Institutionen för systemteknik
Department of Electrical Engineering

**Examensarbete**

# Model Predictive Control of a Tricopter

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

**Karl-Johan Barsk**

LiTH-ISY-EX--12/4607--SE

Linköping 2012


**Linköpings universitet**
**TEKNISKA HÖGSKOLAN**

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

# Model Predictive Control of a Tricopter

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan vid Linköpings universitet
av

**Karl-Johan Barsk**

LiTH-ISY-EX--12/4607--SE

Handledare: **Niklas Wahlström**
ISY, Linköpings universitet

Examinator: **Daniel Axehill**
ISY, Linköpings universitet

Linköping, 20 juni 2012

**Avdelning, Institution**
Division, Department

Department of Electrical Engineering
Department of Electrical Engineering
SE-581 83 Linköping

**Datum**
Date

2012-06-20

**Språk**
Language

□ Svenska/Swedish
☒ Engelska/English
□ _____

**Rapporttyp**
Report category

□ Licentiatavhandling
☒ Examensarbete
□ C-uppsats
□ D-uppsats
□ Övrig rapport
□ _____

**ISBN**
—

**ISRN**
LiTH-ISY-EX--12/4607--SE

**Serietitel och serienummer**
Title of series, numbering

**ISSN**
—

**URL för elektronisk version**

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-79066

**Titel**
Title

Modellprediktiv reglering av en tricopter
Model Predictive Control of a Tricopter

**Författare**
Author

Karl-Johan Barsk

**Sammanfattning**
Abstract

In this master thesis, a real-time control system that stabilizes the rotational rates of a tricopter, has been studied. The tricopter is a rotorcraft with three rotors.

The tricopter has been modelled and identified, using system identification algorithms. The model has been used in a Kalman filter to estimate the state of the system and for design of a model based controller.

The control approach used in this thesis is a model predictive controller, which is a multivariable controller that uses a quadratic optimization problem to compute the optimal control signal. The problem is solved subject to a linear model of the system and the physical limitations of the system.

Two different types of algorithms that solves the MPC problem have been studied. These are explicit MPC and the fast gradient method. Explicit MPC is a pre-computed solution to the problem, while the fast gradient method is an online solution.

The algorithms have been simulated with the Kalman filter and were implemented on the microcontroller of the tricopter.

**Nyckelord**
Keywords

MPC, explicit MPC, fast gradient, Kalman filter, PEM, tricopter

## Abstract

In this master thesis, a real-time control system that stabilizes the rotational rates of a tricopter, has been studied. The tricopter is a rotorcraft with three rotors.

The tricopter has been modelled and identified, using system identification algorithms. The model has been used in a Kalman filter to estimate the state of the system and for design of a model based controller.

The control approach used in this thesis is a model predictive controller, which is a multivariable controller that uses a quadratic optimization problem to compute the optimal control signal. The problem is solved subject to a linear model of the system and the physical limitations of the system.

Two different types of algorithms that solves the MPC problem have been studied. These are explicit MPC and the fast gradient method. Explicit MPC is a precomputed solution to the problem, while the fast gradient method is an online solution.

The algorithms have been simulated with the Kalman filter and were implemented on the microcontroller of the tricopter.

# Acknowledgments

I would like to begin to show my gratitude for my parents and my brother, for their support during the thesis work. They have always tried to push me further when the work has felt hopeless, but also tried to limit my effort, when there have been late nights and stressful moments. It has been a secure feeling to know that they have been there as support.

My examiner, Daniel, have been very helpful with my thesis work, by pushing me further, inspiring me not to give up and the discussions of different issues with the control algorithms. He has also been helpful with the different flight experiments and the try out of the implemented control algorithms. My supervisor, Niklas, has been very helpful with discussion of the different solutions and by reviewing the report very carefully. He has reminded me to be in the readers clothes, when describing the different methods. I like to thank both of you, for your patience.

There have been a lot of issues with the hardware, mainly, with the Atmega2560 but also with programming skills. By solving these problems, Joakim Gebart has been a great source of information to the different problems. I could not have solved some of the problems without your tips.

I would like to show my gratitude to my fellow companions, who also have been placed in B-huset's dark corridors. They have been patient with my complainant when there were issues with the hardware, but we have also had nice "fika" moments.

At last, I would like to thank the Department of Automatic Control for letting me do the master thesis at the department. But also the time when I was an assistant in basic course of automatic control and introductory course for MATLAB. I have learned a lot during this time and I do not regret anything.

*Linköping, June 2012*
*Karl-Johan Barsk*

# Contents

# Notation

### Symbols, Operators and Functions

| Notation | Denotation |
|---|---|
| $x(t)$ | Time continuous representation of variable $x$ at time $t$. |
| $\dot{x}(t)$ | Time derivative of $x(t)$. |
| $x_k$ | Time discrete representation of variable $x$ at time $kT$. |
| $\hat{x}_{k+1\|k}$ | Estimation of x at time $k+1$ given measurement at time $kT$. |
| $E(x)$ | Expected value of x. |
| $Cov(x)$ | Covariance of $x$. |
| | |
| $\mathbf{R}$ | Reference signal to a system. |
| $\mathbf{U}$ | Input signal to a system. |
| $\mathbf{U}_v$ | Virtual input signal to a system. |
| $\mathbf{Y}$ | Output signal to a system. |
| $\mathbf{X}$ | States of a state-space system. |
| | |
| $\mathfrak{B}$ | (Denotation of the) Local coordinate system. |
| $\mathfrak{E}$ | (Denotation of the) Geographical coordinate system. |
| $\mathfrak{G}$ | (Denotation of the) Global coordinate system. |
| $(x, y, z)^j$ | Translational position in coordinate system $j$. |
| $(u, v, w)^j$ | Translational velocity in coordinate system $j$. |
| $(\phi, \theta, \psi)$ | Euler angles (Roll, Pitch, Yaw) which describes the rotation of the local coordinate system in the global. |
| $(p, q, r)^j$ | The angular rates of coordinate system $j$. |
| $Q_i^v$ | Rotation around axis $i$ by an angle $v$. |
| $\lambda_{lat}$ | Geographical coordinate in degrees that specifies north-south position. |
| $\lambda_{lng}$ | Geographical coordinate in degrees that specifies east-west position. |
| | |
| $f_i$ | Thrust force from rotor $i$. |
| $\tau_i$ | Torque induced by rotor $i$. |
| $k_t$ | Thrust constant. |
| $k_\tau$ | Torque constant. |
| $\alpha$ | Tilt angle of the back rotor. |

### Abbreviations and Acronyms

| Notation | Denotation |
|---|---|
| APM1 | Ardu Pilot Mega 1 the main board |
| BIBO | Bounded-Input Bounded-Output |
| DOF | Degrees Of Freedom |
| EEPROM | Electrically Erasable Programmable Read Only Memory |
| ESC | Electronic Speed Control |
| GPS | Global Positioning System |
| IMU | Inertial Measurement Unit |
| LP | Low Pass |
| MPC | Model Predictive Control |
| PEM | Prediction Error Method |
| PI | Proportional-Integral |
| RAM | Random Access Memory |
| RC | Radio Control |
| SPI | Serial Peripheral Interface |
| UAV | Unmanned Aerial Vehicle |

# 1

# Introduction

To facilitate missions in hazardous environments, flying platforms that are small, agile and are able to take of vertically are of interest. A platform that fulfills these requirements is an UAV (Unmanned Aerial Vehicle) in the form of a multicopter combined with a good control system. A multicopter is a rotorcraft that has more than two rotors, because a rotorcraft with two rotors is called helicopter (bicopter). Multicopters have fixed blades with a pitch that are not possible to control, as it does for a helicopter (through the swashplate), to control the direction of the rotor thrust. instead, the speed of the rotors are varied to achieve motion control of a multicopter. For a tricopter, there is also a servo attached that can tilt one of the motors and by that achieve a change in motion.

Multirotor aircrafts are often used in model and radio controlled projects because of the simple construction and control. Due to the number of rotors, the size of them does not need to be large in comparison with a helicopter that only has one rotor to induce enough force to lift it up.

UAVs are often used in places where it is difficult for a man to operate in such as hazardous environments, steep terrain etc. It is also a suitable and a cheap tool for surveillance. A camera mounted on a multicopter is a very flexible way to survey an area. UAVs can easily be designed for a specific mission and as such it is a very flexible tool.

This thesis is about stabilizing a tricopter, see Figure 2.1. It is an aircraft with three rotors and a tail servo. This has the advantage of a helicopter with quick yaw movements, but also the advantage of a quadcopter that is a more robust platform with its four rotor blades than a helicopter is with its only main rotor to induce thrust force. One disadvantage with the tricopter is that it is a very cross-linked system, which will be explained in Chapter 3.

The control method that will be used in this thesis is based on a dynamical model of the tricopter. The method chosen in this project is MPC (Model Predictive Controller), Maciejowski [2001]. Two types of algorithms are studied to solve this problem. One offline solution and one online solution.

## 1.1 Problem formulation

The master thesis work is done on assignment of the Automatic Control Division at the Department of Electrical Engineering at Linköping University. The aim is to create a control system for a tricopter, as the one in Figure 2.1, that as a primary objective stabilizes the orientation and as a secondary objective also controls the position of the aircraft, while explicitly taking the limits of the control signals into account. A control strategy that could be used to satisfy these specification is a MPC. This controller needs a model of the system to calculate the control signal. That means that the tricopter has to be described by differential, or difference, equations and the model parameters of the tricopter have to be estimated.

The proposed controller has been implemented on an Atmel AtMega2560, which has limited computational resources. This means that an online algorithm that solves the MPC problem in a computationally efficient manner, has to be used. Two such alternatives that will be investigated further are the fast gradient method, Richter et al. [2011], and explicit MPC, Bemporad et al. [2002].

## 1.2 Related work

Flying machine is not a new invention and especially a helicopter, which has quite similar dynamics as the rotorcraft covered in this thesis. There has been some modeling of multicopters, such as a helicopter (bicopter), tricopter and quadcopter, in previous research. The articles [Cai et al., 2006] and [Kim et al., 2002] are about how a helicopter is modeled and identified. In [Castillo et al., 2004] and [Mistler et al., 2001] the articles covers the modeling and control system design of a quadcopter. These articles have some interesting aspects such as the way how the coordinate systems are defined and how the models are structured and identified. Since the platform of the aircraft is different, the dynamical equations are not of interest. Furthermore, there are articles that covers modeling and controlling of a tricopter. In the articles [Salazar-Cruz et al., 2008] and [Yoo et al., 2010] a tricopter is modeled and identified.

## 1.3 Thesis outline

The thesis is organized as follows:

- Chapter 3 presents how to define the dynamical equation that describes the movements of the aircraft.
- Chapter 4 deals with the identification of the model that describes the movements of the tricopter.
- Chapter 5 presents methods to filter out noise. A linear frequency selective filter and a filter based on the system model are discussed.
- Chapter 6 deals with the definition of the control algorithms.
- Chapter 7 presents the results from the data collection, system identification and simulation of the controller.
- Chapter 8 presents the conclusions that have been taken regarding the results.
- Chapter 8.2 presents what can be done in the future considering the results from this thesis.

# 2

# Tricopter

The tricopter that has been used in this thesis is a small model rotorcraft with three arms that has a brushless electrical motor attached to each one of them. These arms are attached to a plate and the arms are shaped as the letter Y, where the angle between any two arms is 120° and the length of the arms is 0.50 m. One of the motors is attached to a servo that can tilt the motor. A picture of the tricopter can be seen in Figure 2.1. The control loop consists of an inner and an outer loop. The controller in this thesis uses only the inner loop to stabilize the rotational rates of the tricopter. That is because the microcontroller did not have enough computational resource to handle both the inner and outer loop. On each of the motors a rotor is attached. These rotor blades have fixed angles

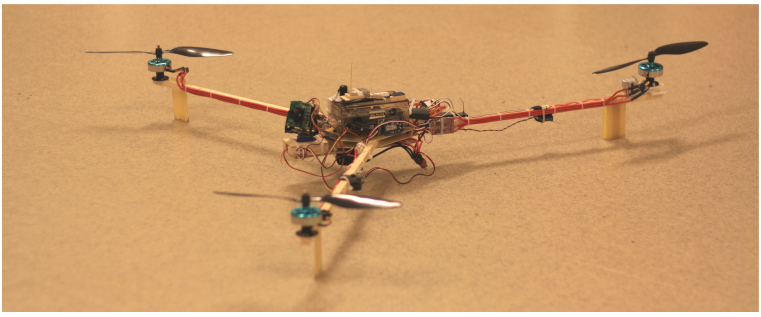


***Figure 2.1:*** *Tricopter that has been used in this thesis work.*

and therefore the airflow depends on the direction of the rotation of the rotor blade. Each of these blades can been seen as identical which results in the same rotational direction and same airflow for a given angular rate.

## 2.1 System overview

The tricopter that is used in this thesis has a hardware platform named ArduPiloMega1 ,APM1, from DIY Drones, DIY Drones [2012a], which includes an ArduPilot, IMU-oilpan, barometer, GPS and a sonar sensor. The ArduPilot has an Atmel AtMega2560 microcontroller, Atmel Inc. [2012], with an open-source control program called ArduCopter, DIY Drones [2012b]. This is an auto-pilot with a PI-controller that uses reference signals from a radio controller, RC, to compute the pulse-width modulated, PWM, signals to three electrical speed controllers, ESC, and to a servo. These control the rate of each electrical motor, and the angle of the servo that can tilt one of the motors. This platform is upgraded with an AtMega2560 microcontroller which has 256kB flash program memory instead of 128kB program memory as the regular AtMega1280 has. The microcontroller has 16MHz clock frequency. The APM1 has also a 16MB electrically erasable programmable read-only memory, EEPROM, which is used to log data and save calibration parameters and settings for the control system. The memory is connected with a serial peripheral interface, SPI. A view over the system can be seen in Figure 2.2.

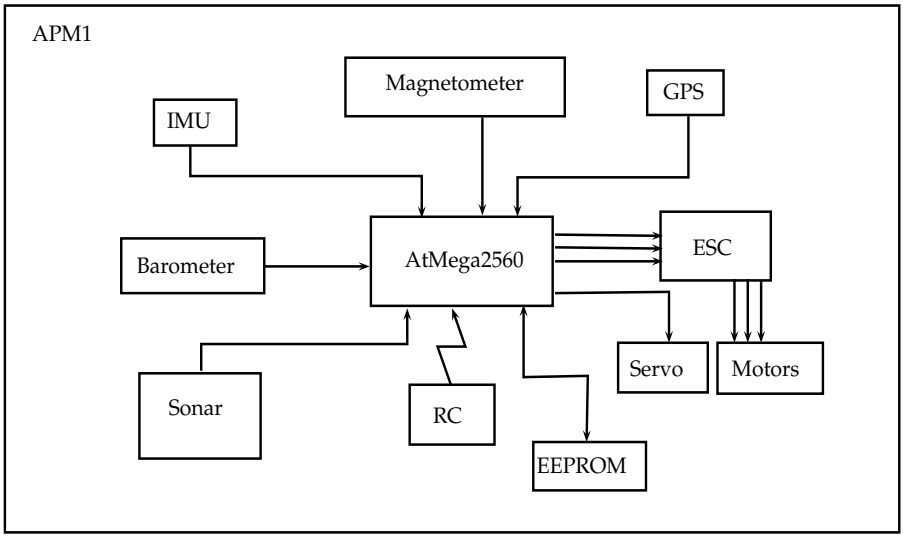



***Figure 2.2:*** *System overview of the hardware where all the sensors are connected to the AtMega2560 microcontroller. The ESC receive PWM signals from the microcontroller and sends analog signals to the motors and the servo. An external memory, EEPROM, is connected to the microcontroller and used for storing data from logging functions.*

### 2.1.1 RC

The radio controller is used to send reference values to the tricopter. Since the controller in this thesis controls the rotational rates of the tricopter, these reference values will be the rotational rates of the tricopter (roll, pitch and yaw). These are the three reference signals to the controller. There is also a throttle that only controls the rates of the electrical motors equally. The throttle does not affect the servo. This gives that there are four signals that can be controlled with the two sticks on the RC, see Figure 2.3. The throttle, the roll rate, the pitch rate and the yaw rate

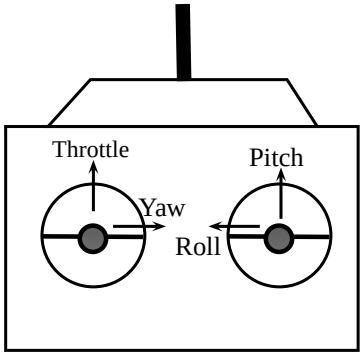



***Figure 2.3:*** *RC control with two sticks, where each stick can move in two directions. The left stick controls the throttle, which controls all the electrical motors equally, and the yaw rate of the tricopter. The right stick controls the roll and pitch rates of the tricopter.*

### 2.1.2 ESC

The ESC is used to control the brushless electrical motors. The motors take an analog signal as input but the microcontroller does not have enough power to control the motors, which means that a external controller must be used. The ESC gets an digital PWM, Pulse Width Modulated, signal which it then convert into an analog electrical signal to the motors. The ESC takes input values between 1000 and 2000, which is the width of the pulse in microseconds $[\mu s]$. The pulse width 1000 corresponds to motors shutdown and 2000 to full speed. The PWM signal to the servo is limited to $(1200, 1800)$ $[\mu s]$, where 1500 $[\mu s]$ centers the servo, since there is a physical limitation of the rotational angle of the servo.

## 2.2 Sensor module

The sensors that are attached on the tricopter are all connected to the microcontroller. These together form a sensor module that measures the movement of the tricopter. These measurements are used to get information of the tricopter's location and orientation, to be able to compute and apply the correct input signals

to achieve the desired behavior of the rotorcraft. The sensors that are attached on the tricopter are: inertal measurement unit (IMU), magnetometer, global positioning system (GPS), barometer and ultrasonic sensor. All the sensor measurements are made by the ArduCopter software.

### 2.2.1 IMU

The IMU-oilpan is an IMU which consists of an accelerometer and a gyroscope. The accelerometer measures the proper acceleration that the tricopter is experiencing. The gyroscope measures the angular velocities of the tricopter. By integrating the angular velocities, the orientation of the tricopter can be obtained. There is a problem by doing this, since it will result in a drift in the estimation of the orientation of the tricopter. This is due to the bias error in the gyroscope and by integrating the constant error it will result in a linear function and the estimation of the rotation will drift. The rotational rates are presented in radians per second $[rad/s]$.

### 2.2.2 Magnetometer

The magnetometer is used to measure the strength and direction of the magnetic field of the Earth. This can be used to compute in which direction, along the Earth's surface, in which the tricopter is pointing at. Because the estimation of the orientation will drift as mentioned above, the result will be that the orientation of the tricopter cannot be determined accurately with an IMU. This drift can be compensated by combining the estimated orientation of the tricopter from the IMU with the measurement from the magnetometer. Since the magnetometer is a compass, the information is presented as an angle $[rad]$ the tricopter is rotated in relation to the magnetic field of the Earth.

### 2.2.3 GPS

The GPS, Global Positioning System, sensor measures the position of the tricopter and the position is presented in geographical coordinates, i.e. longitude and latitude [°], and if the altitude is used from GPS measurements, it is given in altitude [m] above sea level. The GPS also measures the absolute velocity of the rotorcraft, and is given in meters per second [m/s].

### 2.2.4 Sonar

The sonar sensor is an ultrasonic sensor that can measure distances by sending a high frequency sound wave and then receive the echo of the signal. By calculating the time interval between sending the signal and receiving the signal the distance to the nearest object can be determined.

### 2.2.5 Barometer

The barometer is a sensor that can measure the atmospheric pressure. Since the atmospheric pressure is decreasing with increasing altitude this sensor can be used to calculate the altitude $[m]$ of the tricopter in relation to the sea level. The

problem with this sensor is that it is not very accurate. The atmospheric pressure depends on the humidity of the air, temperature, wind etc. so the barometer has to be very accurately calibrated to get good measurements.

A summary of which signals these sensors are measuring:

- The position of the tricopter given in geographical coordinates.
- The altitude of the tricopter in relation to the ground.
- The absolute translational velocity of the tricopter.
- The direction of the tricopter i relation the magnetic field of the Earth.
- The rotational rates of the tricopter.

## 2.3 Control loop

To be able to fly a model-sized tricopter, some kind of controller is needed to stabilize the system and balance the rotor velocities so that the system follows reference values that are sent from a RC. That is because the dynamics of the system are very fast, cross-linked and an open loop system will be unmanageable for a person to control.

The ArduCopter already includes this kind of autopilot, but it is a PI, Proportional-Integral, controller that stabilizes the angular rates and will therefore be changed to a controller based on the model of the system, considering its limitations.

The control loop of the tricopter can be seen as one inner and one outer loop, see Figure 2.4. The inner loop is a faster one and controls the rotational rates of the tricopter. The frequency of this loop is 50Hz, which gives a hard deadline of 20 ms the loop has to compute the input signal to the system. The outer loop is a slower one and this controls translational position, translational velocity and rotational angles of the tricopter. In this thesis, only the inner loop is considered.

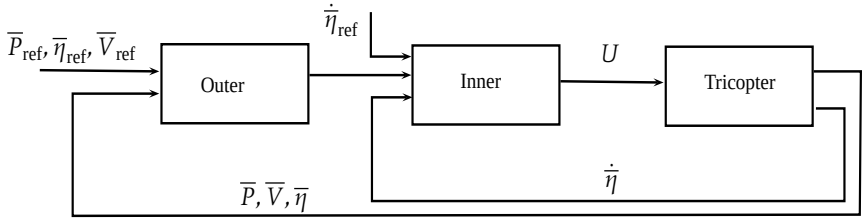



***Figure 2.4:*** *Closed loop for a tricopter with an inner and an outer controller. $\dot{\overline{\eta}}$ are the rotational rates of the tricopter, $\overline{\eta}$ are the rotational angles of the tricopter, $\overline{P}$ is the position of the tricopter and $\overline{V}$ is the velocity of the tricopter. The index ref denotes the reference signal.*

This thesis has two objectives, one primary and one secondary. In the primary objective the orientation of the tricopter is stabilized, and in the secondary objective both the orientation and position are stabilized. In the primary objective

the inner-loop will consist of the rate controller that will be defined in Chapter 6, and the pilot is the outer loop. The pilot is closing the loop by visually observing the tricopters orientation, position and velocity, and then sends reference values of the rotational rates to the controller of the tricopter to obtain the desired behavior of the rotorcraft. The rotational rates are measured and controlled by the control system.

In the secondary objective, all the measurements are used to stabilize both the position and orientation, and the autopilot controls both the outer and inner loop.

# 3

# System modeling

To be able to use the control methods in this thesis, a dynamical model of the tricopter needs to be derived. This means that the tricopter has to be mathematically described by either differential or difference equations. To be able to derive the force, kinematic and kinetic equations the coordinate system, input and output signals have to be defined.

The primary objective is to stabilize the rotational rates of the tricopter in the inner loop, see Figure 2.4, while the pilot stabilizes the orientation of the tricopter. It is therefore reasonable to define a model structure that only describes the angular rates of the aircraft. This gives a simple (*black-box*) model of the tricopter. Another way to represent the *black-box* model is to use the virtual signals as input to the model that controls the roll, pitch and yaw rates, respectively. This gives a virtual (*black-box virtual*) model of the system.

The secondary objective of the work is to stabilize both the position, the translational velocity, the rotational angles and the rotational rates of the tricopter. By that the dynamical model has to include the position and the velocity of the tricopter, but also the orientation of the tricopter compared to the *black-box* model. This extended model, in relation to the *black-box* and the *black-box virtual*, can be described with the *physical* model of the tricopter by deriving the force, kinetic and kinematic equations. The kinetic equations will be the same for the *black-box*, the *black-box virtual* and the *physical* model, which is the description of the rotational rates of the physical system.

The outline of this chapter is firstly to introduce the tricopter as a system in different coordinate systems to be able to define in- and output signals. When these are defined, the relation between the input and output of the tricopter is defined by a state space model. The structure of this state space model is defined

in three ways: A physical model that is derived by the laws of physics, a black-box model where only input and outputs signals are known and a virtual model that is similar as the black-box model, but it is combined with a linear transformation of the input signals. Further on in this thesis, only the *black-box* and the *black-box virtual* model are used, and the *physical* model acts to describe cross-coupled the system is.

## 3.1 System

To be able to derive a mathematical model of the tricopter, it has first to be defined as a system. The in- and output signals have to be defined, where output signals are the signals that can be measured with the sensors defined in Chapter 2. Before we start modeling, the coordinate systems have to be defined such that the output signals can be expressed.

### 3.1.1 Coordinate system

There are three different coordinate systems of interest that need to be defined. The local, which is seen as the body fixed coordinates of the tricopter and denoted $\mathfrak{B}$ and the global, which is the coordinate system of the Earth (environment) and denoted $\mathfrak{G}$. A third coordinate system is also introduced to describe the geographical coordinates, longitude and latitude, which the GPS uses for positioning. This coordinate system is denoted $\mathfrak{E}$.

The tricopter has three arms formed as a Y, and a rotor is placed at the end of each arm. The coordinate system for the tricopter will be defined as in Figure 3.1. The $X^{\mathfrak{B}}$-axis is defined in the direction straight ahead seen from the view of the tricopter, and $Y^{\mathfrak{B}}$-axis to the right. The $Z^{\mathfrak{B}}$-axis is defined straight down from the center of mass of the tricopter. The $X^{\mathfrak{B}}$-axis can be seen as the desired forward direction during a flight. Since the rotor in the back (denoted no. 1) is attached to a servo that can tilt an angle $\alpha$, this angle has to be defined. The angle $\alpha$ is defined positive when the rotor is tilted to the right, see Figure 3.1.

The Earth's coordinate system is defined such that the $X^{\mathfrak{G}}$-axis is pointing north, $Y^{\mathfrak{G}}$-axis points to the west and the $Z^{\mathfrak{G}}$-axis points upwards, and can be seen in Figure 3.2b.

