DEGREE PROJECT, IN OPTIMIZATION AND SYSTEMS THEORY , SECOND LEVEL

STOCKHOLM, *SWEDEN 2015*

Design, Modeling and Control of an Octocopter

OSCAR OSCARSON

KTH ROYAL INSTITUTE OF TECHNOLOGY

SCI SCHOOL OF ENGINEERING SCIENCES

Design, Modeling and Control of an Octocopter

O S C A R O S C A R S O N

Master's Thesis in Optimization and Systems Theory (30 ECTS credits)
Master Programme in Applied and Computational Mathematics
(120 credits)
Royal Institute of Technology year 2015
Supervisor at Saab: Stefan Thorstenson
Supervisor at KTH: Xiaoming Hu
Examiner: Xiaoming Hu

TRITA-MAT-E 2015:63
ISRN-KTH/MAT/E--15/63--SE

Royal Institute of Technology
SCI School of Engineering Sciences

KTH SCI
SE-100 44 Stockholm, Sweden

URL: www.kth.se/sci

Abstract

This master's thesis project from the Department of Mathematics at KTH, Royal Institute of Technology, Stockholm, Sweden was carried out at Saab Dynamics, Linköping, Sweden. In this thesis, an octocopter was studied, which is a multirotor vehicle, a rotorcraft with more than two rotors. Multirotors have recently become very popular and have various interesting applications needing further research. Since the market for powerful credit-card-sized computers is continuously increasing, the development and research of multirotors is simplified.

The main purpose of this thesis project was to develop a complete open-sourced octocopter including control laws which should be used in future thesis projects at Saab Dynamics. Since the authors field is within Applied Mathematics, this thesis report has focused on the theoretical and analytical part of the project rather than the development process. Nevertheless, the report includes and gives a detailed overview of the complete octocopter and its final configuration. In addition, the report features system identifications which were carried out in order to estimate important properties of the vehicle.

In order to carry out mathematical analyses and propose control laws, a mathematical model of the vehicle was required. Since the vehicle moves in 6DoF (six-degrees of freedom), a suitable coordinate system handling these freedoms was needed. In this thesis, Euler angles and quaternions were used for representing attitude. The complete nonlinear model was derived using Newton's second law of motion in a rotating reference frame. Additional effects such as aerodynamics, precession torques and motor dynamics were analyzed and modeled.

The main content of this report which can be divided into two part deals with the control of the octocopter. The first part investigates approaches for converting of the model control inputs, forces and torques, to corresponding motor commands, angular-rates. This issue is seldom covered in research articles proposing control laws for multirotors and requires special attention when developing multirotors. By minimizing the L^2-norm, deriving control boundaries and using a priority algorithm which handles situations where the control demanded is greater than the momentary available control input, it was shown that the conversion between these properties was possible, even in critical situations.

The second part proposes control laws and approaches for controlling the vehicle in 6DoF. Three different control strategies have been proposed: pilot-based, attitude and position control. The pilot-based control is intended to be used by a pilot, controlling the in aviation standard roll, pitch and yaw-rate. The control method used is a full state feedback controller using a reduced observer, observing the motor dynamics. The attitude controller is an alteration of the pilot-based controller, controlling roll, pitch and yaw. Lastly, a position controller is derived, controlling the translational position of the vehicle, allowing for autonomous flying. The position controller uses a nonlinear Lyapunov based controller where the control inputs are converted to desired attitude reference inputs, send to the attitude controller.

The control laws were evaluated using a Simulink model where the complete nonlinear system was implemented. All derived control laws showed promising results and were able to accomplish desired behavior. The model featured a visualization of the vehicle in 6DoF running in real time and enabled for the use of a pc gaming controller, allowing for training and testing of e.g., the pilot-based controller.

At the end, real flying data is presented and analyzed using the pilot-based controller. The report is finished of with a discussion, covering previously chapters of the thesis, proposing future interesting research and work.

Sammanfattning

Detta examensarbete från Institutionen för Matematik vid KTH, Kungliga Tekniska Högskolan, Stockholm, Sverige har genomförts vid Saab Dynamics, Linköping, Sverige. Under examensarbetet har en oktokopter studerades, vilket är en multirotor dvs. en helikopter med fler än två rotorer. Multirotorer har under den senaste tiden blivit allt populärare och har flera intressanta tillämpningsområden som kräver vidare forskning. Då marknaden för kraftfulla datorer i kreditkortsformat kontinuerligt ökar, förenklas utvecklingen och forskningen på multirotorer.

Huvudsyftet med examensarbetet var att utveckla en komplett användarvänlig oktokopter inklusive styrlagar vars syfte var att användas i framtida examensarbeten på Saab Dynamics. Då författarens huvudinriktning är inom tillämpad matematik, har rapportens fokus legat på den teoretiska och analytiska delen av projektet snarare än utvecklingsprocessen. Rapporten ger även en detaljerad översikt över den utvecklade plattformen samt dess slutliga konfiguration. Utöver den teoretiska delen innehåller rapporten även metoder för systemidentifiering som har utformats och genomförts för att kunna uppskatta viktiga storheter av farkosten.

För att kunna utföra matematiska analyser och ta fram styrlagar, krävdes en matematisk modell av fordonet. Eftersom fordonet rör sig i 6DoF (sex frihetsgrader) krävdes ett lämpligt koordinatsystem. I detta projekt användes Euler vinklar och quaternioner för representation av attityd. Den fullständiga olinjära modellen härleddes med användning av Newtons andra lag i ett roterande koordinatsystem. Ytterligare effekter såsom aerodynamik, precession och motordynamik har analyserats och modellerats.

Det huvudsakliga innehållet i denna rapport som består av två delar, behandlar styrningen av oktokoptern. Den första delen undersöker metoder för att omvandla modellens styrsignaler, krafter och moment till motsvarande motorkommandon, vinkelhastigheter. Denna frågeställning behandlas sällan i forskningsartiklar som föreslår styrlagar för multirotorer och kräver särskild uppmärksamhet vid utveckling av multirotorer. Genom att minimera L^2-normen, härleda gränser för styrsignalen och genom att använda en prioriteringsalgoritm som hanterar situation där styrsignalen är större än den momentana tillgängliga, möjliggjordes omvandlingen mellan dessa storheter, även i kritiska situationer.

Den andra delen föreslår styrlagar och metoder för manövrering av fordonet i 6DoF. Tre olika kontrollstrategier presenteras: pilot-baserad reglering, attityd- och positionsreglering. Den pilot-baserade regulatorn är avsedd att användas av en pilot, som styr enligt flygets standardiserade roll-, tippvinklar och girhastigheter. Reglermetoden som används är en fullständig tillståndsåterkoppling med en reducerad observatör som observerar motordynamiken. Attitydregleringen är en modifiering av den pilot-baserade regleringen, styrandes tipp-, roll- och girvinklar. Slutligen togs en olinjär reglermetod fram för reglering av farkostens position i rummet vilket möjliggör autonomt flygande. Denna olinjära styrmetod är framtagen genom att tillämpa en av Lyapunov's designmetoder och där styrsignalerna konverteras till önskade attitydreferenser som skickas till attitydregleringen.

Styrlagarna utvärderades i en Simulinkmodell där hela det olinjära systemet implementerats. Alla framtagna styrlagar visade lovande resultat och uppvisade önskat beteende. Modellen innehöll även en visualisering av fordonet i 6DoF, körandes i realtid samt möjliggjorde användning av en spelkontroll, vilket möjliggör träning och testning av t.ex. den pilot-baserade regulatorn.

I slutet analyseras flygdata utförd med den pilot-baserade regulatorn. Rapporten avslutas med en diskussion som behandlar tidigare avsnitt och föreslår framtida relevant forskning och arbete.

Contents

1 Introduction **1**

2 The Octocopter **3**
2.1 Electronic Hardware and Software . . . 4
2.1.1 Arduino . . . 4
2.1.2 GC Raspi . . . 5
2.1.3 Nav Raspi . . . 5
2.1.4 IMU . . . 8
2.1.5 GPS . . . 8
2.1.6 ESC . . . 8
2.1.7 PDB . . . 9
2.1.8 Battery . . . 9
2.1.9 Radio transmitter and receiver . . . 9
2.1.10 Motors and Propellers . . . 9
2.2 Moment of Inertia Estimation . . . 11
2.2.1 Motor and Propeller . . . 13
2.2.2 The Octocopter . . . 14

3 Modeling **16**
3.1 Euler Angles . . . 16
3.1.1 Euler Rotation Matrix . . . 17
3.1.2 Euler Angular Rates Matrix . . . 19
3.1.3 Time Derivative of Rotating Body Frame . . . 20
3.1.4 Time Derivative of Rotation Matrices . . . 20
3.2 Quaternions . . . 21
3.3 Equations of Motion for a Rigid Body . . . 22
3.4 Body Forces and Torques . . . 24
3.4.1 Motor Forces and Torques . . . 24
3.4.2 Aerodynamics . . . 26
3.4.3 Precession . . . 27
3.5 Motor dynamics . . . 28
3.6 The Nonlinear Model . . . 29
3.7 Linearized Model . . . 30

4 Control **32**
4.1 Motor Control . . . 33
4.1.1 Minimizing the L^2-norm . . . 34
4.1.2 Minimizing Least Squares with Constraints . . . 37
4.1.3 Control Boundaries and Priority Algorithm . . . 38
4.2 Pilot-based Control . . . 39
4.2.1 Controllability and Observability . . . 40
4.2.2 State Feedback Control . . . 41
4.2.3 Reduced Observer . . . 41
4.2.4 Roll and Pitch Control . . . 43

4.2.5 Yaw-rate Control . . . 54
4.3 Attitude Control . . . 57
4.3.1 Yaw Control . . . 57
4.4 Position Control . . . 60
4.4.1 Simplified Nonlinear Position System . . . 60
4.4.2 Convert Position Control to Attitude Reference . . . 61
4.4.3 Lyapunov Based Control . . . 62
4.4.4 Tuning and Simulation of Controller . . . 63

5 Simulation **66**
5.1 Without Disturbances . . . 67
5.1.1 Pilot-based Control . . . 67
5.1.2 Attitude Control . . . 70
5.1.3 Position Control . . . 71
5.2 With Disturbances . . . 72
5.2.1 Pilot-based Control . . . 72
5.2.2 Attitude Control . . . 74
5.2.3 Position Control . . . 75

6 Evaluation of Flying Data **77**
6.1 Pilot-based Control without Integrating States . . . 78
6.2 Pilot-based Control with Integrating States . . . 80

7 Discussion and Future Work **83**
7.1 The Octocopter . . . 83
7.2 Modeling . . . 84
7.3 Control . . . 84
7.4 Simulation . . . 85
7.5 Evaluation of Flying Data . . . 86

A Experimental Results **87**

Bibliography **91**

Chapter 1

Introduction

Multirotor vehicles, rotorcrafts with more than two rotors e.g., octocopters, have lately gained a lot of attention in both the scientific and commercial sphere. With a greater number of rotors, multirotors are very maneuverable and robust. And because of its simple design, costs can be kept reasonably low.

To efficiently maneuver a multirotor, good control strategies are needed. To derive good control strategies, great insight and understanding of the system is required. Although academical papers have addressed the problem of controlling multirotors, deeper understanding and analyses are necessary to improve control. By reading papers and articles addressing the topic, it becomes clear that the majority only derive control strategies on paper and therefore may not be feasible to implement, thus requiring further attention and development. During this thesis the whole chain from mathematical analysis, modeling, control and simulation to testing of a final product is performed, which is fairly uncommon.

The department of Guidance and Navigation at Saab Dynamics in Linköping bought 2013 a radio controlled octocopter, mainly to investigate how it in some manner can be controlled and used. Two previous master thesis have been conducted on the matter. The first one focused on building and testing of the octocopter [8]. During the second thesis a mathematical model was derived, different important properties were measured and a PID and L1 adaptive controller were derived and simulated [12].

This year's master's thesis project at Saab is conducted by the author of this thesis, Oscar Oscarson from KTH, Royal Institute of Technology in collaboration with Markus Mikkelsen from Umeå University. The main purposes of this year's thesis can be divided into two parts. Firstly, Saab Dynamics wants us two technologists to together design and develop an opened sourced hardware and software wise platform using the existing octocopter. A reason for this is that last year's thesis students encountered problems when trying to implement their controller because of poor insight in the existing system. The main requirement is that the new platform should be able to run C code which has been generated from Simulink in MATLAB. This since Simulink and MATLAB is a widely used programming language among engineering students and has a lot of predefined functions and documentation. Furthermore, future thesis projects involving the octocopter will be able to easily implement own code and conduct research including the vehicle.

The second part of this thesis project is conducted individually and includes the mathematical modeling, analysis and control of the octocopter. This theoretical part is going to be the main focus in this thesis report since the authors field is within applied mathematics, nevertheless, the author has done extensive work on the octocopter both hardware and software wise during the entire thesis period.

To give an overview of the thesis report a summary of each chapter is made.

- **Chapter 2 - The Octocopter**
  The first section of this chapter summarizes the electrical hardware and software components implemented during the thesis and is fairly separated from the rest of the thesis report. The second section of this chapter proposes and carries out an experimental method of estimating the moment of inertia of important components of the octocopter.
- **Chapter 3 - Modeling**
  In this chapter the modeling of the octocopter is covered. The first part focuses on deriving two suitable coordinate systems to represent a rigid body in 6DoF (six-degrees of freedom). Then the rigid body dynamics are derived using Newton's second law of motion in a rotating reference frame. Body forces and torques such as motor induced properties, aerodynamics and precession torques are then derived. The motor dynamics are also given here. Finally, the complete nonlinear model is presented and linearized.
- **Chapter 4 - Control**
  This chapter covers the complete control of the octocopter, from state estimation to motor control input. The first section investigates and proposes in detail methods for converting the model control inputs, motor forces and torques, to corresponding motor control inputs, angular-rates. Then three different approaches of controlling the octocopter in 6DoF are presented: pilot-based, attitude and position control.
- **Chapter 5 - Simulation**
  In this chapter appropriate simulations of the derived control approaches are carried out. The chapter is divided into two main sections, simulations with and without disturbances. The disturbances include signals noises, system delays and quantization of the motor control input.
- **Chapter 6 - Evaluation of Flying Data**
  This chapter presents and discusses data logged from a real flying occasion of the developed octocopter. Only data from the pilot-based controller is presented and is divided in two sections, with and without the integrating states.
- **Chapter 7 - Discussion and Future Work**
  The thesis ends with a discussion which is divided into sections related to each previously mentioned chapter. The discussion contains thoughts and suggestions about future research and work, but also discusses problems encountered during the thesis project.

Chapter 2

The Octocopter

The physical octocopter used in this thesis is originally manufactured by the German hobby company MikroKopter and goes under the name Okto XL 4S12 [18]. The Okto XL 4S12 is an ARF (almost ready to fly) octocopter with top performance compared with today's RC multirotors.

As mentioned in chapter 1, last year's thesis encountered software problems due to lack of insight in the system. Therefore, during this year's thesis, hardware components are changed to create a more open-source platform. Also, components responsible for navigation are replaced (IMU and GPS) to improve state estimation and to increase insight in the system. The new main added/replaced hardware components are two Raspberry Pi 2 Model B, Ardunio Nano, IMU and GPS. Mechanical solutions such as mountings for components, landing gear, safety cage are also implemented during the thesis. The new configured octocopter can be seen in Figure 2.1.

Figure 2.1: The new configured octocopter which is developed and used during the thesis. The upper black cases contains e.g., the Arduino Nano. The two Raspberry Pi's are contained in the two transparent cases. The red part on the left in the picture is the IMU, the GPS antenna is on the right.

The added components have increased the total mass from 2.5 kg to 3.8 kg. From now on the total mass will be denoted and given by $m = 3.8$ kg. From Figure 2.1 it can be seen that the octocopter has two different arm lengths. These lengths are of importance and used when modeling and controlling the octocopter. These lengths are measured from the geometric center of the octocopter to the geometric center of a motor and are from now on denoted and given by $l_1 = 0.45$ m and $l_2 = 0.35$ m respectively.

All software running on the new added components is written during the thesis except for the ”Navigation Solution” see Section 2.1.3. For further insight in the code written during the thesis, see [14]. In section 2.1 all the electronic components are specified and discussed in more detail. In section 2.2 an experimental method is derived and carried out for estimating important moments of inertia.

2.1 Electronic Hardware and Software

As mentioned, new electronic hardware and software is added and configured. To illustrate the new configuration a schematic diagram of the electronic hardware including power supply and communication is made, see Figure 2.2.

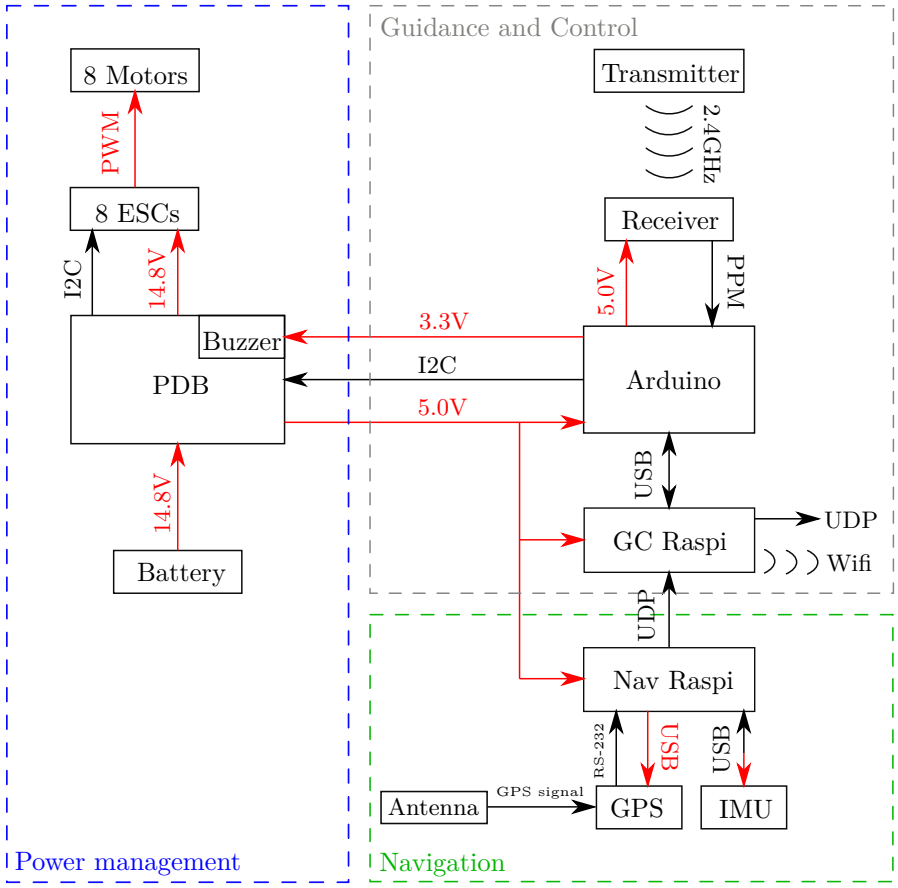

Figure 2.2: Schematic diagram of the electrical hardware of the octocopter. The hardware is divided into the parts: Power management, Navigation and Guidance and Control. Black connections indicate data communication and red connections indicate power supply.

Note that the schematic is somewhat simplified but since the focus of this thesis does not lie in the development process of the octocopter, the scheme fulfills its purpose. The different components in Figure 2.2 are specified and explained in more detail in sections 2.1.1 - 2.1.10.

2.1.1 Arduino

To handle communication with the PDB, Receiver and GC Raspi an Arduino Nano V3.2 is used [1]. The Nano 3.2V has an ATmega328 microcontroller, 14 digital I/O pins, 8 analog input pins and more.

One of the main reasons why the Arduino microcontroller is used, is because it is open-source and well documented. The Ardunio is programmed by using Arduinos IDE (C/C++) which has plenty of predefined libraries. Furthermore, the Nano is only 45×18 mm, weighs 5 g and supports serial communication, external interrupts, PWM, I2C and more. The software and hardware configured with the Nano is written and setup during the thesis. The Arduino measures the battery voltage (indirectly) and enables the buzzer if the battery level is critical.

2.1.2 GC Raspi

The GC Raspi (Guidance and Control Raspberry Pi) communicates with the Arduino, Nav Raspi, runs all control related algorithms and handles the logging of data. The Raspberry Pi 2 Model B is a 900 MHz quad-cored, 1 GB ram credit card-sized single board computer with 4 USB ports, 40 programmable gpio pins, HDMI port, Ethernet port and more [24]. The GC Raspi runs on Raspbian which is a Debian based OS. A D-Link Wireless USB adapter [4] is plugged into one of the USB ports to simplify communication. To make the GC Raspi autonomous, shell scrips is written which e.g., enables auto login, starts relevant written C code, creates and moves logging folders and files. The control algorithms, communications with the Arduino and Nav Raspi and logging of data are written in MATLAB and C and code generated through Simulink to the GC Raspi by Ethernet or WiFi. Currently the GC Raspi runs at 200 Hz.

2.1.3 Nav Raspi

The Nav Raspi (Navigation Raspberry Pi) communicates with the GPS, IMU and GC Raspi and estimates states such as translational and rotational position, velocity and acceleration of the octocopter. The Nav Raspi used is hardware wise identical to the GC Raspi in section 2.1.2. Currently the Nav Raspi runs at 100 Hz.

The Nav Raspi, GPS and IMU are parts from Saab Dynamics "Navigation Solution" where the software is developed by Saab Dynamics. The states are estimated by using a discrete EKF (Extended Kalman Filter). The EKF is the nonlinear version of the Kalman filter which linearizes about the current states and computes the Kalman filter based on the linearization [25, p. 380]. The EKF is not further explained in this thesis. The estimated states $\hat{\boldsymbol{x}}$ from the Nav Raspi were compared (by Saab Dynamics) to the "true" $\boldsymbol{x}$ (estimated by much more powerful components) states, the RMS (root-mean-square) of the error $e_i = x_i - \hat{x}_i$, for some state i is given in Table 2.1.

state x_i	$e_{i_{RMS}}$
x, y	1 m
z	3 m
ϕ, θ	0.1°
ψ	1°
$\dot{x}, \dot{y}, \dot{z}$	0.1 m/s
$\dot{\phi}, \dot{\theta}, \dot{\psi}$	0.1 rad/s
$\ddot{x}, \ddot{y}, \ddot{z}$	0.1 m/s^2

Table 2.1: Table showing the RMS of the error $e_i = x_i - \hat{x}_i$, for some state x_i where $\hat{x}_i$ is the state estimation made by the Nav Raspi.

It is noteworthy to mention that the values in Table 2.1 are computed from data measured after the system has settled in, larger errors occur during the start up phase of the system. According to Saab Dynamics the error $x_i - \hat{x}$ can be modeled as a Gaussian-Markov process. This would require more data

and knowledge about the system. Instead, Gaussian distributed noise $d_i \in \mathcal{N}(\mu_i, \sigma_i^2)$ using $\mu_i = 0$ and $\sigma_i^2 \approx e_{i_{RMS}}^2$ is applied and used when simulating the complete system, see chapter 5.

Noise due to vibrations

Another type of noise, is noise due to vibrations. Since the navigation solution is mounted on the octocopter, noise in the estimated states can be expected due to vibrations originating from the rotating motors and propellers. This noise is experimentally estimated by ramping up all motors in the time interval $t \in [5\ 15]$ with the plastic propellers mounted and then logging relevant states computed by the EKF filter. The result can be seen in Figures 2.3 - 2.4.

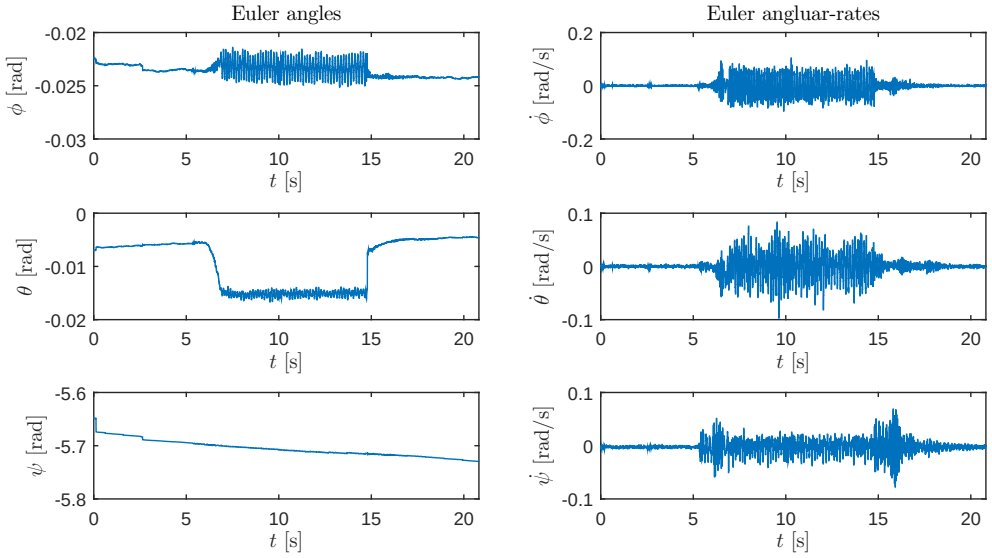

Figure 2.3: Estimated noise due to vibrations, in rotation. All motors have mounted plastic propellers and are ramped up to ≈ 700 rad/s during 10 s.

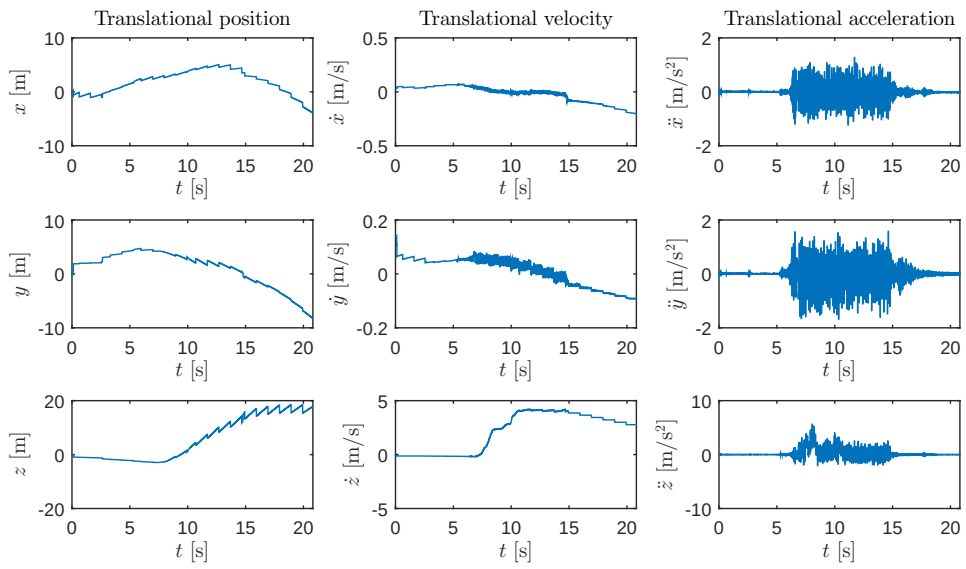

Figure 2.4: Estimated noise due to vibrations, in translation. All motors have mounted plastic propellers and are ramped up to ≈ 700 rad/s during 10 s.

From Figures 2.3 - 2.4 it can be seen that there is some noteworthy noise in the Euler angular-rates and translational accelerations. The noise in the Euler angles is relatively small and decided to be neglected. The small offset in roll θ seen in Figure 2.3 is due to some smaller applied torque generated by the motors. From Figure 2.4 it can be seen that there is some drift in translational position and velocity. This is probably due to a changing GPS position, from which the translational position is calculated and used to estimate the translational velocity. Furthermore, it can be seen that the amplitude of the noise is relatively small and therefore decided to be neglected. Therefore only noise in the Euler angular-rates and translational acceleration is considered. In order to determine the amplitude and frequency content of the noise, a single-sided amplitude spectrum is made using MATLAB's `fft`, see Figure 2.5.

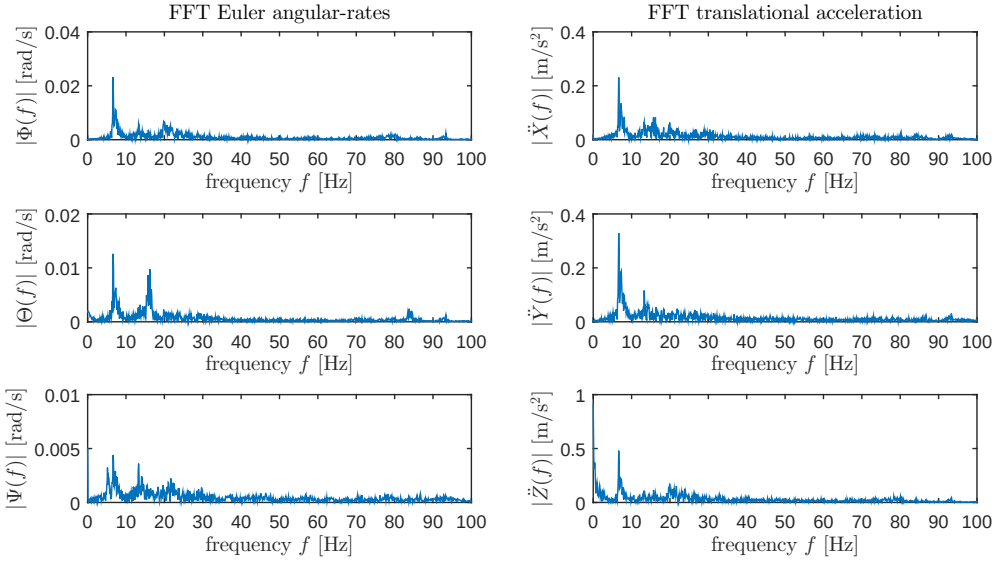

Figure 2.5: Single-sided amplitude spectrum using MATLAB's `fft` of Euler angular-rates and translation acceleration.

