Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2016

# Black-Box Modeling and Attitude Control of a Quadcopter

**Ingrid Kugelberg**

Master of Science Thesis in Electrical Engineering

**Black-Box Modeling and Attitude Control of a Quadcopter**

Ingrid Kugelberg

LiTH-ISY-EX–16/4927–SE

Supervisor: **Jonas Linder**
ISY, Linköping University
**Henrik Jonson**
Saab Dynamics AB

Examiner: **Martin Enqvist**
ISY, Linköping University

*Division of Automatic Control*
*Department of Electrical Engineering*
*Linköping University*
*SE-581 83 Linköping, Sweden*

*To my parents.*

# Abstract

In this thesis, black-box models describing the quadcopter system dynamics for attitude control have been estimated using closed-loop data. A quadcopter is a naturally unstable multiple input multiple output (MIMO) system and is therefore an interesting platform to test and evaluate ideas in system identification and control theory on. The estimated attitude models have been shown to explain the output signals well enough during simulations to properly tune a PID controller for outdoor flight purposes.

With data collected in closed loop during outdoor flights, knowledge about the controller and IMU measurements, three decoupled models have been estimated for the angles and angular rates in roll, pitch and yaw. The models for roll and pitch have been forced to have the same model structure and orders since this reflects the geometry of the quadcopter. The models have been validated by simulating the closed-loop system where they could explain the output signals well.

The estimated models have then been used to design attitude controllers to stabilize the quadcopter around the hovering state. Three PID controllers have been implemented on the quadcopter and evaluated in simulation before being tested during both indoor and outdoor flights. The controllers have been shown to stabilize the quadcopter with good reference tracking. However, the performance of the pitch controller could be improved further as there have been small oscillations present that may indicate a stronger correlation between the roll and pitch channels than assumed.

# Acknowledgments

First of all, I would like to thank my amazing family, for always being there for me, supporting my decisions and for believing in me.

I would also like to thank my examiner Martin Enqvist and my supervisor Jonas Linder from Linköping University for engaging in stimulating discussions, for their support and inspiration.

Finally, I would like to thank Torbjörn Crona, Henrik Jonson, Carl Nordheim and Anders Peterson for giving me the opportunity to write my thesis at Saab Dynamics AB, and for all the support and help.

This thesis is dedicated to my parents.

*Linköping, February 2016*
*Ingrid Kugelberg*

# Contents

# Notation

### Abbreviations

| Abbreviation | Meaning |
|---|---|
| GC | Guidance and control software unit |
| NAV | Navigation software unit |
| PID | Proportional, integral, derivative (controller) |

### Rigid-body Generalized Positions

| Variable | Description |
|---|---|
| $x$ | Longitudinal position relative the inertial frame |
| $y$ | Lateral position relative the inertial frame |
| $z$ | Vertical position relative the inertial frame |
| $\phi$ | Roll angle relative the inertial frame |
| $\theta$ | Pitch angle relative the inertial frame |
| $\psi$ | Yaw angle relative the inertial frame |

### Rigid-body Generalized Velocities

| Variable | Description |
|---|---|
| $u$ | Linear velocity in $x$-axis of the body-fixed frame |
| $v$ | Linear velocity in $y$-axis of the body-fixed frame |
| $w$ | Linear velocity in $z$-axis of the body-fixed frame |
| $p$ | Roll rate, angular velocity around the $x$-axis of the body-fixed frame |
| $q$ | Pitch rate, angular velocity around the $y$-axis of the body-fixed frame |
| $r$ | Yaw rate, angular velocity around the $z$-axis of the body-fixed frame |

# 1

# Introduction

This is the report of a performed at Saab Dynamics in Linköping, Sweden. This chapter aims to give the reader an overview and introduction to this investigated problem. The background, goals and limitations of the thesis are discussed. Furthermore, the methods used to carry out the master's thesis and related work are presented. Finally, a brief overview of the disposition of the report is given.

## 1.1 Background

The use of unmanned areal vehicles (UAVs), or drones, has many interesting applications. Beyond the uses within military applications, UAVs can perform search and rescue operations in hazardous environments, surveillance and inspections of hard to reach places (Waharte and Trigoni, 2010; Nikolic et al., 2013). UAVs can even be used for acrobatic aerial footage in the film making industry or in the future for home delivery of purchased goods (Zhang et al., 2014).

One type of an UAV is a multicopter that is equipped with a control system. This is an agile, flying platform that can be modified in size and capabilities for whatever application the designer has in mind. A multicopter is the designation of a rotorcraft with more than two rotors and a quadcopter is the designation for the special case of four rotors.

## 1.2 Problem Formulation

There has been a lot of research conducted to study quadcopters (Yungao et al., 2011; Gupte et al., 2012; Nagarjuna and Suresh, 2015; Cutler, 2012; Mustapa et al., 2014). Data-based modeling or system identification is an interesting but difficult task and there are many aspects to consider for how it should be done.

A quadcopter is a naturally unstable multiple input multiple output (MIMO) system. Moreover, since the data collection for the system identification will be made outdoors during flight, and not in a testbench, this unstable system will require feedback. The platform will be equipped with initial stabilizing PID controllers to make the quadcopter airworthy.

During data collection it is important to excite the system to obtain data that is informative enough in order to successfully perform system identification (Beltramini et al., 2011). It is assumed that the pilot conducting the experiments is not an experienced quadcopter acrobatics pilot which means that the complexity and approaches of the test flights cannot be designed as arbitrarily as desired.

A common approach when performing system identification is to use a grey-box model, which means to use a physical model of the system but to let some parameters be estimated through experiments (Li, 2014). The second alternative is to use a black-box model which does not take any physical constraints into consideration and only relies on data from test flights to create a predictive mathematical model of the system (Ljung, 2001; Panizza et al., 2015).

After a model of the system has been estimated, a controller should be designed and implemented to replace the PID controllers that were implemented on the platform initially. In order to handle wind and other disturbances, the controller design has to be robust towards these disturbances and towards model errors.

## 1.3 Goals

The main objectives for this master's thesis was to perform system identification of a quadcopter and to design and implement a controller. The final estimated model of the system should be a foundation for future work on the platform. The controller should be designed to control the different dynamics of the quadcopter while in the hovering state and close to hovering.

## 1.4 Limitations

All hardware components and configurations was provided for by Saab Dynamics. There was no time spent working on the hardware in this thesis project.

The data collection was performed outdoors where high-speed flights could be conducted and the necessary precautions could be taken into consideration. Since the wind caused minor problems the data collection was performed when there was as little wind as possible. There was a human pilot conducting the experiments and this imposed a limitation since an arbitrary input could not be used.

## 1.5 Thesis Outline

This thesis is divided into five chapters and is organized as follows. Chapter 2 presents the setup of the quadcopter platform, describing the hardware and software solutions, and a physical model of the quadcopter is also derived there. The next chapter, Chapter 3, treats some system identification theory and the approaches taken and the resulting attitude models.

In Chapter 4, control theory and implementation aspects are presented, followed by the controller design and a discussion regarding the results from both the simulations and the outdoor flights. Finally, in Chapter 5 some final conclusions are presented and future work on the platform is discussed.

# 2

# The Quadcopter Platform

A quadcopter is a naturally unstable platform. By changing the relative speed of the rotors, motion in roll, pitch and yaw can be achieved, creating lateral and longitudinal thrust. Quadcopters are relatively cheap and come in all sizes making them a suitable platform for research.

This chapter treats the quadcopter platform that was used for this thesis. It is explained which hardware it is built of and which software solutions that were used. As previously mentioned in Section 1.4, the platform was provided by Saab Dynamics AB. Therefore, this chapter only aims at giving the reader an overview of the system. Finally, a physical model of the quadcopter will be presented.

## 2.1 Hardware

The complete setup weighs about 680 grams. The specific quadcopter used for this master's thesis can be seen in Figure 2.1. The frame is an *QAV250 Mini FPV*, seen in Figure 2.2, and it is assumed to be rigid. It is designed for hobby users, with plenty of space along the center line to allow for additional equipment to be attached, for example, a camera, and offers plenty of flexibility for the assembler. As can be seen in Figure 2.1, the quadcopter has four propellers and one motor to control each of them. The motors used are the *FX2206-13 2000kv*, which are brushless motors. The propellers used on the platform are the *GEMFAN 5X3*. Each motor is controlled by an electronic speed controller (ESC). The ESC receives a command from the quadcopter control unit and delivers the required power to its motor. The ESCs used on the platform are the *Luminier Mini 20A*. They are small in size but they can deliver high power and are recommended for this frame. To supply the quadcopter with power, the battery *Lumenier 1300 mAh 35c Lipo Battery (XT60)* is used. Its small size allows it to fit on the quadcopter frame and, for this specific setup, the battery offers about 5 minutes of

***Figure 2.1:*** *A photo of the quadcopter platform used. The quadcopter has four propellers. Each one has a motor and an electric speed controller (ESC). The ESC and all other electronics are attached along the lateral center line of the quadcopter. There is also additional free space left on the frame, allowing for an action camera to be attached in the future.*

flight time. The transmitter used here is the *FrSky Taranis X9D Plus*. The unit features a multicolored display and vibration alerts. In line with the industry standard, it has the throttle/rudder stick to the left and the elevator/aileron stick to the right. The receiver used is a *FrSky X8R*, which is very small and light and therefore suitable in this application. Finally, the flight controller consists of an Arduino Micro, a Raspberry Pi, a pulse width modulation (PWM) module and an inertial measurement unit (IMU).

***Figure 2.2:*** *A photo of the quadcopter frame used, the* QAV250 Mini FPV. *The frame is made by Luminier. It is designed to support powerful motors and has an integrated power distribution board for the ESCs and flight electronics.*

## 2.2 Software

In this section, the software solutions used on the platform will be explained. The flight controller is the main software component and the setup is illustrated in Figure 2.3.

The Raspberry Pi has the Guidance and Control unit (GC), the Navigation unit (NAV) and three daemons; the Receiver daemon, the PWM daemon and the Logger daemon. The Receiver daemon stores the latest control command received from the transmitter and, upon request from the GC, sends it to the GC along with a flag. The flag indicates whether or not the control command is new, or the same as the previous command. This allows for good debugging properties of the flight controller.

The communication from the IMU to the Raspberry Pi and the communication from the receiver to the Arduino and to the Raspberry Pi is done via Universal Asynchronous Receiver/Transmitter (UART). All communication on the Raspberry Pi is via the User Data Protocol (UDP) and therefore is the PWM daemon necessary in order to change protocol from UDP to Inter-Integrated Circuit ($I^2C$), which the PWM module needs as input. The communication from the PWM module to the ESCs is done with analog PWM signals. The Logger daemon receives and stores data from the GC.

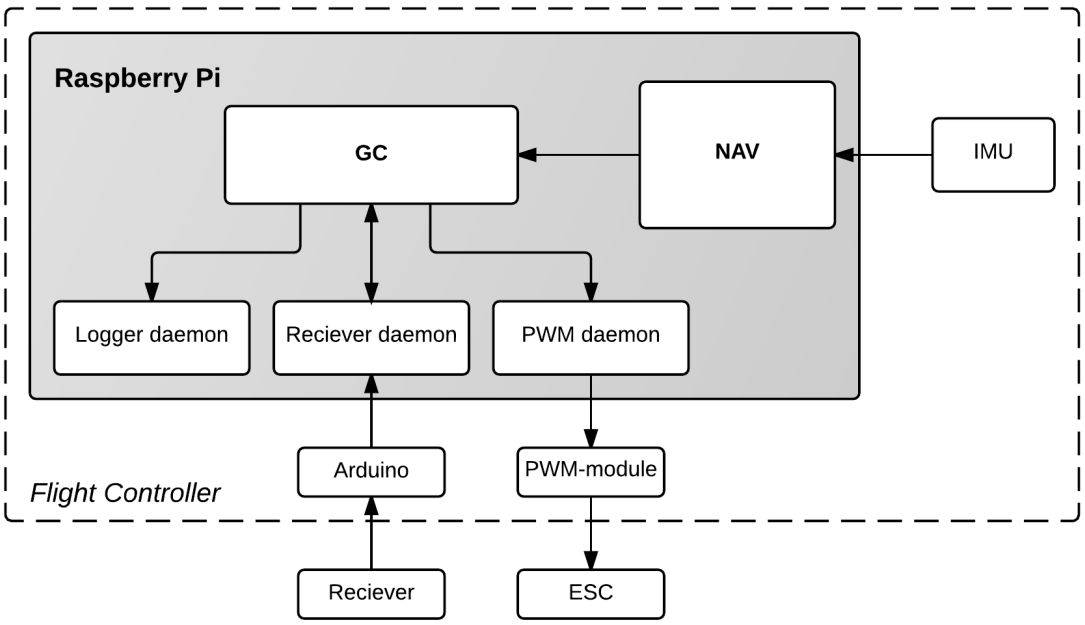



***Figure 2.3:*** *An illustration of how the flight controller is set up. The dashed line shows the flight controller setup and the dark grey box shows the software components on the Raspberry Pi. The components communicating with the flight controller are outside the dashed line.*

***Table 2.1:*** *Stated accuracy of the navigation solution.*

| Navigation Solution Accuracy | |
|---|---|
| Roll Angle | 0.1 deg |
| Pitch Angle | 0.1 deg |
| Yaw Angle | 1 deg |
| Angular Velocity | 5.7 deg/s |

### 2.2.1 Navigation

The Navigation unit (NAV) receives inputs from the IMU and sends out navigation information to the GC. The inputs are processed on the Raspberry Pi and the software is based on a discrete Extended Kalman Filter (EKF) for state estimation. The precision of the NAV can be seen in Table 2.1. The NAV is provided for by Saab Dynamics.

### 2.2.2 Guidance and Control

The Guidance and Control unit (GC) is the main unit of the flight controller and one of the most important parts of the quadcopter. The GC receives navigation information from the NAV and a steering input from the receiver. The GC uses this information to compute a controller output for each of the four motors according to some control algorithm. Initially, the GC will use stabilizing PID controllers to make the quadcopter airworthy but these will later be replaced with controllers designed using the estimated models.

One iteration of the flight controller loop will start when the GC receives a message from the NAV. The process is illustrated in Figure 2.4. The goal is to minimize the controller latency as much possibly. Therefore, the GC will then compute a controller output, according to the information stored in the Receiver daemon, and send the controller output to the PWM daemon. These two steps are the critical and determines the latency of the controller. Once these steps are finished, data will be sent to the Logger daemon. The iteration will end with the GC reading from the Receiver daemon, instead of doing this prior to the other steps. Finally, the GC stays idle for some time. However, this results in a small delay of the pilot's commands of about 10 ms but it was considered to be unnoticeable for the pilot.

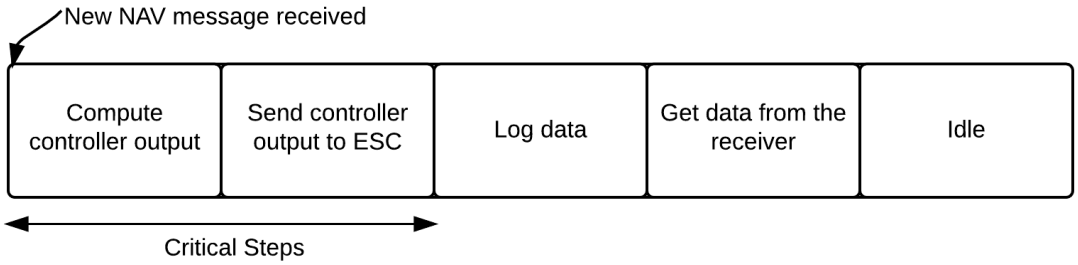



***Figure 2.4:*** *An illustration of the flow of the GC. The process begins when a new NAV message is received. The time delay of the controller is dependent on the first two steps of the sprint, when the controller computes and sends the control output. The data is then logged, a new steering input is fetched from the receiver and finally, the GC stays idle.*

## 2.3 Quadcopter Modeling

In this section, a model of a quadcopter is presented. The model

$$\dot{\boldsymbol{\eta}} = J(\boldsymbol{\eta})\boldsymbol{\nu} \tag{2.1}$$

$$M_{RB}\dot{\boldsymbol{\nu}} + C_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{2.2}$$

from Thornton and Marion (2004), presents a physical model for a rigid-body, where the vector

$$\boldsymbol{\eta} = \begin{bmatrix} x & y & z & \phi & \theta & \psi \end{bmatrix}^T \tag{2.3}$$

is the position and orientation of the quadcopter in the inertial frame and

$$\boldsymbol{\nu} = \begin{bmatrix} u & v & w & p & q & r \end{bmatrix}^T \tag{2.4}$$

is the linear velocities and angular velocities of the quadcopter in the body-fixed frame, and

$$J(\boldsymbol{\eta}) = \begin{bmatrix} R(\boldsymbol{\eta}) & 0_{3\times3} \\ 0_{3\times3} & T(\boldsymbol{\eta}) \end{bmatrix} \tag{2.5}$$

is a transformation matrix of the orientation and position of the body frame with respect to the inertial frame. Finally, $M_{RB}$ is the rigid-body inertia matrix, $C_{RB}(\boldsymbol{\nu})$ represents the centripetal and Coriolis terms and $\tau$ is the generalized forces and torques, all of which are expressed in the body-fixed frame.

The matrix in (2.5) is a transformation matrix with the rotation matrix

$$R(\boldsymbol{\eta}) = \begin{bmatrix} c_\psi c_\theta & -s_\psi c_\phi + c_\psi s_\theta s_\phi & s_\psi s_\phi + c_\psi c_\phi s_\theta \\ s_\psi c_\theta & c_\psi c_\phi + s_\psi s_\theta s_\phi & -c_\psi s_\phi + s_\psi c_\phi s_\theta \\ -s_\theta & c_\theta s_\phi & c_\theta c_\phi \end{bmatrix} \tag{2.6}$$

which describes the relation between the linear velocities in the body-fixed frame and the linear velocities in the inertial frame, and a transformation matrix

$$T(\boldsymbol{\eta}) = \begin{bmatrix} 1 & s_\phi t_\theta & c_\phi t_\theta \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi / c_\theta & c_\phi / c_\theta \end{bmatrix} \tag{2.7}$$

which describes the relation between the angular velocities in the body-frame and angular velocities the inertial frame, where $0_{x \times y}$ is an $x$ by $y$ matrix with all zero elements, $s_x = \sin(x)$, $c_x = \cos(x)$ and $t_x = \tan(x)$. The matrices (2.6) and (2.7) have been created by multiplying the principal rotation matrices for the $z$, $y$ and $x$ axes in that order.

