Autonomous Control of a Miniature Quadrotor Following Fast Trajectories

Master's Thesis, Aalborg University / U.C. Berkeley 2009 - 2010
Department of Electronic Systems

Written by Anders Friis Sørensen

AALBORG UNIVERSITY

Control Engineering
Master's Thesis
Fredrik Bajers Vej 7
Telephone +45 96 35 86 90
http://es.aau.dk

Title:
Autonomous Control of a Miniature Quadrotor Following Fast Trajectories

Project period:
Fall 2009 and spring 2010

Written by:
Anders Friis Sørensen

Group number:
10gr939b

Supervisors:
Asst. Prof. Morten Bisgaard, AAU
Prof. Jakob Stoustrup, AAU
Asst. Prof. Pieter Abbeel, UC Berkeley

Publications: 5

Pages: 78 (98 including appendix)

Handend in: June 3rd, 2010

Abstract:

The purpose of this thesis is to investigate how autonomous robust flight with a miniature quadrotor can be done while minimizing the lag in time of a fast moving reference.
A general introduction to how quadrotors work and how they are controlled are given. A mathematical model is derived for the X-3D quadrotor from Ascending Technology and an Extended Kalman Filter is used with a IMU-driven model to estimate the state doing flight, handle sensor fusion and handle failing sensors.
A LQR using various reference models and a Robust H_∞ controller are derived and tested. Good results was obtained with the LQR, but the H_∞ controller was not able to stabilize the quadrotor in real flight, only in simulation. It is assumed to be caused by the assumption made that the on-board controller is fast enough to ignore any dynamics caused by the controller, motor controller and aerodynamics.
The derived estimator in combination with the LQR, using feed forward of the reference velocity, was able to follow a fast trajectory with a minimum of lag in time.

Preface

This thesis is made as the final step towards my Master's degree. It contains much of the knowledge I have gained in the past two years. Much of my work with quadrotors and rotating frames has initiated in my previous project concerning Autonomous Landing on a Moving Platform. I would therefore like to give a special thank to my previous group consisting of Johannes Friis, Ebbe Nielsen, Jesper Bønding, Anders Jochumsen and Rasmus Foldager for their contributions.
My supervisor Asst. Prof. Morten Bisgaard has been a great support through my entire Master's degree and for that I am very thankful. I would also like to thank my supervisor Prof. Jakob Stoustrup for his inspiration and help with understanding the more advanced control theory. With his help I was allowed to spend one semester at the department of Electrical Engineering and Computer Science at U.C. Berkeley. There, with the help of Asst. Prof. Pieter Abbeel, I gained a more general perspective on control theory and in the field of Reinforcement Learning.
Finally I want to thank Jan Stumpf and Ascending Technology for their help with making the hardware work as intended.

Throughout the thesis the Harvard method has been used to indicate citations. Citations are denoted in square brackets containing the surname of the prime author and the year of publication. All mathematical expressions are described the first time they are used, but a complete list can be found in the nomenclature list in the back of the thesis. All vectors and matrices is denoted with bold font to indicate that they contain multiple dimensions.

Anders Friis Sørensen

Contents

Chapter 1

Introductory

Autonomous controlled aerial vehicles are not a new invention. They were first introduced doing World War I (1917) [Valavanis, 2007, p. 3] as simple and unreliable systems. Throughout the history the Unmanned Aerial Vehicle (UAV) has been an invisible player in many wars, usually used for gathering information, and in a few cases actively in combat. One of the more recent platforms to appear is the quadrotor. The basic idea of the quadrotor has existed since 1907 [Bouabdallah, 2007], but not until recently been flown autonomously. Autonomous flight of the quadrotor has previously been investigated through projects on various universities such as MIT [MIT, 2010], Standford/Berkeley [STARMAC, 2010] and others. Especially more advanced manoeuvres such as obstacle avoidance [STARMAC, 2010] and Simultaneous Localization and Mapping have been investigated using the quadrotor platform. One reason why the quadrotor is good for these applications is that the small size and simple structure makes it easy to obtain very stable flight.

Also outside the academic world is the interest for the quadrotor platform growing. One of the first commercializations of the quadrotor is the Silverlit X-UFO (shown on Figure 1.1), that was sold as a toy from Germany.

Figure 1.1: The Silverlit X-UFO

A few German students visiting MIT used this frame to create a more advanced control board using fast Piezo gyros and Piezo Accelerometers. The students continued this development and in 2003 they started their own company called Ascending Technologies (AscTec). AscTec has since been developing quadrotor frames along with other companies such as Draganflyer and AR Drone.

Recently also the industry has gained an interest in the quadrotor frame, especially when it is combined with autonomous control. Like any other remote controlled flying vehicle a quadrotor is not easy to fly. It takes much time to learn how to fly in confined spaces and not every place is accessible by manual flight, not even with an expert pilot.
Within the past year the technology has allowed microcomputers to become fast enough and sensors small enough that a full autonomous controlled quadrotor can be made to fit in a lunch box and obtain more than half an hours of flight time. New companies such as Sky-Watch (DK) and Microkopter (DE) is aiming directly towards the industrial market. Currently the main target is companies who have the need for inspections in places that is not easily accessed. This could be in areas where it is expensive or dangerous to send up people. Quadrotors also serves as an easy access to live overhead footage or high quality still pictures.

Control of a quadrotor is typically done using either simple hand-tuned feedback compensators or a more advanced model based feedback compensators. Throughout this thesis advantages for both types of control will be discussed.

1.1 Focus of the thesis

The main focus of this thesis can be described as the solution to the following question:

Can robust flight with a miniature quadrotor be done in such a way that a fast moving reference can be followed with minimum lag in time.

This question give rise to several challenges. Robust flight is defined such that the quadrotor is to remain stabilized even when exposed to disturbances in the form of a reference and sensor noise. Robust flight, in this thesis, also includes handling failing sensors and continuing flight with the limited sensibility.
A fast moving reference is defined to be a reference that is moving with a velocity, that would cause a normal feedback controller to lag significantly in time. The challenge with regard to the fast moving reference is to minimize this lag without compromising the steady-state performance of a constant reference.

To obtain robust flight with a minimum lag in time two types of controllers is developed (part III). A robust H_∞ controller and a Linear Quadratic Regulator (LQR) using a reference model. To minimize the time lag feed forward of the velocity of the reference will be used.
An Extended Kalman Filter (EKF) is developed to both minimize the sensor noise and to handle failing sensors (part II). To minimize computation time and to suppress modelling errors the EKF will be based on measurements from the Inertial Measurement Unit (IMU).
Both the controllers and the estimator is based upon knowledge of the dynamics of the system. A general understanding and a mathematical model is therefore being derived to improve the performance of both controllers and estimator (part I).

1.2 The X-3D Quadrotor Test Environment

The quadrotor chosen for experiments is manufactured by the German company AscTec [Gurdan et al., 2007]. The company supplies various models with different versions of firmware. The version used in this project is installed with the ResearchPilot firmware which enables serial communication with the quadrotor. In this way it is possible to read out sensor measurements and send commands over a digital interface. If a pair of X-bee

communication devices are added this can be done over a wireless data link.

In this setup, all sensors are connected to a control PC on which is installed Matlab/Simulink. Matlab/Simulink is used to implement the estimation of states and calculation of the control commands for the quadrotor. An overview of the setup is shown in Figure 1.2.

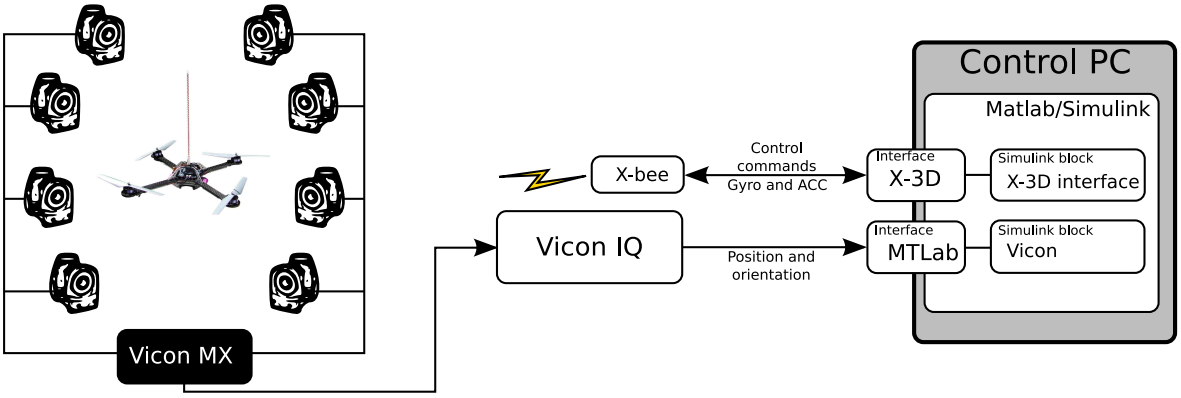

Figure 1.2: Overview of the MTLab

1.2.1 Control Commands

Through the serial communication connector on the X-3D it is possible to communicate flight commands to the quadrotor using the proper protocol [AscTec, 2009]. The flight commands are given as values which are otherwise send from the RC-transmitter for roll, pitch, yaw and thrust. The definition of roll, pitch and yaw commands is explained in more detail in a later chapter. Each value is, according to the protocol, defined as in Table 1.1. In this setup positive roll is defined as a positive rotation around the x-axis and so forth.

Command	Value
roll	-2048 to 2047
pitch	-2048 to 2047
thrust	0 to 4096
yaw	-2048 to 2047

Table 1.1: Commands send to quadrotor

The quadrotor can be controlled by the commands send, but only as long as a command is received at least once every 100ms. If this is not the case the control is given back to RC-transmitter.

Ascending Technologies supplies support for the protocol on the quadrotor, but no public available implementation for the PC exists. In order to establish communication an interface from Matlab/Simulink to the quadrotor has therefore been developed. This interface enables configuration of which sensors to be read, and loading them directly into Simulink. The interface also supplies an easy way to configure which channels that should be autonomously controlled, and which should be controlled manually. This feature is very helpful when developing controllers to a single channel. Finally it also enables the possibility to send commands to the quadrotor directly form Simulink. Additional information on the communication and the development of the interface can be found in Appendix A.

1.2.2 On-board Sensors

Multiple sensors can be read from the quadrotor both in raw and in processed values. An entire list can be found in the manual for the ResearchPilot firmware [AscTec, 2009]. For the estimator developed in this thesis only the raw gyro and acceleration measurements will be used. These will be measured with an update frequency of 100 Hz.

1.2.3 The Motion Tracking Lab (MTLab)

For measurements of the position and orientation the MTLab is used. The MTLab is a room at Aalborg University that is equipped with 7 cameras recording positions of small reflectors mounted on various objects. The reflectors can be seen as the silver balls mounted on the X-3D quadrotor in Figure 1.3.

Figure 1.3: The X-3D quadrotor

The MTLab measures the position in meters with an accuracy better then 1 mm and the orientation, given as either a 3-2-1 Euler rotation or a quaternion. More on the different orientations will be discussed in the following chapter. Part of the development of both controllers and estimator have been conducted at UC Berkeley in California. At UC Berkeley a similar system, called PhaseSpace, was used. The PhaseSpace system uses active markers but is otherwise similar to the Vicon MX system and will not be discussed any further in this thesis.

A more in-depth description of the MTLab can be found in Appendix B.

Part I

Building a Model

Table of Contents

Chapter 2

Modelling: An Introduction

Any mathematical kind of model is simply an approximation of the real world, describing any relevant influence of some input signal on an output signal. In this case the model will describe how the quadrotor moves in a 6 dimensional space depending on the four control commands mentioned in the previous chapter. Throughout the chapters a number of assumptions will be made. This will be noted and discussed to the extent it is found relevant.

2.1 Structure of the Modelling Chapters

This introduction serves to guide the reader and to give an overview of the modelling part. Before the actual modelling of the quadrotor initiates, are some general concepts such as the coordinate systems defined.

The first modelling chapter covers the basics a quadrotor, how is it controlled and the effect of the control commands described from a pilots perspective. Throughout the chapter various details of the X-3D quadrotor are included, in particular the on-board PD-controller that closes the angular rate loop and attenuates some aerodynamic effects. The chapter concludes by defining a continuous-time model. Later the model is linearized and put on state-space form.

In the chapter concerning the sensor modelling, a stochastic model is derived of the sensors available. The sensors in focus are the accelerometers, gyroscopes and Vicon measurements. Together they form the basis for estimating the states of the quadrotor.

2.2 Rotating and Fixed Frames

When dealing with flying objects it is convenient to be able to describe vectors not only in a global frame, but also in a rotating local body frame. Any experiments made in this thesis are done in one specific room. This room is defined to be the inertial frame from where the model will be described. The rotation of the earth is assumed not to affect the flight of the quadrotor. The earth fixed inertial frame is from now referred to as the earth frame. The rotating frame following the attitude of the quadrotor is denoted the body frame. For this assumption to make sense the structure of the quadrotor is assumed to be rigid. That a body is rigid is defined

such that the distance between any two points in the structure will always remain the same no matter how the body is positioned or oriented.

Figure 2.1 shows the two frames. To the left the earth fixed frame denoted E and to the right the body fixed frame denoted B. When a vector is seen with respect to the earth fixed frame it will either be denoted with an ^{e} in front of the vector or nothing at all. Likewise if the vector is seen with respect to the body frame it will be denoted with a ^{b}.

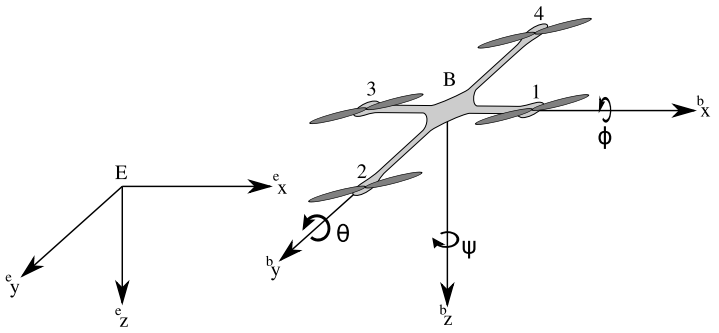

Figure 2.1: The earth frame and body frame

The earth fixed frame is aligned with the earth with the x- and y-axis drawn perpendicular and in the horizontal plane. The z-axis is pointing down which leaves a normal right hand coordinate system in which to describe vectors. This axis-alignment is typically used when dealing with flying objects and is denoted North East Down (NED). In the body frame a similar right hand coordinate system is aligned with the quadrotor as shown in Figure 2.2.

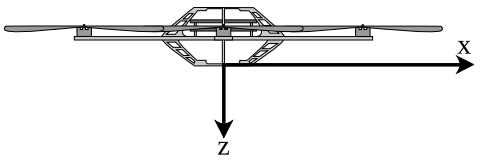

Figure 2.2: The X-3D quadrotor in the body frame

The coordinate system is aligned such that rotor 1 is pointing in the same direction as the positive x-axis. And the center is positioned at the same level as the landing pads. When the quadrotor is located in the center of the test area it will be positioned in position $\begin{bmatrix}0 & 0 & 0\end{bmatrix}^\intercal$. The position of the quadrotor is denoted $\boldsymbol{P} = [x, y, z]^\intercal$. The orientation of the quadrotor will be described as a parametrization of the 3×3 transformation matrix, transforming vectors from the earth frame to the body frame. The parametrizations used in this thesis is either a 3-2-1 Euler parametrization $\boldsymbol{\Theta} = \begin{bmatrix}\phi & \theta & \psi\end{bmatrix}^\intercal$ or the quaternion parametrization $\boldsymbol{q} = \begin{bmatrix}q_0 & q_1 & q_2 & q_3\end{bmatrix}^\intercal$. The angular velocity of the body frame is denoted ${}^b\boldsymbol{\omega}$ and the translatory velocity ${}^b\boldsymbol{v}$.

2.3 Orientation and Rotations

The orientation of the quadrotor will as previously mentioned be described as a parametrization of the transformation from the earth frame to the body frame. In this thesis the quaternion parametrization is used primarily, but also the Euler angle parametrization will be described.

The Euler angles are widely used, since they have a very clear physical interpretation and are of minimum dimensionality. The minimum required dimensionality for describing an orientation in 3 dimensions is 3. However, the orientation can not be both global and non-singular with less then 4 dimensions [Bak, 2002, p. 29]. The parametrizing of the rotational matrix, using Euler angles, includes multiple trigonometric functions, which leaves the transformation non-linear and is subject to gimbal lock.

The quaternion parametrization only involves quadratic expressions, but is still non-linear. Using the quaternions leaves a close to linear kinematic equations and no singularities.

In the remaining sections of this chapter the Euler angle and quaternion parametrizations will be described in further detail.

2.3.1 Euler Angles

The Euler angle parametrization utilizes that the orientation of one cartesian coordinate system, with respect to another, can always be described by three successive rotations. The orientation of a coordinate system can therefore be described by the z,y,x (also called 3-2-1) right-hand rotation sequence that is required to get from earth frame into alignment with the body frame. Other sequences can be used as well, but the 3-2-1 sequence is commonly used when dealing with UAVs. The rotations are denoted as follows:

- Right-hand rotation about the z-axis (positive ψ)
- Right-hand rotation about the new y-axis (positive θ)
- Right-hand rotation about the new x-axis (positive ϕ)

Matrices describing the rotations about the three axises are defined by [Bak, 2002, p. 15] as

$$C_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \sin(\phi) \\ 0 & -\sin(\phi) & \cos(\phi) \end{bmatrix} \tag{2.1}$$

$$C_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & 1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix} \tag{2.2}$$

$$C_z(\psi) = \begin{bmatrix} \cos(\psi) & \sin(\psi) & 0 \\ -\sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

The successive rotations and the rotational matrix from earth frame to body frame is hereby defined as

$$\begin{aligned} {}^b\boldsymbol{C}_e(\boldsymbol{\Theta}) &= C_x(\phi)C_y(\theta)C_z(\psi) \\ &= \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ (-c\phi s\psi + s\phi s\theta s\psi) & (c\phi c\psi + s\phi s\theta s\psi) & s\phi c\theta \\ (s\phi s\psi + c\phi s\theta c\psi) & (-s\phi c\psi + c\phi s\theta s\psi) & c\phi c\theta \end{bmatrix} \end{aligned} \tag{2.4}$$

where $c = cos$ and $s = sin$. Because ${}^b\boldsymbol{C}_e$ is orthonormal [Bak, 2002, p. 13] the inverse transformation can be described at the transpose of ${}^b\boldsymbol{C}_e$.

$${}^b\boldsymbol{C}_e^{-1}(\boldsymbol{\Theta}) = {}^b\boldsymbol{C}_e^{\intercal}(\boldsymbol{\Theta}) \tag{2.5}$$

The rotation from body to earth frame is given by

$$ {}^{e}\boldsymbol{C}_{b}(\boldsymbol{\Theta}) = {}^{b}\boldsymbol{C}_{e}^{\intercal}(\boldsymbol{\Theta}) = \begin{bmatrix} c\theta c\psi & (-c\phi s\psi + s\phi s\theta s\psi) & (s\phi s\psi + c\phi s\theta c\psi) \\ c\theta s\psi & (c\phi c\psi + s\phi s\theta s\psi) & (-s\phi c\psi + c\phi s\theta s\psi) \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \tag{2.6} $$

where $\boldsymbol{\Theta}$ is the vector of Euler angles

2.3.2 Euler Rates

The relationship between Euler angle time derivatives ($\dot{\boldsymbol{\Theta}}$) and the angular velocity (${}^{b}\boldsymbol{\omega}$) can in a similar way be described with a rotational matrix [Bak, 2002, p.25].

$$ \begin{aligned} {}^{b}\boldsymbol{\omega} &= \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + \boldsymbol{C}_{x}(\phi) \left(\begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + \boldsymbol{C}_{y}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} \right) \\ &= \begin{bmatrix} 1 & 0 & -s\theta \\ 0 & c\phi & s\phi c\theta \\ 0 & -s\phi & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \\ &= {}^{b}\boldsymbol{H}_{e}(\boldsymbol{\Theta})\dot{\boldsymbol{\Theta}} \end{aligned} \tag{2.7} $$

Where $\boldsymbol{C}_{x}$ and $\boldsymbol{C}_{y}$ are presented in Equation 2.1 and 2.2. ${}^{b}\boldsymbol{H}_{e}$ is the transformation matrix from earth frame to body frame. By inverting ${}^{b}\boldsymbol{H}_{e}$ the transformation matrix from body rates to the derivative of the Euler angles are found.

$$ {}^{e}\boldsymbol{H}_{b}(\boldsymbol{\Theta}) = {}^{b}\boldsymbol{H}_{e}^{-1}(\boldsymbol{\Theta}) \tag{2.8} $$

Where $t = tan$, and $\boldsymbol{\Theta}$ are Euler angles. It can be shown that [Stevens and Lewis, 2003, p. 28]

$$ \begin{aligned} \dot{\boldsymbol{\Theta}} &= {}^{e}\boldsymbol{H}_{b}(\boldsymbol{\Theta}){}^{b}\boldsymbol{\omega} \\ &= \begin{bmatrix} 1 & t\theta s\phi & t\theta c\phi \\ 0 & c\phi & -s\phi \\ 0 & s\phi/c\theta & c\phi/c\theta \end{bmatrix} {}^{b}\boldsymbol{\omega} \end{aligned} \tag{2.9} $$

In Equation 2.9 it is seen that when θ reaches $\frac{\pi}{2}$ the matrix will be singular. Also given the trigonometric functions this parametrization is not linear.

2.3.3 Quaternions

An alternative way of parametrizing the rotation matrix is by using quaternions. The quaternion has its basis in the Euler axis/angle representation. The Euler axis/angle parametrization is a unit vector ($\boldsymbol{e}$) orthogonal to the plane of rotation and an angle (θ). The direction of the vector gives the direction of the rotation and the angle the amplitude of the rotation.

Quaternions are similar but the rotation axis and angle have been combined to a single 4 dimensional vector of unit length. Three of the dimensions describe the direction of the rotation and the last is used to scale the vector

to have length 1. Equation 2.10 illustrates different representations of the quaternion.

$$\boldsymbol{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = q_0 + q_1\boldsymbol{i} + q_2\boldsymbol{j} + q_3\boldsymbol{k} \tag{2.10}$$

The conversions between the Euler axis/angle and the quaternion might help with the understanding. This is shown in Equation 2.11 and 2.12.

$$\boldsymbol{q} = cos(\frac{\theta}{2}) + sin(\frac{\theta}{2}) \cdot \boldsymbol{e} \tag{2.11}$$

$$\boldsymbol{e} = \frac{q_{1..3}}{\|q_{1..3}\|} \qquad \theta = 2\arccos(q_0) \tag{2.12}$$

This definition of the quaternions ensures a unique quaternion for every value of θ in the range of $\pm\pi$ [Stevens and Lewis, 2003, p. 18].

Quaternions can be multiplied together using the quaternion product operation defined as in Equation 2.13 and 2.14 [Stevens and Lewis, 2003] where $*$ denotes the quaternion multiplication.

$$\boldsymbol{p} * \boldsymbol{q} \triangleq \boldsymbol{\Lambda_p q} \tag{2.13}$$

$$\boldsymbol{\Lambda_p} = \begin{bmatrix} p_0\boldsymbol{I} - [\boldsymbol{p}_{1..3}\times] & \boldsymbol{p}_{1..3} \\ -\boldsymbol{p}_{1..3}^\intercal & p_0 \end{bmatrix} \tag{2.14}$$

Quaternions are not like Euler angles caught in a gimbal lock and the parametrization of the rotational matrix contains no trigonometric functions.