The $\mathrm{X}^{\mathfrak{G}}$-, $\mathrm{Y}^{\mathfrak{G}}$- and $\mathrm{Z}^{\mathfrak{G}}$-axes follows the surface of the Earth. It gives that the vector from a position $P$ on the Earth's surface to the North Pole overlaps with the $X^{\mathfrak{G}}$-axis, and the vector that goes through the point $P$ and is parallel to the equator, overlaps with the $\mathrm{Y}^{\mathfrak{G}}$-axis. Longitude, $\lambda_{\mathrm{lng}}$, is defined as zero degrees at the Royal Obeservatory in Greenwitch and is defined with positive degrees to the East of the zero point. Latitude, $\lambda_{\mathrm{lat}}$, is defined as zero degrees at the equator and is defined with positive degrees to the North of the equator. The geographical coordinates longitude and latitude are introduced to be able to describe a position, $P = (\lambda_{\mathrm{lat}}, \lambda_{\mathrm{lng}})^{\mathfrak{E}} = (x, y)^{\mathfrak{G}}$, on the Earth's surface. This coordinate system is seen as a third one and is only a different way to describe the position of the

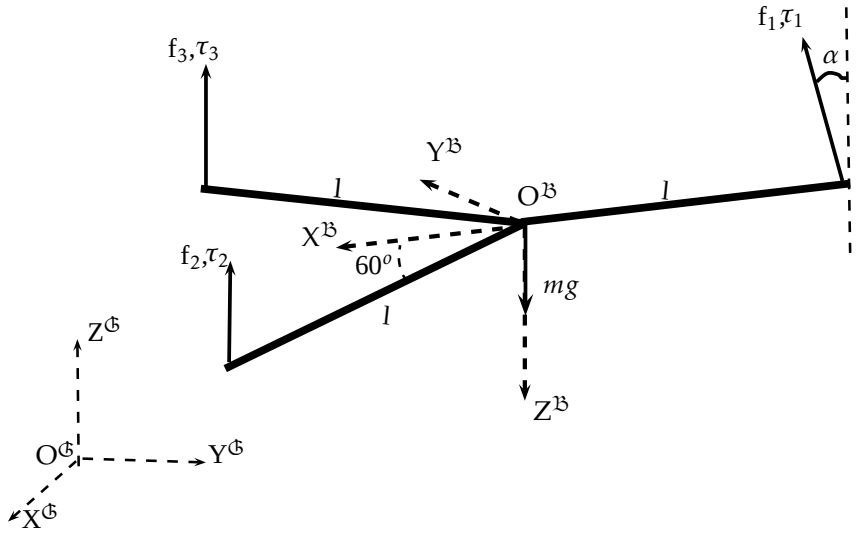
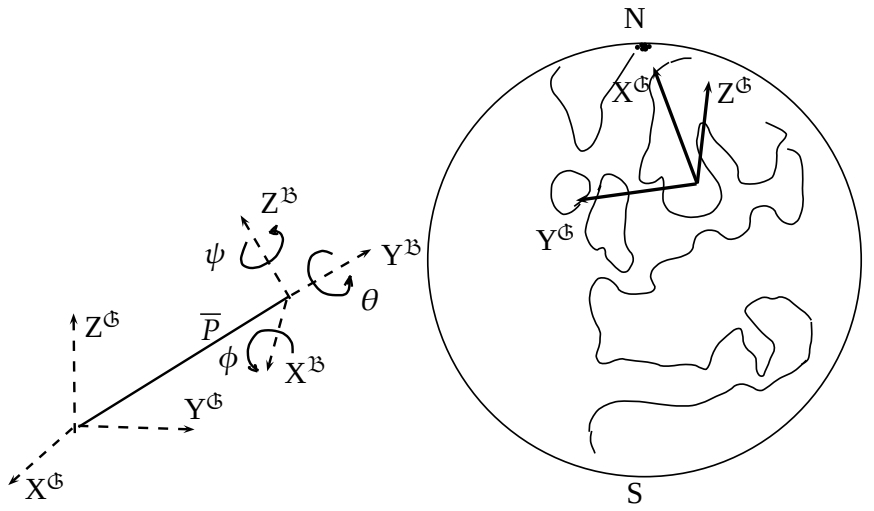



***Figure 3.1:*** *A tricopter with the local coordinate system $\mathfrak{B}$ and the global coordinate system $\mathfrak{G}$. $l$ denotes length, $f$ force, $\tau$ torque and $\alpha$ angle.*




***(a)*** *Coordinate systems* ***(b)*** *Earth coordinate system*

***Figure 3.2:*** *Relationship between the Earth (Global) coodinate system $\mathfrak{G}$ and aircrafts (Local) coordinate system $\mathfrak{B}$. $\overline{P} = (x, y, z)$ is the translation of the coordinate system $\mathfrak{B}$ related to system $\mathfrak{G}$. $\mathfrak{B}$ is rotated with $(\phi, \theta, \psi)$ related to $\mathfrak{G}$. North Pole is denoted with $N$ and South Pole with $S$.*

tricopter. To be able to use the coordinate system $\mathfrak{E}$ with the model since the GPS measures the position in coordinate system $\mathfrak{E}$, a transformation has to be done to

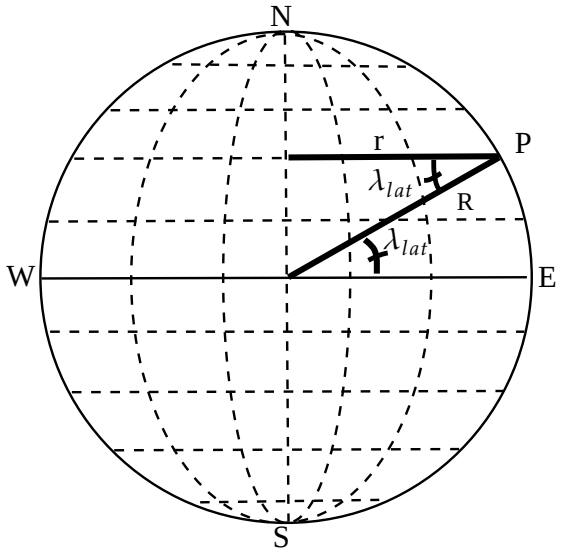



***Figure 3.3:*** *Geographic coordinates. Constant latitude is shown as lines that go from North to South. Constant longitude is shown as lines that go from West to East.*

the coordinate system $\mathfrak{G}$.

The circumference of a circle is $S = 2\pi R$ where R is the radius of the circle. By just taking a part of a circle, an angle $\lambda$ radian, the curve length can be calculated as

$$S = \frac{\lambda}{2\pi} 2\pi R = \lambda R \tag{3.1}$$

The radius $r$ at a point $P$ on the surface to the North-South-axis depends on the latitude $\lambda_{\text{lat}}$. This radius $r$ can be expressed as $r = R_{Earth}\cos(\lambda_{\text{lat}})$, see Figure 3.3. The degrees longitude and latitude have to be transformed to radian instead of degrees. The Earth can be approximated as spherical with radius $R_{Earth} = 6371.0[km]$, which gives that a position $P(x, y)$ on the Earth's surface, Figure 3.3, can be calculated as

$$x = \lambda_{\text{lat}} \frac{\pi}{180} R_{Earth} \tag{3.2a}$$

$$y = \lambda_{\text{lng}} \frac{\pi}{180} R_{Earth} \cos(\lambda_{\text{lat}} \frac{\pi}{180}). \tag{3.2b}$$

How the tricopter's coordinate system is related to the coordinate system of the Earth is illustrated in Figure 3.2a and the relation can be described in a mathematical way with the rotation matrix $Q_{xyz}^{\phi\theta\psi}$. The rotation is represented with Euler angles which is represented with $(\phi, \theta, \psi)$ and is the angle around $X^{\mathfrak{G}}$, $Y^{\mathfrak{G}}$, $Z^{\mathfrak{G}}$-axes, respectively. The rotation is applied to each of the base vectors and the

total rotation is done by first rotating the $Z^{\mathfrak{B}}$-axis with an angle $\psi$, then rotating the $Y^{\mathfrak{B}}$-axis with an angle $\theta$ and at last the $X^{\mathfrak{B}}$-axis with an angle $\phi$

$$Q_{xyz}^{\phi\theta\psi} = Q_{\phi}^{x} Q_{\theta}^{y} Q_{\psi}^{z} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} \mathrm{C}_\theta \mathrm{C}_\psi & \mathrm{C}_\theta \mathrm{S}_\psi & -\mathrm{S}_\theta \\ \mathrm{S}_\phi \mathrm{S}_\theta \mathrm{C}_\psi - \mathrm{C}_\phi \mathrm{S}_\psi & \mathrm{S}_\phi \mathrm{S}_\theta \mathrm{S}_\psi - \mathrm{C}_\phi \mathrm{C}_\psi & \mathrm{S}_\phi \mathrm{C}_\theta \\ \mathrm{C}_\phi \mathrm{C}_\theta \mathrm{C}_\psi + \mathrm{S}_\phi \mathrm{S}_\psi & \mathrm{C}_\phi \mathrm{S}_\theta \mathrm{S}_\psi - \mathrm{S}_\theta \mathrm{C}_\psi & \mathrm{C}_\psi \mathrm{C}_\theta \end{pmatrix} \tag{3.3}$$

where $\mathrm{S}_x$, $\mathrm{C}_x$, $\mathrm{T}_x$ denotes $\sin x$, $\cos x$, $\tan x$, respectively and $Q_i^v$ is the rotation around axis $i$ with an angle $v$. This will map the local coordinates $\mathfrak{B}$ to the global coordinates $\mathfrak{G}$. The rotations around $X^{\mathfrak{B}}$, $Y^{\mathfrak{B}}$, $Z^{\mathfrak{B}}$-axis are denoted as roll, pitch and yaw, respectively.

### 3.1.2 Input and output signals

To be able to control the tricopter and determine a dynamical model of it, the in- and output signals of the system have to be defined.

The tricopter has three electrical motors with rotors which will create an airflow which induces a thrust force to lift the tricopter. This force, $f$, can be approximated as proportional to the square of the angular velocity of the rotating blades i.e. $f = k_t \omega^2$, Kiusaalas and Pytel [2001]. The torque can also be expressed in same way as the thrust force, $\tau = k_\tau \omega^2$. Since the blades have fixed angles, the direction of the airflow depends on the rotational direction of the blade. Therefore the force $f_i$ and torque $\tau_i$ are expressed as

$$f_i = k_t \omega_i |\omega_i| \tag{3.4a}$$
$$\tau_i = k_\tau \omega_i |\omega_i|, \quad \text{for } i = 1, 2, 3. \tag{3.4b}$$

This gives that the angular velocity of the rotor $\omega_i$ is an input signal to the system and denoted as $\mathbf{U}_i$, $i = 1, 2, 3$.

There is a problem with a tricopter, or in general with a multicopter that has odd number of rotors. Each rotor will induce a reaction torque that is equal to the torque generated by the rotating blades but with opposite direction, $\tau_{rt} = -k_\tau \omega_i |\omega_i|$. If the sum of the reaction torques from the rotors will be zero, the induced torque applied on the mass center will be zero and it means that the tricopter will not spin around the $Z^{\mathfrak{B}}$-axis. The total sum of the reaction torque for $N$ rotors can be expressed as $T_{rt} = \sum_{i=1}^{N} \tau_{rt,i}$. If there are odd amount, $M$, of rotors and each rotor has the same rotational rate $\omega_1 = \omega_2 = \omega_3$, which is the case when the rotorcraft is hovering, the total reaction torque, $T_{rt}$, will be non-zero. This is because the terms cannot cancel each other and means that a multicopter like a tricopter will spin around its mass center. It can be compensated by tilting one of the rotors, which in this case will be the back rotor, see Figure 3.1.

The angular velocities and the tilt angle will give the tricopter four signals to

control its movement: three rotor velocities and the tilting angle of the back rotor. These input signals to the system will affect the orientation and translation of the rotorcraft. By approximating the tricopter as a rigid body with 6 degrees of freedom, DOF, the states of the system that describes the motion can be defined as $(x, y, z)^{\mathfrak{G}}, (u, v, w)^{\mathfrak{G}}, (\phi, \theta, \psi)^{\mathfrak{G}}, (p, q, r)^{\mathfrak{B}}$ . These are the (translational) position and velocity of the tricopter in the coordinate system $\mathfrak{G}$, but also the rotational angles in $\mathfrak{G}$ and angular rates in $\mathfrak{B}$, respectively.

The output signals describe which of the states that are measured. By using the sensor measurements, described in Section 2.2, the angular rates, $(p, q, r)^{\mathfrak{B}}$, can be measured with the gyroscope. The yaw orientation $\psi^{\mathfrak{G}}$ can be measured with the magnetometer and that is how the tricopter is oriented in relation to the Earth's magnetic field, which is assumed parallel with the $X^{\mathfrak{G}}$-axis. The GPS measures the translational position $(\lambda_{\text{lat}}, \lambda_{\text{lng}})^{\mathfrak{E}}$, and the altitude $z^{\mathfrak{G}}$ can be measured either with the barometer or the sonar sensor. The GPS also estimates the absolute translational velocity $|V|^{\mathfrak{G}}$

This gives that the outputs are $(\lambda_{\text{lat}}, \lambda_{\text{lng}})^{\mathfrak{E}}, z^{\mathfrak{G}}, |V|^{\mathfrak{G}}, \psi^{\mathfrak{G}}, (p, q, r)^{\mathfrak{B}}$

### 3.1.3 Electrical motor

The input signals to the system are the angular velocities of each rotor and the tilt angle of the servo, defined in Section 3.1.2. These signals are controlled with three ESCs and a servo by a digital PWM signal from the microcontroller. The output from the motor is the rotational velocity and the output from the servo is the tilt angle. The motor and the servo have to be modelled to find out how the *physical* model order will increase if the model of the electrical motor and servo are included.

The rotors are driven by brushless electric motors. The input signal to a DC motor is an analog electrical voltage which causes the motor to rotate. The rotating blades create an airflow which then induces forces and torques. These forces and torques are input signals to the system. The servo, that tilts the rear motor, is also a DC motor where the angle of the motor is controlled. The differential equations of a DC motor are according to [Glad and Ljung, 2004]

$$u(t) = R_1 i(t) + L_1 \frac{d}{dt} i(t) + k\omega(t) \tag{3.5a}$$

$$J \frac{d}{dt} \omega(t) = ki(t) - f\,\omega(t), \tag{3.5b}$$

where the inductance $L_1$ of the coil is assumed to be zero. The input voltage to the system is $u(t)$ and the output $y(t)$ is the angle of the motor. The rotational rate of the motor is $\omega(t)$ and the current through the motor is $i(t)$. The resistance of the wire is $R_1$ and the constant k, with dimension [Vs/rad], is the transformation between the electrical and the mechanical part of the system . The mechanical friction of the motor is $f$. The system can be written in state space representation

as

$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & \frac{-1}{C_1} \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ \frac{C_2}{C_1} \end{pmatrix}$$
$$y(t) = \begin{pmatrix} 1 & 0 \end{pmatrix} x(t) \tag{3.6}$$

where

$$C_1 = \frac{JR_1}{fR_1 + k^2}$$
$$C_2 = \frac{k}{fR_1 + k^2}. \tag{3.7}$$

This gives that the transfer function between input voltage and output angle is

$$G_{servo}(s) = \frac{C_2}{s(1 + sC_1)}. \tag{3.8}$$

The output of the system is the angle of the motor. As the transfer function (3.8) shows, there are two poles. The servo controls the angle of the motor and therefore it can be described with this transfer function.

The angular velocity is controlled of the DC motor and not by the angle of the motor, such as the servo. This gives that the output of the system is the angular velocity of the DC motors, and the transfer function is

$$G_{rotor}(s) = s\frac{C_2}{s(1 + sC_1)} = \frac{C_2}{1 + sC_1} \tag{3.9}$$

and the system has only one stable pole (this gives that the output, angular rate $\omega$, will be constant if the input voltage is constant). The constant, $C_1$, is assumed to be very small so that the system is seen as static, i.e. $y(t) \approx C_2 u(t)$. The assumption is made to reduce the complexity of the model.

## 3.2 Physical motion model

To be able to express the dynamical model of the aircraft, the translational and rotational equations have to be derived. The nonlinear system structure are described in [Glad and Ljung, 2003] and this gives that the dynamics of the tricopter can be expressed with differential equations and these equations will have the form

$$\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{U})$$
$$\mathbf{Y} = h(\mathbf{X}), \tag{3.10}$$

where $\mathbf{X}$ is the state space vector, $\mathbf{U}$ is the input vector and $\mathbf{Y}$ is the output vector. The state space vector includes the position (translation) of the tricopter, the translational velocity, the orientation of the aircraft related to Earth and the rotational velocity related to Earth. This will give that the state space vector has the

following states

$$\mathbf{X} = [(x, y, z)^{\mathfrak{G}}, (u, v, w)^{\mathfrak{G}}, (\phi, \theta, \psi)^{\mathfrak{G}}, (p, q, r)^{\mathfrak{B}}]^T. \tag{3.11}$$

All the states are presented in the global coordinate system i.e. the Earth's coordinate system, $\mathfrak{G}$, which can been seen in Figure 3.2b, except for the rotational velocities which are the angular rates in the local coordinate system $\mathfrak{B}$. The input vector includes the input signals described in Section 3.1.2.

$$\mathbf{U} = [\omega_1, \omega_2, \omega_3, \alpha]^T, \tag{3.12}$$

where $\omega_i$ is the rotational velocity of rotor $i$ and $\alpha$ is the tilt angle of rotor 1.

The output vector $\mathbf{Y}$ includes which states are measured, described in Section 3.1.2, by the sensors described in Section 2.2

$$\mathbf{Y} = [\lambda_{lat}, \lambda_{lng}, z^{\mathfrak{G}}, |V|^{\mathfrak{G}}, \psi^{\mathfrak{G}}, (p, q, r)^{\mathfrak{B}}]^T, \tag{3.13}$$

where $|V|^{\mathfrak{G}}$ is the absolute velocity of the tricopter that is measured with the GPS and $\psi^{\mathfrak{G}}$ is the measured direction of the tricopter measured with the magnetometer. The sensors which measures the states were defined in Chapter 2.

### 3.2.1 Free-body diagram

By their definitions, the external forces that are acting on the aircraft and the torque that is induced by the external forces acting on the mass center, are defined in the coordinate system $\mathfrak{B}$. With a free body diagram, seen in Figure 3.1, and the geometry of the tricopter, see Figure 3.4, these forces $F$ and torques $M$ are defined as

$$F_{ext}^{\mathfrak{B}} = \begin{pmatrix} F_x^{\mathfrak{B}} \\ F_y^{\mathfrak{B}} \\ F_z^{\mathfrak{B}} \end{pmatrix} = \begin{pmatrix} 0 \\ k_t\omega_1|\omega_1|\sin\alpha \\ -k_t(\omega_1|\omega_1|\cos\alpha + \omega_2|\omega_2| + \omega_3|\omega_3|) \end{pmatrix} \tag{3.14a}$$

$$M_{F_{ext}}^{\mathfrak{B}} = \begin{pmatrix} M_x^{\mathfrak{B}} \\ M_y^{\mathfrak{B}} \\ M_z^{\mathfrak{B}} \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{3}}{2} l k_t(\omega_2|\omega_2| - \omega_3|\omega_3|) \\ \frac{1}{2} l k_t(\omega_2|\omega_2| + \omega_3|\omega_3|) - l k_t\omega_1|\omega_1|\cos\alpha + k_\tau\omega_1|\omega_1|\sin\alpha \\ -l k_t\omega_1|\omega_1|\sin\alpha - k_\tau(\omega_1|\omega_1|\cos\alpha + \omega_2|\omega_2| + \omega_3|\omega_3|) \end{pmatrix} \tag{3.14b}$$

where the force $F$ is separated in $F_x, F_y, F_z$ components and the torque is separated in $M_x, M_y, M_z$ components. The position and the translational velocity of the aircraft are described in the coordinate system $\mathfrak{G}$, where $X^{\mathfrak{G}}$-axis is to the north, $Y^{\mathfrak{G}}$-axis to the west and $Z^{\mathfrak{G}}$-axis straight up. That means that the force vector that is defined in the coordinate system $\mathfrak{B}$ has to be rotated so that the forces can be described in the coordinate system $\mathfrak{G}$. The transformation from the coordinate system $\mathfrak{B}$ to $\mathfrak{G}$ is made by rotation matrix $Q_{xyz}^{\phi\theta\psi}$ (3.3), where $\phi$, $\theta$, $\psi$ are the Euler angles for rotation around $X^{\mathfrak{G}}$-, $Y^{\mathfrak{G}}$-, $Z^{\mathfrak{G}}$-axis, respectively. The force vector described in the coordinate system $\mathfrak{G}$ is described as

$$F_{ext}^{\mathfrak{G}} = Q_{xyz}^{\phi\theta\psi} F_{ext}^{\mathfrak{B}} \tag{3.15}$$

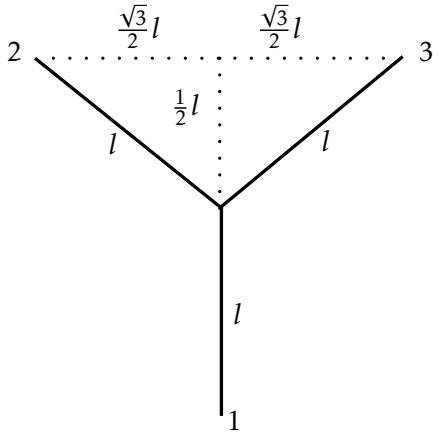



***Figure 3.4:*** *Geometry of a tricopter where $l$ denotes the length of the arms. The number $1, 2, 3$ corresponds to motor $1, 2, 3$, respectively.*

### 3.2.2 Translation

The external forces and the torques, which are induced by the forces with acting on the mass center $O^{\mathfrak{B}}$, have now been defined. The next step is to derive the differential equations of the system. The second law of Newton gives the force equation, Kiusaalas and Pytel [2001]. Mass multiplied with the time derivative of the translational speed vector is equal the directional force vector, $m\dot{V}^{\mathfrak{G}} = \sum F$. The gravitational force is separated from the externals because it is acting on the mass center and is therefore not inducting any torque on the mass center. The force equations are

$$m\dot{V}^{\mathfrak{G}} = \sum F = F^{\mathfrak{G}} + F_g = Q_{xyz}^{\phi\theta\psi} F_{ext}^{\mathfrak{B}} + \begin{pmatrix} 0 \\ 0 \\ -mg \end{pmatrix}$$

$$\begin{aligned}
\dot{u}^{\mathfrak{G}} &= \frac{1}{m}\Big(F_{X,ext}^{\mathfrak{B}} \mathrm{C}_\theta \mathrm{C}_\psi + F_{Y,ext}^{\mathfrak{B}} \mathrm{C}_\theta \mathrm{S}_\psi - F_{z,ext}^{\mathfrak{B}} \mathrm{S}_\theta\Big) \\
\dot{v}^{\mathfrak{G}} &= \frac{1}{m}\Big(F_{X,ext}^{\mathfrak{B}} (\mathrm{S}_\phi \mathrm{S}_\theta \mathrm{C}_\psi - \mathrm{C}_\phi \mathrm{S}_\psi) + F_{Y,ext}^{\mathfrak{B}} (\mathrm{S}_\phi \mathrm{S}_\theta \mathrm{S}_\psi + \mathrm{C}_\theta \mathrm{C}_\psi) + F_{Z,ext}^{\mathfrak{B}} \mathrm{S}_\phi \mathrm{C}_\theta\Big) \\
\dot{w}^{\mathfrak{G}} &= \frac{1}{m}\Big(F_{X,ext}^{\mathfrak{B}} (\mathrm{C}_\phi \mathrm{S}_\theta \mathrm{C}_\psi + \mathrm{S}_\phi \mathrm{S}_\psi) + F_{Y,ext}^{\mathfrak{B}} (\mathrm{C}_\phi \mathrm{S}_\theta \mathrm{S}_\psi - \mathrm{S}_\phi \mathrm{C}_\psi) + F_{Z,ext}^{\mathfrak{B}} \mathrm{C}_\phi \mathrm{C}_\theta\Big) - g
\end{aligned} \tag{3.16}$$

where $\mathrm{S}_x$, $\mathrm{C}_x$, $\mathrm{T}_x$ denotes $\sin x$, $\cos x$, $\tan x$, respectively, and the relation between position and translational velocity is given by

$$\begin{aligned}
\dot{x}^{\mathfrak{G}} &= u^{\mathfrak{G}} \\
\dot{y}^{\mathfrak{G}} &= v^{\mathfrak{G}} \\
\dot{z}^{\mathfrak{G}} &= w^{\mathfrak{G}}
\end{aligned} \tag{3.17}$$

and the absolute velocity is computed as

$$|V|^{\mathfrak{G}} = \sqrt{(u^{\mathfrak{G}})^2 + (v^{\mathfrak{G}})^2 + (w^{\mathfrak{G}})^2}. \tag{3.18}$$

### 3.2.3 Rotation

The differential equations for the angular rates can be determined from the torque equations, by assuming that the tricopter is a rigid body and there are external forces acting on the center of gravity. These are expressed with respect to the body frame, Mistler et al. [2001],

$$J\dot{\Omega} + \Omega \times J\Omega = M_F^{\mathfrak{B}}$$
$$\Omega = (p^{\mathfrak{B}}, q^{\mathfrak{B}}, r^{\mathfrak{B}})^T,$$

where $J$ is an inertia matrix and is assumed to be diagonal, just to simplify the equations and the model. This equation can be written as

$$\begin{aligned}
\dot{p}^{\mathfrak{B}} &= \frac{(I_{yy} - Izz)}{I_{xx}} r^{\mathfrak{B}} q^{\mathfrak{B}} + \frac{1}{I_{xx}} \frac{\sqrt{3}l}{2}(f_2 - f_3) \\
\dot{q}^{\mathfrak{B}} &= \frac{(I_{zz} - Ixx)}{I_{yy}} p^{\mathfrak{B}} r^{\mathfrak{B}} + \frac{1}{I_{yy}}(\frac{l}{2}(f_2 + f_3) - l f_1 \cos\alpha + \tau_1 \sin\alpha) \\
\dot{r}^{\mathfrak{B}} &= \frac{(I_{xx} - Iyy)}{I_{zz}} p^{\mathfrak{B}} q^{\mathfrak{B}} + \frac{1}{I_{zz}}(-l f_1 \sin\alpha - \tau_1 \cos\alpha - \tau_2 - \tau_3),
\end{aligned} \tag{3.19}$$

describing how the angular rates depends on the thrust and drag from the rotors. By studying (3.19) it can be seen that the tricopter is a cross-coupled system. The time derivative of the states depend on the value of the other states. For example, the time derivative of state $p$ depends on the value of $q$ and $r$. Also each input signal changes several of the states. For example, the induced thrust force from rotor no.2, $f_2$, changes both the state $p$ and $q$.

Since the orientation of the tricopter is described with Euler angels, the rotational angle velocities of the tricopter, with respect to the global coordinate system (X,Y,Z)$^{\mathfrak{G}}$, these have to be transformed to the global coordinate system. The transformation is given by

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix}^{\mathfrak{G}} = \begin{pmatrix} 1 & \sin(\phi^{\mathfrak{G}})\tan(\theta^{\mathfrak{G}}) & \cos(\phi^{\mathfrak{G}})\tan(\theta^{\mathfrak{G}}) \\ 0 & \cos(\phi^{\mathfrak{G}}) & -\sin(\phi^{\mathfrak{G}}) \\ 0 & \frac{\sin(\phi^{\mathfrak{G}})}{\cos(\theta^{\mathfrak{G}})} & \frac{\cos(\phi^{\mathfrak{G}})}{\cos(\theta^{\mathfrak{G}})} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}^{\mathfrak{B}} \tag{3.20}$$

which is derived by taking the time derivative of the rotation matrix Q (3.3) and then inverting, Gustafsson [2012]. The derivation of this matrix can be found in appendix A.