Since the purpose of the position models is to control the quadcopter around a hovering state, the models can be linearized around the hovering state, meaning $\phi \approx \theta \approx 0$. By linearizing (2.6) and (2.7), the resulting matrices are

$$R(\boldsymbol{\eta}) = \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

and

$$T(\boldsymbol{\eta}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

The resulting kinematics relation between the linear and angular velocities of the body-fixed frame and the inertial frame is described by

$$\dot{\boldsymbol{\eta}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} u \cos\psi - v \sin\psi \\ u \sin\psi + v \cos\psi \\ w \\ p \\ q \\ r \end{bmatrix} \tag{2.10}$$

Next, let us move on to the kinetic equations. If the center of mass is assumed to be centered at the origin of the body frame and the body has rotational symmetry around the center of mass, the resulting rigid-body inertia matrix is

$$M_{RB} = \begin{bmatrix} mI_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I \end{bmatrix} = \begin{bmatrix} m & 0 & 0 & 0 & 0 & 0 \\ 0 & m & 0 & 0 & 0 & 0 \\ 0 & 0 & m & 0 & 0 & 0 \\ 0 & 0 & 0 & I_{xx} & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{yy} & 0 \\ 0 & 0 & 0 & 0 & 0 & I_{zz} \end{bmatrix} \tag{2.11}$$

where $m$ is the total mass of the quadcopter and $I_{xx}$, $I_{yy}$ and $I_{zz}$ are the mass moments of inertia.

The centripetal and Coriolis terms are represented by the matrix

$$C_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \begin{bmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & 0 & 0 & 0 & I_{zz}r & -I_{yy}q \\ 0 & 0 & 0 & -I_{zz}r & 0 & I_{xx}p \\ 0 & 0 & 0 & I_{yy}q & -I_{xx}p & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{2.12}$$

In the hovering case, it can be assumed that all the linear velocities and all the angular velocities, except for $r$, are equal to zero. By linearizing (2.12) around $u \approx v \approx w \approx p \approx q \approx 0$, the resulting matrix is

$$\bar{C}_{RB}(\boldsymbol{\nu}) = \begin{bmatrix} 0 & -mr & 0 & 0 & 0 & 0 \\ mr & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & (I_{zz} - I_{yy})r & 0 \\ 0 & 0 & 0 & (I_{xx} - I_{zz})r & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{2.13}$$

which replaces (2.12) in (2.2).

The generalized forces $\tau$ can be divided into three components

$$\tau = \tau_{gravitational} + \tau_{damping} + \tau_{actuators} = G(\boldsymbol{\eta}) + D(\boldsymbol{\nu}) + \boldsymbol{\tau}_c(\boldsymbol{u}) \tag{2.14}$$

where $G(\boldsymbol{\eta})$ is the gravitational component, $D(\boldsymbol{\nu})$ is the damping component, $\boldsymbol{\tau}_c(\boldsymbol{u})$ is the forces generated by the actuators and $\boldsymbol{u}$ is the input vector to the motors. The gravitational component is acting in the inertial $z$-direction. By translating the force to the body frame, the resulting force is described by

$$G(\boldsymbol{\eta}) = \begin{bmatrix} -mg\sin\theta \\ mg\cos\theta\sin\phi \\ mg\cos\theta\cos\phi \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.15}$$

Since the hovering case is studied here, the angles $\phi$ and $\theta$ are small and the small angle approximation can be used, where $\sin x \approx x$ and $\cos x \approx 1$. This gives

$$G(\boldsymbol{\eta}) = \begin{bmatrix} -mg\theta \\ mg\phi \\ mg \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.16}$$

The damping component is the linear matrix

$$D(\boldsymbol{\nu}) = D_0\boldsymbol{\nu} = \begin{bmatrix} D_{0,u}u \\ D_{0,v}v \\ D_{0,w}w \\ D_{0,p}p \\ D_{0,q}q \\ D_{0,r}r \end{bmatrix} \tag{2.17}$$

where $D_0$ are the linear damping coefficients. The forces and torques generated by the actuators are assumed to be linear in the input, hence

$$\tau_c(\boldsymbol{u}) = K\boldsymbol{u} = \begin{bmatrix} 0 \\ 0 \\ K_{\text{throttle}}u_{\text{throttle}} \\ K_{\text{roll}}u_{\text{roll}} \\ K_{\text{pitch}}u_{\text{pitch}} \\ K_{\text{yaw}}u_{\text{yaw}} \end{bmatrix} \tag{2.18}$$

where $K_{\text{throttle}}$ is the coefficient of the force affecting the velocity $w$ is the z-direction and $K_{\text{roll}}$, $K_{\text{pitch}}$ and $K_{\text{yaw}}$ are the coefficients of the torques about the body-fixed coordinate axes. The resulting kinetic equations are

$$\begin{bmatrix} \dot{u} \\ \dot{v} \\ \dot{w} \\ \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} (-mrv - mg\theta)/m \\ (mru + mg\phi)/m \\ (mg + K_{\text{throttle}}u_{\text{throttle}})/m \\ ((I_{zz} - I_{yy})qr + D_{0,p}p + K_{\text{roll}}u_{\text{roll}})/I_{xx} \\ ((I_{xx} - I_{zz})pr + D_{0,q}q + K_{\text{pitch}}u_{\text{pitch}})/I_{yy} \\ (D_{0,r}r + K_{\text{yaw}}u_{\text{yaw}})/I_{zz} \end{bmatrix} \tag{2.19}$$

The complete system dynamics of the quadcopter is given by (2.19) and (2.10).

The four motors on the quadcopter are set up as in Figure 2.5. The controller outputs, i.e. $u_{\text{throttle}}$, $u_{\text{roll}}$, $u_{\text{pitch}}$ and $u_{\text{yaw}}$, are linear combinations of the four motor signals according to

$$\begin{bmatrix} u_{\text{throttle}} \\ u_{\text{roll}} \\ u_{\text{pitch}} \\ u_{\text{yaw}} \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{bmatrix}\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} \tag{2.20}$$

where $M_1$, $M_2$, $M_3$ and $M_4$ are the motor signals. This gives

$$\begin{bmatrix} M_1 \\ M_2 \\ M_3 \\ M_4 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix}\begin{bmatrix} u_{\text{throttle}} \\ u_{\text{roll}} \\ u_{\text{pitch}} \\ u_{\text{yaw}} \end{bmatrix} \tag{2.21}$$

which is how the controller outputs are mapped to the motors. The motor signals are in a range of 0 to 1, which corresponds to the minimum and maximum signal power that the ESCs can receive.

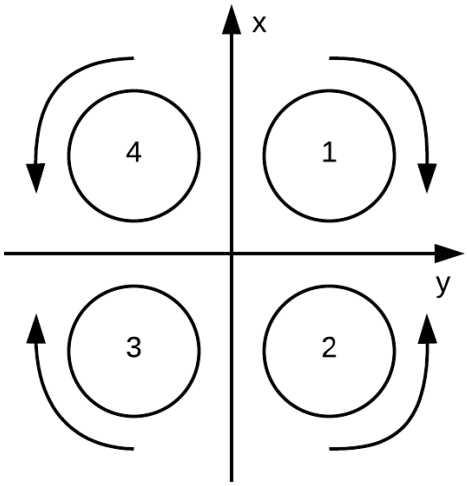



***Figure 2.5:*** *An illustration of the rotational directions of the four motors on the quadcopter seen from above. Motors* 1 *and* 3 *are rotating clockwise and motors* 2 *and* 4 *are rotating counterclockwise. The x-axis is pointing forward.*

# 3

# Model Estimation

The field of system identification concerns how to estimate a model of a system based on observed input-output data and, in some cases, prior knowledge about the system. Two types of models are common when talking about system identification. A grey-box model is based on a physical model of the system, for instance, the quadcopter model described in Section 2.3. But since not all of the system characteristics are entirely known, some parameters of the model are estimated from experimental data. The second type of model is a black-box model. Unlike the grey-box model, the black-box model requires no prior modeling of the system. This is a purely mathematical model and it can be estimated with many different approaches. This thesis will only cover black-box models.

First, some system identification theory will be covered, as well as the chosen method, and the difficulties with closed-loop estimation will be discussed. Finally, the data collection will be explained and the estimated models for attitude will be presented and discussed. The methods used for system identification in this thesis are based on Glad and Ljung (1995) and Ljung (1999).

## 3.1 System Identification

To estimate a model of a system, experimental data in the form of datasets are needed. A dataset $Z_N$ is assumed to consist of $N$ data points where the input and the output are measured and possibly some other signal. $Z_N$ is defined as

$$Z_N = (y(k), u(k), o(k))_{k=1}^{N} \tag{3.1}$$

where $y(k)$ is the output, $u(k)$ is the input and $o(k)$ contains all other measured signals.

A model of the system can be estimated using a linear model structure, such

as the ARX model structure which is described by

$$A(q)y(k) = B(q)u(k) + e(k) \tag{3.2}$$

where $q$ is the forward shift operator, i.e. $qy(k) = y(k+1)$ and $e(k)$ is the disturbances. The polynomials, $A(q)$ and $B(q)$, are defined as

$$\begin{aligned} A(q) &= 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a} \\ B(q) &= b_1 q^{-1} + ... + b_{n_b} q^{-n_b} \end{aligned} \tag{3.3}$$

when the input delay $n_k$ is 1. The ARX model is parametrized by

$$\theta = \begin{bmatrix} a_1 & a_2 & \dots & a_{n_a} & b_1 & \dots & b_{n_b} \end{bmatrix} \tag{3.4}$$

i.e. the parameters of (3.3), and will have different properties depending on the values of the parameters.

The ARX model is one of the simpler models that incorporates the input signal into the model. A disadvantage with this model is that the noise model given by $H(q) = 1/A(q)$ shares the dynamics with the model of the system given by $G(q) = B(q)/A(q)$. This means that $A(q)$ has to be able to describe both some of the input and the disturbance dynamics. In order to successfully do so, the orders of $A(q)$ and $B(q)$ may have to be increased.

The predictor, $\hat{y}(k|\theta)$, is a function of the present and previous inputs and the previous outputs. In general, the ARX model can be represented by the one-step-ahead predictor

$$\hat{y}(k|\theta) = B(q)u(k) + (1 - A(q))y(k) \tag{3.5}$$

Another linear model structure is the ARMAX model, described by

$$A(q)y(k) = B(q)u(k) + C(q)e(k) \tag{3.6}$$

where the signals are defined analogically to the ARX model structure. The polynomials, $A(q)$ and $B(q)$, are defined according to (3.3) and $C(q)$ is defined as

$$C(q) = 1 + c_1 q^{-1} + ... + c_{n_c} q^{-n_c}$$

In general, the ARMAX model structure can be represented by the one-step-ahead predictor

$$C(q)\hat{y}(k|\theta) = B(q)u(k) + (C(q) - A(q))y(k) \tag{3.7}$$

with the parameter vector

$$\theta = \begin{bmatrix} a_1 & \dots & a_{n_a} & b_1 & \dots & b_{n_b} & c_1 & \dots & c_{n_c} \end{bmatrix} \tag{3.8}$$

The advantage with the ARMAX model structure, compared to the ARX model structure, is that it has a more flexible noise model. The disadvantage is that the predictor is nonlinear in the parameters, even though the model itself is linear from input to output.

The model is fitted to the data by estimating the parameters according to the criterion

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}}\, V_N(\theta, Z_N) \tag{3.9}$$

where $V_N(\theta, Z_N)$ is a cost function that depends on the predictor $\hat{y}(k|\theta)$ and the dataset $Z_N$. The cost function can, for instance, be chosen by squaring the prediction error

$$\epsilon(k) = (y(k) - \hat{y}(k|\theta))^T Q(y(k) - \hat{y}(k|\theta)) \tag{3.10}$$

where $Q$ is a positive definite weighting matrix and summing over $k$.

The performance of a model can be evaluated by simulating the open-loop system. However, if the open-loop system is unstable, the closed-loop system can be simulated instead. By using the reference in the validation dataset as the input and the same controller as during the data collection, the simulated inputs and outputs can be compared to the measured data in a validation dataset. To get a measure of how good the fit to the real data is, one can study the goodness of the fit by computing the normalized root mean square error as

$$\text{fit} = 1 - \frac{\sqrt{\sum_{k=1}^{N} (y(k) - \hat{y}(k))^2}}{\sqrt{\sum_{k=1}^{N} \left( y(k) - \frac{1}{N} \sum_{k=1}^{N} y(k) \right)^2}} \tag{3.11}$$

The estimated model is considered to describe the data well when the fit to a validation dataset is high. Therefore, if one model has a higher fit to the validation dataset than another model, the first model is considered to describe the real system better.

The constants $n_a$, $n_b$, $n_c$ and $n_k$, i.e. the model orders, are chosen with cross-validation using the validation dataset. If the model would be validated against the same dataset used to estimate the model, the fit would increase as the model complexity increases and this is called over-fitting. The extra complexity added to the model increases the fit by adjusting to the specific disturbances and other unwanted signals. To be certain that the model describes the system dynamics, and not the specific disturbances present at the data collection, a separate dataset is needed for validation.

## 3.2 Closed-loop System Identification

The estimation has to be performed using data collected during closed-loop operation since the quadcopter is an inherently unstable system. Closed-loop identification is typically more difficult compared to the open-loop case due to the correlation between the noise and the process inputs. A few of the difficulties with closed-loop estimation will be discussed in this section.

The main difficulty with closed-loop identification is that the input $u(t)$ will depend upon the output $y(t)$ which is a function of the disturbance $e(t)$. Hence,

there is a dependence between the input and the disturbance. This correlation can lead to a bias in the parameter estimates. However, the bias can be kept small if the noise model is flexible enough or if the signal-to-noise ratio is high (Forssell and Ljung, 1999).

There exist several approaches to estimate models from closed-loop data. One of these methods is the direct method. In the direct method, a prediction-error system identification method is applied directly as if there was no correlation between the input $u(t)$ and the disturbance $e(t)$. It can be seen as the natural approach to closed-loop identification but it requires the noise-model to be sufficiently rich in order to get consistent estimates. The direct method requires no knowledge about the feedback of the system, no special algorithms or software are needed and unstable systems can be handled as long as the output predictor is stable. Since this thesis is focused on estimation of a model of a quadcopter and not the system identification methods, the direct method will be used straightforwardly.

## 3.3 Data Collection

The data collection was performed outdoors in conditions where the wind was not too strong but noticeable. Since the data had to be collected during flight, some sort of controller had to be implemented initially in order for the pilot to be able to manually control the quadcopter without crashing.

Three PID controllers were implemented on the quadcopter. Two PD controllers in roll and pitch and a P controller in yaw. These controllers were tuned to have as weak feedback as possible with low values for the controller parameters $K_P$ and $K_D$. The data was then collected in a couple of shorter flights, each lasting about 20-60 s with a sampling time of $0.01s$.

## 3.4 Estimated Attitude Models

In this section, the estimated attitude models, i.e. models for the angles and angular rates, will be presented. An estimated model with three degrees of freedom is needed to synthesize controllers for the orientation of the quadcopter. Since the angles were assumed to be small it is assumed that the model can be separated into three decoupled models. The models are estimates of the relationship between the controller output and the angle and the angular rate. The models were estimated using the System Identification Toolbox in Matlab (Ljung, 2014). Section 3.4.1 describes the dynamics of the quadcopter in roll and pitch and Section 3.4.2 describes the dynamics of the quadcopter in yaw.

In order to estimate models that more accurately can predict the future outputs, the estimation dataset was downsampled. The problem with estimating a model from data with a very high sampling frequency is that the poles of the discrete system

$$e^{\lambda_i T_s} \tag{3.12}$$

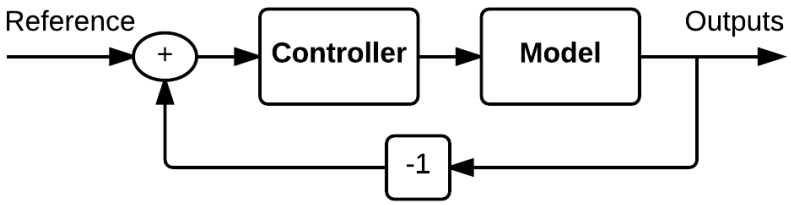



***Figure 3.1:*** *An illustration of the simulation in Simulink using the same controller as during the data collection and the estimated model.*

where $\lambda_i$ are the poles of the continuous system, will be close to the unit circle. If the sampling rate is very high, i.e., $T_s$ is very small, the accuracy of the location of the discrete-time poles will have to be much greater. This can cause problems due to numerical issues. Another reason why downsampling is a good idea is that to estimate a model from data with a very high sampling rate may result in a model being chosen that may not be the best one. When the sampling rate is very high and the signal-to-noise ratio is high, the next output sample will be almost equal to the previous one, and this can be described by a very simple model. If one downsamples the data, the predictor has to be more complex in order to describe the real system dynamics. This should result in a better predictor and a better estimate of the system.

To validate the models, a simulation was done in Matlab using Simulink. All three open-loop systems became unstable when simulating them. Instead, the closed-loop systems were simulated using the same controller structure and parameters as during the data collection, see Figure 3.1. With this approach, both the simulated angle and angular rate could be compared with the measured data. To get a measure of the performance of the models, the fit described in (3.11) was calculated for both the angle and the angular rate.

Since an attitude controller will be implemented in Chapter 4, the controller design will be easier if the estimated models are as simple as possible. Therefore if an ARX model structure can describe the system dynamics accurately enough, there is no need to add complexity by choosing an ARMAX model structure.

### 3.4.1 Roll and Pitch

The dynamics in the roll and pitch direction are similar, due to the geometry of the quadcopter, and can therefore be estimated using the same type of model structure. Both the roll and the pitch models were estimated using a linear ARMAX model structure, as in (3.6). The goal when choosing the model orders was to find one set that gave good results with the same model order in both roll and pitch.

The ARX model structure was used in a first attempt but could not fulfill the requirements of using the same model structure and orders in roll and pitch

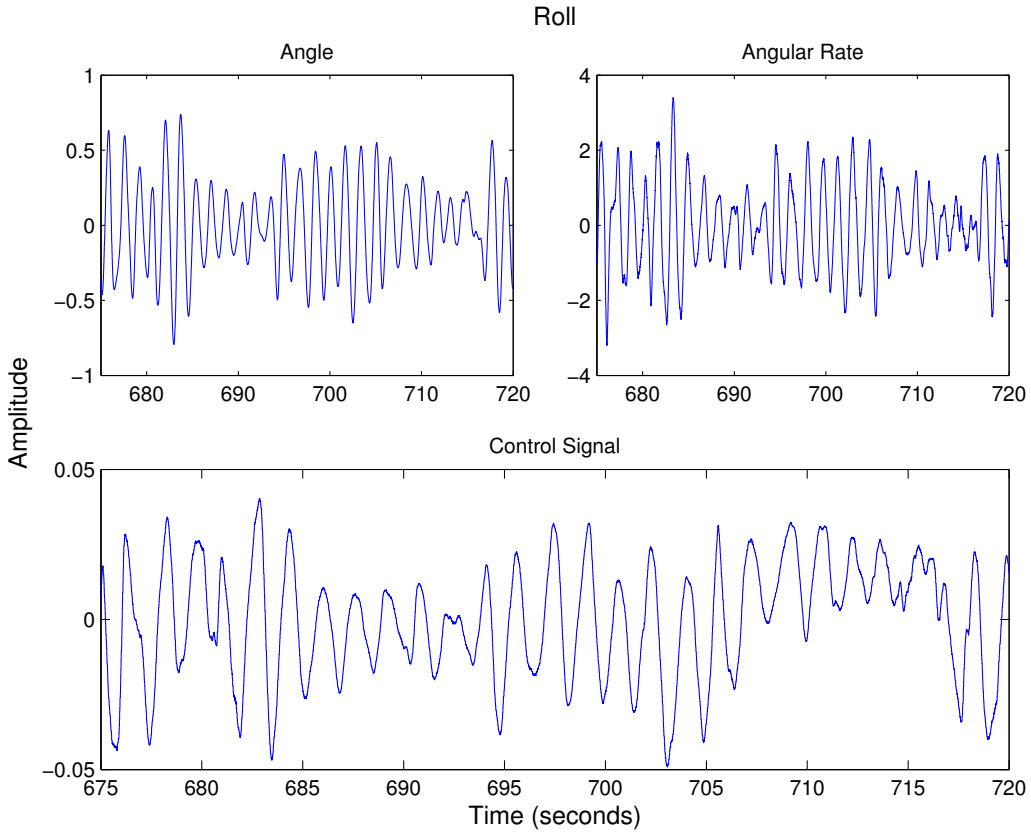



***Figure 3.2:*** *The dataset used to estimate the model in roll. Top left is the roll angle $\phi$, top right is the roll angular rate $p$, and in the bottom is the roll controller output $u_{roll}$.*

while maintaining good performance. However, with the ARMAX model the requirements could be fulfilled and the ARMAX model structure was therefore chosen. This result can be an indication of the need of a more flexible noise model, which the ARMAX model structure has. To estimate the models in roll and pitch, the datasets in Figure 3.2 and Figure 3.3 were used.

