12}\right) \\ J_y &= J_x \\ J_z &= M\frac{R^2}{2} \end{aligned} \tag{4.2}$$

Finally, offsets from the axis of rotation are dealt with by using the parallel axes theorem [35]. A displacement of D from an axis of rotation adds a term to the previous formulas:

$$J_{\text{new}} = J_{\text{center}} + MD^2, \tag{4.3}$$

for each axis of rotation.

The motors have the dimensions $\{M = 0.056\text{Kg}; R = 0.028\text{m}; H = 0.029\text{m}\}$ with displacement $D_x = 0.31\text{m}$ for two of them and $D_y = 0.31\text{m}$ for the other pair, with $D_z = 0.033\text{m}$, the propellers have $\{M = 0.008\text{Kg}; R = 0.125\text{m}; H = 0.01\text{m}\}$ and the same displacement along x and y as the motors, with $D_z = 0.064\text{m}$. The central box has $\{M = 0.486\text{Kg}; L = W0.08\text{m}; H = 0.138\text{m}\}$ and the arms $\{M = 0.015\text{Kg}; W = L = 0.013\text{m}; H = 0.3\text{m}\}$, with no displacement with respect to their axis of rotation (the arms making a cross centred with the quadcopter frame of reference). The inertia moments are given by summing the individual parts:

$$\begin{aligned} J_x &= 2J_{\text{motor}_x} + J_{\text{center}_x} + 2J_{\text{arm}_x} + 2J_{\text{propeller}_x} \\ J_y &= J_x \\ J_z &= 4J_{\text{motor}_z} + J_{\text{center}_z} + 2J_{\text{arm}_z} + 4J_{\text{propeller}_z} \end{aligned} \tag{4.4}$$

The obtained values were $J_x \simeq 0.02$ Kg.m^2 and $J_z \simeq 0.032$ Kg.m^2. This is of course a very coarse estimation, that was not changed over the course of the practical work, since the controller was verified to be robust against

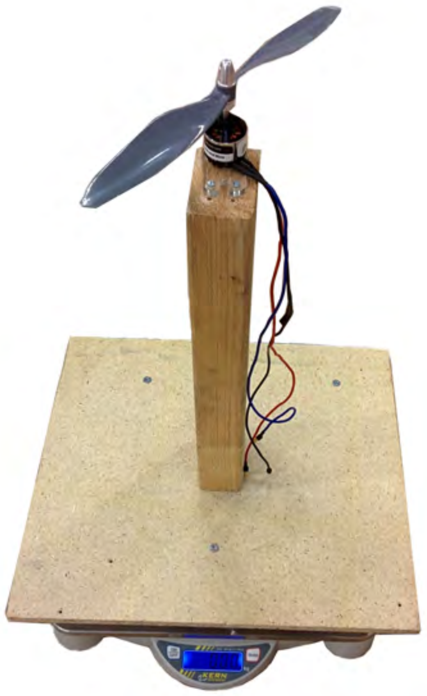

Figure 4.4: Thrust measuring stand

errors in the inertia moments [28], and the obtained results compared well to the simulations.

To get an estimate of the maximum thrust that could be provided by the motors, as well as to get the relationship between PWM and thrust, a simple test stand was built, that attaches a motor to a wooden arm, sitting on top of a scale. When the motor spins, thrust is generated, and the scale shows a difference in measured weight.

Several experiments were carried out, where the command signal was made to vary between the minimum and maximum admissible values. This turned out to produce considerably different results for repeated tests, which can be traced back to changes in the battery level. Even by starting each test with a fully-charged battery, going from the maximum admissible value to the minimum produced significant changes, even with increasing step sizes. In figure 4.5 are depicted the results from three experiments. In the red and black traces, the command signal started at the maximum value and was decreased over the experiment. The blue one depicts an experiment where the command signal started at its minimum and was made to increase. It is clear that there is a drop in maximum thrust in this last experiment. The

Measurement	Regression
Blue	$c_T = 6.665 \times 10^{-4}$ Kg/s
Red	$c_T = 7.0657 \times 10^{-4}$ Kg/s
Black	$c_T = 6.9953 \times 10^{-4}$ Kg/s

Table 4.1: Experimental values for the c_T constant

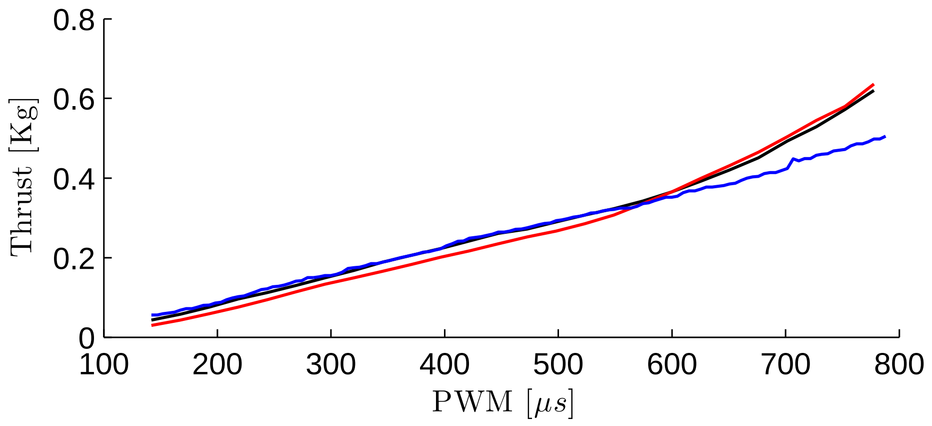

Figure 4.5: Sample of the obtained thrust measurements

battery output current decreased too much and the ESCs were not able to regulate the current that was sent to the motors appropriately.

There is, though, a region that keeps a closer to linear PWM to thrust relationship, and that was chosen to be the operating range of the output command values from the controller. The speed controllers can receive a PWM command signal with width from 1000μs to 2000μs. From the results obtained, and after removing the 1000μs offset, the operating range was defined to be in the $[100, 600]\mu$s range. The constants c_T that were obtained by linear regression on this region are presented in table 4.1. The used value was the lowest one, $c_T = 6.665 \times 10^{-4}$ Kg/s. This is due to the fact that over an experiment, with the battery decreasing, the relationship between PWM and thrust will approach this value. It was deemed more acceptable to risk an underestimation of this constant (that will result in oscillations near the steady state, due to the controller applying larger torques than the computed ones) than an overestimation (that, as the battery decreases, takes control authority from the controller, since the saturating limit prevents this effect to be compensated, as it would with, say, a PID controller).