### 3.2.4 Nonlinear model

The system can be described with the differential equations (3.16 - 3.20). Translational velocity of the tricopter is described with (3.16) and the translational po-

sition is described with (3.17). The rotational rates of the tricopter are described with (3.19) and the rotational angles are described with (3.20). The resulting nonlinear model is

$$\dot{x}^{\mathfrak{G}} = u^{\mathfrak{G}} \tag{3.21a}$$

$$\dot{y}^{\mathfrak{G}} = v^{\mathfrak{G}} \tag{3.21b}$$

$$\dot{z}^{\mathfrak{G}} = w^{\mathfrak{G}} \tag{3.21c}$$

$$\dot{u}^{\mathfrak{G}} = \frac{1}{m}\left(F^{\mathfrak{B}}_{X,ext}\mathrm{C}_\theta\mathrm{C}_\psi + F^{\mathfrak{B}}_{Y,ext}\mathrm{C}_\theta\mathrm{S}_\psi - F^{\mathfrak{B}}_{z,ext}\mathrm{S}_\theta\right) \tag{3.21d}$$

$$\dot{v}^{\mathfrak{G}} = \frac{1}{m}\left(F^{\mathfrak{B}}_{X,ext}(\mathrm{S}_\phi\mathrm{S}_\theta\mathrm{C}_\psi - \mathrm{C}_\phi\mathrm{S}_\psi) + F^{\mathfrak{B}}_{Y,ext}(\mathrm{S}_\phi\mathrm{S}_\theta\mathrm{S}_\psi + \mathrm{C}_\theta\mathrm{C}_\psi) + F^{\mathfrak{B}}_{Z,ext}\mathrm{S}_\phi\mathrm{C}_\theta\right) \tag{3.21e}$$

$$\dot{w}^{\mathfrak{G}} = \frac{1}{m}\left(F^{\mathfrak{B}}_{X,ext}(\mathrm{C}_\phi\mathrm{S}_\theta\mathrm{C}_\psi + \mathrm{S}_\phi\mathrm{S}_\psi) + F^{\mathfrak{B}}_{Y,ext}(\mathrm{C}_\phi\mathrm{S}_\theta\mathrm{S}_\psi - \mathrm{S}_\phi\mathrm{C}_\psi) + F^{\mathfrak{B}}_{Z,ext}\mathrm{C}_\phi\mathrm{C}_\theta\right) - g \tag{3.21f}$$

$$\dot{\phi}^{\mathfrak{G}} = p^{\mathfrak{B}} + \sin(\phi^{\mathfrak{G}})\tan(\theta^{\mathfrak{G}})q^{\mathfrak{B}} + \cos(\phi^{\mathfrak{G}})\tan(\theta^{\mathfrak{G}})r^{\mathfrak{B}} \tag{3.21g}$$

$$\dot{\theta}^{\mathfrak{G}} = \cos(\phi^{\mathfrak{G}})q^{\mathfrak{B}} - \sin(\phi^{\mathfrak{G}})r^{\mathfrak{B}} \tag{3.21h}$$

$$\dot{\psi}^{\mathfrak{G}} = \frac{\sin(\phi^{\mathfrak{G}})}{\cos(\theta^{\mathfrak{G}})}q^{\mathfrak{B}} + \frac{\cos(\phi^{\mathfrak{G}})}{\cos(\theta^{\mathfrak{G}})}r^{\mathfrak{B}} \tag{3.21i}$$

$$\dot{p}^{\mathfrak{B}} = \frac{(I_{yy} - Izz)}{I_{xx}}r^{\mathfrak{B}}q^{\mathfrak{B}} + \frac{1}{I_{xx}}\frac{\sqrt{3}l}{2}(f_2 - f_3) \tag{3.21j}$$

$$\dot{q}^{\mathfrak{B}} = \frac{(I_{zz} - Ixx)}{I_{yy}}p^{\mathfrak{B}}r^{\mathfrak{B}} + \frac{1}{I_{yy}}(\frac{l}{2}(f_2 + f_3) - lf_1\cos\alpha + \tau_1\sin\alpha) \tag{3.21k}$$

$$\dot{r}^{\mathfrak{B}} = \frac{(I_{xx} - Iyy)}{I_{zz}}p^{\mathfrak{B}}q^{\mathfrak{B}} + \frac{1}{I_{zz}}(-lf_1\sin\alpha - \tau_1\cos\alpha - \tau_2 - \tau_3). \tag{3.21l}$$

The output signals are defined in Section 3.1.2. By inverting (3.2a) for longitude, inverting (3.2b) for latitude and using the expression (3.18) for absolute velocity,

will result in following measurement function, $h(\mathbf{X})$

$$\mathbf{Y} = \begin{pmatrix} \lambda_{lat}^{\mathfrak{E}} \\ \lambda_{lng}^{\mathfrak{E}} \\ z^{\mathfrak{G}} \\ \overline{V}^{\mathfrak{G}} \\ \psi^{\mathfrak{G}} \\ p^{\mathfrak{B}} \\ q^{\mathfrak{B}} \\ r^{\mathfrak{B}} \end{pmatrix} = h \begin{pmatrix} x^{\mathfrak{G}} \\ y^{\mathfrak{G}} \\ z^{\mathfrak{G}} \\ u^{\mathfrak{G}} \\ v^{\mathfrak{G}} \\ w^{\mathfrak{G}} \\ \phi^{\mathfrak{G}} \\ \theta^{\mathfrak{G}} \\ \psi^{\mathfrak{G}} \\ p^{\mathfrak{B}} \\ q^{\mathfrak{B}} \\ r^{\mathfrak{B}} \end{pmatrix} = \begin{pmatrix} \frac{180 x^{\mathfrak{G}}}{\pi R_{\text{Earth}}} \\ \frac{180 y^{\mathfrak{G}}}{\pi R_{\text{Earth}} \cos(\frac{x^{\mathfrak{G}}}{R_{\text{Earth}}})} \\ z^{\mathfrak{G}} \\ \sqrt{(u^{\mathfrak{G}})^2 + (v^{\mathfrak{G}})^2 + (w^{\mathfrak{G}})^2} \\ \psi^{\mathfrak{G}} \\ p^{\mathfrak{B}} \\ q^{\mathfrak{B}} \\ r^{\mathfrak{B}} \end{pmatrix}. \tag{3.22}$$

Since this tricopter will be not used for acrobatics in this thesis, the model can be simplified to cover only simple flight cases such as hovering and small rotation around this hovering case. Therefore the mode will be linearized so that the controller and filter will be more computationally efficient.

## 3.3 Linearization

In this control problem, the desired behavior of the tricopter is when it is maintaining a pre-defined position and orientation. This means that the tricopter can be linearized around this equilibrium point $\mathbf{X}_0, \mathbf{U}_0$. With equilibrium point means that the first derivative of the states are equal to zero,

$$f(\mathbf{X}_0, \mathbf{U}_0) \equiv 0.$$

In [Glad and Ljung, 2003] the linearization of the dynamics will result in A, B and C matrices, where the elements in the A, B and C matrices are given by $a_{i,j}, b_{i,j}, c_{i,j}$, respectively

$$a_{i,j} = \frac{\partial f_i(\mathbf{X}_0, \mathbf{U}_0)}{\partial \mathbf{X}_j}, \quad b_{i,j} = \frac{\partial f_i(\mathbf{X}_0, \mathbf{U}_0)}{\partial \mathbf{U}_j}, \quad c_{i,j} = \frac{\partial h_i(\mathbf{X}_0)}{\partial \mathbf{X_j}}, \tag{3.23}$$

where $i$ denotes the row and $j$ denotes the column.

The result from the modeling of the tricopter was nonlinear functions which depend on the state-space variable $\mathbf{X}$ and the input signal $\mathbf{U}$. Linearizing these nonlinear equations, using (3.23), will result in the following description of the

dynamics

$$\begin{aligned}\dot{\mathbf{Z}} &= A\mathbf{Z} + B\mathbf{V}\\ \mathbf{X}_0 &= [x_0^{\mathfrak{G}}, y_0^{\mathfrak{G}}, z_0^{\mathfrak{G}}, u_0^{\mathfrak{G}}, v_0^{\mathfrak{G}}, w_0^{\mathfrak{G}}, \phi_0^{\mathfrak{G}}, \theta_0^{\mathfrak{G}}, \psi_0^{\mathfrak{G}}, p_0^{\mathfrak{B}}, q_0^{\mathfrak{B}}, r_0^{\mathfrak{B}}]^T\\ \mathbf{U}_0 &= (\omega_{1,0},\ \omega_{2,0},\ \omega_{3,0},\ \alpha_0)^T\end{aligned}$$

where

$$\begin{aligned}\mathbf{Z} &= \mathbf{X} - \mathbf{X}_0\\ \mathbf{V} &= \mathbf{U} - \mathbf{U}_0\\ \mathbf{X}_0 &= [x_0, y_0, z_0, 0, 0, 0, \pi, 0, \phi_0, 0, 0, 0]^T,\end{aligned} \tag{3.24}$$

which is corresponding to the states when the tricopter is hovering in the air, i.e. the position and orientation are constant, but the rotational and translational velocities are zero. Since the coordinate system $\mathfrak{G}$ has the $Z$-axis upwards and the coordinate system $\mathfrak{B}$ has the $Z$-axis downwards, the rotation between these systems is $[\pi, 0, \phi_0]$. This will result in following matrices

$$A = \begin{pmatrix}
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{4,8} & a_{4,9} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & a_{5,7} & a_{5,8} & a_{5,9} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & a_{6,7} & a_{6,8} & a_{6,9} & 0 & 0 & 0\\
0 & 0 & 0 & 0 & 0 & 0 & a_{7,7} & a_{7,8} & 0 & a_{7,10} & a_{7,11} & 0\\
0 & 0 & 0 & 0 & 0 & 0 & a_{8,7} & 0 & 0 & 0 & a_{8,11} & a_{8,12}\\
0 & 0 & 0 & 0 & 0 & 0 & a_{9,7} & a_{9,8} & 0 & 0 & a_{9,11} & a_{9,12}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{10,11} & a_{10,12}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{11,10} & 0 & a_{11,12}\\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{12,10} & a_{12,11} & 0
\end{pmatrix}, \tag{3.25a}$$

$$B = \begin{pmatrix}
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & b_{4,2} & b_{4,3} & b_{4,4}\\
b_{5,1} & b_{5,2} & b_{5,3} & b_{5,4}\\
b_{6,1} & b_{6,2} & b_{6,3} & b_{6,4}\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & 0 & 0 & 0\\
0 & b_{10,2} & b_{10,3} & 0\\
b_{11,1} & b_{11,2} & b_{11,3} & b_{11,4}\\
b_{12,1} & b_{12,2} & b_{12,3} & b_{12,4}
\end{pmatrix}, \tag{3.25b}$$

$$C = \begin{pmatrix} c_{1,1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c_{2,1} & c_{2,2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{4,4} & c_{4,5} & c_{4,6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{3.25c}$$

The equations for the non-zero matrix elements $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$ can be found in Appendix B.

## 3.4 Black-box model

To begin with, the input and output signals of the system have to be chosen. The selection of the output signals will be the angular rates $(p, q, r)^{\mathfrak{B}}$, which are the rotational velocities around the tricopter's (X, Y, Z)$^{\mathfrak{B}}$-axes. These are also the outputs that are of interest to be stabilized. The input signals to the system are the ones defined in section 3.1.2.

Next step is to choose a model structure which includes the selected in- and output signals. The tricopter is a highly nonlinear and cross-coupled system. Since the tricopter will be stabilized around a point where the rotational rates of the tricopter are zero, the system can be approximated as a linear one with no information about the structure of the dynamics. That will give the structure

$$\begin{aligned} \dot{\mathbf{X}}(t) &= A\mathbf{X}(t) + B\mathbf{U}(t) \\ \mathbf{Y}(t) &= C\mathbf{X}(t) \end{aligned} \tag{3.26}$$

where

$$\begin{aligned} \mathbf{Y}(t) &= [p^{\mathfrak{B}}, q^{\mathfrak{B}}, r^{\mathfrak{B}}]^T \\ \mathbf{U}(t) &= [\omega_1, \omega_2, \omega_3, \alpha]^T \end{aligned}$$

This model is a linearization of (3.19) around the equlibrium point $\mathbf{X}_0 = [p_0, q_0, r_0]^T = [0, 0, 0]^T$ and $\mathbf{U}_0 = [\omega_{1,0}, \omega_{2,0}, \omega_{3,0}, \alpha_0]^T$. This corresponds to that the tricopter has no rotational rates, and the input signals are the rotational velocities of the rotors, to achieve this equilibrium point. The *black-box* model does not consider the position, the velocity or the orientation of the tricopter into account. The measured signals in this case will be the values from gyroscopes which measure the angular rates around the $(X, Y, Z)^{\mathfrak{B}}$-axes. This gives that the output of the *black-box* model will be the rotational rates of the tricopter, $\mathbf{Y} = [p, q, r]^T$.

## 3.5 Black-box virtual model

Another way to represent the *black-box* state space model is to make a linear transformation of the input signals to the *black-box* model. By using the physical description of the rotational rates (3.14b), linearizing these around $\mathbf{U}_0$, approximating $\alpha_0 \approx 0$ and that the torque around $Z^{\mathcal{B}}$-axis only depends on angle $\alpha$, the following equations are obtained:

$$\begin{pmatrix} M_x^{\mathcal{B}} \\ M_y^{\mathcal{B}} \\ M_z^{\mathcal{B}} \end{pmatrix} = \begin{pmatrix} 0 & \sqrt{3}lk_t\omega_{2,0} & -\sqrt{3}lk_t\omega_{3,0} & 0 \\ -2lk_t\omega_{1,0} & lk_t\omega_{2,0} & lk_t\omega_{3,0} & 0 \\ 0 & 0 & 0 & \alpha_0 \end{pmatrix} \begin{pmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \\ \alpha \end{pmatrix}. \tag{3.27}$$

Since the rotational rates $(p^{\mathcal{B}}, q^{\mathcal{B}}, r^{\mathcal{B}})^T \propto (M_x^{\mathcal{B}}, M_y^{\mathcal{B}}, M_z^{\mathcal{B}})^T$, the virtual input signals can be seen as the difference in thrust between the rotors, and effects from the motor tilt angle. This gives that the transformed, virtual, input signals can be expressed as

$$\mathbf{U}_v = \begin{pmatrix} \mathbf{U}_r \\ \mathbf{U}_p \\ \mathbf{U}_y \end{pmatrix} = \begin{pmatrix} C_r(\omega_2 - \omega_3) \\ C_p(\frac{\omega_2+\omega_3}{2} - \omega_1) \\ C_y\alpha \end{pmatrix}, \tag{3.28}$$

where $C_i$ are constants. These are the control signals that the PI-controller in ArduCopter is using to control the rotational rates of the tricopter. The calculation of the PWM signals to each motor and servo, that the PI controller uses are

$$\omega_1 = \omega_{th} - \mathbf{U}_p \tag{3.29a}$$

$$\omega_2 = \omega_{th} + \frac{\sqrt{3}}{2}\mathbf{U}_r + \frac{1}{2}\mathbf{U}_p \tag{3.29b}$$

$$\omega_3 = \omega_{th} - \frac{\sqrt{3}}{2}\mathbf{U}_r + \frac{1}{2}\mathbf{U}_p \tag{3.29c}$$

$$\alpha = \mathbf{U}_y, \tag{3.29d}$$

where $\omega_{th}$ is the common rotational rate of the rotors and is controlled by the throttle stick on the Radio Controller. By solving $(\mathbf{U}_r, \mathbf{U}_p, \mathbf{U}_y)$ from (3.29), the constants in (3.28), $(C_p, C_r, C_y) = (\frac{1}{\sqrt{3}}, \frac{2}{3}, 1)$, can be obtained. The state space representation of the *black-box virtual* model is

$$\begin{aligned} \dot{\mathbf{X}}(t) &= A\mathbf{X}(t) + B\mathbf{U}_v(t) \\ \mathbf{Y}(t) &= C\mathbf{X}(t), \end{aligned} \tag{3.30}$$

where

$$\begin{aligned} \mathbf{Y}(t) &= [p^{\mathcal{B}}, q^{\mathcal{B}}, r^{\mathcal{B}}]^T \\ \mathbf{U}_v(t) &= [\mathbf{U}_r, \mathbf{U}_p, \mathbf{U}_y]^T. \end{aligned}$$

These virtual input signals are then used in (3.29) to compute the PWM signals to the rotors and servo.

## 3.6 Limitations

The tricopter is a physical system where there are limitations by the laws of physics. The only limitations that have been considered in this thesis, are the rotational rates of each motor and the tilt angle of the servo. As mentioned in Chapter 2, the PWM signal to the ESC is limited to an interval $(1000, 2000)$ $[\mu s]$, and the PWM signal to the servo is limited to $(1200, 1800)$ $[\mu s]$. Therefore the signals to the electrical motors and the servo are limited, which gives a bounded input signal.

## 3.7 Discussion

The two model structures that were presented in this chapter differ from each other. The *physical* model was derived from the laws of physics which gives that the dynamical equations are known. This differ from the *black-box* and *black-box virtual* model, which are Black-box models, where only the input and output signals are known, and the dynamics are unknown. In that way the dynamics of the Black-box model can be seen as a virtual description of the reality and not as a physical.

The *physical* model is not used further on in this thesis, due to the complexity of the model. The microcontroller did not have enough computational resources to handle this model. Therefore, only the *black-box* and *black-box virtual* model will be used from now on to compute a filter and a controller. But as mentioned, the *physical* model is a good reference that the system is a highly non-linear and cross-coupled system.

The input signals to the system are the rotational rates of each motor and the tilt angle of the rear motor. In reality the input signals of the system are the signals from the ESC to the motors and servo. As mentioned in Section 3.1.3 the system function of each of the motors has one pole and the servo has two poles. This means that the system in reality will have at least five more poles than the system model of the tricopter. It can be assumed that the pole of each motor is located near the origin. This gives a short rise time for a step response, and therefore the relation between the input signal $u$ and the rotational rate $\omega$ is assumed to be static, i.e. $u(t) \approx \omega(t)$. These five extra poles that the electrical motors and the servo give are neglected. If the model order is too low for estimation, it can be explained by the poles of the electrical motors and the servo that are not included in the system model.

# 4

# System identification

This chapter is about how to identify and validate a system model. There are different kinds of estimation methods, but in this thesis only the PEM, Prediction Error Method, Glad and Ljung [2004], has been used. To identify and validate the model, data sets with input and output signals are needed. The input and output signals are sampled measurements with sample time $T_s$, and since the model is implemented on hardware, the identified model will in this case be a discrete time model of the system

When the model is identified, it has to be validated to confirm that the model can describe the dynamics of the system. To ensure that the model is describing all the dynamics of the system and to prevent overfit, the validation is done with a different data set than the identification.

The model structures defined in Chapter 3 have to be discretized since the used data sets are sampled.

## 4.1 Discretization

The linearized system that is used in this thesis is a linear continuous time system

$$\dot{\mathbf{X}}(t) = A\mathbf{X}(t) + B\mathbf{U}(t)$$
$$\mathbf{Y}(t) = C\mathbf{X}(t) + D\mathbf{U}(t).$$

Since it is a microcontroller that is setting the input signal to the system, this signal can be seen as a picewise constant input signal to the system, Glad and Ljung [2004]

$$\mathbf{U}(t) = \mathbf{U}(t_k) = \mathbf{U}_k, \quad \text{for } t_k \leq t \leq t_{k+1}.$$

This will result in a linear time discrete system

$$\begin{aligned}\mathbf{X}_{k+1} &= \tilde{A}_k\mathbf{X}_k + \tilde{B}_k\mathbf{U}_k\\ \mathbf{Y}_k &= C\mathbf{X}_k + D\mathbf{U}_k,\end{aligned}$$

where

$$\begin{aligned}\tilde{A}_k &= e^{A(t_{k+1}-t_k)}\\ \tilde{B}_k &= \int_0^{t_{k+1}-t_k} e^{A\tau}d\tau B,\end{aligned} \tag{4.1}$$

and $t_{k+1} - t_k = T_s$ is the sampling time. In this way the system can be discretized and this discrete time system can be identified.

## 4.2 Estimation

There are methods to identify both linear models and nonlinear models, but in this thesis the identification of a linear model has been of interest. To estimate a linear model a model structure needs to be defined. In Section 3 three different models were defined: one nonlinear model that was linearized, *black-box* model and *black-box virtual* model. All these three models will be described with a discrete time invariant linear dynamic model with following form

$$\begin{aligned}\mathbf{X}_k &= A(\theta)\mathbf{X}_k + B(\theta)\mathbf{U}_k\\ \mathbf{Y}_k &= C(\theta)\mathbf{X}_k,\end{aligned} \tag{4.2}$$

where $\mathbf{X}$ is the state space vector, $\mathbf{U}$ the input signal vector and $\mathbf{Y}$ is the output signal vector. The matrices depend on the unknown parameter vector $\theta$. The three models differ from each other by the elements in $A, B$ and $C$. The size of the matrices are determined by the number of states, input signals and output signals. The matrix $A$ has the size $n_x \times n_x$, where $n_x$ is the number of states. The matrix $B$ has the size $n_x \times n_u$, where $n_u$ is the number of input signals. The matrix $C$ has the size $n_y \times n_x$ where $n_y$ is the number of output signals.

Both model structures that were presented in Chapter 3 can be identified using PEM. Before defining the estimation method, a disturbance has to be introduced in the model. This term will include the signals that can not be controlled or measured explicitly such as wind and turbulences, but also uncertainty in the measurement. The new dynamical model is described with a disturbance model

$$\begin{aligned}\mathbf{X}_{k+1} &= A(\theta)\mathbf{X}_k + B(\theta)\mathbf{U}_k + w_k\\ \mathbf{Y}_k &= C(\theta)\mathbf{X}_k + e_k,\end{aligned} \tag{4.3}$$

where $w_k$ is the process noise term and $e_k$ is the measurement noise term. These two noise terms are uncorrelated and the process noise term has the covariance $Q_k$ and the measurement noise term has the covariance $R_k$. The model can be written on an innovation form, where the the process noise is seen as the innova-

tion, the new contribution at each time step

$$\begin{aligned} \hat{\mathbf{X}}_{k+1} &= A(\theta)\hat{\mathbf{X}}_k + B(\theta)\mathbf{U}_k + K(\theta)e_k \\ \mathbf{Y}_k &= C(\theta)\hat{\mathbf{X}}_k + e_k, \end{aligned} \tag{4.4}$$

and the states are denoted with $\wedge$ since these differs from the ones in (4.3).

The PEM uses the prediction error $\varepsilon_k$, which is the difference between measured output and predicted output of the linear model, to calculate the model parameters. The measured output at time k is denoted as $\mathbf{Y}_k$ and the predicted output using the model is $\hat{\mathbf{Y}}_{k|\theta}$. The definition of the prediction error is

$$\varepsilon_{k,\theta} = \mathbf{Y}_k - \hat{\mathbf{Y}}_k \tag{4.5}$$

where $\hat{\mathbf{Y}}_k = C(\theta)\hat{\mathbf{X}}_k$. The PEM uses the prediction error $\varepsilon_k$ to solve the parameters $\theta$

$$\varepsilon_{k,\theta} = H(q,\theta)^{-1}\Big(\mathbf{Y}_k - G(q,\theta)\mathbf{U}_k\Big) \tag{4.6}$$

where $G(q,\theta)$ is the transfer function from $\mathbf{U} \rightarrow \mathbf{Y}$ and $H(q,\theta)$ is the transfer function from $e \rightarrow \mathbf{Y}$. The transfer function $G$ can be described by using the model

$$G(q,\theta) = C(\theta)(qI - A(\theta))^{-1}B(\theta), \tag{4.7}$$

and the transfer function $H$ as

$$H(q,\theta) = \Big(I + C(\theta)\Big(qI - A(\theta)\Big)^{-1}K(\theta)\Big). \tag{4.8}$$

It gives that the output can be rewritten as

$$\mathbf{Y}_k = G(q,\theta)\mathbf{U}_k + H(q,\theta)e_k \tag{4.9}$$

and the prediction error $\varepsilon_k$ can be written as $\varepsilon_{k,\theta} = e_{k,\theta} = H(q,\theta)^{-1}(\mathbf{Y}_k - G(q,\theta)\mathbf{U}_k)$. The unknown parameters are obtained by minimizing the variance of the prediction error with respect to $\theta$

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{N}\sum_{k=1}^{N} \varepsilon_k \varepsilon_k^T. \tag{4.10}$$

There are two approaches to solve for the parameters. Either by focusing on minimizing the simulation error or focusing on minimizing the prediction. By focusing on simulation error, the parameters in $G$ are minimized so that the output error, $\mathbf{Y}_k - G(q,\theta)\mathbf{U}_k$, is minimized and then the prediction error, $\varepsilon_k$, is minimized. If focusing on prediction is used, all the parameters are minimized by minimizing the prediction error (4.6).

## 4.3 Validation

After identification of a model it has to be validated to decide if it is able to describe the system dynamics. The methods that were used in this thesis to validate

the quality of the identified model was cross validation of the model output, $k$-step prediction and residual analysis. By using all three validation methods it gives a good view if the quality of the model is good enough to describe the dynamics of the system.

### Cross validation

The cross validation of the model output is a method where the output signal is simulated using the input signals from the data set. This simulated output signal is then compared to the measured output. To see how the simulated output fits the measured output, the ratio between the difference between measured output and simulated output and the difference between measured output and the mean of measured output is used. This ratio is then transformed to percents, i.e.

$$\text{Fit} = 100 \times \left(1 - \frac{|\mathbf{Y}-\hat{\mathbf{Y}}|}{|\mathbf{Y}-\overline{\mathbf{Y}}|},\right) \tag{4.11}$$

where $\mathbf{Y}$ is the measured output, $\hat{\mathbf{Y}}$ is the simulated output and $\overline{\mathbf{Y}}$ is the mean of the measured output signal. If the simulated output is equal to the measured output, i.e. $\hat{\mathbf{Y}} = \mathbf{Y}$, it results in 100% fit and the model is describing the reality correctly. This fit will not in reality be exact 100% because there are disturbances acting on the system which will result in a model deviation. If the fit is equal to zero it means that the simulated output could be assumed to be constant, i.e. $\hat{\mathbf{Y}} = \overline{\mathbf{Y}}$, instead of be predicted with a dynamical model.

### $k$-step prediction

The $k$-step prediction uses the system model to predict the state and output signals $k$-steps ahead given the information at time $k_0$, i.e.

$$\begin{aligned} \hat{\mathbf{X}}_{k_0+k+1|k_0} &= A\hat{\mathbf{X}}_{k_0+k|k_0} + B\mathbf{U}_{k_0+k}, \quad k = 0, 1, \dots \\ \hat{\mathbf{Y}}_{k_0+k|k_0} &= C\hat{\mathbf{X}}_{k_0+k|k_0}. \end{aligned} \tag{4.12}$$

and compare $\hat{\mathbf{Y}}_{k+k_0|k_0}$ with $\mathbf{Y}_{k+k_0}$ in the same way as with the cross validation. This validation gives a good hint about how the model can predict the output signal $k$-step ahead, since the control method that is used in this thesis is a predictive controller. Therefore the model has to be able to predict the signals accurately, so that the correct control signal can be applied to the system.

### Residual analysis

The last validation method is residual analysis. This method is studying the autocorrelation of the prediction error, and how the prediction error is correlated to the input signal. The residual analysis is to compute the autocorrelation

$$\hat{R}_\varepsilon = \frac{1}{N}\sum_{k=1}^{N} \varepsilon_{k+\tau}\varepsilon_k, \quad |\tau| \leq N, \tag{4.13}$$

and the cross correlation

$$\hat{R}_{\varepsilon u} = \frac{1}{N} \sum_{k=1}^{N} \varepsilon_{k+\tau} u_k, \quad |\tau| \leq N, \tag{4.14}$$

where $\varepsilon$ is the prediction error (4.6) and $\tau$ is a time deviation from the measurement at time $k$. If the autocorrelation is zero, or within 99% confidence interval, for $\tau \neq 0$, the output signal depends on earlier time of the output signal. If this is the case, the dynamics do not include all the information since the noise in the model are assumed to be white noise.

If the cross correlation $\hat{R}_{\varepsilon u}$ is zero for $\tau = 0$ and zero, or within 99% confidence interval, for all other values of $\tau$, the prediction error only depends on the input signal at time $k$. Otherwise the prediction error will depend on input signal from an earlier time. If this is the case, there is dynamics that the model can not handle and the model order has to be increased. A correlation between residuals and input for negative lags, is not necessarily an indication of an inaccurate model. There could be a feedback in the system, where current residual affect future input signals.

# 5

# Filtering and state estimation

The tricopter will be affected by different kind of forces during a flight and these can be seen as input signals to the system. Some of them are controllable such as the input signals defined in Section 3.1.2, but there are also signals that are uncontrollable. For example, wind and turbulence affect the tricopter, but these signals cannot be controlled or measured explicitly. By expressing these uncontrollable signals as a noise term, the term can be included in the model, which was done in Section 4.2. The model (4.4) is described in innovation form, i.e. it is the same noise signal that affects the states and the output signal.

There are two different approaches used in this thesis to get rid of the noise from the measurements: a frequency selective filter and a model based filter.

If the noise and the dominating dynamics are separated in frequency, the measured signals can be filtered out with a frequency selective filter, Söderquist [2007] and Gustafsson et al. [2010]. If the noise is in the same frequency as the system dynamics, a model based filter has to be used. If the noise is assumed to be Gaussian, the Kalman Filter approach can be used, Gustafsson [2012]. Both filter approaches used in this thesis are discrete time filters that has been implemented in either the software on tricopter or in MATLAB.

## 5.1 Low pass filter

The frequency selective filter that was used in this theis was a time discrete LP-filter, Low Pass. A linear frequency selective filter is a mapping from an input

sequence $u$ to an output sequence $y$

$$y_k = h * u_k = \sum_{k=-\infty}^{\infty} h_{k-\tau} u_k \,. \tag{5.1}$$

The filter is causal, i.e. the output signal is independent of future values of the input signal, if and only if

$$h_k = 0, \quad k < 0. \tag{5.2}$$

The filter is BIBO, Bounded Input Bounded Output, stable if a bounded input signal results in a bounded output signal. For a linear system this can be shown that the requirement to fulfill BIBO is

$$\sum_{k=-\infty}^{\infty} |h_k| < \infty. \tag{5.3}$$

A basic type of linear and causal time discrete filter can be described with difference equations in the time domain

$$\mathbf{Y}_k + a_1 \mathbf{Y}_{k-1} + \ldots + a_n \mathbf{Y}_{k-n} = b_0 \mathbf{U}_k + b_1 \mathbf{U}_{k-1} + \ldots + b_m \mathbf{U}_{k-m} \,, \tag{5.4}$$

where $\mathbf{Y}_k$ is the output signal, $\mathbf{U}_k$ is the input signal at time $k$. The order of the time shifts in output signals is given by $n$ and shifts of the output signal by $m$. Both $m$ and $n$ have to be greater or equal to zero for the filter to be causal.