The downsampling was applied to both roll and pitch but it will be shown only for roll. As can be seen in Figure 3.4, downsampling with a factor of 2 gave better fit than no downsampling or downsampling with a factor of 3 and was therefore used. The fit of the different values of the downsampling factor, using the same model orders, are presented in Table 3.1. The fit in angular rate increased significantly when increasing the downsampling factor from 1 to 2. However, when the downsampling factor was increased to 3, there were only slight improvements in fit for the angular rate while the fit in angle decreased. Since we aim to control the roll and pitch angles in Chapter 4 by also using the angular rates in our controller design, the downsampling factor was selected to be 2. Both estimation datasets were therefore downsampled from 100 Hz to 50 Hz since this was shown to give the highest fit in both the angles and angular rates.

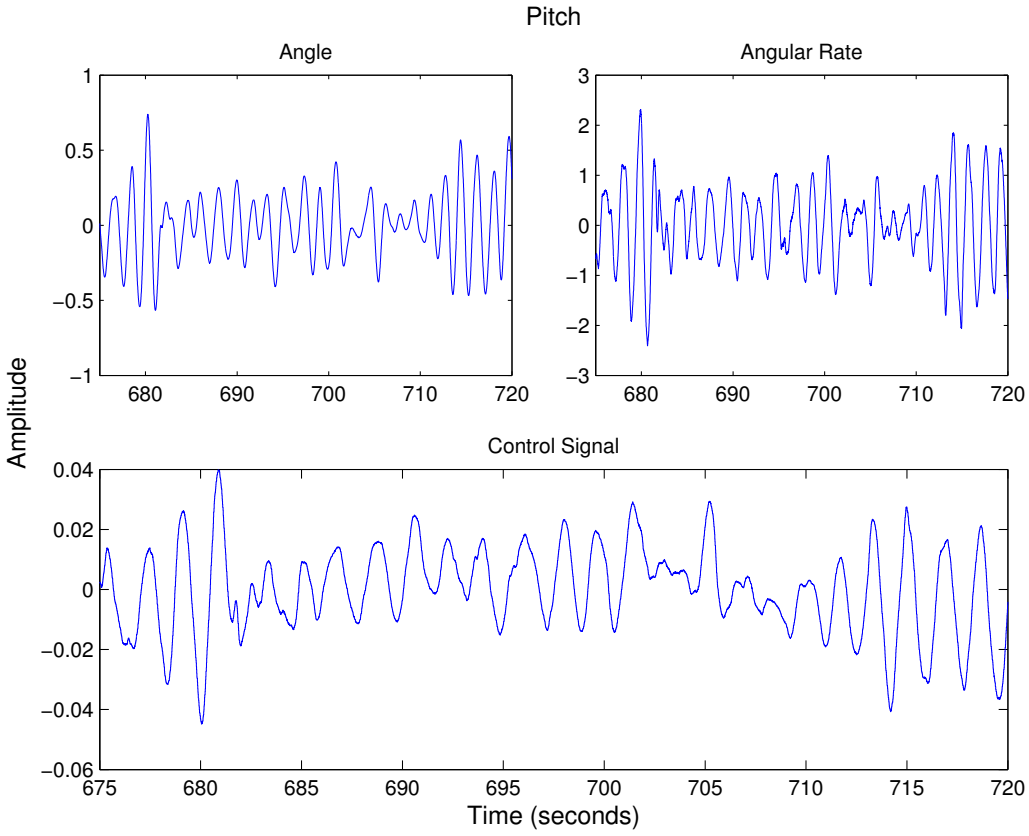



***Figure 3.3:*** *The dataset used to estimate the model in pitch. Top left is the pitch angle $\theta$, top right is the pitch angular rate $q$, and in the bottom is the pitch controller output $u_{pitch}$.*

***Table 3.1:*** *The fit of the roll model to the validation data for different values of the downsampling factor on the estimation dataset.*

| Factor | Fit of $\phi$ | Fit of $p$ |
|---|---|---|
| 1 | 0.4250 | 0.0337 |
| 2 | 0.5000 | 0.4745 |
| 3 | 0.5139 | 0.4697 |

The model orders

$$n_a = \begin{bmatrix} 3 & 4 \\ 4 & 4 \end{bmatrix}, n_b = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, n_c = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \text{ and } n_k = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.13}$$

were proven to be the best for both the roll and the pitch model. The model for roll is

$$\begin{aligned} A_\phi(z)y_\phi(t) &= -A_2(z)y_p(t) + B_\phi(z)u(t) + C_\phi(z)e_\phi(t) \\ A_p(z)y_p(t) &= -A_1(z)y_\phi(t) + B_p(z)u(t) + C_p(z)e_p(t) \end{aligned} \tag{3.14}$$

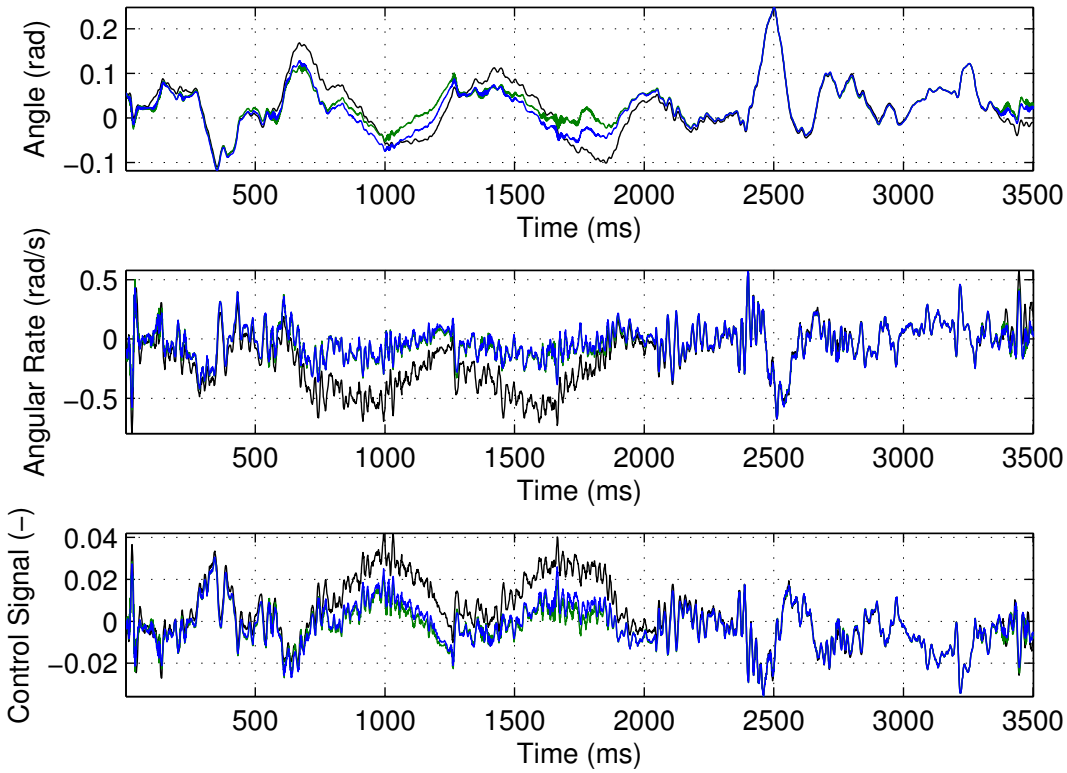



***Figure 3.4:*** *A plot of the error between the simulated data using the estimated model and the measured signals. Black represents the simulated data from a model estimated using a downsampling factor of* 1 *on the estimation dataset, blue a downsampling factor of* 2 *and green a downsampling factor of* 3*.*

where $y_\phi(t)$ is the roll angle, $y_p(t)$ is the roll angular rate, the polynomials are of the form

$$A_x(z) = a_0 + a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3} + a_4 z^{-4} \tag{3.15}$$

$$B_x(z) = b_1 z^{-1} + b_2 z^{-2} \tag{3.16}$$

and

$$C_x(z) = c_0 + c_1 z^{-1} \tag{3.17}$$

The estimated parameters are given in Tables 3.2 and 3.3,

Using this model, the simulation result for the validation data is shown in Figure 3.5. In this simulation, the fit was 0.5000 for $y_\phi(t)$ and 0.4745 for $y_p(t)$. As can be seen, the simulated signals follow the measured signals very well. However, in roll angle there is a clear error between the commanded reference and both the measured and simulated angles. This error is mainly the result of a poor controller but also of a disturbance present during the data collection. The controller did not have enough gain to accurately follow the reference and the side wind enlarged the error even more.

***Table 3.2:*** *The parameters of the $A_x(z)$ polynomials in the roll model. Note that elements that are exactly equal to 0 are not part of the polynomials.*

| | $A_\phi(z)$ | $A_p(z)$ | $A_1(z)$ | $A_2(z)$ |
|---|---|---|---|---|
| $a_0$ | 1.000 | 1.000 | 0 | 0 |
| $a_1$ | −2.946 | −1.545 | −35.62 | −0.003 829 |
| $a_2$ | 2.903 | −0.165 | 105.6 | −0.000 982 5 |
| $a_3$ | −0.9568 | 1.013 | −104.9 | 0.012 24 |
| $a_4$ | — | −0.2926 | 34.95 | −0.007 668 |

***Table 3.3:*** *The parameters of the $B_x(z)$ and $C_x(z)$ polynomials in the roll model.*

| | $B_\phi(z)$ | $B_p(z)$ | | $C_\phi(z)$ | $C_p(z)$ |
|---|---|---|---|---|---|
| $b_1$ | 0.1129 | 6.037 | $c_0$ | 1.000 | 1.000 |
| $b_2$ | −0.1121 | −6.015 | $c_1$ | −0.6433 | −0.7727 |

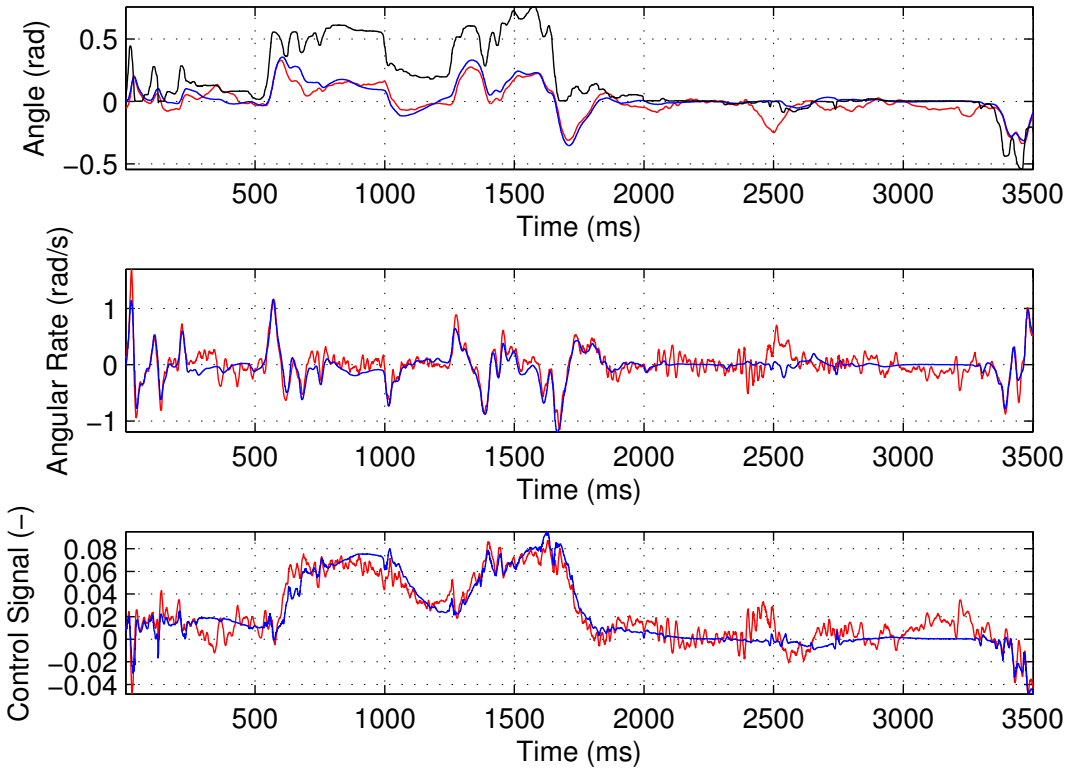



***Figure 3.5:*** *The simulation done in Simulink for the estimated model in roll using the validation data. Red is the measured experimental data, blue is the simulated data and black is the commanded angle.*

***Table 3.4:*** *The parameters of the $A_x(z)$ polynomials in the pitch model. Note that elements that are exactly equal to 0 are not part of the polynomials.*

| | $A_\theta(z)$ | $A_q(z)$ | $A_1(z)$ | $A_2(z)$ |
|---|---|---|---|---|
| $a_0$ | 1 | 1 | 0 | 0 |
| $a_1$ | −2.921 | −1.592 | −34.85 | −0.003 282 |
| $a_2$ | 2.853 | −0.0735 | 105.0 | −0.000 973 1 |
| $a_3$ | −0.9317 | 0.9549 | −105.7 | 0.010 77 |
| $a_4$ | — | −0.2817 | 35.64 | −0.006 693 |

***Table 3.5:*** *The parameters of the $B_x(z)$ and $C_x(z)$ polynomials in the pitch model.*

| | $B_\theta(z)$ | $B_q(z)$ | | $C_\theta(z)$ | $C_q(z)$ |
|---|---|---|---|---|---|
| $b_1$ | 0.034 23 | 2.843 | $c_0$ | 1.000 | 1.000 |
| $b_2$ | −0.033 48 | −2.831 | $c_1$ | −0.548 | −0.706 |

Similarly, the model for pitch is

$$\begin{aligned} A_\theta(z)y_\theta(t) &= -A_2(z)y_q(t) + B_\theta(z)u(t) + C_\theta(z)e_\theta(t) \\ A_q(z)y_q(t) &= -A_1(z)y_\theta(t) + B_q(z)u(t) + C_q(z)e_q(t) \end{aligned} \tag{3.18}$$

where $y_\theta(t)$ is the pitch angle, $y_q(t)$ is the pitch angular rate and $A_x(z)$, $B_x(z)$ and $C_x(z)$ are defined in (3.15) - (3.17) and the parameters are specified in Tables 3.4 and 3.5.

Using this model, the resulting simulation for the validation data is shown in Figure 3.6a. In this simulation, the fit was 0.0264 for $y_\theta(t)$ and 0.4221 for $y_q(t)$. The simulated angle follows the behaviour of the measured angle very well but there is a clear offset causing a low fit. This could be the result of the wind present during the data collection, causing the measured angle from the NAV to differ more from the commanded angle than it would have without any wind, or the result of the center of gravity of the quadcopter not being centered along the x-axis. To simulate this behaviour, a disturbance of +0.05 was added to the Simulink model according to

$$\tilde{u}_{pitch,k} = u_{pitch,k} + v_k \tag{3.19}$$

where $v_k$ is the added disturbance. The resulting simulation is presented in Figure 3.6b. It can be seen that this added disturbance actually pushes the simulation data closer to the measured data, thus giving a higher fit of 0.4773 for $y_\theta(t)$ and 0.4758 for $y_q(t)$.

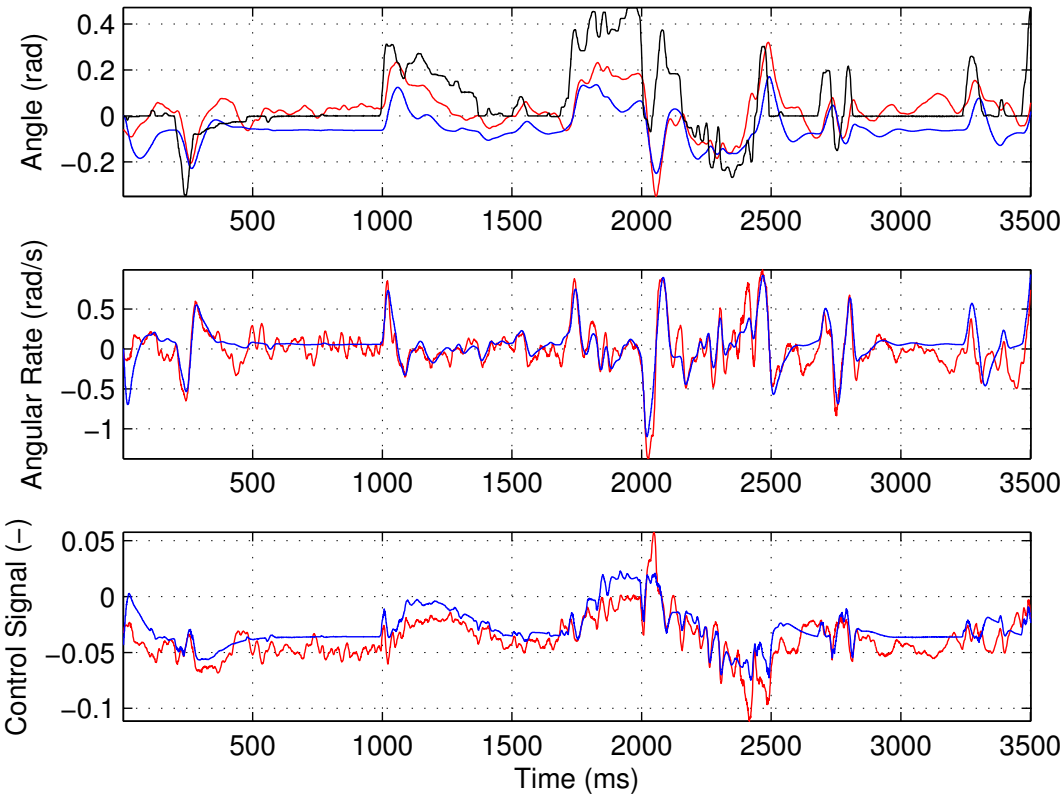
As a comparison, and for further validation of the models, one can study the simulation results if the model orders in roll and pitch are increased or decreased

***Table 3.6:*** *The fit of the roll and pitch models to the validation data when varying the model orders $n_a$, $n_b$ and $n_c$ by one. The fit of the pitch models are with an added system disturbance of* +0.05.