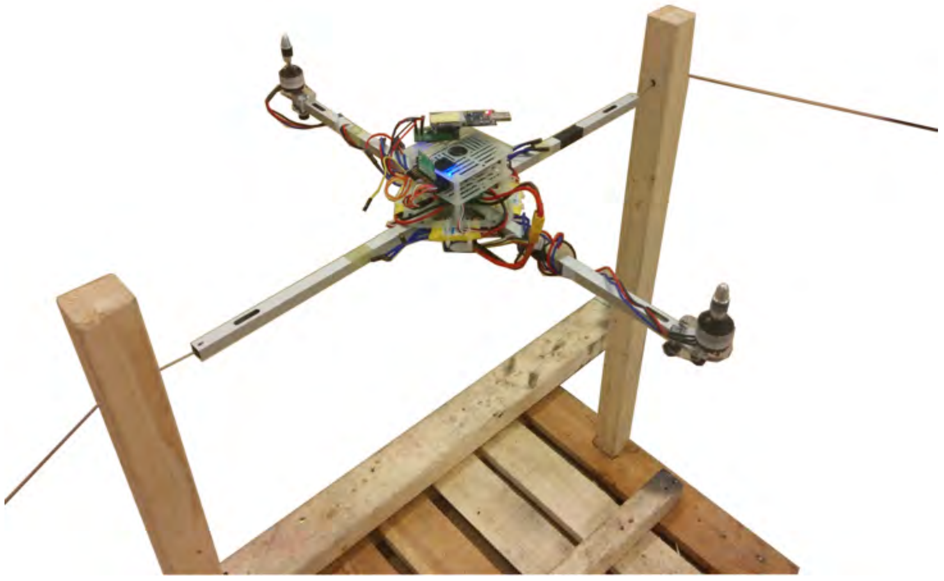
Figure 4.6: One degree of freedom test stand

4.4 Test stand

To properly test the controller, it is fundamental to be able to have a systematic way of experimenting with the different parameters, for repeatability purposes. Clearly, it is not possible to test the attitude controller with the quadcopter being in free space, since the lack of position control would make it easy for the quadcopter to crash, and to explore the controller fully the need to send aggressive reference values is present.

A test stand was built, that allows only one degree of freedom to be available to the quadcopter. It consists of a wooden structure with two wood pillars, see figure 4.6. A metal cable goes through the pillars and passes inside one of the axis of the quadcopter. This allows, by using the two motors on the orthogonal axis, to change the one of the roll or pitch angles. Due to the symmetric nature of the frame, it is irrelevant which one.

Chapter 5

Results

The one degree of freedom stand makes it impossible to test the controller with respect to the yaw. That, together with the absence of propeller drag measurements, resulted in the option of testing the controller for its ability to align the thrust axis, with the drag constant being set as a fifteen times a lower value, based on measurements done in [27]. The maximum torques that were given to the controller were the same as in [28], since that provided good results and increasing this value would make the problems in the thrust relationship estimation more evident. The experimental procedure involves sending a -45 degree reference signal, followed by a 45 degree one, resulting in a 90 degree step reference. The parameters used for the experiments are those in table 5.1. Besides updating the inertia moments to match the real system, the switch curve parameter for φ, r_φ, was changed to 0.5. This allows for smoother transitions between saturated and non-saturated behaviour, which resulted in a less aggressive behaviour of the system. The parameters related to ϑ were omitted, since it is not possible to test for the yaw control on the available stand.

Parameter	Value
φ_l	$10\pi/180$ rad
c_φ	0.817 Nm/rad
$v_{\varphi_{\max}}$	1.425 rad/s
δ_φ	0.1999 Nms/rad
Δ_φ	$5\pi/180$ rad
φ_u	$175\pi/180$ rad
r_φ	0.5
v_φ	0.1 rad/s

Table 5.1: Experimental parameters

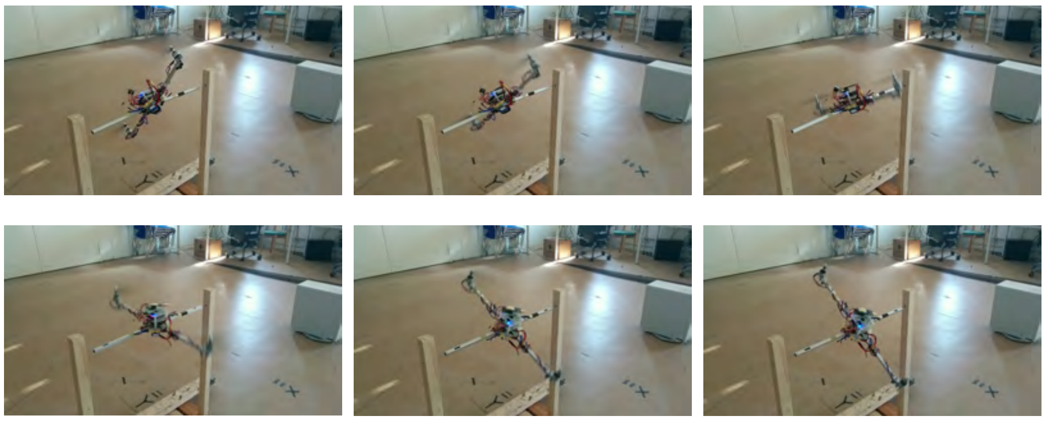
Figure 5.1: Frames from a video depicting the quadcopter responding to a 90 degree step in the reference. All the results present in this report were obtained from similar experiments[1]

5.1 Time-triggered results

The experimental results are depicted in appendix B. There are some important comments to be made, namely, the control torques that are indeed applied to the system achieve the steady state with an offset, and there is some oscillatory behaviour before stabilizing.

The offset can be explained by the fact that lower torques did not affect the system as in the rigid body model, since non modelled effects disturb it. The fact that the thrust constant is not exactly known and changes with the battery level may also lead to sub or overcompensation. This last detail explain the oscillation in the torques. The system ends up applying a larger torque than the expected one, which leads to more abrupt changes in the angular velocity, that needs to be compensated.