A linear discrete time filter can be described by a Z-transform

$$\mathbf{H}(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_m z^{-m}}{1 + a_1 z^{-1} + \ldots + a_n z^{-n}}, \tag{5.5}$$

and the relation between the input and output signal is then given by

$$\mathbf{Y}(z) = \mathbf{H}(z)\mathbf{U}(z). \tag{5.6}$$

The filter can be obtained from (5.4) by using the shift operator $q$, ($y_{k-1} = q^{-1} y_k$), and replacing the $q$ with $z$, since z-transform is a shift operator in the discrete time domain. The filter is stable if and only if the poles of $\mathbf{H}(z)$ are strictly inside the unit circle.

### 5.1.1 Cutoff frequency

The cutoff frequency for a discrete time LP-filter is determined from a continuous time LP-reference-filter

$$H(s) = \frac{b_0 + b_1 s + \ldots + b_m s^m}{1 + a_1 s + \ldots + a_n s^n}, \tag{5.7}$$

where $s$ is the Laplace operator. The cutoff frequency for a continuous time filter is given by $\omega_p$ so that the gain of the filter has dropped with 3dB from the

maximum at this frequency

$$|H(i\omega_p)| = \frac{1}{\sqrt{2}} \max_{\omega} |H(i\omega)| - \tag{5.8}$$

The discrete time filter is determined by first choosing the normed discrete time cutoff frequency $\Omega_p$ [Hz] for the discrete time filter. Then the cutoff frequency $\omega_p$ [Hz] for the time continuous reference filter is selected. This reference filter is then discretized by bi-linear transformation

$$s = \gamma \frac{z-1}{z+1},$$

where $\gamma$ is a real constant that is selected so that

$$\omega_p = \gamma \tan \frac{\Omega_p}{2}. \tag{5.9}$$

A time-continuous filter $H(s)$ has now been determined that fulfills (5.8) and is the transformed with (5.9) to the discrete $H(z) = H(\gamma \frac{z-1}{z+1})$.

## 5.2 Linear Kalman filter

The second filtering approach used in this thesis is model based filter. In this thesis the Kalman filter has been used, which is commonly used filtering method in which the noise is assumed to be Gaussian. The Kalman filter is used to estimate the states in a linear state space model

$$\mathbf{X}_{k+1} = A_k \mathbf{X}_k + B_k \mathbf{U}_k + w_k, \qquad Cov(w_k) = Q_k \tag{5.10a}$$

$$\mathbf{Y}_k = C_k \mathbf{X}_k + e_k, \qquad Cov(e_k) = R_k \tag{5.10b}$$

$$E(\mathbf{X}_0) = \mathbf{X}_{1|0} \tag{5.10c}$$

$$Cov(\mathbf{X}_0) = \mathbf{P}_{1|0} \quad . \tag{5.10d}$$

where $w_k$ is the unmodeled noise and $e_k$ is the measurement noise. This model is not in innovation form as (4.4) because the two noise terms are uncorrelated. The model can be written on innovation form with the noise term $K_k e_k$ affecting the states, but $K_k$ is now the Kalman gain. This gain can be written as

$$K_k = P_{k|k-1} C_k^T \left( C_k P_{k|k-1} C_k^T + R_k \right)^{-1}, \tag{5.11}$$

which is used to present the model in innovation form, i.e. the noise source takes the role of the new contribution at each time in the process. The estimation is done in two steps, measurement update and time update. The measurement update is done by

$$\hat{\mathbf{X}}_{k|k} = \hat{\mathbf{X}}_{k|k-1} + K_k \left( \mathbf{Y}_k - C \hat{\mathbf{X}}_{k|k-1} \right) \tag{5.12a}$$

$$P_{k|k} = P_{k|k-1} - K_k C_k P_{k|k-1}. \tag{5.12b}$$

By using the information from the measured output signals, the confidence of the states can be improved, instead of mere simulation, which gives a much larger

uncertainty of the states.

The time update is done by

$$\hat{\mathbf{X}}_{k+1|k} = A_k \hat{\mathbf{X}}_{k|k-1} + B_k \mathbf{U}_k \tag{5.12c}$$

$$P_{k+1|k} = A_k P_{k|k-1} A_k^T + Q_k. \tag{5.12d}$$

With the time update the state in the next time step is estimated by using the dynamical model and the input signal to the system. Since the measurement update and time update consist of several matrix multiplication, it will be too demanding for the microcontroller to perform the computations. Since the model is time-invariant, the stationary Kalman filter has been used.

### 5.2.1 Stationary Kalman filter

A stationary Kalman filter, Gustafsson et al. [2010], can be used when a time-invariant state space model is used

$$\mathbf{X}_{k+1} = A\mathbf{X}_k + B\mathbf{U}_k + w_k, \qquad Cov(w_k) = Q \tag{5.13}$$

$$\mathbf{Y}_k = C\mathbf{X}_k + e_k, \qquad Cov(e_k) = R. \tag{5.14}$$

If the Kalman filter is applied for the first time at time $k_0$, the Kalman gain $K_k$ will approach to a steady state gain $\overline{K}$ when $k \to \infty$ or when $k_0 \to -\infty$. In the same way the covariance of the estimated state $P_k = Cov(\hat{\mathbf{X}}_k) \to \overline{P}$. This gives that the measurement and time update part of the Kalman filter can be written as

$$\hat{\mathbf{X}}_{k|k} = \hat{\mathbf{X}}_{k|k-1} + \overline{K}\left(\mathbf{Y}_k - C\hat{\mathbf{X}}_{k|k-1}\right) \tag{5.15a}$$

for the measurement update and for the time update

$$\hat{\mathbf{X}}_{k+1|k} = A_k \hat{\mathbf{X}}_{k|k-1} + B_k \mathbf{U}_k \tag{5.15b}$$

where the stationary covariance and Kalman gain is given by

$$\overline{P} = A\overline{P}A^T - A\overline{P}C^T\left(C\overline{P}C^T + R\right)^{-1} C\overline{P}A^T + Q \tag{5.15c}$$

$$\overline{K} = \overline{P}C^T\left(C\overline{P}C^T + R\right)^{-1}, \tag{5.15d}$$

where $\overline{P}$ is the solution to the stationary Riccati equation (5.15c). The stationary Kalman filter is a model based LP-filter.

# 6
# Controller

This aim of this thesis is to create a reliable controller for a tricopter, which was described in Chapter 2. The system model was defined in Chapter 3 and then the identification of the model was described in Chapter 4. Now it is time to define the controller for the system. In this thesis MPC, Model Predictive Control has been used, Maciejowski [2001]. This is a feedback controller that uses the system model to compute the control signal by minimizing a quadratic optimization problem. Two methods are used in thesis to solve this optimization problem. These are explicit MPC, Bemporad et al. [2002], that is a pre-computed solution to the problem, and the fast gradient, Richter et al. [2011], that is an online solver. These methods are used due to their often low computational requirements to obtain the optimal solution. Both methods can in principle give hard real-time guarantees, in the sense that the maximum number of iterations it takes to solve the optimization problem can be pre-calculated. Hence, it is possible to a priori determine whether the requirements of the maximum computational time of 20 ms can be satisfied.

## 6.1 Model Predictive Control

The controller structure that is used in this thesis is a state feedback controller, see Figure 6.1, and the control signal $\mathbf{U}$ can be calculated as $\mathbf{U}_k = L_k(\mathbf{X}_k, \mathbf{R}_k)$, where $\mathbf{X}_k$ is the state space vector and $\mathbf{R}_k$ is the reference signal at time $k$, see Figure 6.1. The reference signal consists of reference values in rotational rates of roll, pitch and yaw, but also the common throttle. The throttle affects all three rotor equally and is controlled with the RC.

$$\mathbf{R} = [\mathbf{R}_{roll}, \mathbf{R}_{pitch}, \mathbf{R}_{yaw}, \mathbf{R}_{th}]^T \tag{6.1}$$

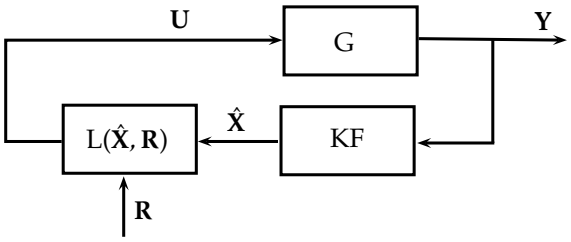



***Figure 6.1:*** *State feedback controller with feedback $L(\cdot)$, Kalman Filter and controlled system G. The input signal to the MPC controller is the reference signal* **R** *and the state* **X**. *The input signal to the system is* **U**.

### 6.1.1 Optimization problem

The control signal is chosen as a solution to a linear quadratic problem

$$\mathcal{U} = [\mathbf{U}_k^T, \mathbf{U}_{k+1}^T, \ldots, \mathbf{U}_{k+N-1}^T]^T \tag{6.2}$$

denoted as $J_N(\mathbf{X}_k, \mathbf{R}_k, \mathbf{U}_k)$ and called cost function, subject to the system dynamics and the limitations of the system. This kind of controller is called MPC, since it uses the system model to predict the output signal at time $k+1, \ldots, k+N$, given the output and the input signals at time $k$.

Instead of using the states in the cost function, the control error, $\mathbf{Y}_k - \mathbf{R}_k$, is of a more interest, since the output signal $\mathbf{Y}_k$ is to follow a reference value $\mathbf{R}_k$ at time $k$. This implies that the cost function $J$ also depends on the given reference signal. The optimization problem can now be written as

$$\min_{\mathcal{U}} J_N(\mathbf{X}_k, \mathbf{R}_k, \mathbf{U}_k) = \min_{\mathcal{U}} \frac{1}{2}\|\mathbf{Y}_{k+N} - \mathbf{R}_{k+N}\|^2_{Q_{1,N}} + \frac{1}{2}\sum_{j=0}^{N-1} \|\mathbf{Y}_{k+j} - \mathbf{R}_{k+j}\|^2_{Q_1} + \|\mathbf{U}_{k+j}\|^2_{Q_2} \tag{6.3a}$$

subject to

$$\mathbf{X}_{k+1} = A\mathbf{X}_k + B\mathbf{U}_k \tag{6.3b}$$

$$\mathbf{Y}_k = C\mathbf{X}_k \tag{6.3c}$$

$$\underline{\mathbf{U}_{i,k}} \leq \mathbf{U}_{i,k} \leq \overline{\mathbf{U}_{i,k}}, \quad i = 1, \ldots, 4 \tag{6.3d}$$

where $Q_1$, $Q_2$ and $Q_{1,N}$ are weighting matrices and $N$ is the horizon for how many time-steps forward the output signal is predicted. The norm $\|x\|^2_Q$ is an Euclidian norm with weight $Q$. The matrices decide how the different control error and input signals affect the cost function. If the weighting is large, the signals will have a greater impact on the cost function than a lower weighted signal. In this way the input signal and control error can be penalized so that they do not differ much from a desired value.

Since the reference signal vector **R** includes the throttle signal, which does not affect the control error, the throttle needs to be cut out from the reference signal

vector

$$\mathbf{R}_k^r = S_r \mathbf{R}_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \mathbf{R}_k \tag{6.4}$$

The upper and lower bound of $\mathbf{U}$ are determined by the limits of the ESC and the servo. The bounds of the signal that controls the servo, $\mathbf{U}_{4,k}$, are $(-300, 300)$ $[\mu s]$, since the PWM value that centers the servo is 1500 $[\mu s]$. The bounds on the signal that controls the ESC, $\mathbf{U}_{i,k},\ i = 1, 2, 3$, are $(1000, 2000)$ $[\mu s]$. Since the motors are controlled by a common throttle signal, the bounds of $\mathbf{U}$ have to depend on this throttle signal. The motor speed cannot be increased if the throttle is at its maximum, or decreased if the motors is shutdown. This gives the bounds on the control signal

$$1000 \leq \mathbf{U}_{i,k} + \mathbf{R}_{th} \leq 2000, \quad i = 1, 2, 3 \tag{6.5a}$$

$$-300 \leq \mathbf{U}_{4,k} \leq 300. \tag{6.5b}$$

With this the lower and upper bound vector of the control signal $\mathbf{U}$ can be defined as

$$\underline{\mathbf{U}_k} = [1000, 1000, 1000, -300]^T \tag{6.6a}$$

$$\overline{\mathbf{U}_k} = [2000, 2000, 2000, 300]^T. \tag{6.6b}$$

These constraints can be rewritten as

$$\mathbf{U}_{i,k} \leq 2000 - \mathbf{R}_{th}, \quad i = 1, 2, 3 \tag{6.7a}$$

$$\mathbf{U}_{4,k} \leq 300 \tag{6.7b}$$

$$-\mathbf{U}_{i,k} \leq -1000 + \mathbf{R}_{th}, \quad i = 1, 2, 3 \tag{6.7c}$$

$$-\mathbf{U}_{4,k} \leq 300 \tag{6.7d}$$

where $\mathbf{U}_{i,k},\ i = 1, 2, 3$ is the PWM signals to the ESCs and the PWM signal to the servo is $\mathbf{U}_{4,k}$. The throttle is denoted $\mathbf{R}_{th}$.

The signals over the entire horizon, $N$, can be expressed as vectors

$$\mathcal{X} = [\mathbf{X}_k^T, \mathbf{X}_{k+1}^T, \ldots, \mathbf{X}_{k+N}^T]^T \tag{6.8a}$$

$$\mathcal{Y} = [\mathbf{Y}_k^T, \mathbf{Y}_{k+1}^T, \ldots, \mathbf{Y}_{k+N}^T]^T \tag{6.8b}$$

$$\mathcal{R} = [\mathbf{R}^{r\,T}_k, \mathbf{R}^{r\,T}_{k+1}, \ldots, \mathbf{R}^{r\,T}_{k+N}]^T. \tag{6.8c}$$

By using this representation of the signals, the cost function can be rewritten as

$$J_N(\mathbf{X}_k, \mathbf{R}_k, \mathbf{U}_k) = \frac{1}{2}(\mathcal{Y} - \mathcal{R})^T \mathcal{Q}_{\mathbf{1}} (\mathcal{Y} - \mathcal{R}) + \frac{1}{2} \mathcal{U}^T \mathcal{Q}_2 \mathcal{U} \tag{6.9}$$

where

$$\mathcal{Q}_1 = \begin{bmatrix} Q_1 & & & \\ & \ddots & & \\ & & Q_1 & \\ & & & Q_{1,N} \end{bmatrix}, \quad \mathcal{Q}_2 = \begin{bmatrix} Q_2 & & & \\ & Q_2 & & \\ & & \ddots & \\ & & & Q_2 \end{bmatrix}. \tag{6.10}$$

The cost function depends on future values of the state $\mathbf{X}$ and the reference signal $\mathbf{R}$ and only the first time value is given. This means that the future states have to be predicted from present value. The first step prediction is given by the model equation (6.3b) and two step prediction can be written as

$$\begin{aligned} \mathbf{X}_{k+2} &= A\mathbf{X}_{k+1} + B\mathbf{U}_{k+1} \\ &= A(A\mathbf{X}_k + B\mathbf{U}_k) + B\mathbf{U}_{k+1} \\ &= A^2\mathbf{X}_k + AB\mathbf{U}_k + B\mathbf{U}_{k+1}. \end{aligned} \tag{6.11}$$

The expression depends only on the present time value of the state. By using the vector representation of the state space vector (6.8a) and of the control signal vector (6.2), the prediction can be written as

$$\mathcal{X} = S_x\mathbf{X}_k + S_u\mathcal{U} \tag{6.12}$$

where

$$S_x = \begin{bmatrix} I \\ A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \quad S_u = \begin{bmatrix} 0 & 0 & \dots & 0 \\ A & 0 & \dots & 0 \\ AB & B & \dots & 0 \\ \vdots & \vdots & \ddots & \dots \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix} \tag{6.13}$$

which is a compact expression for the prediction of the states. Since the output signal is a linear combination of the states, the output at each time step can be rewritten as

$$\begin{aligned} \mathbf{Y}_k &= C\mathbf{X}_k \\ \mathbf{Y}_{k+1} &= C\mathbf{X}_{k+1} \\ &\vdots \\ \mathbf{Y}_{k+N} &= C\mathbf{X}_{k+N} \end{aligned}$$

and these can be written as a vector

$$\mathcal{Y} = \mathcal{M}\mathcal{X} \tag{6.14}$$

where

$$\mathcal{M} = \begin{bmatrix} C & & & \\ & C & & \\ & & \ddots & \\ & & & C \end{bmatrix}. \tag{6.15}$$

By using (6.12) in (6.14) the output vector can be written in terms of the state space vector

$$\mathcal{Y} = \mathcal{M}(S_x \mathbf{X}_k + S_u \mathcal{U}). \tag{6.16}$$

The reference signal is assumed to be constant over the horizon $N$ so $\mathbf{R}_k = \mathbf{R}_{k+1} = \ldots = \mathbf{R}_{k+N}$. The vector with reference signal can be rewritten as

$$\mathcal{R} = \mathcal{S}_R \mathbf{R}_k^r = \begin{pmatrix} S_r \\ \vdots \\ S_r \end{pmatrix} \mathbf{R}_k \tag{6.17}$$

With these equations the cost function (6.9) can be expanded

$$\begin{aligned} J_N(\mathbf{X}_k, \mathbf{R}_k, \mathbf{U}_k) &= \frac{1}{2}\|\mathbf{Y}_{k+N} - \mathbf{R}_{k+N}\|^2_{Q_{1,N}} + \frac{1}{2}\sum_{j=0}^{N-1} \|\mathbf{Y}_{k+j} - \mathbf{R}_{k+j}\|^2_{Q_1} + \|\mathbf{U}_{k+j}\|^2_{Q_2} \\ &= \frac{1}{2}(\mathcal{Y} - \mathcal{S}_R \mathbf{R}_k)^T \mathcal{Q}_1 (\mathcal{Y} - \mathcal{S}_R \mathbf{R}_k) + \frac{1}{2}\mathcal{U}^T \mathcal{Q}_2 \mathcal{U} \\ &= \frac{1}{2}(\mathcal{M}(S_x \mathbf{X}_k + S_u \mathcal{U}) - \mathcal{S}_R \mathbf{R}_k)^T \mathcal{Q}_1 (\mathcal{M}(S_x \mathbf{X}_k + S_u \mathcal{U}) - \mathcal{S}_R \mathbf{R}_k) + \frac{1}{2}\mathcal{U}^T \mathcal{Q}_2 \mathcal{U} \\ &= \frac{1}{2}\mathcal{U}^T (S_u^T \mathcal{M}^T \mathcal{Q}_1 \mathcal{M} S_u + \mathcal{Q}_2)\mathcal{U} + (S_u^T \mathcal{M}^T \mathcal{Q}_1 (\mathcal{M} S_x \mathbf{X}_k + \mathcal{S}_R \mathbf{R}_k))\mathcal{U} \\ &\quad + \mathcal{C}(\mathbf{X}_k, \mathbf{R}_k) \end{aligned} \tag{6.18}$$

where the term $\mathcal{C}(\mathbf{X}_k, \mathbf{R}_k)$ does not include $\mathbf{U}_k$ and can therefore be neglected, since the cost function is minimized with respect to $\mathbf{U}$. The optimization problem can now be expressed in a compact form

$$\min_{\mathcal{U}} \quad \frac{1}{2}\mathcal{U}^T H \mathcal{U} + \theta^T F \mathcal{U} \tag{6.19}$$

$$\text{subject to} \quad \mathbf{X}_{k+1} = A\mathbf{X}_k + B\mathbf{U}_k \tag{6.20}$$

$$\mathbf{Y}_k = C\mathbf{X}_k$$

$$G\mathcal{U} \leq W + E\theta \tag{6.21}$$

where

$$\theta = [\mathbf{X}_k^T, \mathbf{R}_k^T]^T$$

$$H = (S_u^T M^T \mathcal{Q}_1 M S_u + \mathcal{Q}_2) \tag{6.22a}$$

$$F = \begin{pmatrix} S_x^T \mathcal{M}^T \mathcal{Q}_1 \mathcal{M} S_u \\ -\mathcal{S}_R^T \mathcal{Q}_1 \mathcal{M} S_u \end{pmatrix} \tag{6.22b}$$

$$\tag{6.22c}$$

$$G = \begin{pmatrix} I & & & & \\ & I & & & \\ & & \ddots & & \\ & & & & I \\ -I & & & & \\ & -I & & & \\ & & \ddots & & \\ & & & & -I \end{pmatrix} \tag{6.22d}$$

$$W = \begin{pmatrix} \overline{\mathbf{U}}_k & & & \\ & \overline{\mathbf{U}}_{k+1} & & \\ & & \ddots & \\ & & & \overline{\mathbf{U}}_{k+N-1} \\ -\underline{\mathbf{U}}_k & & & \\ & -\underline{\mathbf{U}}_{k+1} & & \\ & & \ddots & \\ & & & -\underline{\mathbf{U}}_{k+N-1}. \end{pmatrix} \tag{6.22e}$$

Since the bounds on the input signal depend on the throttle, the $E$ matrix will be non-zero. The only variable that affects the bounds of the input signal $\mathbf{U}$, is the throttle $\mathbf{R}_{th}$ according to (6.7a). Since the reference vector has the throttle as the last element in the vector according to (6.1), it gives that the matrix $E$ can be defined as

$$E = \begin{pmatrix} 0^{(n_u-1)\times(n_x+n_r)} & -1^{(n_u-1)\times 1} \\ 0^{1\times(n_x+n_r)} & 0 \\ \vdots & \vdots \\ 0^{(n_u-1)\times(n_x+n_r)} & 1^{(n_u-1)\times 1} \\ 0^{1\times(n_x+n_r)} & 0 \\ \vdots & \vdots \end{pmatrix}, \tag{6.23}$$

where $\gamma^{n\times m}$ is a matrix with $\gamma$s and has the size $n \times m$.

### 6.1.2 Move blocking

To decrease the computational time of solving the optimization problem, the amount of optimization variables, $\mathbf{U}$, can be decreased without shortening the length of the prediction horizon, $N$. This can be done by introducing move blocking. This is a method were the control signal is assumed to be constant after a time $N_s$

$$\mathbf{U}_{k+N_s} = \ldots = \mathbf{U}_{k+N}, \quad N_s \leq N, \tag{6.24}$$

which gives that the control signal over the entire prediction horizon $N$ can be written as

$$\mathcal{U}_{Ns} = \Omega_{Ns}\mathcal{U} = \begin{pmatrix} I & & & 0 & \dots & 0 \\ & I & & & & \\ & & \ddots & \vdots & & \vdots \\ & & & I & 0 & \dots & 0 \\ & & & \vdots & \vdots & & \vdots \\ & & & I & 0 & \dots & 0 \end{pmatrix} \mathcal{U}. \tag{6.25}$$

By replacing $\mathcal{U}$ with $\Omega_{Ns}\mathcal{U}$, the optimization problem with move blocking can be written as

$$\min_{\mathcal{U}} \quad \frac{1}{2}\mathcal{U}^T H\mathcal{U} + \theta_k^T F\mathcal{U} + \text{const} \tag{6.26a}$$

$$\text{subject to} \quad G\mathcal{U} \leq W + E\theta \tag{6.26b}$$

$$\begin{aligned} \text{where} \quad \theta_k &= [\mathbf{X}_k^T, \mathbf{R}_k^T]^T \\ H &= \Omega_{Ns}^T (S_u^T \mathcal{M}^T \mathcal{Q}_1 \mathcal{M} S_u + \mathcal{Q}_2)\Omega_{Ns} \\ F &= \begin{pmatrix} S_x^T \mathcal{M}^T \mathcal{Q}_1 \mathcal{M} S_u \Omega_{Ns} \\ -\mathcal{S}_R^T \mathcal{Q}_1 \mathcal{M} S_u \Omega_{Ns} \end{pmatrix} \end{aligned}$$

$$G = \begin{pmatrix} I & & & \\ & I & & \\ & & \ddots & \\ & & & I \\ -I & & & \\ & -I & & \\ & & \ddots & \\ & & & -I \end{pmatrix} \Omega_{Ns}.$$

The matrix elements in $W$ and $E$ remain unchanged, but the size varies with the total number of control signal $\mathbf{U}$ over the entire prediction horizon $N_s$.

## 6.2 Explicit solution

Explicit MPC, Bemporad et al. [2002], is a pre-computed solution of the optimization problem. The strength of this solution is that the problem is pre-computed offline. The parameters, $\theta = [\mathbf{X}_k^T, \mathbf{R}_k^T]^T$, builds a solution space $\mathbf{S}$. The solution space is divided into regions where each region corresponds to a solution $\mathbf{U}_k = F_k\theta + G_k$. The number of regions, $M$, depends on the number of active constraints and free variables.

The online part to this problem is often implemented as search tree to find out in which region the solution is located in, and thereafter use that region to de-

termine $F_k$ and $G_k$. Since the solution is pre-computed the computational time to obtain the solution is low. If the search tree is a binary search tree, the average number of iteration to find the region the solution exists in, will be $log_2(M)$, where $M$ is the number of regions. The worst case requires $M$ iterations to find the optimal solution. In that way, the maximum number of iterations to solve the problem is known and there is knowledge if the controller can solve the problem within 20 $[ms]$.

This results in a fast solver for the optimization problem, but the solver requires a lot of memory space. This is due that the number of regions depends on the number of parameters, the number of optimization variables and the boundaries of the optimization variables. Since the gain $F_k$ and the constant $G_k$, varies with the region, the memory usage will increase with increasing number of regions.

The Karush-Kuhn-Tucker conditions, Hansson [2010], gives that

$$H\mathcal{U} + G^T\lambda = 0, \tag{6.27a}$$
$$G\mathcal{U} \leq W + E\theta \tag{6.27b}$$
$$G_i - W_i - E_i\theta = 0 \tag{6.27c}$$
$$\lambda \geq 0 \tag{6.27d}$$
$$\lambda_i(G_i\mathcal{U} - W_i - E_i\theta) = 0, \tag{6.27e}$$

where $i$ denotes for the $i$:th row of the vector or the matrix. By using these conditions, the control signal can be expressed as

$$\mathcal{U}^* = H^{-1}G_{\mathfrak{A}}^T(G_{\mathfrak{A}}H^{-1}G_{\mathfrak{A}}^T)^{-1}(W_{\mathfrak{A}} + E_{\mathfrak{A}}\theta) \tag{6.28}$$

where $\mathfrak{A}$ denotes the rows of active constraints. This is valid for the active set $\mathfrak{A}$ that fulfills the KKT conditions (6.27)

$$\mathfrak{A}(\theta) = \{i \in \mathfrak{I} | \forall \mathbf{U} : J_N(\mathbf{U}, \theta) = J_N^*(\theta) \Rightarrow G_i\mathcal{U} - E_i - W_i = 0\}. \tag{6.29}$$

The optimal solution is valid for all $\theta$ that satisfy the inequalities in the KKT conditions $A\theta \leq b$, where

$$A = \begin{pmatrix} GH^{-1}G_{\mathfrak{A}}^T(G_{\mathfrak{A}}H^{-1}G_{\mathfrak{A}}^T)^{-1}E_{\mathfrak{A}} - E \\ (G_{\mathfrak{A}}H^{-1}G_{\mathfrak{A}}^T)^{-1}E_{\mathfrak{A}} \end{pmatrix} \tag{6.30}$$

$$b = \begin{pmatrix} W - GH^{-1}G_{\mathfrak{A}}^T(G_{\mathfrak{A}}H^{-1}G_{\mathfrak{A}}^T)^{-1}W_{\mathfrak{A}} \\ -(G_{\mathfrak{A}}H^{-1}G_{\mathfrak{A}}^T)^{-1}W_{\mathfrak{A}} \end{pmatrix}. \tag{6.31}$$

It can be shown that the explicit solution, $\mathbf{U}^*(\theta)$, of the optimization problem is polyhedral piecewise affine. The control signal is calculated as

$$\mathbf{U}_k = \begin{cases} F_1\theta + g_1 & \text{if} \quad A_1\theta \leq b_1 \\ \vdots & \vdots \\ F_M\theta + g_M & \text{if} \quad A_M\theta \leq b_M \end{cases} \tag{6.32}$$

where $M$ is the number of regions.