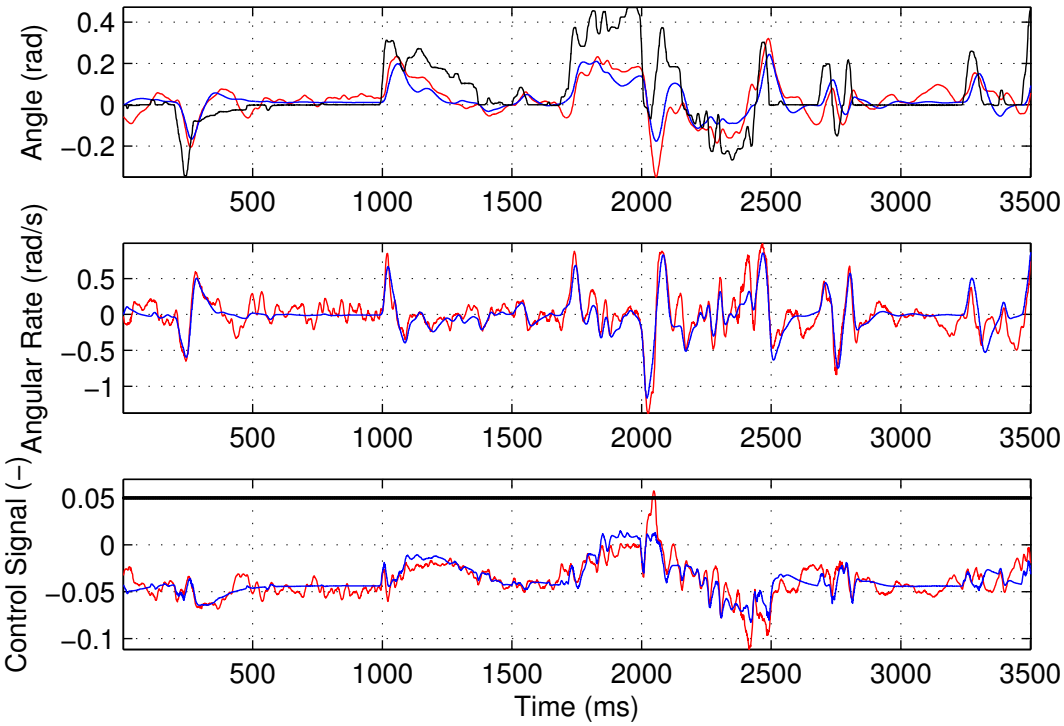
| Diff | Fit in $\phi$ | Fit in $p$ | Fit in $\theta$ | Fit in $q$ |
|---|---|---|---|---|
| 0 | 0.5000 | 0.4745 | 0.4773 | 0.4758 |
| +1 | 0.5177 | 0.4798 | 0.4711 | 0.4846 |
| −1 | 0.2787 | −0.3823 | 0.1207 | −0.0585 |

by one in $n_a$, $n_b$ and $n_c$. By increasing the model orders, the model complexity increases and by decreasing the model orders the model complexity decreases. The resulting simulations can be seen in Figures 3.7a and 3.7b. The fit of the angle and angular rates are presented in Table 3.6. The fit decreased dramatically when decreasing the model orders by one for both pitch and roll. However, when the model orders were increased by one, the fit increased in both angle and angular rate in roll and angular rate in pitch, but decreased in pitch angle. Because of the decrease in pitch angle, the model orders were chosen as presented earlier in this section. This result shows that a less complex model gives less accurate simulations and that a more complex model does not improve the results enough to motivate the added complexity. As previously mentioned, it will be easier to design attitude controllers if the models are as simple as possible.

A plot of the error between the simulated data using the estimated model and the measured signals

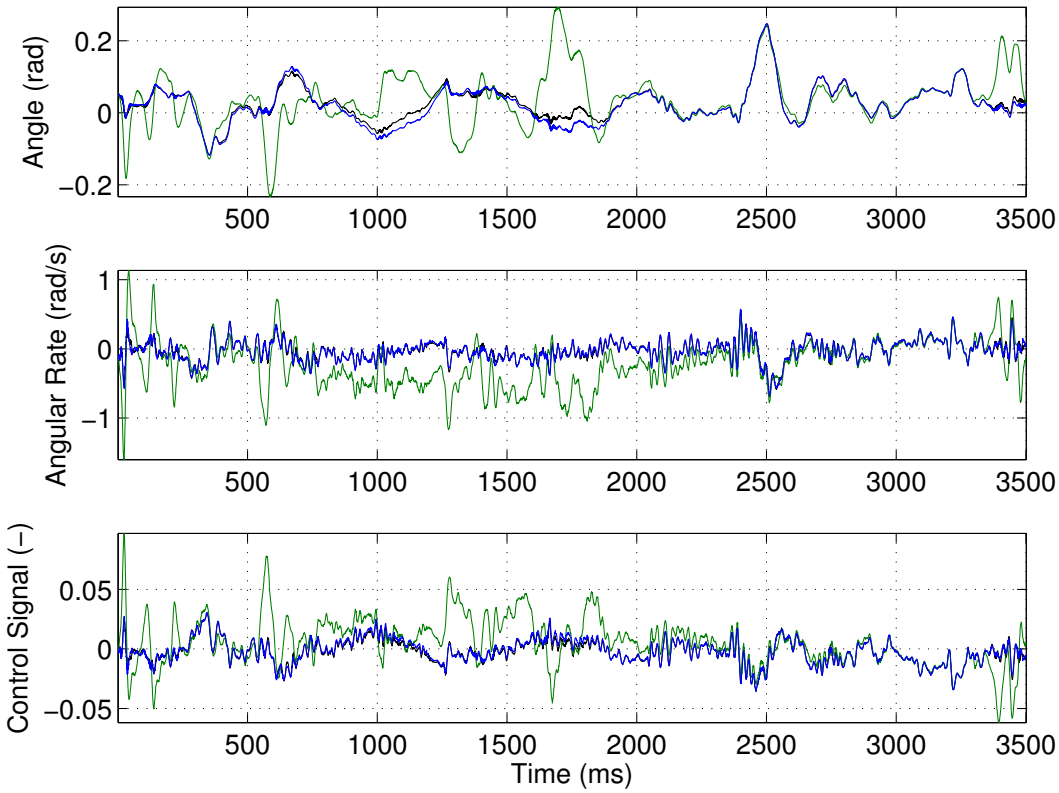



***(a)*** *Without any added system disturbance.*

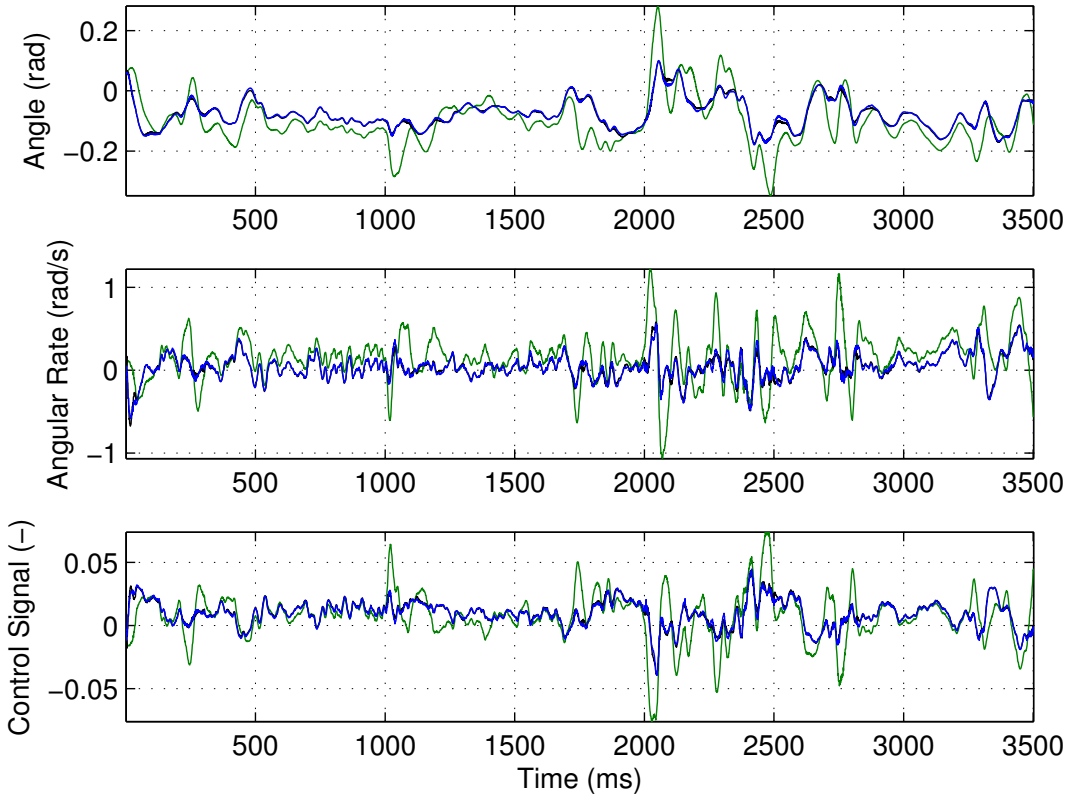



***(b)*** *With an added system disturbance of* +0.05.

***Figure 3.6:*** *The simulations using the validation data for the estimated model in pitch with and without an added system disturbance of a constant* +0.05. *Red is the measured experimental data, blue is the simulated data, black in the top plots is the commanded angle and black in the bottom plot of Figure 3.6b is the added system disturbance.*




***(a)*** *Roll*




***(b)*** *Pitch*

***Figure 3.7:*** *A comparison of different model orders using the validation data. The plots show the error between the simulated and the measured data. Blue is the error when using the chosen model orders, green when the model orders are reduced by* 1 *and black when the model orders are increased by* 1*.*

### 3.4.2 Yaw

The dynamics in yaw was estimated using a linear ARX model structure. The choice of using an ARX model was due its simplicity and the ARX model was seen to perform well. Therefore an ARMAX model was never tested. The experimental data that was used for the model estimation is shown in Figure 3.9. Since we later in Chapter 4 aim to control the yaw angular rate, and not the angle, we will focus only on the fit to the angular rate data.

The estimation dataset was downsampled to 33 Hz instead of 100 Hz, corresponding to a downsampling factor of 3. This was shown to give the highest fit of the angular rate. As can be seen in Figure 3.8, downsampling with a factor of 3 indicated better prediction results than both 2 and 4 and was therefore chosen. The fit of the different values of the downsampling factor are presented in Table 3.7. As can be seen here, the fit in angular rate increased when increasing the downsampling factor from 2 to 3. However, as the downsampling factor was increased to 4 the fit in angular rate decreased slightly. Therefore, the downsampling factor was set to be 3.

Using simulation as the validation method, the model order

$$n_a = \begin{bmatrix} 4 & 4 \\ 3 & 4 \end{bmatrix}, n_b = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \text{ and } n_k = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \tag{3.20}$$

was proven to be the best for the yaw model. The model is

$$\begin{aligned} A_\psi(z)y_\psi(t) &= -A_2(z)y_r(t) + B_\psi(z)u(t) + e_\psi(t) \\ A_r(z)y_r(t) &= -A_1(z)y_\psi(t) + B_r(z)u(t) + e_r(t) \end{aligned} \tag{3.21}$$

where $y_\psi(t)$ is the yaw angle, $y_r(t)$ is the yaw angular rate and $A_x(z)$ and $B_x(z)$ are polynomials, see (3.15) and (3.16), with the parameters given in Tables 3.8 and 3.9.

Using (3.21), the resulting simulation for validation data is shown in Figure 3.10a. In this simulation, the fit was −0.4916 for $y_r(t)$. Same as when studying the simulation results in pitch, it can be seen in Figure 3.10a that the simulated angular rate follows the trends of the measured angle very well but that there is a clear offset causing a low fit. By adding a system disturbance of a constant −0.02, according to

$$\tilde{u}_{yaw,k} = u_{yaw,k} + v_k \tag{3.22}$$

where $v_k$ is the added disturbance, the fit increased to 0.5321 for $y_r(t)$. Simulation results with this disturbance are shown in Figure 3.10b.

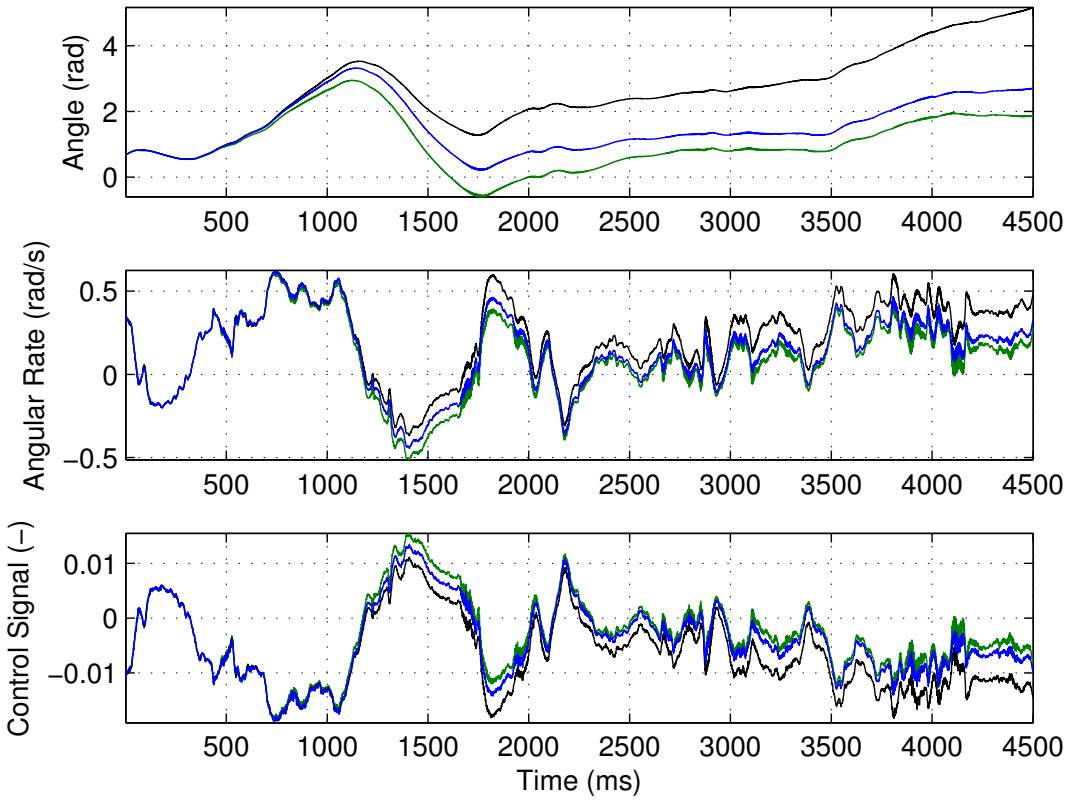



***Figure 3.8:*** *A comparison of different values of the downsampling factor. The plot shows the error between the simulated data using the estimated model and the measured data. Black represents the error between the simulated data and the measured data when the model is estimated using a downsampling factor of 2, blue a downsampling factor of 3 and green a downsampling factor of 4.*

***Table 3.7:*** *The fit of the yaw model to the validation data for different values of the downsampling factor on the estimation dataset.*

| Factor | Fit of $r$ |
|---|---|
| 2 | 0.6550 |
| 3 | 0.6866 |
| 4 | 0.6680 |

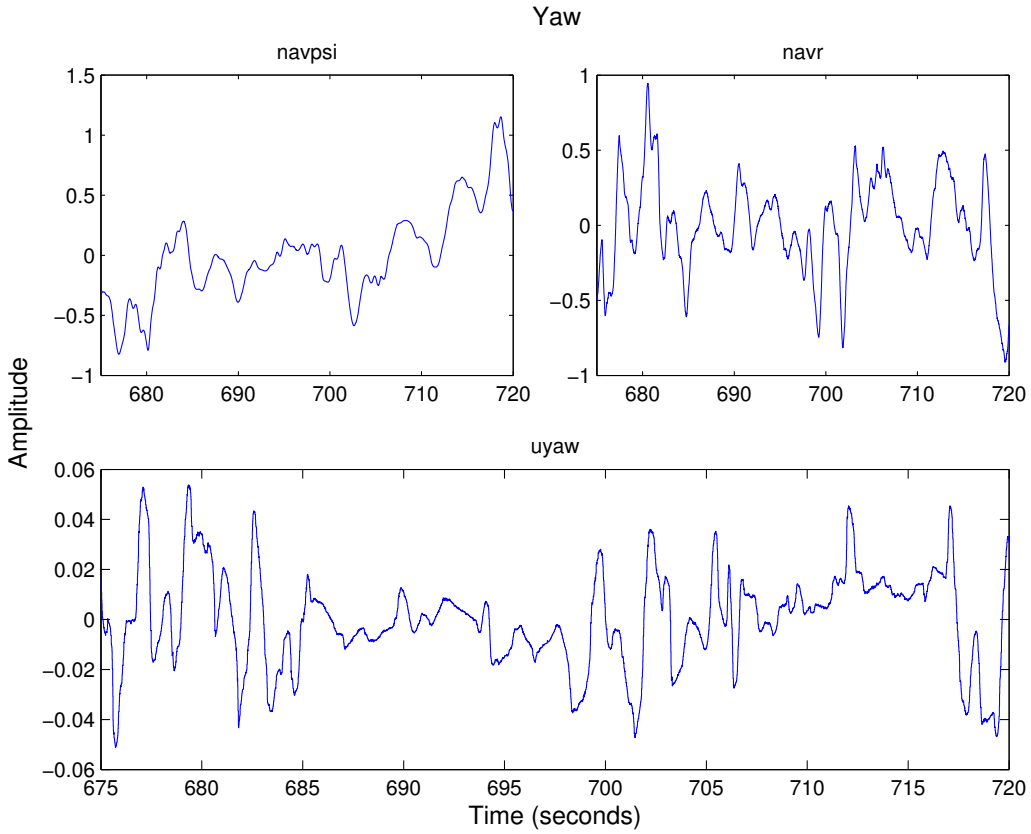


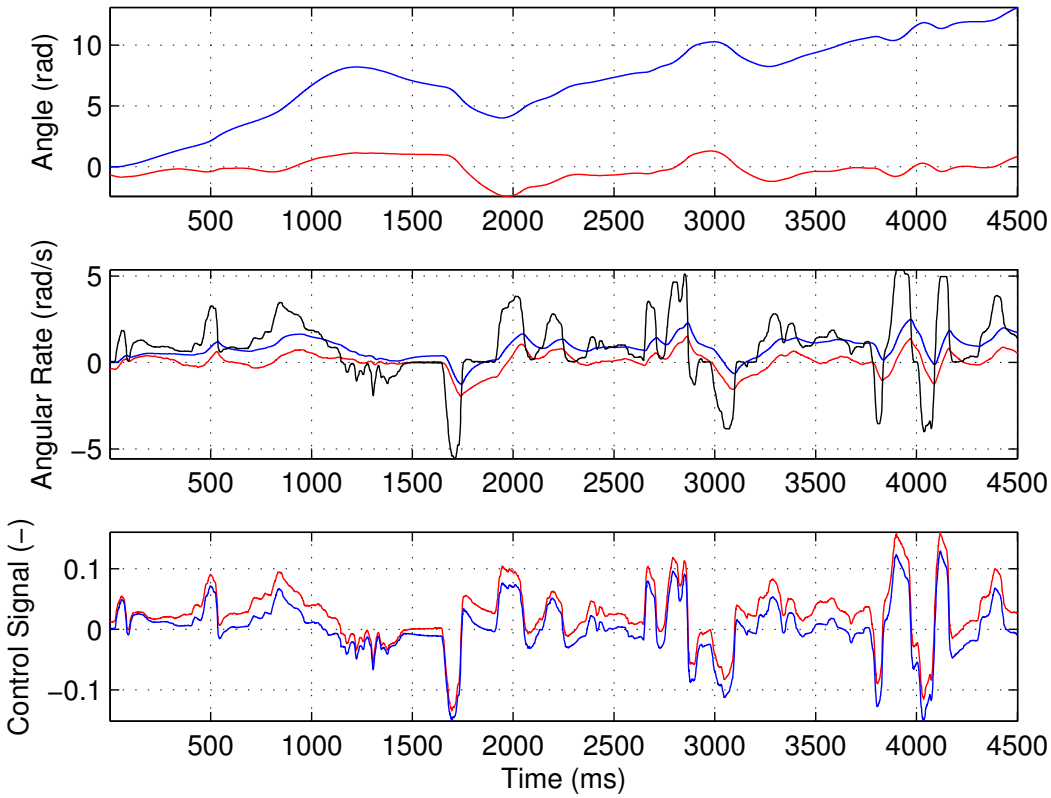

***Figure 3.9:*** *The dataset used to estimate the yaw model. Top left is the yaw angle $\psi$, top right is the yaw angular rate $r$ and in the bottom is the yaw controller output $u_{yaw}$.*