Rotations of vectors using quaternions are convenient, given the unit length. A Euclidean vector written as a quaternion can be seen as a normal quaternion with zero scalar part as shown in Equation 2.15.

$$\boldsymbol{u} = [0 \quad \boldsymbol{u}_{1..3}]^\intercal \tag{2.15}$$

A requirement for a valid transform is that the transformed vector also will have a zero scalar part. The most commonly used transform is the one shown in Equation 2.16 [Bak, 1999, p. 31][Stevens and Lewis, 2003, p.19].

$$\boldsymbol{v} = \boldsymbol{q}^{-1}\boldsymbol{u}\boldsymbol{q} \tag{2.16}$$

$$= \begin{bmatrix} 0 \\ 2\boldsymbol{q}_{1..3}(\boldsymbol{q}_{1..3}^\intercal\boldsymbol{u}) + (q_0^2 - \boldsymbol{q}_{1..3}^\intercal\boldsymbol{q}_{1..3})\boldsymbol{u} - 2q_0(\boldsymbol{q}_{1..3} \times \boldsymbol{u}) \end{bmatrix} \tag{2.17}$$

This transformation can be seen as the parametrization of the rotational matrix $\boldsymbol{C}(\boldsymbol{q})$.

$$\boldsymbol{C}(\boldsymbol{q}) = 2\boldsymbol{q}_{1..3}\boldsymbol{q}_{1..3}^\intercal + (q_0^2 - \boldsymbol{q}_{1..3}^\intercal\boldsymbol{q}_{1..3})\boldsymbol{I} - 2q_0[\boldsymbol{q}_{1..3}\times] \tag{2.18}$$

Where $[\boldsymbol{q}_{1..3}\times]$ is the cross product matrix shown in Equation 2.19.

$$[\boldsymbol{q}_{1..3}\times] = \begin{bmatrix} 0 & -q_3 & q_2 \\ q_3 & 0 & -q_1 \\ -q_2 & q_1 & 0 \end{bmatrix} \tag{2.19}$$

Evaluating the rotational matrix in 2.18 results in the following rotational matrix from earth frame to body frame ${}^{b}\boldsymbol{C}_{e}$ [Stevens and Lewis, 2003, p. 31].

$$ {}^{b}\boldsymbol{C}_{e}(\boldsymbol{q}) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.20} $$

The reverse rotation from body frame to earth frame is given as Equation 2.21.

$$ {}^{e}\boldsymbol{C}_{b}(\boldsymbol{q}) = {}^{b}\boldsymbol{C}_{e}(\boldsymbol{q})^{\intercal} = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix} \tag{2.21} $$

2.3.4 Quaternion Kinematics

Previously the conversion from an Euler axis/angle was described in Equation 2.11. This conversion will now be the basis for a closer look at how the quaternion is changes over time. Let the quaternion $\boldsymbol{q}(t)$ describe the orientation at the current time and let an infinitesimal rotation be described as the unit vector $\hat{\boldsymbol{s}}$ and the magnitude be given by $\hat{\omega}$. Then for a small time interval Δt the rotation can be described as in Equation 2.22 using small angle approximation.

$$ \Delta\boldsymbol{q}(\Delta t) \quad = \quad \begin{bmatrix} 1 \\ \frac{1}{2}\hat{\omega}\hat{\boldsymbol{s}} \end{bmatrix} \tag{2.22} $$

As previously done with Euler angle rotations the quaternions can be combined in a similar way. Therefore $\boldsymbol{q}(t + \Delta t)$ can be described as a quaternion multiplication of the two quaternions.

$$ \boldsymbol{q}(t + \Delta t) \quad = \quad \boldsymbol{q}(t) * [\Delta\boldsymbol{q}(\Delta t) - \boldsymbol{I}_q] \tag{2.23} $$

Where $\boldsymbol{I}_q = [1 \quad 0 \quad 0 \quad 0]^{\intercal}$ is the unity quaternion.

The time derivative of the quaternion $\boldsymbol{q}(t)$ can be described as in Equation 2.24.

$$ \dot{\boldsymbol{q}} \quad = \quad \frac{d\boldsymbol{q}}{dt} = \lim_{\Delta t \to 0} \frac{\boldsymbol{q}(t) * [\Delta\boldsymbol{q}(\Delta t) - \boldsymbol{I}_q]}{\Delta t} \tag{2.24} $$

Defining angular velocity as:

$$ \boldsymbol{\omega} \triangleq \lim_{\Delta t \to 0} \frac{\hat{\omega}\hat{\boldsymbol{s}}}{\Delta t} \tag{2.25} $$

and substituting $\Delta\boldsymbol{q}(\Delta t)$ with 2.22 leaves Equation 2.26 where the time indices are ignored.

$$ \dot{\boldsymbol{q}} \quad = \quad \frac{1}{2}\boldsymbol{q} * \boldsymbol{\omega} \tag{2.26} $$

Using the definition of the quaternion multiplication from 2.14 Equation 2.26 can be written in matrix form.

$$ \dot{\boldsymbol{q}} \quad = \quad \frac{1}{2}\begin{bmatrix} 0 & -\boldsymbol{\omega}^{\intercal} \\ \boldsymbol{\omega} & [\boldsymbol{\omega}\times] \end{bmatrix}\boldsymbol{q} = \frac{1}{2}\begin{bmatrix} 0 & -\omega_1 & -\omega_2 & -\omega_3 \\ \omega_1 & 0 & \omega_3 & -\omega_2 \\ \omega_2 & -\omega_3 & 0 & \omega_1 \\ \omega_3 & \omega_2 & -\omega_1 & 0 \end{bmatrix}\boldsymbol{q} = \frac{1}{2}\boldsymbol{\Omega}_{\boldsymbol{\omega}}\boldsymbol{q} \tag{2.27} $$

Where $\boldsymbol{\Omega}_{\boldsymbol{\omega}}$ is the matrix describing the quaternion multiplication with the angular velocity vector.

Chapter 3

Modelling of a Quadrotor

3.1 How do a Quadrotor Fly?

A quadrotor is equipped with four motors with each a rotor attached. When the rotor spins lift is generated. When the quadrotor is aligned with the horizontal plane and the sum of the lift generated ($\boldsymbol{F}_{lift}$) is equal to the gravitational force the quadrotor is hovering. If F_{lift} is increased the quadrotor will start to climb, if $\boldsymbol{F}_{lift}$ is decreased the quadrotor will start to decent.

Movement in the horizontal plane is done by tilting the quadrotor in the desired direction. When the quadrotor is tilted the direction of $\boldsymbol{F}_{lift}$ is no longer aligned with the earth frame z-axis but can be dissolved in a z-axis component and a horizontal component as seen in Figure 3.1. The horizontal component result in an acceleration in the horizontal plane.

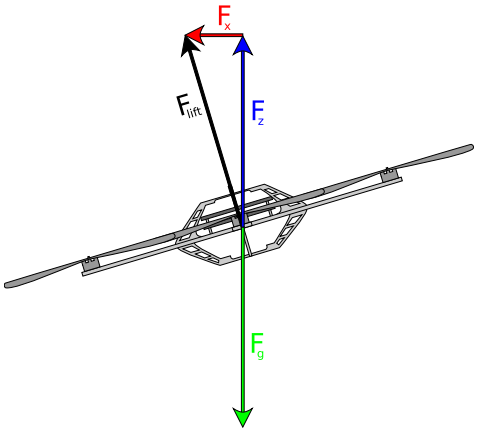

Figure 3.1: $\boldsymbol{F}_{lift}$ dissolved in a z-axis component and a horizontal component

The rotors spin in opposite direction in pairs as illustrated in Figure 3.2. This is to prevent the quadrotor from spinning around the z-axis as an effect of the counter torque generated by the spinning rotors. When the rotors spin in opposite direction the counter torque is equalized when all rotors rotates with equal speed. It should be noted that, in the following figures, the arrows indicate the direction of the rotation and the size is proportional to the speed of the rotor.

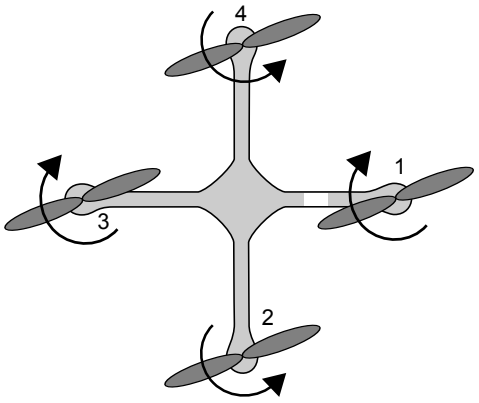

Figure 3.2: Quadrotor in hover

Tilting of the quadrotor is done by altering the speed of the rotors. In the following are the effect of the inputs roll pitch and yaw defined.

Roll is defined as a rotation around the x-axis in the body frame. This is done by increasing the speed of one of the rotors placed along the y-axis and decreasing the speed of the opposite, while maintaining speed on the remaining rotors. Figure 3.3 illustrates the direction of positive roll, which means that rotor 4 increases speed while rotor 2 decreases speed. Note that the white mark indicates the front of the quadrotor.

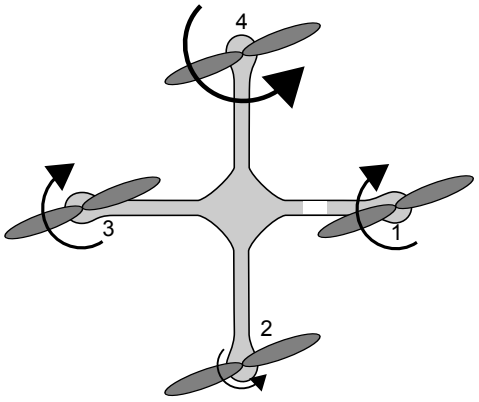

Figure 3.3: Quadrotor performing positive roll

Pitch is defined as the rotation around the y-axis in the body frame. The rotation is done by increasing the speed on one of the rotors placed along the x-axis and decreasing the thrust on the opposite rotor, while maintaining speed on the remaining rotors. The positive pitch is illustrated in Figure 3.4, where rotor 1 increases speed while rotor 3 decreases speed.

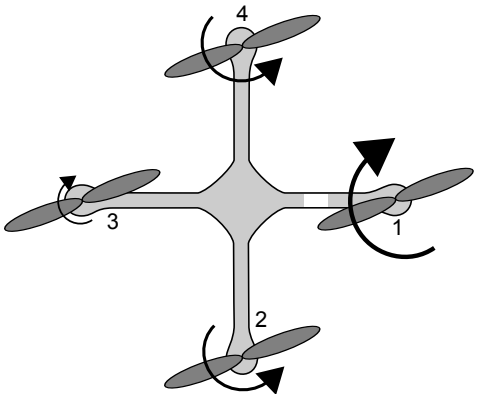

Figure 3.4: Quadrotor performing positive pitch

Yaw is defined as the rotation around the z-axis in the body frame. Yaw is done by altering speed of the rotors along the x-axis compared to the speed on the rotors along the y-axis. The resulting counter torque generated by the rotors will no longer be zero and will result in a rotation around the z-axis. If the rotors on the y-axis are rotating faster than the rotors on the x-axis the body will rotate clockwise (positive yaw) as illustrated in Figure 3.5.

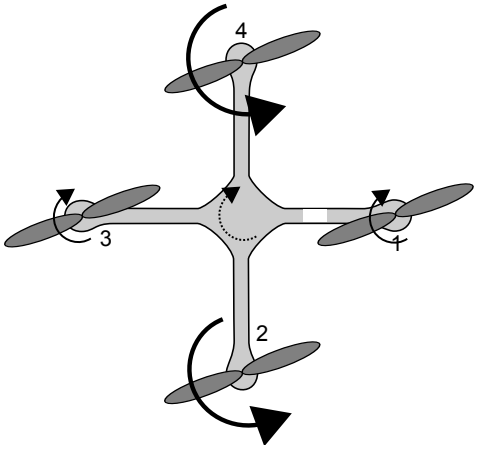

Figure 3.5: Diagram of a quadrotor performing a positive yaw motion

3.2 Model Structure and Assumptions

The structure of the model for the quadrotor that will be derived is illustrated in Figure 3.6. The purpose of this illustration is to give an overview of how the model is structured and how it is derived. The input vector is defined as $\boldsymbol{u} = \begin{bmatrix} u_\phi & u_\theta & u_{thrust} & u_\psi \end{bmatrix}$ containing commands for roll, pitch, thrust and yaw. The input serves as references to the on-board heading hold controller holding the desired angular velocity in the body frame (${}^b\boldsymbol{\omega}$). The force generated by the rotors ($\boldsymbol{F}_{lift}$) is derived using the knowledge about the on-board controller and imperial experiments. Then an expression of the acceleration in body frame (${}^b\dot{\boldsymbol{v}}$) will be derived from the force generated by the lift. The model concludes by including the final integrations and rotations into the earth fixed frame deriving the time derivative of the orientation ($\dot{\boldsymbol{q}}$) and position ($\dot{\boldsymbol{P}}$) in the earth frame. The model will then be linearized and converted to state-space form.

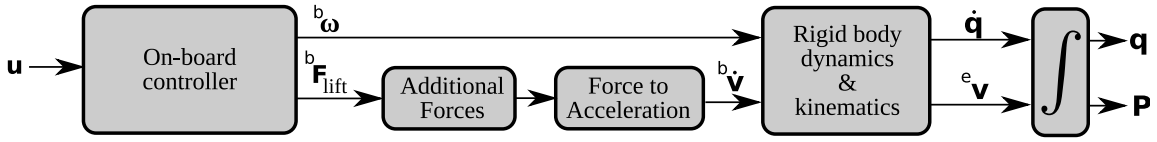

Figure 3.6: Illustration of the model structure for the X-3D quadrotor

In the derivation a number of assumptions are made to simplify the model. These assumptions are either based on prior models derived for quadrotors [Bouabdallah, 2007][Friis et al., 2009] or by empirical observations.

- The quadrotor is assumed to be a rigid body. With the exeption of the roters, that both rotates and are bendable, this is close to be true. The effect of the bending roters is assumed to very small due to the on-board controller.
- The gyro effect due to the angular momentum of the rotors [Bouabdallah, 2007] is neglected due to the on-board controller will compensate for this.
- The on-board angular rate controller and the motor controllers are considered fast enough to neglect both the dynamics of the motors and the dynamic in changing the angular rate. This is further discussed and supported in Section 3.3.
- Due to the small size of the quadrotor and the controlled environment in which it is flying, it is assumed that external effects like wind or turbulence do not affect the quadrotor.

3.3 The On-board Controllers

The purpose of this section is to determine the relation from the input given to the quadrotor (u_ϕ, u_θ, u_{thrust} and u_ψ) to angular rate ${}^b\boldsymbol{\omega}$ and the lift force $\boldsymbol{F}_{lift}$ of the quadrotor. Both factors are depending on the on-board controllers. The knowledge of the on-board software of the X-3D quadrotor is gathered from a research paper about the development of the X-3D quadrotor [Gurdan et al., 2007], the manual [AscTec, 2009] and from personal contact with the developers.

Both the angular velocity controllers and the thrust controller are located on the sensor board that handles all sensor data, filtering, and control. The board is equipped with a 60 MHz ARM7 microprocessor and runs the control loop with an update frequency of 1 kHz.

3.3.1 The Angular Rate Controllers

The rate controllers are structured as three independent PD-controllers. One for each axis. In each controller a value K_{stick} is used as a proportional gain that controls how aggressive the quadrotor reacts on the input. Both K_{stick} and the controller values (K_p and K_d) can be configured when setting up the X-3D quadrotor. The block diagram describing the structure of the controller can be seen in Figure 3.7

From observations it is concluded that the doing normal flight (input bandwidth $\sim$ 0.1 Hz) the quadrotor had no problem following the input references. To verify this the bandwidth was estimated by measuring the rise time of the maximum roll input step. The step response can be seen in Figure 3.8.

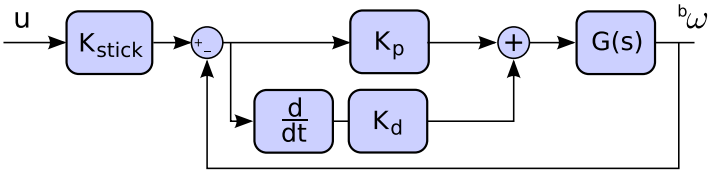

Figure 3.7: Block diagram of the PD-controller structure

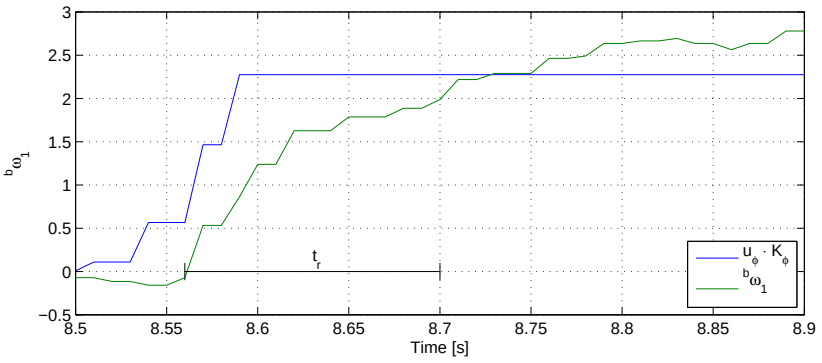

Figure 3.8: Step response of the maximum roll input

The input was chosen to find the worst case bandwidth. It is estimated that with the maximum input, the dynamics of the motor controllers, motors and rotors will have the maximum dampening. On Figure 3.8 the input step is not a step over 1 sample, but more made over several samples. This is cased by the step being made manually doing manual flight. The step corresponds to an angular velocity of 2.25 $\frac{rad}{s}$ (130 $\frac{^o}{s}$) and causes the quadrotor to quickly move towards the wall. This can therefore only be done in manually flight where a pilot is ready to recover the quadrotor. Typically a natural system like this can be estimated with a second order behaviour. The bandwidth of a second order system can according to [Franklin et al., 2006] be approximated as in Equation 3.1.

$$f_{bw} \approx \frac{1.8}{t_r \cdot 2\pi} \approx \frac{0.29}{t_r} \tag{3.1}$$

Given Equation 3.1 the bandwidth can be estimated.

$$f_{bw} \approx \frac{0.29}{0.14} \approx 2.07 \quad [\text{Hz}] \tag{3.2}$$

The rate controller is considered fast enough for the dynamics of the controller to be neglected in this quadrotor model given that the model is to be used for normal flight. The steady state angular rate can according to the manual [AscTec, 2009] be described as in Equations 3.3, 3.4 and 3.5.

$$^b\omega_\phi = \frac{-K_{stick\phi}}{2048} \cdot \frac{2\pi}{360} \cdot u_\phi = K_\phi u_\phi \tag{3.3}$$

$$^b\omega_\theta = \frac{-K_{stick\theta}}{2048} \cdot \frac{2\pi}{360} \cdot u_\theta = K_\theta u_\theta \tag{3.4}$$

$$^b\omega_\psi = \frac{-K_{stick\psi}}{2048} \cdot \frac{2\pi}{360} \cdot u_\psi = K_\psi u_\psi \tag{3.5}$$

However the values was found not to be very accurate and new values of K_ϕ, K_θ and K_ψ was determined using the *system identification toolbox* in Matlab with a zero order transfer function model with a time delay.

Figure 3.9 shows how the angular velocity around the x-axis follows $u \cdot K_\phi$. Part of the delay is assumed to be the communication delay in the series consisting of the transmitter, quadrotor and the Vicon system. The short areas where the values keep constant is due to data corruption in the interface. This is further discussed in Appendix A.

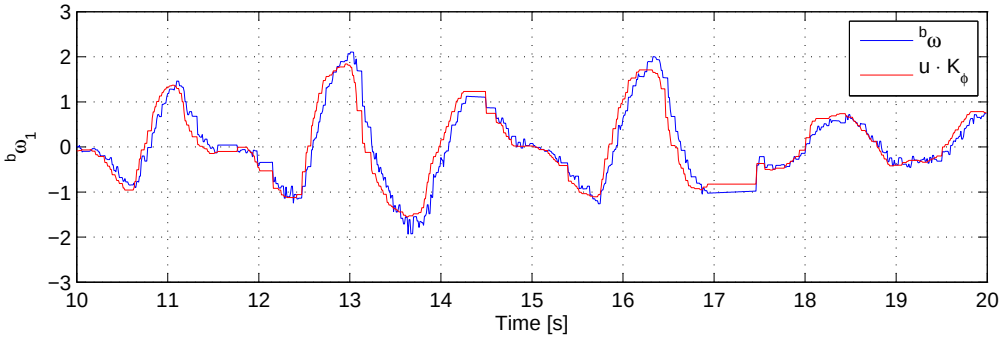

Figure 3.9: Demonstration of how the angular velocity follows the input

3.3.2 Thrust Force Relation

The input determining thrust is proportional to the average speed of the rotors (in perfect hover) when no other input is given. The steady-state relation between the average speed in RPM (Ω_{rotor}) and the thrust input is according to the X-3D developers given as Equation 3.6.

$$\frac{\Omega_{rotor}}{u_{thrust}} = 1.953 \tag{3.6}$$

This however will only be an average, since the quadrotor changes the orientation by in pairs changing the speed of the rotors from the average.

The lift generated by each rotor could be estimated by an aerodynamic model of the rotor, but instead it is chosen to estimate the entire relation from thrust input the lift force. The total force $\boldsymbol{F}_{lift}$ is estimated empirically from the thrust input by measuring how the lift force change while stepping through the input. The experiment is described in more detail in Appendix C.1.

The estimated relation is expressed as the 3rd order polynomial in Equation 3.7 and shown in Figure 3.10 together with the measurements. This however can only be assumed to be correct when a full battery is used.

$${}^b\boldsymbol{F}_{lift} = \begin{bmatrix} 0 \\ 0 \\ 2.073 \cdot 10^{-10} \cdot u_{thrust}^3 - 1.143 \cdot 10^{-6} \cdot u_{thrust}^2 - 1.054 \cdot 10^{-3} \cdot u_{thrust} - 0.903 \end{bmatrix} \tag{3.7}$$

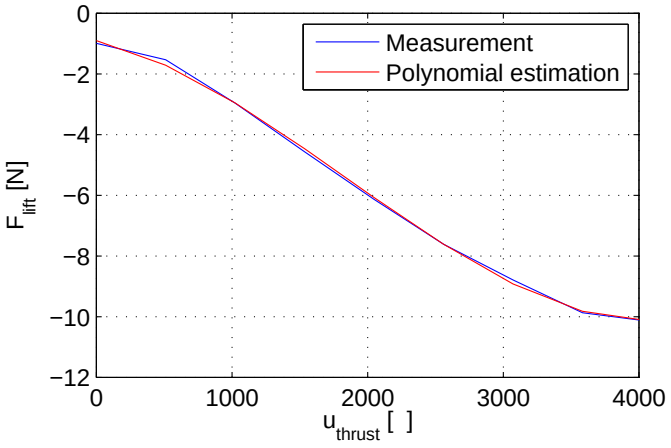

Figure 3.10: Relation between input u_{thrust} and $\boldsymbol{F}_{lift}$

3.4 Forces and Accelerations

The main force apart from $\boldsymbol{F}_{lift}$ affecting the quadrotor is the gravitational force. The gravitational force is always pointed down in the earth fixed frame and can be described as in Equation 3.8.

$$ {}^{e}\boldsymbol{F}_g = m \cdot \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \tag{3.8} $$

Where g is the gravitational acceleration defined to be $9.82 \frac{m}{s^2}$ in Denmark.

Doing normal flight it has been observed that the quadrotor do not rise with a constant acceleration, when the thrust is larger then the gravitational force. This is due to a phenomenon known as induced inflow. Induced inflow is an effect affecting the force generated by a rotating rotor when it starts to move up through the air. When the rotor starts to climb, the airflow through the rotor becomes higher and a smaller force is generated. With a smaller force the vertical speed of the rotor becomes less and the force is once again increased. This effect induces an equilibrium depending on the vertical velocity. Figure 3.11 illustrates this principal.