The smaller interpolation constant r_φ makes the transition between saturated and non saturated behaviour of the controller more evident, and made for smoother behaviour. Disturbances, non modelled dynamics and the low sampling frequency do not allow for a very high value of this variable, as that would result in abrupt switching, which adds to the errors in the torques estimation for creating an oscillating behaviour. Finally, it can be seen that there is a variable delay in the system reaction for each experiment. This can also be traced to the battery level influencing the thrust output.

[1]The full video can be seen at `http://youtu.be/cMjN4u3IWhE`.

5.2 Event-triggered results

The event-triggered results are affected by the low sampling rate that is achievable in the real system. As seen in chapter 3, it is expected for the linearised and heuristic rules to perform better when the triggering function is monitoring the state at a higher frequency, and it was not possible to verify that. The presented simulation results match the experiments that were done, in the model parameters, initial conditions and reference values. The α and δ_e parameters used for rules (3.17) and (3.18) are the ones that in practise provided the best results, or made for important remarks.

For rule (3.12) (figure B.4), the same kind of triggering pattern seen for the simulation results in figure 3.3 shows up, and a considerable delay is noticeable, compared to the simulation, as well as some oscillatory behaviour. This can be once again traced back to a non-reliable thrust output from the motors.

The Lyapunov function derivative was computed only at triggering instants, meaning that just $\dot{V}(\mathbf{x}_k)$ is plotted for the experimental results. It does not allow to see if $\dot{V}(\mathbf{x}_k) \leq 0$ was violated, but offers some visual interpretation of the effect the triggering is having on the rate of change of the Lyapunov Function (2.18).

The linearised rule underperformed considerably on the experiments, requiring a very conservative value of the constant α to ensure a good behaviour, as can be seen in figures B.6 and B.8. The main challenge resides in stabilizing the angular velocity, task that is made difficult with the combination of errors in the thrust estimation and triggering instants that do not occur at optimal time. This is due to the fact that both rule (3.12) and (3.17) depend on the artificial torques and damping matrices from the saturating controller to ensure stability, and those translate to control torques that are not being applied to the system as supposed. The result is a limit cycle kind of behaviour in the angular speed of the system, with the triggering occurring at instants where the controller is unable to compensate for it.

The heuristic rule (3.18) is model independent in the sense that it makes no assumptions about the system, and does not rely on the saturating controller to be applied. Hence, while the results still differ from the simulation (the controller faces the same challenges as in the time-triggered case), the performance either in terms of triggering instants and overall system behaviour was found to be significantly better, see figure B.10. Note that, since the attitude estimation always carry some noise, the absence of triggering found for the steady state does not translate to the experimental results (figure B.12). The results obtained in terms of triggering instants are summarized in table 5.2, for the first 5 seconds of each experiment.

Rule	Control updates
(3.12)	160 (64%)
(3.17), $\alpha = 0.1$	163 (65%)
(3.17), $\alpha = 0.01$	177 (71%)
(3.18), $\delta_e = 0.1$	69 (28%)
(3.18), $\delta_e = 0.01$	214 (86%)
Baseline	250 (100%)

Table 5.2: Triggering instants results

It is clear that with this experimental conditions, the heuristic rule (3.18) provides the best results. With uncertainties in the applied torques and low sampling frequency, the simulation results from table 3.2 can not be verified, and since the heuristic rule is independent from the system and controller models, it does not suffer from those facts the same way as rules (3.12) and (3.17).

Chapter 6

Conclusions

During this project, the saturating controller proposed in [28] was studied and implemented experimentally for the first time, as far as the author could assert. An event-triggering rule to work with it was derived, and its stabilizing properties were verified experimentally, with a linearised rule being proposed to minimize inter-sampling computations.

Two major factors limited the obtained results, namely, the inability to provide more than 50 Hz for the sampling frequency of the controller, due to the embedded platform used for its implementation, and the fact that a reliable application of the computed torques was not possible to attain.

Without a very good estimation of the fundamental values of the quadcopter model, specially, in the case of this specific controller, the PWM to thrust relationship for the motors, there are no practical advantages in using the proposed controller over a simple PID application. This is due to the PID being able to correct errors in the attitude by changing the torques without bound, while the saturating controller has a specified limit.

Having a better identified system model, and a more powerful embedded platform for the computations, it would be possible to perform a more systematic controller characterization and tuning, by specifying some measure of performance to be achieved (settling time of the thrust vector displacement angle, for instance), and select the parameters that minimize it.

6.1 Future work

The attitude control problem for a quadcopter has been extensively studied before, as exposed in chapter 2. It is nevertheless an interesting proposition to study, implement and benchmark several different controllers on the same, well modelled, quadcopter.

Such comparison was not found, at least not in a systematic, repeatable manner, and it could offer some useful insight on whether it is worthy to keep developing new controllers, besides the obvious academic interest.

The event-triggering attitude control of a quadcopter may be of useful application in a scenario where a multitude of agents are operating towards a common goal. Distributed and autonomous control of a group of agents has been subject of recent research [5] [11] [14] [19], just to name a few. In this framework, there is no central planning of how the different agents will behave to reach a certain objective, and they rely on communication between themselves to coordinate. Having an on board attitude control system that only needs attention on a small percentage of the sampling instants allows for the remaining to be devoted to communication and planning tasks.

Algorithms like the one proposed in [24] allow for impressive group behaviour of quadcopters, but no autonomous group coordination is present.

One challenge to address would be the construction of a distributed algorithm that would enable autonomous group formation of quadcopters, while achieving some sort of performance criteria.