Note that the FFT is made during $t \in [5\ 15]$ s, since the motors were running during that time. From Figure 2.5 it can be seen that the most prominent amplitude occur at ≈ 7 Hz. It can also be seen that there is some considerable content at ≈ 15 Hz. It is decided to only consider and model the most prominent signal. The amplitudes for this frequency and state is given in Table 2.2 below.

state	Amp
$\dot{\phi}$	0.02 rad/s
$\dot{\theta}$	0.01 rad/s
$\dot{\psi}$	0.005 rad/s
$\ddot{x},\ \ddot{y}$	0.2 m/s^2
$\ddot{z}$	0.5 m/s^2

Table 2.2: Table showing the amplitudes of the signal content at ≈ 7 Hz, for different states.

The outcome of this approach for determining motor generated vibrations should not be seen as exact but rather as an estimation of size and frequency. These values are used in chapter 5 when simulating the complete system with disturbances.

It is noteworthy to mention that the estimated states are sampled at 100 Hz. Furthermore, the motors rotate at about 700 rad/s which corresponds to 111 Hz. One can argue that the rotating motors will give arise to vibrations near this frequency. By using the Nyquist sampling theorem, the sampling frequency should be at least 200 Hz, in order to measure content to 100 Hz and avoid aliasing. Hence, due to aliasing, the content at 7 Hz could as well be at 107 Hz which is close to the frequency of the rotating motors. In order to accurately determine the frequency content of the vibrations, the sampling frequency has to be increased.

2.1.4 IMU

The IMU (inertial measurement unit) provides acceleration (proper acceleration) and angular rate data to the EKF in section 2.1.3. The IMU used is the MTi-100 IMU manufactured by xsense [27]. The MTi-100 IMU has built in accelerometer, gyroscope, barometer and magnetometer where the two later sensors are currently not used in the EKF. The data is sent with 100 Hz from the 9-pins push-pull connector on the IMU via USB to the Nav Raspi which it also is powered by.

2.1.5 GPS

The GPS provides position data (latitude, longitude and altitude) to the EKF in section 2.1.3. The GPS used is the EVK-6T (evaluation kit) manufactured by u-blox [26]. The kit includes an antenna which is connected by an SMA jack to the evaluation box. The EKV-6T is powered by USB which is connected to the Nav Raspi. The GPS data is read from the evaluation box via a RS-232 serial port which is also connected to the Nav Raspi. The GPS data is sent with 1 Hz.

2.1.6 ESC

There are eight ESCs (electronic speed controller), each controlling a motor by a three phase PWM signal. The ESCs are Mikrokoters BL-Ctrl V2.0 with the possibility to communicate through I2C, PPM and serial [16]. The board measures and provides automatic limitation of current and temperature. The ESCs have 11 bites resolution and two LEDs, one for "error" and one for "OK". Currently I2C is used for sending motor commands to the ESCs at 8 bits resolution $(0 - 255)$, no data from the ESCs are read.

Note that 0 and 255 corresponds to ”motors off” and 254 ”maximum thrust”, see Figure 2.6. Note that with the current configured ESCs it is not possible to reverse the motors.

2.1.7 PDB

The PDB (power distribution board) provides power to all electrical components, distributes the ESCs I2C buss and has a buzzer. The PDB used is Mikrokopters Okto XL PDB V8 [21]. The Ardunio is connected to the PDB to communicate with each ESC via I2C by a single connection. The Ardunio is also connected to the buzzer on the PDB.

2.1.8 Battery

For power supply, the high performing VISLERO LiPo 6600 mAh 4S 20 C 14.8 V is used. The battery is a special flat version with dimensions $145 \times 90 \times 28$ mm and weighs about 715 g [22]. Using a 6600 mAh battery gives an approximate flying time of $14-22$ min depending on flying style [18, Flight times].

2.1.9 Radio transmitter and receiver

For transmitting control signals the Graupner MC-32 HoTT radio controller is used. The MC-32 is a 2.4 GHz premium radio controller with e.g., 16 programmable gimbals/switches, two LCD's and more [7]. For receiving the control signals, Graupner GR-16 HoTT is used [6]. The transmitting signal is a PPM signal of length 30 ms. Currently 4 gimbal axes and 8 switches are used. The PPM signal is read by the Arduino by using external interrupts and then converted to an array of 12 bytes, one for each channel.

2.1.10 Motors and Propellers

The octocopter uses eight Mikrokopter brushless DC motors MK3638 5 mm [20]. The motors are controlled by the ESC which sends a three phase PWM signal. Two types of propellers are currently used, a plastic 12×4.5” (dimater $\times$ slope) [19] and a CFRP (carbon-fiber-reinforced polymer) 12×3.8” [17]. The dynamics of the motors are described in section 3.5. From the previous year's master thesis [12, pp. 56-63], it was found that motor generated thrust and torque are proportional to the squared angular velocities of the motors ω_M i.e.,

$$F_M = c_T \omega^2 \tag{2.1}$$

$$M_M = c_Q \omega^2 \tag{2.2}$$

where for the plastic propeller

$$c_T^{\text{plastic}} = 2.2 \cdot 10^{-5} \text{ Ns}^2 \tag{2.3}$$

$$c_Q^{\text{plastic}} = 4.5 \cdot 10^{-7} \text{ Nms}^2 \tag{2.4}$$

and for the carbon propeller

$$c_T^{\text{carbon}} = 1.9 \cdot 10^{-5} \text{ Ns}^2 \tag{2.5}$$

$$c_Q^{\text{carbon}} = 3.7 \cdot 10^{-7} \text{ Nms}^2. \tag{2.6}$$

From the above values it can be seen that the plastic propeller performs better in means of generating thrust and torque which is due to the greater angle of attack of the blades. Another difference is the stiffness, the carbon propeller is much stiffer and would probably reduce effects such as blade flapping.

In flight, the plastic propellers are used because of their ability to generate thrust and torque and their significant lower price.

As mentioned in section 2.1.6, the motor commands range from $0 - 254$. From now on the motor commands will be denoted u_M and motor angular-rates ω_M. A motor curve with motor commands and corresponding angular-rates is made by using a digital tachometer [2], see Figure 2.6.

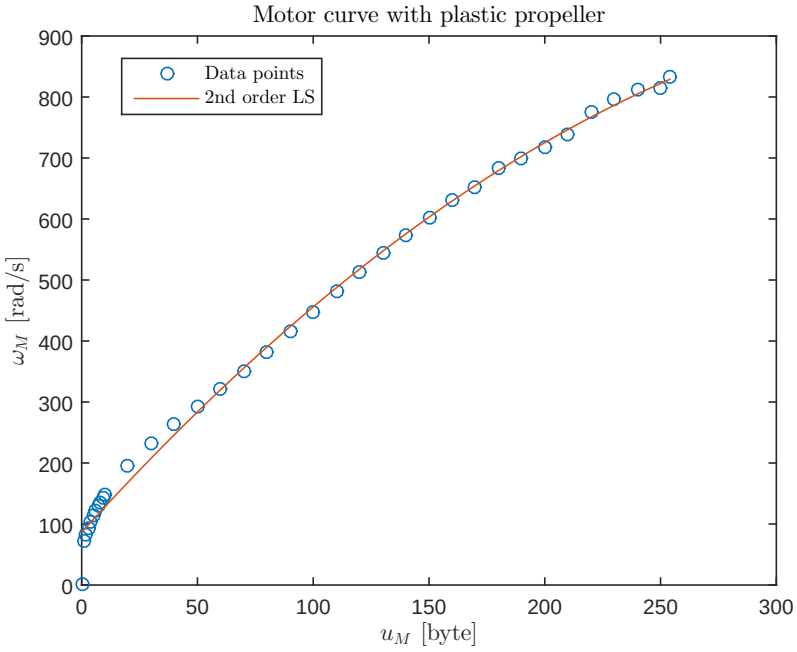

Figure 2.6: Motor curve, with motor commands and corresponding angular-rates. Fitted 2nd order polynomial through the data points marked ∘.

Note that not all motor commands are measured, the data is given in the appendix Table A.1. Furthermore, the measurements are carried out with the plastic propeller mounted.

From Figure 2.6 it becomes clear that a fitted second order polynomial does not match the data very well, especially for smaller u_M. It can also be seen that the motor curve has a more linear behavior when $u_M \gtrsim 20$ byte. To get rid of this nonlinear effect it is decided to have a lower boundary $\omega_M \geq 200$ rad/s. A lower boundary on ω_M above 0 rad/s can also be motivated by means of control. The control algorithm should not be allowed to generate control signals which lie in or are close to the boundary 0 rad/s in any operating condition. Therefore, it is reasonable to have some margin.

Furthermore, it is difficult to accurately measure angular velocities corresponding to motor commands $u_M \gtrsim 200$ byte, since the measurements are fluctuating. At these commands the motor housing becomes very hot and probably affects the performance. Because of this effect it is decided to have an upper boundary $\omega_M \leq 700$ rad/s. By the same analogy as for the lower boundary an upper boundary can also be motivate by means of control and margin.

From this reasoning it is decided to approximate the motor curve by a first order fitted polynomial in LS sense through the data points in the interval $\omega_M \in [200, 700]$ rad/s. The data points and fitted polynomial are illustrated in Figure 2.7.

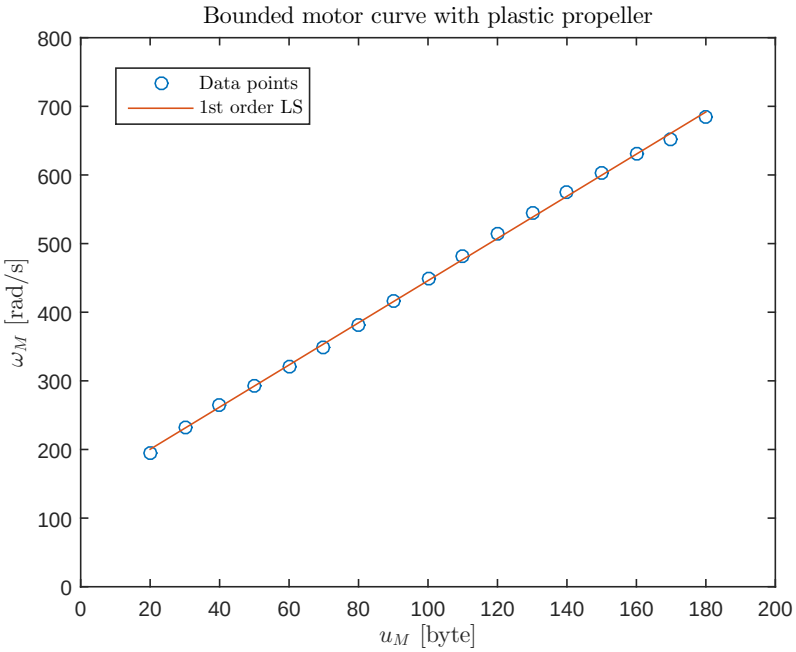

Figure 2.7: Bounded motor curve, with motor commands and corresponding angular-rates. Fitted first order polynomial through the data points marked ∘.

From Figure 2.7 it can be seen that the fitted first order polynomial represents the bounded motor curve very well. The equation describing the polynomial is given by

$$\omega_M = 3.07u_M + 139 \tag{2.7}$$

to three significant figures. In order to convert ω_M to u_M, (2.7) can be used. Solving for u_M gives

$$u_M = 0.326\omega_M - 45.2 \tag{2.8}$$

to three significant figures. Note that more figures for the constants in (2.7) were used when determining (2.8). Also, using (2.8) can result in fractions and since u_M must be an integer (byte), MATLAB's `uint8` is used to convert ω_M to u_M.

2.2 Moment of Inertia Estimation

The new components added to the octocopter have changed its mass and center of gravity and hence its moment of inertia. During the previous year's thesis, the octocopters moment of inertia was estimated analytically where masses and dimensions of components were measured and approximated by different geometrical shapes [12, pp. 54-56]. In this thesis, an experimental approach is chosen for estimating the moments of inertia. The experiment takes use of a bifilar (quadrifiliar) pendulum, which is a two (four) stringed pendulum, where the strings run up parallel to each other attached to e.g., the ceiling. The bifilar pendulum setup of a rod is illustrated in Figure 2.8. The rod has mass m, moment of inertia I and is rotated about the z-axis by an angle θ.

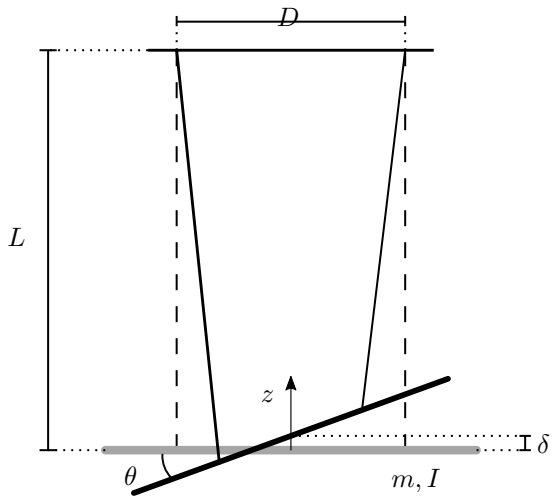

Figure 2.8: Bifilar pendulum of a rod with mass m and moment of inertia I rotated about z by and angle θ. The strings are of length L and separated by the distance D.

Note from Figure 2.8 how the rod has changed its vertical position by δ due to the rotation. By using energy conversion and Lagrangian mechanics the moments of inertia of the above pendulum can be estimated by using its rotational period. The general Lagrangian equation of motion is given by

$$\frac{d}{dt}\left(\frac{\partial L}{\partial \dot{q}_j}\right) - \frac{\partial L}{\partial q_j} = Q_j \tag{2.9}$$

where L is the Lagrangian function, q_j the considered degree of freedom and Q_j non-conservative forces acting in q_j. The Lagrangian function is given by

$$L = T - V \tag{2.10}$$

where T is the kinetic energy and V the potential energy of the system. For the pendulum case, the moment of inertia of interest is the one about the z-axis. Hence, the only degree of freedom ($j = 1$) of interest is $q = \theta$. Furthermore, it is assumed that all non-conservative forces such as damping due to aerodynamic drag, viscosity and kinetics losses due to the strings are neglected, hence $Q = 0$. Before moving on, the vertical displacement δ due to the rotation θ can be determined using trigonometry which gives [15, eqn. (3)]

$$\delta = L\left[1 - \sqrt{1 - \left(\tfrac{D}{\sqrt{2}L}\right)^2 (1 - \cos\theta)}\right]. \tag{2.11}$$

From now on it is assumed that the vertical position of the pendulum is given by $z = \delta$. The vertical velocity is then given by

$$\dot{z} = \frac{\frac{1}{L}\left(\frac{D}{2}\right)^2 \sin\theta}{\sqrt{1 - \left(\frac{D}{\sqrt{2}L}\right)^2 (1 - \cos\theta)}}\dot{\theta}. \tag{2.12}$$

The potential energy of the pendulum is only due to vertical displacement

$$V = mgz \tag{2.13}$$

and the kinetic energy is due to rotation and translation

$$T = \tfrac{1}{2} I\dot{\theta}^2 + \tfrac{1}{2} m\dot{z}^2. \tag{2.14}$$

For the conducted experiments $L \approx 2D$ and $\theta \leq \frac{\pi}{8}$ which together with (2.12) gives $\dot{z}^2 \approx \dot{\theta}^2 \cdot 10^{-3}$ in the worst case, therefore $\dot{z}$ in (2.14) is neglected. By using this assumption and (2.9) - (2.14), one gets

$$I\ddot{\theta} + \frac{\frac{mgD^2}{4L}\sin\theta}{\sqrt{1-\left(\frac{D}{\sqrt{2}L}\right)^2(1-\cos\theta)}} = 0. \tag{2.15}$$

Applying the small angle assumption $\cos\theta \approx 1$ and $\sin\theta \approx \theta$ to (2.15) gives

$$\ddot{\theta} + \frac{mgD^2}{4LI}\theta = 0 \tag{2.16}$$

which has the known solution

$$\theta = c_1\cos(\sqrt{\tfrac{mgD^2}{4LI}}t) + c_2\sin(\sqrt{\tfrac{mgD^2}{4LI}}t) \tag{2.17}$$

where c_1 and c_2 are arbitrary constant. By assuming that $\dot{\theta}(0) = 0$, one gets that $c_2 = 0$. By using this, the moment of inertia can now be expressed by using the periodic time T of (2.16) which gives

$$I = \frac{mg(DT)^2}{16\pi^2 L}. \tag{2.18}$$

This relationship can be used to estimate the moment of inertia of the pendulum in Figure 2.8, by measuring m, D, L and T. Section 2.2.1 and 2.2.2 estimates the moment of inertia for the motor including propeller respectively the octocopter by using this relationship.

2.2.1 Motor and Propeller

In section 3.4.3 it is described how precession torque is generated due to the rotating motors and propellers. The size of the generated torque requires knowledge about the moment of inertia of the rotating motors including propeller which will be denoted $\boldsymbol{I}_M$. It is assumed that the motor coordinate system coincides with the principal axes of inertia, hence the moment of inertia matrix for motor and propeller is given by

$$\boldsymbol{I}_M = \begin{pmatrix} I_{M_{xx}} & 0 & 0 \\ 0 & I_{M_{yy}} & 0 \\ 0 & 0 & I_{M_{zz}} \end{pmatrix}. \tag{2.19}$$

Since the motors only generate rotation about their z-axis, only $I_{M_{zz}}$ needs estimation. To estimate $I_{M_{zz}}$ the experiment described in section 2.2 is carried out. The rotating motor housing including propeller is separated from the motor and hung from the ceiling by two thin strings to match the setup in Figure 2.8, see Figure 2.9

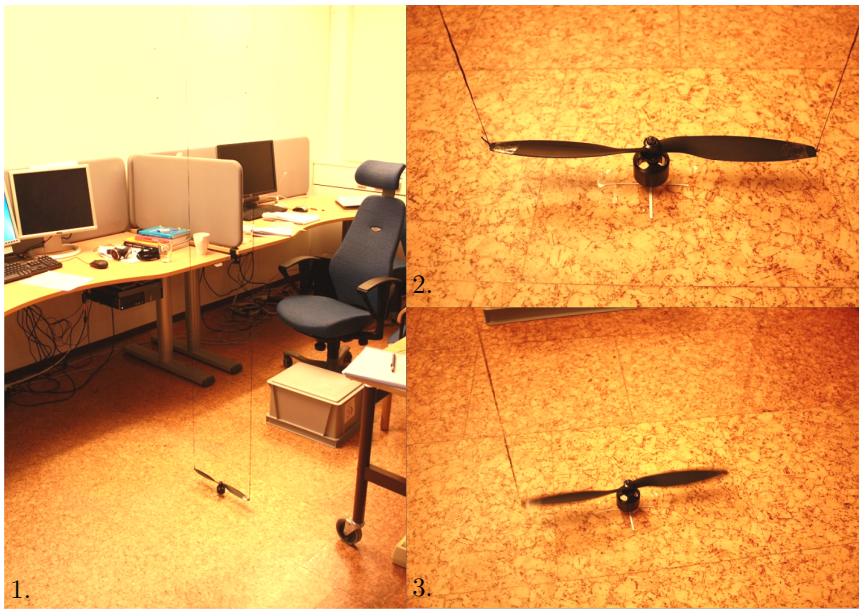

Figure 2.9: Experimental setup for motor and plastic propeller. 1. Hanging from the ceiling by two thin strings. 2. Completely at rest before rotation, cross underneath to mark stationary position 3. Rotating about its vertical axis.

where $L = 2.33$ m, $D = 0.30$ m and $m = 0.064$ kg with the plastic propeller. The motor and propeller are rotated by hand by a small angle $\theta < \frac{\pi}{8}$ which will reduce aerodynamic effects due to smaller rotational velocity and reduce translational movement. A small cross is marked underneath the object when stationary in order to increase accuracy upon release, see Figure 2.9. It is made sure that the two strings are parallel to each other and the object is horizontal in order to increase accuracy. The strings are thin and have limited elasticity to reduce their contribution to the experiment. The total time t is measured for a fixed number of periods N, the period time is then given by $T = \frac{t}{N}$. The experiment is carried out for $N = (10\ 10\ 15\ 15\ 20\ 20\ 25\ 25)$ i.e., two measurements for each N. The experiment is performed for the two different propeller types section 2.1.10. The measurements are given in the appendix Tables A.2-A.3. The mean value of T is computed and the moment of inertia is estimated by using (2.18). The result is shown below to three significant figures for the different propeller types.

$$I_{M_{zz}}^{\text{plastic}} = 1.13 \cdot 10^{-4} \text{ kgm}^2 \tag{2.20}$$

$$I_{M_{zz}}^{\text{carbon}} = 1.07 \cdot 10^{-4} \text{ kgm}^2 \tag{2.21}$$

From the result, it is reasonable that the motor with mounted plastic propeller has a larger moment of inertia since it weighs more, although the difference between them is probably within measurement errors.

2.2.2 The Octocopter

An important property of the octocopter is its moment of inertia $\boldsymbol{I}$. In section 3.3 it can bee seen how it affects the equations of motion. As previously mentioned, new components have been added and the total mass and geometry has changed from the previous year's master thesis. Therefore, a new estimation of $\boldsymbol{I}$ is necessary. Once again, it is assumed that the body coordinate system coincides with the principal axes of inertia, hence the moment of inertia matrix for octocopter is given by

$$\boldsymbol{I} = \begin{pmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{pmatrix}. \tag{2.22}$$

To estimate $\boldsymbol{I}$ the method described in section 2.2 is used. Before moving on, note that the number of strings in the experiment setup Figure 2.8 does not affect the equation (2.18). Now the same procedure as in section 2.2.1 is followed. Figure 2.10 illustrates how the moment of inertia about z is estimated

Figure 2.10: Experiment setup for octocopter. 1. Hanging from the ceiling by four strings. 2. Completely at rest before rotation. 3. Rotating about its vertical axis.

where in this case $L = 2.09$ m, $D = 0.64$ m and $m = 3.8$ kg. Note that the plastic propellers are mounted since they are mainly used and the difference in $\boldsymbol{I}_M$ is neglectable compared to the carbon propellers according to (2.20) and (2.21). Furthermore, the strings used in Figure 2.10 were later exchanged with thinner 0.21 mm PE-braided fishing line [3] to further reduce contribution from the strings. A cross was also marked on the ground as in Figure 2.9 to increase accuracy upon release. All the measured values are found in the appendix Tables A.4-A.6. The moment of inertia for each axis was once again determined by using (2.18). The result is shown below to three significant figures.

$$\boldsymbol{I} = \begin{pmatrix} 0.135 & 0 & 0 \\ 0 & 0.129 & 0 \\ 0 & 0 & 0.222 \end{pmatrix} \text{ kgm}^2 \tag{2.23}$$

By comparing these values to last year's estimations there is an increase in all axis which is reasonable since the new components have increased the total mass by 52 %. The increase of I_{xx} and I_{yy} is most significant and about 26 % respectively 22 % which is due to the new components added in z. The increase of I_{zz} is about 11 %.

Chapter 3

Modeling

When deriving control algorithms it is crucial to have good knowledge and insight in the system requiring control. When modeling an object in all 6DoF i.e., three translations and three rotations, a suitable coordinate system is required. A common approach is to use two reference frames, one *inertial* reference frame denoted by $_I$ which can be compared to the earth, a fixed reference system. The second reference system will be called the *body* reference frame and will be denoted by $_B$. The body reference frame is positioned in the center of mass of the octocopter which is assumed to coincide its geometric center. Furthermore, in this thesis it is assumed that the octocopter can be approximated by a rigid body, more about this later. To illustrate the two reference frames see Figure 3.1 below, note that the x_B-axis is aligned with the forward defined direction of the octocopter.

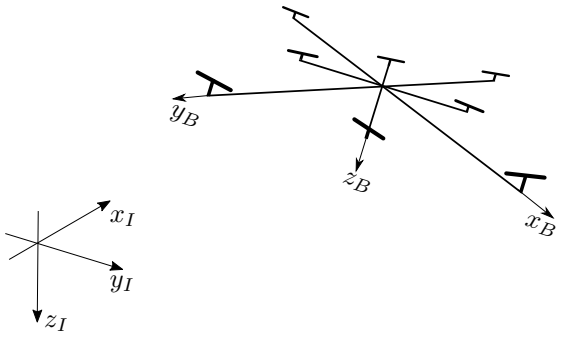

Figure 3.1: The body frame $_B$ moves freely with respect to a fixed inertial frame $_I$.

It is necessary to be able to transform from the different frames in order to express physical properties in the respective frames. In this thesis Euler angles are used, which are presented in section 3.1. To transform properties between frames, Euler rotation and angular-rates matrices can be used, sections 3.1.1 - 3.1.2. An alternative method is also presented, quaternions, which is preferable when simulating in 6DoF, see section 3.2. The general equations of motions for a rigid body in 6DoF are derived in section 3.3. The body forces and torques acting on the octocopter are explained and discussed in section 3.4. The motor dynamics are given in section 3.5. The complete nonlinear model is compactly presented in section 3.6. Lastly, in section 3.7 the system is linearized.

3.1 Euler Angles

A common choice to represent rotations in e.g., aviation is to use the Euler angle representation. The Euler angles represent a sequential rotation about some axes. In this thesis the three Cartesian axes x,

y and z are used where the respective rotation angle is going to be donated by ϕ, θ and ψ. The rotation sequence z-y-x is illustrated in Figure 3.2.

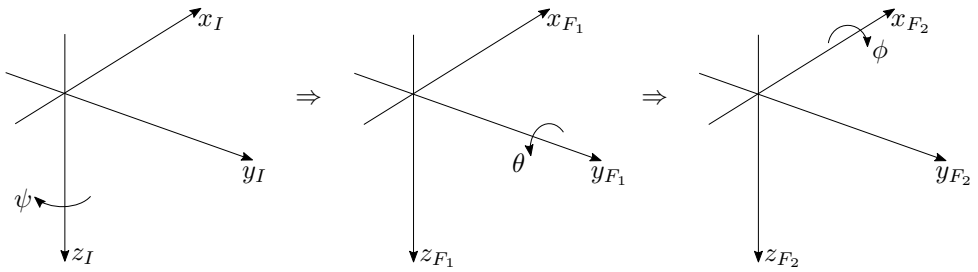

Figure 3.2: Sequential rotation z-y-x of the coordinate system by using Euler angles where $_I$, $_{F_1}$ and $_{F_2}$ are reference frames which change due to rotation.

Note that the "normal" Cartesian system is rotated by π about the x-axis, so that the Euler angles ϕ, θ and ψ coincide with aviation standards for roll, pitch and yaw. Furthermore, the rotation sequence is crucial and does not commute, more about Euler angles in the next section 3.1.1. Before moving on, translational and rotational position of the rigid body in $_I$ are denoted by

$$\boldsymbol{r}_I = \begin{pmatrix} x_I \\ y_I \\ z_I \end{pmatrix}, \ \boldsymbol{\eta} = \begin{pmatrix} \phi \\ \theta \\ \psi \end{pmatrix}. \tag{3.1}$$

3.1.1 Euler Rotation Matrix

The rotation between $_I$ and $_B$ can be performed in twelve possible ways using rotation matrices. In this report the aviation standards for roll, pitch and yaw are followed. When going from $_I$ to $_B$, this corresponds to rotating first about the z-axis, then y-axis and finally the x-axis. This is equivalent to the rotation sequence z-y-x (or x-y-z, $_B$ to $_I$), see Figure 3.3.

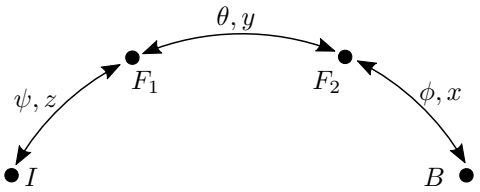

Figure 3.3: Rotation from $_I$ to $_B$ frame by using Euler angle sequence z-y-x or $_B$ to $_I$ with x-y-z.

Here the two helping frames $_{F_1}$ and $_{F_2}$ are introduced, which will be used when deriving the final rotation matrices. Rotation from $_I$ to $_B$ can be performed by using rotation matrices. The rotation matrix from e.g., $_I$ to $_{F_1}$ is going to be denoted by $\boldsymbol{R}_{I \to F_1}$. The rotation matrices, for rotation about different axes, are now derived by using simple trigonometry. When going from the inertial frame $_I$ to $_{F_1}$ i.e., rotation about the z-axis by angle ψ, one gets the following rotation of the reference frame, see Figure 3.4.

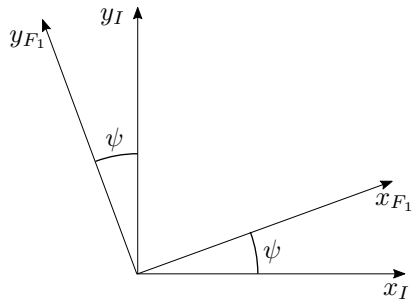

Figure 3.4: Rotation from $_I$ to $_{F_1}$ with ψ.

From which the following relation can be derived

$$\begin{pmatrix} x_{F_1} \\ y_{F_1} \\ z_{F_1} \end{pmatrix} = \begin{pmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_I \\ y_I \\ z_I \end{pmatrix} = \boldsymbol{R}_{I\to F_1}\boldsymbol{r}_I. \tag{3.2}$$

In the same manner the rotation matrices $\boldsymbol{R}_{F_1\to F_2}$ and $\boldsymbol{R}_{F_2\to B}$ can be derived, see Figures 3.5 - 3.6 and equations (3.3) - (3.4).