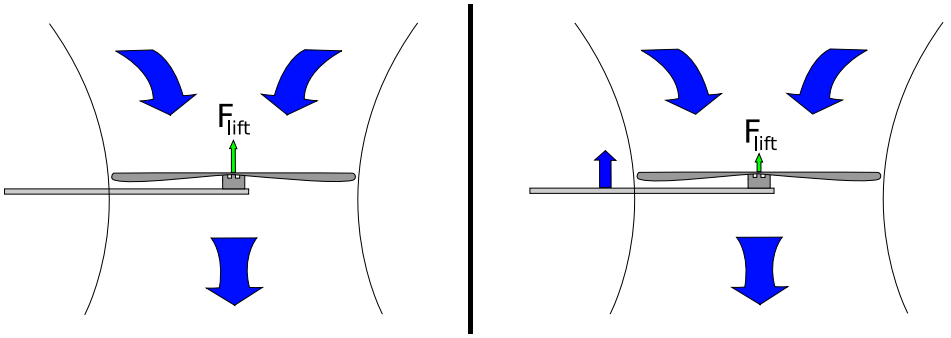

Figure 3.11: Illustration of the phenomenon known as induced inflow

The phenomenon causes the quadrotor to find an equilibrium approximately with a constant velocity. The phenomenon of induced inflow is well researched and good approximations have been made. In this model

however it is found sufficient to use a constant feedback from the vertical velocity to estimate the force generated by the induced inflow. The constant I_i is found by imperial tests in the MTLab and found to approximately $I_i = -0.423$. Further information of the experiment can be found in Appendix C.2.

$$ {}^b\boldsymbol{F}_i = I_i {}^b v_3 \tag{3.9} $$

The total force acting upon the quadrotor, using this model, can be described as in Equation 3.10.

$$ {}^b\boldsymbol{F} = {}^b\boldsymbol{F}_{lift} + {}^e\boldsymbol{F}_g + {}^b\boldsymbol{F}_i \tag{3.10} $$

The rotational matrix ${}^b\boldsymbol{C}_e$ can be used to describe the gravitational force in the body frame. The rotational matrix was previously derived in Section 2.3. Equation 3.11 describes the summed forces.

$$ {}^b\boldsymbol{F} = {}^b\boldsymbol{F}_{lift} + m {}^b\boldsymbol{C}_e \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + I_i \begin{bmatrix} 0 \\ 0 \\ v_3 \end{bmatrix} \tag{3.11} $$

The translatory acceleration of the body frame with respect to the body frame (${}^b\dot{\boldsymbol{v}}$) is now derived. ${}^e\boldsymbol{v}$ describes the translatory velocity of the body frame relative to the earth frame. Newtons second law states that acceleration is force divided by mass [Newton, 1833, p. 15].

$$ \dot{\boldsymbol{v}} = \frac{1}{m}\boldsymbol{F} \tag{3.12} $$

The velocity ${}^e\boldsymbol{v}$ is described relative to the earth frame in Equation 3.13 and the derivative in Equation 3.14.

$$ {}^e\boldsymbol{v} = {}^e\boldsymbol{C}_b {}^b\boldsymbol{v} \tag{3.13} $$

$$ {}^e\dot{\boldsymbol{v}} = {}^e\boldsymbol{C}_b {}^b\dot{\boldsymbol{v}} + {}^e\dot{\boldsymbol{C}}_b {}^b\boldsymbol{v} \tag{3.14} $$

The derivative of the direct cosine matrix can be expressed as the cross product seen in Equation 3.15 [Hughes, 1986, p.23].

$$ {}^e\dot{\boldsymbol{C}}_b {}^b\boldsymbol{v} = {}^e\boldsymbol{C}_b({}^b\boldsymbol{\omega} \times {}^b\boldsymbol{v}) \tag{3.15} $$

Equation 3.12 and Equation 3.15 is inserted into Equation 3.14 and the expression is transformed to body frame and simplified.

$$ \frac{1}{m}{}^e\boldsymbol{F} = {}^e\boldsymbol{C}_b {}^b\dot{\boldsymbol{v}} + {}^e\boldsymbol{C}_b {}^b\boldsymbol{\omega} \times {}^b\boldsymbol{v} \tag{3.16} $$

$$ \frac{1}{m}{}^b\boldsymbol{C}_e {}^e\boldsymbol{F} = {}^b\dot{\boldsymbol{v}} + {}^b\boldsymbol{\omega} \times {}^b\boldsymbol{v} \tag{3.17} $$

$$ {}^b\dot{\boldsymbol{v}} = \frac{1}{m}{}^b\boldsymbol{F} - {}^b\boldsymbol{\omega} \times {}^b\boldsymbol{v} \tag{3.18} $$

3.5 Model Summery

The previously described rotational matrices and kinematics can be used to describe the entire relations between the input commands $\begin{bmatrix} u_\phi & u_\theta & u_\psi & u_{thrust} \end{bmatrix}^\mathsf{T}$ and the change in position $\boldsymbol{P}$, orientation $\boldsymbol{q}$ and velocity ${}^b\boldsymbol{v}$. In

Equations 3.19 to 3.22 this relation is summarized.

$$\dot{\boldsymbol{P}} = {}^{e}\boldsymbol{C}_{b}\,{}^{b}\boldsymbol{v} \tag{3.19}$$

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{\Omega}_{\boldsymbol{\omega}}\boldsymbol{q} \tag{3.20}$$

$${}^{b}\dot{\boldsymbol{v}} = \frac{1}{m}{}^{b}\boldsymbol{F}_{lift} + {}^{b}\boldsymbol{C}_{e}\begin{bmatrix}0\\0\\g\end{bmatrix} + \frac{I_i}{m}\begin{bmatrix}0\\0\\v_3\end{bmatrix} - {}^{b}\boldsymbol{\omega} \times {}^{b}\boldsymbol{v} \tag{3.21}$$

$${}^{b}\boldsymbol{\omega} = \begin{bmatrix}K_\phi \cdot u_\phi\\K_\theta \cdot u_\theta\\K_\psi \cdot u_\psi\end{bmatrix} \tag{3.22}$$

$\boldsymbol{F}_{lift}$ is an expression for the 3rd order polynomial that maps the thrust input to the force generated by the rotors. Because the polynomial is highly nonlinear the thrust force will be used as input to the system, such that the polynomial will be kept out of the model and the controller.

Chapter 4

The Linear Quadrotor Model

To use the model to tune a linear controller requires a strictly linear model. Therefore a general linearization is done here with the purpose of describing the model as a linear state-space model. First step will be identifying the non-linear parts and find appropriate linearization methods and operating points.

Since a quadrotor is usually required to be stabilized around hover this is chosen as the operating point. Equations 4.1 to 4.3 shows the values of states in the operating point.

$$\bar{\boldsymbol{q}} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\intercal \tag{4.1}$$

$${}^b\bar{\boldsymbol{v}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal \tag{4.2}$$

$${}^b\bar{\boldsymbol{\omega}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\intercal \tag{4.3}$$

Most of the non-linearities of the model previously described is due to states multiplied together. This is linearized using a 1st order Taylor approximation as illustrated in Equation 4.4.

$$\begin{aligned} a \cdot b &\approx (\bar{a} + \tilde{a})(\bar{b} + \tilde{b}) = \bar{a}\bar{b} + \bar{a}\tilde{b} + \tilde{a}\bar{b} + \tilde{a}\tilde{b} \\ &\approx \bar{a}\bar{b} + \bar{a}\tilde{b} + \tilde{a}\bar{b} \end{aligned} \tag{4.4}$$

Before the actual linearization the matrices used is calculated when the operating point is inserted.

$${}^e\boldsymbol{C}_b(\boldsymbol{q})\Big|_{\boldsymbol{q}=\bar{\boldsymbol{q}}} = {}^b\boldsymbol{C}_e(\boldsymbol{q})\Big|_{\boldsymbol{q}=\bar{\boldsymbol{q}}} = \boldsymbol{I} \tag{4.5}$$

$${}^b\boldsymbol{\Omega}_{\boldsymbol{\omega}}\Big|_{\boldsymbol{\omega}=\bar{\boldsymbol{\omega}}} = \boldsymbol{0} \tag{4.6}$$

$$\left[{}^b\boldsymbol{\omega}\times\right]\Big|_{\boldsymbol{\omega}=\bar{\boldsymbol{\omega}}} = \boldsymbol{0} \tag{4.7}$$

Linearization of the position and orientation can be done very simple given the chosen method as shown in Equations 4.8 and 4.9.

$$\dot{\boldsymbol{P}} \approx {}^e\boldsymbol{C}_b(\boldsymbol{q})\Big|_{\boldsymbol{q}=\bar{\boldsymbol{q}}} {}^b\boldsymbol{v} + {}^e\boldsymbol{C}_b(\boldsymbol{q}){}^b\bar{\boldsymbol{v}} = {}^b\boldsymbol{v} \tag{4.8}$$

$$\dot{\boldsymbol{q}} \approx \frac{1}{2}\left({}^b\boldsymbol{\Omega}_{\boldsymbol{\omega}}\Big|_{\boldsymbol{\omega}=\bar{\boldsymbol{\omega}}}\boldsymbol{q} + \boldsymbol{\Omega}_{\boldsymbol{\omega}}\bar{\boldsymbol{q}}\right) = \frac{1}{2}\begin{bmatrix} 0 & \omega_1 & \omega_2 & \omega_3 \end{bmatrix}^\intercal \tag{4.9}$$

The acceleration is divided into multiple steps treating each segment by it self. Firstly is the values of the rotational matrix is inserted from Equation 2.20 to linearize the gravitational force.

$$ {}^{b}\boldsymbol{C}_{e}\,{}^{e}\boldsymbol{F}_{g} \;=\; {}^{b}\boldsymbol{C}_{e}\begin{bmatrix}0\\0\\g\end{bmatrix}m = \begin{bmatrix}2(q_1q_3-q_0q_2)\\2(q_2q_3+q_0q_1)\\q_0^2+q_1^2+q_2^2-q_3^2\end{bmatrix}g\cdot m \tag{4.10} $$

The 1st order Taylor approximation is again used when two different states are multiplied. For the last expression it is assumed that q_0^2 can be estimated to be 1 to avoid an affine expression with a bias.

$$ {}^{b}\boldsymbol{C}_{e}\,{}^{e}\boldsymbol{F}_{g} \;\approx\; \begin{bmatrix}-2\cdot q_2\\2\cdot q_1\\1\end{bmatrix}g\cdot m \tag{4.11} $$

The cross product can be approximated like the position and orientation using the 1st order Taylor.

$$ {}^{b}\boldsymbol{\omega}\times{}^{b}\boldsymbol{v} \;=\; [{}^{b}\boldsymbol{\omega}\times]{}^{b}\boldsymbol{v} \approx [{}^{b}\boldsymbol{\omega}\times]\Big|_{\boldsymbol{\omega}=\bar{\boldsymbol{\omega}}}{}^{b}\boldsymbol{v} + {}^{b}\boldsymbol{\omega}{}^{b}\bar{\boldsymbol{v}} = \mathbf{0} \tag{4.12} $$

When the segments are put together the resulting expression is not linear but affine with the bias of the gravitational force. But by redefining the input force as $F_z = (F_{lift} + g\cdot m)$ this affine relation becomes linear.

$$ \dot{\boldsymbol{v}} \;\approx\; \frac{1}{m}\left({}^{b}\boldsymbol{F}_{lift} + \begin{bmatrix}-2\cdot q_2\\2\cdot q_1\\1\end{bmatrix}g\cdot m + I_i\begin{bmatrix}0\\0\\v_3\end{bmatrix}\right) = \begin{bmatrix}-2g\cdot q_2\\2g\cdot q_1\\\frac{1}{m}\left(F_{lift}+g\cdot m+I_iv_3\right)\end{bmatrix} \tag{4.13} $$

$$ \;=\; \begin{bmatrix}-2g\cdot q_2\\2g\cdot q_1\\\frac{1}{m}\left(F_z+I_iv_3\right)\end{bmatrix} \tag{4.14} $$

For an overview the equations 4.8, 4.9 and 4.14 are shown together in the following equations.

$$ \dot{x} \;=\; {}^{b}v_1 \tag{4.15} $$
$$ \dot{y} \;=\; {}^{b}v_2 \tag{4.16} $$
$$ \dot{z} \;=\; {}^{b}v_3 \tag{4.17} $$

$$ \dot{q_0} \;=\; 0 \tag{4.18} $$
$$ \dot{q_1} \;=\; \tfrac{1}{2}\omega_\phi \tag{4.19} $$
$$ \dot{q_2} \;=\; \tfrac{1}{2}\omega_\theta \tag{4.20} $$
$$ \dot{q_3} \;=\; \tfrac{1}{2}\omega_\psi \tag{4.21} $$

$$ {}^{b}\dot{v}_1 \;=\; -2q_2\cdot g \tag{4.22} $$
$$ {}^{b}\dot{v}_2 \;=\; 2q_1\cdot g \tag{4.23} $$
$$ {}^{b}\dot{v}_3 \;=\; \tfrac{1}{m}\left(F_z+I_iv_3\right) \tag{4.24} $$

$$ \dot{\omega}_\phi \;=\; K_\phi\cdot u_\phi \tag{4.25} $$
$$ \dot{\omega}_\theta \;=\; K_\theta\cdot u_\theta \tag{4.26} $$
$$ \dot{\omega}_\psi \;=\; K_\psi\cdot u_\psi \tag{4.27} $$

4.1 state-space Model

The typical way of describing a Multiple Input Multiple Output (MIMO) system is with the state-space representation. This form is also required in calculating the model based controllers developed in a later chapter.

The state-space form consists of the following system:

$$\dot{\boldsymbol{x}}_s = \boldsymbol{A}\boldsymbol{x}_s + \boldsymbol{B}\boldsymbol{u}_s \tag{4.28}$$

$$\tag{4.29}$$

Where $\boldsymbol{x}_s$ is the state vector described in Equation 4.30 and $\boldsymbol{u}_s$ the input described in Equation 4.31.

$$\boldsymbol{x}_s = \begin{bmatrix} \boldsymbol{P} & \boldsymbol{q} & {}^{v}\boldsymbol{v} \end{bmatrix}^{\mathsf{T}} \tag{4.30}$$

$$\boldsymbol{u}_s = \begin{bmatrix} u_\phi & u_\theta & F_z & u_\psi \end{bmatrix}^{\mathsf{T}} \tag{4.31}$$

The system matrices are described below.

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -2g & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2g & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{I_i}{m} \end{bmatrix} \tag{4.32}$$

$$\boldsymbol{B} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2}K_\phi & 0 & 0 & 0 \\ 0 & \frac{1}{2}K_\theta & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}K_\psi \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{m} & 0 \end{bmatrix} \tag{4.33}$$

Chapter 5

Verification of the Quadrotor Model

In the previous two chapters both the linearized and non-linear models have been derived. To evaluate how well the two models perform it is desired to compare them with the behaviour of the real quadrotor. This will be the main focus in this chapter.

Comparing a quadrotor model with a real quadrotor is not straight forward because the quadrotor is not a stable system without a controller. Slight difference in the modelling of the angular velocity will very quickly turn the quadrotor upside down and the states depending on the translatory acceleration will be flawed.

In this chapter the performance of the models is evaluated by modeling the time derivative velocity (${}^b\dot{\boldsymbol{v}}$) and time derivative quaternion ($\dot{\boldsymbol{q}}$) from a state vector from real flight. The true derivative of the velocity and quaternion is calculated from the measurements and compared with the result of the model.

In this way the state of the model will never be allowed to run away. It is in this way a 1 step prediction. Verifying the model in this way have some limitations. It will be difficult to see much dynamic from the modeled derivatives, but since the derivatives are mostly kinematic equations without much dynamic this is accepted.

To record the behaviour of the real quadrotor two test flight where made. One where large input was applied to roll and pitch input and one with a more calm flight close to the operating point. The first test flight is the basis on which the time derivative of the quaternion is further examined and the second flight is used when the time derivative of the velocity is examined. A more detailed description on the test flight and the full measurements can be seen in Appendix C.3.

5.1 Time Derivative of the Quaternion

In general both the non-linear and the linear model follows the tendencies of the system well. Figure 5.1 shows a part of the measurement with the result of the two models. There is small deviations in amplitude, but this is estimated not to be enough to have significant effect.

The biggest difference can be seen when $\dot{q}_3$ is examined. Here the assumption that there is no dynamics in the input angular velocity relation is not very accurate. This is shown in detail on Figure 5.2. The extra angular

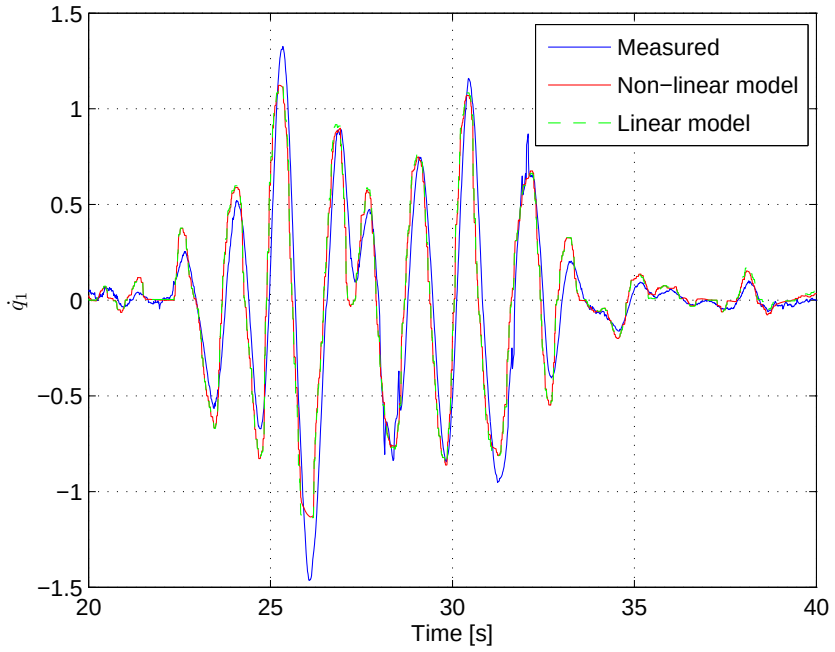

Figure 5.1: Sample of modelled and measured $\dot{q}_1$

velocity is most likely caused by the angular momentum being build up as the quadrotor rotates around the z-axis and the on-board controllers lag of compensating for this momentum. The linear model is very close to the non-linear model, which indicated that the most dominating effects remain even after the linearization. The overall result is considered acceptable for use in the estimator and controllers.

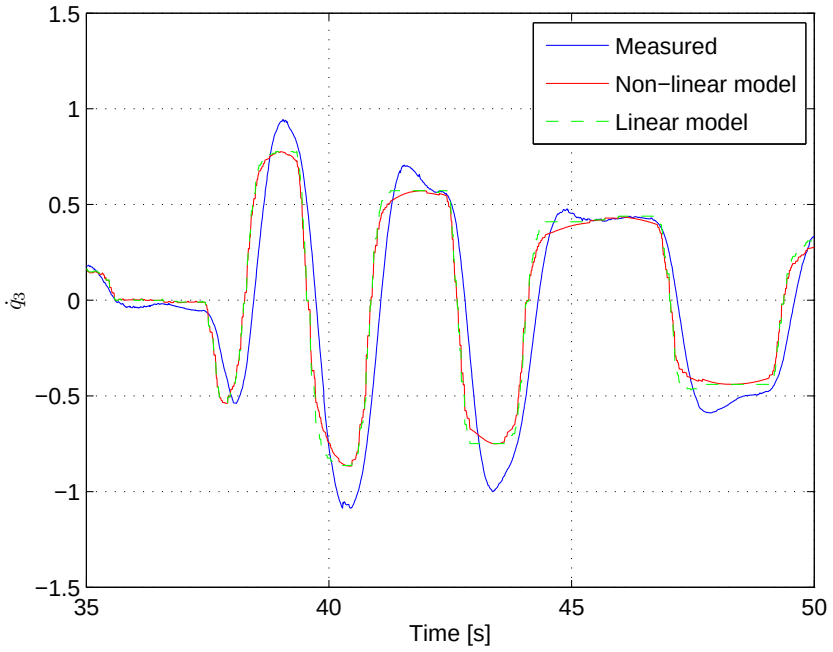

Figure 5.2: Sample of modelled and measured $\dot{q}_3$

5.2 Time Derivative of the Velocity

Figure 5.3 and 5.4 shows the measured acceleration in the body frame with the result of the models. The measured horizontal accelerations are very close to the result of the models and both follows the dynamic very well.

The vertical acceleration however seem to have an offset and parts where the models deviates from the measurement. This offset can very well be contributed the battery. The effect of the battery being discharged over time can be represented as the force generated by the quadrotor slowly declines as time pass. On a down pointing z-axis this will result in a general higher value of the measured acceleration compared with the model. This is what is indicated on Figure 5.4.

If the total measurements found in Appendix C.3 is further examined the offset can be seen to be increasing as time passes which also indicates that the battery could be the cause. This corresponds well with the assumption made of the force polynomial only being accurate when a fully charged battery is used.

On Figure 5.4 close to 46 and 57 seconds in the measurement, the non-linear model seems almost to go in the opposite direction of the measurements. At these two areas the quadrotor was not in the range of the Vicon system. The first area the quadrotor flew out of range in the y-axis and the second it was flown to close to the ceiling. In this period the measurements are not considered valid and will not be further discussed.

Even considering the offset on the acceleration in the vertical axis it is estimated that both the linear and the non linear model will be sufficient to use in the estimator and controllers.

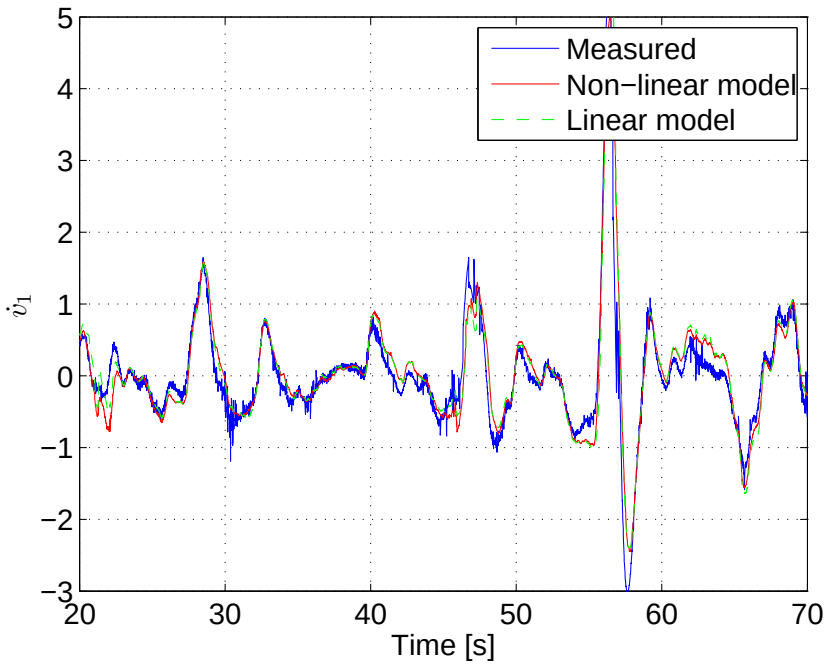

Figure 5.3: Sample of modelled and measured $\dot{v}_1$

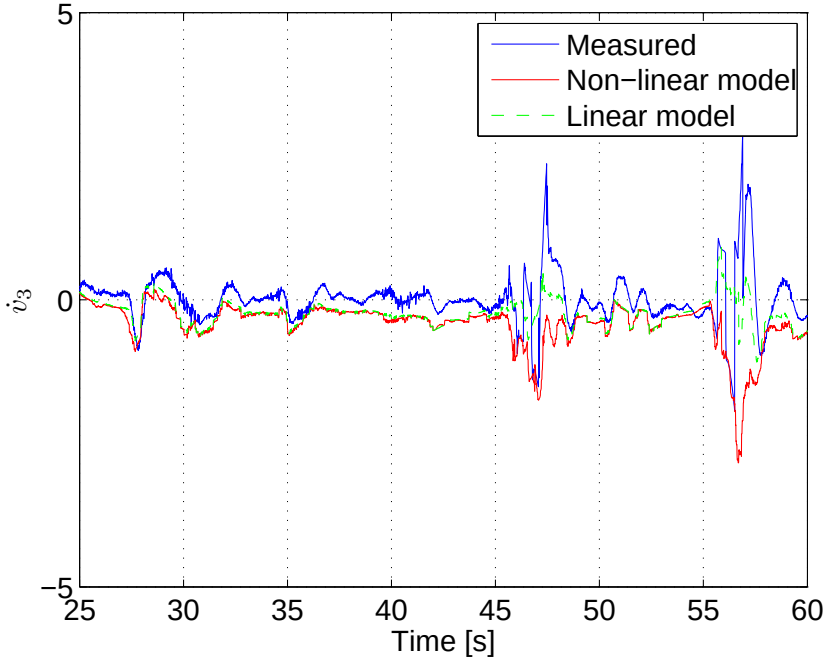

Figure 5.4: Sample of modelled and measured $\dot{v}_3$

Chapter 6

Sensor Model

The purpose of this chapter is to present the various sensors used in the estimation and derive any scaling needed.

6.1 Position and Orientation

For measuring the position and orientation the Vicon system in the MTLab is used. Being part of the MTLab it is already calibrated and scaled to use SI units. To give an impression of how accurate these sensors are the position and orientation measured and the variance recorded. The result is listed in Table 6.1.

	x-axis	y-axis	z-axis
Variance $\boldsymbol{P}$	$4.16 \cdot 10^{-7}$	$3.28 \cdot 10^{-9}$	$2.91 \cdot 10^{-7}$

	q_0	q_1	q_2	q_3
Variance $\boldsymbol{q}$	$1.09 \cdot 10^{-9}$	$5.17 \cdot 10^{-6}$	$9.74 \cdot 10^{-6}$	$3.62 \cdot 10^{-6}$

Table 6.1: Variance for the Vicon system

6.2 Angular Velocity and Acceleration

For measuring angular velocity and acceleration the on-board gyroscopes and accelerometers are used. The values are retrieved through the X-3D interface described in Appendix A. The data received through the interface are raw values directly sampled from the on-board ADC. The values are, in the ResearchPilot documentation, stated to be withing the range of 0 and 1023.

Further analysis of the actual gyro and the accelerometer shows that they both have a linear output, and the data sheets states a conversion factor to SI units. But given the lag of detailed information on the ADC, the conversion factors are determined using Matlab's *system identification toolbox*. The results of the estimated conversion factors are listed in Table 6.2. A comparison of the converted sensor values and the true values (based on Vicon measurements) are shown in Figure 6.1 and 6.2 for the gyroscope and accelerometer.

	1st axis	2nd axis	3rd axis
$\boldsymbol{K}_{gyro}$	0.0144	0.0143	0.0127
$\boldsymbol{K}_{acc}$	0.0086	0.0077	0.0111

Table 6.2: Sensor scales for the gyroscope and accelerometer

On the graf showing the on-board sensor some vertical periods are present. This is due to short communication failures resulting in no new values are received from the quadrotor. The phenomenon is further described in Appendix A concerning the X-3D quadrotor interface. Also on the lowest plot in Figure 6.2 the on-board sensor measurement can be seen to always stay above approximately 4 $\frac{m}{s^2}$. This is assumed to be caused by the saturation of the sensor. In this case the gravitational acceleration have been subtracted from the measurement (knowing the orientation from the Vicon system) to compare with the second derivative of the position. The accelerometer measures originally always the $-9.82\frac{m}{s^2}$ on the z-axis when the quadrotor is close to hover. This indicates that the saturation of the sensor, given the measurements, is close to $-14\frac{m}{s^2}$.