***Table 3.8:*** *The parameters of the $A_x(z)$ polynomials in the yaw model. Note that elements that are exactly equal to 0 are not part of the polynomials.*

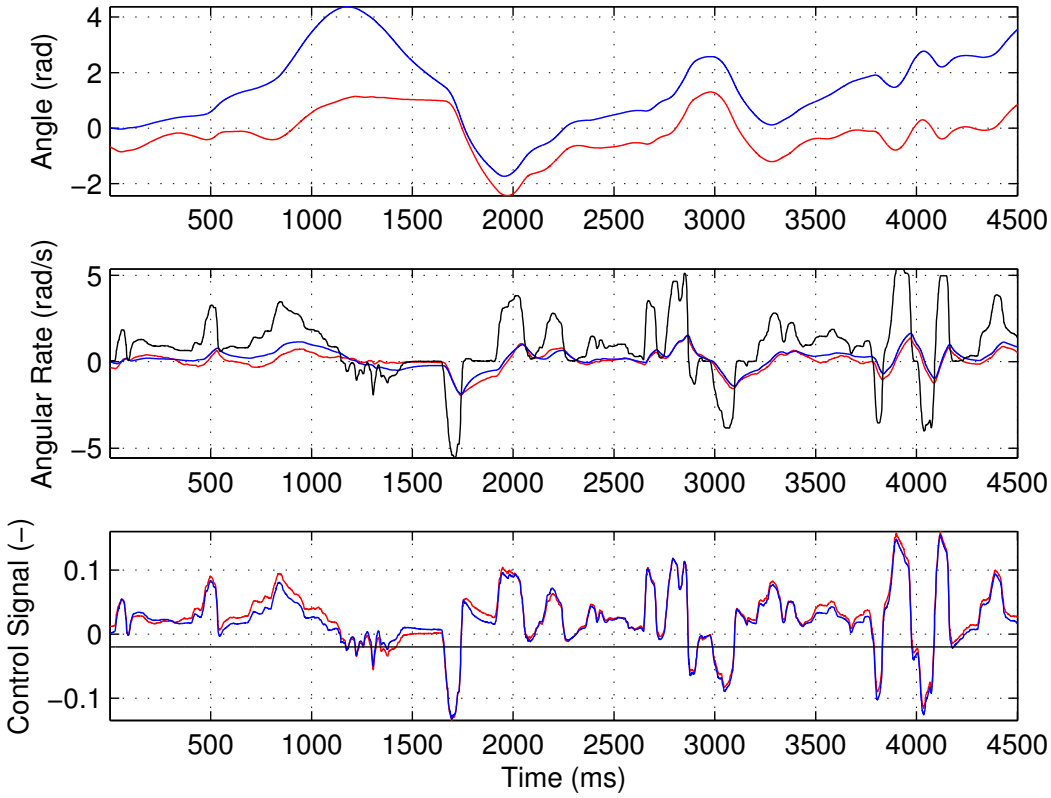
| | $A_\psi(z)$ | $A_r(z)$ | $A_1(z)$ | $A_2(z)$ |
|---|---|---|---|---|
| $a_0$ | 1.000 | 1.000 | 0 | 0 |
| $a_1$ | −2.753 | −1.743 | −0.6279 | −0.007 034 |
| $a_2$ | 2.665 | 0.7582 | 1.188 | 0.009 113 |
| $a_3$ | −1.062 | 0.266 | −0.562 | −0.004 711 |
| $a_4$ | 0.1504 | −0.2676 | — | 0.002 415 |

***Table 3.9:*** *The parameters of the $B_x(z)$ polynomials in the yaw model. Note that elements that are exactly equal to 0 are not part of the polynomials.*

| | $B_\psi(z)$ | $B_r(z)$ |
|---|---|---|
| $b_1$ | 0.028 06 | 0.3664 |
| $b_2$ | −0.0255 | 0 |




***(a)*** *No added system disturbance*




***(b)*** *With an added system disturbance of* −0.02.

***Figure 3.10:*** *The simulations using the validation data for the estimated model in yaw with and without an added system disturbance of a constant* −0.02. *Red is the measured experimental data, blue is the simulated data, black in the middle plots is the commanded angular rate and black in the bottom plot of Figure 3.10b is the added system disturbance.*

# 4

# Control Design

A control system uses sensor measurements to give feedback to the inputs of the system in order to make corrections to achieve a desired performance. A system that uses feedback from sensor measurements to compute the inputs is referred to as a closed-loop system. A system that does not is referred to as an open-loop system.

This chapter begins with an overview of some control basics. Next, the design of the attitude controller will be explained and presented. Finally, the results and performance of the controller will be discussed.

## 4.1 Feedback Control

This section will cover some linear control theory. A general control loop is illustrated in Figure 4.1, where $G(s)$ is the plant, i.e. the real physical system or a set of equations describing the system dynamics and $F_r(s)$ and $F_y(s)$ are the parametrized linear controllers.

From Figure 4.1 the closed-loop transfer function

$$G_c(s) = \frac{F_r(s)G(s)}{1 + F_y(s)G(s)} \tag{4.1}$$

describing the relationship between the reference $r(t)$ and the output $y(t)$ can be derived. If $F_r(s)$ and $F_y(s)$ are chosen to be equal then

$$G_c(s) = \frac{F(s)G(s)}{1 + F(s)G(s)} \tag{4.2}$$

follows from (4.1).

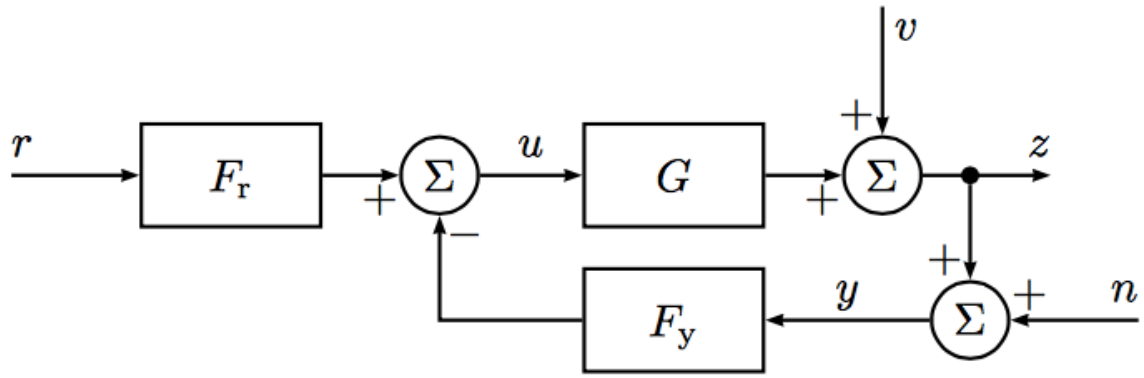



***Figure 4.1:*** *A illustration of the general closed-loop system when using a linear controller parametrized by the transfer functions $F_r(s)$ and $F_y(s)$. $G(s)$ is the plant, $r(t)$ is the reference signal, $u(t)$ is the controller output, $v(t)$ is an output disturbance, $z(t)$ is the controlled variable, $n(t)$ is the measurement noise and $y(t)$ is the output.*

There exist many different control algorithms for linear systems. The most common one is the PID controller. The ideal PID-controller is defined as

$$u(t) = K_P e(t) + K_I \int_{t_0}^{t} e(t)dt + K_D \frac{de(t)}{dt} \tag{4.3}$$

where $u(t)$ is the control output and $e(t)$ is the control error defined as $e(t) = r(t) - y(t)$ (Glad and Ljung, 2006). The design parameters $K_P$, $K_I$ and $K_D$ are chosen to achieve the desired closed-loop performance. Applying the Laplace transform on (4.3), the resulting equation is

$$U(s) = (K_P + \frac{K_I}{s} + K_D s)E(s) \tag{4.4}$$

When implementing a PID controller on a real system, (4.4) can be discretized to

$$u_k = K_P e_k + K_I \sum_{j=1}^{k} e_j T_s + K_D \frac{e_k - e_{k-1}}{T_s} \tag{4.5}$$

using a specified sampling time $T_s$, where

$$\sum_{j=1}^{k} e_j T_s \tag{4.6}$$

will be referred to as the *integrator part*.

## 4.2 Implementation Aspects

To implement the controllers on the quadcopter and to guarantee a safe flight, some practical issues had to be solved. First, conditional integration was implemented according to Algorithm 1 in order to avoid windup. See Åström (2006) for more information about windup and conditional integration. In Algorithm 1, $T_s$ is the sampling time, $u_{\text{throttle}}$, $u_{\text{roll}}$, $u_{\text{pitch}}$ and $u_{\text{yaw}}$ are the controller outputs and $i_{\text{roll}}$, $i_{\text{pitch}}$ and $i_{\text{yaw}}$ are the integrator parts in (4.6). The sum of the absolute values of the controller outputs are compared to 1, which is the maximum motor signal an ESC can handle, see (2.21). If the sum is less than 1, the integrator parts are updated. If the sum is greater or equal to 1, the integrator parts are not updated and stays the same as in the previous iteration. Since the sum of the controller outputs, and not the motor signals, are compared to 1, this algorithm is simplified. However, this solution prevents the integrator parts from increasing when the motors are saturated in most cases.

Next, since the motor signal that the ESCs can deliver to the motors is limited, a prioritization of the controller outputs was done according to Algorithm 2, where $u$ is a vector containing $u_{\text{throttle}}$, $u_{\text{roll}}$, $u_{\text{pitch}}$ and $u_{\text{yaw}}$. According to Section 2.3 the motor signals are in the range of 0 to 1 and Algorithm 2 prioritizes the controller outputs and prevents the controller outputs from giving negative motor signals. For example, if $u_{\text{throttle}}$ is 0.4 then according to (2.21) $M_1$, $M_2$, $M_3$ and $M_4$ will all have an output contribution of 0.4 from this controller output. If now $u_{\text{roll}}$ is set to 0.5 and $u_{\text{pitch}} = u_{\text{yaw}} = 0$ then (2.21) gives the outputs to the ESCs as $M_1 = -0.1$, $M_2 = -0.1$, $M_3 = 0.9$ and $M_4 = 0.9$. In this case, Algorithm 2 will limit $u_{\text{roll}}$ and therefore preventing $M_1$ and $M_2$ from getting negative values. The outputs are being prioritized in the order; roll, pitch and last yaw, since the control in roll and pitch are more crucial than yaw to keep the quadcopter stable. Algorithm 2 is simplified due to only taking the controller outputs, and not the motor signals, into account. However, most cases of saturation will be taken care of.

**Algorithm 1** Anti Windup Solution

**if** $|u_{\text{throttle}}| + |u_{\text{roll}}| + |u_{\text{pitch}}| + |u_{\text{yaw}}| < 1$ **then**
    $i_{\text{roll}}(k+1) = i_{\text{roll}}(k) + T_s(r_{\text{roll}}(k) - \phi(k))$.
    $i_{\text{pitch}}(k+1) = i_{\text{pitch}}(k) + T_s(r_{\text{pitch}}(k) - \theta(k))$.
    $i_{\text{yaw}}(k+1) = i_{\text{yaw}}(k) + T_s(r_{\text{yaw}}(k) - r(k))$.

**Algorithm 2** Output Prioritization

$c = 0.5 - |u_{\text{throttle}} - 0.5|$
**for** $i = 2 : 4$ **do**
    $u(i) = \max(-c), \min(u(i), c))$
    $c = c - |u(i)|$

**Algorithm 3** Keep Integrators Deactivated During the Initial Stages of Takeoff

1: **if** $u_{\text{throttle}}(k) < 0.3$ **then**
2: $i_{\text{roll}}(k+1) = 0.$
3: $i_{\text{pitch}}(k+1) = 0.$
4: $i_{\text{yaw}}(k+1) = 0.$

Another important implementation aspect is how to turn on and off the integral action during flight. Since a small error in attitude when in the starting position will keep integrating, the integral action could potentially cause problems during takeoff. Once the quadcopter actually lifts off, it is possible that the integrator parts have become very large, which could cause stability problems. To avoid this problem, one can either manually switch on the integrators during flight, once safely off the ground. This solution requires the integrator parts to also be reset when switched on. Another solution, which is the one implemented on the quadcopter according to Algorithm 3, is to keep the integrators switched off until the throttle reaches a certain limit. This allows the integrators to function as planned without needing the pilot's action. If the throttle is greater than 0.3 the integrator parts will update according to Algorithm 1, but if the throttle is less than 0.3 the integrator parts will be set to zero. While in hovering, the needed throttle is about 0.6-0.7. This means that the throttle will rarely be less than 0.3 during flight. Therefore, the algorithm allows the integrators to update normally as soon as possible but keeps them at zero until takeoff.

The algorithms have been implemented in Matlab and translated to C-code with the Matlab Coder software.

## 4.3 Attitude Controller

In order to control the orientation of the quadcopter, an attitude controller has been implemented. The attitude controller consists of three decoupled controllers, a roll controller, a pitch controller and a yaw controller, described in Sections 4.3.1, 4.3.2 and 4.3.3, respectively.

To choose the parameters of the three controllers, they were simulated together with the estimated models using different values on the controller parameters. As a safety measure, to further validate the controllers, the controllers were designed to control the estimated models as well as possible while also controlling three theoretical models of the real system, supplied by Saab Dynamics AB. The theoretical models are in continuous time and are presented in each section. These continuous-time models were created by trying to mimic the bode plots of the estimated models in the frequency range of 5 to 30 rad/s. The controllers were then designed using both models which should make the resulting controllers more robust against uncertainty outside the frequency range 5 to 30 rad/s.

### 4.3.1 Roll Controller

Figure 4.2 illustrates the quadcopter's attitude controller in roll. The theoretical model of the roll dynamics is

$$\begin{aligned} y_\phi &= \frac{15000}{s(s+30)(s+1)} \\ y_p &= \frac{15000}{(s+30)(s+1)} \end{aligned} \tag{4.7}$$

and is derived from a motor with a bandwidth of 30 rad/s, an amplification of 500 from the motor signals to angular rate, a factor $s + 1$ corresponding to a weakly attenuated angular rate and a pure integration from angular rate to angle. Figure 4.3 compares the bode plots of the theoretical model and the estimated model. As can be seen, the models share the same crossover frequency but differ greatly in the lower frequencies and some in the higher. As previously mentioned, a more robust controller can be designed by also studying the theoretical model. Using both models, the roll controller was tuned manually to track the reference using an as small controller output as possible. The resulting controller parameters are presented in Table 4.1.

Using the chosen controller parameters, two simulations were performed with both models, see Figure 4.4. The first simulations were done with $K_I$ set to zero, see the top plots, and the second ones with $K_I$ set according to Table 4.1, see the bottom plots in Figure 4.4. Both simulations are with the closed-loop system with the designed controller using a reference from a logged dataset. By simulating the controller with and without $K_I$ set to zero, the difference is apparent. For the estimated model, the control error is much larger when $K_I$ is set to zero and almost not present with $K_I$ not set to zero, except for the larger roll angles where the error is still present. For the theoretical model, there is almost no control error, even when $K_I$ is zero. It was not possible to achieve better performance with the estimated model without causing the closed loop system with the theoretical model to become unstable.