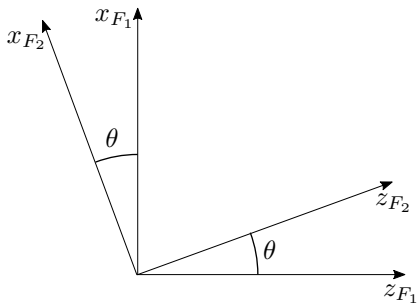

Figure 3.5: Rotation from $_{F_1}$ to $_{F_2}$ with θ.

$$\begin{pmatrix} x_{F_2} \\ y_{F_2} \\ z_{F_2} \end{pmatrix} = \begin{pmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} x_{F_1} \\ y_{F_1} \\ z_{F_1} \end{pmatrix} = \boldsymbol{R}_{F_1\to F_2}\boldsymbol{r}_{F_1}. \tag{3.3}$$

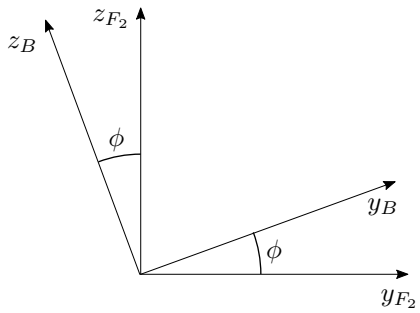

Figure 3.6: Rotation from $_{F_2}$ to $_B$ with ϕ.

$$\begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \begin{pmatrix} x_{F_2} \\ y_{F_2} \\ z_{F_2} \end{pmatrix} = \boldsymbol{R}_{F_2\to B}\boldsymbol{r}_{F_2}. \tag{3.4}$$

The rotation matrix $\boldsymbol{R}_{I\to B}$ can now be obtained by using (3.2) - (3.4) which gives

$$\begin{aligned} \boldsymbol{R}_{I\to B} &= \boldsymbol{R}_{F_2\to B}\boldsymbol{R}_{F_1\to F_2}\boldsymbol{R}_{I\to F_1} \\ &= \begin{pmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ c_\phi s_\theta c_\psi + s_\phi s_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi & c_\phi c_\theta \end{pmatrix} \end{aligned} \tag{3.5}$$

where $\cos x \equiv c_x$ and $\sin x \equiv s_x$. From e.g., Figure 3.4 and (3.2) it can be seen that $\boldsymbol{R}_{I\to F_1} = \boldsymbol{R}_{F_1\to I}^T$, which is a property that holds for all rotation matrices. Hence, the rotation matrix from $_B$ to $_I$ can be obtained as following $\boldsymbol{R}_{B\to I} = \boldsymbol{R}_{F_1\to I}\boldsymbol{R}_{F_2\to F_1}\boldsymbol{R}_{B\to F_2} = \boldsymbol{R}_{I\to F_1}^T\boldsymbol{R}_{F_1\to F_2}^T\boldsymbol{R}_{F_2\to B}^T = (\boldsymbol{R}_{F_2\to B}\boldsymbol{R}_{F_1\to F_2}\boldsymbol{R}_{I\to F_1})^T = \boldsymbol{R}_{I\to B}^T$. Hence

$$\boldsymbol{R}_{B\to I} = \begin{pmatrix} c_\theta c_\psi & s_\phi s_\theta c_\psi - c_\phi s_\psi & c_\phi s_\theta c_\psi + s_\phi s_\psi \\ c_\theta s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & c_\phi s_\theta s_\psi - s_\phi c_\psi \\ -s_\theta & s_\phi c_\theta & c_\phi c_\theta \end{pmatrix} \quad . \tag{3.6}$$

3.1.2 Euler Angular Rates Matrix

In order to achieve continuity between properties in the body and inertial frames, it is important to be able to relate the body angular-rates with the time derivative of the Euler angles $\dot{\boldsymbol{\eta}}$. To do this, it is common to use a transformation matrix which relates the body and Euler angular-rates. The transformation matrix will be denoted $\boldsymbol{W}$ and uses the same indexing as Euler rotation matrices. Before deriving these transformation matrices, the angular-rates of the body frame are defined. The body angular-rates are the denoted by

$$\boldsymbol{\omega} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \tag{3.7}$$

where i.e., ω_x is the angular-rate of the body about its x-axis. In general $\boldsymbol{\omega} \neq \dot{\boldsymbol{\eta}}$, therefore the transformation matrix for the angular rates from frame $_I$ to $_B$ is derived. A simple way of doing so, is to imagine the rotation sequence z-y-x, the first angular rate $\dot{\psi}$ has to undergo two rotations, $_{F_1\to B}$, to align with the body frame, $\dot{\theta}$ has to undergo one rotation, $_{F_2\to B}$ and $\dot{\phi}$ non. Therefore,

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \boldsymbol{R}_{F_1\to B}\begin{pmatrix} 0 \\ 0 \\ \dot{\phi} \end{pmatrix} + \boldsymbol{R}_{F_2\to B}\begin{pmatrix} 0 \\ \dot{\theta} \\ 0 \end{pmatrix} + \begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix}$$

which by using (3.2) - (3.4) can be rewritten as

$$\begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{pmatrix} \begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \boldsymbol{W}_{I\to B}\dot{\boldsymbol{\eta}}. \tag{3.8}$$

The transformation matrix from $\boldsymbol{\omega}$ to $\dot{\boldsymbol{\eta}}$, can be computed by taking the inverse of $\boldsymbol{W}_{I\to B}$ which gives

$$\begin{pmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \boldsymbol{W}_{B\to I}\boldsymbol{\omega}. \tag{3.9}$$

By inspecting $\boldsymbol{W}_{B\to I}$, one can see that for some θ problems might occur, this since $\sec\theta = \frac{1}{\cos\theta}$, hence singularities occur when $\theta = \frac{\pi}{2} + \pi k$ for $k \in \mathbb{N}$. This phenomenon is usually called *gimbal lock* [10, p. 13]. When using the rotation sequence z-y-x i.e., aviation standard, this problems arises when the vehicle has pitch $\theta = \pm\pi$. Under normal operating conditions the octocopter will not experience such a large pitch, however this singularity can be avoided by using quaternions, see section 3.2.

3.1.3 Time Derivative of Rotating Body Frame

The relation between the time derivative of a vector in the body frame and the inertial frame is of importance when deriving the equations of motions in 6DoF and is now given. Let $\boldsymbol{r}$ be an arbitrary and general position vector following the rotation of a rotating reference frame e.g., the body frame. Furthermore, let $(\dot{\boldsymbol{r}})_X$ be the time derivative taken in frame $_X$. Assume that the body frame is rotating with the angular rate $\boldsymbol{\omega}$. The time derivative of $\boldsymbol{r}$ with respect to a fixed inertial reference frame will now be motivated by using Figure 3.7.

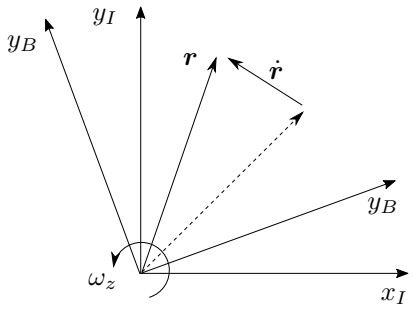

Figure 3.7: Rotation of the body frame, illustrated about the z-axis.

From Figure 3.7, one can argue that the time derivative of $\boldsymbol{r}$ in the inertial frame is

$$(\dot{\boldsymbol{r}})_I = \boldsymbol{\omega} \times \boldsymbol{r} \tag{3.10}$$

since $(\dot{\boldsymbol{r}})_B = 0$, [10, p. 7]. Now assume that the body frame $_B$ also is moving with velocity $\boldsymbol{v}$, thus

$$(\dot{\boldsymbol{r}})_B = \boldsymbol{v}. \tag{3.11}$$

Using (3.10) and (3.11) gives an expression for the time derivative of $\boldsymbol{r}$ in the inertial frame

$$(\dot{\boldsymbol{r}})_I = (\dot{\boldsymbol{r}})_B + \boldsymbol{\omega} \times \boldsymbol{r}. \tag{3.12}$$

3.1.4 Time Derivative of Rotation Matrices

The time derivative for rotation matrices is here derived and can be applied when deriving the equations of motions for a rigid body. Let $\boldsymbol{r}_X$ be a vector expressed in frame $_X$ and $(\dot{\boldsymbol{r}})_X$ be the time derivative taken in frame $_X$. Furthermore, assume that $\boldsymbol{r}$ is rotating with angular rate $\boldsymbol{\omega}$ with respect to the initial frame so $\boldsymbol{\omega} = \boldsymbol{\omega}_I = -\boldsymbol{\omega}_B$. By definition

$$\boldsymbol{r}_B = \boldsymbol{R}_{I\to B}\boldsymbol{r}_I. \tag{3.13}$$

Taking the time derivative in the inertial frame of (3.13) gives

$$\begin{aligned}(\dot{\boldsymbol{r}}_B)_I &= (\dot{\boldsymbol{R}}_{I\to B})_I\boldsymbol{r}_I + \boldsymbol{R}_{I\to B}(\dot{\boldsymbol{r}}_I)_I \\ &= \{(\dot{\boldsymbol{r}}_I)_I = 0\} \\ &= (\dot{\boldsymbol{R}}_{I\to B})_I\boldsymbol{r}_I.\end{aligned} \tag{3.14}$$

But from (3.10) one gets

$$\begin{aligned}(\dot{\boldsymbol{r}}_B)_I &= -\boldsymbol{\omega}_B \times \boldsymbol{r}_B \\ &= -\boldsymbol{\omega}_B \times \boldsymbol{R}_{I\to B}\boldsymbol{r}_I.\end{aligned} \tag{3.15}$$

Hence, combining (3.14) and (3.15) gives

$$(\dot{\boldsymbol{R}}_{I\to B})_I = -\boldsymbol{\omega}_B \times \boldsymbol{R}_{I\to B}. \tag{3.16}$$

By the same analogy one gets that

$$(\dot{\boldsymbol{R}}_{B\to I})_B = \boldsymbol{\omega}_I \times \boldsymbol{R}_{B\to I}. \tag{3.17}$$

3.2 Quaternions

As mentioned in section 3.1.2, singularities arise when using Euler angles, rotation matrices and having a pitch angle θ of $\pm\pi$. Although such a large pitch angle is not within usual operating conditions, it might occur when simulating an unstable or uncontrolled system. To get around this problem an alternative representation of an object in 6DoF is used, quaternions. Quaternions are members of a noncommutative division algebra first invented by William Rowan Hamilton in 1843 [28]. A quaternion is a four-element vector consisting of one real part and three imaginary parts. The unit imaginary parts are denoted by i, j and k and the fundamental formula of quaternions is [28, eq. (1)]

$$i^2 = j^2 = k^2 = ijk = -1.$$

Quaternions can be used for many things and the math behind them is extensive. In this thesis the ability to express rotations is used. For clarification, the four valued unit quaternion can express rotations in three dimensions which can be compared to the Euler's unit formula $e^{it} = \cos t + i \sin t$, which can express a rotation in two dimensions using three components i.e., $(t,\ \cos t,\ \sin t)$. The quaternion q is defined as [28, eq. (20)]

$$q = q_0 + q_1 i + q_2 j + q_3 k$$

from which the quaternion vector $\boldsymbol{q}$ is defined [28, eq. (25)]

$$\boldsymbol{q} \equiv \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^\intercal \equiv \begin{pmatrix} s & \boldsymbol{v} \end{pmatrix}^\intercal \tag{3.18}$$

where s is a called the scalar part and $\boldsymbol{v}$ the vector part corresponding to the imaginary unit vectors i, j and k. The norm of a quaternion is given by [28, eq. (24)]

$$||\boldsymbol{q}|| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} \tag{3.19}$$

and called a unit quaternion if $||\boldsymbol{q}|| = 1$. From now on assume that the imaginary unit vectors i, j and k coincide with the Cartesian axes x, y and z. Furthermore, a unit quaternion on the form

$$\boldsymbol{q} = \begin{pmatrix} \cos\frac{\phi}{2} & \sin\frac{\phi}{2}\boldsymbol{v} \end{pmatrix}^\intercal \tag{3.20}$$

can be used to express a rotation by ϕ about an arbitrary axis $\boldsymbol{v}$ in $\mathbb{R}^3$ [10, eq. (175)]. Note that $\boldsymbol{v}$ has to be scaled such that $||\boldsymbol{q}|| = 1$. A sequence of rotation about multiple axes can be achieved by using quaternion multiplication. Quaternions multiplication will here be denoted $\otimes$ and for two arbitrary quaternions $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ the multiplications is defined by [28, eq. (26)-(27)]

$$\begin{aligned}\boldsymbol{q}_1 \otimes \boldsymbol{q}_2 &= \begin{pmatrix} s_1 & \boldsymbol{v}_1 \end{pmatrix}^\intercal \otimes \begin{pmatrix} s_2 & \boldsymbol{v}_2 \end{pmatrix}^\intercal \\ &= \begin{pmatrix} s_1 s_2 - \boldsymbol{v}_1 \cdot \boldsymbol{v}_2 & s_1\boldsymbol{v}_2 + s_2\boldsymbol{v}_1 + \boldsymbol{v}_1 \times \boldsymbol{v}_2 \end{pmatrix}^\intercal\end{aligned} \tag{3.21}$$

where $\cdot$ is the dot product and $\times$ the cross product. The final operation necessary for rotating an arbitrary vector in $\mathbb{R}^3$ by using quaternions is the quaternions conjugate $\bar{\boldsymbol{q}}$ which is given by [28, eq. (21)]

$$\bar{\boldsymbol{q}} = \begin{pmatrix} s & -\boldsymbol{v} \end{pmatrix}^\intercal. \tag{3.22}$$

Now it is possible to use the above operations and quaternions to perform a rotation of an arbitrary vector in $\mathbb{R}^3$. Assume that $\boldsymbol{r}_B \in \mathbb{R}^3$ is in the body reference frame $_B$ and shall be rotated to an inertial reference frame $_I$. This can be achieved by a unit quaternion $\boldsymbol{q}_{B\to I}$ and the following operation [10, eq. (122)]

$$\begin{pmatrix} 0 & \boldsymbol{r}_I \end{pmatrix}^\intercal = \boldsymbol{q}_{B\to I} \otimes \begin{pmatrix} 0 & \boldsymbol{r}_B \end{pmatrix}^\intercal \otimes \bar{\boldsymbol{q}}_{B\to I}. \tag{3.23}$$

By letting $\boldsymbol{q}_{B\to I} = (q_0 \ q_1 \ q_2 \ q_3)$ and performing the above quaternions multiplication one can derive a rotation matrix $\boldsymbol{Q}_{B\to I}$ relating $\boldsymbol{r}_B$ and $\boldsymbol{r}_I$, this will give [10, eq. (125)-(127)]

$$\boldsymbol{r}_I = \begin{pmatrix} q_0^2+q_1^2-q_2^2-q_3^2 & 2(q_1q_2-q_0q_3) & 2(q_0q_2+q_1q_3) \\ 2(q_0q_3+q_1q_2) & q_0^2-q_1^2+q_2^2-q_3^2 & 2(q_2q_3-q_0q_1) \\ 2(q_1q_3-q_0q_2) & 2(q_0q_1+q_2q_3) & q_0^2-q_1^2-q_2^2+q_3^2 \end{pmatrix} \boldsymbol{r}_B = \boldsymbol{Q}_{B\to I}\boldsymbol{r}_B \tag{3.24}$$

where the following holds $\boldsymbol{Q}_{I\to B} = \boldsymbol{Q}_{B\to I}^\intercal$ [10, eq. (126)-(127)]. By combining (3.20) and (3.21) the quaternion $\boldsymbol{q}_{B\to I}$ can be formed, performing the Euler rotation sequence x-y-z using Euler angles, which gives

$$\boldsymbol{q}_{B\to I} = \begin{pmatrix} \cos\frac{\psi}{2} \\ \sin\frac{\psi}{2}\boldsymbol{e}_z \end{pmatrix} \otimes \begin{pmatrix} \cos\frac{\theta}{2} \\ \sin\frac{\theta}{2}\boldsymbol{e}_y \end{pmatrix} \otimes \begin{pmatrix} \cos\frac{\phi}{2} \\ \sin\frac{\phi}{2}\boldsymbol{e}_x \end{pmatrix} \tag{3.25}$$

where $\boldsymbol{e}_x, \boldsymbol{e}_y$ and $\boldsymbol{e}_z$ are the Cartesian unit vectors respectively. So by above means, an arbitrary vector in $\mathbb{R}^3$ can be rotated by using quaternions and Euler angles. When simulating an object by using quaternions it is often necessary to be able to convert unintuitive quaternions to e.g., Euler angles. For the rotation sequence x-y-z one gets by using the rotation matrices $\boldsymbol{R}_{B\to I}$ and $\boldsymbol{Q}_{B\to I}$ and solving for ϕ, θ and ψ [10, eq. (73)]

$$\begin{aligned} \phi &= \operatorname{atan2}(2(q_0q_1+q_2q_3),\ q_0^2-q_1^2-q_2^2+q_3^2) \\ \theta &= \arcsin 2(q_0q_2-q_1q_3) \\ \psi &= \operatorname{atan2}(2(q_0q_3+q_1q_2),\ q_0^2+q_1^2-q_2^2-q_3^2) \end{aligned} \tag{3.26}$$

where atan2 is the four-quadrant inverse tangent and is used to expand the output interval to $[-\pi, \pi]$, since arctan $\in [-\frac{\pi}{2},\ \frac{\pi}{2}]$. To be able to produce time continuous quaternions its time derivative $\dot{\boldsymbol{q}}$ is necessary. By using the angular-rate $\boldsymbol{\omega}$ of an object, $\dot{\boldsymbol{q}}$ can be expressed as [10, eq. (156)]

$$\dot{\boldsymbol{q}} = \tfrac{1}{2}\boldsymbol{q} \otimes \begin{pmatrix} 0 & \boldsymbol{\omega} \end{pmatrix}^\intercal. \tag{3.27}$$

3.3 Equations of Motion for a Rigid Body

Now the equations of motions for a freely moving rigid body are derived. A rigid body is an idealization of a solid body where deformations are neglected. Meaning that the distance between particles within the object will remain constant even when subjected to external forces. This idealization is often made and does also apply well in this case. Furthermore, it is assumed that the body reference frame is positioned in the center of mass of the octocopter which is assumed to coincide with its geometric center. These assumptions greatly simplifies the calculations made in order to derive the equations of motions.

There are many methods and approaches for deriving the equations of motions for a rigid body. The most common ones are the Lagrangian method which uses conservation of energy and the Euler-Langrange equation or Newtonian mechanics which uses Newton's second law in a rotating reference frame. In this

thesis, the later one is used. Newton's second law for a rigid body in $\mathbb{R}^3$ in an inertial frame, affected by the external forces and torques $\boldsymbol{F}$ and $\boldsymbol{M}$ receptively is given by

$$\dot{\boldsymbol{p}} = \boldsymbol{F} \tag{3.28}$$

$$\dot{\boldsymbol{L}} = \boldsymbol{M} \tag{3.29}$$

where $\boldsymbol{p}$ is the linear momentum and $\boldsymbol{L}$ the angular momentum which can be expressed as

$$\boldsymbol{p} = m\boldsymbol{v} \tag{3.30}$$

$$\boldsymbol{L} = \boldsymbol{I}\boldsymbol{\omega}. \tag{3.31}$$

Here m is the mass, $\boldsymbol{I}$ the moment of inertia and $\boldsymbol{\omega}$ the angular velocity of the rigid body. From now on the "derivative index" is dropped and it is assumed that the derivative is taken in the same frame as in which the vector is represented in, i.e., $\dot{\boldsymbol{r}}_X$ is the time derivative of $\boldsymbol{r}_X$ in frame $_X$. Before going on, it is assumed that $\dot{m} = 0$, which is true for the electric octocopter. Next, let $\boldsymbol{I}$ be the inertial matrix of the rigid body in the body frame. The equations of motion in the inertial frame are easily derived using (3.28) - (3.31) giving

$$\dot{\boldsymbol{v}}_I = \frac{1}{m}\boldsymbol{F_I} \tag{3.32}$$

$$\dot{\boldsymbol{\omega}}_I = \boldsymbol{I}_I^{-1}(\boldsymbol{M_I} - \dot{\boldsymbol{I}}_I\boldsymbol{\omega}_I). \tag{3.33}$$

The equations of motion in the body frame can not be derived as easily since the body frame is rotating. One approach of doing this, is to apply rotation matrices and their time derivatives to obtain the equations of motions in the body frame. By definition

$$\boldsymbol{v}_B = \boldsymbol{R}_{I\to B}\boldsymbol{v}_I$$

and taking the time derivative gives

$$\begin{aligned}
\dot{\boldsymbol{v}}_B &= \dot{\boldsymbol{R}}_{I\to B}\boldsymbol{v}_I + \boldsymbol{R}_{I\to B}\dot{\boldsymbol{v}}_I \\
&= \{(3.17),\ (3.28),\ (3.30)\} \\
&= -\boldsymbol{\omega}_B \times \boldsymbol{R}_{I\to B}\boldsymbol{v}_I + \boldsymbol{R}_{I\to B}\frac{1}{m}\boldsymbol{F_I} \\
&= -\boldsymbol{\omega}_B \times \boldsymbol{v}_B + \frac{1}{m}\boldsymbol{F_B}.
\end{aligned}$$

In the same manner, by definition

$$\boldsymbol{L}_B = \boldsymbol{R}_{I\to B}\boldsymbol{L}_I$$

and taking the time derivative gives

$$\begin{aligned}
\dot{\boldsymbol{L}}_B &= \dot{\boldsymbol{R}}_{I\to B}\boldsymbol{L}_I + \boldsymbol{R}_{I\to B}\dot{\boldsymbol{L}}_I \\
&= \{(3.16),\ (3.29)\} \\
&= -\boldsymbol{\omega}_B \times \boldsymbol{R}_{I\to B}\boldsymbol{L}_I + \boldsymbol{R}_{I\to B}\boldsymbol{M_I} \\
&= -\boldsymbol{\omega}_B \times \boldsymbol{L}_B + \boldsymbol{M_B}
\end{aligned}$$

and by using that $\boldsymbol{L}_B = \boldsymbol{I}\boldsymbol{\omega}_B$ gives

$$\begin{aligned}
\dot{\boldsymbol{L}}_B &= \dot{\boldsymbol{I}}_B\boldsymbol{\omega}_B + \boldsymbol{I}_B\dot{\boldsymbol{\omega}}_B \\
&= \{\boldsymbol{I}_B = 0, \text{ in the body frame}\} \\
&= \boldsymbol{I}_B\dot{\boldsymbol{\omega}}_B.
\end{aligned}$$

Which gives the final body equations of motions

$$\dot{\boldsymbol{v}}_B = \frac{1}{m}\boldsymbol{F_B} - \boldsymbol{\omega}_B \times \boldsymbol{v}_B \tag{3.34}$$

$$\dot{\boldsymbol{\omega}}_B = \boldsymbol{I}_B^{-1}(\boldsymbol{M_B} - \boldsymbol{\omega}_B \times \boldsymbol{I}_B\boldsymbol{\omega}_B). \tag{3.35}$$

3.4 Body Forces and Torques

Now the body forces $\boldsymbol{F}_B$ and torques $\boldsymbol{M}_B$ in (3.34) - (3.35) are defined. For a freely moving rigid body these forces are usually easier expressed in the body system than the inertial, therefore properties following in this section are expressed in the body system. The body system follows the octocopter as seen in Figure 3.8.

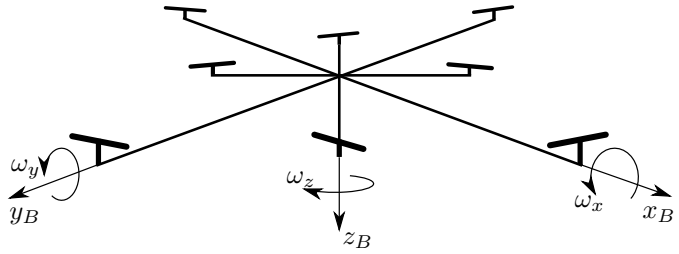

Figure 3.8: The octocopter with aligned body frame with their respective angular velocities.

There are many types of forces and torques which arise and can be modeled but not all of them are as contributing and important. It is assumed that the most prominent ones are the forces and torques generated by the motors, gravitational force, aerodynamic effects and precession torque. Aerodynamics give arise to many phenomenons which can extensively be modeled but are only noted in this thesis. Such effect are e.g., ground effect, down wash, blade flapping and skin friction drag. Precession torque is the effect due to applied moment on a spinning object e.g., toy-gyro rotating in a cone-shaped motion.

3.4.1 Motor Forces and Torques

The eight rotating motors including propellers are used to generate motor forces $\boldsymbol{F}_M$ and torques $\boldsymbol{M}_M$ which are used to stabilize and control the vehicle. Furthermore, the motors rotate in different directions allowing for stability and controllability in yaw. The numbering and rotational direction of each motor is illustrated in Figure 3.9.

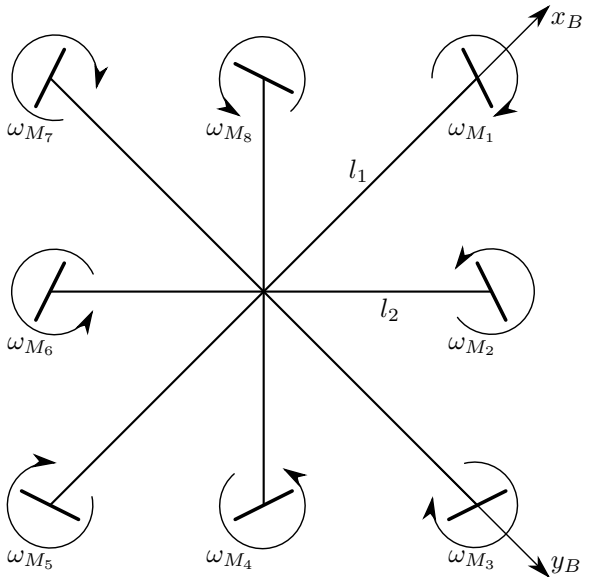

Figure 3.9: The octocopter seen from above. Showing the numbering of each motor and their respective rotational direction and the two different arm lengths l_1 and l_2.

From Figure 3.9 it can be seen that odd and even numbered motors have different arms lengths, l_1 respectively l_2. The position of the geometric center of each motor i, in the body system, is given by

$$\begin{aligned}
\boldsymbol{r}_{M_1} &= \begin{pmatrix} l_1 & 0 & 0 \end{pmatrix}^\intercal \\
\boldsymbol{r}_{M_2} &= \tfrac{1}{\sqrt{2}} \begin{pmatrix} l_2 & l_2 & 0 \end{pmatrix}^\intercal \\
\boldsymbol{r}_{M_3} &= \begin{pmatrix} 0 & l_1 & 0 \end{pmatrix}^\intercal \\
\boldsymbol{r}_{M_4} &= \tfrac{1}{\sqrt{2}} \begin{pmatrix} -l_2 & l_2 & 0 \end{pmatrix}^\intercal \\
\boldsymbol{r}_{M_5} &= \begin{pmatrix} -l_1 & 0 & 0 \end{pmatrix}^\intercal \\
\boldsymbol{r}_{M_6} &= \tfrac{1}{\sqrt{2}} \begin{pmatrix} -l_2 & -l_2 & 0 \end{pmatrix}^\intercal \\
\boldsymbol{r}_{M_7} &= \begin{pmatrix} 0 & -l_1 & 0 \end{pmatrix}^\intercal \\
\boldsymbol{r}_{M_8} &= \tfrac{1}{\sqrt{2}} \begin{pmatrix} l_2 & -l_2 & 0 \end{pmatrix}^\intercal .
\end{aligned} \tag{3.36}$$

To increase the ability to yaw, the motors are tilted by an angle $\alpha = 3^\circ$ with respect to the z_B axis. The motors are tilted in different directions to accomplish stability in yaw. Even and odd numbered motors are tilted in two different directions, see Figure 3.10.

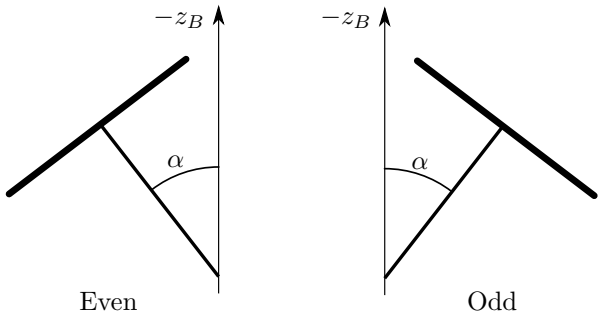

Figure 3.10: Showing how the motors are tilted by an angle α with respect to the z_B axis, seen towards the center of the octocopter. Note that odd and even numbered motors are tilted in different directions.