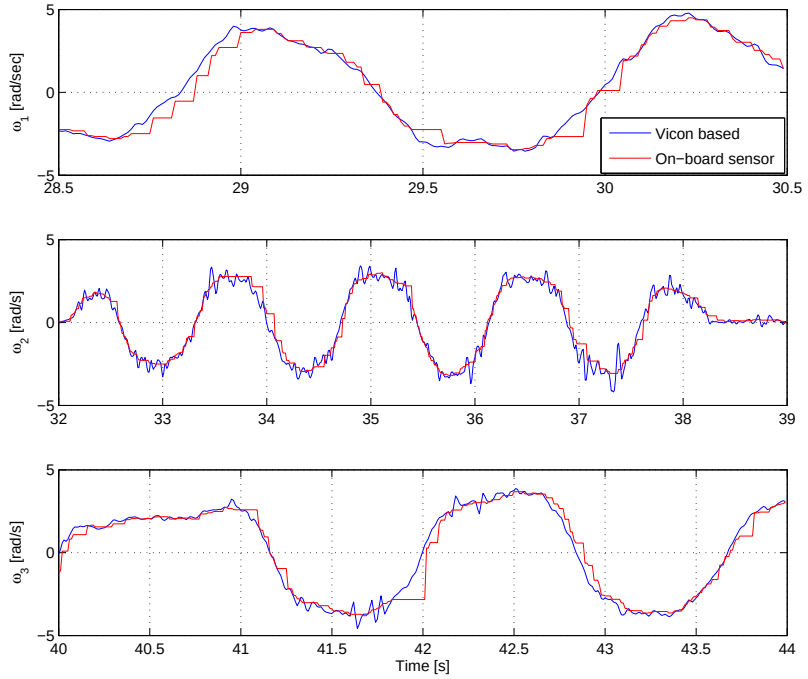

Figure 6.1: Converted gyroscope measurement compared with Vicon measurement

The conversion factor is in both sensors depending on the temperature, but in this setup the quadrotor is always used in the same room, with a constant temperature. Any changes in the conversion factor due to temperature are therefore ignored.

As with the Vicon system the variance of the sensors have been measured as it is needed to determine the co-variance matrix for the estimator in the next Part. The variances are listed in Table 6.3.

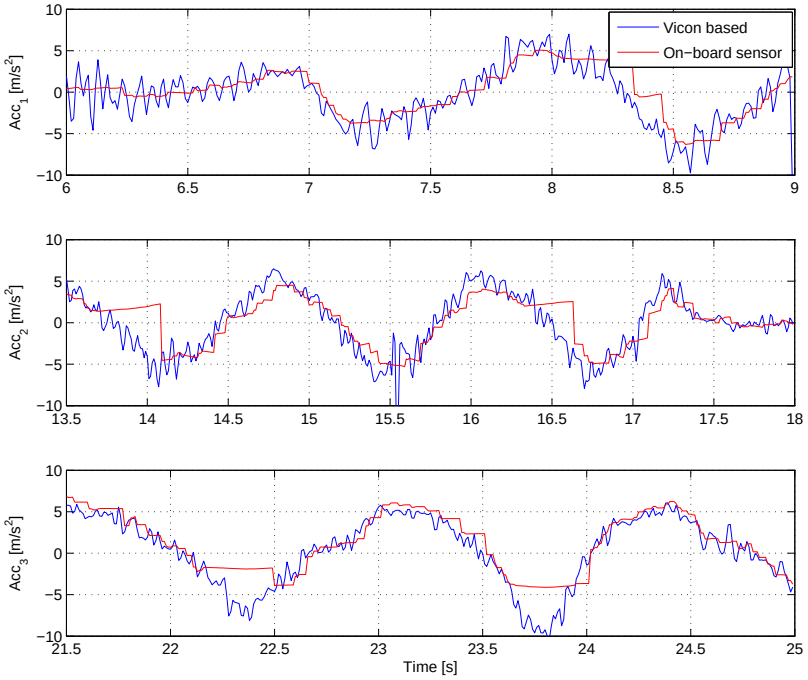

Figure 6.2: Converted accelerometer measurement compared with Vicon measurement

Variance	1st axis	2nd axis	3rd axis
Gyro	$1.24 \cdot 10^{-4}$	$1.94 \cdot 10^{-4}$	$8.54 \cdot 10^{-4}$
Accelerometer	$4.60 \cdot 10^{-3}$	$2.62 \cdot 10^{-3}$	$3.54 \cdot 10^{-3}$

Table 6.3: Variance for the gyro and accelerometer sensors

Part II

State Estimation

Table of Contents

Chapter 7

State Estimation and Sensor Fusion

An important step in obtaining a good flight performance for the quadrotor, is to know where it is. In other words to have a good estimate of the state vector. This chapter describes how this estimate is made given the system model and measurements made in flight.

One might question why an estimator is necessary when measurements are available. If perfect measurements (no noise or uncertainty) could be made for all states an estimator would not be necessary but unfortunately perfect measurements do not exist in the real world. Usually it is not possible to have sensors measuring all states, and the states that are measured are corrupted by measuring noise from the sensor, or some disturbing effects from the surrounding world. All these factors cause uncertainty in the measurements. Instead of perceiving the measurements as the actual truth, they are thought of as noisy samples of the actual state. In this way the measurements become random variables whose mean are the true state.

Several tools exist dealing with estimation problems such as this. A group of filters that are especially efficient are the Bayes filters. Using Bayes filters it is possible to fuse the predicted state based on the previous estimation, with the measurements form the sensors [Thrun et al., 2006]. This is done using Bayes theorem (shown in Equation 7.1).

$$P(\boldsymbol{x}|\boldsymbol{z}) \quad = \quad \frac{P(\boldsymbol{z}|\boldsymbol{x})P(\boldsymbol{x})}{P(\boldsymbol{z})} \tag{7.1}$$

$P(\boldsymbol{x})$ is typically referred to as the prior probability distribution, z the data, and $P(\boldsymbol{x}|\boldsymbol{z})$ the posterior probability distribution.

One particular kind of Bayes filter is the Kalman filter. The Kalman filter assumes that the state and measurements are random variables with a Gaussian distribution. Not only does the filter propagate the state (the mean of the random variable) but it also propagates the uncertainty in form of a covariance matrix. The type of filter used to estimate the system states for the quadrotor is an EKF. Where as the normal Kalman filter only supports linear systems and sensor models, the EKF supports nonlinear models. As shown later the filters are very closely related. The EKF simply linearizes the models using a first order Taylor approximation.

The EKF is placed in between the sensors and the controller as shown in the control scheme in Figure 7.1.

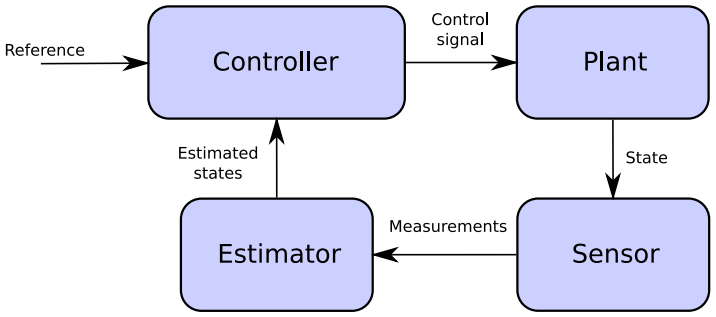

Figure 7.1: Control scheme with estimator

7.1 The Extended Kalman Filter

The EKF is a very powerful and elegant tool to know when dealing with estimation. The mathematical simplicity and intuitive procedure leaves the EKF as a simple way to not only estimate the state of the system but also to merge multiple sensors and also to handle failing sensors. How all of this is done in more detail will be revealed throughout the remainder of this chapter. This section will primarily describe the mathematical background of the EKF which hopefully helps the understanding of why this is such a powerful tool.

To give a short overview, all the essential EKF equations are listed in Table 7.1 [Grenwal and Andrews, 2008] and can be divided into three categories: Estimation model, prediction and update.

Estimation Model

$$\boldsymbol{x}_k = \phi_{k-1}(\boldsymbol{x}_{k-1}) + w_{k-1}, \qquad w_k \sim \mathcal{N}(0, \boldsymbol{Q}_k)$$
$$\boldsymbol{z}_k = h_k(\boldsymbol{x}_k) + v_k, \qquad v_k \sim \mathcal{N}(0, \boldsymbol{R}_k)$$
$$\boldsymbol{\Phi}_{k-1}^{[1]} \approx \left.\frac{\partial \phi_k}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}^{+}}$$
$$\boldsymbol{H}_k^{[1]} \approx \left.\frac{\partial h_k}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k^{-}}$$

Prediction step

$$\hat{\boldsymbol{x}}_k^- = \phi_{k-1}(\hat{\boldsymbol{x}}_{k-1}^+)$$
$$\boldsymbol{P}_k^- = \boldsymbol{\Phi}_{k-1}^{[1]} \boldsymbol{P}_{k-1}^+ \boldsymbol{\Phi}_{k-1}^{[1]\intercal} + \boldsymbol{Q}_{k-1}$$

Update step

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^{[1]\intercal} \left[\boldsymbol{H}_k^{[1]} \boldsymbol{P}_k^- \boldsymbol{H}_k^{[1]\intercal} + \boldsymbol{R}_k\right]^{-1}$$
$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k(\boldsymbol{z}_k - \boldsymbol{H}_k^{[1]} \hat{\boldsymbol{x}}_k^-)$$
$$\boldsymbol{P}_k^+ = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k^{[1]}\right) \boldsymbol{P}_k^-$$

Table 7.1: Discrete EKF Equations

The notation used in Table 7.1 will be used throughout this chapter. The indices $_k$ and $_{k-1}$ indicate the current sample and the previous sample. When a matrix is denoted with an $^{[1]}$ it is a 1st order linear Taylor approximation of a nonlinear function. Ex. $\boldsymbol{\Phi}^{[1]}$ is a linearization of $\phi(\boldsymbol{x})$. The $^+$ and $^-$ indicate a priori or a posterior estimate respectively. That a variable is an estimate is also indicated by the $\hat{}$. By this definition $\hat{\boldsymbol{x}}_k^-$ is the priori

estimate of the state vector at the k sample.

The basic function of both the EKF and the Kalman filter is to merge information from a prediction model with the information from the sensors. The procedure is illustrated in Figure 7.2 and can be described by the following three steps.

1. The current priori state ($\hat{\boldsymbol{x}}_k^-$) is predicted using the previously estimated state ($\hat{\boldsymbol{x}}_{k-1}^+$) and the prediction model.
2. The merging factor also called the Kalman gain ($\boldsymbol{K}_k$), is calculated.
3. The priori state is updated with the sensor information ($\boldsymbol{z}_k$) using the Kalman gain

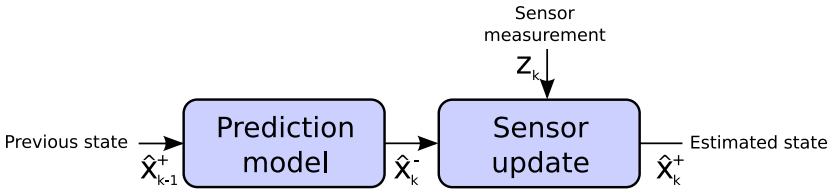

Figure 7.2: Procedure of the EKF and the Kalman filter

7.1.1 Estimation model

The most essential function of a Kalman filter is to merge the information, known from the model, with the information obtained from the sensors. Therefore an important part of the filter is the prediction model. As the name reveals the purpose is to predict the next state given the previous state. The normal Kalman filter uses a linear transition matrix for the prediction where as the EKF supports a non-linear prediction model by doing on-line linearization using the Jacobian matrix. Since the both filters are Gaussian filters both models need to have a zero mean Gaussian noise model as shown in Equation 7.2 and 7.3.

$$\boldsymbol{x}_k = \phi_{k-1}(\boldsymbol{x}_{k-1}) + w_{k-1} \tag{7.2}$$

$$w_k \sim \mathcal{N}(0, \boldsymbol{Q}_k) \tag{7.3}$$

Also a sensor model is needed to relate the measurements to the state and likewise the noise model is required to be Gaussian.

$$\boldsymbol{z}_k = h_k(\boldsymbol{x}_k) + v_k \tag{7.4}$$

$$v_k \sim \mathcal{N}(0, \boldsymbol{R}_k) \tag{7.5}$$

The Jacobian matrix needed for the on-line linearization is the first order partial derivatives of all states as shown in Equations 7.6 and 7.7. The linearization of the system matrix is done with the previous a posteriori state and the linearization of the sensor model is done using the current a priori state.

$$\boldsymbol{\Phi}_{k-1}^{[1]} \approx \left.\frac{\partial \phi_k}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}^+} \tag{7.6}$$

$$\boldsymbol{H}_k^{[1]} \approx \left.\frac{\partial h_k}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_k^-} \tag{7.7}$$

7.1.2 Prediction

The prediction of the state is done using the nonlinear prediction model such as Equation 7.8 describes.

$$\hat{\boldsymbol{x}}_k^- = \phi_{k-1}(\hat{\boldsymbol{x}}_{k-1}^+) \tag{7.8}$$

Also the estimation covariance $\boldsymbol{P}_k$ is propagated through the model. This time however, the linearized model is to be used for the prediction. The estimation covariance indicates how reliable the estimate is. To understand how this covariance is propagated through the model one can start with the definition of the estimated covariance being the squared residual as shown in Equation 7.9.

$$\boldsymbol{P}_k^- = E[\tilde{\boldsymbol{x}}_k^- \tilde{\boldsymbol{x}}_k^{-\mathsf{T}}] \tag{7.9}$$

Where $\tilde{\boldsymbol{x}}_k^- = \hat{\boldsymbol{x}}_k^- - \boldsymbol{x}_k$ is the residual between the estimate and the true state. The residual can be introduced by subtracting $\boldsymbol{x}_k$ from the linearized state prediction.

$$\begin{aligned} \hat{\boldsymbol{x}}_k^- - \boldsymbol{x}_k &= \boldsymbol{\Phi}_{k-1}^{[1]} \hat{\boldsymbol{x}}_{k-1}^+ - \boldsymbol{x}_k \\ \tilde{\boldsymbol{x}}_k^- &= \boldsymbol{\Phi}_{k-1}^{[1]} (\hat{\boldsymbol{x}}_{k-1}^+ - \boldsymbol{x}_{k-1}) - w_{k-1} \\ &= \boldsymbol{\Phi}_{k-1}^{[1]} \tilde{\boldsymbol{x}}_{k-1}^+ - w_{k-1} \end{aligned}$$

The actual covariance propagation can be described by multiplying with $\tilde{\boldsymbol{x}}_k^-$ and talking the estimated value on both sides.

$$\begin{aligned} \boldsymbol{P}_k^- &= E[\tilde{\boldsymbol{x}}_k^- \tilde{\boldsymbol{x}}_k^-] = \boldsymbol{\Phi}_{k-1}^{[1]} E[\tilde{\boldsymbol{x}}_{k-1}^+ \tilde{\boldsymbol{x}}_{k-1}^{+\mathsf{T}}] \boldsymbol{\Phi}_{k-1}^{[1]\mathsf{T}} + E[w_{k-1} w_{k-1}] \\ &= \boldsymbol{\Phi}_{k-1}^{[1]} \boldsymbol{P}_{k-1}^+ \boldsymbol{\Phi}_{k-1}^{[1]\mathsf{T}} + \boldsymbol{Q}_{k-1} \end{aligned} \tag{7.10}$$

7.1.3 Update

The final step to the sensor updated estimate is a balance between the model prediction and the sensor measurement. The weight called the Kalman gain is calculated on the basis of the error covariance (how good the model prediction is) and the sensor covariance (how much the sensors are trusted) as shown in Equation 7.11.

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k^{[1]T} \left[\boldsymbol{H}_k^{[1]} \boldsymbol{P}_k^- \boldsymbol{H}_k^{[1]T} + \boldsymbol{R}_k \right]^{-1} \tag{7.11}$$

The derivation of the Kalman gain and the following Kalman equations are left out for simplicity but they can be found in many textbooks concerning this subject [Grenwal and Andrews, 2008][Thrun et al., 2006].

The update of the priori state to the posteriori state is done using Equation 7.12 where the posteriori state is the sum of the priori state and the weighted difference between the sensor measurement and the modelled sensor measurement.

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{H}_k^{[1]} \hat{\boldsymbol{x}}_k^-) \tag{7.12}$$

The error covariance update is like with the prediction based on the update of the state. The error covariance update is given in Equation 7.13.

$$\boldsymbol{P}_k^+ = \left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k^{[1]} \right) \boldsymbol{P}_k^- \tag{7.13}$$

7.2 The IMU-driven Estimator

An IMU-driven estimator has previously been used with success to estimate the states of a UAV [Bisgaard, 2007][Van Der Merwe
The basic idea, of using IMU-driven estimators, is to use the measured acceleration and angular velocity as the input to the model. In the case of the quadrotor this leaves only the rigid body kinematic and dynamics. With this simplified model it is estimated that the prediction will be more accurate than with the previously described model that is based on the input to the quadrotor. With the IMU-driven model the errors caused by battery discharging and slightly off input scaling will have no effect.

Given that part of the model is now changed also the state vector have to be reviewed. When dealing with sensors it can be useful to estimate any offsets to compensate for these in the model. To the previously described model state is therefore appended six extra states to estimate the offset of the gyroscope and accelerometer. The state vector now contains the position ($\boldsymbol{P}$), the translatory velocity (${}^b\boldsymbol{v}$), the orientation ($\boldsymbol{q}$), the gyro bias ($\boldsymbol{\omega}_b$) and the accelerometer bias ($\boldsymbol{a}_b$).

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{P}_{[1\times3]} & \boldsymbol{q}_{[1\times4]} & {}^b\boldsymbol{v}_{[1\times3]} & \boldsymbol{\omega}_{b[1\times3]} & \boldsymbol{a}_{b[1\times3]} \end{bmatrix}^\intercal \tag{7.14}$$

The measurements of acceleration and angular velocity is preprocessed (see Chapter 6) with a scale conversion factor to convert to SI units. The input vector for the IMU-driven model includes the processed gyro measurement ($\bar{\boldsymbol{\omega}}$) and the processed accelerometer measurement ($\bar{\boldsymbol{a}}$).

$$\boldsymbol{u} = \begin{bmatrix} \bar{\boldsymbol{\omega}}_{[1\times3]} & \bar{\boldsymbol{a}}_{[1\times3]} \end{bmatrix}^\intercal \tag{7.15}$$

The remaining sensors are used without any preprocessing and affects the filter through the sensor state vector. The sensor state vector contains the measurements of the position ($\boldsymbol{P}_{vicon}$) and orientation ($\boldsymbol{q}_{vicon}$) from the Vicon system.

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{P}_{vicon[1\times3]} & \boldsymbol{q}_{vicon[1\times4]} \end{bmatrix}^\intercal \tag{7.16}$$

7.2.1 IMU-driven Model

The model used for the IMU-driven Estimator is based on the model derived in Chapter 3, but deviates in the case of input. The IMU-driven model uses inertial sensors as input and not the control commands send to the quadrotor. The model equations are listed in Equations 7.17 to 7.21

$$\dot{\boldsymbol{P}} = {}^e\boldsymbol{C}_b\,{}^b\boldsymbol{v} \tag{7.17}$$

$${}^b\dot{\boldsymbol{v}} = \bar{\boldsymbol{a}} + \boldsymbol{a}_b + {}^b\boldsymbol{C}_e \begin{bmatrix} 0 & 0 & g \end{bmatrix}^\intercal - {}^b\boldsymbol{\omega} \times {}^b\boldsymbol{v} \tag{7.18}$$

$$\dot{\boldsymbol{q}} = \tfrac{1}{2}\boldsymbol{\Omega}_{\bar{\omega}}\boldsymbol{q} \tag{7.19}$$

$$\dot{\boldsymbol{\omega}}_b = \boldsymbol{0} \tag{7.20}$$

$$\dot{\boldsymbol{a}}_b = \boldsymbol{0} \tag{7.21}$$

The transformation matrices between earth and body frame ${}^b\boldsymbol{C}_e$ and ${}^e\boldsymbol{C}_b$ are described in Section 2.3 on page 10. $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{\omega}}$ is the processed measurements of the acceleration and angular velocity. The combined non-linear continuous model is defined as $f_t(\boldsymbol{x})$.

For the IMU-driven model to be used in the EKF a few derivations are necessary. Previously described equations form the basis for the non-linear prediction model, but a linearization in form of a Jacobian matrix is still needed.

7.2.2 Derivation of the Jacobian

As previously mentioned the online linearization is done using the Jacobian matrix of the system. The Jacobian matrix consists of the partial derived states given the previous a posteriori state. This described mathematically looks like in Equation 7.22.

$$\boldsymbol{\Phi}_{k-1}^{[1]} \approx \frac{\partial \phi_k}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}^{+}} \tag{7.22}$$

$$= \begin{bmatrix} \frac{\partial \phi_{1,k}}{\partial \boldsymbol{x}_1} & \frac{\partial \phi_{1,k}}{\partial \boldsymbol{x}_2} & \cdots \\ \frac{\partial \phi_{2,k}}{\partial \boldsymbol{x}_1} & \frac{\partial \phi_{2,k}}{\partial \boldsymbol{x}_2} & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix} \tag{7.23}$$

The prediction model needed for the described EKF must be a discrete model. The discretization is done using the Zero-Order-Hold (ZOH) method in a way such that the discrete prediction model can be described as in Equation 7.24.

$$\boldsymbol{\Phi}_{k-1}^{[1]} = \boldsymbol{I} + T_s \cdot \frac{\partial f_t(\boldsymbol{x})}{\partial \boldsymbol{x}}\Big|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{k-1}^{+}} \tag{7.24}$$

The sample time of the EKF is chosen to be 100 Hz ($T_s = 0.01$), and given the slow changes of the states it is estimated that a ZOH discretization is sufficient. When the linearization of $f_t(\boldsymbol{x})$ is written on matrix form it becomes more apparent where the actual linearization happens.

$$\begin{bmatrix} \dot{\boldsymbol{P}} \\ \dot{\boldsymbol{q}} \\ {}^b\dot{\boldsymbol{v}} \\ \dot{\boldsymbol{\omega}}_b \\ \dot{\boldsymbol{a}}_b \end{bmatrix} = \begin{bmatrix} 0 & \frac{\partial^e \boldsymbol{C}_b {}^b\boldsymbol{v}}{\partial \boldsymbol{q}} & {}^e\boldsymbol{C}_b & 0 & 0 \\ 0 & \frac{1}{2}\boldsymbol{\Omega}_{\bar{\omega}} & 0 & \frac{1}{2}\frac{\partial \boldsymbol{\Omega}_{\bar{\omega}}\boldsymbol{q}}{\partial \boldsymbol{\omega}_b} & 0 \\ 0 & \frac{\partial^b \boldsymbol{C}_e \boldsymbol{g}}{\partial \boldsymbol{q}} & [\boldsymbol{\omega}\times] & \frac{\partial [\boldsymbol{\omega}\times]^b \boldsymbol{v}}{\partial \boldsymbol{\omega}_b} & \boldsymbol{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{q} \\ {}^b\boldsymbol{v} \\ \boldsymbol{\omega}_b \\ \boldsymbol{a}_b \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{1}{2}\frac{\partial \boldsymbol{\Omega}_{\bar{\omega}}\boldsymbol{q}}{\partial \bar{\boldsymbol{\omega}}} & 0 \\ \frac{\partial [\boldsymbol{\omega}\times]^b \boldsymbol{v}}{\partial \bar{\boldsymbol{\omega}}} & \boldsymbol{I} \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{\boldsymbol{\omega}} \\ \bar{\boldsymbol{a}} \end{bmatrix} \tag{7.25}$$

The partial derivatives can each be calculated as a vector of partial derivatives of the single states as done in Equations 7.26 to 7.29.

$$\frac{\partial^e \boldsymbol{C}_b {}^b\boldsymbol{v}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial^e \boldsymbol{C}_b {}^b\boldsymbol{v}}{\partial q_0} & \frac{\partial^e \boldsymbol{C}_b {}^b\boldsymbol{v}}{\partial q_1} & \frac{\partial^e \boldsymbol{C}_b {}^b\boldsymbol{v}}{\partial q_2} & \frac{\partial^e \boldsymbol{C}_b {}^b\boldsymbol{v}}{\partial q_3} \end{bmatrix} \tag{7.26}$$

$$\frac{\partial \boldsymbol{\Omega}_{\bar{\omega}}\boldsymbol{q}}{\partial \boldsymbol{\omega}} = \begin{bmatrix} \frac{\partial \boldsymbol{\Omega}_{\bar{\omega}}}{\partial \omega_1}\boldsymbol{q} & \frac{\partial \boldsymbol{\Omega}_{\bar{\omega}}}{\partial \omega_2}\boldsymbol{q} & \frac{\partial \boldsymbol{\Omega}_{\bar{\omega}}}{\partial \omega_3}\boldsymbol{q} \end{bmatrix} \tag{7.27}$$

$$\frac{\partial^b \boldsymbol{C}_e \boldsymbol{g}}{\partial \boldsymbol{q}} = \begin{bmatrix} \frac{\partial^b \boldsymbol{C}_e}{\partial q_0}\boldsymbol{g} & \frac{\partial^b \boldsymbol{C}_e}{\partial q_1}\boldsymbol{g} & \frac{\partial^b \boldsymbol{C}_e}{\partial q_2}\boldsymbol{g} & \frac{\partial^b \boldsymbol{C}_e}{\partial q_3}\boldsymbol{g} \end{bmatrix} \tag{7.28}$$

$$\frac{\partial [\boldsymbol{\omega}\times]^b \boldsymbol{v}}{\partial \boldsymbol{\omega}} = \begin{bmatrix} \frac{\partial [\boldsymbol{\omega}\times]}{\partial \omega_1}{}^b\boldsymbol{v} & \frac{\partial [\boldsymbol{\omega}\times]}{\partial \omega_2}{}^b\boldsymbol{v} & \frac{\partial [\boldsymbol{\omega}\times]}{\partial \omega_3}{}^b\boldsymbol{v} \end{bmatrix} \tag{7.29}$$

Each of the above stated partial derivatives have been calculated using Maple and is for convenience not listed here, but can be found in the file `phi.m` on the attached CD.