Appendix A

Linearised triggering rule derivation

We need to compute the partial derivatives that composes $\nabla \mathbf{T}(\hat{\mathbf{q}})$ in order to apply (3.17) With:

$$\begin{aligned}
A &= \sqrt{1-q_{p_k}^2} \\
B &= \sqrt{1-q_{w_k}^2} \\
C &= c_\vartheta \int_0^{\vartheta_k} \Lambda_{\vartheta_l}^{\vartheta_u}(\epsilon) d\epsilon \\
D &= \frac{c_\varphi}{A} \\
E &= \frac{c_\vartheta}{B} \\
F &= \arccos(q_{w_k}) \\
G &= \arccos(q_{p_k})
\end{aligned}$$

we have:

$$\left.\frac{\partial T}{\partial q_x}\right|_{t=t_k} = \begin{bmatrix} D\Lambda_{\varphi_l}^{\varphi_u}(\varphi_k) - q_{p_k}^3 C \\ -q_{z_k} q_{p_k}^3 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ 0 \end{bmatrix}$$

$$\left.\frac{\partial T}{\partial q_y}\right|_{t=t_k} = \begin{bmatrix} q_{z_k} q_{p_k}^3 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ D\Lambda_{\varphi_l}^{\varphi_u}(\varphi_k) - q_{p_k}^3 C \\ 0 \end{bmatrix}$$

$$\left.\frac{\partial T}{\partial q_z}\right|_{t=t_k} = \begin{bmatrix} q_{y_k} q_{p_k}^3 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ -q_{x_k} q_{p_k}^3 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ q_{p_k}^4 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \end{bmatrix}.$$

$\left.\frac{\partial T}{\partial q_p}\right|_{t=t_k}$ and $\left.\frac{\partial T}{\partial q_w}\right|_{t=t_k}$ will depend on the value of φ and ϑ, respectively, due to (2.14). Since $\frac{\partial \varphi}{\partial q_p} = -\frac{2}{\sqrt{1-q_p^2}}$, we have, for $0 \leq \varphi \leq \varphi_l$:

$$\left.\frac{\partial T}{\partial q_p}\right|_{t=t_k} = \begin{bmatrix} \left(-\frac{2D}{A} + \frac{2q_{p_k}DG}{A^2} - 3q_{p_k}^2 C\right) q_{x_k} + 3q_{z_k} q_{y_k} q_{p_k}^2 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ \left(-\frac{2D}{A} + \frac{2q_{p_k}DG}{A^2} - 3q_{p_k}^2 C\right) q_{y_k} - 3q_{z_k} q_{x_k} q_{p_k}^2 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ 4q_{z_k} q_{p_k}^3 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \end{bmatrix}$$

for $\varphi_l < \varphi \leq \varphi_u$:

$$\left.\frac{\partial T}{\partial q_p}\right|_{t=t_k} = \begin{bmatrix} \left(\frac{\varphi_l D q_{p_k}}{A^2} - 3q_{p_k}^2 C\right) q_{x_k} + 3q_{z_k} q_{y_k} q_{p_k}^2 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ \left(\frac{\varphi_l D q_{p_k}}{A^2} - 3q_{p_k}^2 C\right) q_{y_k} - 3q_{z_k} q_{x_k} q_{p_k}^2 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ 4q_{z_k} q_{p_k}^3 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \end{bmatrix}$$

and, for $\varphi_u < \varphi \leq \pi$:

$$\left.\frac{\partial T}{\partial q_p}\right|_{t=t_k} = \begin{bmatrix} \left(-\frac{2D\varphi_l}{A\left(\varphi_u-\pi\right)} + \frac{D\varphi_l q_{p_k}\left(2G-\pi\right)}{A^2\left(\varphi_u-\pi\right)} - 3q_{p_k}^2 C\right) q_{x_k} + 3q_{z_k}q_{y_k}q_{p_k}^2 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ \left(-\frac{2D\varphi_l}{A\left(\varphi_u-\pi\right)} + \frac{D\varphi_l q_{p_k}\left(2G-\pi\right)}{A^2\left(\varphi_u-\pi\right)} - 3q_{p_k}^2 C\right) q_{y_k} - 3q_{z_k}q_{x_k}q_{p_k}^2 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \\ 4q_{z_k}q_{p_k}^3 E\Lambda_{\vartheta_l}^{\vartheta_u}(\vartheta_k) \end{bmatrix}$$

For $\vartheta = 2\arccos(q_w)$ we get $\frac{\partial\vartheta}{\partial q_w} = -\frac{2}{\sqrt{1-q_w^2}}$ and $\frac{\partial(\vartheta)^2}{\partial q_w} = -\frac{8\arccos(q_w)}{\sqrt{1-q_w^2}}$. When $0 \leq \vartheta \leq \vartheta_l$, $\int_0^{\vartheta(t)} \Lambda_{\vartheta_l}^{\vartheta_u}(\epsilon)d\epsilon = \frac{\vartheta^2}{2}$ and, as such:

$$\left.\frac{\partial T}{\partial q_w}\right|_{t=t_k} = \begin{bmatrix} 4q_{p_k}^3 q_{x_k} EF + \frac{2q_{w_k}q_{z_k}q_{p_k}^3 q_{y_k} EF}{B^2} - \frac{2q_{z_k}q_{p_k}^3 q_{y_k} E}{B} \\ 4q_{p_k}^3 q_{y_k} EF - \frac{2q_{w_k}q_{z_k}q_{p_k}^3 q_{x_k} EF}{B^2} + \frac{2q_{z_k}q_{p_k}^3 q_{x_k} E}{B} \\ \frac{2q_{z_k}q_{p_k}^4 q_{w_k} EF}{B^2} - \frac{2q_{z_k}q_{p_k}^4 E}{B} \end{bmatrix}$$

When $\vartheta_l < \vartheta \leq \vartheta_u$, $\int_0^{\vartheta(t)} \Lambda_{\vartheta_l}^{\vartheta_u}(\epsilon)d\epsilon = \frac{\vartheta_l^2}{2} + \vartheta_l\left(\vartheta - \vartheta_l\right)$:

$$\left.\frac{\partial T}{\partial q_w}\right|_{t=t_k} = \begin{bmatrix} 2\vartheta_l q_{p_k}^3 q_{x_k} E + \frac{\vartheta_l q_{w_k}q_{z_k}q_{p_k}^3 q_{y_k} E}{B^2} \\ 2\vartheta_l q_{p_k}^3 q_{y_k} E - \frac{\vartheta_l q_{w_k}q_{z_k}q_{p_k}^3 q_{x_k} E}{B^2} \\ \frac{\vartheta_l q_{z_k}q_{p_k}^4 q_{w_k} E}{B^2} \end{bmatrix}$$