## 6.3 Fast gradient

The second method to solve the optimization problem (6.26) discussed in this thesis is an online algorithm, called the fast gradient method, Richter et al. [2011]. The strength of this algorithm is that it is an online solver and therefore the implemented controller does not require a lot of memory space as the explicit MPC does. Using the fast gradient method, the maximum number of iterations it takes to solve the problem, can be pre-calculated, similarly to the explicit MPC. This gives information about how many iterations it requires to obtain a solution with specified quality, and therefore there is a knowledge if the controller can solve the problem within the sample time (20 ms).

The optimization problem (6.3) can be written in general form as

$$\begin{aligned} f^* &= \min f(z) \\ \text{subject to} \quad & z \in \mathbb{C}, \end{aligned} \tag{6.33}$$

where $z$ is the optimization variable and $\mathbb{C}$ is the solution set. The fast gradient algorithm used in this thesis, is according to the algorithm in [Richter et al., 2011],

**loop**
$z_{i+1} = arg\min_{z\in\mathbb{C}} \|z - y_i + \frac{1}{L}\nabla f(y_i)\|^2$
compute $\alpha_{i+1} \in (0,1)$ from $\alpha_{i+1}^2 = (1-\alpha_{i+1})\alpha_i^2 + \frac{\mu\alpha_{i+1}}{L}$
$\beta_i = \frac{(\alpha_i(1-\alpha_{i+1}))}{(\alpha_i^2+\alpha_{i+1})}$
$y_{i+1} = z_{i+1} + \beta(z_{i+1} - z_i)$
$i = i + 1$
**end**,

where $\mu$ is a convexity parameter, $L$ is Lipschitz constant and the start value of $\alpha_0$ fulfills $0 < \sqrt{\frac{\mu}{L}} < \alpha_0 < 1$.

The optimal solution to the problem is given by $z^* = \mathbf{U}^*$. This optimal solution $\mathbf{U}_\varepsilon$ is obtained when an absolute tolerance, $\varepsilon$, is fulfilled

$$J_N(\mathbf{U}_\varepsilon|\theta) - J_N^*(\theta) \leq \varepsilon, \quad \varepsilon > 0 \tag{6.34}$$

$$\theta = [\mathbf{X}^T, \mathbf{R}^T]^T. \tag{6.35}$$

The maximum number of iterations is determined by this absolute tolerance. If the tolerance is small, the solution will be very close to the optimal, but will require more iterations. The opposite, if the tolerance is high. The solution is an optimal control signal that depends on the state $\mathbf{X}_k$ and reference signal $\mathbf{R}_k$

$$\mathbf{U}_k^* = L_k(\mathbf{X}_k, \mathbf{R}_k) \tag{6.36}$$

# 7
# Results

Now when the theory behind the methods used in this thesis is presented, it is time to present the usage of these methods and the results. Data collections have been made in three different flight experiments, and one experiment to measure the bias error of the gyroscope. After the data collection, the data have been pre-processed. Data was split up into an identification and a validation set, before identification of the model. The three models that were presented in Chapter 3, were identified. In this case, a reduced version of the *physical* model that only describes the rotational rates of the tricopter, was identified. After the models passed the validation part, an observer based on these identified models, were computed. Finally, when the models were identified and observers were computed, the MPC problem was solved with the two algorithms. The controller was simulated to see that it acts as desired, and thereafter implemented with the observer on the microcontroller.

The limitations of the microcontroller gave that there was not enough resources to implement the *physical* model (3.25), and the identification of the reduced version of the *physical* model was not successful. Therefore the *black-box* and the *black-box virtual* model have been used to implement the control loop on the microcontroller.

## 7.1 Data collection

To be able to identify the models defined in Chapter 3, the input and output signals must be known. These signals can be logged and saved on the external 16Mb memory on the APM1, with the built-in function in ArduCopter software. The frequency of the inner loop is 50Hz, and therefore the signals are sampled

with the same frequency. That gives that frequencies above half of the sampling frequency (Nyquist frequency) can not be correctly determined. But since the purpose with the tricopter in this case is not to perform any advanced flight trajectories, the 50Hz loop will be satisfactory.

The data collection has been divided into four different experiments. Three flight experiments and one experiment where the bias error of the IMU were measured.

### 7.1.1 Flight experiment no. 1: Hovering

In the first flight experiment, the tricopter is hovering with no, or minimal, rotational rates, see Figure 7.1. The purpose with this case is to compute the linearization point $\mathbf{X}_0$, and $\mathbf{U}_0$ .

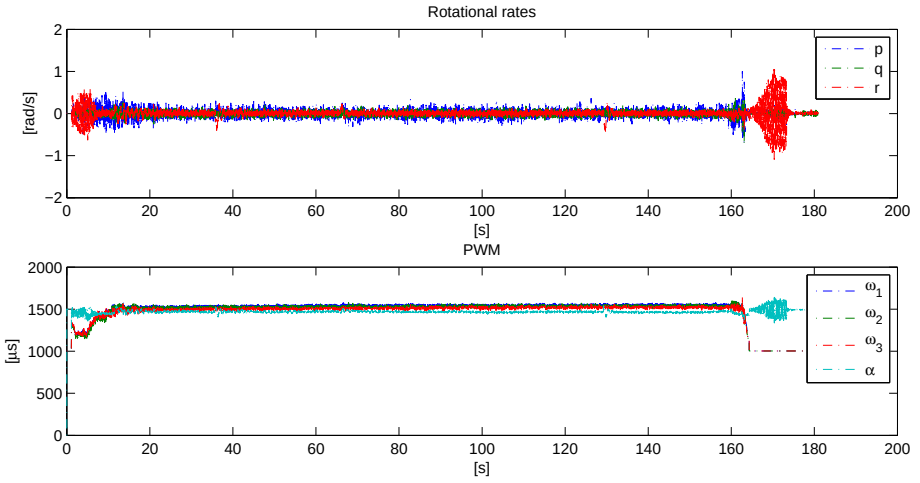



***Figure 7.1:*** *Data collection of the tricopter hovering to obtain the linearization point* $\mathbf{U}_0$.

The rotational rates of the rotor blades depend mainly on three factors: the common throttle $\omega_{th}$, the control signal $\omega_i$ and the linearization input signal $\omega_{i,0}$. The linearization input signal is assumed to be non-zero, since the tricopter is assumed not to be balanced. Therefore it requires different rotational velocities of the motors to achieve the equilibrium point, when the tricopter has no rotational rates.

The common throttle is controlled by the reference throttle signal $\mathbf{R}_{th}$, and acts on all three motors equally. The angle $\alpha$ of the servo is also zero when the tricopter has no rotational rates. While the tricopter is hovering, the servo is tilted with the angle $\alpha_0$ to compensate for the sum of the reaction torques from the rotating motors and blades. If the tricopter is hovering and there are no rotational rates of the tricopter, the control signals are set equal to zero, i.e. $\mathbf{U}_i = 0, \quad i = 1, 2, 3$ and

$\alpha = \mathbf{U}_4 = 0$. This gives that the input signals to the tricopter, while it is hovering, are

$$\mathbf{U} = \begin{pmatrix} \omega_{th} + \omega_1 + \omega_{1,0} \\ \omega_{th} + \omega_2 + \omega_{2,0} \\ \omega_{th} + \omega_3 + \omega_{3,0} \\ \alpha + \alpha_0 \end{pmatrix} = \begin{pmatrix} \omega_{th} + \omega_{1,0} \\ \omega_{th} + \omega_{2,0} \\ \omega_{th} + \omega_{3,0} \\ \alpha_0 \end{pmatrix}. \tag{7.1a}$$

The start and landing are included in the data set, which give that these parts have to be removed, since they contains transients. By removing the throttle signal from the input signals, the linearization input signal was obtained as

$$\mathbf{U}_0 = [\omega_{1,0}, \omega_{2,0}, \omega_{3,0}, \alpha_0]^T = [15.63, 0.85, -16.01, 1467.7]^T \quad \mu\text{s}. \tag{7.2}$$

### 7.1.2 Flight experiment no. 2: Decoupled excitation of dynamics

In the second flight experiment the dynamics of the tricopter are excited. The purpose with this this experiment is to excite the dynamics separately in the three rotational directions, i.e. roll, pitch and yaw, see Figure 7.2. This way, it

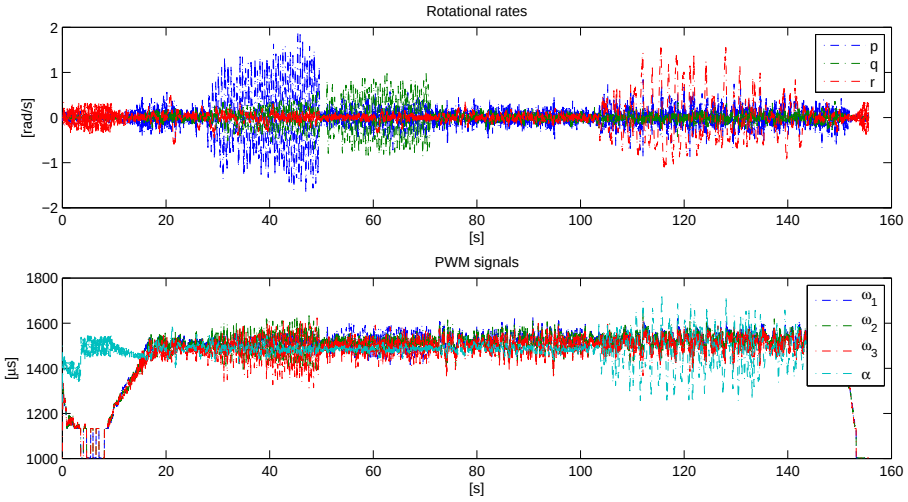



***Figure 7.2:*** *Flight experiment no. 2*

is possible to extract knowledge of how the tricopter acts as a decoupled system, when the three rotational rates are excited separately from each other. The input signals that were applied to the system, were random signals. By doing this, the input sequence consists of several frequencies. Since it was the rotational rates of the tricopter that were controlled, it was very difficult to perform a step response in roll and pitch, as there is a risk that the tricopter will rotate too much, which might cause the tricopter to crash.

### 7.1.3 Flight experiment no. 3: Coupled excitation of dynamics

In this case, compared to the previous case, the rotational rates were not excited separately from each other, see Figure 7.3. The purpose with this experiment is

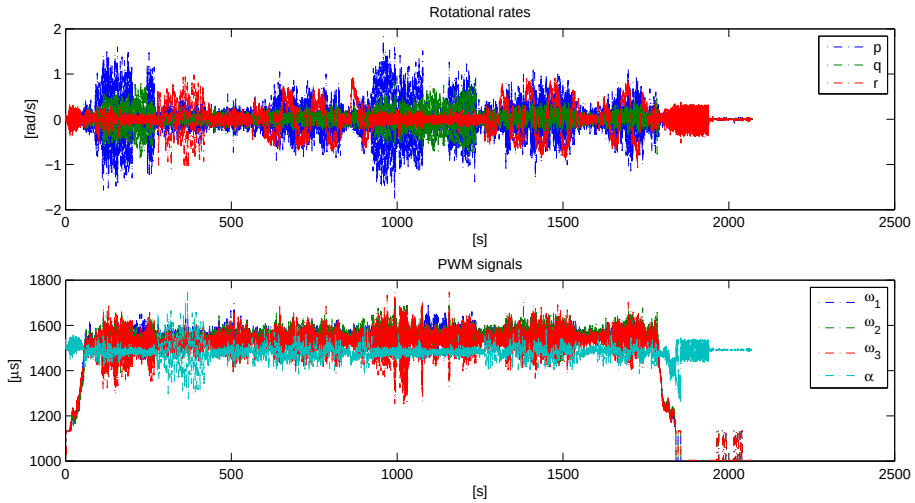



***Figure 7.3:*** *Flight experiment no. 3*

to get knowledge of how the tricopter behaves as a coupled system, when several of the rotational rates are excited at the same time.

### 7.1.4 Experiment: Bias error of IMU

The last experiment that was performed, was a non-flight data collection. In this experiment, the tricopter was standing still on the floor with the motors turned off, see Figure 7.4. The purpose with this experiment is to calculate the bias error in the IMU, so that it can be removed in the control algorithm.

The output signals of the system are the rotational rates, $(p, q, r)^T$ with a measurement noise

$$\mathbf{Y}_k = \mathbf{X}_k + e_k. \tag{7.3}$$

The measurement noise in this case is the disturbance in the gyroscope. Since the gyroscope has a bias error, assumed to be constant, this noise term can be rewritten as

$$e_k = e_k^d + e_0, \tag{7.4}$$

where $e_0$ is the bias error and $e_k^d$ is the disturbance. This disturbance is assumed to be white noise with zero mean and covariance $R_k$. When the tricopter was standing still on the floor, the rotational rates of the rotor craft were equal to zero. That gave that only the measurement noise was the output signal, i.e $\mathbf{Y}_k = e_k^d + e_0$.

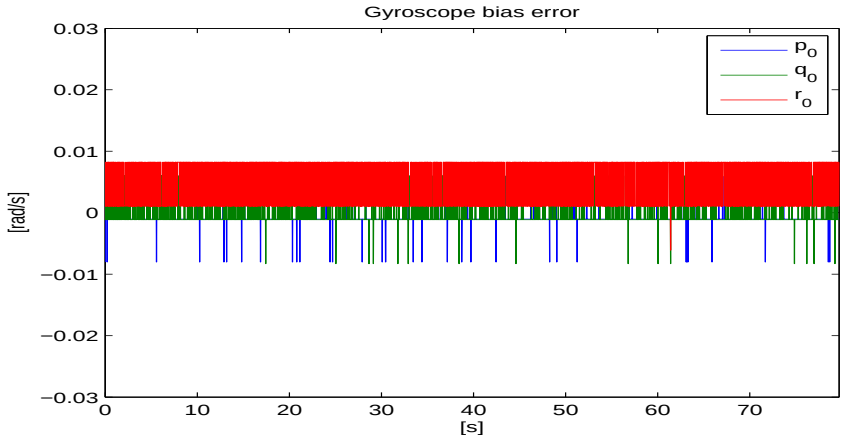



***Figure 7.4:*** *Measured bias error in gyroscope*

This noise term will be the uncertainty of the gyroscope. The measurement is quantized and by using the histogram in Figure 7.5, the bias error can be determined as the mean value

$$e_0 = [-9.72 \cdot 10^{-4},\ -4.92 \cdot 10^{-5},\ 4.27 \cdot 10^{-3}]^T, \tag{7.5}$$

and the covariance of the disturbance term was calculated as

$$R_k = Cov(e_k^d) = \begin{pmatrix} 1.30 \cdot 10^{-6} & -0.02 \cdot 10^{-6} & -0.06 \cdot 10^{-6} \\ -0.02 \cdot 10^{-6} & 7.0 \cdot 10^{-6} & -0.24 \cdot 10^{-6} \\ -0.06 \cdot 10^{-6} & -0.24 \cdot 10^{-6} & 12.67 \cdot 10^{-6} \end{pmatrix}. \tag{7.6}$$

Even if the measurement is quantized and the distribution is not Gaussian, the measurement noise is assumed to be Gaussian with mean $e_0$ and covariance $R_k$.

## 7.2 Pre-processing

The data that was logged with built-in functions in ArduCopter software, was not pre-processed prior to storage. This means that the data includes disturbances, which affect the identification of the model. The identification method, PEM, will try to minimize the prediction error and therefore also try to fit the model with the disturbance noise. Therefore the noise in the measured signals, has to be minimized or only the parts from the experiments where the signal to noise ratio is high, are used in the identification.

The data collection was performed from when the tricopter stood still on the ground and until the tricoper had landed. However, only relevant parts of the experiment were used for identification. For example, take-off and landing maneuvers were not considered.

The measurements from the gyroscope included the bias error and this error is

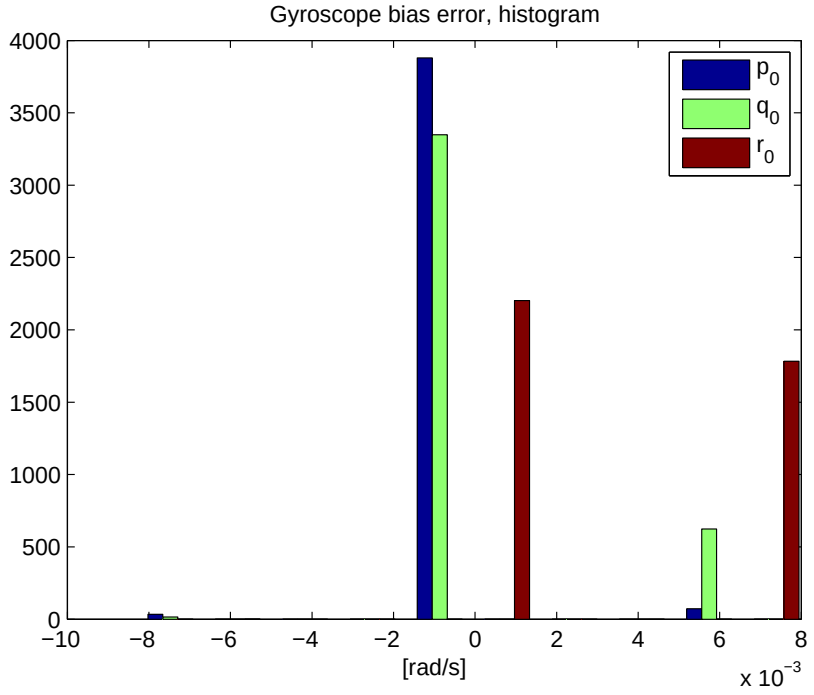



***Figure 7.5:*** *The values of the gyroscope*

affecting the measurements. If the noise term is equal to zero, the linearization point $\mathbf{X}_0$ (rotational rates equal to zero) will give the output signal equal to zero. But the output signal, $\mathbf{Y}_0$, was not $[0, 0, 0]^T$ due to the bias error. All levels, including the common throttle and the linearization point, must be removed from the data sets.

### 7.2.1 Filtering

The measurement noise will affect the identification of the system. Therefore, it would be good if this noise could be minimized without any loss of the information of the dynamics. By assuming that the noise is in a higher frequency band than the dynamics of the tricopter, this noise can be reduced by an LP-filter. A spectrum of the output data from flight experiment no. 3 can be seen in Figure 7.6, and the spectra is presented for roll rate. The frequency peak is similar, a bit higher, than for pitch rate and is much greater than for the yaw rate. The peak frequency of the roll rate is approximately 11 $[rad/s]$.

An LP filter with cutoff frequency 20 [rad/s] (to avoid the peak frequency), was used in the SYSTEM IDENTIFICATION GUI to filter the signals, see Figure 7.7. By filtering the noise out, there will also be information loss of the dynamics, which can be seen in the figure. In the interval 100 to 150 $[s]$, there can be seen that

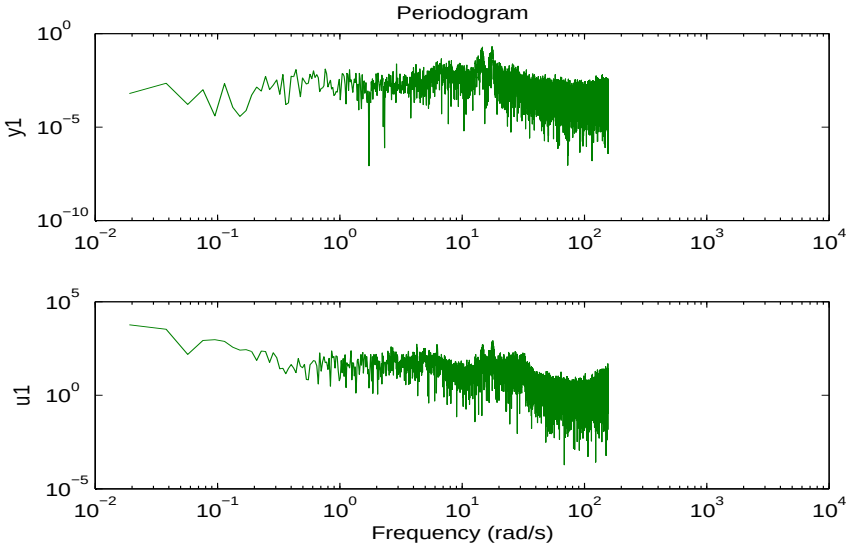



***Figure 7.6:*** *Data spectra of the roll rate from flight experiment no. 3*

the filtered output signal is missing information that the non-filtered signal has. The conclusion is that the noise is in the same frequency band as the dynamics of the tricopter. The measured signal is not white noise, since it is colored by the dynamics of the tricopter. To reduce the signal-to-noise ratio, the parts where the ratio is high are picked from the measured data.

### 7.2.2 Data sets for estimation and validation

The estimation of the model is performed with an estimation set including the input and output signals to the system. The validation of the model is performed with a different set. This gives that the interesting parts of the experiments have to split up into two different sets. Only the parts from the experiments where the signal to noise ratio is high, have been used in the estimation and validation sets. These parts were merged by the MATLAB function `merge(data1, data2, ...)` into one estimation set and one validation set. By doing this, the data will be seen as three different experiments in one data set. Otherwise there will be discontinuous parts if the data parts are include in one data set.

#### Data set for physical and black-box models

The *black-box* and *physical* model uses the angular velocities of the rotors as input signal and the rotational rates of the tricopter as output signal. This gives that the same data sets can be used for the both models.

The estimation set is built up by intervals of interest in flight experiment no.2 and no. 3. The estimation and validation sets consist of following intervals:

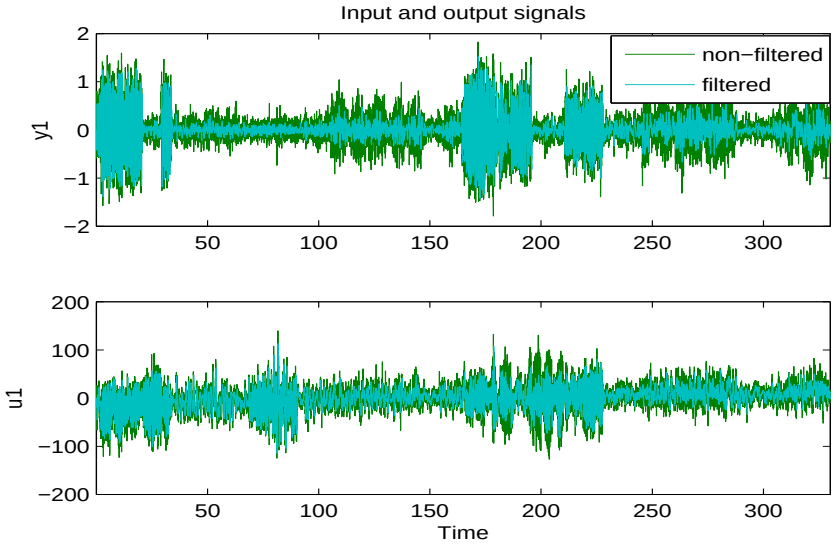



***Figure 7.7:*** *Data spectra of the roll rate, (y1 = p), from flight experiment no. 3, filtered with 20 [rad/s].*

| **Estimation set** | |
|---|---|
| Experiment | $[s]$ |
| flight no. 2 | (25.66, 50.28) |
| flight no. 2 | (50.46, 71.80) |
| flight no. 3 | (108.90, 165.80) |

| **Validation set** | |
|---|---|
| Experiment | $[s]$ |
| flight no. 2 | (102.94, 126.12) |
| flight no. 3 | (21.28, 42.82) |
| flight no. 3 | (181.66, 219.60) |
| flight no. 3 | (227.28, 249.84) |

### Data set for black-box virtual model

The estimation set that is used to identify the *black-box virtual* model, is only picked from flight no. 2. This is because the *black-box virtual* model is using the virtual input signals. The model will try to act as a decoupled system, i.e. only the virtual roll input should affect the roll rate, etc. The interesting parts from flight experiment no. 2, where the signal to noise ratio is high, are picked. These parts where split up into two sets and the resulting estimation and validation sets consists of following intervals:

| **Estimation set** | |
|---|---|
| Flight | $[s]$ |
| flight no. 2 | (25.66, 37.96) |
| flight no. 2 | (54.48, 62.64) |
| flight no. 2 | (101.50, 111.44) |

| **Validation set** | |
|---|---|
| Flight | $[s]$ |
| flight no. 2 | (37.98, 50.28) |
| flight no. 2 | (62.66, 70.80) |
| flight no. 2 | (111.50121.40) |

## 7.3 Limitations of hardware

Before identifying the models, the hardware was tested to see what the computational limits are and what order of model can be estimated. The data logging was performed at 50Hz and the desired control loop will have the same update frequency of the control signal. This gives that the observer and the controller will have to finish their tasks within 20ms, otherwise the control loop would be too slow and miss the hard deadline. The dynamics of the tricopter will probably be in a frequency band a bit higher than what the logging can observe. This can give problem with the identification of the system. Since the microcontroller combined with the external memory was limited to log the input and output signals with higher frequency, the 50Hz logging has been used.

There is also a limit with the memory in the micro controller. The RAM, Random Access Memory, is limited to 8kB and the flash memory, which is the memory for the program code, is limited to 128kB. This later limitation is due to the combination of usage of old version datatypes in the ArduCopter code and the compiler version of AVR GCC, AVR Gnu Compiler Collection, AVR-GCC [2012]. Therefore the entire program memory, 256kB, cannot be used. Another limit with the microcontroller is that the microcontroller use 16 bits addressing, which limits the size of the arrays in the C-code. One array cannot be greater than $2^{16} = 65\,536$ bytes, since the indexing of the array elements can not be greater than that. With the usage of the datatype `float` as the representation of the matrix elements, the size of each element will be $2^2 = 4$ bytes. This gives that one array cannot have more than $2^{14} = 16384$ elements.

The explicit MPC use a binary search tree to find out in which region the optimal solution is in. This search tree is generated as an array by the function `mpt_exportST(ctrl_struct)`, which can be found in MPT TOOLBOX in MATLAB, and generates a C-code array of the matrix. The search tree consists of: $n_x$ state variables, $n_r = 4$ reference variables, one constant term and two values that gives information about the left and the right subtree. This gives that there cannot be more than $\frac{16384}{n_x+4+1+2} = \frac{16384}{n_x+7}$ rows in the search tree array. Since the RAM is limited to 8kB, the arrays are implemented in the program space instead of the

RAM memory.

A test of the computational efficiency of the microcontroller was performed. Random matrix multiplications that corresponds to the measurement and time update of the dynamical Kalman filter, see Section 5.2, were performed and clocked. A random 12 x 12 state space model was tested, which corresponds to the physical model that was linearized. This test resulted in that the filter can not finish the task within 20ms and the entire *physical* model (3.21) that is linearized, can not be implemented on this microcontroller.

A dynamical Kalman filter was also implemented with a random 4th order state space system. The time to complete the measurement and the time update was approximately 20ms, which gives no time for the controller to compute the control signal. The part that is computationally demanding is the matrix inversion. This is because of the handling of floating points. The microcontroller cannot handle floating points and will therefore have to emulate these. This gives that all divisions are very computationally demanding. Since the matrices in the state space model are time-invariant, the stationary Kalman filter, see Section 5.2.1, was used. The matrix multiplications in the measurement and time update of the stationary Kalman filter, can be pre-calculated since they are constant. Therefore the microcontroller can complete the measurement and time update in approximately 3 $[ms]$, using the *black-box* model.

The explicit MPC cannot be implemented without using move blocking with $N_s = 1$, i.e. the control signal will be constant after the first time step in the optimization problem. If $N_s > 1$, the number of regions will be too big and the result is that the search tree cannot be implemented on the microcontroller. By using this horizon for the control signal, the MPC can be seen as an LQ-controller that has constraints on the control signal.

## 7.4 System identification

With the data pre-processed and the limitations of the hardware known, the identification of the model can be performed. The first step with the system identification was to choose a model structure. Three different model structures were used to identify the system to see which one describes the dynamics of the system most accurately. The models were identified using SYSTEM INDENTIFICATION TOOLBOX GUI in MATLAB with the estimation method PEM, as described in (4.10).