The remaining sensors used for the EKF already have a linear relation to the states and is given by the matrix $\boldsymbol{H}$ used in the relation in Equation 7.30.

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{x} \tag{7.30}$$

$$\begin{bmatrix} \boldsymbol{P}_{vicon} \\ \boldsymbol{q}_{vicon} \end{bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{P} \\ \boldsymbol{q} \\ {}^b\boldsymbol{v} \end{bmatrix}$$

Finally for the EKF is the previously determined variance of the Vicon system used as the sensor covariance matrix $\boldsymbol{R}$ and the model covariance matrix $\boldsymbol{Q}$ is determined by manually tuning the estimator.

7.3 Fault Handling

While testing the IMU-based EKF a few situations occurred that affected the estimated too much to be acceptable. One situation were while using an additional sensor measuring the air pressure. The pressure sensor was not able to be update the pressure information at the same rate as the EKF. An EKF uses every sample as a new and equally weighted sample, which in this case gave a faulty result.

To ensure the sustainability of the EKF a small algorithm was implemented to handle variating sampling times.

Algorithm 1 Handling of variating sampling times

if $\boldsymbol{z}_{i,k} == \boldsymbol{z}_{i,k-1}$ **then**
  $\boldsymbol{R}_{i,i} = 10^8$
end if

Algorithm 1 basically states that if a sensor reading is detected to be equal to the prior, the variance for this sensor is set sufficiently high for the EKF to ignore the sensor reading. After each sample the sensor covariance matrix $\boldsymbol{R}$ is reset and the algorithm will have no permanent effect.

7.3.1 Flying Outside Vicon Range

Another more frequently occurring fault is the Vicon sensors failing due to the limited Vicon range. As explained in a previous chapter, the Vicon system relies on reflectors on the quadrotor to see the orientation and position using cameras. When either the quadrotor is flying with to great of an angle or the quadrotor flies too close to a wall the Vicon system is not capable of delivering good measurements. Whenever the quadrotor is lost from the sight of the Vicon system the position is kept constant and the orientation is reset. In this way it is possible to determine whenever the Vicon system fails.

To compensate for the failures a similar algorithm as Algorithm 1 is implemented to make sure the EKF ignores the Vicon sensors whenever the measurement fails.

Algorithm 2 Handling failing Vicon measurements

if $\boldsymbol{q} == \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}^\mathsf{T}$ **then**
  $\boldsymbol{R}_{1..7,1..7} = 10^8$
end if

Using this algorithm when ever the quadrotor is flying out of Vicon range results in an estimation based only on the on-board sensors. Several test have been made and a satisfactory estimation performance have been obtained.

7.4 Analysis of Estimator Performance

To verify that the estimator performs as intended the quadrotor is flown manually around both within and outside the range of the Vicon system. The Figures 7.3 and 7.4 document one of these measurements.

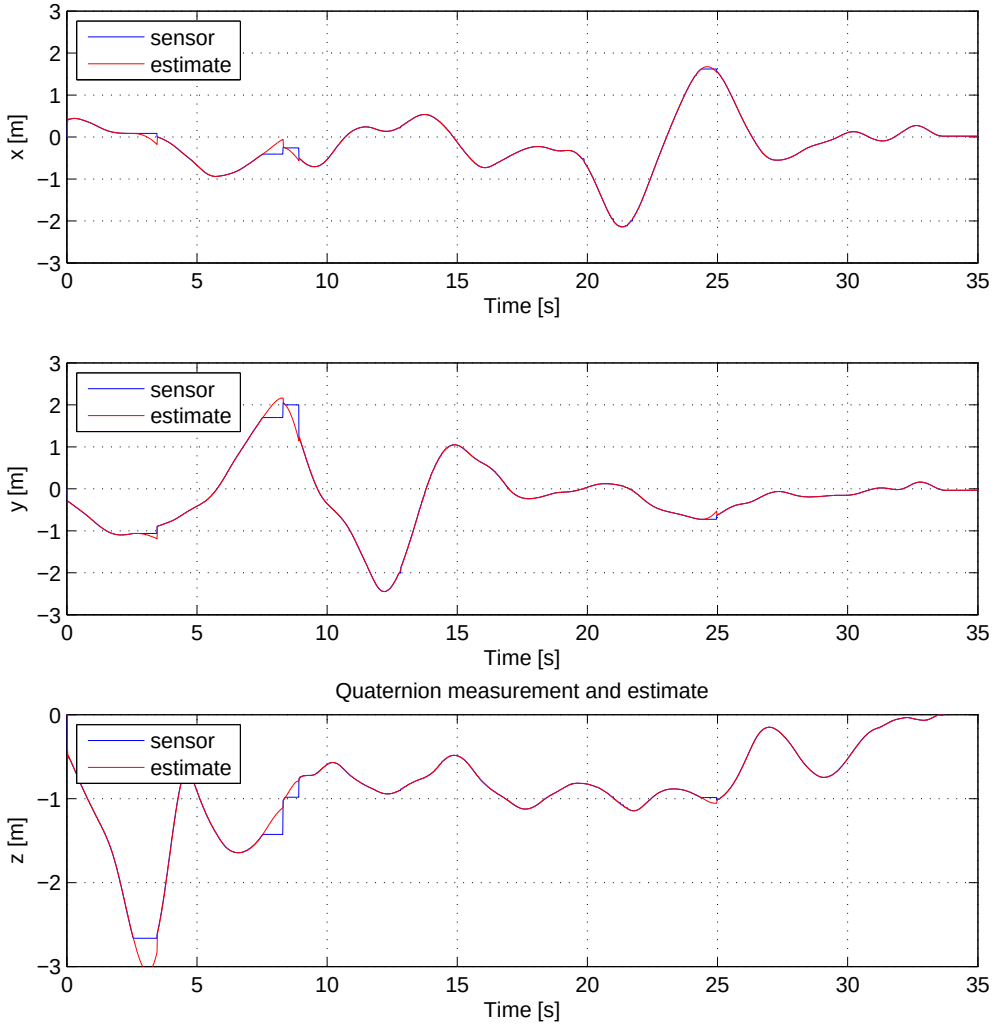

Figure 7.3: A test flight with the position shown

On these figures several occasions of Vicon failures can be found. At approximately 4 and 7 sec the boundary of the range is reached on the z-axis and the y-axis. This do, as predicted, result in short fall outs. Figures 7.5 and 7.6 shows a closer look at the fall out at 7 sec.

On these figures it can be seen that the estimator follows a continuous path when the sensor fails and the path includes dynamics that enables the orientation and the position to still be close when the Vicon sensor is restored.

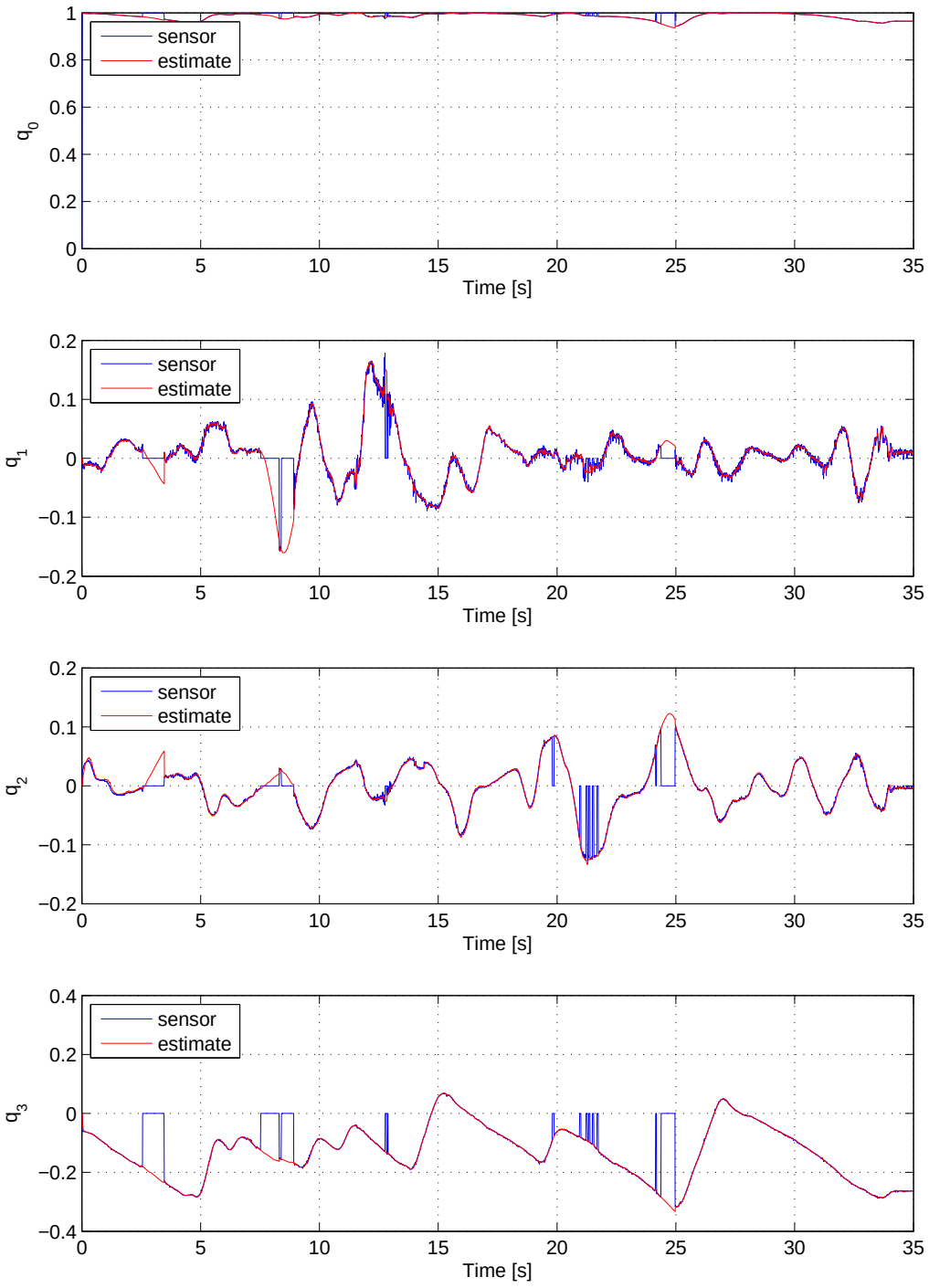

Figure 7.4: A test flight with the quaternion shown

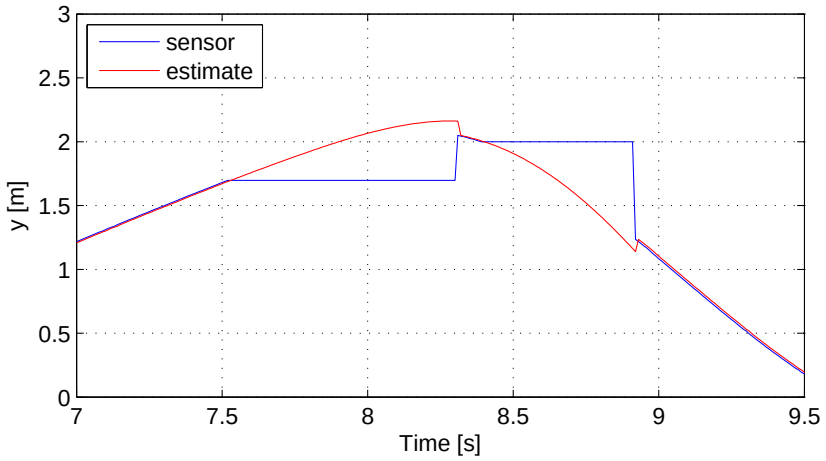

Figure 7.5: A position sample from the test flight

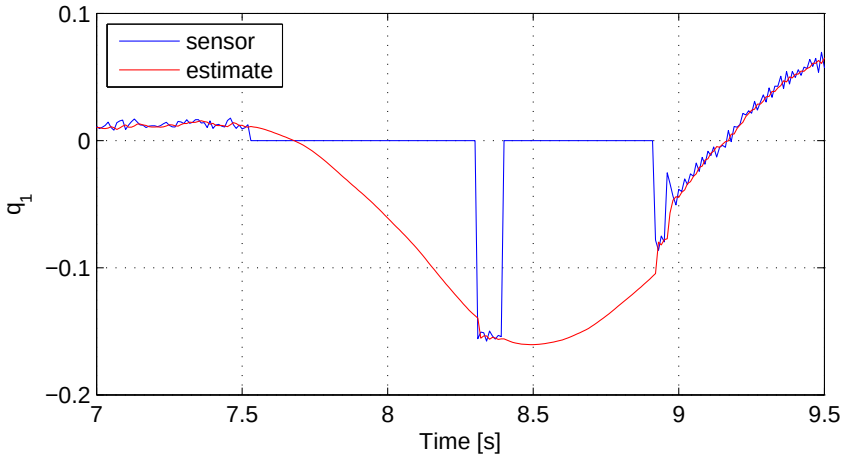

Figure 7.6: A quaternion sample from the test flight

On Figure 7.4 can also be seen other effects. When the Vicon sensors fail around 4 sec into the test flight the q_1 and q_2 values continues further than intended. This is because the previously discussed gyroscope bias not yet have been estimated. With the current settings it takes around 3 sec where the quadrotor should be steady on the ground. If the quadrotor is flying, the time for the bias estimate to settle is longer as indicated by Figure 7.7.

Given this and the performance of other estimation tests the IMU-based estimator is considered good enough to use in combination with the controllers.

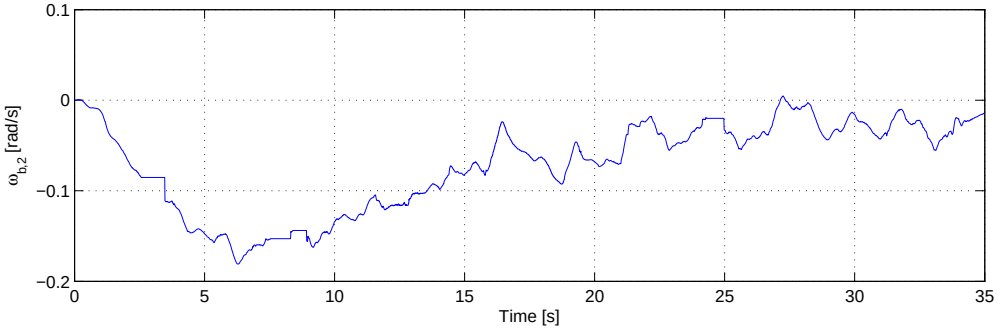

Figure 7.7: Estimate of the gyro bias on the 2nd axis

Part III

Control Algorithms

Table of Contents

Chapter 8

Controlling a Quadrotor

The controller is the final step towards enabling autonomous flight. In the previous chapters details of how the quadrotor is able to navigate depending on the input signal have been discussed. As well as how to more accurately determine the actual state of the quadrotor depending on the measurements from the sensors. This information will in the next few chapters be employed when developing three different controllers.

All the controllers will be feedback controllers. In other words, they will all be based on the difference between part of the state vector and some given reference describing the desired trajectory. The basic idea of the feedback controller is illustrated in Figure 8.1. Where $\boldsymbol{r}$ is the reference, $\boldsymbol{e}$ is the error between the current state and the reference, $K(s)$ is the controller, $\boldsymbol{u}$ is the input to the system, $G(s)$ is the system and $\boldsymbol{y}$ is the output of the system.

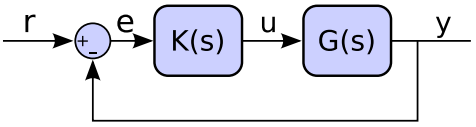

Figure 8.1: The classical feedback controller

The most simple form of a controller that is able to stabilize the quadrotor and have it follow a reference is the PID controller. The PID controller developed for the quadrotor is in fact a controller consisting of 10 PID controllers all connected in a particular way and together form controllers for the position, orientation and velocity. The PID controller is fairly intuitive to understand, but on a more complex MIMO system as the quadrotor the intuitive understanding becomes less transparent. The tuning of PID controller is usually done manually for each controlled state, and this becomes difficult as the complexity of the system increases.

The PID controller is an easy way of making the quadrotor fly, but the performance is usually not as good as with model based controllers. There have been developed a PID controller for the quadrotor which enables it to be stabilized, but the controller is not discussed further in this thesis. It will be used only as a reference to evaluate the performance of the other two controllers.

Another way of formulating the feedback controller is in the state-space form as illustrated in Figure 8.2. The state-space formulation is in principle the same as the classical feedback controller, only the controller K becomes a feedback matrix weighting the states in a particular way to generate the input. When using a

reference, part of the control matrix is used to feed forward the reference.

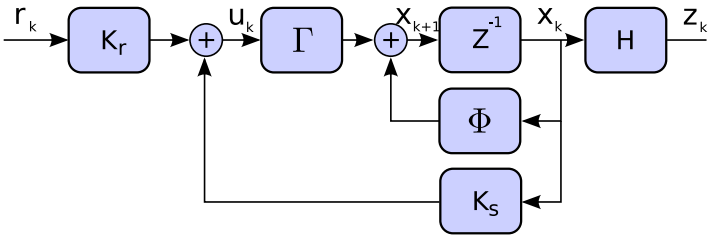

Figure 8.2: The state-space feedback controller

In calculating the feedback matrix many methods exist. The two methods that will be discussed further the Linear Quadratic method and the H_∞ method.

Chapter 9

Linear Quadratic Control

The first controller to be derived for the quadrotor is the discrete LQR. The LQR is a well known method to obtain a simple optimal controller. The LQR is a model based controller and through out this chapter the discrete linear models listed in Equations 9.1 and 9.2 are used as the models for the quadrotor and the reference.

$$\boldsymbol{x}_s(k+1) = \boldsymbol{\Phi}_s \boldsymbol{x}_s(k) + \boldsymbol{\Gamma}_s \boldsymbol{u}(k) \tag{9.1}$$

$$\boldsymbol{x}_r(k+1) = \boldsymbol{\Phi}_r \boldsymbol{x}_r(k) \tag{9.2}$$

Section 9.1 concerns the general derivation of the LQR algorithm and can be skipped. In Section 9.2 the general LQR algorithm is appended with a reference model, allowing the quadrotor to follow a desired trajectory. The final evaluation of the controller will be done together with the other controllers in Chapter 11.

9.1 The General LQR Algorithm

LQR is an interesting case of an optimal controller. What makes it so interesting is the quadratic cost function shown in Equation 9.3.

$$J = \boldsymbol{x}^\intercal \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^\intercal \boldsymbol{R} \boldsymbol{u} \tag{9.3}$$

A cost function describes the immediate cost of being in state $\boldsymbol{x}$ and talking input $\boldsymbol{u}$. The quadratic cost function of the LQR consists of the quadratic weights $\boldsymbol{Q}$ and $\boldsymbol{R}$, weighing respectively the current state and input. What makes this quadratic cost function so interesting is the simple methods to find the minimum (in this case the optimal) value by finding the value where the derivative reaches zero. This is basically the essential step in how the LQR algorithms are derived. The fist step is to define a value function as in Equation 9.4.

$$V(\boldsymbol{x}) = \sum_{k=0}^{\infty} \boldsymbol{x}_k{}^\intercal \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k{}^\intercal \boldsymbol{R} \boldsymbol{u}_k \tag{9.4}$$

In general, a value function is a function of the state describing the total cost going from the current state and following a control policy either through some horizon or until an equilibrium with zero cost is obtained (infinite horizon) [Thrun et al., 2006]. In this case the value function describes the sum the cost function through all

samples until the desired stabilized state is reached where no input is necessary. More specific it is an expression of the value function using the optimal control policy that is sought, called the optimal value function (Equation 9.5).

$$V^{\star}(\boldsymbol{x}) = \min_{\boldsymbol{u}} \sum_{k=0}^{\infty} \boldsymbol{x}_k{}^{\intercal} \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k{}^{\intercal} \boldsymbol{R} \boldsymbol{u}_k \tag{9.5}$$

The expression of value function can be expressed in the current cost and cost for all future samples as in Equation 9.6. Where $\boldsymbol{x}_0$ and $\boldsymbol{u}_0$ describes the current state and the current input.

$$V(\boldsymbol{x}) = \boldsymbol{x}_0{}^{\intercal} \boldsymbol{Q} \boldsymbol{x}_0 + \boldsymbol{u}_0{}^{\intercal} \boldsymbol{R} \boldsymbol{u}_0 + \sum_{k=1}^{\infty} \boldsymbol{x}_k{}^{\intercal} \boldsymbol{Q} \boldsymbol{x}_k + \boldsymbol{u}_k{}^{\intercal} \boldsymbol{R} \boldsymbol{u}_k \tag{9.6}$$

Using the control policy described in Equation 9.7, the future cost can be described by a quadratic parametrization $\boldsymbol{P}$ of the state as in Equation 9.8 [Andersen, 2009].

$$\boldsymbol{u} = \boldsymbol{K} \boldsymbol{x} \tag{9.7}$$

$$V(\boldsymbol{x}) = \boldsymbol{x}^{\intercal} \boldsymbol{P} \boldsymbol{x} \tag{9.8}$$

Combining 9.8 and 9.6 yields an expression of the value function depending only on the current state, the current input and the next state. Including the model the expression can be simplified to only depend on the current state and input as shown in Equation 9.9. The indices $_0$ and $_1$ denotes the current and next state/input respectively.

$$\begin{aligned} V(\boldsymbol{x}) &= \boldsymbol{x}_0{}^{\intercal} \boldsymbol{Q} \boldsymbol{x}_0 + \boldsymbol{u}_0{}^{\intercal} \boldsymbol{R} \boldsymbol{u}_0 + \boldsymbol{x}_1{}^{\intercal} \boldsymbol{P}_1 \boldsymbol{x}_1 \\ V(\boldsymbol{x}) &= \boldsymbol{x}^{\intercal} \boldsymbol{Q} \boldsymbol{x} + \boldsymbol{u}^{\intercal} \boldsymbol{R} \boldsymbol{u} + (\boldsymbol{\Phi} \boldsymbol{x} + \boldsymbol{\Gamma} \boldsymbol{u})^{\intercal} \boldsymbol{P}_1 (\boldsymbol{\Phi} \boldsymbol{x} + \boldsymbol{\Gamma} \boldsymbol{u}) \end{aligned} \tag{9.9}$$

The optimal value function can be described, as previously mentioned, as the smallest possible value function hence where the partial derivative with regards to the input is zero (given the quadratic structure). The partial derivative is calculated in Equation 9.10 and the optimal input given as the structure from 9.7 in Equation 9.11.

$$\frac{\partial V(\boldsymbol{x})}{\partial \boldsymbol{u}} = 0 \tag{9.10}$$

$$\begin{aligned} 2\boldsymbol{R}\boldsymbol{u} + 2\boldsymbol{\Gamma}^{\intercal} \boldsymbol{P}_1 (\boldsymbol{\Phi} \boldsymbol{x} + \boldsymbol{\Gamma} \boldsymbol{u}) &= 0 \\ \boldsymbol{R}\boldsymbol{u} + \boldsymbol{\Gamma}^{\intercal} \boldsymbol{P}_1 \boldsymbol{\Gamma} \boldsymbol{u} &= -\boldsymbol{\Gamma}^{\intercal} \boldsymbol{P}_1 \boldsymbol{\Phi} \boldsymbol{x} \\ \boldsymbol{u} &= -\left[\boldsymbol{R} + \boldsymbol{\Gamma}^{\intercal} \boldsymbol{P}_1 \boldsymbol{\Gamma}\right]^{-1} \boldsymbol{\Gamma}^{\intercal} \boldsymbol{P}_1 \boldsymbol{\Phi} \boldsymbol{x} \end{aligned} \tag{9.11}$$

To derive final expression of the optimal value function in Equation 9.13 and the feedback matrix in Equation 9.12 are combined with 9.9. The optimal parametrization $\boldsymbol{P}$ can be expressed as in Equation 9.14.

$$\boldsymbol{K} = -\left[\boldsymbol{R} + \boldsymbol{\Gamma}^{\intercal} \boldsymbol{P}_1 \boldsymbol{\Gamma}\right]^{-1} \boldsymbol{\Gamma}^{\intercal} \boldsymbol{P}_1 \boldsymbol{\Phi} \tag{9.12}$$

$$V^{\star}(\boldsymbol{x}) = \boldsymbol{x}^{\intercal} \boldsymbol{Q} \boldsymbol{x} + (\boldsymbol{K} \boldsymbol{x})^{\intercal} \boldsymbol{R} (\boldsymbol{K} \boldsymbol{x}) + (\boldsymbol{\Phi} \boldsymbol{x} + \boldsymbol{\Gamma} \boldsymbol{K} \boldsymbol{x})^{\intercal} \boldsymbol{P}_1 (\boldsymbol{\Phi} \boldsymbol{x} + \boldsymbol{\Gamma} \boldsymbol{K} \boldsymbol{x}) \tag{9.13}$$

$$\boldsymbol{P}_0 = \boldsymbol{Q} + \boldsymbol{K}^{\intercal} \boldsymbol{R} \boldsymbol{K} + (\boldsymbol{\Phi} + \boldsymbol{\Gamma} \boldsymbol{K})^{\intercal} \boldsymbol{P}_1 (\boldsymbol{\Phi} + \boldsymbol{\Gamma} \boldsymbol{K}) \tag{9.14}$$

The actual values of $\boldsymbol{P}$ and $\boldsymbol{K}$ can be found by simply iterating from a guessed $\boldsymbol{P}$ until the feedback matrix do no longer change significantly as illustrated in Algorithm 3.