Finally, for $\vartheta_u \leq \vartheta \leq \pi$, the integral becomes $\frac{\vartheta_l^2}{2} + \vartheta_l\left(\vartheta_u - \vartheta_l\right) + \frac{\vartheta_l\left(\vartheta^2 - \vartheta_u^2\right)}{2\left(\vartheta_u - \pi\right)} + \frac{\vartheta_l\pi\left(\vartheta_u - \vartheta\right)}{\vartheta_u - \pi}$ and the last partial derivative is

$$\left.\frac{\partial T}{\partial q_w}\right|_{t=t_k} = \begin{bmatrix} -2q_{p_k}^3 q_{x_k}\vartheta_l E\left(\dfrac{\pi-2F}{\vartheta_u-\pi}\right) - \dfrac{2q_{z_k}q_{p_k}^3 q_{y_k}\vartheta_l E}{B\left(\vartheta_u-\pi\right)} + \dfrac{q_{z_k}q_{p_k}^3 q_{w_k} q_{y_k}\vartheta_l\left(2F-\pi\right)E}{B^2\left(\vartheta_u-\pi\right)} \\ -2q_{p_k}^3 q_{y_k}\vartheta_l E\left(\dfrac{\pi-2F}{\vartheta_u-\pi}\right) + \dfrac{2q_{z_k}q_{p_k}^3 q_{x_k}\vartheta_l E}{B\left(\vartheta_u-\pi\right)} - \dfrac{q_{z_k}q_{p_k}^3 q_{w_k} q_{x_k}\vartheta_l\left(2F-\pi\right)E}{B^2\left(\vartheta_u-\pi\right)} \\ -\dfrac{2q_{z_k}q_{p_k}^4 E\vartheta_l}{B\left(\vartheta_u-\pi\right)} - \dfrac{q_{z_k}q_{p_k}^4 q_{w_k} E\vartheta_l\left(\pi-2F\right)}{B^2\left(\vartheta_u-\pi\right)} \end{bmatrix}$$

Appendix B

Experimental results

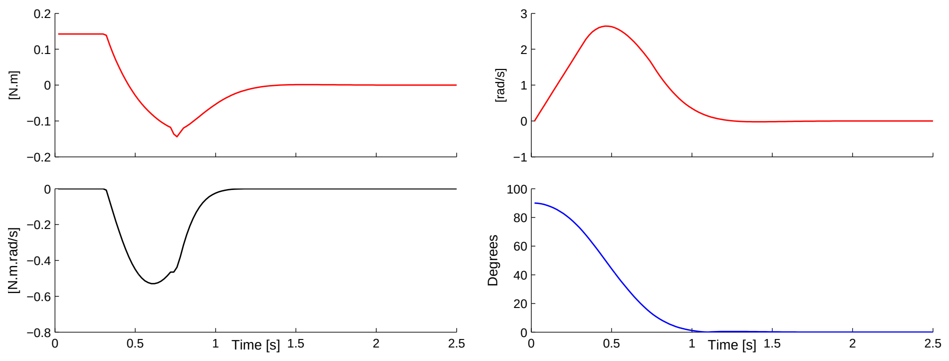

Figure B.1: Simulated results for the single axis experience

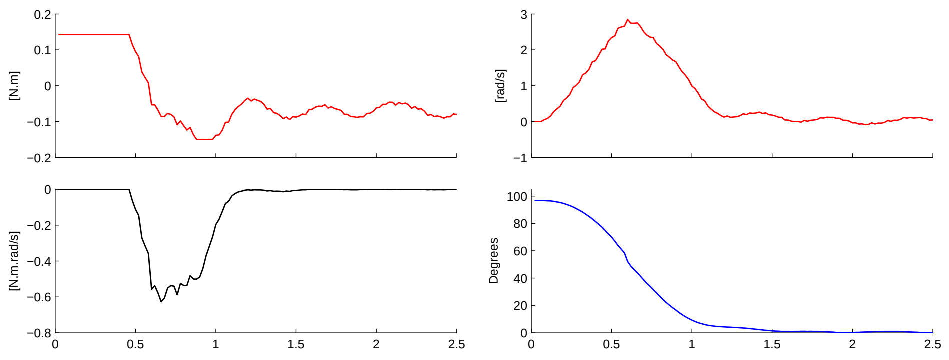

Figure B.2: Experimental results for the single axis experience

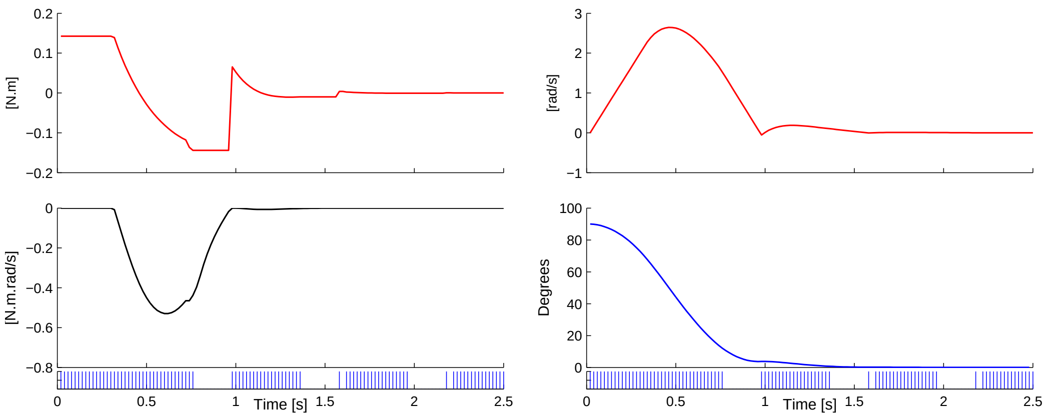

Figure B.3: Simulated results for the experimental setup with rule (3.12)

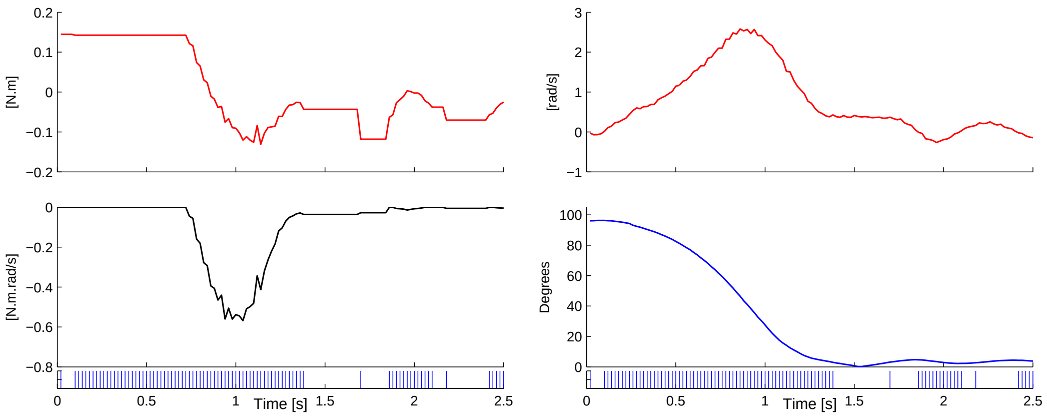

Figure B.4: Experimental results for rule (3.12)