### 7.4.1 Model structure

To perform a system identification a model structure has to be chosen. The core of the structure is a linear discrete time state space model

$$\begin{aligned} \mathbf{X}_{k+1} &= A(\theta)\mathbf{X}_k + B(\theta)\mathbf{U}_k \\ \mathbf{Y}_k &= C(\theta)\mathbf{X}_k \end{aligned}$$

where the matrix elements decide the model structure. The three different model structures that were used: the *physical*, the *black-box* and the *black-box virtual* model. The structure of the *physical* model is known, which makes it a Grey-box model. Both the *black-box* and the *black-box virtual* models are Black box models, since the structure of their dynamics is unspecified. The difference between the two Black box models, are the input signals to the model. The input signals to the virtual model are a linear transformation, (3.28), of the input signals to the *black-box* model.

### Grey-box structure

The structure of the *physical* model is a model where the structure of the system matrices are known. The equations that describe the rotational rates of the tricopter (3.19), are derived from laws of physics. These are continuous time equations that are nonlinear. The equations are linearized around $[\mathbf{X}_0, \mathbf{U}_0] = [0, 0, 0, \omega_{1,0}, \omega_{2,0}, \omega_{3,0}, \alpha_0]$ and the resulting linear continuous time model structure is

$$A(\theta) = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad B(\theta) = \begin{pmatrix} 0 & b_{10,2} & b_{10,3} & 0 \\ b_{11,1} & b_{11,2} & b_{11,3} & b_{11,4} \\ b_{12,1} & b_{12,2} & b_{12,3} & b_{12,4} \end{pmatrix},$$

$$C(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{7.7}$$

$$\mathbf{Y}(t) = \mathbf{X}(t) = [p(t), q(t), r(t)]^T \tag{7.8}$$

$$\mathbf{U}(t) = [\omega_1, \omega_2, \omega_3, \alpha]^T, \tag{7.9}$$

where the parameters $b_{i,j}$ are presented in Appendix B. Since the electrical motors and the servo are controlled by the PWM signals, the delay in the motors and servo, defined in Section 3.1.3, can be introduced. The three first elements in the main diagonal of the $A$ matrix, are assigned value $-10^{-9}$, so that these three poles are numerically in the left side of the complex plane. This will reduce the initial problem with a marginally unstable system. The resulting time continuous $A$ and $B$ matrices are presented in Appendix C. This model was discretized as per Equation (4.1) before the identification to obtain the discrete model that will be identified.

### Black-box model structure

The *black-box* model structure is a Black box model where the input signals are the rotational rates of the rotors, $\omega_i$, $i = 1, 2, 3$ and the tilt angle of the servo, $\alpha$. The output signals of this model are the rotational rates of the tricopter, $(p, q, r)$. The dynamics of this model are unknown, and since it is a Black box model, the order of the state can be varied to make the model include all the dynamics of

the tricopter. The model structure was

$$A(\theta) = \begin{pmatrix} a_{1,1} & \dots & a_{1,n_x} \\ \vdots & & \vdots \\ a_{n_x,1} & \dots & a_{n_x,n_x} \end{pmatrix}, \quad B(\theta) = \begin{pmatrix} b_{1,1} & \dots & b_{1,4} \\ \vdots & & \vdots \\ b_{n_x,1} & \dots & b_{n_x,4} \end{pmatrix},$$
$$C(\theta) = \begin{pmatrix} c_{1,1} & \dots & c_{1,n_x} \\ \vdots & & \vdots \\ c_{3,1} & \dots & c_{3,n_x} \end{pmatrix}, \tag{7.10}$$

where the matrix elements $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$ are the parameters that were identified. The order of the model is given by number of the states $n_x$. The output signals are

$$\mathbf{Y}_k = [p_k, q_k, r_k]^T, \tag{7.11}$$

and the input signals to the model are

$$\mathbf{U}_k = [\omega_1, \omega_2, \omega_3, \alpha]^T. \tag{7.12}$$

#### Black-box virtual model structure

The *black-box virtual* model structure is similar to the *black-box* model structure. It is a model where the input and output signals are known and the dynamics is unknown. The difference between the two models is the input signal. The *black-box virtual* model uses the virtual input signals, $\mathbf{U}_v$, from (3.28), and the model is given by

$$A(\theta) = \begin{pmatrix} a_{1,1} & \dots & a_{1,n_x} \\ \vdots & & \vdots \\ a_{n_x,1} & \dots & a_{n_x,n_x} \end{pmatrix}, \quad B(\theta) = \begin{pmatrix} b_{1,1} & \dots & b_{1,3} \\ \vdots & & \vdots \\ b_{n_x,1} & \dots & b_{n_x,3} \end{pmatrix},$$
$$C(\theta) = \begin{pmatrix} c_{1,1} & \dots & c_{1,n_x} \\ \vdots & & \vdots \\ c_{3,1} & \dots & c_{3,n_x} \end{pmatrix}, \tag{7.13}$$

where the matrix elements $a_{i,j}$, $b_{i,j}$ and $c_{i,j}$ are the parameters that were identified. The order of the model is given by the number of the states $n_x$. The output signals are

$$\mathbf{Y}_k = [p_k, q_k, r_k]^T, \tag{7.14}$$

and the input signals to the model are

$$\mathbf{U}_k = [\mathbf{U}_r, \mathbf{U}_p, \mathbf{U}_y]^T. \tag{7.15}$$

### 7.4.2 Model estimation

With the limitations of the system are known and the model structures defined, it is time to identify the system using the three different model structures. The estimation of the model parameters was done with the PEM method and the data

set used in the estimation was the estimation set defined in Section 7.2. The identified model was validated against the validation data set.

### Estimation of Grey-box model

First the *physical* model structure was used to identify the system. This is a Grey-box model, since the structure of the model is known. The Grey-box can be generated by function `idgrey('model', param, Ts, 'd', aux, Ts)` in MATLAB, where the model structure is written as a function denoted `'model'`. This model in then used as an initial model to identify a state-space mode with the PEM. Since the model is not a convex function, there need to be a good initialization of the parameters. Otherwise, there is a chance that the minimization of the prediction error will result in a local minimum.

The initial parameters to the model are unknown. So these parameters where taken from [Yoo et al., 2010], which has a different tricopter and therefore the parameters would not be the same, see Table 7.1. But it is a good initial guess, as to which order of magnitude the parameters of inertia, $I_i$, and the aerodynamic constant, $k_i$, is. The initial parameters are given by By using these parameters,

***Table 7.1:*** *Tricopter parameters*

| Parameter | Value | Unit (SI) |
|---|---|---|
| $I_{xx}$ | 0.2 | $kgm^2$ |
| $I_{yy}$ | 0.1 | $kgm^2$ |
| $I_{zz}$ | 0.1 | $kgm^2$ |
| $k_\tau$ | $2^{-7}$ | $Nms^2/rad^2$ |
| $k_t$ | $2^{-6}$ | $Ns^2/rad$ |
| $l$ | 0.5 | $m$ |
| $\omega_{1,0}$ | $2\pi \cdot 50$ | $rad/s$ |
| $\omega_{2,0}$ | $2\pi \cdot 50$ | $rad/s$ |
| $\omega_{3,0}$ | $2\pi \cdot 50$ | $rad/s$ |
| $\alpha_0$ | 0.12 | $rad$ |

the resulting model had a fit of 0.006% by using the cross validation, described in Section 4.3. Since the fit is approximately 0%, this means that the model can describe the dynamics of the system, with the same accuracy as a constant value does. By changing the parameters, the model fit could not be increased, which could be due to some kind of a numerical problem. Therefore this model has not been used in this thesis to compute the Kalman filter and the controller.

### Estimation of Black-box model

The next model structure that was used for identification of the system, was the *black-box* and the *black-box virtual* model. These two models are of a Black-box structure, where only the in- and output signals are known. The order that gave best fit was a model with order 8. This model was then validated in three ways, and these three methods are described in Section 4.3.

A simulation of the output was made using the validation data set. The simulated roll rate, using the *black-box* and the *black-box virtual* model can be seen in Figures 7.8, 7.9, respectively.

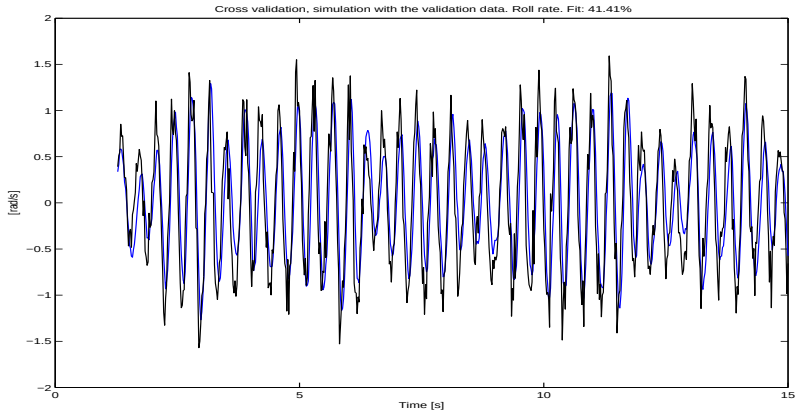



***Figure 7.8:*** *Cross validation of the black-box model with simulation of the roll rate, given the validation data set.*

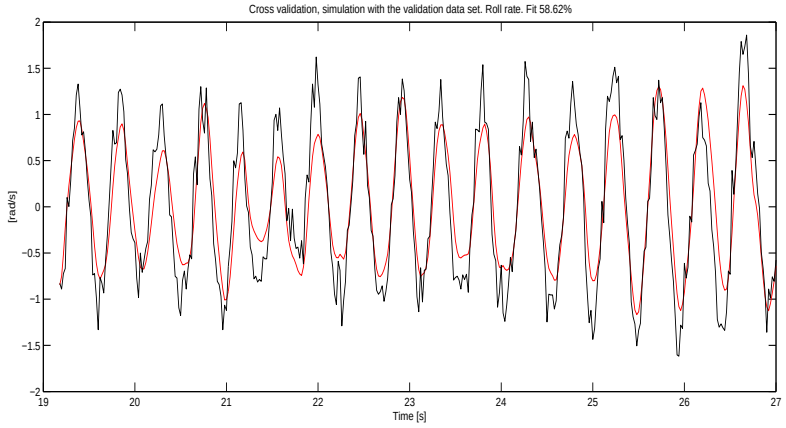



***Figure 7.9:*** *Cross validation of the black-box virtual model with simulation of the roll rate, given the validation data set.*

The fits for the three simulated outputs were: The fit of the simulated outputs were not that high, but the simulated output appears to follow the measured output. The simulated roll rate using the *black-box virtual* model was a bit better than by using the *black-box* model. Otherwise the two models simulate the output signal with comparable accuracy.

| **Cross validation of** *black-box* **model** | | | |
|---|---|---|---|
| Outputs: | p (roll) | q (pitch) | r (yaw) |
| Fit [%] | 41.41 | 43.08 | 42.7 |
| **Cross validation of** *black-box virtual* **model** | | | |
| Outputs: | p (roll) | q (pitch) | r (yaw) |
| Fit [%] | 58.62 | 45.23 | 42.25 |

The next validation method is the k-step prediction. A step length of 5 was used, since it will be a bit longer than the prediction length in the MPC, and it is therefore a good validation if the output 5 steps ahead can be predicted accurately. The 5-step prediction using the *black-box* and the *black-box virtual* model can be seen in Figures 7.10, 7.11, respectively. The fits for the three predicted outputs

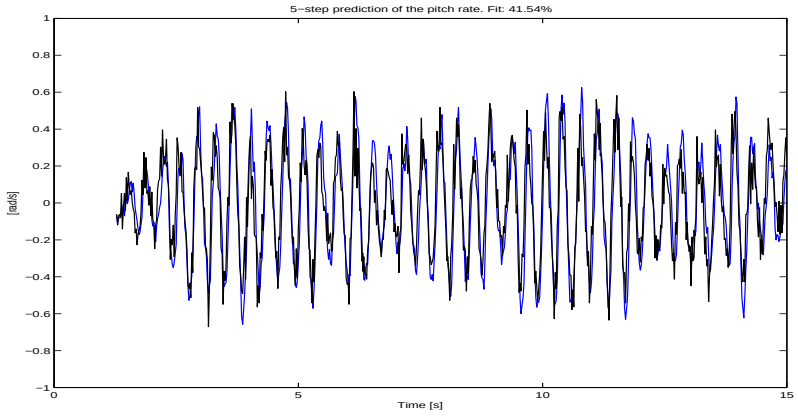



***Figure 7.10:*** *Validation of the black-box model by using 5-step prediction of the pitch rate, given the validation data set.*

were: The conclusion from the 5-step prediction was that the *black-box virtual*

| **5-step prediction with** *black-box* **model** | | | |
|---|---|---|---|
| Outputs: | p (roll) | q (pitch) | r (yaw) |
| Fit [%] | 31.39 | 41.54 | 52.18 |
| **5-step prediction with** *black-box virtual* **model** | | | |
| Outputs: | p (roll) | q (pitch) | r (yaw) |
| Fit [%] | 60.07 | 51.83 | 51.03 |

model can predict the output signal more accurately than the *black-box* model. All three rates where predicted with more than 50% accuracy, using the *black-box virtual* model. This compared to the *black-box* model, which only predicted the yaw rate with more than 50% accuracy.

The last validation method is the residual analysis. This gives a hint if the output

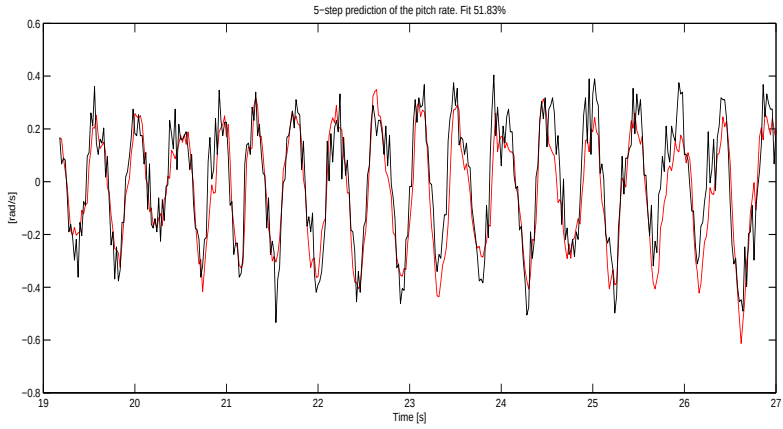



***Figure 7.11:*** *Validation of the black-box virtual model by using 5-step prediction of the pitch rate, given the validation data set.*

signal depends on earlier time values of the output signal and how the output signal signal is cross correlated with the input signal. The residual analysis for the *black-box* and the *black-box virtual* model can be seen in Figures 7.12, 7.13, respectively.

The result from the residual analysis is that the *black-box* model has much more correlation between the input signal $\alpha$ (tilt angle) and output signal $r$ (yaw rate), than the *black-box virtual* model. The autocorrelation of the prediction error is non-zero for non-zero time steps. This gives a hint that the model is not including all the dynamics of the system. Since the microcontroller cannot handle model with order greater that 8, these two models are used to predict and simulate the output signals.

A model of order 8 resulted in a control loop with a computational time close to the deadline (20 [*ms*]). Therefore the model order was reduced to 5, with the MATLAB function `mod5 = balred(mod8,5)`. Another reason was that the explicit MPC was difficult to implement with a model of order 8, due to a large search tree. The same validation were performed for order 5, as for order 8, and there were barely noticable difference in the model fit. Therefore, the model of order 5 has been used further on.

These resulting discrete time Black-box models have zeros outside the unit circle, i.e. in the right side of the complex plane in the continuous time case, which gives that there will be a performance limit of the controller, according to [Glad and Ljung, 2003]. That is because the gain of the system during fast sequences, has the opposite sign compared to slower sequences. Therefore it will be difficult to tune the controller for both slow and fast sequences. The focus will be on slower sequences, since the output signal is desired to follow a reference value.

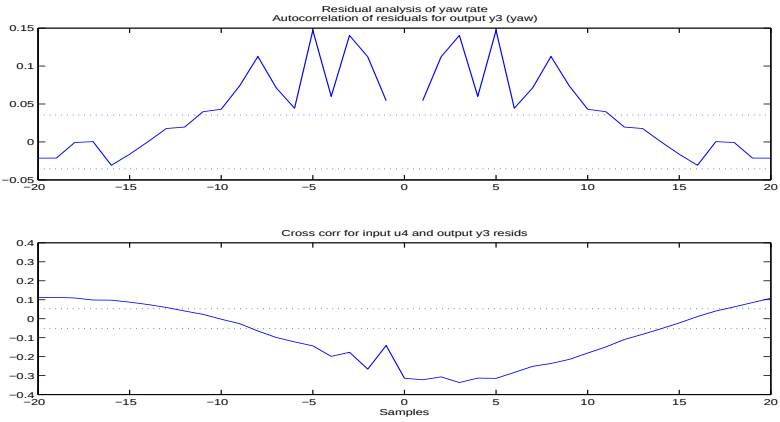



***Figure 7.12:*** *Validation of the black-box model by residual analysis of the yaw rate. The autocorrelation of r and the cross-correlation between α and r have been computed.*

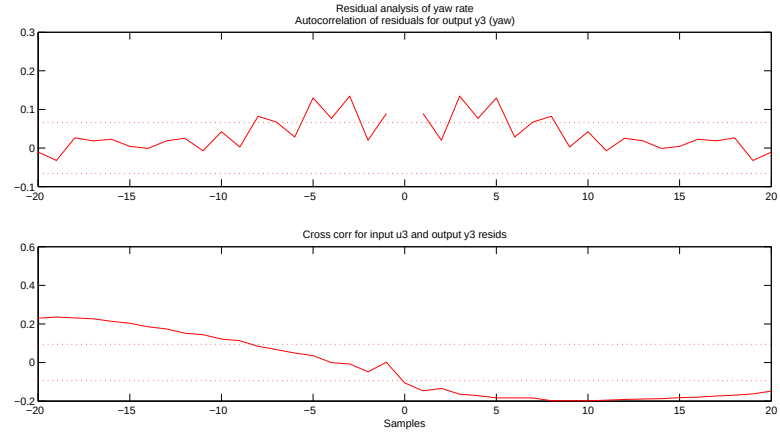



***Figure 7.13:*** *Validation of the black-box virtual model by residual analysis of the yaw rate. The autocorrelation of r and the cross-correlation between α and r have been computed.*

## 7.5 Kalman Filter

The Kalman filter used in this thesis was the stationary Kalman filter, defined in Section 5.2.1. The dynamical Kalman filter could not be implemented on the microcontroller due to the limitations of the hardware. Since the estimated system model is a linear, time-invariant model, the stationary Kalman filter could be used. The computational time for the measurement and time update of the sta-

tionary Kalman filter is approximately 3 [*ms*], for a model with 3 output signals, 5 states and 4 input signals. This will give the control algorithm approximately 17 [*ms*] to compute the optimal control signal.

Since the *physical* model could not describe the dynamics accurately, the *black-box* and *black-box virtual* models were used to compute a stationary Kalman filter.

The stationary Kalman gain (5.15d) can be computed by solving the stationary Riccati equation (5.15c) with the function `dlqe` in MATLAB. The resulting filters were then used to filter the flight experiment no. 3 to see that the filter can estimate the state of the system, i.e. estimate the output signal, since $\hat{\mathbf{Y}} = C\hat{\mathbf{X}}$. The estimated output signal from the Kalman filter, based on the *black-box* and the *black-box virtual* model, can be compared with the measured output signals in Figures 7.14, 7.15, respectively.

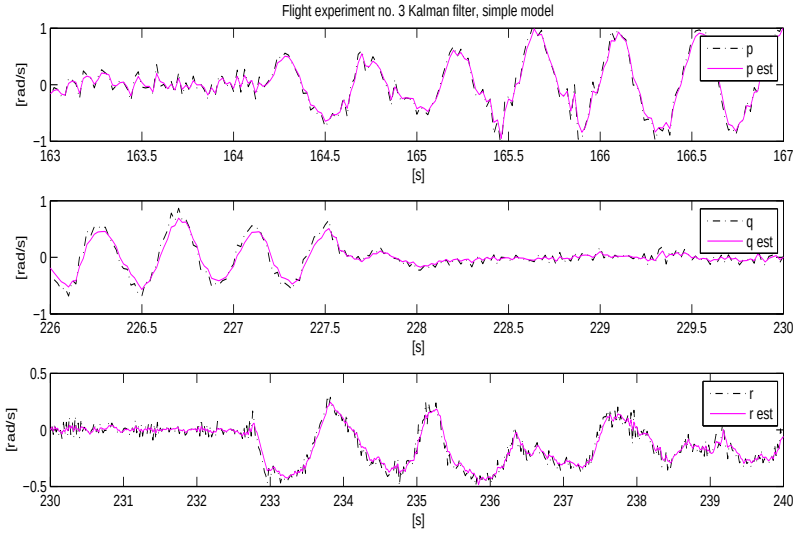



***Figure 7.14:*** *Kalman filtering the measurement from flight experiment no. 3, using the black-box model.*

The result are two stationary Kalman filters, based on the *black-box* and the *black-box virtual* model, which can estimate the output signal accurately. These Kalman filters can be used to estimate the states, given the output and input signal, that will be used in the MPC. The implemented filter will have the measurement and time update according to (5.15). As can be seen in Figure 7.14 and 7.15, the output signal can be estimated accurately and the noise is reduced. The output signal roll rate, $p$, estimated with the *black-box* model has a high process noise since the model uncertainty was high. Therefore the observer will rely on the measurement more than on the model.

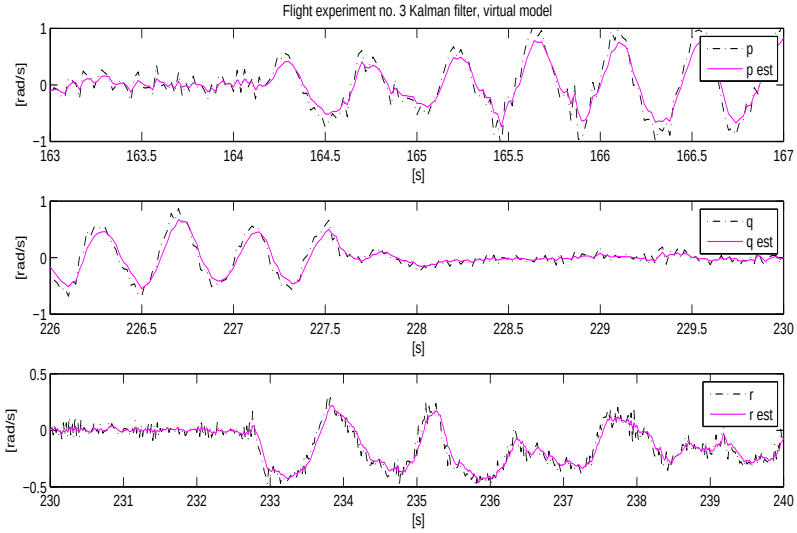



***Figure 7.15:*** *Kalman filtering the measurement from flight experiment no. 3, using the black-box virtual model.*

## 7.6 Controller

The controller used in this thesis is an MPC. This is a controller that uses the system model to compute the optimal control signal, given the state of the tricopter and the reference signals sent from the RC. This differs from the PI-controller that is originally implemented in ArduCopter software, that only use the deviation of the output signals from the reference signal to compute the control signal. The acro mode (acrobat) in ArduCopter is a rate controller and this controls the same output signals as the MPC, based on the *black-box* and the *black-box virtual* model.

The two strategies to solve the optimization problem (6.26) was explicit MPC, Section 6.2, and the fast gradient method, Section 6.3. Since there are limitations in the hardware, these algorithms have to satisfy the limitations, to be able to be implemented on the microcontroller.

Since the regions of the explicit MPC require memory space for the search tree, the search tree array cannot consists of more than 4096 elements.

Both the explicit MPC and fast gradient method have to be able to compute the optimal solution within 17 $[ms]$, (20 $[ms]$ with the measurement and time update of the Kalman filter). The explicit MPC has an advantage compared to the fast gradient method. The explicit MPC is pre-solved and the search tree is a binary tree, which makes the computational time to obtain the solution, compared to the fast gradient method, very low. This works only when the amount of regions is low. If the search tree is very large, the search time will increase. This com-

pared to the fast gradient where the computational time depends on number of iterations that have to be made, which is bounded.

### 7.6.1 Explicit MPC

The first algorithm used in this thesis to solve the MPC problem, was the explicit MPC. This is a pre-computed solution of the MPC problem. The online solution contains a search tree to find the region where the optimal solution can be found and then use the region to compute the control signal. The MPC problem (6.26) is constructed by the MPT TOOLBOX in MATLAB.

The MPT TOOLBOX is also used to generate the search tree and to export the tree to C-code. The function
`[Pn,Fi,Gi,activeConstraints,Phard,details]=mpt_mpqp(Matrices)`
was used to compute the search tree, `Pn` (not binary), and the matrices $F$ and $G$ to the polyhedral piecewise linear solution (6.32), i.e. $\mathbf{U}_k = F_k[\mathbf{X}_k^T, \mathbf{R}_k^T]^T + G_k$. The search tree was collected as a controller struct, `tricptCtrl`, and the search tree in the controller struct was changed to a binary tree with the function `tricptCtrlST = mpt_searchTree(tricptCtrl)` and then exported to C-code by `mpt_exportST(tricptCtrlST)`.

The explicit MPC algorithm was very demanding to implement on the microcontroller. If no move blocking was used, the *black-box* and *black-box virtual* model could not be implemented, since the order of the model had to be less than 4. Therefore move blocking was introduced to simplify the complexity of the solution. The length of the move blocking could not be larger than 1, i.e. $N_s = 1$, since otherwise the number of regions would be too large and the microcontroller would not have enough memory for the search tree. The difference $N - N_s$ determines how many steps the predicted control signal will be constant. Therefore the prediction horizon $N$ was chosen few steps greater than the prediction horizon of the control signal, $N_s$.

### 7.6.2 Fast gradient

The second algorithm used in this thesis to solve the MPC problem, was the fast gradient method. This is an online method to solve the optimization problem. The FIORDOS toolbox, FiOrdOs [2012], for MATLAB was used to generate the solver. This is a C-code generator of the fast gradient solver that can be used both in MATLAB, with MATLAB's C-compiler (MEX), and in C-applications.

The problem formulation is similar to the formulation of the explicit MPC. But in this case the constraints will be formulated as a box $\mathbf{U} \in \mathbb{R}^4$, that depends on the free throttle parameter, $\mathbf{R}_{th}$,

$$\mathbf{U} \in \mathbb{U} = \left\{ \mathbf{U} \in \mathbb{R}^{4N_s} | l \leq \mathbf{U} \leq u \right\}, \tag{7.16}$$

where $l$ and $u$ are the lower and upper bounds, (6.26b). These are seen as parameters, since the lower and upper bound of the control signal depends on the throttle signal.

The matrices in (6.26) are placed in a struct `s`. This struct includes also the settings for the solver. Pre-conditioning is a method to reduce the condition number of the objective function via a linear change of variables. This helps to increase the convergence speed. The pre-conditioning was made by the function call `res = s.setAutoPreconditioning('algo')`, where `'algo'` denotes for fast gradient.

The sequence $\{\beta_k\}_{k=0}^{k_{max}-1}$ in the fast gradient algorithm can be precalculated with the function `s.setSettings('algo','precalcbetas','true')`.

The main reason why the fast gradient method was used in this thesis, was that the maximum number of iterations can be pre-calculated. This is done with the function `it = s.certify('algo','ε')`, where $\varepsilon$ is the absolute tolerance for the solver.

The optimization problem is placed in a struct denoted `s` and the solver is generated by `s.generateCode('prefix','tricpt_')`. The resulting solver from the C-code generator is placed in two files:
`tricpt_solver.c` and `tricpt_solver.h`.

The only computations that need to be made online is the computations of the constraints, since they depend on the throttle, and the computation of the linear term in (6.26), i.e.

$$F^T\theta = F^T[\mathbf{X}^T, \mathbf{R}^T]^T, \tag{7.17}$$

since the parameter $\theta$ is varying in time. Therefore, the only matrix that needs to be saved on the microcontroller, is $F$.