Algorithm 3 Iterates towards a feedback matrix $\boldsymbol{K}$

$\boldsymbol{P} = \boldsymbol{Q}$
$\boldsymbol{K}' = \mathbf{0}$
while $\|K - K'\| > 10^{-5}$ **do**
  $\boldsymbol{K}' = \boldsymbol{K}$
  $\boldsymbol{K} = -\left[\boldsymbol{R} + \boldsymbol{\Gamma}^\intercal \boldsymbol{P}_1 \boldsymbol{\Gamma}\right]^{-1} \boldsymbol{\Gamma}^\intercal \boldsymbol{P}_1 \boldsymbol{\Phi}$
  $\boldsymbol{P} = \boldsymbol{Q} + \boldsymbol{K}^\intercal \boldsymbol{R} \boldsymbol{K} + (\boldsymbol{\Phi} + \boldsymbol{\Gamma}\boldsymbol{K})^\intercal \boldsymbol{P} (\boldsymbol{\Phi} + \boldsymbol{\Gamma}\boldsymbol{K})$
end while

9.2 Following a Fast Trajectory

Throughout this section the model for the quadrotor is appended with a model of a reference in a way such that an expression for the error between the reference state and the quadrotor state becomes apparent. For this combined models a new weight matrix is derived such that the general LQR algorithm can be used to calculate matrices $\boldsymbol{K}_s$ and $\boldsymbol{K}_r$ illustrated in 9.1. Figure 9.1 also illustrates how the matrices from the LQR are connected with the quadrotor and the EKF.

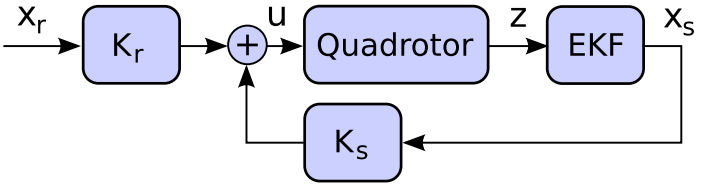

Figure 9.1: Overview of how the feedback matrices, derived through LQR, is connected

Additionally to the models described in Equations 9.1 and 9.2 is the output of the models now described. The matrices $\boldsymbol{H}_s$ and $\boldsymbol{H}_r$ describes the part of the system state that is to follow the reference and the reference respectable.

$$\begin{aligned} \boldsymbol{y}_r(k) &= \boldsymbol{H}_s \boldsymbol{x}_s(k) \\ \boldsymbol{r}(k) &= \boldsymbol{H}_r \boldsymbol{x}_r(k) \end{aligned}$$

Even though the quadrotor have 6 Degrees of Freedom (DoF) the 4 inputs limits the controllable DoF to 4. The states chosen to control is the position (x, y and z) and the orientation around the z-axis (q_3). This yields the following $\boldsymbol{H}_s$ matrix.

$$\boldsymbol{H}_s = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{9.15}$$

The state of the reference ($\boldsymbol{x}_r$) is appended to the system state ($\boldsymbol{x}_s$) to make a combined prediction model as done in Equation 9.16.

$$\begin{bmatrix} \boldsymbol{x}_s(k+1) \\ \boldsymbol{x}_r(k+1) \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Phi}_s & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\Phi}_r \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_s(k) \\ \boldsymbol{x}_r(k) \end{bmatrix} + \begin{bmatrix} \boldsymbol{\Gamma}_s \\ \mathbf{0} \end{bmatrix} \boldsymbol{u}(k) \tag{9.16}$$

The error ($\boldsymbol{e}$) is defined as the difference between the output of the models. In the combined model, this can be described as a single matrix $\boldsymbol{H}$ as done in Equation 9.17.

$$\boldsymbol{e}(k) = \boldsymbol{H}_r\boldsymbol{x}_r - \boldsymbol{H}_s\boldsymbol{x}_s = \begin{bmatrix} -\boldsymbol{H}_s & \boldsymbol{H}_r \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_s(k) \\ \boldsymbol{x}_r(k) \end{bmatrix} = \boldsymbol{H}\boldsymbol{x}(k) \tag{9.17}$$

The cost function can now be redefined to be a quadratic weighting of the error and the input as described in Equation 9.18.

$$J = \boldsymbol{e}^\intercal \boldsymbol{Q}_e \boldsymbol{e} + \boldsymbol{u}^\intercal \boldsymbol{R}\boldsymbol{u} \tag{9.18}$$

Combining 9.17 and 9.18 yields a cost function with a similar structure as the default LQR algorithm.

$$\begin{aligned} J &= (\boldsymbol{H}\boldsymbol{x})^\intercal \boldsymbol{Q}_e \boldsymbol{H}\boldsymbol{x} + \boldsymbol{u}^\intercal \boldsymbol{R}\boldsymbol{u} \\ &= \boldsymbol{x}^\intercal (\boldsymbol{H}^\intercal \boldsymbol{Q}_e \boldsymbol{H})\boldsymbol{x} + \boldsymbol{u}^\intercal \boldsymbol{R}\boldsymbol{u} \end{aligned} \tag{9.19}$$

From Equation 9.19 can a new expression for the weight matrix easily be isolated as shown in Equation 9.20.

$$\boldsymbol{Q} = \boldsymbol{H}^\intercal \boldsymbol{Q}_e \boldsymbol{H} = \begin{bmatrix} -\boldsymbol{H}_s \\ \boldsymbol{H}_r \end{bmatrix} \boldsymbol{Q}_e \begin{bmatrix} -\boldsymbol{H}_s & \boldsymbol{H}_r \end{bmatrix} \tag{9.20}$$

Using the LQR algorithm described in Section 9.1 a combined feedback matrix $\boldsymbol{K}$ can be calculated. The matrix will be a combination of the matrices $\boldsymbol{K}_r$ and $\boldsymbol{K}_s$ as shown in Equation 9.21.

$$\boldsymbol{u} = \begin{bmatrix} \boldsymbol{K}_s & \boldsymbol{K}_r \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_s \\ \boldsymbol{x}_r \end{bmatrix} = \boldsymbol{K}_s\boldsymbol{x}_s + \boldsymbol{K}_r\boldsymbol{x}_r \tag{9.21}$$

The feedback matrices will interact with the quadrotor as previously shown on Figure 9.1, but can also be described with the linear models as in Figure 9.2.

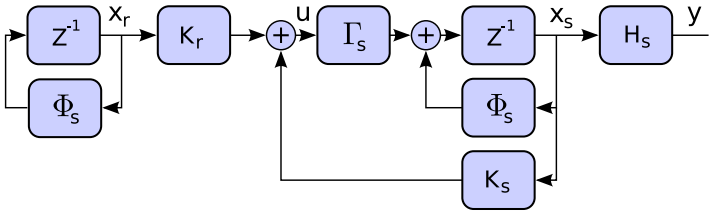

Figure 9.2: Structure of the feedback matrices with the linear models

9.2.1 The Reference Model

The quadrotor is desired to be able to follow a trajectory and not only stay at one point. Therefore the reference is initially modelled as constantly moving reference with a constant velocity, as described in Equation 9.22.

$$\boldsymbol{r}(k) - \boldsymbol{r}(k-1) = \boldsymbol{r}(k-1) - \boldsymbol{r}(k-2) \tag{9.22}$$

This expression can easily be rearranged to become a discrete state-space model.

$$\begin{aligned} \boldsymbol{r}(k) &= 2\boldsymbol{r}(k-1) - \boldsymbol{r}(k-2) \\ \boldsymbol{r}(k+1) &= 2\boldsymbol{r}(k) - \boldsymbol{r}(k-1) \end{aligned} \tag{9.23}$$

By defining the reference state as in Equation 9.24 the state-space model can be described as in Equation 9.25.

$$\boldsymbol{x}_r(k) = \begin{bmatrix} \boldsymbol{r}(k) \\ \boldsymbol{r}(k-1) \end{bmatrix} \tag{9.24}$$

$$\boldsymbol{x}_r(k+1) = \begin{bmatrix} 2\boldsymbol{I} & -\boldsymbol{I} \\ \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \boldsymbol{x}_r(k) \tag{9.25}$$

Where the reference can be described as in Equation 9.26.

$$\boldsymbol{r}(k) = \boldsymbol{H}_r \boldsymbol{x}_r = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \end{bmatrix} \boldsymbol{x}_r \tag{9.26}$$

In practise the reference used must be the state of the reference ($\boldsymbol{x}_r$), so the previous sample must be included. Because the sample rate is much faster then the changes in the reference it is chosen to filter the previous state to avoid the otherwise noisy residual.

A similar model is calculated using a constant position model ($\boldsymbol{x}_r(k) = \boldsymbol{r}(k)$) to compare the effect of the models.

Chapter 10

Robust H_∞ Control

The second type of controller to be derived is the robust H_∞ controller. H_∞ control is based on minimizing a performance channel, in this case, including a disturbance in the form of a reference and sensor noise.

The structure of the system and the controller are illustrated in Figure 10.1.

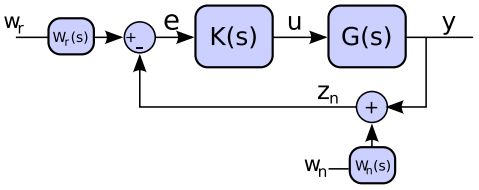

Figure 10.1: The basic configuration for the Robust H_∞ controller

The disturbances can be seen as the reference ($\boldsymbol{w}_r$) and the sensor noise ($\boldsymbol{w}_n$). The frequency content is described using the wights $W_r(s)$ and $W_n(s)$. $W_r(s)$ has a low-pass characteristic and $W_n(s)$ has a high-pass characteristic.
The method chosen to derive the controller can be described in three steps:

1. Convert setup to standard 2×2 block form
2. Calculating the Matrix Inequality describing the H_∞ solution
3. Use a Linear Matrix Inequality (LMI) solver to derive the feed forward and feedback matrices

The standard 2×2 block form, also known as the standard problem in robust control theory, is a form that encapsulates the entire system as a linear system with two inputs and to outputs as illustrated in Figure 10.2.

The channel from $\boldsymbol{w}$ to $\boldsymbol{z}$ is called the performance channel, and the purpose of deriving a controller is to minimize this channel. The general input to the system is $\boldsymbol{u}$ and the measurable output is $\boldsymbol{y}$. The challenge of deriving the standard problem is to incorporate the disturbances and to define the performance channel.

In this computation of the controller, a particular type of solution have been selected. A solution the requires solving a LMI. The steps to do this involves setting up one LMI containing all requirements and solving it

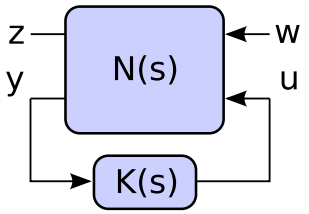

Figure 10.2: The standard problem

using a LMI solver. These last two steps are very general and will only be described shortly.

10.1 The Standard Problem

The standard problem is a way to formulate a more complex system on a standardized form with only a few input and output. When it is written on state-space form it can be described with 8 matrices with the correlation shown in Equation 10.1. The top row is the state propagation and the remaining are the correlation of the general input/output and the performance channel.

$$\begin{bmatrix} \dot{\boldsymbol{x}} \\ \boldsymbol{z} \\ \boldsymbol{y} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A} & \boldsymbol{B_1} & \boldsymbol{B_2} \\ \boldsymbol{C_1} & \boldsymbol{D_{11}} & \boldsymbol{D_{12}} \\ \boldsymbol{C_2} & \boldsymbol{D_{21}} & \boldsymbol{0} \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{w} \\ \boldsymbol{u} \end{bmatrix} \tag{10.1}$$

The state-space formulation is shown on a block diagram form in Figure 10.3.

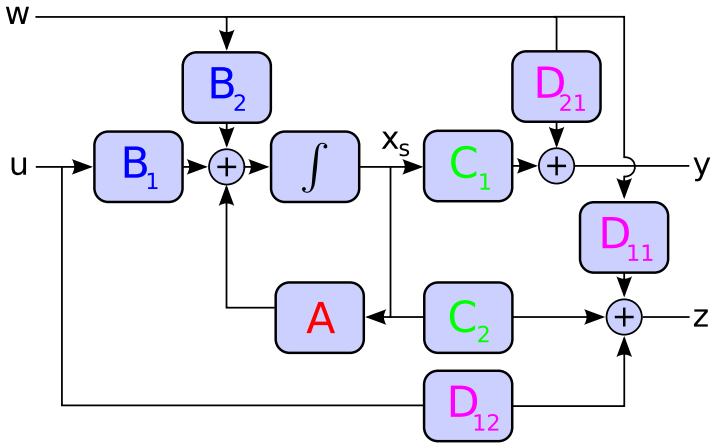

Figure 10.3: The structure of the standard problem

The normal states space formulation can be recognized as the matrices $\boldsymbol{A}$, $\boldsymbol{B_1}$ and $\boldsymbol{C_1}$. The colors symbolizes the function of the matrices. red indicates a feedback matrix, blue an input matrix, green, an output matrix and purple a feed forward matrix.

The weights shown on Figure 10.1 are implemented as state-space filters with each their own matrices propagation, input and output matrices. To include all variables are the input redefined as the following, where $\boldsymbol{x}_s$ is the states of the system, $\boldsymbol{x}_r$ the low-pass weighted reference state, $\boldsymbol{x}_n$ the high-pass weighted sensor noise state, $\boldsymbol{w}_r$ the reference and $\boldsymbol{w}_n$ the sensor noise.

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{x}_s \\ \boldsymbol{x}_r \\ \boldsymbol{x}_n \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} \boldsymbol{w}_r \\ \boldsymbol{w}_n \end{bmatrix} \tag{10.2}$$

With this definition the matrices $\boldsymbol{A}$, $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ can be redefined as in Equation 10.3

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A_s} & \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{A_r} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{A_n} \end{bmatrix} \quad \boldsymbol{B_1} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{B_r} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{B_n} \end{bmatrix} \quad \boldsymbol{B_2} = \begin{bmatrix} \boldsymbol{B_s} \\ \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix} \tag{10.3}$$

The performance channel is based on the residual of the state and the reference as shown in Figure 10.1 and can be described as the output of the system subtracted from the filtered reference ($\boldsymbol{C_r x_r} - \boldsymbol{C'_s x_s}$) as shown in

Equation 10.5. The matrix $\boldsymbol{C_s'}$ is a sub-matrix of $\boldsymbol{C_s}$, but only describing the outputs that is used as a reference as illustrated in Equation 10.4.

$$\boldsymbol{C_s'} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{10.4}$$

A requirement of the general H_∞-solution is for the $\boldsymbol{D_{12}}$ matrix to have full column rank [Toeffner-Clausen et al., 2001]. This is done by adding a factor $\boldsymbol{\epsilon}$. This also serves as a precaution to limit the input since the $\boldsymbol{\epsilon u}$ will be part of the performance channel.

$$\boldsymbol{C_1} = \begin{bmatrix} -\boldsymbol{C_s'} & \boldsymbol{C_r} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \quad \boldsymbol{D_{11}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix} \quad \boldsymbol{D_{12}} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{10.5}$$

The out is described at the measurable system output added with the high pass filtered sensor noise ($\boldsymbol{C_s x_s} + \boldsymbol{C_n x_s} + \boldsymbol{D_n w_n}$) as described in Equation 10.6.

$$\boldsymbol{C_2} = \begin{bmatrix} \boldsymbol{C_s} & \boldsymbol{0} & \boldsymbol{C_n} \end{bmatrix} \quad \boldsymbol{D_{21}} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{D_n} \end{bmatrix} \tag{10.6}$$

10.1.1 Weights as filters

The weights are as previously mentioned implemented as filters. The types of filters chosen are to be first order high-pass and low-pass filters with the structure shown in Equation 10.7 where a denotes a pole, b a zero and k a scalar gain.

$$W_r = \frac{y(s)}{u(s)} = \frac{k}{s+a} \qquad W_n = \frac{y(s)}{u(s)} = \frac{k(s+b)}{s+a} \tag{10.7}$$

The transfer functions are converted to state-space form like in the following example:

$$\begin{aligned} \frac{y(s)}{u(s)} &= \frac{k}{s+a} \\ y(s)(s+a) &= k \cdot u(s) \\ \dot{y} + a \cdot y &= k \cdot u \\ \dot{y} &= -a \cdot y + k \cdot u \end{aligned}$$

When x is defined to $x = y$ the following standard state-space matrices can be used.

$$\boldsymbol{A} = -a \quad \boldsymbol{B} = k \quad \boldsymbol{C} = 1 \quad \boldsymbol{D} = 0 \tag{10.8}$$

For the high-pass case with both a pole and a zero the model can be described with the state-space model in Equation 10.9.

$$\boldsymbol{A} = -a \quad \boldsymbol{B} = b - a \quad \boldsymbol{C} = 1 \quad \boldsymbol{D} = k \tag{10.9}$$

The weight for the reference consists of several low-pass filters since every state of the reference is weighted. The state-space weight for the reference is listed in Equation 10.10.

$$\boldsymbol{A_r} = -\begin{bmatrix} a_1 & 0 & 0 & 0 \\ 0 & a_2 & 0 & 0 \\ 0 & 0 & a_3 & 0 \\ 0 & 0 & 0 & a_4 \end{bmatrix} \quad \boldsymbol{B_r} = \begin{bmatrix} k_1 & 0 & 0 & 0 \\ 0 & k_2 & 0 & 0 \\ 0 & 0 & k_3 & 0 \\ 0 & 0 & 0 & k_4 \end{bmatrix} \quad \boldsymbol{C_r} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10.10}$$

Similar do the state-space noise wight consist of diagonal matrices containing the values from Equation 10.9.

10.2 Matrix Inequalities

Given a standard state-space system $G(s)$ given as

$$\begin{aligned} \dot{\boldsymbol{x}} &= \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{w} \\ \boldsymbol{z} &= \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{w} \end{aligned}$$

the nominal H_∞ condition $\|G(s)\|_\infty < \gamma$ is, according to [Stoustrup, 2010], satisfied if and only if there exists a vector $\boldsymbol{X} = \boldsymbol{X}^* > 0$ such that

$$\boldsymbol{A}^*X + X\boldsymbol{A} + \boldsymbol{C}^*\boldsymbol{C} - (X\boldsymbol{B} + \boldsymbol{C}^*\boldsymbol{D})(\boldsymbol{D}^*\boldsymbol{D} - \gamma^2 I)^{-1}(\boldsymbol{B}^*X + \boldsymbol{D}^*\boldsymbol{C}) < 0 \tag{10.11}$$

This inequality can be transfered to robust control theory as done in [Stoustrup, 2010] and yield the following LMI

$$\begin{bmatrix} \boldsymbol{F}_{11} & \boldsymbol{F}_{12} & \boldsymbol{F}_{13} \\ \boldsymbol{F}_{12}^\intercal & -\boldsymbol{R}_1 & \boldsymbol{0} \\ \boldsymbol{F}_{13}^\intercal & \boldsymbol{0} & -\boldsymbol{I} \end{bmatrix} < 0 \tag{10.12}$$

Where the matrices $\boldsymbol{F}_{11}$, $\boldsymbol{F}_{12}$, $\boldsymbol{F}_{13}$ and $\boldsymbol{R}_1$ is described as in Equations 10.13 to 10.16.

$$\boldsymbol{R}_1 = \boldsymbol{I} - \boldsymbol{D}_{11}^\intercal \boldsymbol{D}_{11} \tag{10.13}$$

$$\boldsymbol{F}_{11} = \boldsymbol{Y}\boldsymbol{A}^\intercal + \boldsymbol{A}\boldsymbol{Y} + \boldsymbol{W}^\intercal \boldsymbol{B_2}^\intercal + \boldsymbol{B_2}\boldsymbol{W} \tag{10.14}$$

$$\boldsymbol{F}_{12} = \boldsymbol{B_1} + \boldsymbol{Y}\boldsymbol{C_1}^\intercal \boldsymbol{D_{11}} + \boldsymbol{W}^\intercal \boldsymbol{D_{12}}^\intercal \boldsymbol{D_{11}} \tag{10.15}$$

$$\boldsymbol{F}_{13} = \boldsymbol{Y}\boldsymbol{C_1}^\intercal + \boldsymbol{W}^\intercal \boldsymbol{D_{12}}^\intercal \tag{10.16}$$

One feedback matrix (including reference and noise feedback) can be calculated as in Equation 10.17

$$\boldsymbol{F} = \boldsymbol{W}\boldsymbol{Y}^{-1} \tag{10.17}$$

10.3 Simulation

To verify that the controller is able to stabilize the quadrotor the controller is tested with the nonlinear model. In Figure 10.4 is the simulation of a step on each axis shown. Since a step in the yaw reference would bring the quadrotor permanently out of the operating point this is disregarded.

All simulations show satisfying results, with a performance on the x and y-axis that indicates a bandwidth of approximately 0.2 Hz. This controller is on purpose tuned this slow to have as low a feedback matrix as possible. This H_∞ controller have been tested on the quadrotor in real flight with an adverse result. The controller was not capable of stabilizing the quadrotor in hover.

The feed forward and feedback controller matrices was investigated to find the cause of this result. The feedback matrix was compared with the one of the LQR and the feedback matrix of the H_∞ controller was significantly higher.

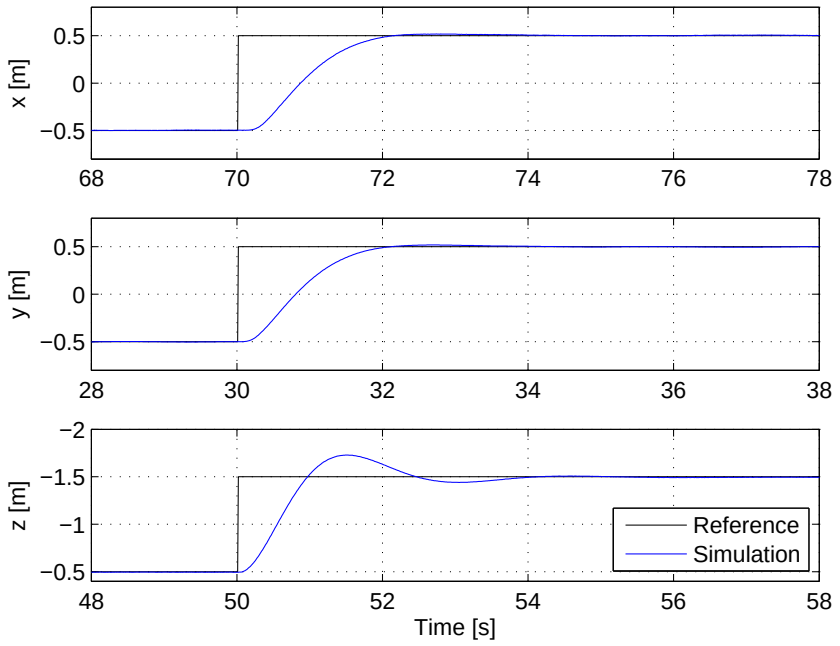

Figure 10.4: Simulation of step response of H_∞ controller

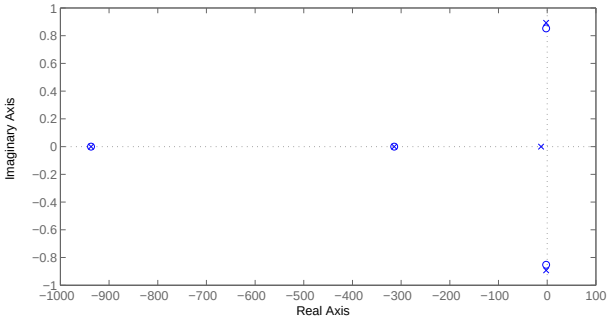

Figure 10.5: Pole/zero map of the closed loop transfer function from reference to position on the y-axis

If the closed loop system poles are analysed an indication for this behaviour can be found. In Figure 10.5 is shown a pole/zero map for the closed loop transfer function from reference to position on the y-axis.

The pole/zero map indicates that the controller dampens much of the effect of the original poles in zero (from the integrators), and only one pole remains to have a dominating effect. The actual movement of the poles, depending on the parameters that can be adjusted for the H_∞ controller, is very difficult to predict. But in the case that all the other poles remains dampened by the controller and the pole original is moved from zero and is following the normal root-locus behaviour the increasing feedback will cause the pole to further increase the bandwidth. If the normal root-locus is followed by all poles and the zeros are not further moved (not very likely), this system is not able to become unstable as further feedback gain is increased.