From Figures 3.9 - 3.10 the direction of each motor i can be determined, in the body system, and are given by the unit vectors

$$\begin{aligned}
\boldsymbol{e}_{M_1} &= \begin{pmatrix} 0 & -s_\alpha & -c_\alpha \end{pmatrix}^\intercal \\
\boldsymbol{e}_{M_2} &= \begin{pmatrix} -\frac{1}{\sqrt{2}} s_\alpha & \frac{1}{\sqrt{2}} s_\alpha & -c_\alpha \end{pmatrix}^\intercal \\
\boldsymbol{e}_{M_3} &= \begin{pmatrix} s_\alpha & 0 & -c_\alpha \end{pmatrix}^\intercal \\
\boldsymbol{e}_{M_4} &= \begin{pmatrix} -\frac{1}{\sqrt{2}} s_\alpha & -\frac{1}{\sqrt{2}} s_\alpha & -c_\alpha \end{pmatrix}^\intercal \\
\boldsymbol{e}_{M_5} &= \begin{pmatrix} 0 & s_\alpha & -c_\alpha \end{pmatrix}^\intercal \\
\boldsymbol{e}_{M_6} &= \begin{pmatrix} \frac{1}{\sqrt{2}} s_\alpha & -\frac{1}{\sqrt{2}} s_\alpha & -c_\alpha \end{pmatrix}^\intercal \\
\boldsymbol{e}_{M_7} &= \begin{pmatrix} -s_\alpha & 0 & -c_\alpha \end{pmatrix}^\intercal \\
\boldsymbol{e}_{M_8} &= \begin{pmatrix} \frac{1}{\sqrt{2}} s_\alpha & \frac{1}{\sqrt{2}} s_\alpha & -c_\alpha \end{pmatrix}^\intercal
\end{aligned} \tag{3.37}$$

where $\cos x \equiv c_x$ and $\sin x \equiv s_x$. The size of the generated force and torque from a motor is proportional to the squared motor angular-rate, see section 2.1.10. Hence, for each motor i the force and torque generated is given by

$$T_{M_i} = c_T \omega_{M_i}^2 \tag{3.38}$$

$$Q_{M_i} = c_Q \omega_{M_i}^2 \tag{3.39}$$

where c_T is the thrust constant and c_Q the torque constant. Note that the torque acting on the octocopter generated from a motor is in the opposite of the motor direction (counter torque). Now the total generated motor forces and torques can be derived by using basic mechanics. The motor generated forces acting on the octocopter are given by

$$\boldsymbol{F}_M = \sum_{i=1}^{8} \boldsymbol{e}_{M_i} T_{M_i} \tag{3.40}$$

and the torques by

$$\boldsymbol{M}_M = \sum_{i=1}^{8} \boldsymbol{r}_{M_i} \times \boldsymbol{e}_{M_i} T_{M_i} + \sum_{i=1}^{8} (-1)^{i+1} \boldsymbol{e}_{M_i} Q_{M_i}. \tag{3.41}$$

Note that the term $(-1)^{i+1}$ in (3.41) arises because of the different rotational directions of the motors.

3.4.2 Aerodynamics

As mentioned before, extensive modeling regarding aerodynamics can be made e.g., ground effect, down wash, blade flapping, skin friction drag. However, it requires good physical knowledge and insight in components and physical experiments to determine aerodynamic properties. The most common and easily modeled aerodynamic effect is form drag. Form drag is most dominant at large velocities and arises due to the movement of a reference area through a fluid. Without going into detail, form drag is given by

$$F_{\text{Drag}} = -\frac{1}{2} \rho C_D v^2 \tag{3.42}$$

where ρ is the air density, C_D the drag coefficient and A the reference area which is perpendicular to the velocity. In the case of the octocopter, drag is generated both due to the movement in translation, but

also due to rotation. Drag force generated due to translation is given by

$$\boldsymbol{F}_D = -\frac{1}{2}\rho C_D \begin{pmatrix} A_{yz} v_x |v_x| \\ A_{xz} v_y |v_y| \\ A_{xy} v_z |v_z| \end{pmatrix} \tag{3.43}$$

where $|\cdot|$ is the absolute value, which is necessary since the aerodynamic drag is acting in opposite direction of the velocity vector. Note that e.g., A_{xy} is the reference area of the xy-plane. Drag generated due to rotation about the z-axis (xz-plane) is illustrated in Figure 3.11.

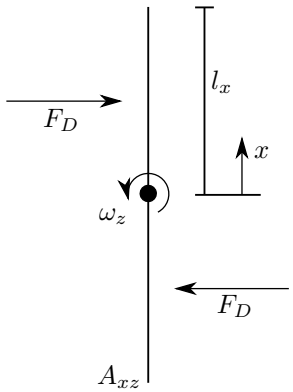

Figure 3.11: Drag generated due to rotation about the z-axis, in the xz-plane. F_D is acting at $x = \frac{l}{2},\ -\frac{l}{2}$ and thus generating a torque. Where A_{xz} is the reference area.

From Figure 3.11 it becomes clear that a torque due to drag is generated opposite the rational direction ω_z. For geometrical reasons the drag force is acting at $x = \frac{l}{2},\ -\frac{l}{2}$ which also can be shown by integrating pressure and angular-rate over the interval $x \in [0,\ l_x]$. Hence, the total torque due to drag in all directions is given by

$$\boldsymbol{M}_D = -\frac{1}{2}\rho C_D \begin{pmatrix} A_{xy} \omega_x |\omega_x| l_x \\ A_{xy} \omega_y |\omega_y| l_y \\ 8 A_{yz} \omega_z |\omega_z| l_z \end{pmatrix}. \tag{3.44}$$

Note that here it is assumed that $A_{xz} = A_{yz}$. The numerical eight in the last row of (3.44) occurs since there are eight reference areas in the yz-plane (eight arms). The area and length parameters were chosen by assuming that octocopter can be approximated by a cube, which results in $A_{xy} = A_{xz} = A_{yz} = 0.04$ m^2 and $l_x = l_y = l_z = 0.2$ m. The following aerodynamic parameters were chosen $\rho = 1.2$ kg/m^3 (air at 20°) and $C_D = 1$ (cube).

3.4.3 Precession

Precession is a motion due to change in orientation of an object rotating about an axis. It is the same phenomenon occurring when a rotating toy-gyro is moving in a cone-shaped motion. So the precession rotation itself does not occur about the axis the object is rotating about. This effect does also affect the octocopter, each motor itself can be seen as toy-gyro rotating and undergoing precession, generating counter moments when the whole octocopter undergoes rotation. Figure 3.12 is trying to illustrate this effect on a tilted disc rotating with angular-rate ω_D

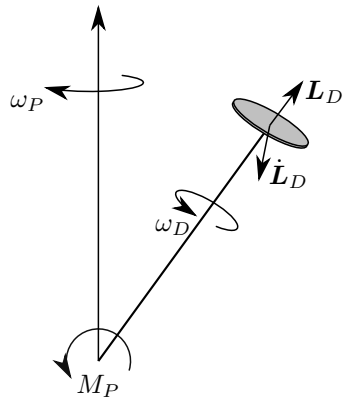

Figure 3.12: Disc with angular momentum $\boldsymbol{L}_D$ rotating with angular-rate ω_D. An applied torque M_P causes the disc to precess with angular-rate ω_P.

where ω_P is the angular-rate of the precession, $\boldsymbol{L}_D$ the angular momentum of the rotating disc and M_P the applied precession torque. The direction of the torque M_P causes $\boldsymbol{L}_D$ to change in the same direction as M_P [5], hence $M_P = \dot{\boldsymbol{L}}_D$ therefore ω_P is clockwise. By using the same motivations about time derivatives of rotating reference systems and cross products in section 3.1.3, one gets that the total precession torque $\boldsymbol{M}_P$ in arbitrary directions can be expressed as [5]

$$\boldsymbol{M}_P = \dot{\boldsymbol{L}}_D = \boldsymbol{\omega}_P \times \boldsymbol{L}_D = \boldsymbol{\omega}_P \times \boldsymbol{I}_D \boldsymbol{\omega}_D. \tag{3.45}$$

Thus, in the case of the rotating motors, the total precession torque generated is given by

$$\boldsymbol{M}_P = \sum_{i=1}^{8} (-1)^{i+1} \boldsymbol{\omega} \times \boldsymbol{I}_M \boldsymbol{e}_{M_i} \omega_{M_i} \tag{3.46}$$

where $\boldsymbol{\omega}$ is the angular-rate of the octocopter. Note that the term $(-1)^{i+1}$ is needed since the direction of each ω_{M_i} is defined as seen in Figure 3.9.

3.5 Motor dynamics

As mentioned in section 2.1.10 the motors are brushless DC motors. An approach to derive a mathematical model of the motors is to use electrical and mechanical relationships. But this requires good knowledge about the motor and its properties. So instead, the motor model derived in the previous year's master's thesis [12, pp. 44-53] is used. The motor dynamics are approximated by a first order low-pass filter where numerical values were found through experiments. The motor dynamics for a motor i are given by

$$\omega_{M_i}(s) = \frac{K_M}{\tau_M s + 1} U_{M_i}(s), \quad \tau_M = \begin{cases} \tau_{M_+} \text{ if } \dot{\omega}_{M_i}(t) \geq 0 \\ \tau_{M_-} \text{ else} \end{cases} \tag{3.47}$$

where K_M is the static motor angular-rate gain, τ_M the time constant and U_{M_i} the input for motor i. Note that τ_M varies between ramping up (τ_{M_+}) or down (τ_{M_-}) the motor. These values were experimentally determined during the previous year's master's thesis and resulted in $\tau_{M_+} = 0.1$ s and $\tau_{M_-} = 0.2$ s. By taking the Laplace transform of (3.47) one gets

$$\dot{\omega}_{M_i}(t) = \frac{1}{\tau_M} \left(-\omega_{M_i}(t) + K_M u_{M_i}(t) \right). \tag{3.48}$$

The gain K_M is just a scaling factor and is from now on assumed to be 1 since in section 2.1.10 a relation between the motor command u_M and motor angular velocity ω_M is found. The different values on the time constant give arise to nonlinearities, these are neglected and from now on it is assumed that $\tau_M = \frac{\tau_{M_+} + \tau_{M_-}}{2} = 0.15$ s.

In section 4.1 it is found that it is possible to convert motor forces and torques to corresponding motor angular-rates $\boldsymbol{\omega}_M$ if the motor forces and torques are kept within some boundaries. By assuming that $\boldsymbol{\omega}_M$ are held within these boundaries and the mentioned assumptions on K_M and τ_M, (3.47) and (3.48) can be rewritten from $u_{M_i} \rightarrow \omega_{M_i}$ to e.g., $u_x \rightarrow F_{M_x}$ i.e., the control input is directly affecting motor forces $\boldsymbol{F}_M$ and torques $\boldsymbol{M}_M$. In vector form this can be written as

$$\boldsymbol{v}_M(s) = \frac{1}{\tau_M s + 1}\boldsymbol{U}(s), \quad \boldsymbol{v}_M(s) = \begin{pmatrix} \boldsymbol{F}_M(s) \\ \boldsymbol{M}_M(s) \end{pmatrix}, \quad \boldsymbol{U}(s) = \begin{pmatrix} U_x(s) \\ U_y(s) \\ U_z(s) \\ U_\phi(s) \\ U_\theta(s) \\ U_\psi(s) \end{pmatrix} \tag{3.49}$$

and in the time domain

$$\dot{\boldsymbol{v}}_M(t) = \frac{1}{\tau_M}\left(-\boldsymbol{v}_M(t) + \boldsymbol{u}(t)\right), \quad \boldsymbol{v}_M(t) = \begin{pmatrix} \boldsymbol{F}_M(t) \\ \boldsymbol{M}_M(t) \end{pmatrix}, \quad \boldsymbol{u}(t) = \begin{pmatrix} u_x(t) \\ u_y(t) \\ u_z(t) \\ u_\phi(t) \\ u_\theta(t) \\ u_\psi(t) \end{pmatrix} \tag{3.50}$$

where $\boldsymbol{U}(s)$ respectively $\boldsymbol{u}(t)$ are from now on the control inputs.

3.6 The Nonlinear Model

The final nonlinear model of the octocopter is compactly presented here. Firstly the derived body forces and torques from sections 3.4.1 - 3.4.3 are gathered as following

$$\boldsymbol{F}_B = \boldsymbol{F}_M + \boldsymbol{F}_D + mg\boldsymbol{R}_{I\rightarrow B}\boldsymbol{e}_z \tag{3.51}$$

$$\boldsymbol{M}_B = \boldsymbol{M}_M + \boldsymbol{M}_D + \boldsymbol{M}_P \tag{3.52}$$

where now the gravitational force is added. Note the positive sign on mg since the z-axis is pointing down, see Figure 3.8. Together with derived rigid body equations of motion (3.34)-(3.35) the body translational and rotational accelerations are given by the following nonlinear differential equation

$$\dot{\boldsymbol{v}}_B = \tfrac{1}{m}(\boldsymbol{F}_M + \boldsymbol{F}_D) + g\boldsymbol{R}_{I\rightarrow B}\boldsymbol{e}_z - \boldsymbol{\omega}_B \times \boldsymbol{v}_B \tag{3.53}$$

$$\dot{\boldsymbol{\omega}}_B = \boldsymbol{I}_B^{-1}(\boldsymbol{M}_M + \boldsymbol{M}_D + \boldsymbol{M}_P - \boldsymbol{\omega}_B \times \boldsymbol{I}_B\boldsymbol{\omega}_B). \tag{3.54}$$

Note that $\boldsymbol{F}_D$ and $\boldsymbol{M}_D$, $\boldsymbol{M}_p$ are functions of $\boldsymbol{v}_B$ respectively $\boldsymbol{\omega}_B$. Furthermore, the control input $\boldsymbol{u}$ is affecting $\dot{\boldsymbol{F}}_M$ and $\dot{\boldsymbol{M}}_M$ as described in (3.50)

The above properties are all defined in the body frame. Usually the purpose is to control the octocopter with respect to an inertial frame, e.g., controlling the Euler angles. Therefore, some operation is required, transforming properties in the body frame to the inertial frame. In this thesis, two methods of doing so have been proposed. The first method uses the Euler angles rotation matrix (3.6) and the Euler angular-rate matrix (3.9). Together with (3.53) - (3.54), the translational and rotational position in the inertial frame can be determined from

$$\boldsymbol{v}_I = \boldsymbol{R}_{B\rightarrow I}\boldsymbol{v}_B \tag{3.55}$$

$$\dot{\boldsymbol{\eta}}_I = \boldsymbol{W}_{B\rightarrow I}\boldsymbol{\omega}_B. \tag{3.56}$$

This method is going to be used when analyzing and deriving control algorithms since it is intuitive to work with Euler angles. The drawback is that singularities might occur when updating the Euler angles using the Euler angular-rates matrix $\boldsymbol{W}_{B\to I}$, see section 3.1.2.

These singularities can be eliminated by using the second approach proposed, quaternions, see section 3.2. This method uses the quaternion rotation matrix (3.24) to convert between the two frames. The initial quaternion is computed by using (3.25) and Euler angles. The quaternion is then updated by using (3.27). So by using quaternions the translational position in the inertial frame can be determined from

$$\boldsymbol{v}_I = \boldsymbol{Q}_{B\to I}\boldsymbol{v}_B \tag{3.57}$$

$$\dot{\boldsymbol{q}} = \tfrac{1}{2}\boldsymbol{q} \otimes \begin{pmatrix} 0 & \boldsymbol{\omega}_B \end{pmatrix}^\intercal \tag{3.58}$$

where $\boldsymbol{Q}_{B\to I}(\boldsymbol{q})$. The rotational position i.e., Euler angles $\boldsymbol{\eta}$, can be obtained by using (3.26). This method is going to be used when simulating the nonlinear system, since gimbal lock can be avoided.

3.7 Linearized Model

In order to apply linear control theory, a linear system is needed. Almost all systems are in some sense nonlinear. When approximating a nonlinear system by a linear one, it is important to determine if the approximation is good and can be motivated. By assuming that the octocopter is kept within some equilibrium i.e., hovering, a linearization of the nonlinear system can be motivated. Furthermore, linear systems are often easier to control and to analyze and many popular control methods are only applicable on linear systems. Therefore, a linearization of the nonlinear system in section 3.6 is made. The linearized system can be derived by doing a first order T.E (Taylor expansion) about some equilibrium. For an arbitrary nonlinear system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u})$ with an equilibrium in $\boldsymbol{x}_0$ and $\boldsymbol{u}_0$ a first order T.E gives

$$\dot{\boldsymbol{x}} \approx \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}_0} \boldsymbol{x} + \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{u}}\right|_{\boldsymbol{u}_0} \boldsymbol{u} \quad \text{where } \boldsymbol{f}(\boldsymbol{x}_0, \boldsymbol{u}_0) = \boldsymbol{0}.$$

In the case of the octocopter, a natural selection of equilibrium is rotation free hovering which implies that $\boldsymbol{v}_B = \boldsymbol{\omega}_B = \boldsymbol{0}$. Without going into detail, linearization of the nonlinear body forces result in

$$\begin{aligned} \boldsymbol{F}_D &\approx \boldsymbol{0} \\ \boldsymbol{M}_D &\approx \boldsymbol{0} \\ \boldsymbol{M}_P &\approx \boldsymbol{0} \end{aligned} \tag{3.59}$$

which is intuitive since no translation nor rotation is accomplished. Rotation free hovering also implies $\dot{\boldsymbol{\eta}} = \boldsymbol{0}$ hence linearization about the equilibrium $\boldsymbol{\eta}_0 = \boldsymbol{0}$ gives the rotation and angular rate matrices

$$\begin{aligned} \boldsymbol{R} &\approx \boldsymbol{I} \\ \boldsymbol{W} &\approx \boldsymbol{I}. \end{aligned} \tag{3.60}$$

Note that the linearization holds for any of the derived Euler rotation matrices. Linearization of the control input $\boldsymbol{u}$ about hovering can simply be accomplished by a change of variable which gives

$$\tilde{\boldsymbol{u}} = \boldsymbol{u} + \begin{pmatrix} 0 & 0 & -mg & 0 & 0 & 0 \end{pmatrix}^\intercal \tag{3.61}$$

where $\tilde{\boldsymbol{u}}$ is the actual control input to the system. By now combining (3.53)-(3.54), (3.59)-(3.61) and the nonlinear model using Euler angular rates matrix (3.56) one gets that

$$\begin{cases} \dot{\boldsymbol{v}}_I &= \boldsymbol{I}\dot{\boldsymbol{v}}_B \\ \dot{\boldsymbol{v}}_B &= \frac{1}{m}\boldsymbol{F}_M \\ \dot{\boldsymbol{\eta}}_I &= \boldsymbol{I}\boldsymbol{\omega}_B \\ \dot{\boldsymbol{\omega}}_B &= \boldsymbol{I}_B^{-1}\boldsymbol{M}_M \end{cases} \Rightarrow \begin{cases} \dot{\boldsymbol{v}}_I &= \frac{1}{m}\boldsymbol{F}_M \\ \ddot{\boldsymbol{\eta}}_I &= \boldsymbol{I}_B^{-1}\boldsymbol{M}_M \end{cases} \tag{3.62}$$

Now a complete linearized model including the motor dynamics is presented. By using the relationship (3.50) relating the control input to the motor forces and torques and (3.62) the translation and rotation accelerations of the octocopter can be written as

$$\begin{pmatrix} \dot{\boldsymbol{v}}_I \\ \ddot{\boldsymbol{\eta}} \\ \dot{\boldsymbol{v}}_M \end{pmatrix} = \frac{1}{\tau_M m} \begin{pmatrix} \tau_M \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \tau_M m \boldsymbol{I}_B^{-1} \\ -m\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & -m\boldsymbol{I} \end{pmatrix} \boldsymbol{v}_M + \frac{1}{\tau_M} \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{pmatrix} \boldsymbol{u} \tag{3.63}$$

where $\boldsymbol{0}, \boldsymbol{I} \in \mathbb{R}^{3\times 3}$. Next a state-space representation of the complete linearized system is made i.e., from control input to position and rotation in space. It is fairly obvious that the system (3.63) is decoupled e.g., only the states $(z, \dot{z}, \ddot{z})$ and input u_z will affect the states $(z, \dot{z}, \ddot{z})$. Therefore, it is not necessary to write the complete system by using just one state-space representation. Instead, each decoupled subsystem is written in the state-space form

$$\begin{aligned} \dot{\boldsymbol{x}}_i &= \boldsymbol{A}_i \boldsymbol{x}_i + \boldsymbol{B}_i u_i \\ \boldsymbol{y}_i &= \boldsymbol{C}_i \boldsymbol{x}_i + \boldsymbol{D}_i u_i, \quad i = 1, 2, \ldots, 6 \end{aligned} \tag{3.64}$$

where

$$\boldsymbol{x}_i = \begin{pmatrix} x_{3i-2} \\ x_{3i-1} \\ x_{3i} \end{pmatrix}, \ \boldsymbol{y}_i = \begin{pmatrix} y_{2i-1} \\ y_{2i} \end{pmatrix} \tag{3.65}$$

and

$$\boldsymbol{A}_i = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & \boldsymbol{c}_i \\ 0 & 0 & -\frac{1}{\tau_M} \end{pmatrix}, \ \boldsymbol{B}_i = \begin{pmatrix} 0 \\ 0 \\ \frac{1}{\tau_M} \end{pmatrix}, \ \boldsymbol{C}_i = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}, \ \boldsymbol{D}_i = \boldsymbol{0}. \tag{3.66}$$

Here $\boldsymbol{c} = (\frac{1}{m}, \frac{1}{m}, \frac{1}{m}, \frac{1}{I_{xx}}, \frac{1}{I_{yy}}, \frac{1}{I_{zz}})$ and e.g., $i = 4$ corresponds to the states and control input

$$\boldsymbol{x}_4 = \begin{pmatrix} x_{10} \\ x_{11} \\ x_{12} \end{pmatrix} = \begin{pmatrix} \phi \\ \dot{\phi} \\ M_{M_\phi} \end{pmatrix}, \ u_4 = u_\phi. \tag{3.67}$$

Note that each last state x_{3i} is not measured, see section 2.1.3. From the state-space model (3.64) with (3.66) a general transfer function matrix $\boldsymbol{G}_i(s)$ can be formulated, relating the control input to the output. It can be obtained by performing a Laplace transform on (3.64) and solving for the transfer function matrix from $\boldsymbol{U}(s)$ to $\boldsymbol{Y}(s)$, this gives

$$\boldsymbol{G}_i(s) = \boldsymbol{C}_i(s\boldsymbol{I} - \boldsymbol{A}_i)^{-1}\boldsymbol{B}_i + \boldsymbol{D}_i. \tag{3.68}$$

By then using the state-space matrices (3.66) the transfer function matrix $\boldsymbol{G}_i(s)$ is given by

$$\boldsymbol{G}_i(s) = \frac{1}{s^2(\tau_M s + 1)} \begin{pmatrix} \boldsymbol{c}_i \\ \boldsymbol{c}_i s \end{pmatrix} \tag{3.69}$$

corresponding to subsystem i.

Chapter 4

Control

In this chapter the complete state and motor control of the octocopter is covered. An octocopter is a good example of vehicle in need of control. The vehicle has unstable dynamics and would without control not be flyable. This chapter contains the major content of this thesis report and is carefully handled. This chapter can be divided into two parts. The first part of this chapter, section 4.1 focuses on proposing a method for converting the control input i.e., motor forces and torques into corresponding motor commands. Then, in section 4.2 - 4.4 control laws are derived for controlling states of the octocopter in 6DoF. In section 4.2 a pilot-based controller controlling the Euler angles ϕ, θ and Euler angular-rate $\dot{\psi}$ is derived. In section 4.3 the pilot-based controller is alternated to an attitude control which controls all Euler angles. Both the pilot-based and attitude controllers use a full state feedback controller with a reduced observer, observing the motor dynamics. Lastly, in section 4.4, a position controller is designed using the nonlinear Lyapunov based control, controlling the translational positions x, y and z.

The control chain of the whole octocopter from desired reference to motor control input can be divided into several parts. An illustration of this control chain can be seen in Figure 4.1

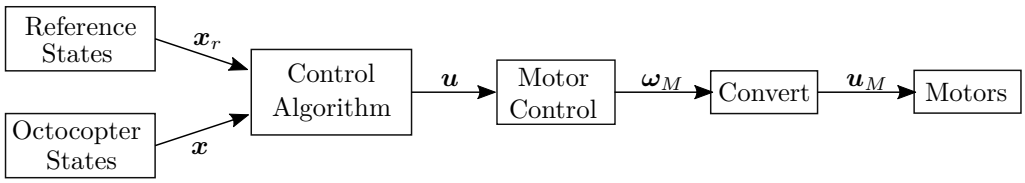

Figure 4.1: The control chain of the octocopter.

where

- **Reference States** - Are the current desired reference states $\boldsymbol{x}_r$ of the octocopter e.g., inputs from radio receiver or a given trajectory. Note that the reference could partially or complete be computed within the block e.g., the octocopter is autonomous.
- **Octocopter States** - Are the current states of the octocopter $\boldsymbol{x}$ e.g., position and velocity. This block uses sensor data and signal fusion technic(s) to compute and estimate the current states.
- **Control Algorithm** - This block computes the system control input $\boldsymbol{u}$ by using the methods described in sections 4.2-4.4. The algorithm may use $\boldsymbol{x}$ and $\boldsymbol{x}_r$.
- **Motor Control** - Converts the model control input $\boldsymbol{u}$ into corresponding motor control inputs $\boldsymbol{\omega}_M$ by using the derived approach in section 4.1.
- **Convert** - This block implements the equation 2.8 which relates $\boldsymbol{\omega}_M$ to $\boldsymbol{u}_M$ and is used for converting between these properties.

- **Motors** - The motors of the octocopter, section 2.1.10. The motor dynamics are given in section 3.5.

4.1 Motor Control

In this section a method for converting the model control input $\boldsymbol{u}$ to motor control inputs $\boldsymbol{u}_M$ i.e., $\omega_{M_i}^2$, is proposed. As mentioned in section 3.5, the model control inputs are affecting the motor forces and torques, not the angular-rates controlling the motors. This requires a method of converting the model control inputs to corresponding angular-rates and is a crucial and important part when designing an octocopter. In section 3.4.1 the relation between the motor forces, torques and motor angular velocities were found to be given by (3.40) - (3.41). From these relationships, a matrix $\boldsymbol{\Gamma} \in \mathbb{R}^{6\times 8}$ can be defined which relates the squared motor angular velocities $\omega_{M_i}^2$ with the model control input in translation $\boldsymbol{u}_t = (u_x,\ u_y,\ u_z)^\intercal$ and rotation $\boldsymbol{u}_r = (u_\phi,\ u_\theta,\ u_\psi)^\intercal$. Let

$$\boldsymbol{\Gamma} = \begin{pmatrix} \boldsymbol{f}_1 & \boldsymbol{f}_2 & \cdots & \boldsymbol{f}_8 \\ \boldsymbol{m}_1 & \boldsymbol{m}_2 & \cdots & \boldsymbol{m}_8 \end{pmatrix} \tag{4.1}$$

where

$$\begin{aligned} \boldsymbol{f}_i &= c_T \boldsymbol{e}_{M_i} \\ \boldsymbol{m}_i &= c_T \boldsymbol{r}_{M_i} \times \boldsymbol{e}_{M_i} + (-1)^{i+1} c_Q \boldsymbol{e}_{M_i}. \end{aligned}$$

By using $\boldsymbol{\Gamma}$ the control inputs $\boldsymbol{u}_t$, $\boldsymbol{u}_r$ can now be determined from to the squared motor angular-rates as following

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_t \\ \boldsymbol{u}_r \end{pmatrix} = \boldsymbol{\Gamma} \boldsymbol{\omega}_{M^2} \tag{4.2}$$

where

$$\boldsymbol{\omega}_{M^2} \equiv \begin{pmatrix} \omega_{M_1}^2 & \omega_{M_2}^2 & \cdots & \omega_{M_8}^2 \end{pmatrix}^\intercal.$$

Since $\boldsymbol{u} \in \mathbb{R}^{6\times 1}$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{6\times 8}$ the linear system (4.2) is underdetermined i.e., more unknowns than equations. This means that there might exist infinitely many solutions to (4.2), which gives arise to the question: if there exists infinitely many solutions, how should a solution be chosen?

First, it is proven that (4.2) has infinitely many solutions by using the Roché-Capelli theorem [23]. Before doing so, for convenience, from now on in this section $\boldsymbol{A} = \boldsymbol{\Gamma}$, $\boldsymbol{x} = \boldsymbol{\omega}_{M^2}$ and $\boldsymbol{y} = \boldsymbol{u}$. Roché-Capelli theorem states that a system $\boldsymbol{Ax} = \boldsymbol{y}$ is consistent, solution(s) exists, if and only if

$$\text{rank}(\boldsymbol{A}) = \text{rank}(\boldsymbol{A}|\boldsymbol{y})$$

where $\boldsymbol{A}|\boldsymbol{y}$ is the augmented matrix. Furthermore, if the system is consistent, the following holds

1. The system has exactly one solution if $n = \text{rank}(\boldsymbol{A})$.
2. The system has infinitely many solutions if $n > \text{rank}(\boldsymbol{A})$.

where n is the number of unknowns. In the case of (4.2), it is checked in MATLAB that $\text{rank}(\boldsymbol{A}) = 6$ i.e., full rank, which also is fairly intuitive. Since $\boldsymbol{A}$ has full rank, the augmented matrix $\text{rank}(\boldsymbol{A}|\boldsymbol{y})$ must also have full rank since it cannot be less than $\text{rank}(\boldsymbol{A})$. Hence according to Roché-Capelli the system (4.2) is consistent. Now it is left to show that infinitely many solutions exists. Since the number of unknowns $n = 8 > \text{rank}(\boldsymbol{A}) = 6$, the system has infinitely many solutions. Since (4.2) has infinitely solutions, it is of importance to determine how a solution should be chosen. Approaches for choosing a solution is discussed in the following sections 4.1.1-4.1.3. Before doing so, in section 2.1.10 lower and

upper boundaries $\omega_{M_{\min}}$ respectively $\omega_{M_{\max}}$ on the angular velocities of the motors are defined. These boundaries i.e., inequality constraints can be written as

$$\boldsymbol{B}\boldsymbol{x} \leq \boldsymbol{c} \tag{4.3}$$

where

$$\boldsymbol{B} = \begin{pmatrix} -\boldsymbol{I} \\ \boldsymbol{I} \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} -\mathbf{1}\omega_{M_{\min}} \\ \mathbf{1}\omega_{M_{\max}} \end{pmatrix}. \tag{4.4}$$

Note that $\boldsymbol{I} \in \mathbb{R}^{8\times 8}$ is the identity matrix and $\mathbf{1} \in \mathbb{R}^{8\times 1}$ is a vector with ones.