### 7.6.3 Limitations of the controller

The microcontroller has limited computational resources as mentioned in Section 7.3. This gives that the complexity of the controller has to be low. The explicit MPC is bounded by the size of the memory, while the fast gradient method is bounded by the computational time of each iteration. The number of regions of the explicit MPC solver gave us that a horizon length of 2 or more, was not tractable on the microcontroller, if not move blocking was introduced. By introducing move blocking with $N_s = 1$, the prediction horizon could be increased, since the control signals are constant through the entire prediction horizon. That way, there are two horizon parameters to the MPC problem. But these parameters should not differ too much from each other. If the control signal is constant for a longer period, there will be problem when the constraints become active, since the control signal will be limited.

The fast gradient method is limited to the number of iterations, and therefore the maximum number of iterations was pre-calculated with the FIORDOS TOOLBOX in MATLAB. The maximum number of iteration depends on the absolute tolerance, $\varepsilon$, (6.34). That is a measurement how the value of the cost function deviates form the optimal value.

## 7.7 Simulation

The control loop has been simulated before the implementation on the tricopter has been accomplished. The simulation was made to ensure that the behavior of the controller was the desired one and to ensure that the output signal was tracking the reference signal. The parameters of the controller are the three weighting matrices in the optimization problem (6.9). By selecting these matrices, the desired behavior of reference tracking of the three rates and the response from control signals can be achieved. Since the time horizon was very short, $N = 2$, the end weight had a significant impact on the control error. By using the LQ-weight as the weight of end state in (6.3), the controller will have better performance with such short horizon. This weight was calculated by solving the algebraic Riccati equation with respect to $S$,

$$A^T S A^T - S - A^T S B (Q_2 + B^T S B)^{-1} B^T S A + C^T Q_1 C = 0, \tag{7.18}$$

and where the end term is given as

$$\begin{aligned} &(\mathbf{Y} - \mathbf{R})^T Q_{1N} (\mathbf{Y} - \mathbf{R}) \Leftrightarrow \\ &(\mathbf{X} - C^\dagger \mathbf{R})^T C^T Q_{1N} C (\mathbf{X} - C^\dagger \mathbf{R}) \Leftrightarrow \\ &(\mathbf{X} - C^\dagger \mathbf{R})^T S (\mathbf{X} - C^\dagger \mathbf{R}). \end{aligned} \tag{7.19}$$

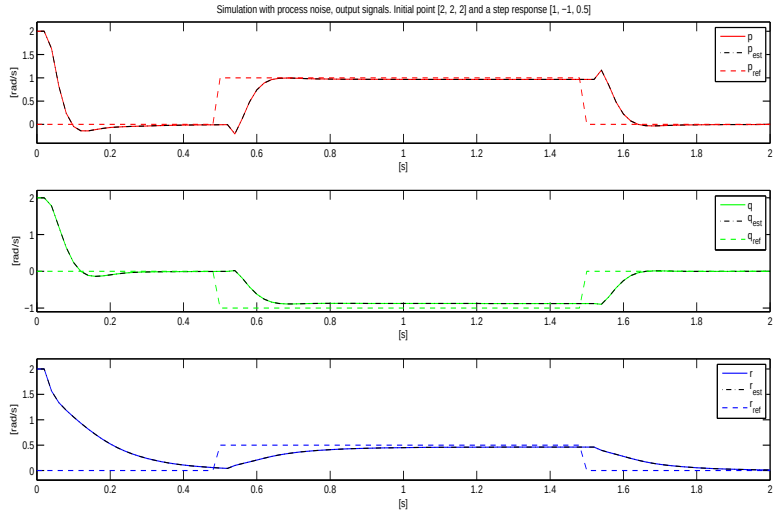
Instead of using the control error $(\mathbf{Y} - \mathbf{R}) = (C\mathbf{X} - \mathbf{R})$, the matrix $C$ was pseudo-inverted so that the reference signals are transformed to the same dimensions as the states. Since $C$ is not an invertible matrix, the pseudo inverse $C^\dagger$ was used. This weighting of the end state will give the MPC the same behavior as a LQ controller, as long as none of the constraints are active.

The simulation has been performed with an online solver, `quadprog`, in MATLAB. This is because the computation of the explicit MPC is time demanding, and the resulting controller has the same behavior as explicit MPC. The resulting controller was then used to compare the explicit MPC with the fast gradient method.

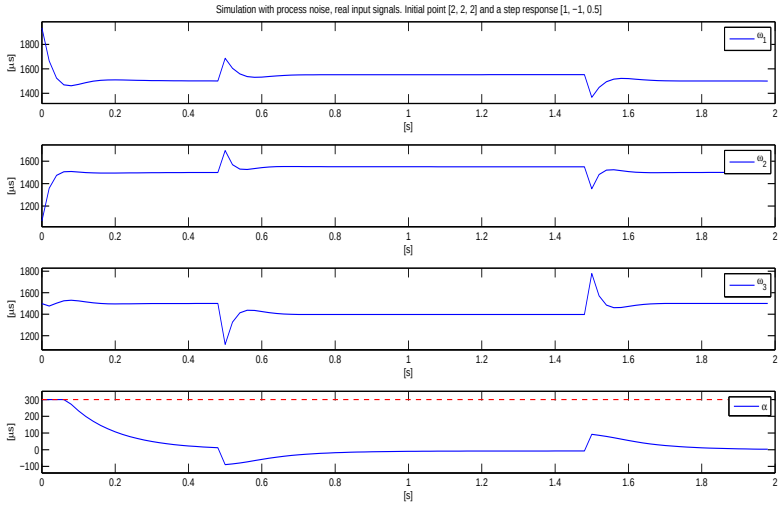
### 7.7.1 Simulation using black-box and black-box virtual model

The MPC controller based on the *black-box* and the *black-box virtual* model were tuned and simulated. This simulation had an initial rotational rate value of $[2, 2, 2]^T$ and after the output signals had stabilized, a step response was performed with a step of $[1, -1, 0.5]^T$. The simulation of the *black-box virtual* model gave good reference tracking and behavior, see Figure 7.16a, 7.16b for the output and the PWM signals, respectively.

The same simulation was performed with the *black-box* model, see Figures 7.17a, 7.17b, for the output and input signals, respectively. In the simulation, the tracking of the reference signal was good, but the input signals were not symmetrical. Only control signal no. 3 ($\omega_3$) was performing. Control signals no. 1 and 2, ($\omega_1$ and $\omega_2$), seem only to be balancing the tricopter. This behavior can be seen in the height of the peaks of the different input signals. Motors no. 1 and no. 2



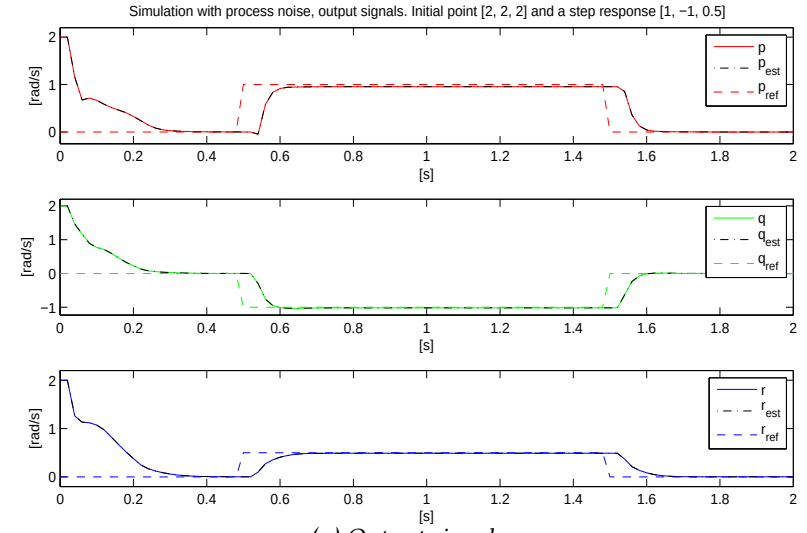

*(a) Output signals*



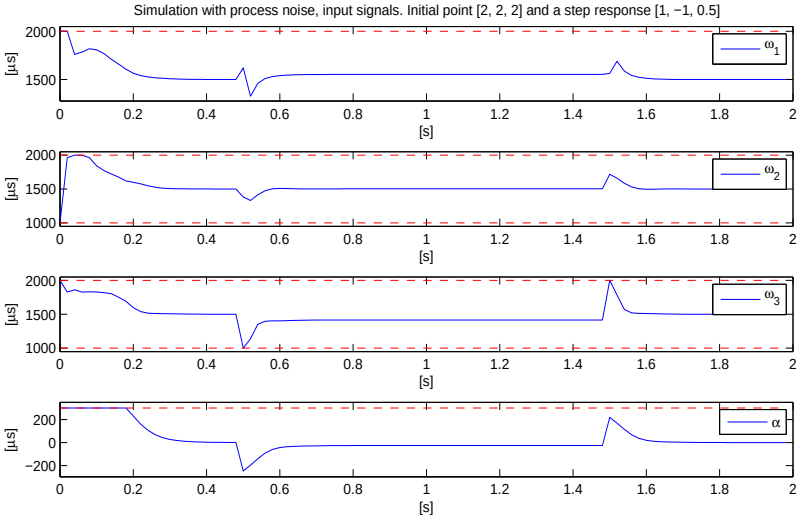

*(b) Input signals*

***Figure 7.16:*** *Simulation of control loop using the black-box virtual model. The PWM signals to the motors and the servo are presented in the case when the tricopter was released from a initial state and a step response was performed.*

had peaks of absolute magnitude ≈150, while motor no. 3 had a absolute peak of ≈400.




*(a) Output signals*




*(b) Input signals*

***Figure 7.17:*** *Simulation of control loop using the black-box model. The PWM signals to the motors and the servo are presented in the case when the tricopter was released from a initial state and a step response was performed.*

### 7.7.2 Comparing explicit MPC with fast gradient

Now when the controllers has been simulated it is of interest to compare the two algorithms that have been used in this thesis to implement the controller. The

***Table 7.2:*** *Computational time of the two solvers, and compare the solution to an pre-computed solution in* MATLAB. *Results are simulated with the black-box model.*

| **Quadprog** MATLAB | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | $J(\mathbf{U}_\varepsilon, \theta)$ | | | **U** | | **t** $[ms]$ |
| | | 4039.228 | 112.76 | -49.501 | 83.916 | 215.4 | - |
| **Explicit MPC** | | | | | | | |
| | | | | | **U** | | **t** $[ms]$ |
| | | - | 112.76 | -49.501 | 83.916 | 215.4 | $\approx 3$ |
| **Fast gradient** | | | | | | | |
| **iterations** | $\varepsilon$ | $J(\mathbf{U}_\varepsilon, \theta)$ | | | **U** | | **t** $[ms]$ |
| 12 (fixed) | - | 4076.011 | 85.895 | -71.974 | 90.938 | 218.49 | $\approx 15.5$ |
| 25 | $10^5$ | 4043.177 | 103.02 | -58.674 | 84.935 | 214.34 | $> 20$ |
| 56 | $10^4$ | 4039.235 | 112.36 | -49.879 | 83.961 | 215.36 | $> 20$ |
| 73 | $10^3$ | 4039.228 | 112.71 | -49.554 | 83.922 | 215.4 | $> 20$ |
| 91 | $10^2$ | 4039.228 | 112.76 | -49.507 | 83.917 | 215.4 | $> 20$ |

most interesting part, is to observe the computational time of the two solvers and the accuracy of the solution. Given a random state of the tricopter, the controller will try to compute the optimal solution to reach the reference level again. This initial state was introduced as $\mathbf{X} = [0.040119, -0.011655, -0.025267, 0, 0]^T$ which corresponds to rotational rates $\mathbf{Y} = [0.2, 0.3, 0.4]^T rad/s$. The reference level was set to $\mathbf{R} = [0, 0, 0, 1500]^T$ $\mu$s, and the 1500 corresponds to half-throttle.

The results are compared to one offline solution computed in MATLAB and are presented in Table 7.2. As the result shows in the Table, the explicit MPC can compute the solution very accurately compared to the solution in MATLAB. The explicit MPC is also very fast to compute the solution, compared to the fast gradient method. The solution computed by the fast gradient solver, depends on the maximum number of iterations, which is in turn determined by the absolute tolerance $\varepsilon$. If the numbers of iteration is fixed to 12, without using the pre-calculated iterations, the computational time of the solver is approximately 16 $[ms]$. This gives that the control loop, including Kalman filter, will be able to compute the control signal within the hard deadline. But the solution is not very accurate, compared to the optimal solution computed with `quadprog` in MATLAB. Since the PWM signals are integers, the amount of iterations that need to be performed to obtain accurate solution, is approximately 56. This gives that the fast gradient algorithm is too demanding for the microcontroller to compute the optimal solution.

It is interesting to simulate the system using the fast gradient method, with it-

erations less than 25, and compare the result with the offline solver `quadprog` and explicit MPC. These simulations are performed with the maximum number of iterations 2, 5, 12 and 25. There are also introduced process and measurement noise in the simulation, to make the simulation more realistic. In that way, it can be studied how the control loop acts on disturbances. These simulations can be seen in Figures 7.18, 7.19, 7.20, 7.21, respectively.

As can be seen in the simulations, the behavior of the output signals does not vary that much, even if the maximum number of iterations is 2. The explicit MPC will give identical solution, with some numerical differences, as the offline solver `quadprog`. Since the explicit MPC has very low computation time compared to the fast gradient, and the solution is optimal, this controller has been used to implement the controller on the tricopter.

The benefits of the explicit MPC is that it can compute the optimal signal in approximately 3 $[ms]$. This compared to the fast gradient which requires more than 20 $[ms]$ to compute the optimal control signal. The fast gradient method can compute the control signal within 20 $[ms]$ with number of iterations equal to 12 or less, but it will not be an optimal control signal. However, the effect on the output signal will be barely notable.

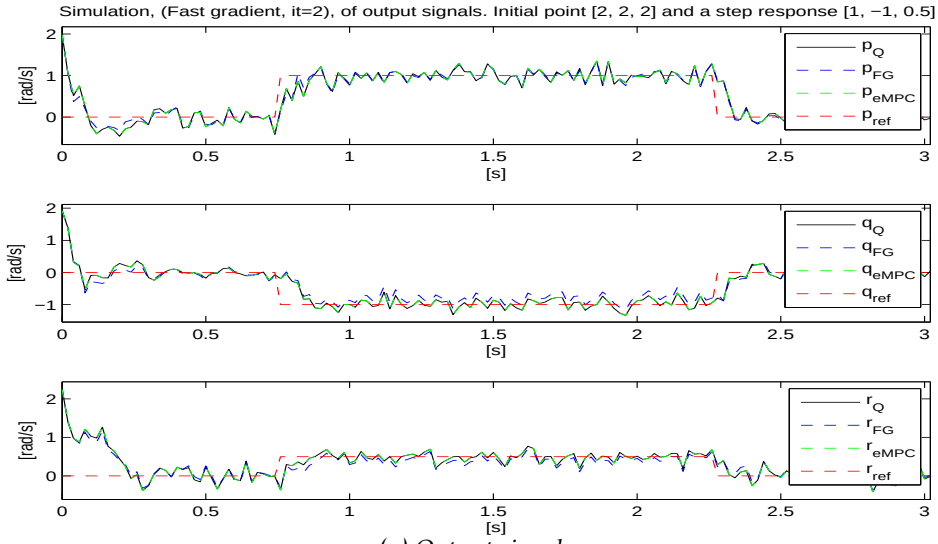
One benefit with the fast gradient is that the controller is lightweight compared to explicit MPC, in the sense of usage of memory space. The memory usage of the control loop using explicit MPC is approximately 35kB, compared to fast gradient that only requires approximately 5kB of memory usage.

## 7.8 Implementations

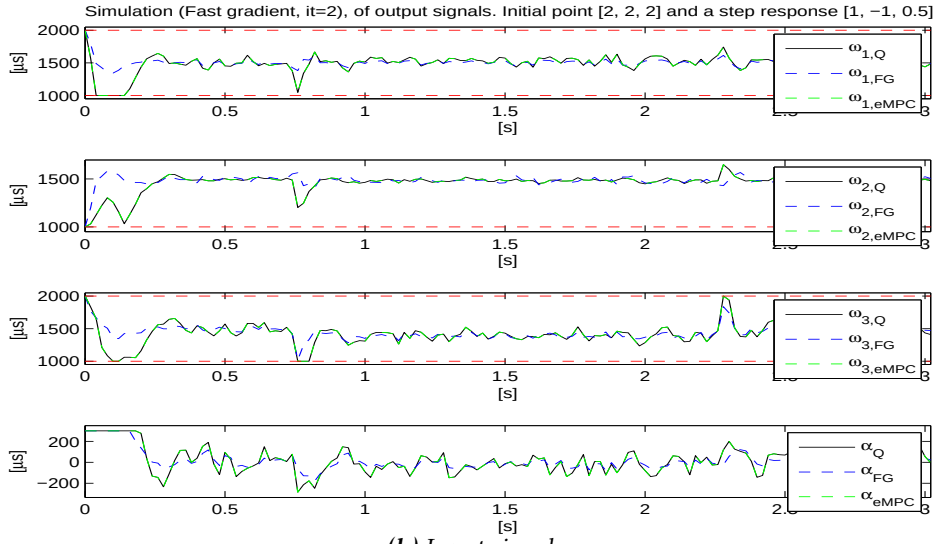
The *black-box* model had the problem that only one motor was performing, which also can be seen in the simulations, Figure 7.17b. The physical tricopter was not stabilized in a way that the simulation presented, which can be explained through that the model is not that accurate.

The *black-box virtual* model was performing much better than the *black-box* model. The stabilization of the pitch and yaw rate were good, but the stabilization of the roll rate was not aggressive enough. A flight experiment was performed using this controller. The stabilization was not that good to perform a flight more than lifting 5cm from the ground. The controller parameters could be tuned to obtain better stabilization, but unfortunately there was lack of time to do it in this thesis. One reason to that it was not possible to obtain the desired performance could be the non-minimum-phase zero that was present in this model. Similarly, the fast gradient algorithm was not tested to see how the stabilization works with low amount of iterations.

The result is, that a robust multivariable controller such as MPC, can be implemented on simple hardware as the Atmega2560, using the explicit MPC or fast gradient method. The control loop is also including a stationary Kalman filter,



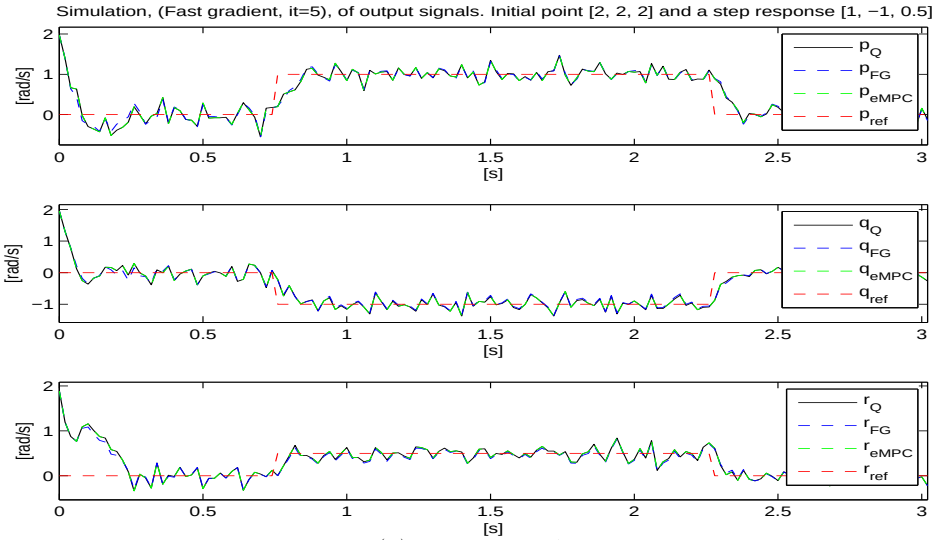

***(a)*** *Output signals*



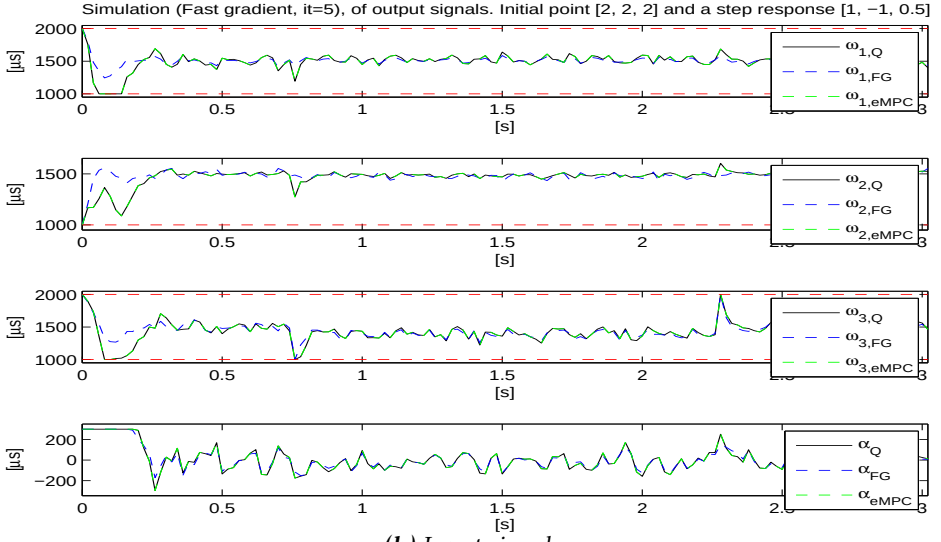

***(b)*** *Input signals*

***Figure 7.18:*** *Simulation using quadprog, explicit MPC and the fast gradient. The number of iterations used with the fast gradient method is 2.*

but a controller with a higher complexity will be too demanding for this micro-controller.

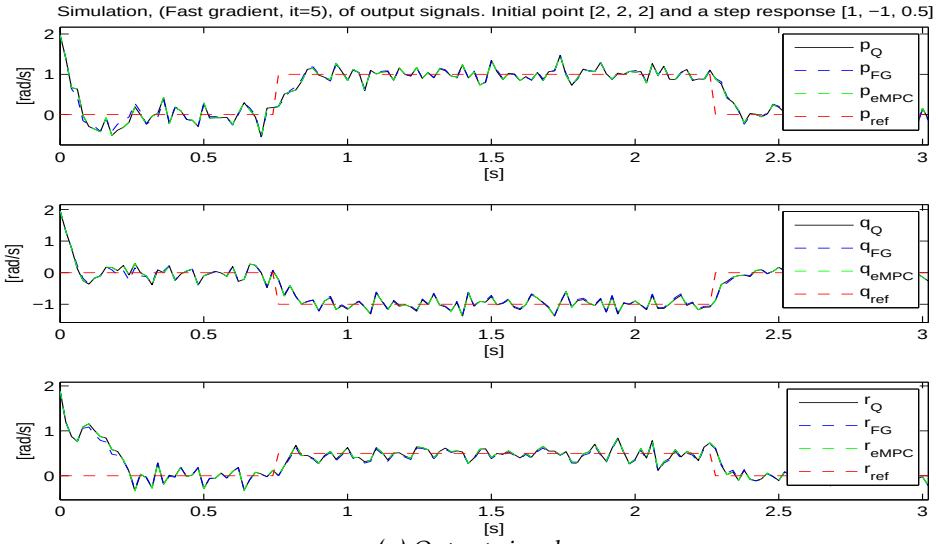



*(a) Output signals*

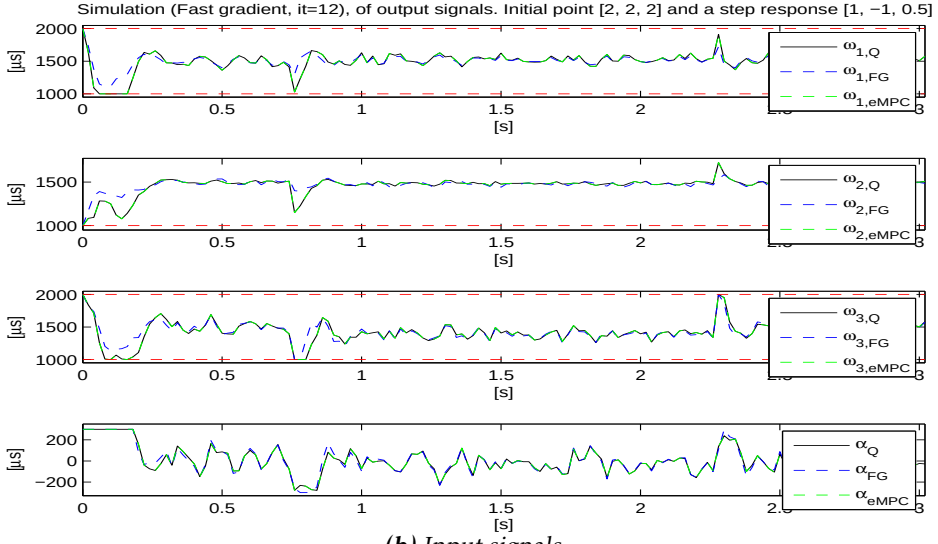



*(b) Input signals*

***Figure 7.19:*** *Simulation using quadprog, explicit MPC and fast gradient. The number of iterations, used with the fast gradient method, is 5.*

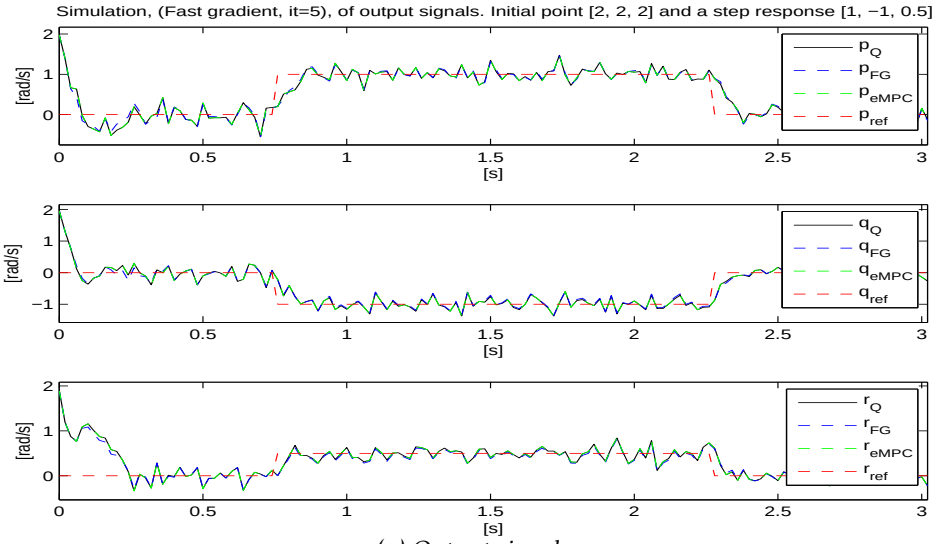



***(a)*** *Output signals*

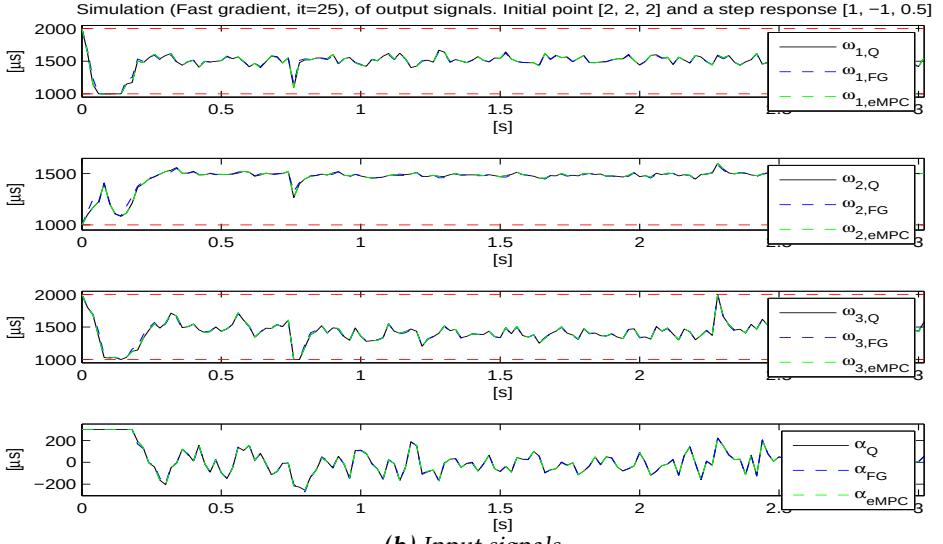



***(b)*** *Input signals*

***Figure 7.20:*** *Simulation using quadprog, explicit MPC and fast gradient. The number of iterations, used with the fast gradient method, is 12.*




*(a) Output signals*




*(b) Input signals*

***Figure 7.21:*** *Simulation using quadprog, explicit MPC and fast gradient. The number of iterations, used with the fast gradient method, is 25.*

# 8

# Conclusions and future work

Regarding to the results in previous chapter, conclusions and future work will be presented in this chapter.