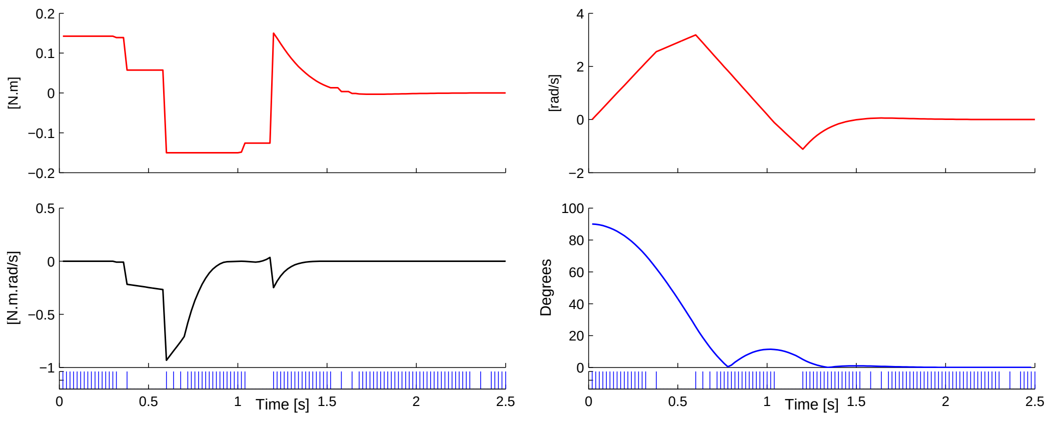

Figure B.5: Simulated results for the linearised rule (3.17). $\alpha = 0.1$

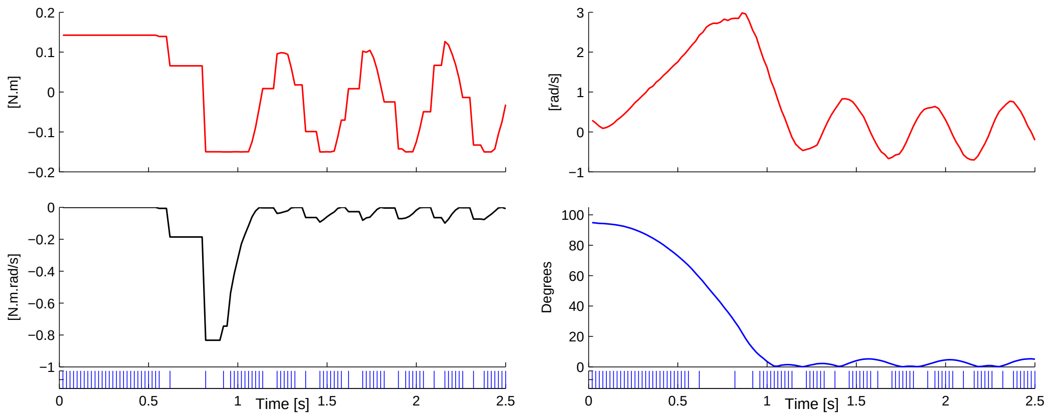

Figure B.6: Experimental results for the linearised rule, $\alpha = 0.1$

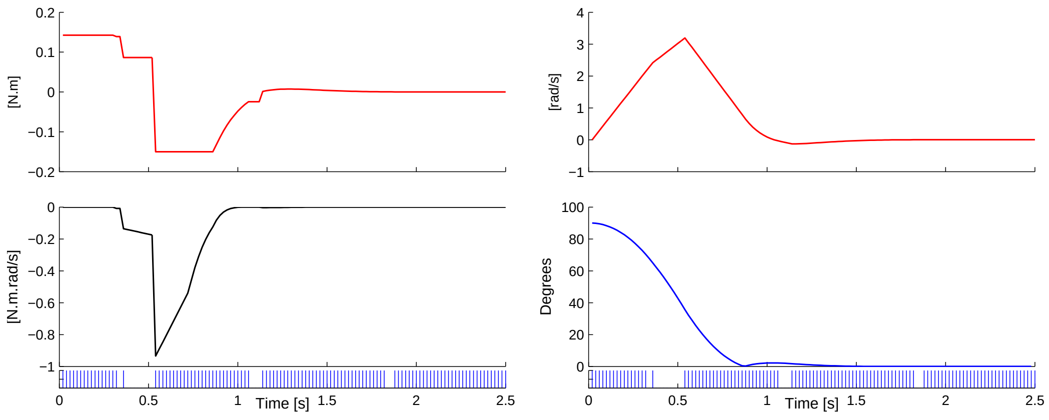

Figure B.7: Simulated results for rule (3.17), $\alpha = 0.01$

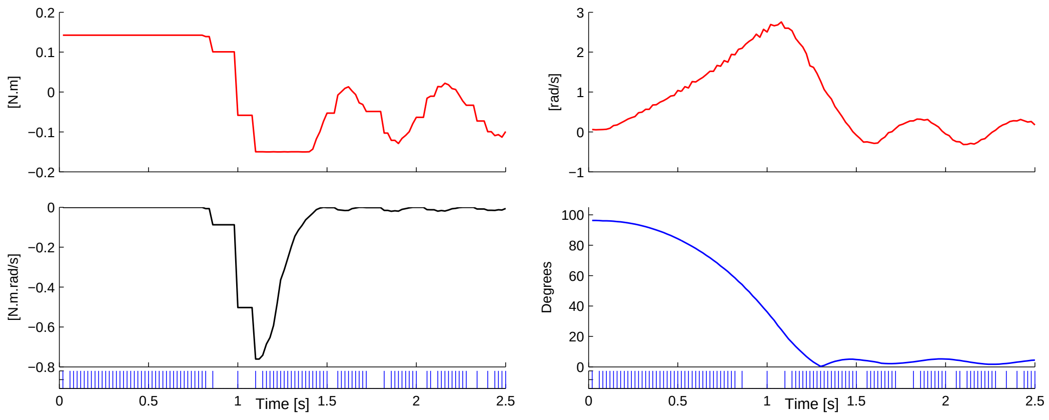

Figure B.8: Experimental results for rule (3.17), $\alpha = 0.01$

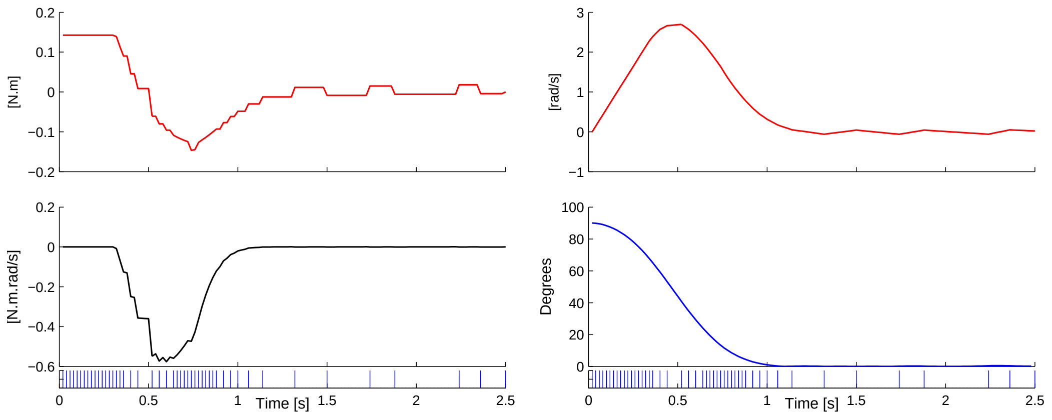