4.1.1 Minimizing the L^2-norm

One approach of finding a solution to (4.2) is to minimize the L^2-norm of $\boldsymbol{x}$ i.e., $||\boldsymbol{x}||_2$. Since minimizing the square root of a function is the same as minimizing just the function itself and since $||\boldsymbol{x}||_2^2 = \boldsymbol{x}^\intercal\boldsymbol{x}$, the following optimization is formulated

$$\min_{\boldsymbol{x}} \quad \boldsymbol{x}^\intercal\boldsymbol{x} \quad \text{s.t} \quad \begin{cases} \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} = \mathbf{0} \\ \boldsymbol{B}\boldsymbol{x} - \boldsymbol{c} \leq \mathbf{0} \end{cases} \tag{4.5}$$

where $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{A}$ and $\boldsymbol{B}$ are defined in section 4.1. The optimization problem (4.5) can be solved by using Lagrange multipliers and the KKT (Karush-Kuhn-Tucker) conditions [13]. The details about this approach are not discussed here. For (4.5) the Lagrange function is given by

$$\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\mu}) = \boldsymbol{x}^\intercal\boldsymbol{x} + \boldsymbol{\lambda}^\intercal(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}) + \boldsymbol{\mu}^\intercal(\boldsymbol{B}\boldsymbol{x} - \boldsymbol{c}) \tag{4.6}$$

where $\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ are the Lagrange multipliers determined from to the conditions in (4.5). By using the KKT conditions, the following conditions arise

- $$\left.\frac{\partial \boldsymbol{L}}{\partial \boldsymbol{x}}\right|_* = 2\boldsymbol{x}^{*\intercal} + \boldsymbol{\lambda}^{*\intercal}\boldsymbol{A} + \boldsymbol{\mu}^{*\intercal}\boldsymbol{B} = \mathbf{0} \tag{4.7}$$
- $$\boldsymbol{\mu}^{*\intercal}(\boldsymbol{B}\boldsymbol{x}^* - \boldsymbol{c}) = \mathbf{0} \tag{4.8}$$
- $$\boldsymbol{\mu}^* \geq \mathbf{0} \tag{4.9}$$
- $$\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{y} = \mathbf{0} \tag{4.10}$$
- $$\boldsymbol{B}\boldsymbol{x}^* - \boldsymbol{c} \leq \mathbf{0} \tag{4.11}$$

Firstly, from (4.7) one easily can see that $\frac{\partial^2 \boldsymbol{L}}{\partial^2 \boldsymbol{x}} > 0$, hence $\boldsymbol{x}^*, \boldsymbol{\lambda}^*, \boldsymbol{\mu}^*$ will provide global optimality given that the KKT conditions are fulfilled. From the KKT condition (4.8)

$$\boldsymbol{\mu}^* = \mathbf{0} \text{ or } \boldsymbol{B}\boldsymbol{x}^* = \boldsymbol{c}$$

where the second condition is not feasible since the motors cannot both fulfill the lower and upper limit simultaneously, hence $\boldsymbol{\mu}^* = \mathbf{0}$. By using that, (4.7) and (4.10) one gets that

$$\boldsymbol{\lambda}^* = -2(\boldsymbol{A}\boldsymbol{A}^\intercal)^{-1}\boldsymbol{y}$$

which inserted back into (4.7) gives

$$\boldsymbol{x}^* = \boldsymbol{A}^\intercal(\boldsymbol{A}\boldsymbol{A}^\intercal)^{-1}\boldsymbol{y}. \tag{4.12}$$

So the optimization problem (4.5) has the optimal solution (4.12) if the KKT condition (4.11) is satisfied. The solution matrix $\boldsymbol{A}^\intercal(\boldsymbol{A}\boldsymbol{A}^\intercal)^{-1}$ is called the right pseudo-inverse matrix and will be denoted $\boldsymbol{A}^+$.

As mentioned, the KKT condition (4.11) determines if the solution $\boldsymbol{x}^*$ is optimal. Although these condition are necessary, they do not give a very good intuition of how the solution space of (4.5) looks like. Therefore, a heuristic approach is applied to get a better understanding of the boundary of the solution space. The method chosen is to simply vary two-three elements in $\boldsymbol{y}$ and keep the rest constant while by brute-force determine boundaries at which $\boldsymbol{x}^*$ no longer satisfies (4.11). Since $\boldsymbol{y}$ is the control input i.e., motor forces and torques, it is determined that a natural choice is to vary the control force u_z together with some control torque u_ϕ, u_θ, u_ψ and let $u_x = u_y = 0$. It is natural to vary the control u_z together with some torque since the applied force in z determines how much torques will be available, which also is going to become clear from the Figures below. Furthermore, the control forces u_x and u_y are set to zero since they are very small in relation to u_z and are not going to be used for translational purpose. Firstly only two variables are varied, which resulted in Figures 4.2-4.4.

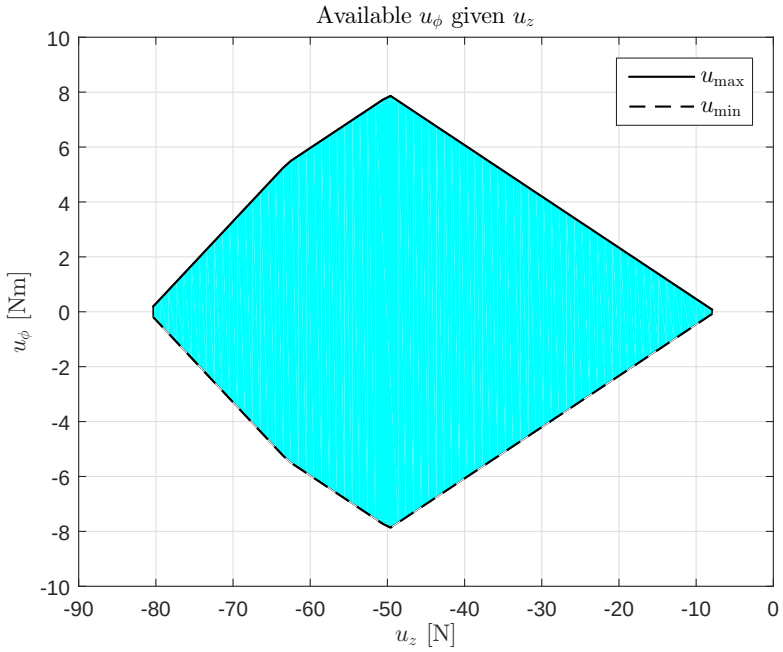

Figure 4.2: Available u_ϕ given u_z. The marked area between and including the two boundaries $u_{\min}$ and $u_{\max}$ is the area where the torque u_ϕ can be generated provided u_z.

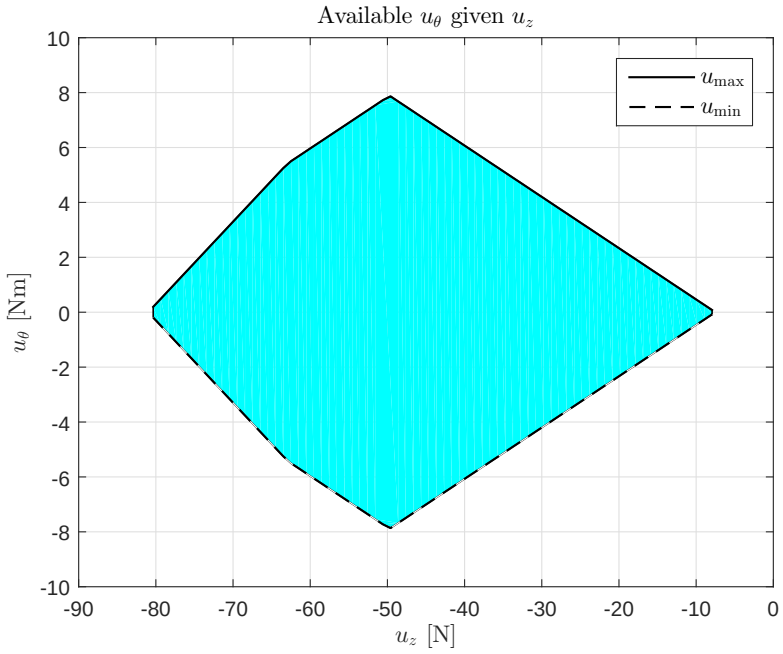

Figure 4.3: Available u_θ given u_z. The marked area between and including the two boundaries $u_{\min}$ and $u_{\max}$ is the area where the torque u_θ can be generated provided u_z.

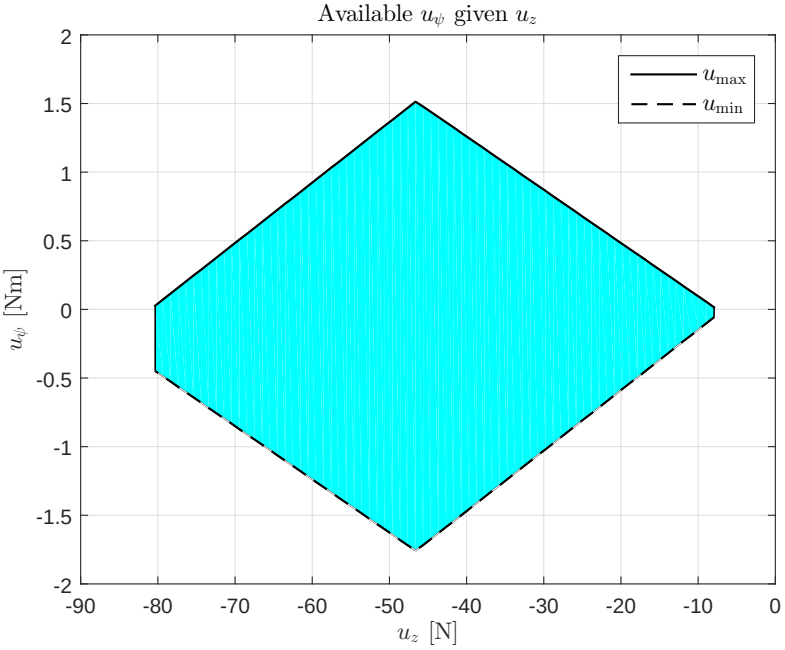

Figure 4.4: Available u_ψ given u_z. The marked area between and including the two boundaries $u_{\min}$ and $u_{\max}$ is the area where the torque u_ψ can be generated provided u_z.

Firstly, from Figures 4.2 - 4.3 it can be seen that torque available in ϕ and θ is the same, which is due to symmetry. Secondly, note that the available force in z begins at $u_z \approx -8$ N and ends at $u_z \approx -80$ N, this is due to the lower and upper boundaries, $200 \leq \omega_M \leq 700$ rad/s.

Furthermore, there is a breakpoint at $u_z \approx -62$ N, this change in slope is due to the tilted motors and

will now be explained. In the interval $u_z = [-62, -8]$ N, the torque u_ϕ is the restricting factor. This means that increasing u_ϕ will result in hitting the lower boundaries on ω_M despite the fact that one could generate more u_z. This will cause the linear slope up to $u_z \approx -62$ N which can be seen in the Figures 4.2 - 4.3. But when $u_z = [-62 - 80]$ N, the force u_z is the restricting factor. This means that there is simply not enough of control in z left to generate torque in ϕ or θ i.e., hitting the upper boundaries on ω_M. Hence, a new linear slope from $u_z \approx -62$ onward can be observed in Figures 4.2 - 4.3.

According to Figure 4.4 the torque u_ψ available is not symmetrical in ψ. This is due to the fact that the arm lengths of the octocopter are different and the motors are tilted. The tilted motors will generate torque in ψ since the arm lengths are different, the longer arms directed counterclockwise will generate a greater amount of torque in negative ψ. Hence, the non symmetrical distribution of u_ψ in ψ.

Lastly, a plot is made where three elements in $\boldsymbol{y}$ are varied, the force u_z and the torques u_ϕ, u_ψ. Because of the mentioned symmetry of the torques in ϕ and θ the torque u_θ is not studied. For illustration a colormap is made where the color corresponds to the minimum force u_z required to generate the torques u_ϕ and u_ψ, see Figure 4.5.

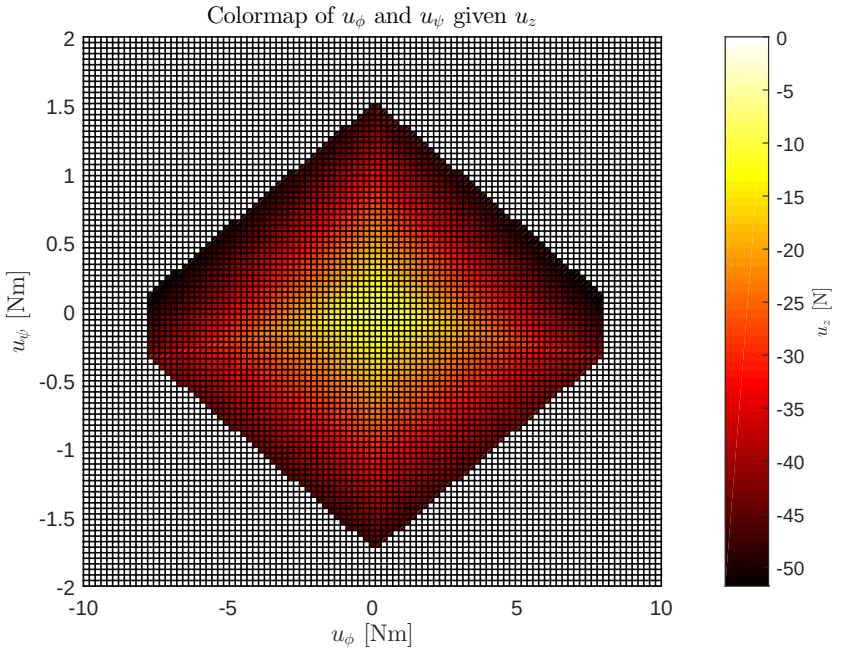

Figure 4.5: Colormap of u_ϕ and u_ψ given u_z. Where u_z is the minimum force required to generate the torques u_ϕ and u_ψ.

From the colormap above, a non symmetrical distribution of u_z about ϕ-axis can be observed i.e., less force u_z is required to generate negative torque u_ψ, which also can be observed in Figure 4.4. One can also see that more negative torque u_ψ is possible to generate. From the Figure it also becomes clear that here the torques u_ϕ and u_ψ are the restricting factors since $|u_z| < 80$ N.

4.1.2 Minimizing Least Squares with Constraints

As concluded in section 4.1.1 a solution to (4.5) does not always exist, which raises the question: what should be done when no solution exists? When no solution to (4.5) exist, there will not exist any solution which both satisfies the equality and inequality constraints set on $\boldsymbol{x}$. It is obvious that the inequality constraints always have to be met since the octocopter can only operate in the space defined by the inequality constraints. An approach of finding an alternative solution is to compute a $\boldsymbol{x}^*$ which generates forces and torques as close as possible corresponding to the control input $\boldsymbol{y}$. Therefore, it is

reasonable to minimize the least squared error of $\boldsymbol{Ax} - \boldsymbol{y}$ subject to the inequality constraints. Hence, the following optimization problem is formulated

$$\min_{\boldsymbol{x}} \quad ||\boldsymbol{Ax} - \boldsymbol{y}||_2 \quad \text{s.t} \quad \boldsymbol{Bx} \leq \boldsymbol{c}. \tag{4.13}$$

This problem can numerically be solved by using Lagrange multipliers and by introducing slack variables to get it on the same form as in (4.5). This is not done here, but a numerical solution can be obtained by using MATLAB's function `lsqlin` which minimizes the least squared norm subject to equality and/or inequality constraints. But since the processes of finding an optimal solution to (4.13) is iterative and time consuming, it is determined that it is currently not feasible to implement such a process into the software of the octocopter. This method can be applied when it is possible to run it completely separated from the thread running the control of the octocopter. Therefore, an alternative method is needed when no solution to (4.5) exists, which runs in constant time. Instead, when no solution to (4.5) exists, the control demands are prioritized according to an algorithm which is explained in detail in the next section 4.1.3.

4.1.3 Control Boundaries and Priority Algorithm

From the discussion in sections 4.1.1 and 4.1.2 it is decided to use the right pseudo inverse $\boldsymbol{A}^+ \equiv \boldsymbol{\Gamma}^+$ to convert the motor control input $\boldsymbol{u}$ to angular motors velocities $\boldsymbol{\omega}_M$. When using $\boldsymbol{\Gamma}^+$ it is of importance to keep the desired control forces and torques within the boundaries seen in Figures 4.2 - 4.5, in order to ensure existence of solution. Therefore, control boundaries and a priority algorithm are now derived.

Boundaries

A natural operating point for an octocopter is hovering, which implies $u_z = -mg$. Furthermore, from now on the forces $u_x = u_y = 0$, since translation in $x,\ y$ is going to be accomplished by using the angles ϕ, θ and ψ. To change these angles, the motor torques $u_\phi,\ u_\theta$ and u_ψ are used. Now numerical boundaries on the motor torques are set. These boundaries can be chosen in many ways, it is decided to not be too conservative when doing so. This since the most extreme operating conditions (e.g., maximal roll, pitch and yaw commands simultaneously) will considerably and unnecessarily reduce control limits. From Figures 4.2 - 4.5 and by using the derived $\boldsymbol{\Gamma}^+$ the following boundaries are chosen

$$\begin{aligned} |u_\phi|, |u_\theta| &\leq 4.0 \text{ Nm} \\ |u_\psi| &\leq 1.0 \text{ Nm}. \end{aligned} \tag{4.14}$$

Due to symmetrical reasons, the control boundaries on ϕ and θ are determined to be the same. Furthermore, the boundaries on u_ϕ and u_θ are selected so that both of these control signals can hit their limits but still produce valid motor control signals. By investigating Figure 4.4 it can be seen that the maximal available torque about ψ when hovering is $u_\psi \approx 1.15$ Nm, therefore the limit on u_ψ is reduced some to have a margin and to increase the control available about ϕ and θ. Note that it is not possible to generate $u_\psi = 1$ Nm if u_ϕ and u_θ are at their limits. The maximum torque u_ψ which can be generated when u_ϕ and u_θ are at their limits is only $u_\psi \approx 0.1$ Nm.

Priority Algorithm

As mentioned, the limits in (4.14) can not be hit all at once. In order to handle situations where control demands cannot be achieved, a priority algorithm is used. As mentioned, both u_ϕ and u_θ can hit their limits in any possible way. Furthermore, it is determined that control in ϕ and θ is more crucial than ψ, since changing ψ will not by itself change translation. Hence, u_ϕ and u_θ are priorities higher than u_ψ. The priority algorithm is now described by pseudo code, see Algorithm 1.

Algorithm 1 Priority Algorithm

```
1: function priority_algorithm(u, Γ⁺)
2:     if check_control(u, Γ⁺) then return u
3:     u_1 ← 0
4:     u_2 ← 0
5:     if check_control(u, Γ⁺) then return u
6:     h ← 0.1
7:     n_ψ ← ⌈|u_6|/h⌉
8:     for i ← 1 to n_ψ do
9:         u_6 ← reduce_control(u_6, h)
10:        if check_control(u, Γ⁺) then return u
11:    u_6 ← 0
12:    n_φθ ← ⌈max(|u_4|, |u_5|)/h⌉
13:    for j ← 1 to n_φθ do
14:        u_4 ← reduce_control(u_4, h)
15:        u_5 ← reduce_control(u_5, h)
16:        if check_control(u, Γ⁺) then return u
17:    u_4 ← 0
18:    u_5 ← 0
19:    return u

20: function check_control(u, Γ⁺)
21:    ω_M^2 ← Γ⁺u
22:    if ∀i 200 ≤ ω_{M_i} ≤ 700 then
23:        return true
24:    else
25:        return false

26: function reduce_control(u_k, h)
27:    if |u_k| > h then
28:        u_k ← u_k − sign(u_k)h
29:    else
30:        u_k ← 0
31:    return u_k
```

The main function, priority_algorithm($\boldsymbol{u}$, $\boldsymbol{\Gamma}^+$) first checks if the incoming control $\boldsymbol{u}$ is possible to accomplish by using the helping function check_control($\boldsymbol{u}$, $\boldsymbol{\Gamma}^+$), which simply checks if $\boldsymbol{\omega}_M$ lies within the boundaries. Secondly, it sets $u_x = u_y = 0$ and checks if the control is valid. Thirdly, it decreases u_ψ by the step $h = 0.1$ until zero by using the helping function reduce_control(u_k, h), during each iteration it is checked if the control is valid. Lastly, u_ϕ and u_θ are decreased by h until zero, during each iteration it is checked if the control is valid. Note that it is made sure that $u_\phi = u_\theta = u_\psi = 0$ after each for-loop. If no solution with the current u_z is found, all other elements in $\boldsymbol{u}$ will be zero.

Note, it is here assumed that the limits on $\boldsymbol{u}$ in (4.14) are met, this is checked in the implemented code. Furthermore, the algorithm always runs in finite constant time $\mathcal{O}(9/h) = \mathcal{O}(1)$ since h is fixed and does not dependent on the input data. In the worst case, if $h = 0.1$ the algorithms executes ≈ 600 lines of code before it halts. Hence, in C this time will be neglectable.

4.2 Pilot-based Control

The pilot-based controller is designed in such a way that an active human pilot can operate the octocopter. There are of course many approaches of doing so, but a common way is to control the Euler angles roll ϕ,

pitch θ, angular yaw-rate $\dot{\psi}$ and the motor force in z, F_{M_z}. Since the control in z is quite trivial for the pilot and the estimations of translational movement is not as accurate as rotational, see section 2.1.3, it is determined that no feedback controller in z is needed, only open-loop control.

There are many control techniques which are appropriate for this type of system. It is decided to use a linear control method. Partially since linear control methods are easier to implement and derive than a nonlinear control method and since last year's thesis investigated a PID controller. Therefore, it is decided to use state feedback control. State feedback control or pole-placement is a commonly used linear control method. This method allows for arbitrary pole-placement of a LTI (linear time invariant) system as long as all states are measurable and the system is controllable [25, p. 237]. Since the third state of each subsystem in (3.66) is not measured, an observer is required for the use of full state feedback. For this purpose a reduced observer is used and derived in section 4.2.3. Before that, it is shown that the system (3.64) is both controllable and observable, see section 4.2.1.

When tuning a controller, it is of importance that the control is fast enough. According to [9], the average reaction time, vision to hand, is 267 ms. Therefore, it is concluded that the rise time of the controller should be about this value. A somewhat slower control will not cause any major problems since the pilot has to learn how the system behaves. In means of oscillations, it is decided that the pilot-based controller should not produce large overshoots.

Since the control input is bounded, see section 4.1.3, it is important to produce control inputs within these boundaries. According to these control boundaries, the control input in yaw is going to be considerably smaller than the one in roll and pitch. Also, since translational movement can not be accomplished by only changing yaw, it is determined that the lower boundary on u_ψ is not going to be an issue. The magnitude of the control signal is partially dependent on the demanded reference. Therefore, it is crucial to determine reasonable reference values. It is decided that sufficient maximum roll and pitch references are $\frac{\pi}{8}$ rad and maximum reference in yaw-rate 2 rad/s. These values are going to be of importance when tuning the controllers.

4.2.1 Controllability and Observability

Controllability and observability describe how the states of a system are affected by the inputs and how they affect the outputs. These concepts are of importance when analyzing and for understanding the behavior of the system. The definitions of controllability and observability are given by [25, p. 45]. But roughly, a LTI system is said to be controllable if all states can reach any state in finite time. Furthermore, a system is said to be observable if it lacks unobservable states. Where an unobservable state is a state that cannot be seen in the output. For a LTI system the controllability and observability matrices are here denoted $\boldsymbol{\mathcal{S}}$ respectively $\boldsymbol{\mathcal{O}}$ and must have full rank for the system (3.64) to be controllable and observable. The matrices are given by

$$\boldsymbol{\mathcal{S}}(\boldsymbol{A},\boldsymbol{B}) = \begin{pmatrix} \boldsymbol{B} & \boldsymbol{AB} & \boldsymbol{A}^2\boldsymbol{B} & \cdots & \boldsymbol{A}^{n-1}\boldsymbol{B} \end{pmatrix} \tag{4.15}$$

$$\boldsymbol{\mathcal{O}}(\boldsymbol{A},\boldsymbol{B}) = \begin{pmatrix} \boldsymbol{C} \\ \boldsymbol{CA} \\ \boldsymbol{CA}^2 \\ \vdots \\ \boldsymbol{CA}^{n-1} \end{pmatrix} \tag{4.16}$$

where n is the order of the system. By investigation it is quite clear that the system (3.64) with (3.66) is going to be both controllable and observable, but this is now shown. It is first noted that $n = 3$. The controllability matrix for the system is given by

$$\boldsymbol{\mathcal{S}}(\boldsymbol{A}_i,\boldsymbol{B}_i) = \begin{pmatrix} 0 & 0 & \frac{\boldsymbol{c}_i}{\tau_M} \\ 0 & \frac{\boldsymbol{c}_i}{\tau_M} & -\frac{\boldsymbol{c}_i}{\tau_M^2} \\ \frac{1}{\tau_M} & -\frac{1}{\tau_M^2} & \frac{1}{\tau_M^2} \end{pmatrix} \tag{4.17}$$

which clearly has full rank i.e., 3 if $\boldsymbol{c}_i, \tau_M \neq 0$. So the system is controllable. Now it is shown that the system also is observable by checking the observability matrix

$$\boldsymbol{\mathcal{O}}(\boldsymbol{A}_i, \boldsymbol{C}_i) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \boldsymbol{c}_i \\ 0 & 0 & \boldsymbol{c}_i \\ 0 & 0 & -\frac{\boldsymbol{c}_i}{\tau_M} \end{pmatrix} \tag{4.18}$$

which clearly has full (column) rank i.e., 3 if $\boldsymbol{c}_i, \tau_M \neq 0$.

4.2.2 State Feedback Control

As mentioned, state feed back control allows for arbitrary pole-placement as long as all states are measurable and the system is controllable. In the previous section 4.2.1 it is shown that the system (3.64) is both observable and controllable. But since the last state in (3.64) with (3.66) is not measured, an observer estimating this state is needed. All states will be made available by using a reduced observer, see section 4.2.3. In the single input case the general state feedback control is given by

$$u = -\boldsymbol{L}\tilde{\boldsymbol{x}} \tag{4.19}$$

where the change of variable $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_r$ is introduced to track some reference $\boldsymbol{x}_r$. By applying the controller and variable change to the system (3.64) one gets that

$$\dot{\tilde{\boldsymbol{x}}} = (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{L})\tilde{\boldsymbol{x}} \tag{4.20}$$

which is asymptotically stable if all the eigenvalues of $\boldsymbol{A}-\boldsymbol{B}\boldsymbol{L}$ have negative real part which implies

$$\lim_{t\to\infty} \tilde{\boldsymbol{x}} \to \boldsymbol{0} \Rightarrow \lim_{t\to\infty} \boldsymbol{x} \to \boldsymbol{x}_r. \tag{4.21}$$

Hence, the controller tracks the reference $\boldsymbol{x}_r$. Conditions and appropriate values on $\boldsymbol{L}$ are derived in sections 4.2.4-4.2.5.

To be able to apply integral action for control of ϕ and θ a new integrating state is necessary. Adding an extra integrating state can reduce the effect of modeling errors and disturbances, it can be compared to the I-part in a PID controller. The state-space matrices for $i = 4, 5$ in (3.64) with the added integrating state will thus be given by

$$\boldsymbol{A}_i = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \boldsymbol{c}_i \\ 0 & 0 & 0 & -\frac{1}{\tau_M} \end{pmatrix}, \ \boldsymbol{B}_i = \begin{pmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{\tau_M} \end{pmatrix}, \ \boldsymbol{C}_i = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \ \boldsymbol{D}_i = \boldsymbol{0}. \tag{4.22}$$

Note that the state $\dot{\psi}$ already has the integrating state ψ and therefore can use the previous state-space matrices (3.66). Also, the new system with the above state-space matrices is still controllable and observable, which can be shown by using (4.15)-(4.16).