## 8.1 Conclusions

The conclusions regarding the results are divided into three parts. One regarding the hardware, one regarding the system identification and one regarding the controller.

### 8.1.1 Hardware

The hardware that was used in this thesis was a platform, APM1, from ArduPilot, that in an upgraded version, with larger program memory, specially for ArduCopter. This uses a simple microcontroller, Atmega2560, that only has 16MHz clock frequency and does not handle floating point arithmetics. The internal memory consists of a 8kB RAM and 256kB program memory.

This limits the complexity of the controller. The desire was to implement a control loop that handles both the inner and outer loop, and in that way the autopilot will be able to control the tricopter autonomously. With the usage of the Atmega2560, the controller was only able to handle the rate control of the tricopter and the pilot will act as the outer loop.

The sample frequency of the data collection was 50Hz, which was probably to low to observe all the dynamics of the tricopter. This gives that a much more powerful hardware could have been used, so that the sample frequency of the control loop could have been much higher. By that the dynamics of the tricopter could have been observed more correctly.

A lesson learned from the thesis, is to get knowledge of the limitations of the hardware. Obvious offline solutions can be very hard, or even impossible, to implement on the proposed hardware. It is not so often the code has to be optimized so that the solution can be implemented on the hardware. One optimization can be the computational time of the algorithm, so that it fulfills the deadlines in real-time systems. But lack of memory is very rare today, when the storage capacity is several gigabytes, or at least several megabytes. The computational resources are also increasing with the new processors, such as ARM, which provides hardware support for floating point arithmetics and a clock frequency of several hundred of megahertz. With that, the complexity of the controller can be increased.

### 8.1.2 System identification

The system identification was performed on data that was sampled with 50Hz. Since the bandwidth of the tricopter was unknown it was hard to determine the sample frequency. There is a rule of thumb, that the sample frequency should be approximately ten times the bandwidth of the system. The choice of the sample frequency is acceptable, if the bandwidth of the tricopter is less than 5Hz. But since the velocities changes much faster than that, the system is most probably undersampled. The identification of the simplified *physical* model gives a hint of that. But since the hardware was limiting the sampling frequency, the built-in logging functions of the ArduCopter code were used and therefore the 50Hz sampling was used.

The Black box models were estimated with a order of 8, but were reduced to 5 to reduce the complexity. A correct model of the tricopter should at least include the delay in the motors and the servo, which result in a model of order 8 (three states from the rotational rates, one state from each motor and two states from the servo) . By combining the order of 8 and higher sampling frequency, the model fit could be increased.

There was an attempt to identify a diagonal model of order 3, that represent a decoupled model where the tricopter is seen as a low pass filter. But since the tricopter is a highly cross-coupled system, this attempt was not successful. Therefore a black-box model using the virtual signals were used to represent a decoupled system.

### 8.1.3 Controller

The purpose with this thesis was to implement a robust controller on a simple hardware as the Atmega2560. This goal was achieved, both with the fast gradient method and explicit MPC. Both algorithms were able to compute the control signal within the hard deadline of the control loop, after the complexity of the problem was reduced. One simplification made to the controller was the introduction of move blocking, where the control signal will be constant through the entire prediction horizon, with control signal horizon $N_s = 1$, and a prediction horizon $N = 2$. The desired prediction horizon should be enough to cover all the transients within the horizon. Since the prediction horizon was equal to 2, all the

transients were possible not to be covered within this horizon.

The absolute tolerance of the fast gradient algorithm was selected such that the algorithm could compute the control signal within 20 [*ms*] (including the Kalman filter). By increasing the tolerance the number of iterations were decreased, but instead the solution differed from the optimal solution. The simulation that compared explicit MPC and the fast gradient method with an optimal offline solution, gave that the effect on the output signals were barely noticeable even if the input signals were not the optimal.

## 8.2 Future work

The control loop that was implemented on the hardware, was only the inner loop. That was because the hardware was not able to handle a more complex model than the Black box model of order 5. By changing the hardware to one with more computational resources, the model could be done more complex, and by that also include the outer loop in the control loop.

This thesis can be seen as a basis for MPC controlling a tricopter, or even for just modeling the rotorcraft. All the differential equations that describe the dynamics of the tricopter, are determined by the laws of physics and presented as nonlinear equations in this thesis. If a hardware with more computational resources are used, the model does not have to be linearized. By that, more advanced flight trajectories could be performed with the rotorcraft.

The controller could be a nonlinear MPC, using e.g. the fast gradient algorithm, if the hardware can represent floating data points and therefore the solver will be much faster. The limitation of future work is just bounded by the imagination. The tricopter could be used for different operations. For example, if a camera is attached to the tricopter, it could be used for surveillance or observing environments difficult for the man to visit.

# Bibliography


Atmel Inc. http://www.atmel.com/devices/atmega2560.aspx, Accessed March 2012. Cited on page 6.

AVR-GCC. http://www.nongnu.org/avr-libc/, Accessed January 2012. Cited on page 55.

Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N. Pistikopoulos. The explicit linear quadratic regulator for constrained systems. *Automatica*, 38:3–20, 2002. Cited on pages 2, 37, and 43.

G. Cai, B.M. Chen, K. Peng, M. Dong, and T.H. Lee. Modeling and control system design for a uav helicopter. In *Control and Automation, 2006. MED '06. 14th Mediterranean Conference on*, pages 1 –6, june 2006. Cited on page 2.

P. Castillo, A. Dzul, and R. Lozano. Real-time stabilization and tracking of a four-rotor mini rotorcraft. *IEEE Transactions on Control Systems Technology*, 12(4): 510 – 516, july 2004. Cited on page 2.

DIY Drones. http://code.google.com/p/ardupilot-mega, Accessed January 2012a. Cited on page 6.

DIY Drones. http://code.google.com/p/arducopter, Accessed January 2012b. Cited on page 6.

FiOrdOs. http://fiordos.ethz.ch/dokuwiki/doku.php, Accessed April 2012. Cited on page 66.

T. Glad and L. Ljung. *Reglertori - Flervariabla och olinjära metoder*. Studentlitteratur AB, Lund, 2nd edition, 2003. Cited on pages 17, 22, and 62.

T. Glad and L. Ljung. *Modellbygge och Simulering*. Studentlitteratur AB, Lund, 2nd edition, 2004. Cited on pages 16 and 27.

F. Gustafsson. *Statistical Sensor Fusion*. Studentlitteratur AB, Lund, 1st edition, 2012. Cited on pages 20 and 33.

F. Gustafsson, L. Ljung, and M. Millnert. *Signal Processing*. Studentlitteratur AB, Lund, 1st edition, 2010. Cited on pages 33 and 36.

A. Hansson. *Model Predictive Control*, October 2010. Cited on page 44.

H.J. Kim, D.H. Shim, and S. Sastry. Nonlinear model predictive tracking control for rotorcraft-based unmanned aerial vehicles. In *American Control Conference, 2002. Proceedings of the 2002*, volume 5, pages 3576 – 3581 vol.5, 2002. Cited on page 2.

J. Kiusaalas and A. Pytel. *Engineering Mechanics Dynamics*. Thomson learning, 2nd edition, 2001. Cited on pages 15 and 19.

J. M. Maciejowski. *Predictive Control with Constraints*. Prentice Hall, 2001. Cited on pages 2 and 37.

V. Mistler, A. Benallegue, and N.K. M'Sirdi. Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In *Robot and Human Interactive Communication, 2001. Proceedings. 10th IEEE International Workshop on*, pages 586 –593, 2001. Cited on pages 2 and 20.

S. Richter, C. N. Neil, and M. Morari. Computational complexity certification for real-time MPC with input constraints based on the fast gradient method. *IEEE Transactions on Automatic Control*, PP(99):1, November 2011. Cited on pages 2, 37, and 45.

S. Salazar-Cruz, F. Kendoul, R. Lozano, and I. Fantoni. Real-time stabilization of a small three-rotor aircraft. *Aerospace and Electronic Systems, IEEE Transactions on*, 44(2):783 –794, april 2008. Cited on page 2.

Sune Söderquist. *Från insignal till utsignal*. Tryckeriet Erik Larsson AB, 1st ed. edition, 2007. Cited on page 33.

Dong-Wan Yoo, Hyon-Dong Oh, Dae-Yeon Won, and Min-Jea Tahk. Dynamic modeling and control system design for tri-rotor uav. In *Systems and Control in Aeronautics and Astronautics (ISSCAA), 2010 3rd International Symposium on*, pages 762 –767, june 2010. Cited on pages 2 and 59.

# A

# Kinematic equation

The derivation of the kinematic equations is done by applying the rotation matrix to each unit vector and then the result is differentiated

$$\begin{pmatrix} p \\ q \\ r \end{pmatrix} = \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + Q_{\phi}^{x} \begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + Q_{\phi}^{x} Q_{\theta}^{y} \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix}. \tag{A.1}$$

By solving the angle derivatives, the equation can be rewritten as

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin(\phi)\tan(\theta) & \cos(\phi)\tan(\theta) \\ 0 & \cos(\phi) & -\sin(\phi) \\ 0 & \frac{\sin(\phi)}{\cos(\theta)} & \frac{\cos(\phi)}{\cos(\theta)} \end{pmatrix} \begin{pmatrix} p \\ q \\ r \end{pmatrix}. \tag{A.2}$$

# B

# Linearization

$$a_{4,8} = \frac{\partial \dot{u}}{\partial \theta} = -\frac{1}{m}\Big(F^{\mathfrak{B}}_{X,0}\mathrm{S}_{\theta_0}\mathrm{C}_{\psi_0} + F^{\mathfrak{B}}_{Y,0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} + F^{\mathfrak{B}}_{Z,0}\mathrm{C}_{\theta_0}\Big)$$

$$a_{4,9} = \frac{\partial \dot{u}}{\partial \psi} = -\frac{1}{m}\Big(F^{\mathfrak{B}}_{X,0}\mathrm{C}_{\theta_0}\mathrm{S}_{\psi_0} + F^{\mathfrak{B}}_{Y,0}\mathrm{C}_{\theta_0}\mathrm{C}_{\psi_0}\Big)$$

$$a_{5,7} = \frac{\partial \dot{v}}{\partial \phi} = -\frac{1}{m}\Big(F^{\mathfrak{B}}_{X,0}(\mathrm{C}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{C}_{\psi_0} + \mathrm{S}_{\phi_0}\mathrm{S}_{\psi_0}) + F^{\mathfrak{B}}_{Y,0}(\mathrm{C}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} - \mathrm{S}_{\phi_0}\mathrm{C}_{\psi_0}) + F^{\mathfrak{B}}_{Z,0}\mathrm{C}_{\phi_0}\mathrm{C}_{\theta_0}\Big)$$

$$a_{5,8} = \frac{\partial \dot{v}}{\partial \theta} = \frac{1}{m}\Big(F^{\mathfrak{B}}_{X,0}\mathrm{S}_{\phi_0}\mathrm{C}_{\theta_0}\mathrm{C}_{\psi_0} + F^{\mathfrak{B}}_{Y,0}\mathrm{S}_{\phi_0}\mathrm{C}_{\theta_0}\mathrm{S}_{\psi_0} - F^{\mathfrak{B}}_{Z,0}\mathrm{S}_{\phi_0}\mathrm{S}_{\theta_0}\Big)$$

$$a_{5,9} = \frac{\partial \dot{v}}{\partial \psi} = \frac{1}{m}\Big(-F^{\mathfrak{B}}_{X,0}(\mathrm{S}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} + \mathrm{C}_{\phi_0}\mathrm{C}_{\psi_0}) + F^{\mathfrak{B}}_{Y,0}(\mathrm{S}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{C}_{\psi_0} - \mathrm{C}_{\phi_0}\mathrm{S}_{\psi_0})\Big)$$

$$a_{6,7} = \frac{\partial \dot{w}}{\partial \phi} = -\frac{1}{m}\Big(F^{\mathfrak{B}}_{X,0}(\mathrm{S}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{C}_{\psi_0} + \mathrm{C}_{\phi_0}\mathrm{S}_{\psi_0}) + F^{\mathfrak{B}}_{Y,0}(\mathrm{S}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} + \mathrm{C}_{\phi_0}\mathrm{C}_{\psi_0}) + F^{\mathfrak{B}}_{Z,0}\mathrm{S}_{\phi_0}\mathrm{C}_{\theta_0}\Big)$$

$$a_{6,8} = \frac{\partial \dot{w}}{\partial \theta} = \frac{1}{m}\Big(F^{\mathfrak{B}}_{X,0}\mathrm{C}_{\phi_0}\mathrm{C}_{\theta_0}\mathrm{C}_{\psi_0} + F^{\mathfrak{B}}_{Y,0}\mathrm{C}_{\phi_0}\mathrm{C}_{\theta_0}\mathrm{S}_{\psi_0} - F^{\mathfrak{B}}_{Z,0}\mathrm{C}_{\phi_0}\mathrm{S}_{\theta_0}\Big)$$

$$a_{6,9} = \frac{\partial \dot{v}}{\partial \psi} = \frac{1}{m}\Big(-F^{\mathfrak{B}}_{X,0}(\mathrm{C}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} + \mathrm{S}_{\phi_0}\mathrm{C}_{\psi_0}) + F^{\mathfrak{B}}_{Y,0}(\mathrm{C}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{C}_{\psi_0} + \mathrm{S}_{\phi_0}\mathrm{S}_{\psi_0})\Big)$$

$$a_{7,7} = \frac{\partial \dot{\phi}}{\partial \phi} = q_0 \mathrm{C}_{\phi_0} \mathrm{T}_{\theta_0} - r_0 \mathrm{S}_{\phi_0} \mathrm{T}_{\theta_0}, \qquad a_{7,8} = \frac{\partial \dot{\phi}}{\partial \theta} = -g \mathrm{S}_{\phi_0} \mathrm{S}_{\theta_0}$$

$$a_{7,10} = \frac{\partial \dot{\phi}}{\partial p} = 1, \qquad a_{7,11} = \frac{\partial \dot{\phi}}{\partial q} = \mathrm{S}_{\phi_0} \mathrm{T}_{\theta_0}, \qquad a_{7,12} = \mathrm{C}_{\phi_0} \mathrm{T}_{\theta_0}$$

$$a_{8,7} = \frac{\partial \dot{\theta}}{\partial \phi} = q_0 \mathrm{S}_{\phi_0} - r_0 \mathrm{C}_{\phi_0}, \qquad a_{8,11} = \frac{\partial \dot{\theta}}{\partial q} = \mathrm{C}_{\phi_0}, \qquad a_{8,12} = \frac{\partial \dot{\theta}}{\partial r} = -\mathrm{S}_{\phi_0}$$

$$a_{9,7} = \frac{\partial \dot{\psi}}{\partial \phi} = q_0 \frac{\mathrm{C}_{\phi_0}}{\mathrm{C}_{\theta_0}} - r_0 \frac{\mathrm{S}_{\phi_0}}{\mathrm{C}_{\theta_0}}, \qquad a_{9,8} = \frac{\partial \dot{\psi}}{\partial \theta} = (q_0 \mathrm{S}_{\phi_0} + r_0 \mathrm{C}_{\phi_0}) \frac{\mathrm{T}_{\theta_0}}{\mathrm{C}_{\theta_0}}$$

$$a_{9,11} = \frac{\partial \dot{\psi}}{\partial q} = \frac{\mathrm{S}_{\phi_0}}{\mathrm{C}_{\theta_0}}, \qquad a_{9,12} = \frac{\partial \dot{\psi}}{\partial r} = \frac{\cos \phi_0}{\cos \theta_0}$$

$$a_{10,11} = \frac{\partial \dot{p}}{\partial q} = \frac{I_{yy} - I_{zz}}{I_{xx}} r_0, \quad a_{10,12} = \frac{\partial \dot{p}}{\partial r} = \frac{I_{yy} - I_{zz}}{I_{xx}} q_0$$

$$a_{11,10} = \frac{\partial \dot{q}}{\partial p} = \frac{I_{zz} - I_{xx}}{I_{yy}} r_0, \quad a_{11,12} = \frac{\partial \dot{q}}{\partial r} = \frac{I_{zz} - I_{xx}}{I_{yy}} p_0$$

$$a_{12,10} = \frac{\partial \dot{r}}{\partial p} = \frac{I_{xx} - I_{yy}}{I_{xx}} q_0, \quad a_{12,11} = \frac{\partial \dot{r}}{\partial q} = \frac{I_{xx} - I_{yy}}{I_{zz}} p_0$$

$$b_{4,1} = \frac{\partial \dot{u}}{\partial \omega_1} = \frac{1}{m} 2 k_t \omega_{1,0} (\mathrm{S}_{\alpha_0} \mathrm{C}_{\theta_0} \mathrm{S}_{\psi_0} + \mathrm{C}_{\alpha_0} \mathrm{S}_{\theta_0})$$

$$b_{4,2} = \frac{\partial \dot{u}}{\partial \omega_2} = \frac{1}{m} (2 k_t \omega_{2,0} \mathrm{S}_{\theta_0})$$

$$b_{4,3} = \frac{\partial \dot{u}}{\partial \omega_3} = \frac{1}{m} (2 k_t \omega_{3,0} \mathrm{S}_{\theta_0})$$

$$b_{4,4} = \frac{\partial \dot{u}}{\partial \alpha} = \frac{1}{m} k_t \omega_{1,0} |\omega_{1,0}| (\mathrm{C}_{\alpha_0} \mathrm{C}_{\theta_0} \mathrm{S}_{\psi_0} - \mathrm{S}_{\alpha_0} \mathrm{S}_{\theta_0})$$

$$b_{5,1} = \frac{\partial \dot{v}}{\partial \omega_1} = \frac{1}{m} 2 k_t \omega_{1,0} (\mathrm{S}_{\alpha_0} (\mathrm{S}_{\phi_0} \mathrm{S}_{\theta_0} \mathrm{S}_{\psi_0} - \mathrm{C}_{\theta_0} \mathrm{C}_{\psi_0}) - \mathrm{C}_{\alpha_0} \mathrm{S}_{\phi_0} \mathrm{C}_{\theta_0})$$

$$b_{5,2} = \frac{\partial \dot{v}}{\partial \omega_2} = \frac{-1}{m} (2 k_t \omega_{2,0} \mathrm{S}_{\phi_0} \mathrm{C}_{\theta_0})$$

$$b_{5,3} = \frac{\partial \dot{v}}{\partial \omega_3} = \frac{-1}{m}(2k_t\omega_{3,0}\mathrm{S}_{\phi_0}\mathrm{C}_{\theta_0})$$

$$b_{5,4} = \frac{\partial \dot{v}}{\partial \alpha} = \frac{1}{m}k_t\omega_{1,0}|\omega_{1,0}|(\mathrm{C}_{\alpha_0}(\mathrm{S}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} + \mathrm{C}_{\theta_0}\mathrm{C}_{\psi_0}) + \mathrm{S}_{\alpha_0}\mathrm{S}_{\phi_0}\mathrm{C}_{\theta_0})$$

$$b_{6,1} = \frac{\partial \dot{w}}{\partial \omega_1} = \frac{1}{m}2k_t\omega_{1,0}(\mathrm{S}_{\alpha_0}(\mathrm{C}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} - \mathrm{S}_{\phi_0}\mathrm{C}_{\psi_0}) - \mathrm{C}_{\alpha_0}\mathrm{C}_{\phi_0}\mathrm{C}_{\theta_0})$$

$$b_{6,2} = \frac{\partial \dot{w}}{\partial \omega_2} = \frac{1}{m}2k_t\omega_{2,0}\mathrm{C}_{\phi_0}\mathrm{C}_{\theta_0}$$

$$b_{6,3} = \frac{\partial \dot{w}}{\partial \omega_3} = \frac{1}{m}2k_t\omega_{3,0}\mathrm{C}_{\phi_0}\mathrm{C}_{\theta_0}$$

$$b_{6,4} = \frac{\partial \dot{w}}{\partial \alpha} = \frac{1}{m}2k_t\omega_{1,0}|\omega_{1,0}|(\mathrm{C}_{\alpha_0}(\mathrm{C}_{\phi_0}\mathrm{S}_{\theta_0}\mathrm{S}_{\psi_0} + \mathrm{S}_{\phi_0}\mathrm{C}_{\psi_0}) + \mathrm{S}_{\alpha_0}\mathrm{C}_{\phi_0}\mathrm{C}_{\theta_0})$$

$$b_{10,2} = \frac{\partial \dot{p}}{\partial \omega_2} = \frac{1}{I_{xx}}\sqrt{3}lk_t\omega_{2,0}$$

$$b_{10,3} = \frac{\partial \dot{p}}{\partial \omega_3} = \frac{-1}{I_{xx}}\sqrt{3}lk_t\omega_{3,0}$$

$$b_{11,1} = \frac{\partial \dot{q}}{\partial \omega_1} = \frac{1}{I_{yy}}2\omega_{1,0}(k_\tau\mathrm{S}_{\alpha_0} - lk_t\mathrm{C}_{\alpha_0})$$

$$b_{11,2} = \frac{\partial \dot{q}}{\partial \omega_2} = \frac{1}{I_{yy}}lk_t\omega_{2,0}$$

$$b_{11,3} = \frac{\partial \dot{q}}{\partial \omega_3} = \frac{1}{I_{yy}}lk_t\omega_{3,0}$$

$$b_{11,4} = \frac{\partial \dot{q}}{\partial \alpha} = \frac{1}{I_{yy}}\omega_{1,0}|\omega_{1,0}|(kk_t\mathrm{S}_{\alpha_0} + k_\tau\mathrm{C}_{\alpha_0})$$

$$b_{12,1} = \frac{\partial \dot{r}}{\partial \omega_1} = \frac{-1}{I_{zz}}2\omega_{1,0}(lk_t\mathrm{S}_{\alpha_0} + k_\tau\mathrm{C}_{\alpha_0})$$

$$b_{12,2} = \frac{\partial \dot{r}}{\partial \omega_2} = \frac{-1}{I_{zz}}2k_\tau\omega_{2,0}$$

$$b_{12,3} = \frac{\partial \dot{r}}{\partial \omega_3} = \frac{-1}{I_{zz}}2k_\tau\omega_{3,0}$$

$$b_{12,4} = \frac{\partial \dot{r}}{\partial \alpha} = \frac{1}{I_{zz}} \omega_{1,0} |\omega_{1,0}| (k_\tau \mathrm{S}_{\alpha_0} - l k_t \mathrm{C}_{\alpha_0})$$

$$c_{1,1} = \frac{\partial h_1}{\partial x} = \frac{180}{\pi R_{Earth}}$$

$$c_{2,1} = \frac{\partial h_2}{\partial x} = \frac{-180 y_0}{\pi R_{Earth}^2} \frac{\sin(\frac{x_o}{R_{Earth}})}{\cos^2(\frac{x_0}{R_{Earth}})}$$

$$c_{2,2} = \frac{\partial h_2}{\partial y} = \frac{180}{\pi R_{Earth} \cos(\frac{x_0}{R_{Earth}})}$$

$$c_{4,4} = \frac{\partial h_4}{\partial u} = \frac{u_0}{\sqrt{u_0^2 + v_0^2 + w_0^2}}$$

$$c_{4,5} = \frac{\partial h_4}{\partial v} = \frac{v_0}{\sqrt{u_0^2 + v_0^2 + w_0^2}}$$

$$c_{4,6} = \frac{\partial h_4}{\partial w} = \frac{w_0}{\sqrt{u_0^2 + v_0^2 + w_0^2}}$$

Where S, C, T denotes sin, cos, tan, respectively.

# C

# Linearized physical model

The continuous time *physical* model that describes the rotational rates of the tricopter 3.19. The model includes the model of the electrical motors and the servo. The model is given by

$$A(\theta) = \begin{pmatrix} -10^{-9} & 0 & 0 & 0 & a_{1,5} & a_{1,6} & 0 & 0 \\ 0 & -10^{-9} & 0 & a_{2,4} & a_{2,5} & a_{2,6} & a_{2,7} & 0 \\ 0 & 0 & -10^{-9} & a_{3,4} & a_{3,5} & a_{3,6} & a_{3,7} & 0 \\ 0 & 0 & 0 & a_{4,4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & a_{5,5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & a_{6,6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & a_{8,8} \end{pmatrix}, \tag{C.1}$$

$$B(\theta) = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ b_{4,1} & 0 & 0 & 0 \\ 0 & b_{5,2} & 0 & 0 \\ 0 & 0 & b_{6,3} & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & b_{8,4} \end{pmatrix}, \tag{C.2}$$

$$C(\theta) = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{C.3}$$

$$\mathbf{Y}(t) = [p(t), q(t), r(t)]^T \quad \mathbf{U}(t) = [\mathbf{U}_1, \mathbf{U}_2, \mathbf{U}_3, \mathbf{U}_s]^T, \tag{C.4}$$

where the output signal $\mathbf{Y}$ are the rotational rates of the tricopter. The input signal $\mathbf{U}$ are the PWM signals to each motor, $\mathbf{U}_i$, $i = 1, 2, 3$, and the servo, $\mathbf{U}_s$. The matrix elements $a_{i,j}$ and $b_{i,j}$ are given by

$$a_{1,5} = \frac{\partial \dot{p}}{\partial \omega_2} = \frac{1}{I_{xx}} \sqrt{3} l k_t \omega_{2,0}$$

$$a_{1,6} = \frac{\partial \dot{p}}{\partial \omega_3} = \frac{-1}{I_{xx}} \sqrt{3} l k_t \omega_{3,0}$$

$$a_{2,4} = \frac{\partial \dot{q}}{\partial \omega_1} = \frac{1}{I_{yy}} 2\omega_{1,0} (k_\tau \mathrm{S}_{\alpha_0} - l k_t \mathrm{C}_{\alpha_0})$$

$$a_{2,5} = \frac{\partial \dot{q}}{\partial \omega_2} = \frac{1}{I_{yy}} l k_t \omega_{2,0}$$

$$a_{2,6} = \frac{\partial \dot{q}}{\partial \omega_3} = \frac{1}{I_{yy}} l k_t \omega_{3,0}$$

$$a_{2,7} = \frac{\partial \dot{q}}{\partial \alpha} = \frac{1}{I_{yy}} \omega_{1,0} |\omega_{1,0}| (k k_t \mathrm{S}_{\alpha_0} + k_\tau \mathrm{C}_{\alpha_0})$$

$$a_{3,4} = \frac{\partial \dot{r}}{\partial \omega_1} = \frac{-1}{I_{zz}} 2\omega_{1,0} (l k_t \mathrm{S}_{\alpha_0} + k_\tau \mathrm{C}_{\alpha_0})$$

$$a_{3,5} = \frac{\partial \dot{r}}{\partial \omega_2} = \frac{-1}{I_{zz}} 2 k_\tau \omega_{2,0}$$

$$a_{3,6} = \frac{\partial \dot{r}}{\partial \omega_3} = \frac{-1}{I_{zz}} 2 k_\tau \omega_{3,0}$$

$$a_{3,7} = \frac{\partial \dot{r}}{\partial \alpha} = \frac{1}{I_{zz}} \omega_{1,0} |\omega_{1,0}| (k_\tau \mathrm{S}_{\alpha_0} - l k_t \mathrm{C}_{\alpha_0})$$

$$a_{i,i} < 0, \; i = 5, 6, 8, \tag{C.5}$$

and $b_{4,1}$, $b_{5,2}$, $b_{6,3}$ and $b_{8,4}$ are constants.