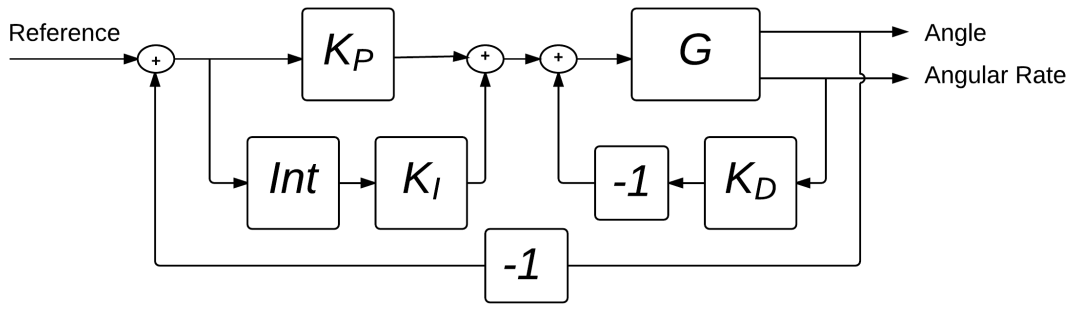



***Figure 4.2:*** *A illustration of the controller used to control the attitude in roll. Int is an integrator and G is the quadcopter.*

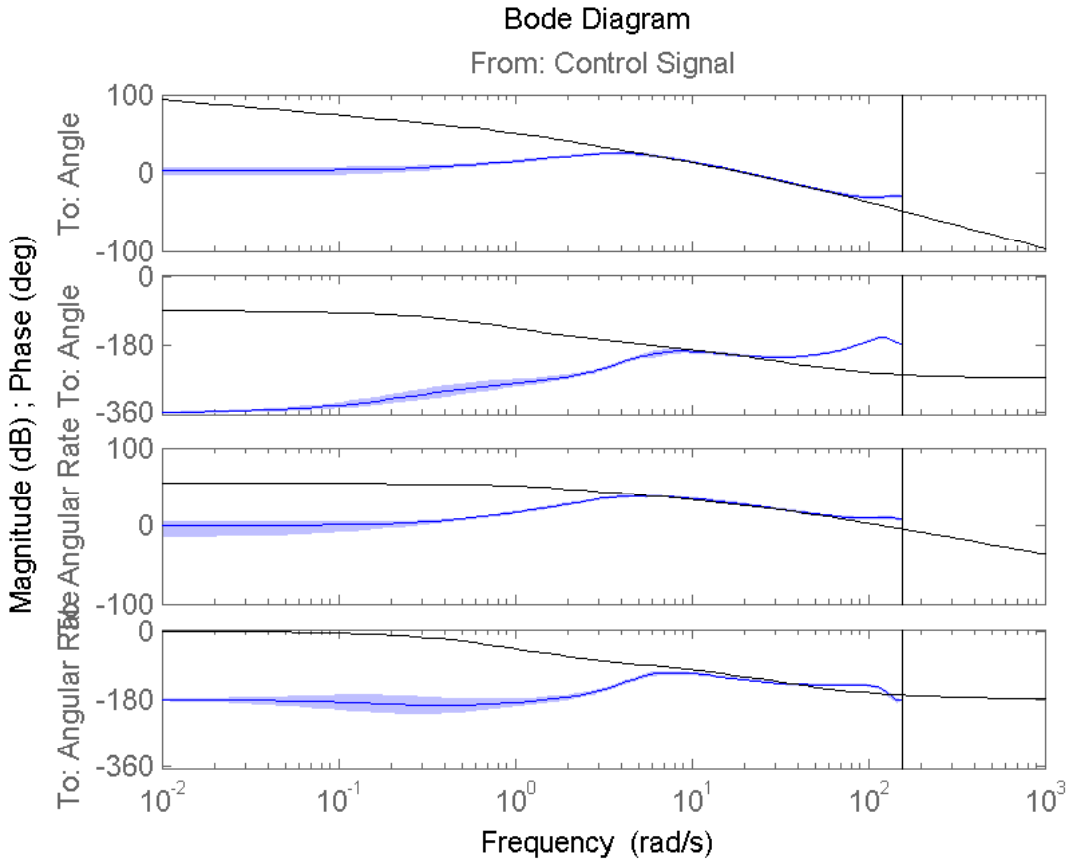
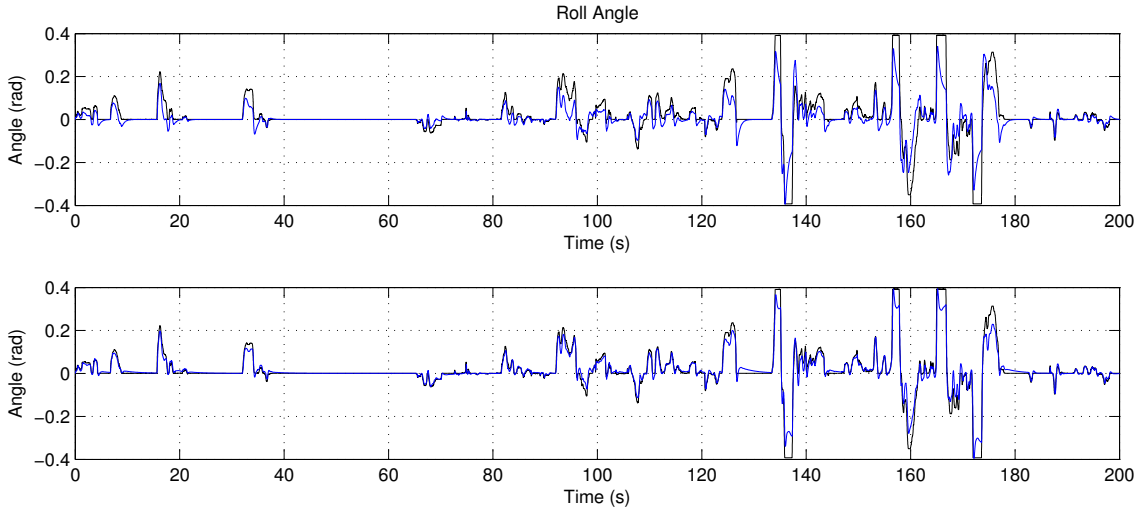


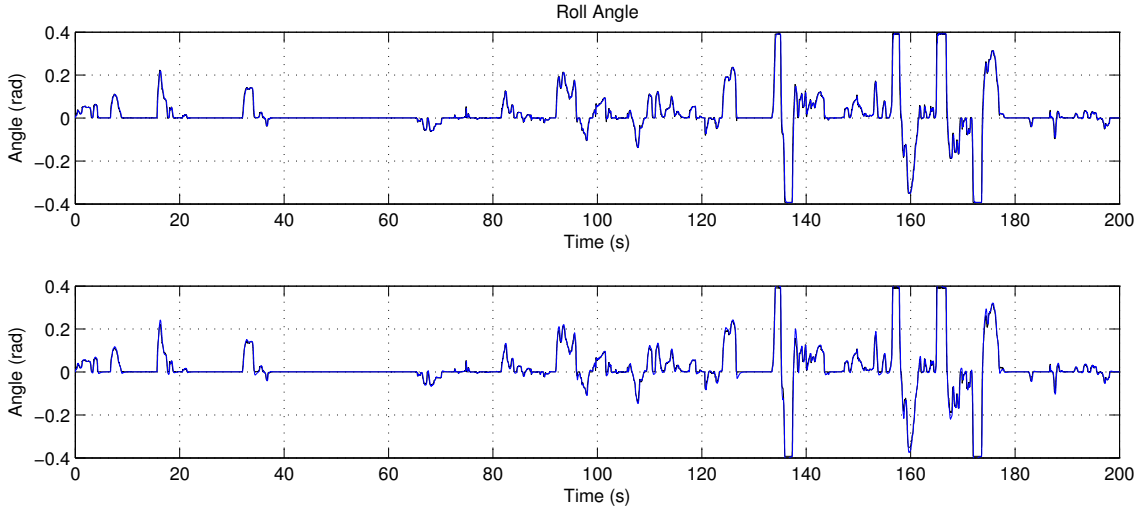

***Figure 4.3:*** *A bode plot comparing the estimated model for roll, in blue, and the theoretical model, in black, also showing the confidence interval of one standard deviation for the estimated model.*

***Table 4.1:*** *The parameters of the roll controller*

| Roll | |
|---|---|
| $K_P$ | 0.5 |
| $K_I$ | 0.8 |
| $K_D$ | 0.1 |




***(a)*** *The estimated model*




***(b)*** *The theoretical model*

***Figure 4.4:*** *The simulation of the complete closed-loop roll system, using a reference from a dataset collected during the system identification experiments, showing the performance of the controller. The top plots are with $K_I$ set to zero and the bottom plots are with $K_I$ set according to Table 4.1. Black is the commanded angle and blue is the simulated angle.*

### 4.3.2 Pitch Controller

To control the attitude in pitch, a controller with the same structure as for the controller in roll was implemented, see Section 4.3.1. The theoretical model of the pitch dynamics is

$$\begin{aligned} y_\theta &= \frac{9000}{s(s+30)(s+1)} \\ y_q &= \frac{9000}{(s+30)(s+1)} \end{aligned} \tag{4.8}$$

and is derived from a motor with a bandwidth of 30 rad/s, an amplification of 300 from the motor signals to angular rate, a factor $s + 1$ corresponding to a weakly attenuated angular rate and a pure integration from angular rate to angle. Note that the amplification is weaker than in roll. This could be because the quadcopter is longer along the x-axis than the y-axis and this causes the moment of inertia to be larger. Figure 4.5 shows a comparison of the bode plots of the theoretical model and the estimated model. As can be seen, the models share the same crossover frequency but differ more in the lower frequencies and some in the higher. Using both models, a more robust controller could be designed. The pitch controller was tuned manually to track the reference using an as small controller output as possible. The resulting controller parameters are presented in Table 4.2.

Using the chosen pitch controller, two simulations were performed with both models, see Figure 4.6. The first simulations were with $K_I$ set to zero, see the top plots, and the second ones with $K_I$ set according to Table 4.2, see the bottom plots. Both simulations are on the closed-loop system with the designed controller using a reference from a logged dataset. For the estimated model, the control error decreases when the integral action is added but is still present. For the theoretical model, the control error decreases remarkably when the integral action is added and appears to be almost zero.

***Table 4.2:*** *The parameters of the pitch controller*

| Pitch | |
|---|---|
| $K_P$ | 0.7 |
| $K_I$ | 0.5 |
| $K_D$ | 0.07 |

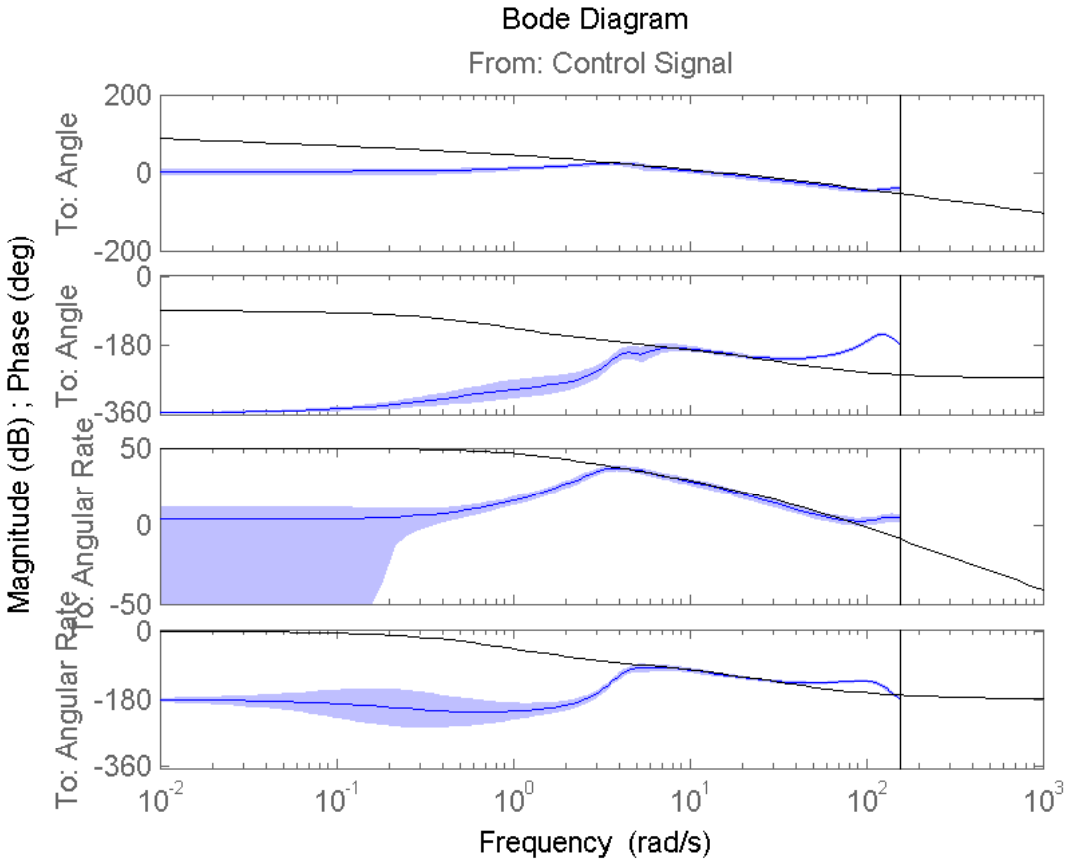
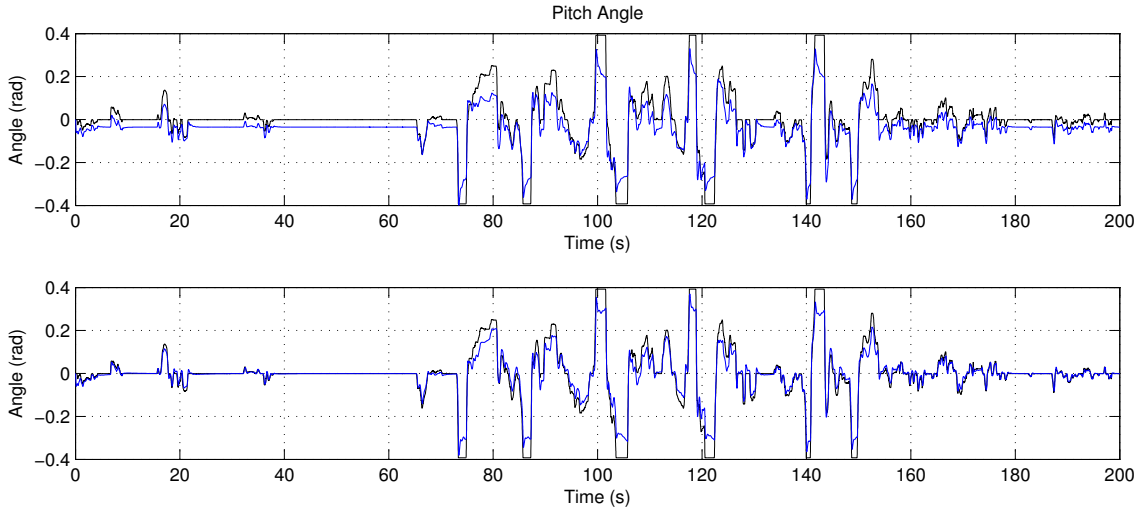
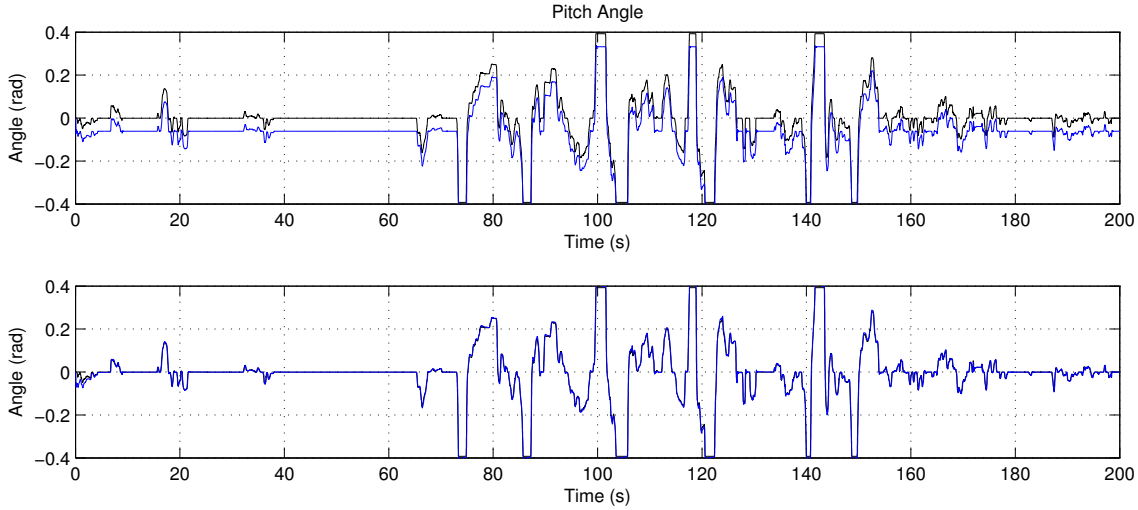



***Figure 4.5:*** *A bode plot comparing the estimated model for pitch, in blue, and the theoretical model, in black, also showing the confidence interval of one standard deviation for the estimated model.*




*(a) The estimated model*




*(b) The theoretical model*

***Figure 4.6:*** *The simulation of the complete closed-loop pitch system, using a reference from a dataset collected during the system identification experiments, showing the performance of the controller. The top plots are with $K_I$ set to zero and the bottom plots are with $K_I$ set according to Table 4.2. Black is the commanded angle and blue is the simulated angle.*

### 4.3.3 Yaw Controller

Unlike roll and pitch, which are controlled in angle, yaw will be controlled in angular rate. This approach is considered to be the more common way for a pilot to steer a quadcopter in yaw. The yaw controller was designed using a PID controller where $K_D$ was equal to zero. The controller illustrated in Figure 4.7 was implemented.

The theoretical model of the yaw dynamics is

$$\begin{aligned} y_\psi &= \frac{750}{s(s+30)(s+0.7)} \\ y_r &= \frac{750}{(s+30)(s+0.7)} \end{aligned} \tag{4.9}$$

and is derived from a motor with a bandwidth of 30 rad/s, an amplification of 35.7 from the motor signals to angular rate, a factor $s + 1$ corresponding to a weakly attenuated angular rate and a pure integration from angular rate to angle. Figure 4.8 shows the bode plots of the theoretical model and the estimated model. The models share the same crossover frequency and amplitude curves in the entire range. However, the phase differ at low frequencies. Using both models, a more robust controller could be designed. The yaw controller was tuned manually to track the reference while keeping the controller output within reasonable limits. The resulting controller parameters are presented in Table 4.3.

Using the chosen yaw controller, two simulations were performed with both models. The first simulations were done without the integral action, see the top plots in Figure 4.9, and the second ones with the integral action, see the bottom plots. Both simulations are on the closed-loop system with the designed controller using a reference from a logged dataset. When simulating the controller with and without the integral action, the difference was not as apparent as in the roll and pitch cases. However, one can see that there is a clear improvement when performing the larger steps in yaw angular rate for both the estimated model and the theoretical model when the integral action is added.