The reason why this controller becomes unstable in the real system, is most likely that some dynamics are absent in the current model. One place where a dampening have been identified, but not modelled was in Section 3.3 on page 18 where the on-board controller was discussed. The choice of not modelling the dynamics of the controller, motor controller and aerodynamic effect can very well be the cause of the H_∞ controller not being able to stabilize the real quadrotor. The performance of the H_∞ controller is not investigated further, and only the LQR and PID will be evaluated doing real flight in the following chapter.

Chapter 11

Evaluation of Controllers

The developed controllers are tested under various conditions in real flight to evaluate the performance and to identify areas that may be improved. Firstly in Section 11.1 are the controllers tested in hover using various sampling frequencies for the Vicon measurements. This is done to partly evaluate the controller in hover, but also to determine how well the estimator and the controller collaborate. Section 11.2 demonstrates the step response of the closed loop system. In Section 11.3 are the controllers tested with various types of trajectories. In this section it is illustrated how the various structure of the controller affects the performance. In Section 11.4 shortly discusses how the quadrotor would perform doing take-off and landing operations. A short demonstration of hovering in low hight shows how well the quadrotor can be controlled.

The main tests are documented on video, and can be found on the CD in the folder Videos.

11.1 Hover using Various Vicon Sampling Frequencies

To evaluate the performance of the estimator combined with the controllers and to estimate whether a 100 Hz sampling frequency on the Vicon system is a requirement, three different sampling frequencies was tested doing hover with the developed LQR with a constant position reference model. The reference was locked to the position $\begin{bmatrix} 0 & 0 & -1 \end{bmatrix}$ doing the entire flight. Figure 11.1 shows the x-position doing an autonomous flight of 20 seconds.

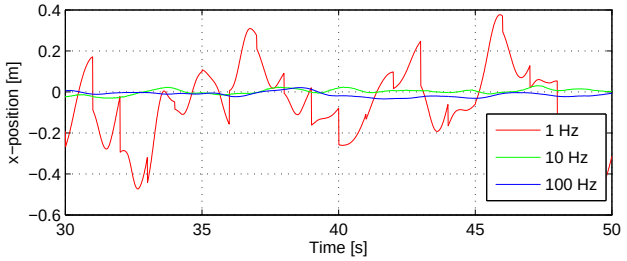

Figure 11.1: Hovering with Vicon sample frequencies of 1 Hz, 10 Hz and 100 Hz

The x-position shown is the result of the estimator, in other words the state that is used by the controller. From the figure a clear difference can be seen in using 1 Hz and 10 Hz sampling frequency. This difference is also illustrated on Figure 11.2, where the figure is shown as if it was seen from the ceiling of the MTLab.

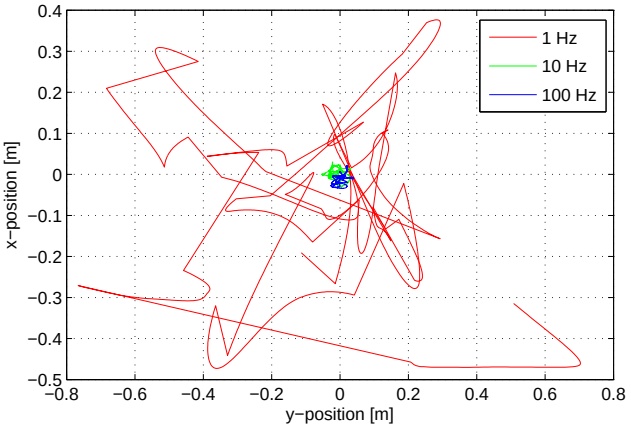

Figure 11.2: Hovering with Vicon sample frequencies of 1 Hz, 10 Hz and 100 Hz

The very low sampling frequency allows the estimator to drift from the actual state, and this is believed to be the cause for this inaccurate flight when only using 1 Hz Vicon measurements. Figure 11.3 shows the estimate and measurements time correlated to better illustrate this belief. In almost every sample, the estimate is shown to be far from the true state, where the furthest is more the 20 cm. This is believed to be an effect of the accelerometers and gyroscopes being scaled with a little error and that the estimator continuously is correcting the acceleration and gyro bias. When using this low of a sampling frequency the update will be unreliable, since the movements of the quadrotor are fast compared to the sampling frequency.

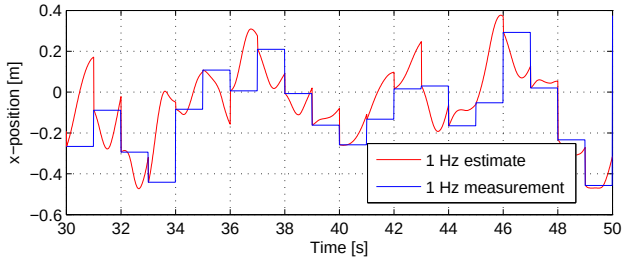

Figure 11.3: Hovering with Vicon sample frequency of 1 Hz. Figure shows how estimates drifts between samples

Also on the Figure 11.2 the accuracy of the controller using 10 Hz and 100 Hz Vicon update frequencies can be seen to be approximately similar. From this it can be concluded that a 100 Hz update frequency is not necessary when using an observer. What sampling frequency that would be enough is not further investigated, but a 10 Hz sampling frequency is used for the remaining flight tests.

11.2 Step Response

On Figure 11.4 are 3 step responses shown, one for each controlled axis. Both a simulation done with the nonlinear model, and an actual measurements are shown. The x and y-axis control indicate a bandwidth of approximately 0.2 Hz. With the LQR this can be done much faster, but an increasing feedback of the orientation makes the orientation of the quadrotor slightly oscillate. This is not further analyzed, but it is assumed that a better dynamics model would help the problem.

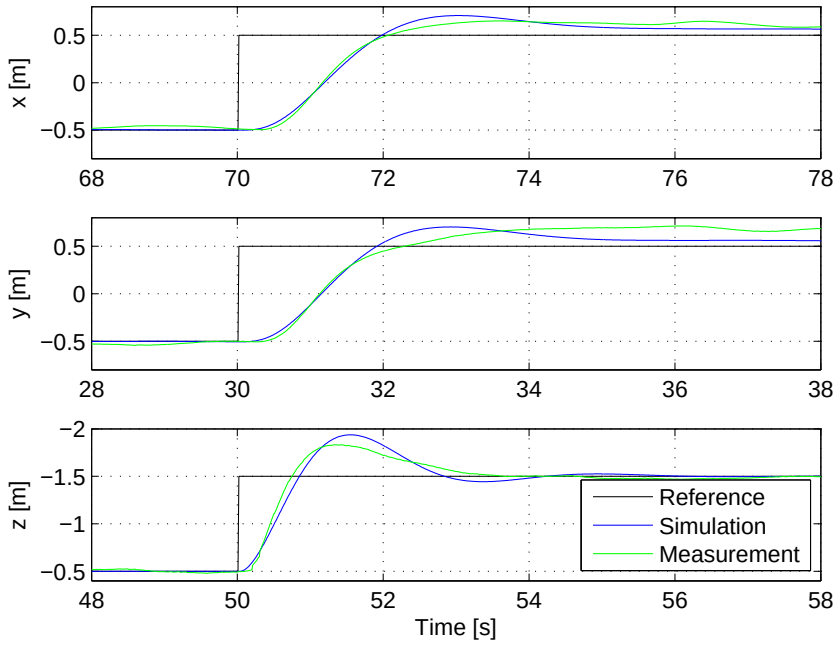

Figure 11.4: Simulation and measurement of step response of LQR controller

11.3 Following a Trajectory

The controllers are tested when following trajectories. The controllers tested are the PID controller, the LQR with a constant position reference and the LQR with a constant velocity reference. All controllers have been tuned to be general applicable, but not in particular to this test scenario. In Figure 11.5 is the first test trajectory shown. The reference is moved clockwise around in a continuous square of 0.75m $\times$ 0.75m with a period time of 20 sec. The trajectories of the controllers shown is the mean of 5 complete round-trips.

All paths of the controllers can be seen to be outside the square. This is believed to an effect of the not so tight tuning of the controller. When the controllers are turned more tight the error becomes less, but the general flight more unstable. The PID and the LQR with constant position reference model can be seen to have a similar flight pattern. They both cut the corners a bit, and this effect is caused by a short lag in time. This is illustrated in Figure 11.6.

The LQR with the constant velocity model can be seen to follow the reference not only in position, but also in

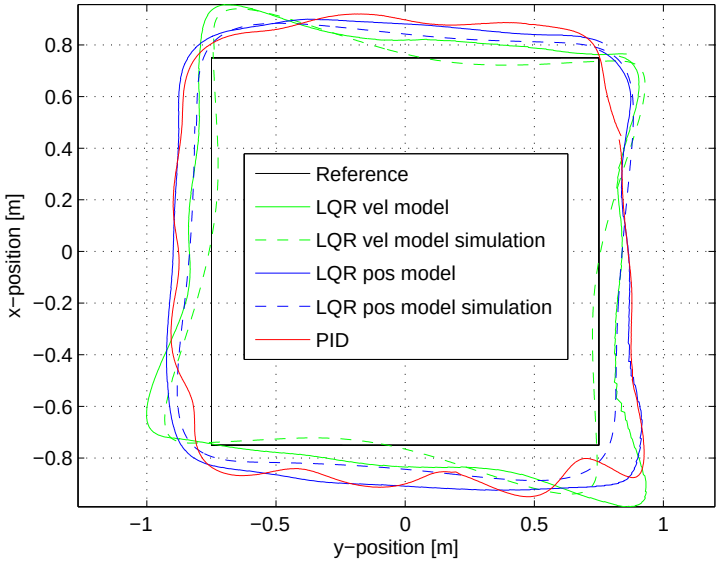

Figure 11.5: Tracking a square reference in x and y-axis with different controllers

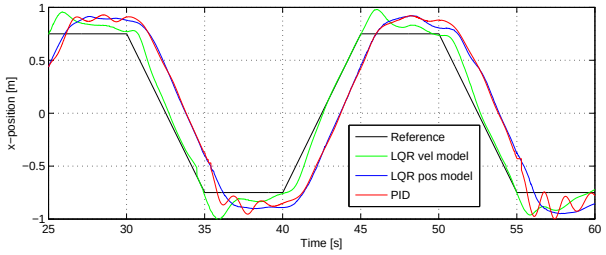

Figure 11.6: Tracking a square reference in x and y-axis with different controllers

velocity. This however results in a overshoot at every corner when the reference suddenly changes direction. The controller however quickly compensates.

A similar trajectory in the y/z-plane is tested as shown in Figure 11.7. Because of the much more direct control the reference hight can be seen to be followed much closer then the y-reference.

However there is still an overshoot from all controllers. It can also be seen more easily how the quadrotor cuts corners with the PID and LQR with the constant position model. The lower right corner is very close to be outside Vicon range which caused the quadrotor to loose position and orientation very shortly when descending.

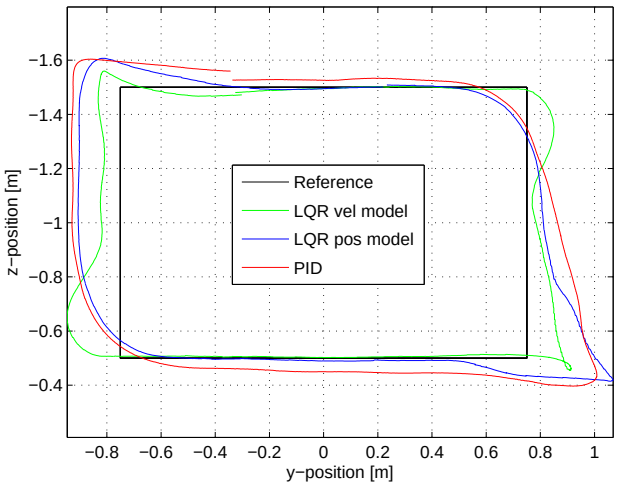

Figure 11.7: Tracking a rectangular reference in z and y-axis with different controllers

11.4 Hovering in Low Height

To demonstrate how the quadrotor behaves in doing landing and take-off a short flight was made in low height. The reference was locked in the center off the room with a height of 2 cm. Figure 11.8 illustrates how the quadrotor moved in the x/y-plane and how the quadrotor was able to hover very close the reference.

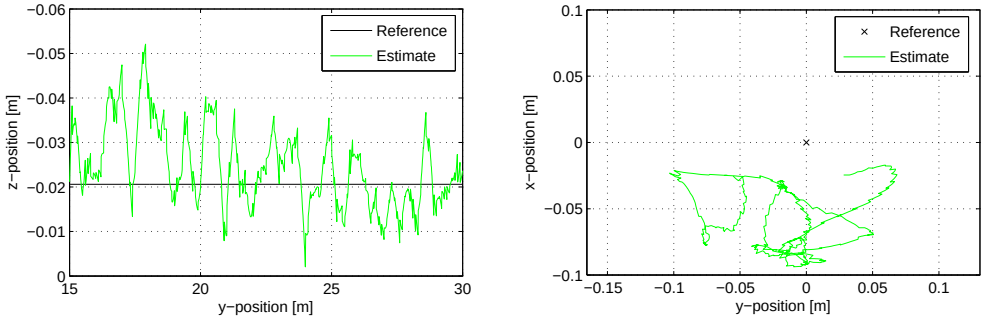

Figure 11.8: Hovering with a hight reference of 2 cm

When the quadrotor hovers this close to the ground, it is affected by the ground effect. The ground effect is the effect of the rotors pushing down the air faster then the air can be transfered away. This is commonly referred to as a high pressure cushion. The cushion gives more lift to the quadrotor, but leaves it more unstable in the x/y-plane.

11.5 Overall Evaluation

The controllers are all able to stabilize the quadrotor, but in following moving trajectories the LQR with a constant velocity model clearly stands out being able to match the position time correlated with the reference.

The overshoot on the test trajectories could, to some extent, be avoided if the test was done with trajectories that were feasible for the quadrotor.
When the quadrotor was flown with a disturbance from the ground effect the LQR seemed capable of keeping the quadrotor stabilized.

Part IV

Epilogue

Table of Contents

Chapter 12

Conclusion

In the first part of this thesis was a general introduction to quadrotors given. How they are able to fly and navigate. The 6 DoF was described mathematically so a prediction of the behaviour could be made depending on the input.
The mathematical model was used to estimate the state of the quadrotor, even when some sensors was failing, and to be the basis for the model based controllers. This chapter will shortly summarize the most important conclusion from the thesis, compare these with the original objective and discuss what future improvement can be made.

The model derived for the X-3D quadrotor encapsulates most of the behaviours of the quadrotor, but in using the model for the model based controllers it was found that the dynamic description was not sufficient. It is estimated that the assumption made regarding the on-board controller being fast enough to ignore the dynamics do not hold. The transfer function including the effects of the on-board controller, the motor controller and the aerodynamics must be modelled. Also models was derived for the on-board sensors, and these were found satisfactory.

The on-board sensors was used for the IMU-based estimator. The result of the estimator being driven by the IMU gave good results. The EKF turned out be an elegant way of handling sensor failures. The quadrotor is capable of flying without the position and orientation sensor for a short while ($\sim$ 1 second) and use various asynchrony sampling frequencies for the sensors. It was however, observed that it becomes very difficult to estimates the constantly changing on-board sensor bias with a update frequency of the position sensor as low as 1 Hz. This resulted in an estimate that drifted much between the measurements.

Two model based controllers was derived for the quadrotor, a LQR with both a constant position and constant velocity reference model and a robust H_∞-controller. The LQR stabilized the quadrotor without problems. The measurements done with the LQR controllers shows similar results both in real flight and in simulations. When using a reference model that modelled constant velocity the quadrotor was capable of following a reference with less lag given the feed forward of the velocity of the reference. This concludes that a fast trajectory can be followed, as long as the quadrotor and the controller allows the necessary velocity. Both controllers flies well as long as the state is kept close to the operating point. If further navigation is needed when flying in high velocity it might become necessary to change the time invariant LQR to a time varying on-line linearizing LQR to be able to move outside the chosen operating point. If only few of the states are changed alternative control strategies can be considered such as gain scheduling or Linear Parameter Varying (LPV) control. As

the frequency of the reference increases a small prediction horizon might contribute to keeping the quadrotor stabilized.

The H_∞ controller derived showed good results in the simulation, but failed to work in real flight. It have been reasoned that the main factor of the effect shown in real flight is based in the missing poles in the model. H_∞ was in general found to be an simple way of incorporating specific types of structure or performance channels, but the calculation is more time consuming then a quadratic minimization.

In general the quadrotor platform combined with the MTLab was found to be a good combination to try out control strategies. Features such as stabilized autonomous landing and take-off should be no problem to perform with the quadrotor as it is able to hover stable in a hight of 2 cm. However before actual fast trajectories can be tested, a larger Vicon range is needed. The range at the given time is limited to 1m $\times$ 1m $\times$ 2m which is to limited. Through test and simulation it have been shown that it is possible to obtain robust flight, even when sensors shortly fail, and be able to follow a desired fast moving trajectory with a minimum of lag in time.

12.1 Future Work

The quadrotor is an interesting platform and will most likely soon become a more commonly sight in peoples everyday life. But before this happens some issues are still to be takes care of. First of all the X-3D quadrotor used in this thesis was only equipped with an on-board accelerometer and a gyroscope. If a quadrotor is to be used outside a Vicon system sensors such as magnetometers and GPS must be used. To further stabilize the quadrotor laser range finders or sonars can be considered to find fixed distances to surrounding objects.

Especially the estimation problem becomes a challenge with flying quadrotors outside the Vicon MX system. If the quadrotor intend to base the orientation on the magnetometer it becomes subject to much magnetically interference when flying indoor or close to objects with electrical signals. This and other estimation challenges needs further to be addressed.

When quadrotors are flying freely subjects as object avoidance and fault detection also becomes increasingly important. These as well as many other application specific areas are already now being researched for future use.

The quadrotor has great potential as an autonomous drone in the future, the stable and relatively harmless platform allows it to fly places where people is present and do task such as video surveillance or industrial inspection.

Bibliography

[Andersen, 2009] Andersen, P. (2009). Optimal control. internet (pdf).

[AscTec, 2009] AscTec (2009). *ResearchPilot Manual*. `http://asctec.de/downloads/researchpilot_manual.pdf`.

[Bak, 1999] Bak, T. (1999). *Spacecraft Attitude Determination - a Magnetometer Approach*. PhD thesis, Aalborg University.

[Bak, 2002] Bak, T. (2002). *Lecture Notes - Modeling of Mechanical Systems*. AAU, 1 edition.

[Bisgaard, 2007] Bisgaard, M. (2007). *Modeling,Estimation,and Control of a Helicopter Slung Load System*. PhD thesis, Aalborg University.

[Bouabdallah, 2007] Bouabdallah, S. (2007). Design and control of quadrotors with application to autonomous flying. Master's thesis, ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE.

[Franklin et al., 2006] Franklin, G. F., Powell, J. D., and Emami-Naeini, A. (2006). *Feedback Control of Dynamic Systems*. Prentice Hall, 5 edition.

[Friis et al., 2009] Friis, J., Nielsen, E., Andersen, R., Boending, J., Jochumsen, A., and Friis, A. (2009). Autonomous landing on a moving platform. Technical report, Aalborg University.

[Grenwal and Andrews, 2008] Grenwal, M. S. and Andrews, A. P. (2008). *Kalman Filtering, Theory and Practice Using MATLAB*. Wiley, third edition. ISBN-13: 978-0-470-17366-4.

[Gurdan et al., 2007] Gurdan, D., Stumpf, J., Achtelik, M., Klaus-Michael, Doth, Hirzinger, G., and Rus, D. (2007). Energy-efficient autonomous four-rotor flying robot. *IEEE*.

[Hughes, 1986] Hughes, P. C. (1986). *Space Attitude Dynamics*. Wiley.

[MIT, 2010] MIT (2010). Uav swarm health management project. `http://vertol.mit.edu`.

[Newton, 1833] Newton, I. (1833). *Philosophiae naturalis principia mathematica*. G. Brookman, 1 edition.

[STARMAC, 2010] STARMAC (2010). `http://hybrid.eecs.berkeley.edu/starmac`.

[Stevens and Lewis, 2003] Stevens, B. L. and Lewis, F. L. (2003). *Aircraft Control and Simulation*. Wiley, 2 edition.

[Stoustrup, 2010] Stoustrup, J. (2010). Robust lecture slides. `http://www.control.aau.dk/~jakob/courses/robust/robust5.pdf`.

[Thrun et al., 2006] Thrun, S., Burgard, W., and Fox, D. (2006). *Probabilistic Robotics*. The MIT Press. ISBN-13: 978-0-262-20162-9.

[Toeffner-Clausen et al., 2001] Toeffner-Clausen, S., Andersen, P., and Stoustrup, J. (2001). Robust control. internet (pdf).

[Valavanis, 2007] Valavanis, K. P. (2007). *Advances in Unmanned Aerial Vehicles*, volume 33 of *Intelligent Systems, Control and Automation: Science and Engineering*. Springer. ISBN: 978-1-4020-6113-4.

[Van Der Merwe et al., 2004] Van Der Merwe, R., Wan, E., and Julier, S. (2004). Sigma-point Kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation. In *Proceedings of the AIAA Guidance, Navigation & Control Conference*. Citeseer.

Part V

Appendices

Table of Contents

Appendix A

The X-3D Interface

The X-3D quadrotor from Ascending Technologies that have been used in all practical experiments was modified with the ResearchPilot firmware developed by Ascending Technologies. This enables serial communication with the sensor board handling the on-board control of the quadrotor. This communication was used to read some of the on-board sensors so the information could be used in the high-level control. The quadrotor was further equipped with a X-bee serial module enabling wireless serial communication. Unfortunately Ascending Technology only supplies the firmware enabling the communication and a short description of the protocol. They do not supply any implemented software to handle the communication on the control PC.

To be able to use the on-board sensors as part of the estimation a interface have been developed and further implemented as a Matlab/Simulink block. The purpose of this appendix is to shortly describes the main classes and the flow of the interfaces so if others intend to modify the interface this would ease their task.

The actual communication protocol can be found in the documentation of the ResearchPilot [AscTec, 2009] and is not discussed further in this appendix expect when it is relevant to the development of the interface.

A.1 Classes

The main classes used in this interface are the following: `Message`, `SerialSocket` and `Serial`. `Serial` is the operating system specific way of sending and receiving bytes on a serial line. `SerialSocket` is a wrapper for the serial communication, allowing complete messages to be send and received with one command. The class `Message` is the container of data that is used doing transitions.

A.1.1 Message

The class `Message` is parent to several sub-classes as illustrated in Figure A.1. The main class includes basic functions such as reading the data and length as well as getting the type of the class. The sub classes represent the three kinds of transfers used from the protocol. `ConfigMsg` is the configuration message determining what data that will be send from the quadrotor. `CommandMsg` is the command send from Matlab/Simulink to the quadrotor with control commands. `DataMsg` is the message from the quadrotor to Matlab/Simulink containing the measurements of the local sensors.

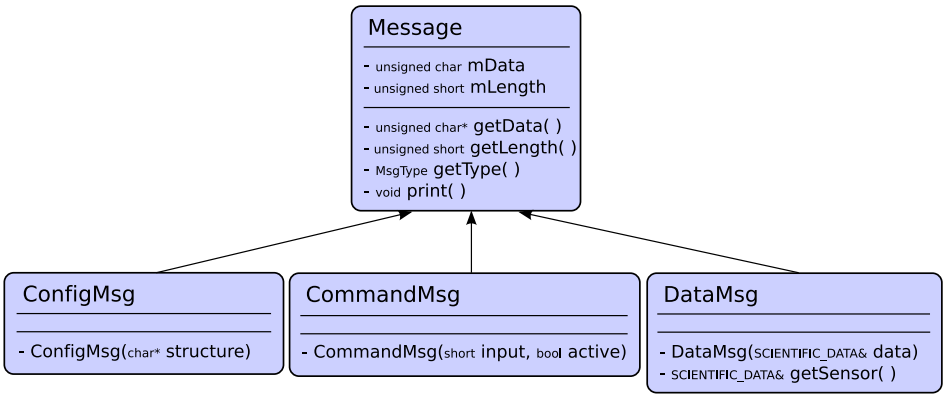

Figure A.1: The `Message` class

A.1.2 SerialSocket

The `SerialSocket` class is the only access to the serial port, therefore this class also contains the only instance of the `Serial` object. When no new messages are received the `SerialSocket` contains an instance of the old message and this is returned. Newly received data is stored in a buffer containing all data from the previous received message and forward. Before a message is returned it is tested whether the CRC matches to make sure no data corruption has occurred.

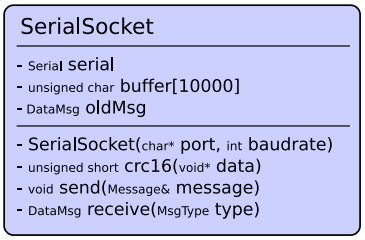

Figure A.2: The `SerialSocket` class

The class diagram for the `SerialSocket` is illustrated in Figure A.2

A.1.3 Serial

As previously mentioned the `Serial` class contains all the information necessary to communicate with the serial connection on the specific operating system. Currently the class only supports the Linux operating system, but would be easy to implement support for others such as Windows or Mac OS X. The class diagram is illustrated in Figure A.3.