Figure B.9: Simulated results for the heuristic rule (3.18), $\delta_e = 0.1$

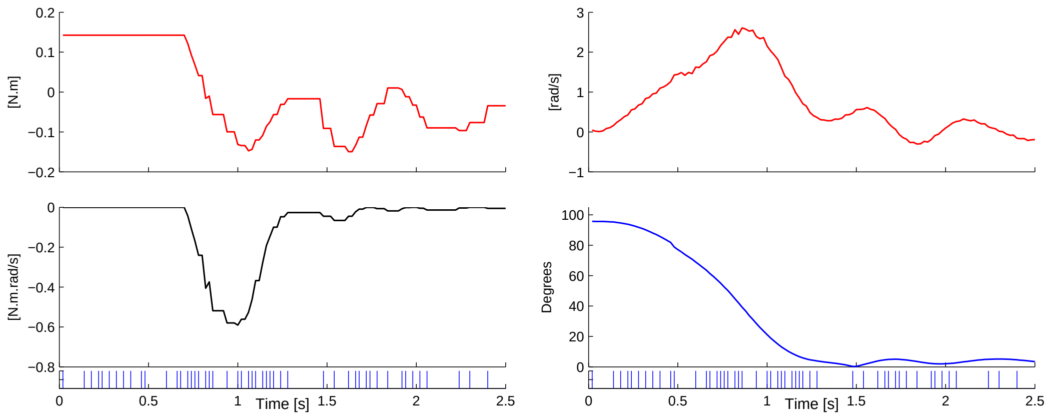

Figure B.10: Experimental results for rule (3.18), $\delta_e = 0.1$

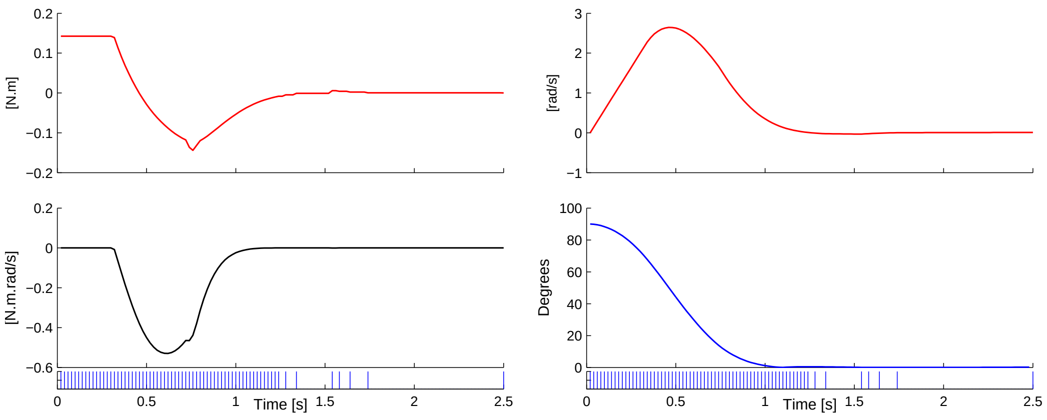

Figure B.11: Simulated results for rule (3.18), $\delta_e = 0.01$

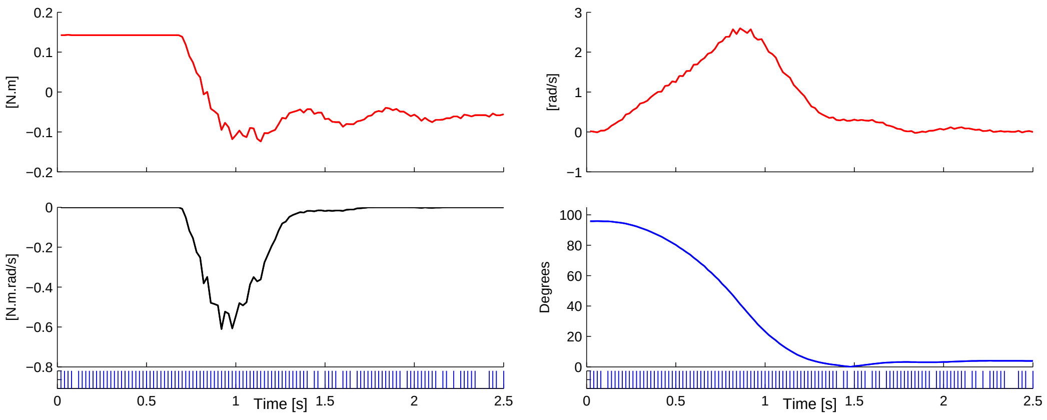

Figure B.12: Experimental results for rule (3.18), $\delta_e = 0.01$

Bibliography

[1] J.R. Ragazzini and G.F. Franklin. *Sampled-data control systems.* McGraw-Hill series in control systems engineering. McGraw-Hill, 1958.