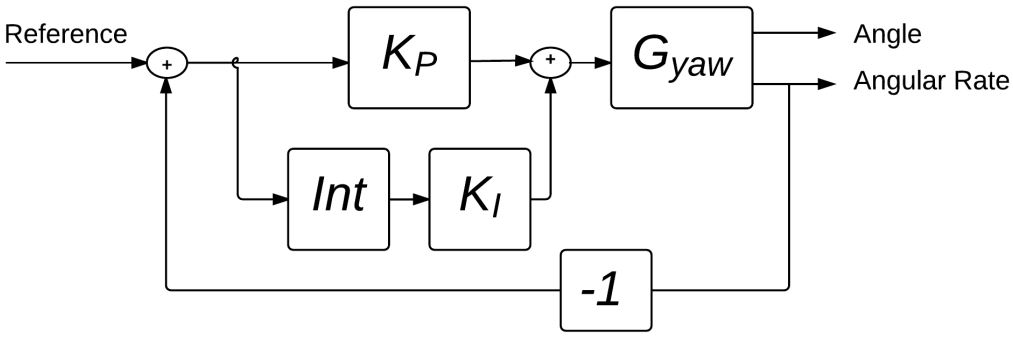



***Figure 4.7:*** *A illustration of the yaw controller. Int is an integrator and G is the quadcopter.*

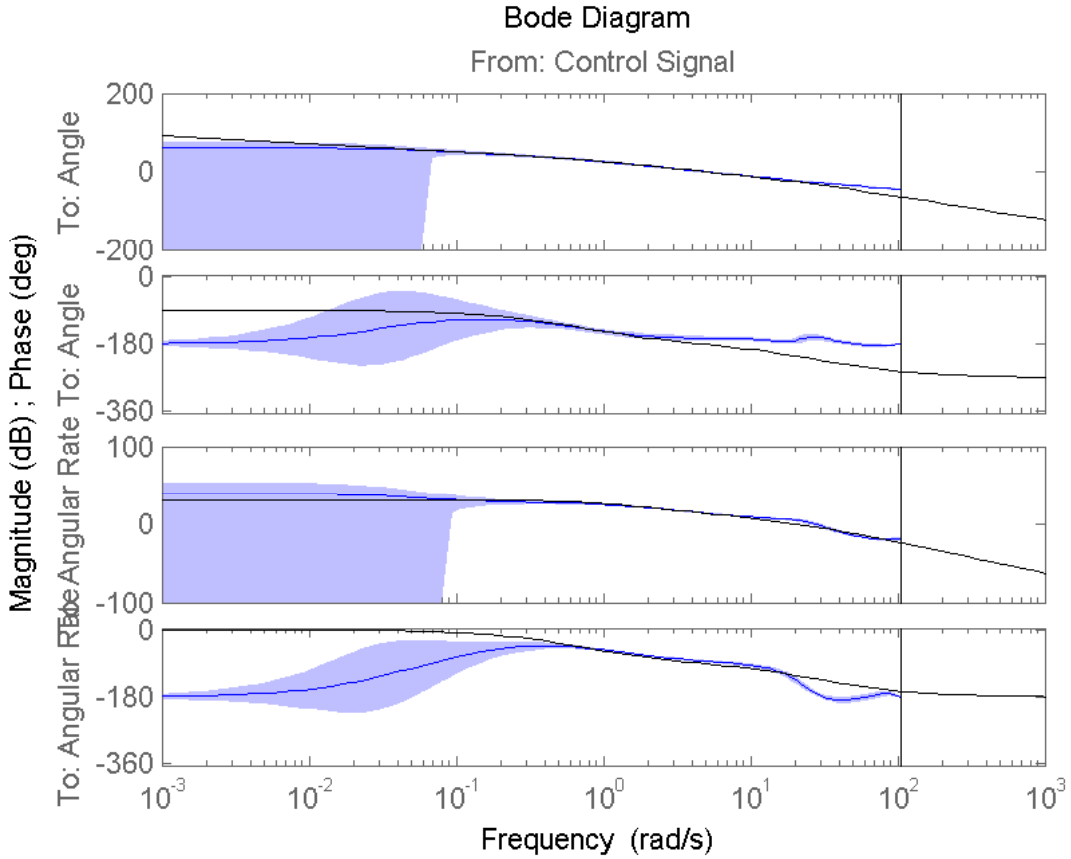
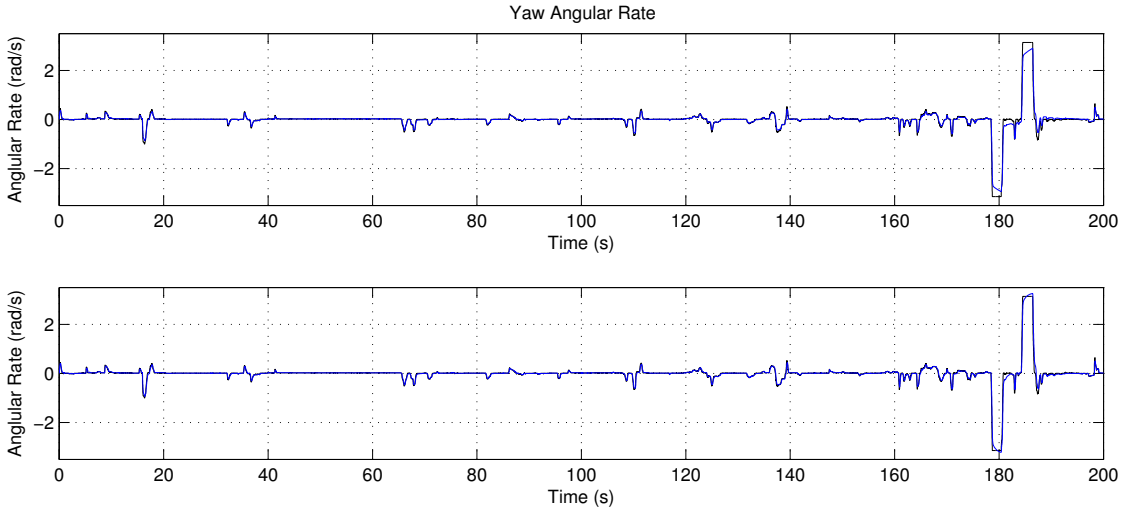


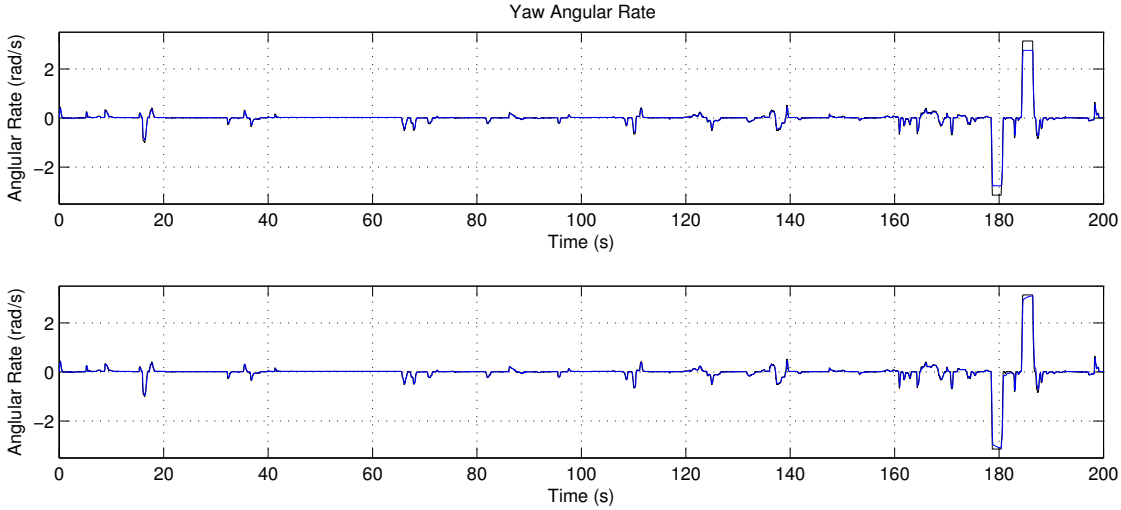

***Figure 4.8:*** *A bode plot comparing the estimated model for yaw, in blue, and the theoretical model, in black, also showing the confidence interval of one standard deviation for the estimated model.*

***Table 4.3:*** *The parameters of the yaw controller*

| Yaw | |
|---|---|
| $K_P$ | 0.2 |
| $K_I$ | 0.2 |




***(a)*** *The estimated model*




***(b)*** *The theoretical model*

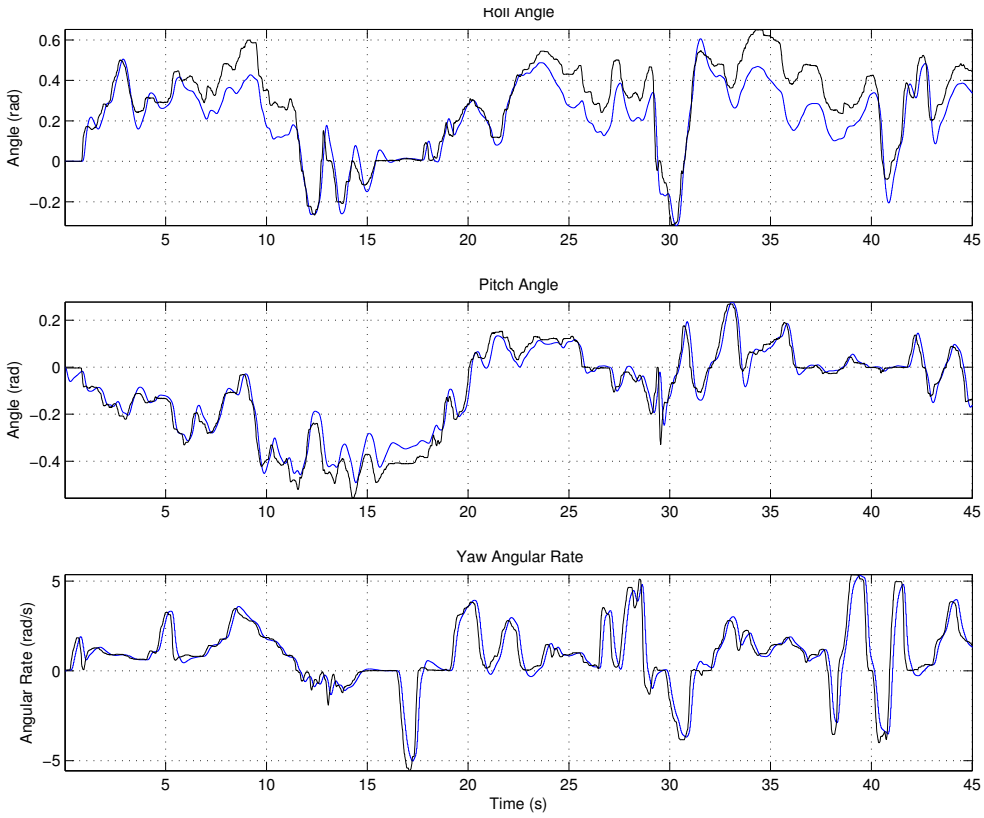
***Figure 4.9:*** *The simulation of the complete closed-loop yaw system, using a reference from a dataset collected during the system identification experiments, showing the performance of the controller. The top plots are with $K_I$ set to zero and the bottom plots are with $K_I$ set according to Table 4.3. Black is the commanded angle and blue is the simulated angle.*

## 4.4 Simulation

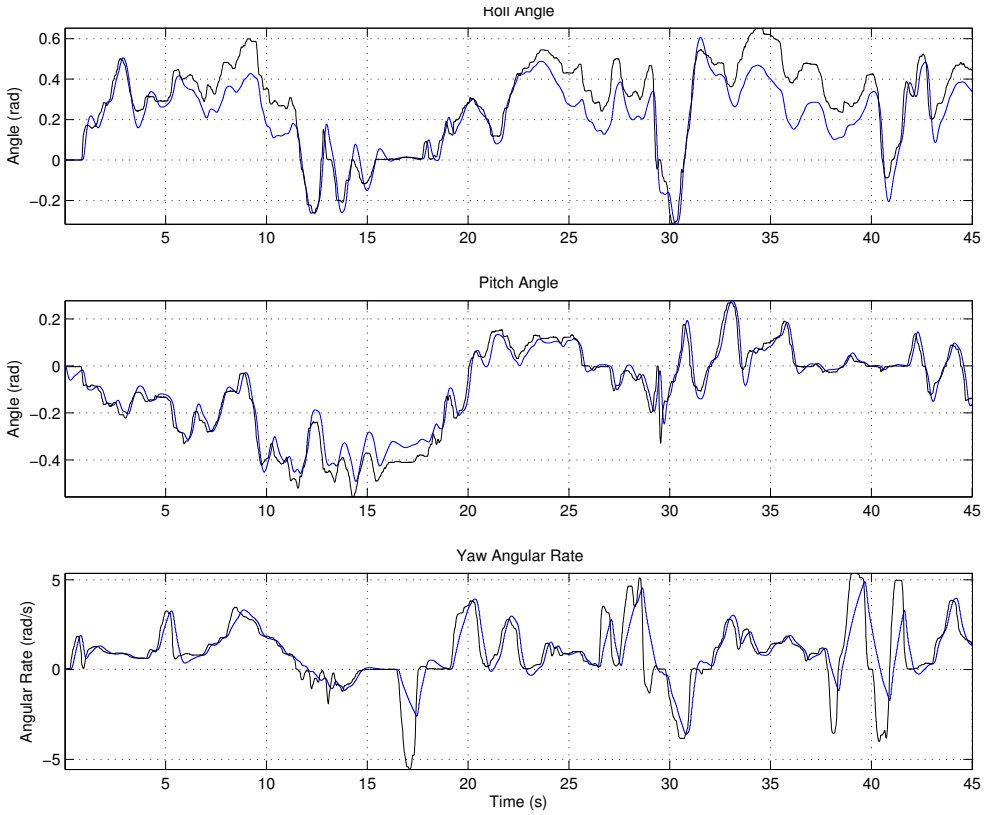
All three controllers have been simulated in Simulink together with the estimated models using the same block that was later used to automatically generate the code that was implemented on the quadcopter. This function contains functionality for integrator anti-windup and controller saturation according to Section 4.2. The throttle was set to a value of 0.6 in both cases, corresponding to an estimate of the amount of throttle necessary to make the quadcopter hover at a constant height.

Figure 4.10a shows the results of the simulation without limitations on the controller outputs. The reference tracking, i.e. tracking of the commanded angles and angular rates, performs especially well in pitch angle and yaw angular rate, and performs well in roll angle too. However, note that there is a steady state error for larger roll angles. This steady state error was expected since it was also seen in Section 4.3.1.

With added limitations, according to Section 4.2, the resulting simulation can be seen in Figure 4.10b. It can be seen that roll and pitch appears to have about the same performance as without the limitations. However, in yaw there is a clear difference and the tracking appears to be much worse. This is a consequence of the chosen prioritization order. Since yaw is prioritized last, the controller output is limited. This performance decline is enlarged by the fact that major manoeuvres are performed in all three directions at the same time.




***(a)*** *No limit on the controller outputs*




***(b)*** *With a limit on the sum of the controller outputs*

***Figure 4.10:*** *The simulation of the three controllers together with (a) and without (b) a limit on the sum of the controller outputs. Black is the commanded angles and angular rates and blue is the simulated angles and angular rates.*

## 4.5 Outdoor and Indoor Flights

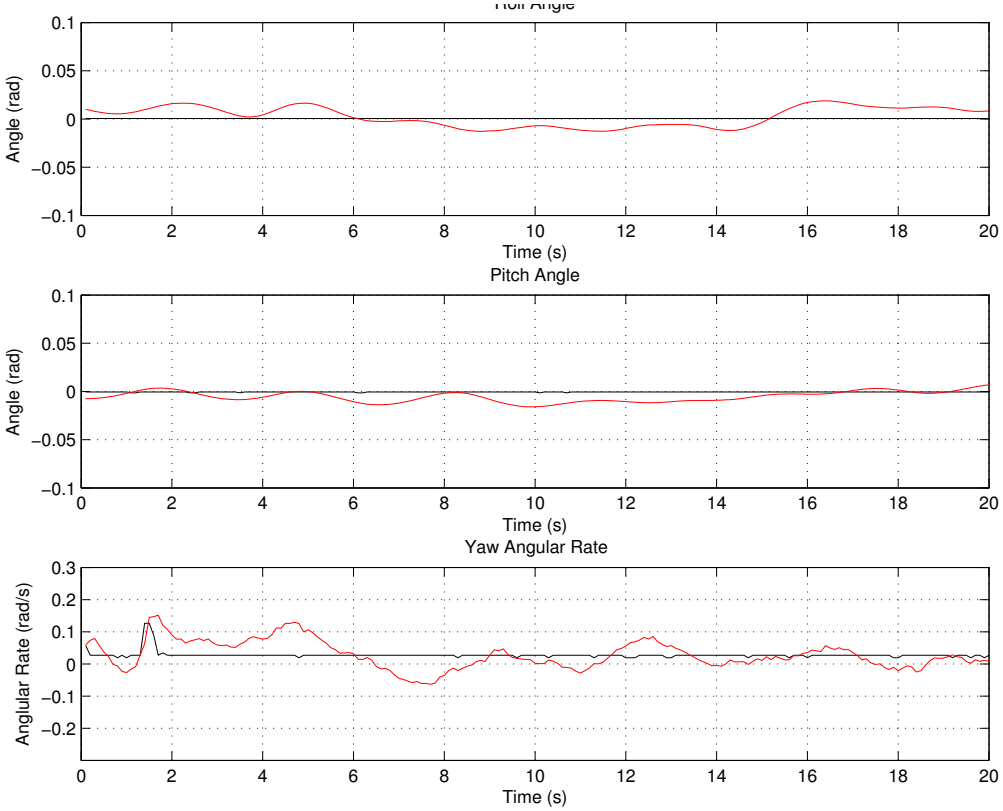
A series of outdoor and indoor flight tests have been performed once the controllers in Section 4.3 had been implemented on the quadcopter and the controllers had been tested in simulations according to Section 4.4. The outdoor tests were performed on a day with a very weak wind of approximately 1 to 2 m/s. As a safety measure, all three controllers were first been tested without the integrator part. The tests were been executed in the low to high risk order, yaw, pitch and last roll, to guarantee that as many tests as possible could be completed. When all three controllers had been tested separately, all three of them were switched on at the same time.

With all three complete controllers active, a hovering manoeuvre was tested. As can be seen in Figure 4.11a, the quadcopter managed to stay in the hovering state for all of the 10 s when the commanded angles and angular rates were equal to zero. However, to keep the quadcopter in the air, the pilot was asked to give as much throttle as necessary to keep the quadcopter at a constant height. As can be seen in the bottom plot of Figure 4.11a, there were small changes in the commanded yaw angular rate. This was the result of yaw and throttle being commanded by using the same stick on the controller. The pilot accidentally gave a non-zero command in yaw. If more flight test were to be carried out, the yaw angular rate could have been programmatically set to zero, when performing a hovering test, to prevent this from happening again.