4.2.3 Reduced Observer

As mentioned, the state feedback controller requires all states to be measurable and the system to be controllable. But since the third state of each subsystem in (3.66) and forth in (4.22) is not measured, an observer is required for the use of full state feedback. Before moving on it is noted that the state space matrices (3.66) and (4.22) are observable, see section 4.2.1. Since only one state of each subsystem

requires estimation a reduced observer is suggested. Consider the system below where the states $\boldsymbol{x}_1 \in \mathbb{R}^{p\times 1}$ are measured and $\boldsymbol{x}_2 \in \mathbb{R}^{q\times 1}$ are not measured with the scalar control input u

$$\begin{pmatrix}\dot{\boldsymbol{x}}_1\\ \dot{\boldsymbol{x}}_2\end{pmatrix} = \begin{pmatrix}\boldsymbol{A}_{11} & \boldsymbol{A}_{12}\\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{pmatrix} + \begin{pmatrix}\boldsymbol{B}_1\\ \boldsymbol{B}_2\end{pmatrix} u \tag{4.23}$$

$$\boldsymbol{y} = \begin{pmatrix}\boldsymbol{I} & \boldsymbol{0}\end{pmatrix}\begin{pmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{pmatrix} = \boldsymbol{x}_1 \tag{4.24}$$

where $\boldsymbol{I} \in \mathbb{R}^{p\times p}$ and $\boldsymbol{0} \in \mathbb{R}^{p\times q}$. Let the estimate of $\boldsymbol{x}_2$ be given by $\hat{\boldsymbol{x}}_2$. The approach used for finding a reduced observer is the same one used for a full state Luenberger observer. Since only the states $\boldsymbol{x}_2$ are in need of estimation one gets that

$$\dot{\hat{\boldsymbol{x}}}_2 = \boldsymbol{A}_{21}\boldsymbol{x}_1 + \boldsymbol{A}_{22}\hat{\boldsymbol{x}}_2 + \boldsymbol{B}_2 u + \boldsymbol{d} \tag{4.25}$$

where $\boldsymbol{d}$ is called a driver function, driving the estimated $\hat{\boldsymbol{x}}_2$ towards $\boldsymbol{x}_2$. The matrix $\boldsymbol{K} \in \mathbb{R}^{q\times p}$ should be selected such that the $\hat{\boldsymbol{x}}_2 \to \boldsymbol{x}_2$ fast enough. This means that $\boldsymbol{d} \to \boldsymbol{0}$ as $\hat{\boldsymbol{x}}_2 \to \boldsymbol{x}_2$. An appropriate candidate for $\boldsymbol{d}$ can be found by using the equation for $\dot{\boldsymbol{x}}_1$ (4.24), one gets that

$$\boldsymbol{d} = \boldsymbol{K}(\dot{\boldsymbol{x}}_1 - \boldsymbol{A}_{11}\boldsymbol{x}_1 - \boldsymbol{A}_{12}\hat{\boldsymbol{x}}_2 - \boldsymbol{B}_1 u) \tag{4.26}$$

which equals zero if $\boldsymbol{x}_2 = \hat{\boldsymbol{x}}_2$. It is now shown that the proposed driver assures that the estimate error $\boldsymbol{e}_2 = \boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2$ converges to zero. Taking the time derivative of $\boldsymbol{e}_2$ gives

$$\begin{aligned}
\dot{\boldsymbol{e}}_2 &= \dot{\boldsymbol{x}}_2 - \dot{\hat{\boldsymbol{x}}}_2\\
&= \boldsymbol{A}_{21}\boldsymbol{x}_1 + \boldsymbol{A}_{22}\boldsymbol{x}_2 + \boldsymbol{B}_2 u - \boldsymbol{A}_{21}\boldsymbol{x}_1 - \boldsymbol{A}_{22}\hat{\boldsymbol{x}}_2 - \boldsymbol{B}_2 u - \boldsymbol{d}\\
&= \boldsymbol{A}_{22}(\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2) - \boldsymbol{K}(\dot{\boldsymbol{x}}_1 - \boldsymbol{A}_{11}\boldsymbol{x}_1 - \boldsymbol{A}_{12}\hat{\boldsymbol{x}}_2 - \boldsymbol{B}_1 u)\\
&= \boldsymbol{A}_{22}(\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2) - \boldsymbol{K}(\boldsymbol{A}_{11}\boldsymbol{x}_1 + \boldsymbol{A}_{12}\boldsymbol{x}_2 + \boldsymbol{B}_1 u - \boldsymbol{A}_{11}\boldsymbol{x}_1 - \boldsymbol{A}_{12}\hat{\boldsymbol{x}}_2 - \boldsymbol{B}_1 u)\\
&= \boldsymbol{A}_{22}(\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2) - \boldsymbol{K}\boldsymbol{A}_{12}(\boldsymbol{x}_2 - \hat{\boldsymbol{x}}_2)\\
&= (\boldsymbol{A}_{22} - \boldsymbol{K}\boldsymbol{A}_{12})\boldsymbol{e}_2
\end{aligned}$$

which has the known solution $\boldsymbol{e}_2 = e^{\boldsymbol{A}_{22}-\boldsymbol{K}\boldsymbol{A}_{12}}\boldsymbol{e}_2(t_0)$ and is asymptotically stable if all eigenvalues to $\boldsymbol{A}_{22} - \boldsymbol{K}\boldsymbol{A}_{12}$ have real negative part. The differentiation $\dot{\boldsymbol{x}}_1$ in (4.26) can amplify measurement noise, this can be avoided by applying the change of variable $\boldsymbol{z} = \hat{\boldsymbol{x}}_2 - \boldsymbol{K}\boldsymbol{x}_1$ which with (4.25) and (4.26) yields the final reduced observer

$$\dot{\boldsymbol{z}} = \dot{\hat{\boldsymbol{x}}}_2 - \boldsymbol{K}\dot{\boldsymbol{x}}_1 \tag{4.27}$$

$$= (\boldsymbol{A}_{21} - \boldsymbol{K}\boldsymbol{A}_{11})\boldsymbol{x}_1 + (\boldsymbol{A}_{22} - \boldsymbol{K}\boldsymbol{A}_{12})(\boldsymbol{z} + \boldsymbol{K}\boldsymbol{x}_1) + (\boldsymbol{B}_2 - \boldsymbol{K}\boldsymbol{B}_1)u. \tag{4.28}$$

For the LTI system with the state-space matrices (4.22) the matrices in (4.24) are given by

$$\begin{aligned}
&\boldsymbol{A}_{11} = \begin{pmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & 0\end{pmatrix},\ \boldsymbol{A}_{12} = \begin{pmatrix}0\\ 0\\ \boldsymbol{c}_i\end{pmatrix},\ \boldsymbol{A}_{21} = \begin{pmatrix}0 & 0 & 0\end{pmatrix},\ \boldsymbol{A}_{22} = -\tfrac{1}{\tau_M},\\
&\boldsymbol{B}_1 = \begin{pmatrix}0\\ 0\\ 0\end{pmatrix},\ \boldsymbol{B}_2 = \tfrac{1}{\tau_M},\ \boldsymbol{K} = \begin{pmatrix}k_1 & k_2 & k_3\end{pmatrix}.
\end{aligned} \tag{4.29}$$

By using these one easily gets the eigenvalue of $\boldsymbol{A}_{22} - \boldsymbol{K}\boldsymbol{A}_{12}$ to be $\lambda = -\boldsymbol{c}_i k_3 - \frac{1}{\tau_M}$ and since stability requires $\mathrm{Re}(\lambda) < 0$ one gets that $\mathrm{Re}(k_3) > -\frac{1}{\tau_M \boldsymbol{c}_i}$. Since the parameters k_1 and k_2 do not affect the stability of the observer these can be arbitrarily set. For simplicity $k_1 = k_2 = 0$ which together with (4.28) and (4.29) gives the reduced observer

$$\dot{z} = -(\tfrac{1}{\tau_M} + k\boldsymbol{c}_i)(z + kx_3) + \tfrac{1}{\tau_M}u, \quad z(0) = z_0 \tag{4.30}$$

where $k \equiv k_3$ and the estimated state is given by $\hat{x}_4 = z + kx_3$. Because the estimated state $\hat{x}_4$ equals the delayed input u, section 3.5, the initial condition z_0 is chosen such that $\hat{x}_{04} = \frac{u}{2}$, which gives $z_0 = \frac{u}{2} - kx_{03}$. Note that the equivalent matrices for the system with (3.66) will give arise to the same condition and observer. The value of k is determined after the state feedback controller has been designed since the observer should be faster than the system.

4.2.4 Roll and Pitch Control

As mentioned, the states roll and pitch have an extra integrated state and can be represented by the state space equations given by (4.22). Since the goal is to track some reference state ϕ_r and θ_r, the following variable change is made

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_r = \begin{pmatrix} \int (x_2 - x_{2r})dt \\ x_2 - x_{2r} \\ x_3 \\ x_4 \end{pmatrix} \tag{4.31}$$

where e.g., $x_2 = \phi$, $x_3 = \dot{\phi}$ and $x_4 = M_{M_\phi}$. Note that this variable change does not affect the state space matrices (4.22). In order to determine conditions on $\boldsymbol{L} = (l_1 \ l_2 \ l_3 \ l_4)$, the characteristic polynomial is computed

$$|\lambda \boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{B}\boldsymbol{L})| = \tfrac{1}{\tau_M}(\tau_M \lambda^4 + (1 + l_4)\lambda^3 + l_3 \boldsymbol{c}_i \lambda^2 + l_2 \boldsymbol{c}_i \lambda + l_1 \boldsymbol{c}_i) \tag{4.32}$$

where $\tau_M, \ \boldsymbol{c}_i > 0$. By then using Routh-Hurwitz stability criterion for a fourth-order polynomial one gets the following conditions on $\boldsymbol{L}$

$$\begin{aligned} l_1, \ l_2, \ l_3 > 0, \ l_4 > -1 \\ \frac{(1 + l_4) l_3}{l_2} > \tau_M \\ \frac{(1 + l_4)(l_2 l_3 \boldsymbol{c}_i - (1 + l_4) l_1)}{l_2^2 \boldsymbol{c}_i} > \tau_M \end{aligned} \tag{4.33}$$

which guarantee stable roots to (4.32). Before moving on it is important to note that the conditions (4.33) guarantee stability for the full linearized system (4.22).

Now appropriate values for $\boldsymbol{L}$ are chosen such that the controller behaves in a desirable manner as discussed in the beginning of section 4.2. There are numerous ways of selecting $\boldsymbol{L}$. One can directly study the poles of the system or investigate how changing $\boldsymbol{L}$ affects the system. It is important to understand that there are infinitely many $\boldsymbol{L}$ which provide stability. Therefore, $\boldsymbol{L}$ should be selected such that the system behaves in an desired way which is quite subjective, especially when it comes to a pilot-based controller. Besides stability, the control input u is bounded. This means that $\boldsymbol{L}$ should be chosen such that u lies (under normal operating conditions) within the limits defined in section 4.1.3. It is determined that u should be kept within its defined boundaries when a step in the reference signal from the lower boundary to the upper boundary is applied e.g., $\phi \in [-\frac{\pi}{8}, \ \frac{\pi}{8}]$. This will in a sense cover the most demanding control signals under normal operating conditions.

Furthermore, the parameter for the reduced observer has to be set, during the tuning of the state feedback controller $k = 2$ that gives a pole at $\lambda \approx -21$ which initially should provide a sufficiently fast observer. Before determining $\boldsymbol{L}$, it is important to notice that the observer initially requires some time before it has settled in, this since the initial condition given in (4.30) does not correspond the true initial state. This is now illustrated by simulating the LTI system in roll with the state feedback $\boldsymbol{L} = (2, \ 5, \ 2, \ 1)$ which are reasonable values (compare to Table 4.1), see Figure 4.6.

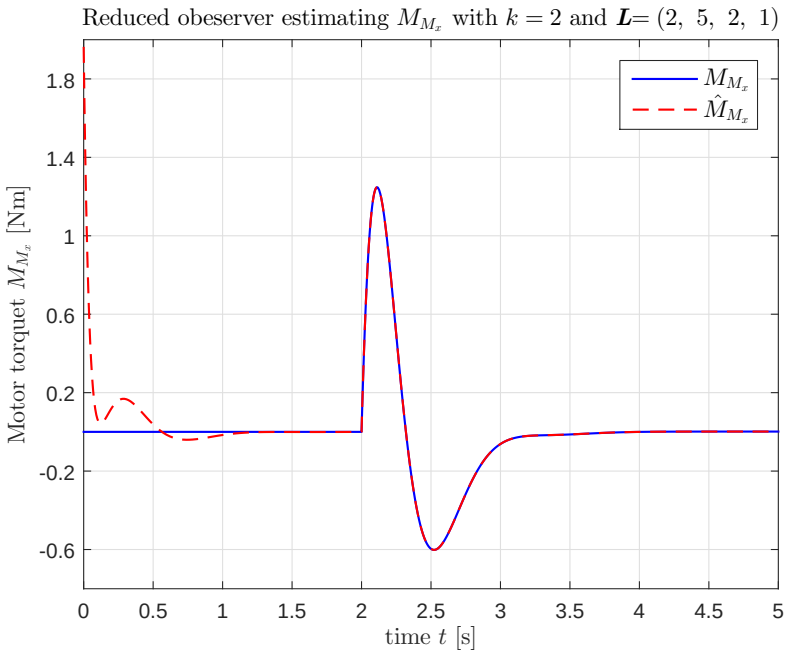

Figure 4.6: Reduced observer estimating M_{M_ϕ} with $k = 2$ and $\boldsymbol{L} = (2,\ 5,\ 2,\ 1)$. Step in roll (at $t = 2$ s) with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From Figure 4.6 it can be seen that it takes about 1 s before the observer has settled in and estimates the state M_{M_ϕ} well. Therefore, in order to not be too conservative, all simulations following in this section will give the observer the same initial condition as the true state.

It is now described and illustrated how $\boldsymbol{L}$ is selected for achieving desired control in roll. It is chosen to initially not have any integrating action. Integrating action is good when rejecting disturbances and compensating for modeling errors but will also increase overshoot and settling time, this will later be compensated for by filtering the reference signal. But for now $l_2 = 0$ and is added and tuned after the tuning of l_2, l_3 and l_4 is done. By letting $l_1 = 0$, stability of the integrating state is obviously not achieved. If one does not need to stabilize the integrating state the last stability criterion in (4.33) can be ignored, this can be shown by using Routh-Hurwitz stability criterion for a third-order polynomial, compare also to (4.37). Therefore, initially $\boldsymbol{L} = (0,\ 1,\ 1,\ 1)$. Each parameter in $\boldsymbol{L}$ is then varied to study how the system is affected e.g., rise time, settling time, overshoot, undershoot. Also, the generated control signal, estimated state by the observer and closed loop systems poles are investigated.

Firstly, the parameter l_2 is varied, which can be seen as the proportional part of the controller. This since, applying a step in the reference ϕ_r when the octocopter is stationary in rotation i.e., $\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$, the magnitude of the control input u_ϕ is mainly affected by the proportional part. Hence, l_2 should be selected such that $|u_\phi| \leq 4$ Nm, see section 4.1.3. As mentioned, the LTI system is simulated for the maximum allowed positive step $\phi_r = \frac{\pi}{8}$ in the reference where the system initially has full negative roll $\phi_0 = -\frac{\pi}{8}$, see Figure 4.7.

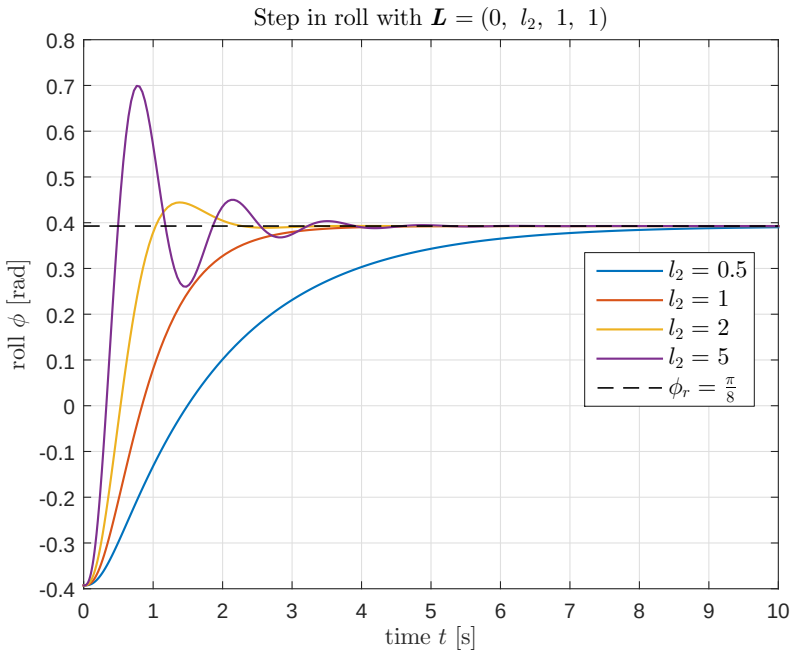

Figure 4.7: State feedback with $\boldsymbol{L} = (0,\ l_2,\ 1,\ 1)$. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From Figure 4.7 it can bee seen how the parameter l_2 affects the step response. Increasing l_2 will increase rise time but also the possibility for oscillations. Again, note that l_2 can not be tuned too large since it will produce invalid control signals. Now the LTI system with state feedback is simulated by varying l_3, which can be seen as the derivative part of the controller, see Figure 4.8.

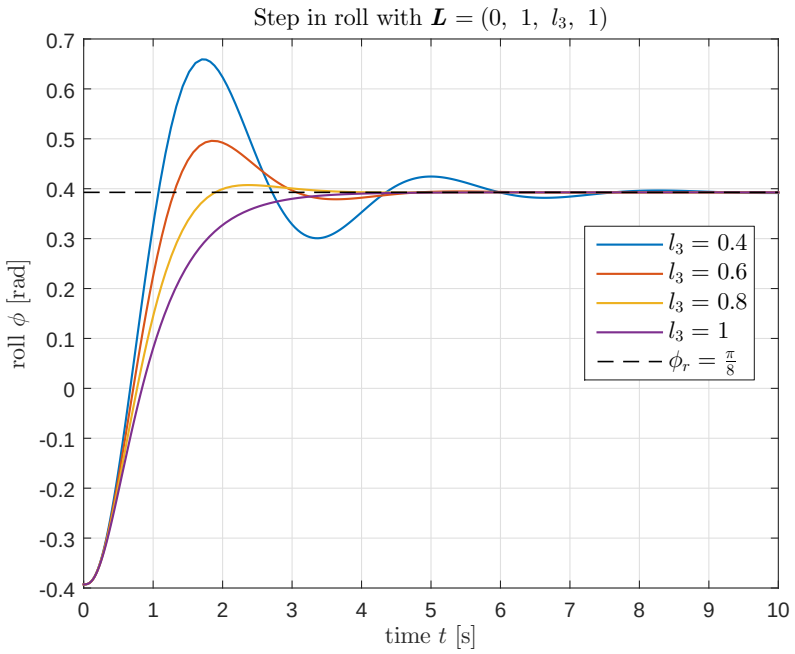

Figure 4.8: State feedback with $\boldsymbol{L} = (0,\ 1,\ l_3,\ 1)$. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From Figure 4.8 it can bee seen how the parameter l_3 affects the step response. Increasing l_3 will reduce oscillations but increase the rise time. The parameter l_3 is going to be tuned such that the system with the chosen l_2 and without any integrating part does not produce any overshoot i.e oscillations. Furthermore, l_3 should not be selected too large since the state $\dot{\phi}$ might experience noise. Now the LTI system with state feedback is simulated by varying l_4, which can be seen as the second derivative (motor torque) part of the controller, see Figure 4.9.

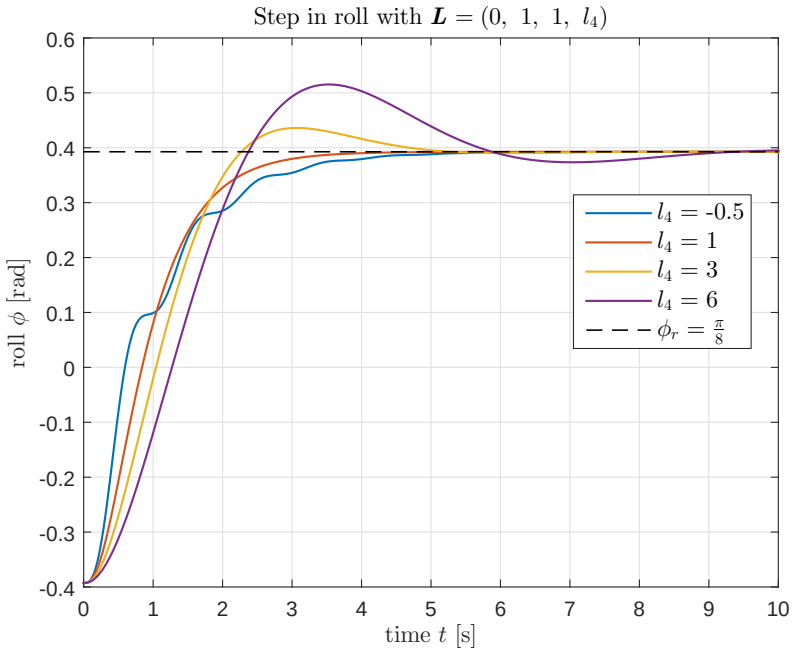

Figure 4.9: State feedback with $\boldsymbol{L} = (0,\ 1,\ 1,\ l_4)$. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From Figure 4.9 it can bee seen how the parameter l_4 affects the step response. Increasing l_4 will to some point increase the rise time, which after this point starts to decrease again. Increasing l_4 will also increase the possibility for overshoot. Furthermore, it can be seen that there exists a crucial point where the step response experiences oscillations during the initial rise, see $l_4 = -0.5$. This is due to the fact that there exists a significant large time constant in the motor dynamics, see section 3.5. The parameter l_4 is selected after l_2, l_3 and such that the rise time is kept low but the oscillations during the initial rise are removed. The final parameter tuned is l_4, which determined how much the integrating state is affecting the system. The LTI system with state feedback is simulated by varying l_1, see Figure 4.10.

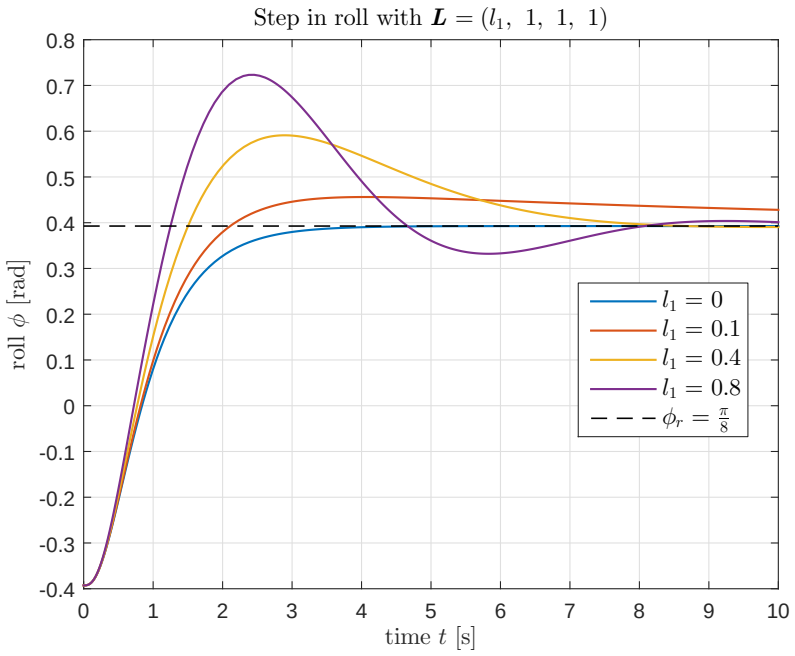

Figure 4.10: State feedback with $\boldsymbol{L} = (l_1,\ 1,\ 1,\ 1)$. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From Figure 4.9 it can bee seen how the parameter l_1 affects the step response. Increasing l_1 will increase rise time but also overshoot. As mentioned, having an integrating part is good when rejecting disturbances and handling modeling errors. Furthermore, it is decided that the integrating part should not be too large and should be seen as the parameter fine-tuning the step response. It is therefore decided to have an integrating part which corresponds to $\approx 50\%$ of the proportional part i.e. $l_1 \approx \frac{l_2}{2}$. So when $l_2 = 1$ this corresponds to $l_1 \approx 0.4$. From Figure 4.9 one can see that this corresponds in a quite large overshoot which decays over a longer period. This is due to the fact that the step in reference will build up an integration part before the response reached its final reference value. This effect can be reduced by filtering the reference value used in the integrating state, see (4.31). It is decided to implement a first order low-pass filter, given by

$$X_{R_F}(s) = G_{IF}(s)X_R(s) = \frac{1}{\tau_{IF}s + 1}X_R(s) \tag{4.34}$$

where τ_{IF} is the filter time constant and is tuned for appropriate behavior. Note that only the reference used in the integrating state is filtered. The LTI system with state feedback and using the integrating part is now simulated for different values on τ_{IF}, see Figure 4.11.

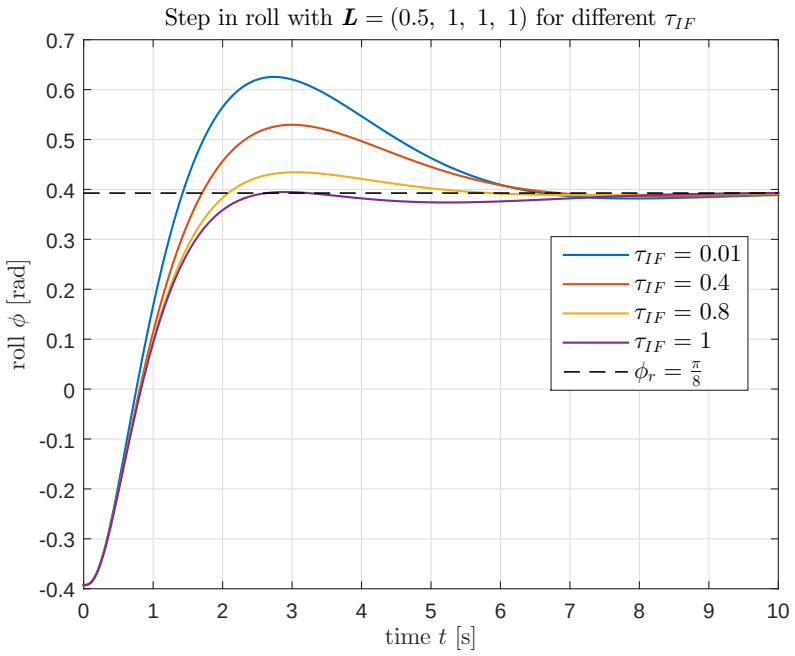

Figure 4.11: State feedback with $\boldsymbol{L} = (0.5,\ 1,\ 1,\ 1)$ for different τ_{IF}. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From Figure 4.11 it can bee seen how the parameter τ_{IF} affects the step response. Increasing τ_{IF} will decrease the rise time but also decrease the overshoot. Furthermore, it can be seen that larger values on τ_{IF} will result in undershoot. The parameter τ_{IF} is selected such that no (or very little) undershoot is present. The above explained procedure for tuning all the mentioned parameters is followed and the final values are given in Table 4.1.

l_1	l_2	l_3	l_4	τ_{IF}	k
2.0	5.0	1.9	1.0	0.37	2.0

Table 4.1: Table showing the final values parameters of the state feedback controller, low-pass filter and reduced observer, in roll.

The final system is simulated when applying the previous used step response, see Figure 4.12.

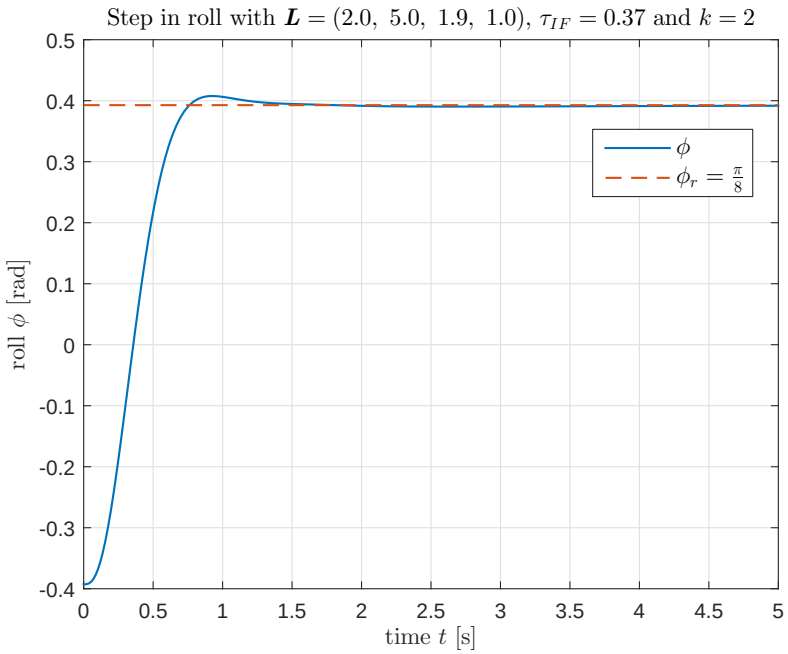

Figure 4.12: State feedback with $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, $\tau_{IF} = 0.37$ and $k = 2$. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From the above Figure it can be seen that the reference value is well tracked. By using MATLAB properties such as rise time (10-90 %), settling time (2 % error), overshoot and undershoot are computed, see Table 4.2.

Rise time	0.43 s
Settling time	0.71 s
Overshoot	1.9 %
Undershoot	0.30 %

Table 4.2: Table showing rise time, settling time, overshoot and undershoot for the chosen state feedback controller in roll, corresponding to Figure 4.12.

According to the beginning of this section, the pilot-based controller should be fast and allows for a small overshoot, in this case only 1.9 %. From Table 4.2 it can be seen that the rise time and settling time are not far away from the previously mentioned normal reaction time of a human which is about 300 ms. The undershoot present is considered neglectable. Next the control signal is shown, see Figure 4.13.