[2] K.J. Åström and B. Wittenmark. *Computer controlled systems: theory and design.* Prentice-Hall information and system sciences series. Prentice-Hall, 1984.

[3] M. D. Shuster. "Survey of attitude representations". In: *Journal of the Astronautical Sciences* 41 (Oct. 1993), pp. 439–517.

[4] Panagiotis T. "New control laws for the attitude stabilization of rigid bodies". In: *13th IFAC Symposium on Automatic Control in Aerospace.* 1994, pp. 316–321.

[5] T. Vicsek et al. "Novel type of phase transition in a system of self-driven particles". In: *Physical review letters* 75.6 (1995), p. 1226.

[6] K. J. Åström and B. Bernhardsson. "Comparison of periodic and event based sampling for firstorder stochastic systems". In: *Proceedings of the 14th IFAC World Congress.* 1999.

[7] R. Ortega et al. "Putting energy back in control". In: *IEEE Control Systems Magazine* (Apr. 2001), pp. 18–33.

[8] T. Hamel et al. *Dynamic Modelling And Configuration Stabilization For An X4-Flyer.* 2002.

[9] H.K. Khalil. *Nonlinear Systems.* Prentice Hall PTR, 2002. ISBN: 9780130673893.

[10] P. Pounds et al. "Design of a Four-Rotor Aerial Robot". In: *Australasian Conference on Robotics and Automation.* 2002, pp. 145–150.

[11] A. Jadbabaie, J. Lin, and A. S. Morse. "Coordination of groups of mobile autonomous agents using nearest neighbor rules". In: *IEEE Transactions on Automatic Control* 48.6 (2003), pp. 988–1001.

[12] S. Bouabdallah, P. Murrieri, and R. Siegwart. “2004-b, “Design and Control of an Indoor Micro Quadrotor”. In: *Proc. of Int. Conf. on Robotics and Automation*. 2004.

[13] S. Bouabdallah, A. Noth, and R. Siegwart. “PID vs LQ control techniques applied to an indoor micro quadrotor”. In: *IEEE International conference on intelligent robots and systems*. 2004, pp. 2451–2456.

[14] W. Ren and R. W. Beard. “Consensus Seeking in Multiagent Systems Under Dynamically Changing Interaction Topologies”. In: *IEEE Transactions on Automatic Control* 50.5 (2005), p. 655.

[15] A. Tayebi and S. McGilvray. “Attitude stabilization of a VTOL quadrotor aircraft”. In: *Control Systems Technology, IEEE Transactions on* 3 (Apr. 2006), pp. 562–571.

[16] P. Adigbli et al. “Nonlinear Attitude and Position Control of a Micro Quadrotor using Sliding Mode and Backstepping Techniques”. In: *European Micro Air VehicleConference and Flight Competition* (2007).

[17] P. Tabuada. “Event-Triggered Real-Time Scheduling of Stabilizing Control Tasks”. In: *IEEE Transactions on Automatic Control* (2007).

[18] Karl J. Aström. “Event Based Control”. In: *Analysis and Design of Nonlinear Control Systems*. Ed. by A. Astolfi and L. Marconi. Springer Berlin Heidelberg, 2008, pp. 127–147.

[19] D. Dimarogonas and E. Frazzoli. “Distributed event-triggered control strategies for multi-agent systems”. In: *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE. 2009, pp. 906–910.

[20] W. Premerlani and P. Bizard. *Direction Cosine Matrix IMU: Theory*. 2009.

[21] A. Eqtami, D. Dimarogonas, and K. Kyriakopoulos. “Event-triggered control for discrete-time systems”. In: *American Control Conference (ACC), 2010*. IEEE. 2010, pp. 4719–4724.

[22] N.A. Chaturvedi, A.K. Sanyal, and N.H. McClamroch. “Rigid-Body Attitude Control”. In: *Control Systems, IEEE* 31.3 (June 2011), pp. 30–51.

[23] N. Marchand, S. Durand, and F. Guerrero-Castellanos. *A general formula for the stabilization of event-based controlled systems*. 2011.

[24] A. Kushleyev, V. Kumar, and D. Mellinger. “Towards A Swarm of Agile Micro Quadrotors”. In: *Proceedings of Robotics: Science and Systems*. Sydney, Australia, July 2012.

[25] D. Lehmann and K. H. Johansson. *Event-triggered PI control subject to actuator saturation.* 2012.

[26] Robert E. Mahony, V. Kumar, and P. Corke. "Multirotor Aerial Vehicles: Modeling, Estimation, and Control of Quadrotor." In: *IEEE Robot. Automat. Mag.* (2012), pp. 20–32.

[27] F. Castellanos et al. "Event-triggered nonlinear control for attitude stabilization of a quadrotor". In: *Journal of Intelligent and Robotic Systems* (2013).

[28] O. Fritsch, B. Henze, and B. Lohmann. "Fast and Saturating Attitude Control for a Quadrotor Helicopter". In: *Control Conference (ECC), 2013 European* (Sept. 2013), pp. 3851 –3857.

[29] Oliver Fritsch, Bernd Henze, Boris Lohmann, et al. "Fast and Saturating Thrust Direction Control for a Quadrotor Helicopter." In: *Automatisierungstechnik, vol. 61, no. 3* (Mar. 2013), 172–182.

[30] WPMH Heemels, MCF Donkers, and Andrew R Teel. "Periodic event-triggered control for linear systems". In: *Automatic Control, IEEE Transactions on* 58.4 (2013), pp. 847–861.

[31] A. Honglei et al. "Backstepping-Based Inverse Optimal Attitude Control of Quadrotor". In: *International Journal of Advanced Robotic Systems* (2013).

[32] C. R. Paiva. *Os Quaterniões de Hamilton.* 2013.

[33] *APM:Copter website.* URL: `http://copter.ardupilot.com/`.

[34] K. E. Årzén. *A simple event-based PID controller.*

[35] T. Bresciani. *Modelling, Identification and Control of a Quadrotor Helicopter.*

[36] G.P. Cyril. *The Bréguet-Richet Quad-Rotor Helicopter of 1907.*

[37] *Oemichen 1922.* URL: `http://www.aviastar.org/helicopters_eng/oemichen.php`.