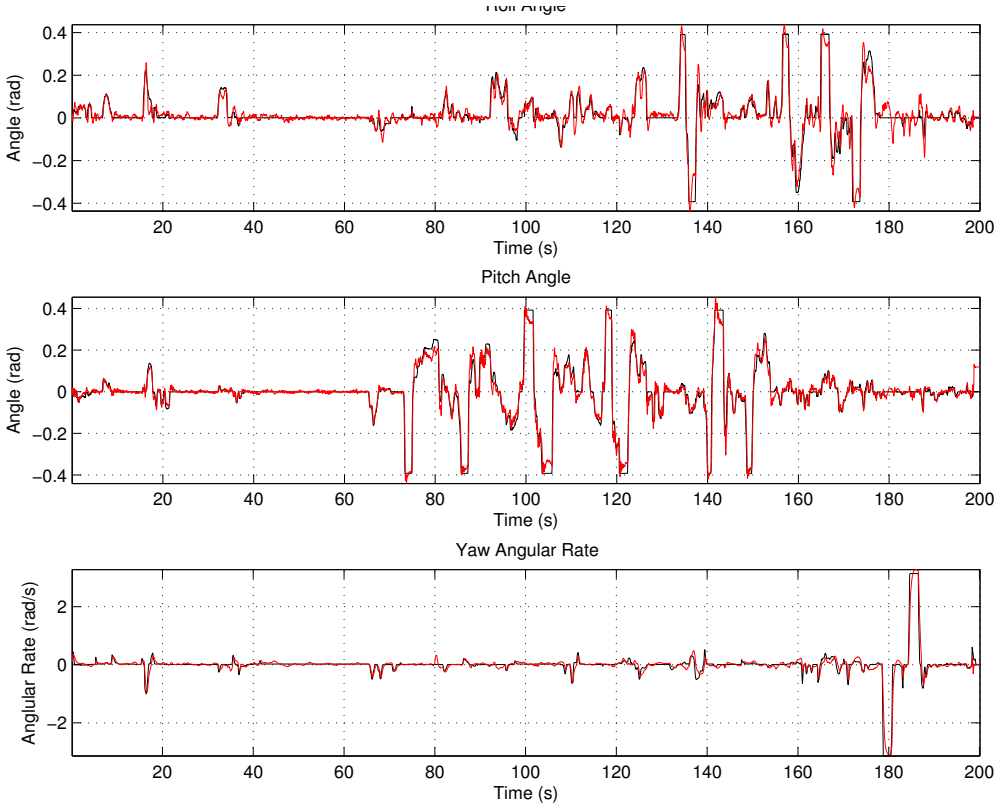
The entire outdoor flight test can be seen in Figure 4.11b. Figure 4.11a is taken from time 40 s to 60 s into the flight. The plot of the full flight shows very good tracking performance in both roll, pitch and yaw. To further evaluate the performance of the controllers, the fit was once again measured. The fit of the simulated signals to the commanded angle and angular rate were 0.6535 for the roll angle, 0.7548 for the pitch angle and 0.6703 for the yaw angular rate.

Next, steps of different magnitudes have been tested in each channel. All step responses have been performed indoors in a large sports center. Figures 4.12 and 4.13 show three steps in roll and pitch of 5°, 10° and 15°. As can been seen, the pitch controller shows a oscillatory behaviour. The roll controller performs relatively well but also shows small oscillations. Previous conclusions found the pitch model to have a slightly worse fit than the roll model. This could be one cause of the oscillatory behaviour of the pitch controller. Another possible cause could be that the simulations in Section 4.4 were performed using the estimated decoupled models. The difference in performance during flight, especially in roll, can be an indication of a potential cross coupling between roll and pitch. If this is the case, there might exist stronger correlations between the roll and pitch channels than previously seen.

Figure 4.14 shows three steps in yaw angular rate of $50°/s$, $100°/s$ and $150°/s$. The yaw controller performed well for all three steps. The step responses of the yaw controller are slower than in the other channels but the oscillations are very small.




*(a) The hovering test*




*(b) The complete flight*

***Figure 4.11:*** *The data from the outdoor flight test. The top plot shows time 40 s to 60 s into the flight and the bottom plot shows the complete flight. Black is the commanded angle and angular rates and red is the measured angles and angular rates.*

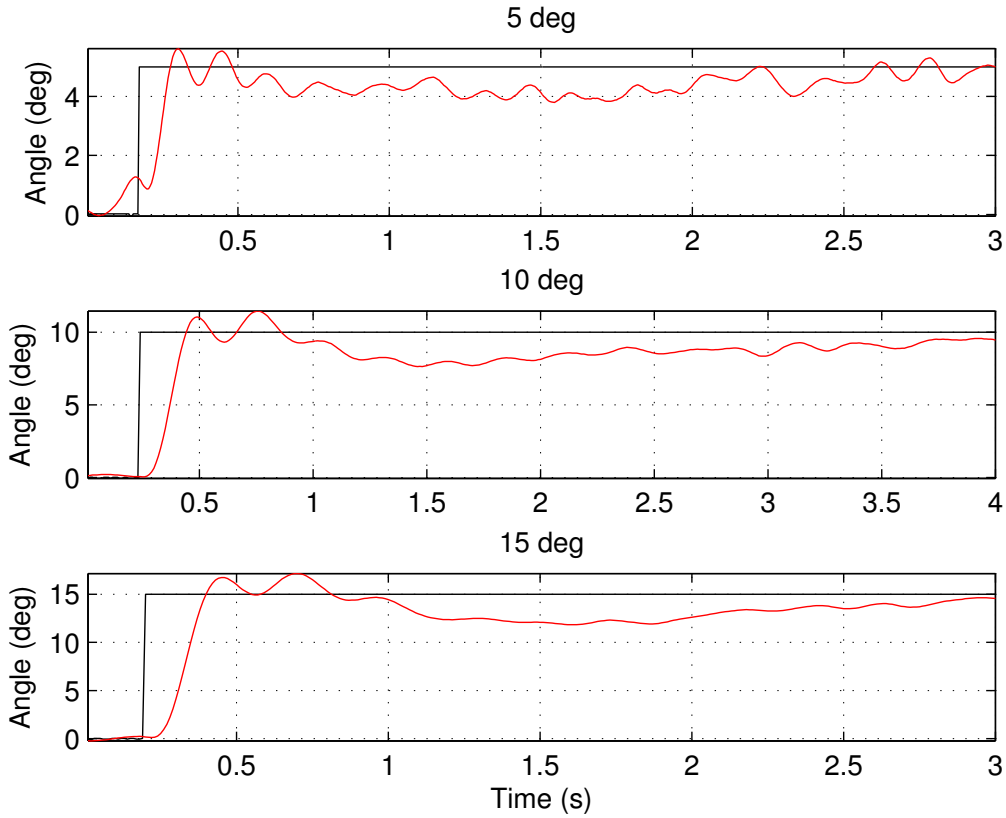



***Figure 4.12:*** *The data from the indoor flight test where three steps in roll were performed. Black is the commanded angle and red is the logged angle.*

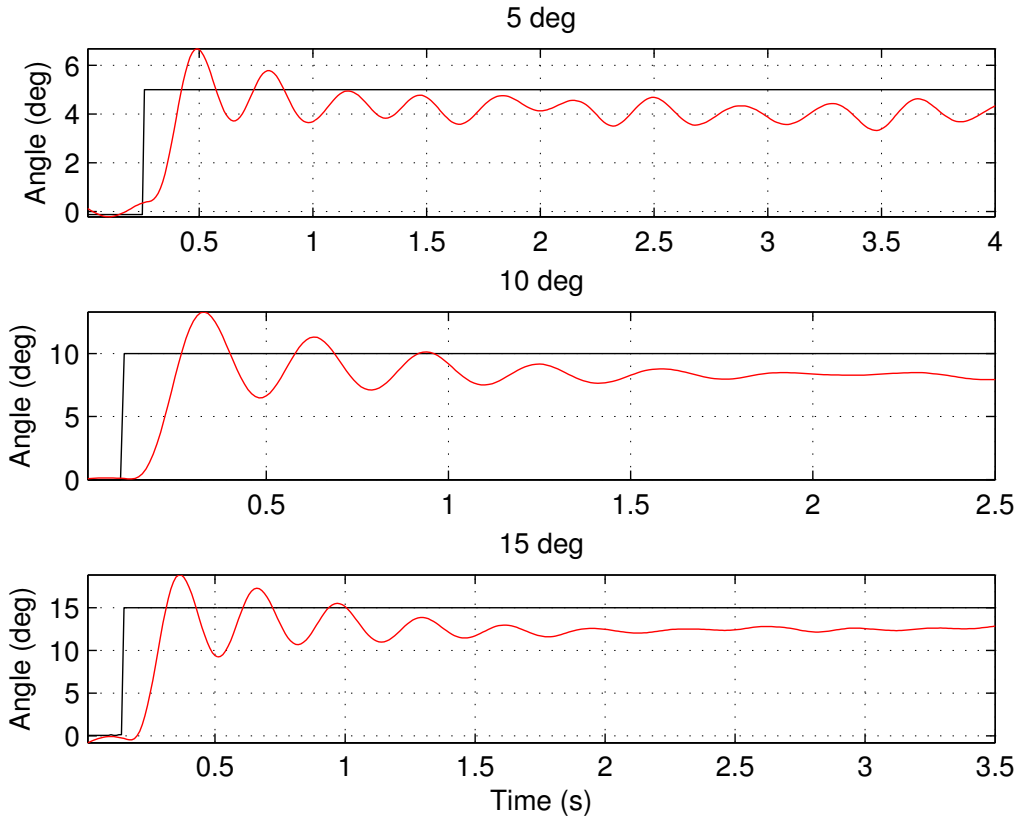



***Figure 4.13:*** *The data from the indoor flight test where three steps in pitch were performed. Black is the commanded angle and red is the logged angle.*

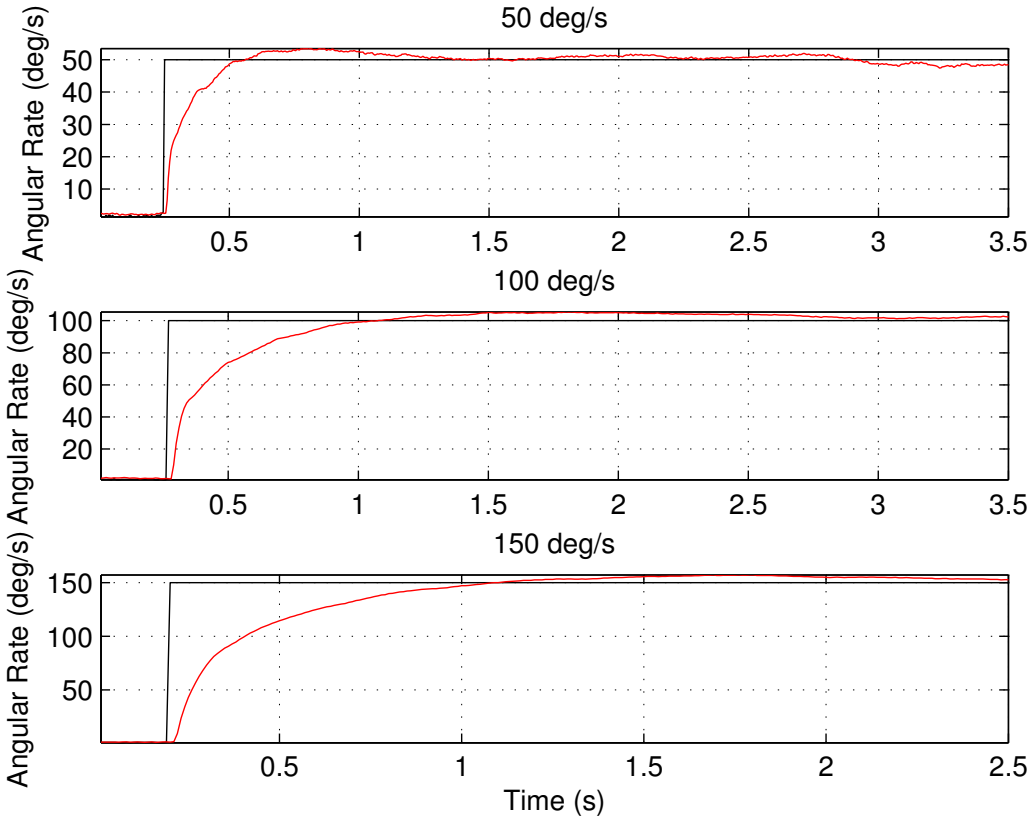



***Figure 4.14:*** *The data from the indoor flight test where three steps in yaw were performed. Black is the commanded angle and red is the logged angular rate.*

# 5

# Conclusions and Future Work

In this thesis, black-box models describing the quadcopter system dynamics for attitude control have been estimated using closed-loop data. The models have shown good simulation performance and they can be used in simulations to properly tune a PID controller for outdoor flight purposes. This chapter gives a summary and a discussion regarding the results presented in Chapter 3 and Chapter 4. This is followed by a presentation of ideas for future development.

## 5.1 Conclusions

The models for the roll and pitch dynamics were forced to have the same model structure and orders since this reflects the geometry of the quadcopter. This approach resulted in the need of an ARMAX model structure for the roll and pitch dynamics, instead of an ARX model structure. This could be an indication of the need for a more flexible noise model. However, the yaw dynamics could be estimated with an ARX model structure. The simulation performance was good for all three models but slightly worse for the pitch model.

The estimation datasets were downsampled to make it easier to obtain informative model estimates. This resulted in higher fit for all the models. Furthermore, although the data collection was performed in closed-loop, the direct method gave good results.

All three models describing the system dynamics were shown to provide accurate enough simulations in order to design the attitude controllers. The attitude controllers performed well during the test flights, and were able to control the quadcopter around the hovering state while only requiring the pilot to handle the throttle to keep the desired altitude. The step responses from the indoor flights show that the performance of the roll and yaw controllers are good. However, the performance of the pitch controller could be improved further since there are

oscillations present.

To conclude, these black-box models and attitude controllers provide a good foundation for future work on the platform. Some potential ideas are presented in the next section.

## 5.2 Future Work

A first improvement would be to include sensors for measuring the longitudinal and lateral position as well as the height above ground, for instance, using a global positioning system (GPS) sensor and an ultrasonic range finder. With a height measurement, a model could be estimated to describe the throttle dynamics. The attitude models and the height model will together describe all of the quadcopter movements. With this complete model and the additional measurements, an interesting problem is to implement a position controller, in order to guarantee the quadcopter from drifting sideways and to allow it to hover, without any input from a pilot.

With the position controller mentioned above implemented, a second idea is to design a control algorithm to allow for fully autonomous flight, including autonomous takeoff and landing. Once this functionality has been implemented, the possibilities for further development are many.

Another improvement would be to modify Algorithms 1 and 2 in Section 4.2 to take the motor signals into account and not only the controller outputs. With this modification the performance of the attitude controllers should be improved.

The attitude controllers in this thesis were designed using simulations. Another interesting improvement would be to use some kind of uncertainty measure of the estimated model, and to use that measure in the controller design to automate the process of designing more robust controllers when the model uncertainty is higher, and to design a more aggressive controller when the model uncertainty is lower, according to Section 4.3. This tool would facilitate the controller design process and make it more automated.

The final idea is to implement a controller with additional robustness properties. One could, for example, design a controller that is robust towards changes in mass or moment of inertia. With this robust controller implemented, the quadcopter could pick up and drop off objects of unknown weights and shapes. Furthermore, it would be interesting to investigate if a less oscillatory controller can be designed and how the pitch channel influences the roll channel.

# Bibliography


Fabio Beltramini, Marco Bergamasco, and Marco Lovera. Experiment design for MIMO model identification, with application to rotorcraft dynamics. In *Proceedings of 18th IFAC World Congress*, pages 14392–14397, Milano, Italy, 2011. Cited on page 2.

Mark Cutler. Design and control of an autonomous variable-pitch quadrotor helicopter. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, August 2012. Cited on page 1.

Urban Forssell and Lennart Ljung. Closed-loop identification revisited. *Automatica*, 35(7):1215–1241, July 1999. ISSN 0005-1098. Cited on page 18.

T. Glad and L. Ljung. *Modeling of dynamic systems*. Prentice Hall, 1995. Cited on page 15.

T. Glad and L. Ljung. *Reglerteknik, Grundläggande teori*. Studentlitteratur AB, 4:7 edition, 2006. ISBN 9789144022758. Cited on page 34.

S. Gupte, P.I.T. Mohandas, and J.M. Conrad. A survey of quadrotor unmanned aerial vehicles. In *Southeastcon, 2012 Proceedings of IEEE*, pages 1–6, Orlando, FL, USA, March 2012. doi: 10.1109/SECon.2012.6196930. Cited on page 1.

Qianying Li. Grey-box system identification of a quadrotor unmanned aerial vehicle. Master's thesis, Delft University of Technology, The Netherlands, 2014. Cited on page 2.

Lennart Ljung. *System Identification: Theory for the user*. PTR Prentice Hall Information and System Sciences Series., 2nd edition, 1999. Cited on page 15.

Lennart Ljung. Black-box models from input-output measurements. In *Instrumentation and Measurement Technology Conference, 2001. IMTC 2001. Proceedings of the 18th IEEE*, volume 1, pages 138–146 vol.1, May 2001. doi: 10.1109/IMTC.2001.928802. Cited on page 2.

Lennart Ljung. *System identification toolbox: user's guide*. MathWorks, 2014b edition, 2014. Cited on page 18.

Z. Mustapa, S. Saat, S.H. Husin, and N. Abas. Altitude controller design for multicopter UAV. In *Computer, Communications, and Control Technology (I4CT), 2014 International Conference on*, pages 382–387, Sept 2014. doi: 10.1109/I4CT.2014.6914210. Cited on page 1.

K. Nagarjuna and G.R. Suresh. Design of effective landing mechanism for fully autonomous unmanned aerial vehicle. In *Signal Processing, Communication and Networking (ICSCN), 2015 3rd International Conference on*, pages 1–6, Madras Institute of Technology, Anna University, Chennai, India, March 2015. doi: 10.1109/ICSCN.2015.7219864. Cited on page 1.

J. Nikolic, M. Burri, J. Rehder, S. Leutenegger, C. Huerzeler, and R. Siegwart. A UAV system for inspection of industrial facilities. In *Aerospace Conference, 2013 IEEE*, pages 1–8, March 2013. doi: 10.1109/AERO.2013.6496959. Cited on page 1.

P. Panizza, F. Riccardi, and M. Lovera. In *Workshop on Advanced Control and Navigation for Autonomous Aeroespace Vehicles*, Seville, Italy, June 2015. doi: 10.1016/j.ifacol.2015.08.060. Cited on page 2.

S. Thornton and J. Marion. *Classical Dynamics of Particles and Systems*. Brooks/Cole, Cengage Learning, 2004. ISBN 9780534408961. Cited on page 9.

S. Waharte and N. Trigoni. Supporting search and rescue operations with UAVs. In *Emerging Security Technologies (EST), 2010 International Conference on*, pages 142–147, Sept 2010. doi: 10.1109/EST.2010.31. Cited on page 1.

Yang Yungao, Xian Bin, Yin Qiang, Li Haotao, and Zeng Wei. Current research situation of frameworks and control of quadrotor unmanned aerial vehicles. In *Control Conference (CCC), 2011 30th Chinese*, Shandong, China, 2011. Cited on page 1.

Hanlin Zhang, Sixiao Wei, Wei Yu, E. Blasch, Genshe Chen, Dan Shen, and K. Pham. Scheduling methods for unmanned aerial vehicle based delivery systems. In *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd*, pages 6C1–1–6C1–9, Oct 2014. doi: 10.1109/DASC.2014.6979499. Cited on page 1.

Hägglund Tore Åström, Karl J. *Advanced PID Control*. ISA, 2006. Cited on page 35.