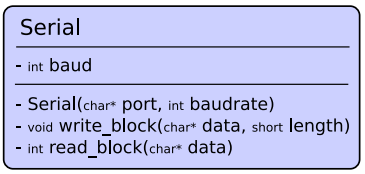

Figure A.3: The `Serial` class

A.2 Procedure

The procedure for communicating with the quadrotor is illustrated in the flow chart in Figure A.4. Initially the structure of the data message configured by choosing the sensors that is to be read. For the experiments done the following sensors have been read.

- The gyroscopes (3 axis)
- The accelerometer (3 axis)
- The pressure sensor
- The manual transmitter commands

The configuration is send as a `ConfigMsg` using the `SerialSocket`. This is done in the initialization of the Simulink simulation. For every sample the control commands from the Simulink send to the quadrotor as a `CommandMsg` and the sensor measurements are received as a `DataMsg`.

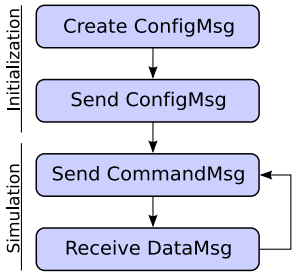

Figure A.4: The flow of the X-3D interface

A.3 Evaluation

The interface have proven stable under both long and short flight. Even though the bandwidth should support the amount of data transfered, the rate of sending commands had to be slowed down to 33 Hz to avoid breaks in the received sensor measurements. Because 33 Hz is still very fast compared to the desired flight trajectory it is estimated that this will have no affect on the final result.

Some times the receiving rate of 100 Hz of the sensor measurements is not kept by the quadrotor and two or three following samples end up being identical. When this was further analyzed it turned out to be minor data errors causing the `SerialSocket` to discard the messages.

Appendix B

Motion Tracking Lab (MTLab)

The MTLab was developed for testing real-time flight of small autonomous helicopters. Original it consists of both the Vicon MX system and a remote controller both linked to Matlab/Simulink. For the experiments done in this thesis however only the Vicon MX system is used in combination with the X-3D interface.

The MTLab is a 5x6 m room with 7 cameras mounted on the walls and connected through gigabit network to the Vicon MX system. The cameras record the position of reflectors mounted on the object. These reflectors reflect the infrared light send out by the cameras. In the ViconIQ software an object is defined on the basis of the position of the reflectors. The position and orientation of this object is then send out on the ordinary network, when a connection is made form a client computer running Matlab/Simulink. The Vicon MX system is currently capable of sending this information with an update rate of 100 Hz.

B.1 Simulink Block

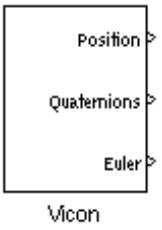

Figure B.1: The MTLab Vicon Simulink block

The Vicon Simulink block supplied by the MTLab outputs position, the orientation in a 3-2-1 Euler rotation, the orientation in a quaternion and the orientation in the direct cosine matrix (earth to body rotation). In this project only the position, the Euler rotation and the quaternions will be used. In Table B.1 is the boundaries and units listed.

Both the Euler rotation and the quaternions are calculated from the Direct Cosine Matrix in the Simulink interface.

Data	symbols	Unit	Boundary
Position	[x]	m	± 1.5
	[y]	m	± 1.5
	[z]	m	0..2
Orientation (Euler)	$[\phi \quad \theta \quad \psi]$	rad	$\pm\pi$
Orientation (quaternion)	$[q_0 \quad q_1 \quad q_2 \quad q_3]$	-	$-1..1$

Table B.1: Data units and boundaries

Appendix C

Measurements Journals

C.1 Polynomial Relation of u_{trust} and the force F_{lift}

The objective of this measurement journal is to find a polynomial relation from the collective input u_{thrust} from the transmitter to the thrust force acting on the quadrotor in near hover flight. To find this, the quadrotor is attached to a box that is more heavy than the quadrotor can lift. Measurements of changes in the weight are then recorded while the input is increasing. The amount of weight loss is proportional to the force induced as long as the acceleration is kept constant. This originates from Newtons second law.

$$F = m \cdot a \tag{C.1}$$

Where
F is the force [N]
m is the mass [kg]
a is the acceleration [m/s^2]

C.1.1 Procedure

The procedure for the experiment is as follows.

1. MTLab is powered on (Vicon information must be available).
2. A control PC is started with Matlab/Simulink.
3. The X-3D quadrotor is equipped with a newly charged battery and attached to a box with a bigger mass than the quadrotor is capable of lifting.
4. The quadrotor and box is then placed on a scale on a elevated (0.75 m) platform as shown in Figure C.1. The rotors needs to be free from the platform to avoid any ground effect.
5. The test signal for thrust is applied while the scale is read off and recorded.
6. When the test signal is done, the motors are returning to idle and the experiment is done.

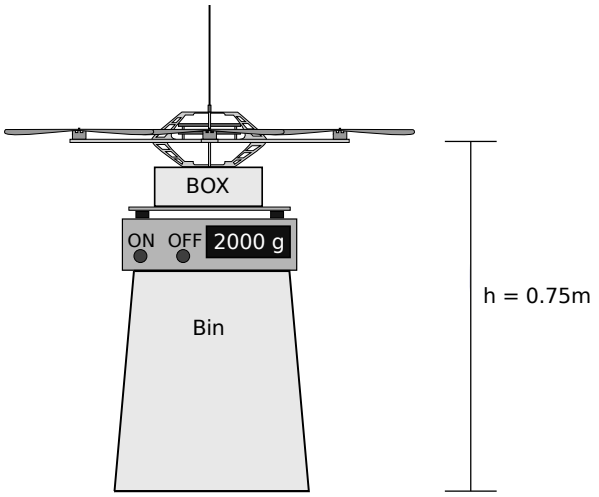

Figure C.1: The set-up used in the thrust test

The scale used is a Rådvad Elevtronic IS075 kitchen scale measuring at a resolution of 1 gram.

C.1.2 Results

The thrust force can then be expressed as in Equation C.2

$$F_t = \Delta m * g \tag{C.2}$$

Were: F_t is the thrust force [N]
Δm is the change in mass [kg]
g is the gravitational acceleration [m/s^2]

The results of the measurements can be seen in Table C.1.

	Measurement 1		Measurement 2	
Initial weight [kg]	2.0040		2.0050	
RC input	m [kg]	Δm [kg]	m [kg]	Δm [kg]
0	1.9060	0.0980	1.9040	0.1010
500	1.8460	0.1580	1.8490	0.1560
1000	1.6950	0.3090	1.7030	0.3020
1500	1.5440	0.4600	1.5400	0.4650
2000	1.3780	0.6260	1.3800	0.6250
2500	1.2400	0.7640	1.2300	0.7750
3000	1.1100	0.8940	1.1100	0.8950
3500	1.0100	0.9940	1.0000	1.0050
4000	0.9800	1.0240	0.9700	1.0350

Table C.1: The results of the force measurement, where the initial weight is the total mass of the quadrotor and the box. Δm is the total mass minus the measured mass at a given input. m is the mass measured by the scale

The calculated force of each input step is shown on Figure C.2 as a continuous graph. To estimate the graph as a polynomial the least squares problem for the following equation is minimised.

$$a_3x^3 + a_2x^2 + a_1x + a_0 \quad = \quad f(x) \tag{C.3}$$

Where $f(x)$ is the result of one measurements and a_0 - a_3 is parameters to a 3rd order polynomial. Figure C.2 also shows the 3rd order polynomial that is fitted to the measured graph. The polynomial can be described by the parameters $a_3 = 2.07 \cdot 10^{-10}$, $a_2 = -1.43 \cdot 10^{-6}$, $a_1 = -1.05 \cdot 10^{-3}$, and $a_0 = -0.90$.

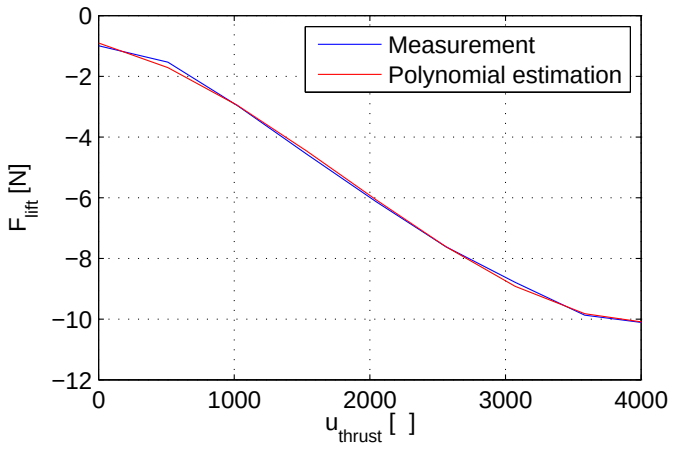

Figure C.2: The measured results plotted together with the F_{lift}

C.2 Determination of Induced Inflow Constant

The objective of this test is to determine the constant that approximates the best mathematical description of the vertical behaviour of the quadrotor. Doing manually flight it has been observed that the quadrotor do not continue to accelerate when a constant input (more than hover) is applied. A more precise description would be that the quadrotor finds a constant vertical velocity with a constant input. With this observation in mind a constant is to be found that results in a constant velocity at a constant input.

The formulation of the total force can be described as in Equation C.4 following the derivation in Section 3.4 in the main thesis.

$$^b\boldsymbol{F} \quad = \quad ^b\boldsymbol{F}_{lift} + m\,^b\boldsymbol{C}_e \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + I_i \begin{bmatrix} 0 \\ 0 \\ v_3 \end{bmatrix} \tag{C.4}$$

The induced inflow constant is found doing steps on the vertical axis. Therefore the quadrotor is assumed parallel with the horizontal plane. This eliminates forces on the x- and y-axis. When a constant velocity is obtained the acceleration must be zero. Therefore the total force must also be zero.

$$0 \quad = \quad F_{lift} + mg + I_i v_3 \tag{C.5}$$

Under these circumstances the induced inflow constant (I_i) can be derived knowing the input force and the velocity.

$$I_i \quad = \quad \frac{-(F_{lift} + mg)}{v_3} \tag{C.6}$$

C.2.1 Procedure

Measurements of the input force and the velocity under the previous described conditions are done in the following way:

1. MTLab is powered on
2. The control PC is started with Matlab Simulink and the x3d_interface.mdl model is opened
3. The quadrotor flown near hover in the center of the room
4. From hover multiple steps on the thrust input is made resulting in the quadrotor slowly moving upward
5. All steps are logged and saved

The input force is calculated using the polynomial approximation derived in Appendix C.1.

$$F_{lift} \quad = \quad f(u_{thrust}) \tag{C.7}$$

C.2.2 Results

A measurement of the vertical steps can be seen in Figure C.3.

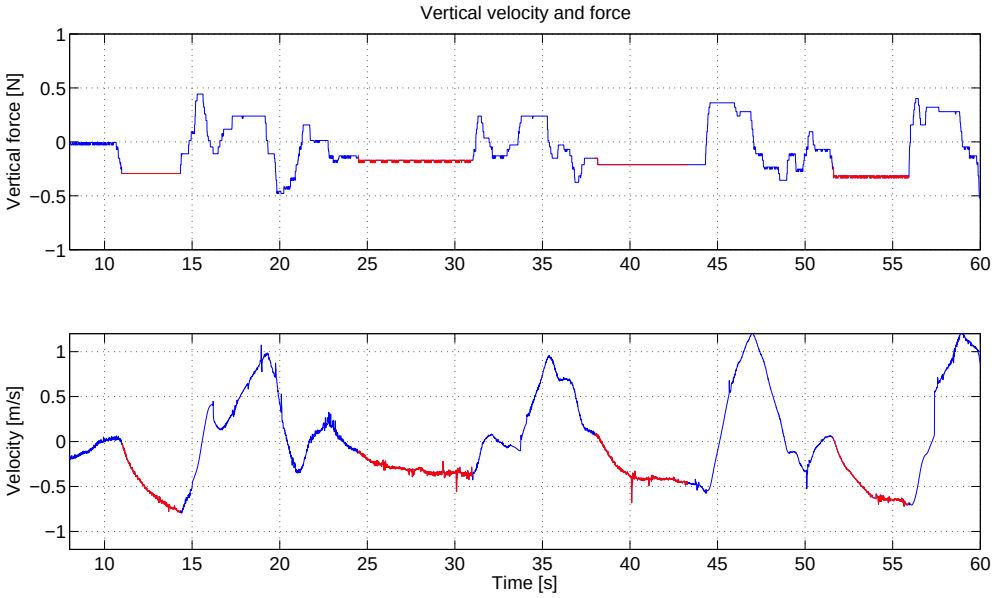

Figure C.3: Thrust steps and vertical velocity

In total 7 measurements have been made. The input force, estimated velocity and the calculated I_i is listed in Table C.2.

The induced inflow constant used in the model is the mean of the calculated values.

$$\bar{I}_i \quad = \quad -0.423 \quad \sigma^2 = 0.004 \tag{C.8}$$

F_{lift} $[N]$	v_3 $[\frac{m}{s}]$	I_i
-0.450	-1.20	-0.375
-0.439	-1.12	-0.392
-0.490	-1.15	-0.426
-0.294	-0.92	-0.320
-0.180	-0.36	-0.500
-0.210	-0.44	-0.477
-0.330	-0.70	-0.471

Table C.2: Result of measurements

C.3 Model Verification

The model verification is made from two measurements. One where large amplitude input signals on roll and pitch is used and one where only small input signals are used. In recording the orientation and velocity the developed EKF is used to minimize noise.

In both measurements the time derivatives are calculated using the Matlab command `diff`. The model output is calculated using the equations listed in the end of the Chapters 3 and 4. The result of the entire $\dot{\boldsymbol{q}}$ measurement is shown in Figure C.4. And the result of the entire ${}^b\dot{\boldsymbol{v}}$ measurement is shown in Figure C.5.

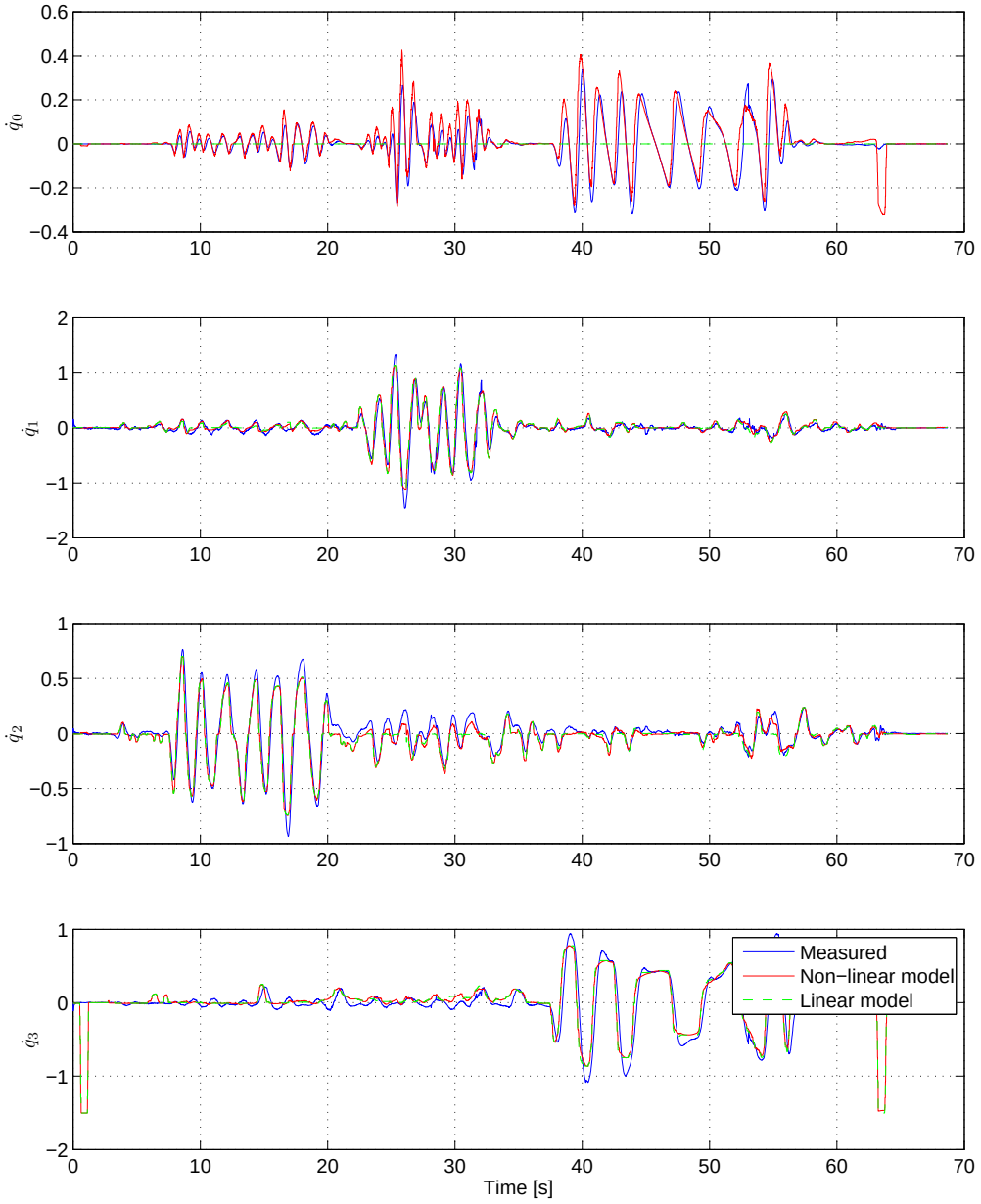

Figure C.4: Modelled and measured $\dot{\boldsymbol{q}}$

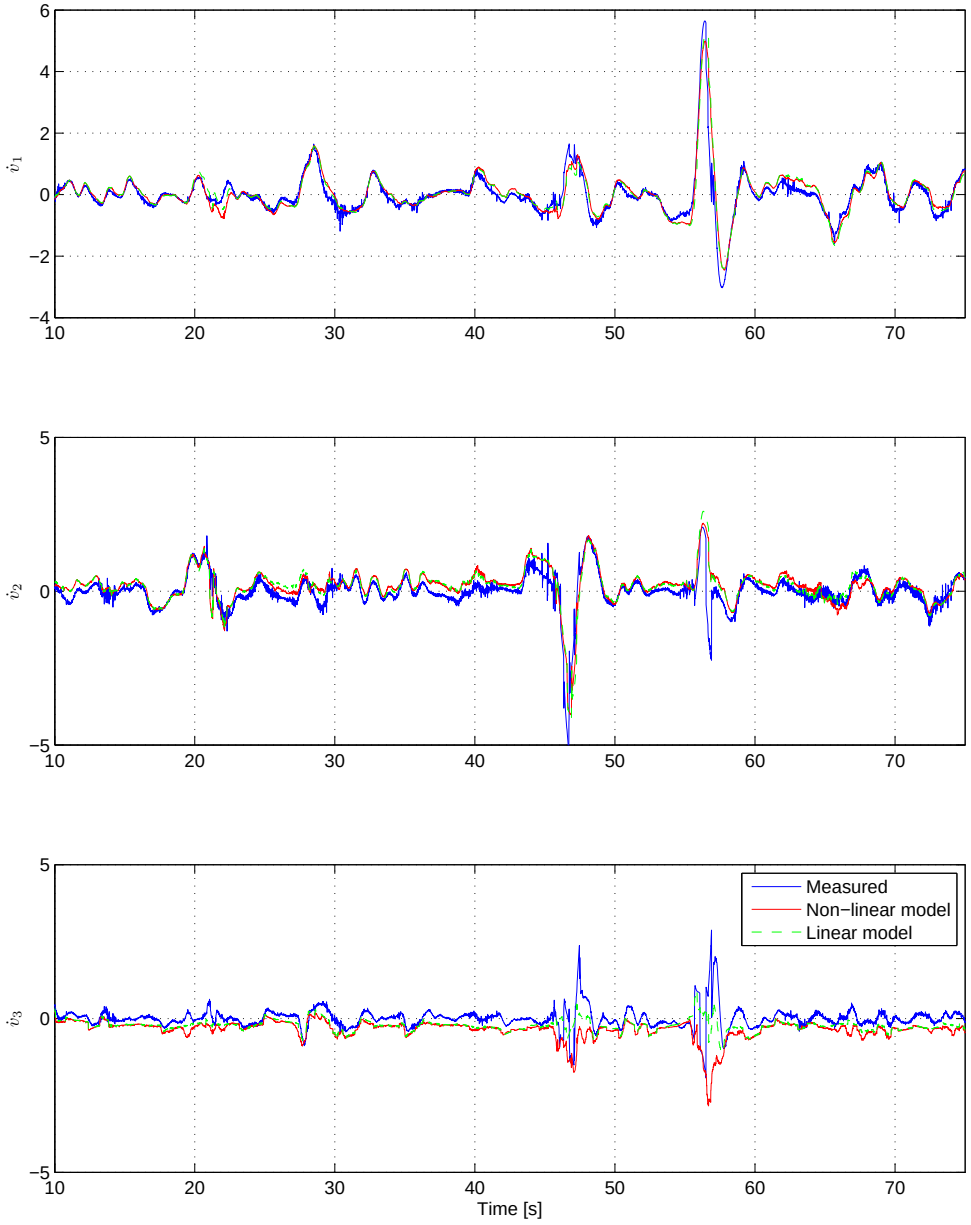

Figure C.5: Modelled and measured $\dot{\boldsymbol{v}}$

Appendix D

CD Contents

The CD attached to the report contains relevant information, Matlab scripts, videos and measurements results. Following is a description of the folders on the CD and their content.

x3d Contains all Matlab files and measurement results used throughout the project

Report A pdf version of the report

Videos Videos of the hover, step and square test made

Pictures Pictures of the quadrotor flying outdoor

Nomenclature

$\boldsymbol{\phi}$	Nonlinear progression model
$\boldsymbol{\Phi}_k^{[1]}$	First order Taylor approximation
$\boldsymbol{\Theta}$	Euler angle parameterization
$\boldsymbol{A}$	System matrix continuous time
$\boldsymbol{B}$	Input matrix continuous time
$\boldsymbol{C}$	Output matrix continuous time
$\boldsymbol{D}$	Feed forward matrix continuous time
$\boldsymbol{F}$	Total force affecting the quad rotor in body frame
$\boldsymbol{H}$	Output matrix discrete time
$\boldsymbol{H}_r$	Reference model output matrix discrete time
$\boldsymbol{K}_r$	Reference compensation (LQR)
$\boldsymbol{K}_s$	Feedback matrix that minimises the performance function (LQR)
$\boldsymbol{K}_{acc}$	Scaling of raw accelerometer measurements to $\frac{m}{sec^2}$
$\boldsymbol{K}_{gyro}$	Scaling of raw gyro measurement to $\frac{rad}{sec}$
$\boldsymbol{N}_r$	Reference compensation (H_∞)
$\boldsymbol{N}_s$	Feedback matrix that minimises the performance function (H_∞)
$\boldsymbol{P}$	Position in earth frame
$\boldsymbol{Q}$	Weight matrix of the states
$\boldsymbol{q}$	Quaternion parameterization
$\boldsymbol{q}_0, \boldsymbol{q}_1, \boldsymbol{q}_2, \boldsymbol{q}_3$	Quaternion parameterization
$\boldsymbol{R}$	Weight matrix of the inputs
$\boldsymbol{u}$	Input vector

$\boldsymbol{x}_r$	Reference state vector
$\boldsymbol{x}_s$	Quadrotor state vector
$\boldsymbol{y}$	Output vector
Γ	Input matrix discrete time
$\hat{\boldsymbol{x}}^+$	A posterior estimate
$\hat{\boldsymbol{x}}^-$	A priori estimate
$\mathcal{J}$	LQR cost function
Ω_{rotor}	Angular velocity of rotors
Φ	System matrix discrete time
ϕ, θ, ψ	Euler angle parameterization
${}^b\boldsymbol{C}_e$	Direct cosine transformation from earth to body frame
${}^b\boldsymbol{F}_g$	Gravitational force acting on the body
${}^b\boldsymbol{F}_i$	Force generated by the induced inflow
${}^b\boldsymbol{F}_{lift}$	Force generated by the rotors
${}^b\boldsymbol{H}_e$	Transformation of rates and angles from earth to body frame
${}^e\boldsymbol{C}_b$	Direct cosine transformation from body to earth frame
${}^e\boldsymbol{H}_b$	Transformation of rates and angles from body to earth frame
g	Gravitational acceleration
I_i	Induced inflow coefficient
m	Mass of quad rotor
$P(\boldsymbol{x})$	A priori probability
$P(\boldsymbol{z})$	Probability of $\boldsymbol{z}$
P_k	Estimate covariance matrix
Q_k	Model covariance matrix
R_k	Sensor covariance matrix
u_ω	Vector containing S_ϕ, S_θ, S_ψ
u_ϕ	Roll input to the on-board controller
u_ψ	Yaw input to the on-board controller
u_θ	Pitch input to the on-board controller
u_{thrust}	Collective input to the on-board controller
x, y, z	Position in earth frame

${}^{b}\boldsymbol{\omega}$	Body rates
${}^{b}\boldsymbol{v}$	Translatory velocity in body frame
${}^{e}\boldsymbol{\omega}$	Euler rates
${}^{e}\boldsymbol{v}$	Translatory velocity of body frame in earth frame