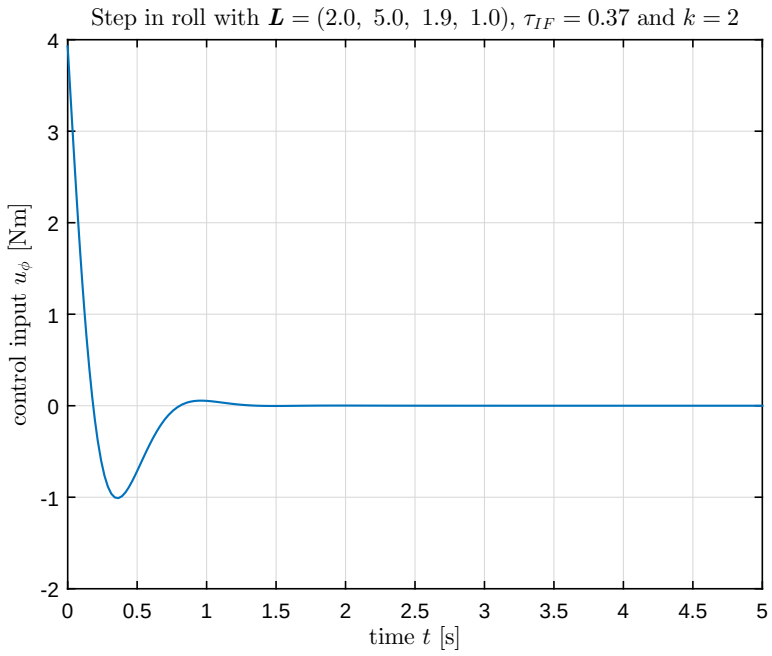

Figure 4.13: Control signal u_ϕ with state feedback $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, $\tau_{IF} = 0.37$ and $k = 2$. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From the above Figure it can be seen that the control signal is such that $|u_\phi| \leq 4$ Nm, hence the boundary condition is satisfied. Finally, the estimated state $\hat{M}_{M_\phi}$ from the reduced observer is investigated, see Figure 4.14.

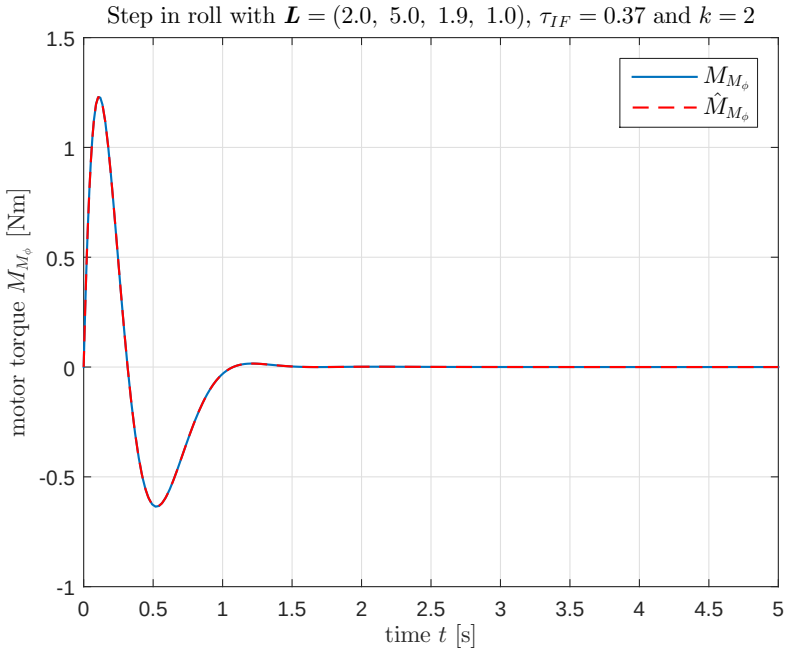

Figure 4.14: State estimation of M_{M_ϕ} with state feedback $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, $\tau_{IF} = 0.37$ and $k = 2$. Step in roll with initial condition $\phi_0 = -\frac{\pi}{8}$ and reference $\phi_r = \frac{\pi}{8}$.

From the above Figure it can be seen that the state is well estimated by using the reduced observer where $k = 2$. It is important to note that the initial condition to the observer is the same as the true state. Furthermore, no disturbances or modeling errors are present in these simulations. It can be shown that the determined $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$ provides asymptotic stability by using the derived condition in (4.33), furthermore the eigenvalues closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in pitch are computed and given in Table 4.3.

λ_1	λ_2	λ_3	λ_4
$-4.49 + 5.72i$	$-4.49 - 5.72i$	-3.87	-0.483

Table 4.3: Table showing eigenvalues of the closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in roll where $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, using three significant figures.

From Table 4.3 one can see that all eigenvalues have real negative parts i.e., the system is asymptotically stable. Furthermore, the two first poles have imaginary parts. By using the observer parameter $k = 2$ one gets that the observer pole is about 3 times larger than the largest pole of the closed-looped system.

By using the same procedure as above the mentioned parameters are selected to derive control of the pitch angle θ. The values found are given in Table 4.4.

l_1	l_2	l_3	l_4	τ_{IF}	k
2.0	5.0	1.9	1.0	0.37	2.0

Table 4.4: Table showing the final values parameters of the state feedback controller, low-pass filter and reduced observer, in pitch.

As one can see from the above table, the exact same values are used as for roll control. The only difference between roll and pitch is the moment of inertia component which according to section 2.2.2 only differs with ≈ 4 %. The step response of the LTI system with the state feedback controller is once again illustrated, see Figure 4.15.

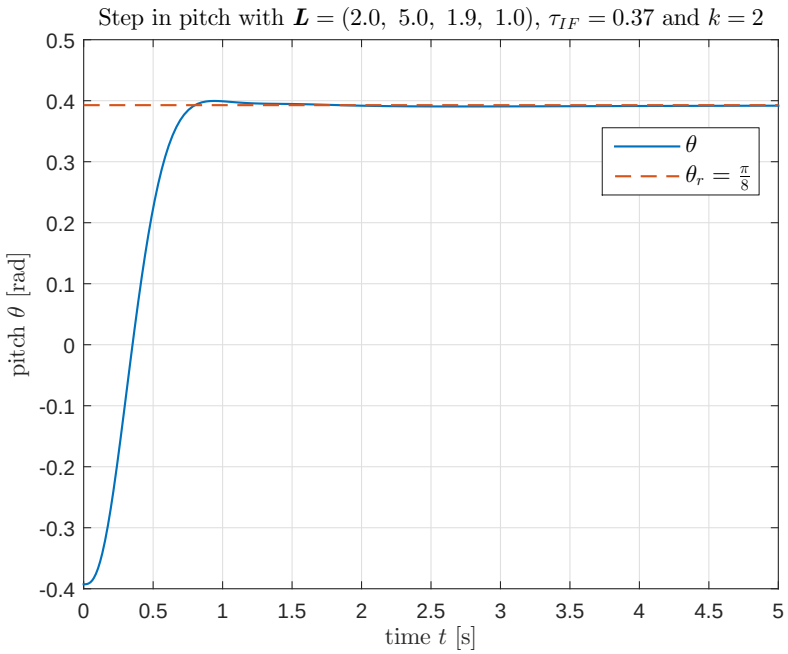

Figure 4.15: State feedback with $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, $\tau_{IF} = 0.37$ and $k = 2$. Step in pitch with initial condition $\theta_0 = -\frac{\pi}{8}$ and reference $\theta_r = \frac{\pi}{8}$.

From the above Figure it can be seen that the reference value is well tracked. Once again, properties such as rise time, settling time, overshoot and undershoot are computed, see Table 4.5.

Rise time	0.44 s
Settling time	0.72 s
Overshoot	0.88 %
Undershoot	0.27 %

Table 4.5: Table showing rise time, settling time, overshoot and undershoot for the chosen state feedback controller in pitch, corresponding to Figure 4.15.

By comparing Table 4.5 to the one for roll control Table 4.2 it can be seen that there is no great difference. By using the same values as for roll control the rise time and settling time are almost the same. The overshoot and undershoot have slightly decreased. Next the control signal is shown, see Figure 4.16.

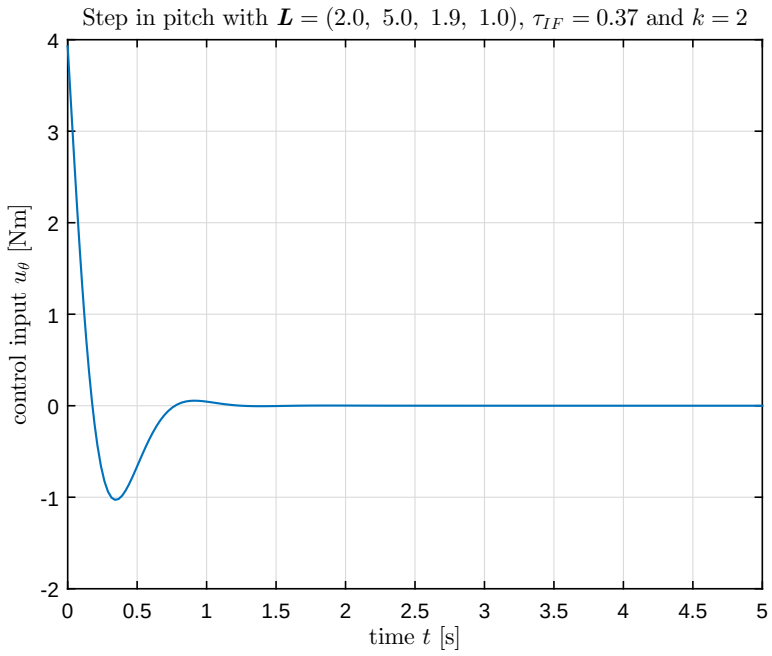

Figure 4.16: Control signal u_θ with state feedback $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, $\tau_{IF} = 0.37$ and $k = 2$. Step in pitch with initial condition $\theta_0 = -\frac{\pi}{8}$ and reference $\theta_r = \frac{\pi}{8}$.

Again, from the above Figure it can be seen that the control signal is such that $|u_\theta| \leq 4$ Nm, hence the boundary condition is satisfied. Finally, the estimated state $\hat{M}_{M_\theta}$ from the reduced observer is investigated, see Figure 4.17.

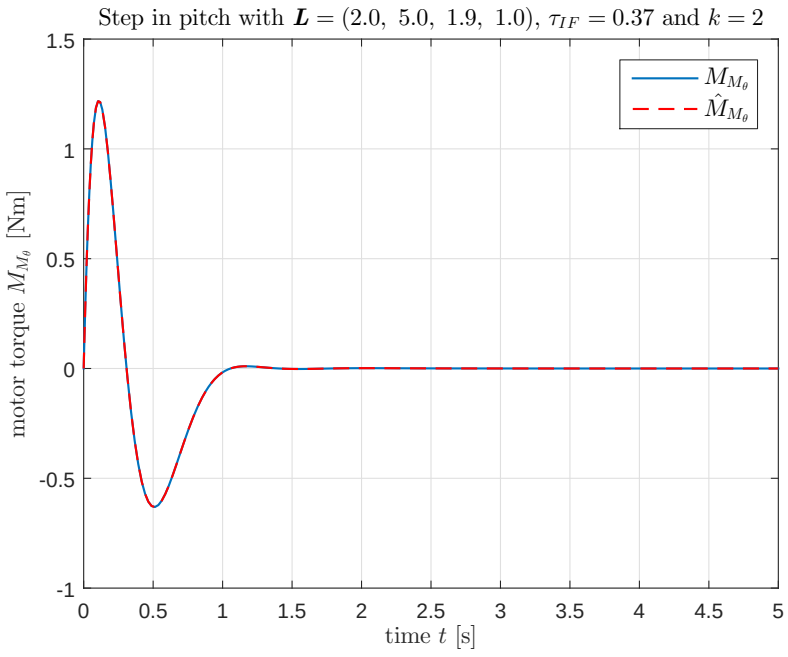

Figure 4.17: State estimation of M_{M_θ} with state feedback $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, $\tau_{IF} = 0.37$ and $k = 2$. Step in pitch with initial condition $\theta_0 = -\frac{\pi}{8}$ and reference $\theta_r = \frac{\pi}{8}$.

From the above Figure it can be seen that the state in pitch also is well estimated by using the reduced observer where $k = 2$. The eigenvalues closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in pitch are computed and given in Table 4.6.

λ_1	λ_2	λ_3	λ_4
$-4.59 + 6.10i$	$-4.59 - 6.10i$	-3.67	-0.483

Table 4.6: Table showing eigenvalues of the closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in pitch where $\boldsymbol{L} = (2.0,\ 5.0,\ 1.9,\ 1.0)$, using three significant figures.

From Table 4.6 one can see that all eigenvalues have real negative parts i.e., the system is asymptotically stable. The poles do not differ much from the poles for the closed-looped system in roll, hence the same reasoning about the observer can be made here.

4.2.5 Yaw-rate Control

Since control in yaw-rate is desired, no extra integrating state for the system has to be added, only a change of variable is required. The following change of variable is made

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \boldsymbol{x}_r = \begin{pmatrix} \int (x_2 - x_{2r})dt \\ x_2 - x_{2r} \\ x_3 \end{pmatrix} \tag{4.35}$$

where $x_2 = \dot{\psi}$ and $x_3 = M_{M_\psi}$. Since now $\tilde{\boldsymbol{x}} \in \mathbb{R}^{3\times 1}$, $\boldsymbol{L} = (l_1\ l_2\ l_3)$ and therefore the characteristic polynomial is reduced to

$$|\lambda \boldsymbol{I} - (\boldsymbol{A} - \boldsymbol{BL})| = \tfrac{1}{\tau_M}(\tau_M \lambda^3 + (1 + l_3)\lambda^2 + l_2 \boldsymbol{c}_i \lambda + l_1 \boldsymbol{c}_i). \tag{4.36}$$

By using Routh-Hurwitz stability criterion for a third-order polynomial one gets the following conditions on $\boldsymbol{L}$

$$\begin{gathered} l_1,\ l_2 > 0,\ l_3 > -1 \\ \frac{(1 + l_3)l_2}{l_1} > \tau_M \end{gathered} \tag{4.37}$$

which guarantee stable roots to (4.36). Now the same procedure for selecting $\boldsymbol{L}$ as for roll and pitch is followed. The found values are given in Table 4.7.

l_1	l_2	l_3	τ_{IF}	k
0.12	0.20	-0.30	0.60	1.0

Table 4.7: Table showing the final values parameters of the state feedback controller, low-pass filter and reduced observer, in yaw-rate.

From Table 4.7 one can see that the parameters are smaller in size compared to the ones for controlling roll and pitch. This since the available control input in yaw u_ψ is considerably less, see section 4.1.3. Furthermore, the determined $\boldsymbol{L}$ provides stability according to (4.37). Also, note that now $k = 1$, which provides a sufficiently fast observer and gives $\lambda \approx 11$, compare to Table 4.9. By using the determined state feedback, low-pass filter and observer, the step response in Figure 4.18 is acquired.

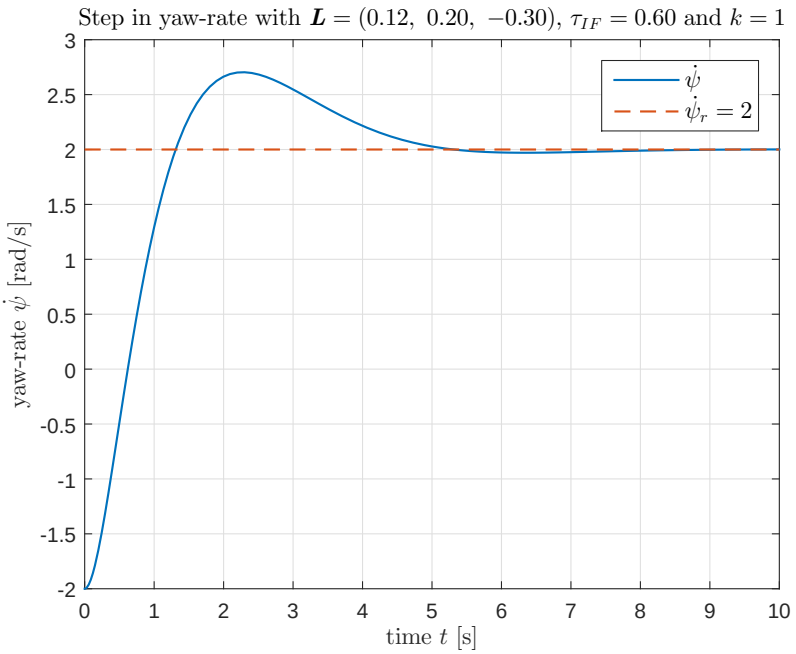

Figure 4.18: State feedback with $\boldsymbol{L} = (0.12,\ 0.20,\ -0.30)$, $\tau_{IF} = 0.60$ and $k = 1$. Step in yaw-rate with initial condition $\dot{\psi}_0 = -2$ and reference $\dot{\psi}_r = 2$.

From the above Figure one can directly note the larger overshoot compared to the previous step responses in roll and pitch. It is determined to have larger overshoot when controlling the yaw-rate because of modeling errors and disturbances due to e.g., aerodynamic effects. Aerodynamic effects are much more prominent when controlling rate rather than position, since e.g., air drag is acting on the rotating octocopter which would generate a counter torque, see section 3.4.2. Hence, this overshoot is going to be reduced when simulating the full nonlinear system. The rise time, settling time, overshoot and undershoot are computed and given in Table 4.8.

Rise time	0.91 s
Settling time	4.6 s
Overshoot	18 %
Undershoot	0.74 %

Table 4.8: Table showing rise time, settling time, overshoot and undershoot for the chosen state feedback controller in yaw-rate, corresponding to Figure 4.18.

From the table it is clear that the performance is not as good as the controller in roll and pitch, which is mainly due to the limited control input. But as mentioned in the beginning of section 4.2 fast response in yaw-rate is not as crucial as in roll and pitch. The undershoot is also more prominent compared to the previous controls but still relatively small. Now the control signal for a step in yaw-rate is investigated, see Figure 4.19.

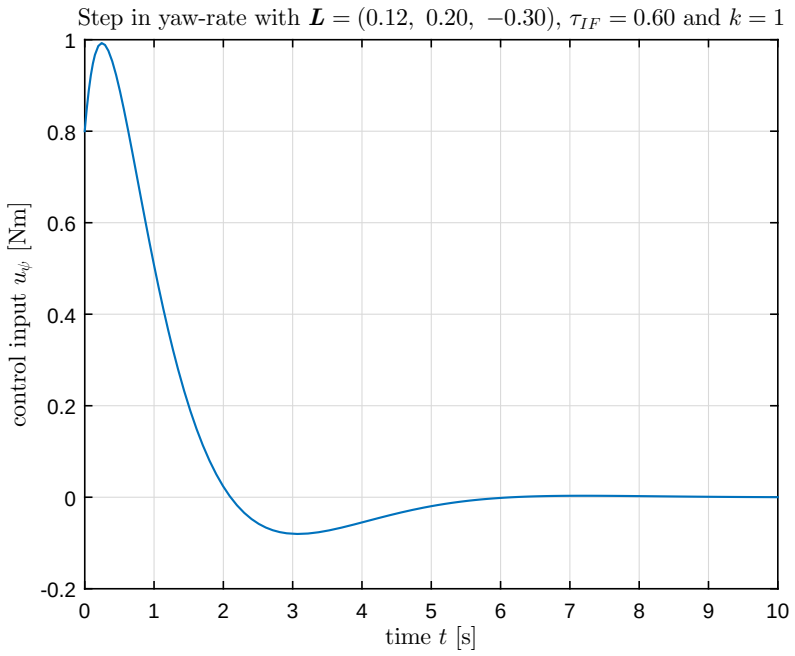

Figure 4.19: Control signal u_ψ with state feedback $\boldsymbol{L} = (0.12,\ 0.20,\ -0.30)$, $\tau_{IF} = 0.60$ and $k = 1$. Step in yaw-rate with initial condition $\dot{\psi}_0 = -2$ and reference $\dot{\psi}_r = 2$.

From Figure 4.19 it can be seen that $|u_\psi| \leq 1.0$ Nm, thus the boundary on the control signal is satisfied. Also, note the behavior of u_ψ in the beginning, this is due to the fact that $l_3 < 0$. Finally, the state estimation from the reduced observer is examined, see Figure 4.20.

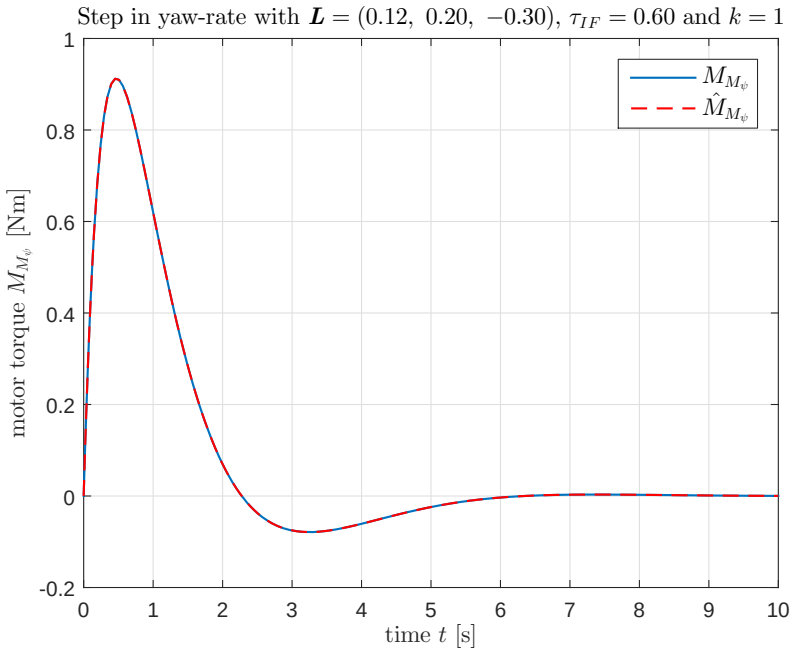

Figure 4.20: State estimation of M_{M_ψ} with state feedback $\boldsymbol{L} = (0.12,\ 0.20,\ -0.30)$, $\tau_{IF} = 0.60$ and $k = 1$. Step in yaw-rate with initial condition $\dot{\psi}_0 = -2$ and reference $\dot{\psi}_r = 2$.

From the Figure it becomes clear that the state is well estimated by letting $k = 1$. The eigenvalues closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in yaw-rate are computed and given in Table 4.9.

λ_1	λ_2	λ_3
$-0.780 + 0.743i$	$-0.780 - 0.743i$	-3.11

Table 4.9: Table showing eigenvalues of the closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in yaw-rate where $\boldsymbol{L} = (0.12,\ 0.20,\ -0.30)$, using three significant figures.

From the table it can be seen that the poles proved asymptotic stability and once again the two first poles have imaginary parts. By using the observer parameter $k = 1$ one gets that the observer pole is about 4 times larger than the largest pole of the closed-looped system.

4.3 Attitude Control

In this section an attitude controller is derived. The attitude controller controls all Euler angles roll ϕ, pitch θ and yaw ψ. In this thesis the attitude controller is going to be used when controlling the position of the octocopter, see section 4.4. The desired Euler angles are going to be computed by the position controller and are there seen as control inputs. To keep things simple, it is decided to use the same control method as for controlling roll and pitch in section 4.2. Therefore, only a controller for yaw is required. Hence, here the state-space matrices are given by 4.22 where now $i = 6$. So all analysis done in section 4.2 is applicable here.

Before moving on, it is important to note that the control input u_ψ is bounded and not as large as for roll and pitch, see section 4.1.3. As in the previous section, the controller should be designed such that for a reasonable step, the controller does not hit the defined control boundary. It is decided that the controller should be able to go from $\psi_0 = 0$ to $\psi_r = \pi$ without hitting any boundaries. Also, the rise and settling times are expected to be significantly larger than the ones obtained for the controllers in roll and pitch. In terms of oscillations, they should be kept small since the attitude controller is going to be given direct control inputs from the position controller, see section 4.4.

4.3.1 Yaw Control

By using the above requirements on yaw and the same method as in the previous section 4.2 the following controller parameters are found, see Table 4.10.

l_1	l_2	l_3	l_4	τ_{IF}	k
0.10	0.30	0.50	-0.20	1.4	1.0

Table 4.10: Table showing the final values parameters of the state feedback controller, low-pass filter and reduced observer, in yaw.

From the above table it can be seen that the size of the values are once again small compared to Tables 4.1 and 4.4, which is due to the bounds on u_ψ. Furthermore, the determined $\boldsymbol{L}$ provides stability according to (4.37), even with $l_3 < 0$. Also, note that now $k = 1$, which provides a sufficiently fast observer and gives $\lambda \approx 11$, compare to Table 4.12. By using the determined state feedback, low-pass filter and observer, the step response in Figure 4.21 is acquired.

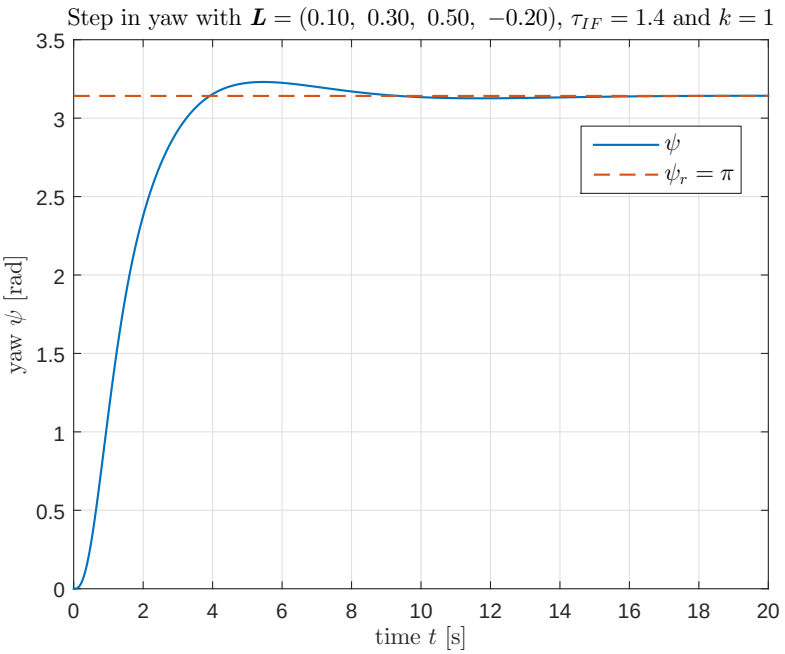

Figure 4.21: State feedback with $\boldsymbol{L} = (0.10,\ 0.30,\ 0.50,\ -0.20)$, $\tau_{IF} = 1.4$ and $k = 1$. Step in yaw with initial condition $\psi_0 = 0$ and reference $\psi_r = \pi$.

From Figure 4.21 it can be seen that the reference value is followed well. The step response is relatively slow compared to control of roll and pitch, see Figures 4.13 and 4.16. Some overshoot and undershoot can be observed. Once again, properties such as rise time, settling time, overshoot and undershoot are computed, see Table 4.11.

Rise time	2.24 s
Settling time	6.8 s
Overshoot	2.8 %
Undershoot	0.48 %

Table 4.11: Table showing rise time, settling time, overshoot and undershoot for the chosen state feedback controller in yaw, corresponding to Figure 4.21.

From the table it is clear that the performance is not as good as the controller in roll and pitch, which mainly is due to the limited control input. But as previously explained, translational movement cannot be accomplished by only changing the yaw angle, hence this control is not as crucial and therefore a slower controller can be allowed. The overshoot and undershoot are within the same region as for the roll and pitch controllers, see Table 4.2 and 4.5. Now the control signal for a step in yaw is investigated, see Figure 4.19.

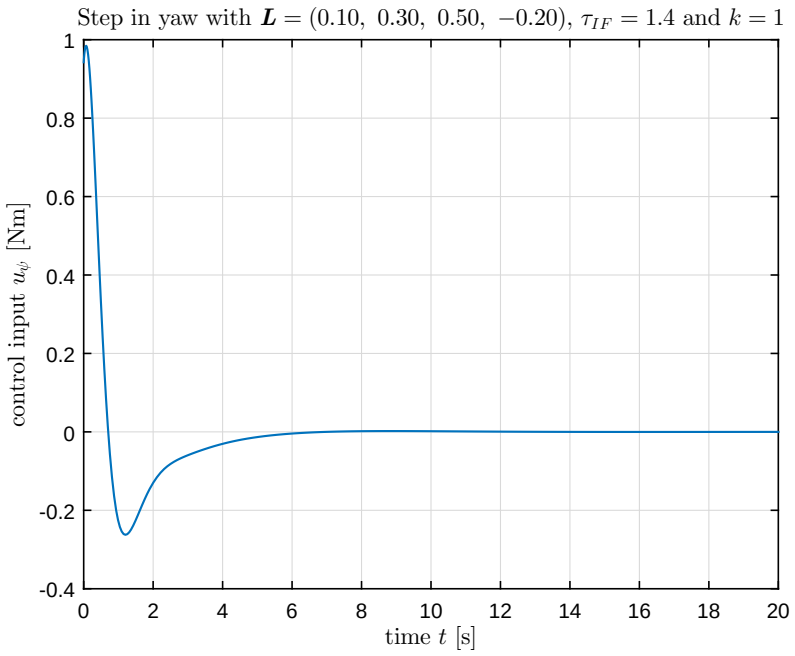

Figure 4.22: Control signal u_ψ with state feedback $\boldsymbol{L} = (0.10,\ 0.30,\ 0.50,\ -0.20)$, $\tau_{IF} = 1.4$ and $k = 1$. Step in yaw with initial condition $\psi_0 = 0$ and reference $\psi_r = \pi$.

From the above Figure it can be seen that once again the control signal is such that $|u_\psi| \leq 1$ Nm, hence the control boundary condition is satisfied. Finally, the estimated state $\hat{M}_{M_\psi}$ from the reduced observer is investigated, see Figure 4.23.

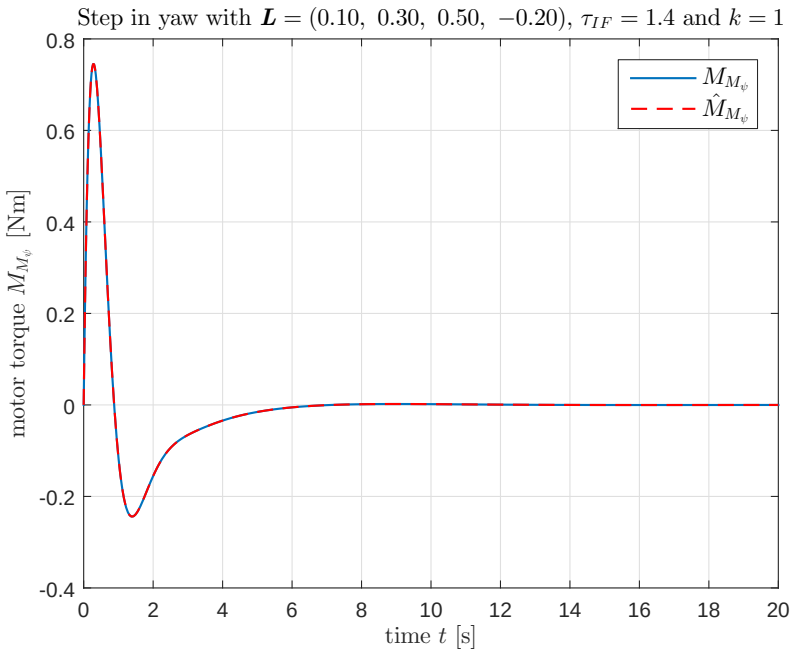

Figure 4.23: State estimation of M_{M_ψ} with state feedback $\boldsymbol{L} = (0.10,\ 0.30,\ 0.50,\ -0.20)$, $\tau_{IF} = 1.4$ and $k = 1$. Step in yaw with initial condition $\psi_0 = 0$ and reference $\psi_r = \pi$.

From the Figure it can be seen that using $k = 1$ estimates the state M_{M_ψ} well. The eigenvalues closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in yaw are computed and given in Table 4.12.

λ_1	λ_2	λ_3	λ_4
$-2.33 + 2.49i$	$-2.33 - 2.49i$	$-0.336 + 0.382i$	$-0.336 - 0.382i$

Table 4.12: Table showing eigenvalues of the closed-loop system $\boldsymbol{A} - \boldsymbol{BL}$ in yaw where $\boldsymbol{L} = (0.10,\ 0.30,\ 0.50,\ -0.20)$, using three significant figures.

From Table 4.12 it can be seen that all eigenvalues have real negative parts i.e., the system is asymptotically stable. Furthermore, all poles have an imaginary part. By using $k = 1$, the reduced observer pole is about 4 times larger than the largest poles of the closed loop system, which should provide a sufficiently fast observer.

4.4 Position Control

In this section a position controller is derived. As the name states, the purpose of the position controller is to achieve some desired translational position x_I, y_I and z_I. When controlling the octocopter by autonomous means, a position controller is very useful. By giving the octocopter some predefined path to follow, the position controller enables the vehicle to follow the path without the use of a pilot.

To change the translational position of the octocopter the controls u_x, u_y and u_z can be used. But, the control inputs u_x and u_y are very small and not feasible to use in order to control translational movement. Instead, to change position of the octocopter, the Euler angles roll ϕ, pitch θ and yaw ψ are used. Therefore, the position controller will treat the Euler angles as control inputs. In section 4.3 an attitude controller is derived controlling the Euler angles. From that section and previous analysis it is concluded that the control of yaw is slower than the controls of roll and pitch. Furthermore, translational movement can be accomplished by only changing roll and pitch angles, hence the yaw angle is not going to be actively used. In order to change vertical position z, the motor force F_{Mz} is used and controlled. Note that even though $\psi = 0$, it is important to use the control of yaw to compensate for modeling errors and disturbances. Furthermore, the attitude controller derived in section 4.2.4 and 4.3.1 has considerable rise times, therefore it is important that the position control input is not to aggressive.

To compute position control inputs, a nonlinear controller is derived using Lyapunov based control, see section 4.4.3. To use the Euler angles and the attitude controller from the previous section, a method for converting position control inputs to attitude references is derived in section 4.4.2. The final controller is presented and simulated in section 4.4.4.

4.4.1 Simplified Nonlinear Position System

Now the nonlinear system for controlling position in the inertial frame is derived. As mentioned, the position control cannot be too fast since the attitude controller is used which has considerable rise times. Therefore, it is decided to neglect the rise time in the motor dynamics and assume that the motor commands sent are instantly archived i.e., $u_z = F_{M_z}$. As previously mentioned, the control inputs $u_x = u_y = 0$ and are not going to be used to change translational position. Also, the yaw angle is not going to be used for position control more than controlling it to be zero, hence $\psi = 0$. By applying these assumptions, using the rotation matrix $\boldsymbol{R}_{B\to I}$, the body forces derived in section 3.4 and (3.32), the acceleration $\dot{\boldsymbol{v}}_I$ is given by

$$m\dot{\boldsymbol{v}}_I = \boldsymbol{R}_{B\to I}(\phi, \theta, 0)(u_z \boldsymbol{e}_z + \boldsymbol{F}_D) + mg\boldsymbol{e}_z. \tag{4.38}$$

The above expression can be written on scalar form as

$$\begin{aligned}
\ddot{x}_I &= \tfrac{1}{m}\left[c_\theta F_{Dx} + s_\phi s_\theta F_{Dy} + c_\phi s_\theta (u_z + F_{Dz})\right] \\
\ddot{y}_I &= \tfrac{1}{m}\left[c_\phi F_{Dy} - s_\phi (u_z + F_{Dz})\right] \\
\ddot{z}_I &= \tfrac{1}{m}\left[-s_\theta F_{Dx} + s_\phi c_\theta F_{Dy} + c_\phi c_\theta (u_z + F_{Dz})\right] + g.
\end{aligned} \tag{4.39}$$

The system (4.39) is now rewritten on a more general form where all states up to position are added

$$\begin{aligned}
x_I &: \begin{cases} \dot{x}_1 = x_2 \\ \dot{x}_2 = u_1 + h_1 \end{cases} \\
y_I &: \begin{cases} \dot{x}_3 = x_4 \\ \dot{x}_4 = u_2 + h_2 \end{cases} \\
z_I &: \begin{cases} \dot{x}_5 = x_6 \\ \dot{x}_6 = u_3 + h_3 \end{cases}
\end{aligned} \tag{4.40}$$

where

$$\begin{aligned}
u_1 &= \tfrac{1}{m} c_\phi s_\theta u_z \\
u_2 &= -\tfrac{1}{m} s_\phi u_z \\
u_3 &= \tfrac{1}{m} c_\phi c_\theta u_z + g
\end{aligned} \tag{4.41}$$

and

$$\begin{aligned}
h_1 &= \tfrac{1}{m}\left[c_\theta F_{Dx} + s_\phi s_\theta F_{Dy} + c_\phi s_\theta F_{Dz}\right] \\
h_2 &= \tfrac{1}{m}\left[c_\phi F_{Dy} - s_\phi F_{Dz}\right] \\
h_3 &= \tfrac{1}{m}\left[-s_\theta F_{Dx} + s_\phi c_\theta F_{Dy} + c_\phi c_\theta F_{Dz}\right].
\end{aligned} \tag{4.42}$$

Thus, here the control inputs are given by u_i and the nonlinear "disturbances" by h_i. In the next section 4.4.2 it is shown how the control inputs u_i can be converted to attitude references ϕ_r, θ_r and u_z.

4.4.2 Convert Position Control to Attitude Reference

To track some desired position, the position controller uses the attitude controller to accomplish some desired references ϕ_r, θ_r and motor force u_z, note $\psi_r = 0$. Therefore, it is necessary to convert the position control inputs u_i from section 4.4.1 to corresponding attitude and force references. From (4.41) these inputs can be solved, which gives

$$\begin{aligned}
u_z &= \frac{m(u_3 - g)}{c_\phi c_\theta} \\
\phi_r &= -\arcsin\left(\frac{m u_2}{u_z}\right) \\
\theta_r &= \arcsin\left(\frac{m u_1}{c_{\phi_r} u_z}\right).
\end{aligned} \tag{4.43}$$

Note that the system (4.41) can be solved in many ways. The solution chosen here ensures that a solution exists when $\phi, \theta = 0$. Also, the solution order is crucial, when computing the references ϕ_r, θ_r and u_z, it is important to keep the order in which the equations are written in 4.43. From the equation it becomes clear that singular points exist e.g., $u_z = 0$. To make sure that no such singularities occur during control, boundaries on the references are set. Since the controller uses the attitude controllers for roll and pitch derived in section 4.2.4, natural boundaries are $\phi_r,\ \theta_r \in [-\frac{\pi}{8},\ \frac{\pi}{8}]$. The boundary on u_z is decided to

be set such that $-2mg \leq u_z \leq \frac{mg}{2}$, which provides some margin to the boundaries on u_z, see Figures 4.2-4.4. For clarification, the above equation and used boundaries are now explained by pseudo code, see Algorithm 2.

Algorithm 2 Convert Position Control to Attitude Reference

function convert_position($\boldsymbol{u}$, ϕ, θ)
  $u_z \leftarrow \frac{m(\boldsymbol{u}_3 - g)}{c_\phi c_\theta}$
  if $u_z > \frac{mg}{2}$ **then**
    $u_z \leftarrow \frac{mg}{2}$
  else if $u_z < -2mg$ **then**
    $u_z \leftarrow -2mg$
  if $|\frac{m\boldsymbol{u}_2}{u_z}| > \sin(\frac{\pi}{8})$ **then**
    $\phi_r \leftarrow -\text{sign}(\frac{m\boldsymbol{u}_2}{u_z})\frac{\pi}{8}$
  else
    $\phi_r \leftarrow -\arcsin(\frac{m\boldsymbol{u}_2}{u_z})$
  if $|\frac{m\boldsymbol{u}_1}{c_{\phi_r} u_z}| > \sin(\frac{\pi}{8})$ **then**
    $\theta_r \leftarrow \text{sign}(\frac{m\boldsymbol{u}_1}{c_{\phi_r} u_z})\frac{\pi}{8}$
  else
    $\theta_r \leftarrow \arcsin(\frac{m\boldsymbol{u}_1}{c_{\phi_r} u_z})$
  $\psi_r = 0$
  return $u_z,\ \phi_r,\ \theta_r$ and ψ_r

4.4.3 Lyapunov Based Control

Now a control for stabilizing (4.40) is derived. Since the system in need of control is nonlinear, it is decided to now use a nonlinear control method. There are many applicable methods which are more or less suitable depending on the system. By looking at the system (4.40) one can see that it will be easy to cancel the nonlinearities and achieve a stabilizing control by using Lyapunov based control. Lyapunov based control has its fundamentals in Lyapunov's direct method [11, Lecture 4] which can be used to determine stability of a nonlinear system.

By using Lyapunov's theorem for global asymptotic stability, the control can be chosen such that global asymptotic stability of the system (4.40) can be achieved. Lyapunov's theorem for global asymptotic stability is now given [11, Lecture 4 p. 14]. Let $\dot{\boldsymbol{x}} = f(\boldsymbol{x}) \in \mathbb{R}^n$ and $f(\mathbf{0}) = \mathbf{0}$. If there exist a $\mathbb{C}^1$ function $V : \mathbb{R}^n \to \mathbb{R}$ such that

1. $V(\mathbf{0}) = 0$
2. $V(\boldsymbol{x}) > 0,\ \forall \boldsymbol{x}_i \neq 0$
3. $\dot{V}(\boldsymbol{x}) < 0,\ \forall \boldsymbol{x}_i \neq 0$
4. $V(\boldsymbol{x}) \to \infty\ ||\boldsymbol{x}|| \to \infty$

then $\boldsymbol{x} = \mathbf{0}$ is global asymptotically stable. Before proposing a controller, the variable change $\tilde{x}_i = x_i - x_{r_i}$ for $i = 1, 3, 5$ in (4.40) is made to track some reference in x, y and z. Note that it is assumed that x_{r_i} is a constant, so $\dot{\tilde{x}}_i = \dot{x}_i = x_{2i}$ for $i = 1, 3, 5$. This is going to cause some smaller errors when simulating the system for continues references in position. But since the change in reference will be relatively small this will not be of any issue here. Also, by this assumption, noise in the reference signal will not be enhanced. Then by applying the Lyapunov function $V = \frac{1}{2}\left(\sum_{i=1,3,5} \tilde{x}_i^2 + \sum_{i=2,4,6} x_i^2\right)$, one instantly sees that condition 1, 2 and 4 are satisfied. The time derivative of V gives

$$\begin{aligned}\dot{V} &= \tilde{x}_1\dot{x}_1 + x_2\dot{x}_2 + \tilde{x}_3\dot{x}_3 + x_4\dot{x}_4 + \tilde{x}_5\dot{x}_5 + x_6\dot{x}_6 \\ &= \tilde{x}_1 x_2 + x_2(u_1 + h_1) + \tilde{x}_3 x_4 + x_4(u_2 + h_2) + \tilde{x}_5 x_6 + x_6(u_3 + h_3).\end{aligned}$$

By letting $u_i = -\tilde{x}_{2i-1} - c_i x_{2i} - h_i$ where $c_i > 0$, one gets

$$\dot{V} = -c_1 x_2^2 - c_2 x_4^2 - c_3 x_6^2 < 0, \ \forall \boldsymbol{x}_i \neq 0.$$

Hence, the proposed controller provides global asymptotic stability of the system (4.40). Furthermore, the tracking of x_{i_r} is accomplished since $\tilde{x}_i = 0$ for $i = 1, 3, 5$. The parameter c_i is now going to be tuned such that the system behaves in a suitable manner, see section 4.4.4.

4.4.4 Tuning and Simulation of Controller

The parameter c_i defined in section 4.4.3 is now chosen. The effect of c_i is similar to the effect of the parameter l_3 which adjusts the derivative part in the previously used state feedback controller, see Figure 4.8. Hence, increasing c_i will decrease oscillations but increase rise time. Although, now c_i is tuning the squared effect of the derivative of the system 4.40. The parameter is chosen such that no oscillations are present when simulating the complete nonlinear system for each translational component separately, note that in Figure 4.24 all translational components are simulated simultaneously.

Before moving on, it is noted that the Lyapunov based controller will produce unbounded control inputs. But since the position control inputs are converted to attitude references, see section 4.4.2, no problems arise due to the set boundaries on the attitude references. Therefore, it is decided to simulate the system for large reference values in position so that the boundaries on the attitude references are reached and tested. Now the complete nonlinear system is simulated using the position controller with $c_1 = c_2 = 1.6$ and $c_3 = 1.8$, the reference $x_r = y_r = 10$ m, $z_r = -10$ m (z-axis down) and initial condition $x_0 = y_0 = z_0 = 0$ m, see Figure 4.24.

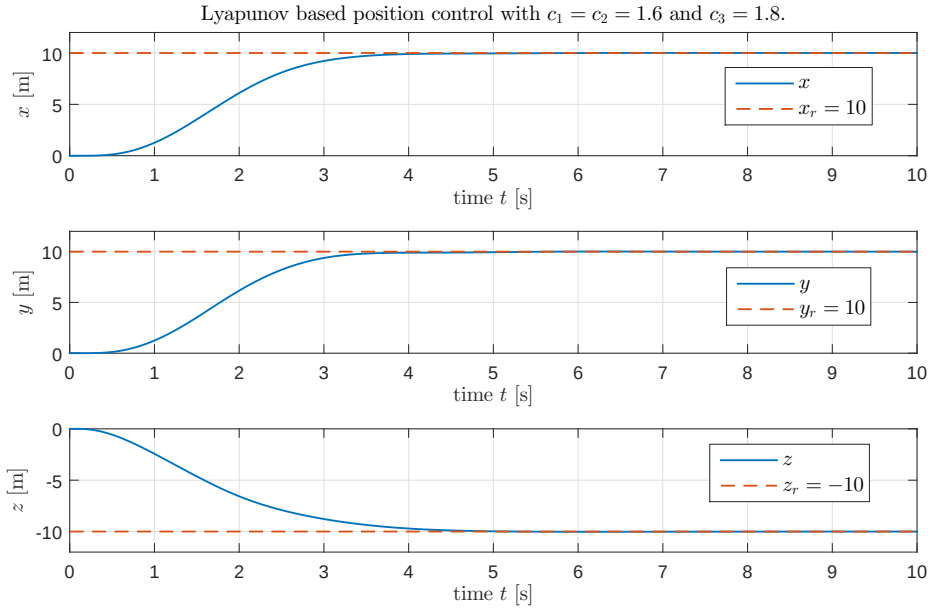

Figure 4.24: Lyapunov based position control with $c_1 = c_2 = 1.6$ and $c_3 = 1.8$. Step in position with reference $x_r = y_r = 10$ m, $z_r = -10$ and initial condition $x_0 = y_0 = z_0 = 0$ m.

From the above Figure it can be seen that the controller tracks the desired position well. It takes about 5 s before the system has reached the desired position. Some minor oscillations in Figure 4.24 can be observed. These oscillations arise are due to cross coupling effects in the attitude controller which are not compensated for. The corresponding control inputs from the position controller u_i are illustrated in Figure 4.25.

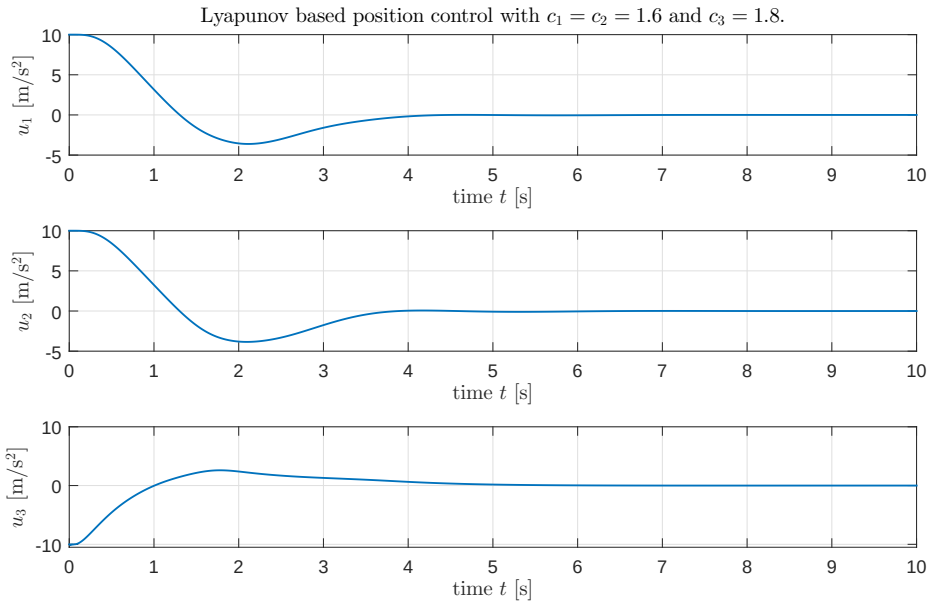

Figure 4.25: Showing the position control input from the Lyapunov based position control with $c_1 = c_2 = 1.6$ and $c_3 = 1.8$. Step in position with reference $x_r = y_r = 10$ m, $z_r = -10$ and initial condition $x_0 = y_0 = z_0 = 0$ m.

From the Figure one can be see that the control inputs from the position controller are not bounded. The effect of the boundaries on the attitude controller is now going to be illustrated by plotting the corresponding attitude references values ϕ_r, θ_r and u_z, see Figure 4.26.

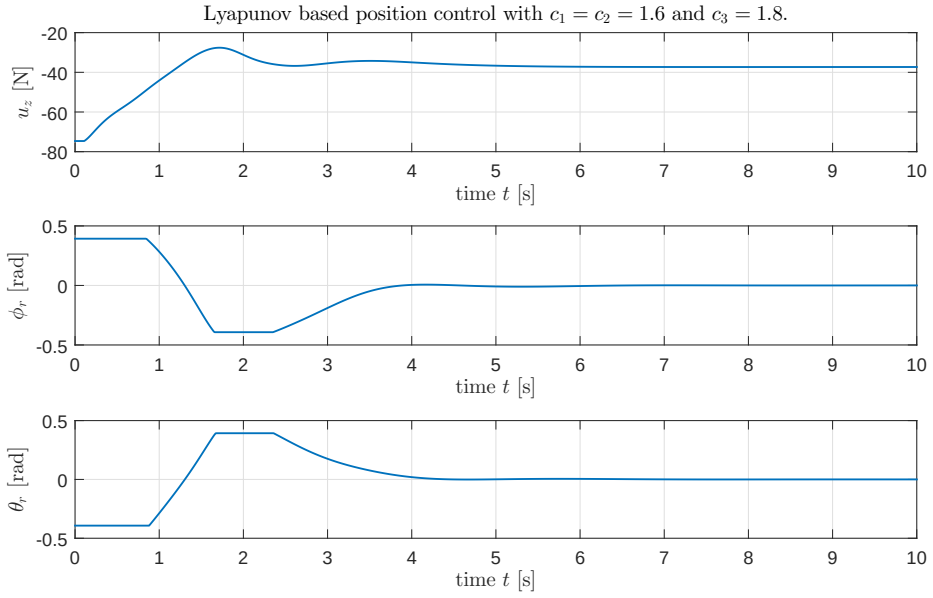

Figure 4.26: Showing the converted attitude references values from the Lyapunov based position control with $c_1 = c_2 = 1.6$ and $c_3 = 1.8$. Step in position with reference $x_r = y_r = 10$ m, $z_r = -10$ and initial condition $x_0 = y_0 = z_0 = 0$ m.

From the Figure it can now be seen how the attitude reference boundaries defined in section 4.4.2 are hit. When the controller approaches the desired position the attitude control references are decreased and stabilizes the octocopter. Hence, the approach to convert position control inputs to attitude references works well even when the reference in position is large. For more simulations of the position controller see chapter 5.

Chapter 5

Simulation

In this chapter simulations of the complete nonlinear model derived in chapter 3 are carried out. The simulations are done using MATLAB's simulation tool Simulink. The model implemented makes use of the previously mentioned quaternions, section 3.2. This since quaternions allow for simulations in all 6DoF without any concerns for singularities which might occur when using Euler transformation matrices. The simulations are run in discrete time with a fixed-step solver at 100 Hz, which is the rate at which the Nav Raspi sends data, section 2.1.3. Almost all blocks used in the Simulink model are written by the author and implemented by using MATLAB function blocks. Furthermore, most of the blocks have programmed masks which enables for easily adjustable simulations. The code itself is not discussed in this thesis but a visualization of the final model is made and can be seen in Figure 5.1.

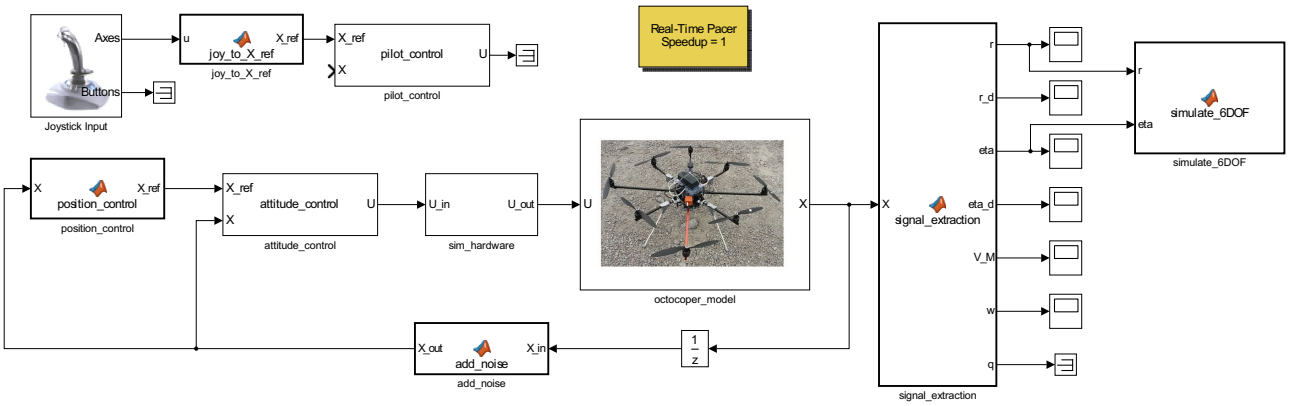

Figure 5.1: Showing the root structure of the Simulink model used for simulations of the complete nonlinear system.

From the above Figure the root structure of the Simulink model can be seen. Note that here the position controller is used with the attitude controller. The pilot-based controller is not connected to the system. The reference input to the pilot-based controller comes here from a pc gaming controller which is made possible by using the predefined Simulink Joystick Input block. Also, the Real-Time Pacer block is downloaded from mathwork's homepage and used to enable simulations in real time. The block sim_hardware simulates the quantization of the motor control input since $u_M \in [0,\ 254]$ byte. This block also adds a delay to the control input. It is decided that a delay of 20 ms should be sufficiently large. The block octocopter_model implements all nonlinear dynamics, including the varying time constant of the motor dynamics. The block ad_noise, adds the noise discussed in section 2.1.3. To visualize the state of the octocopter a block simulate_6DOF in Simulink is made, written in MATLAB code. The simulation data is used to visually display the movement of the octocopter in 6DoF, see Figure 5.2.

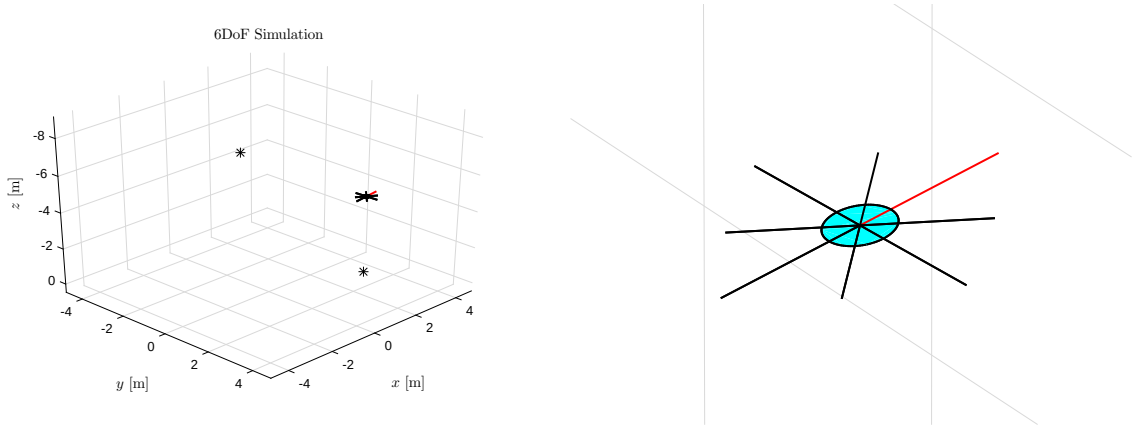

Figure 5.2: 6DoF simulation in Simulink. Full scale plot on the left where the black stars indicate translational position and are used as a visual guidance. The right Figure is zoomed in on the octocopter, the red arm indicates the forward defined direction of the vehicle.

This visualization is very useful when simulating the system, especially when the pilot-based controller is used and the reference input comes from a pc gaming controller. This allows for training and testing of the controller in a controlled environment.

Now simulations are carried out for the previous derived control methods: pilot-based, attitude and position control. The simulations are divided into two main sections, one without disturbances and one with, section 5.1 and 5.2 respectively. The disturbances are the mentioned noises, system delays and quantization of the motor control input.

5.1 Without Disturbances

The simulations following in this section do not experience signal noises, system delays or quantization of motor control input. The simulations are carried out for the controllers: pilot-based, attitude and position control.

5.1.1 Pilot-based Control

Now simulations using the pilot-based control are carried out. First, simulations are performed for all controlled states separately i.e., roll ϕ, pitch θ and yaw-rate $\dot{\psi}$. At the end, a final simulation of all states together is made. It is decided that the reference inputs(s), state(s) and control inputs(s) of the controlled states are the most relevant and therefore plotted. During the simulations the control in z, $u_z = -mg$, hence hovering. The result can be seen in Figures 5.3-5.6.

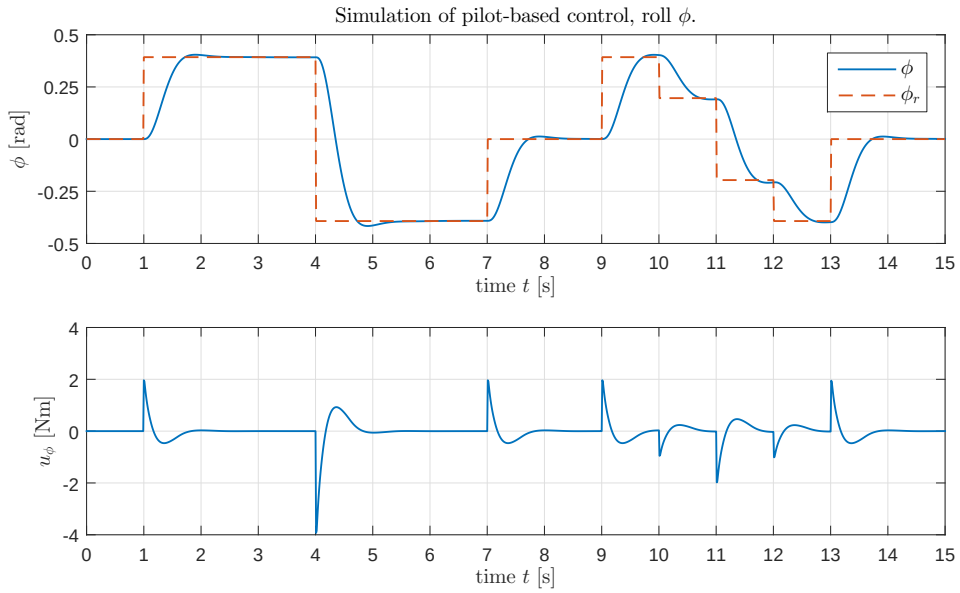

Figure 5.3: Simulation of pilot based-control in roll ϕ with corresponding control input u_ϕ. Without noise, delay and quantization.

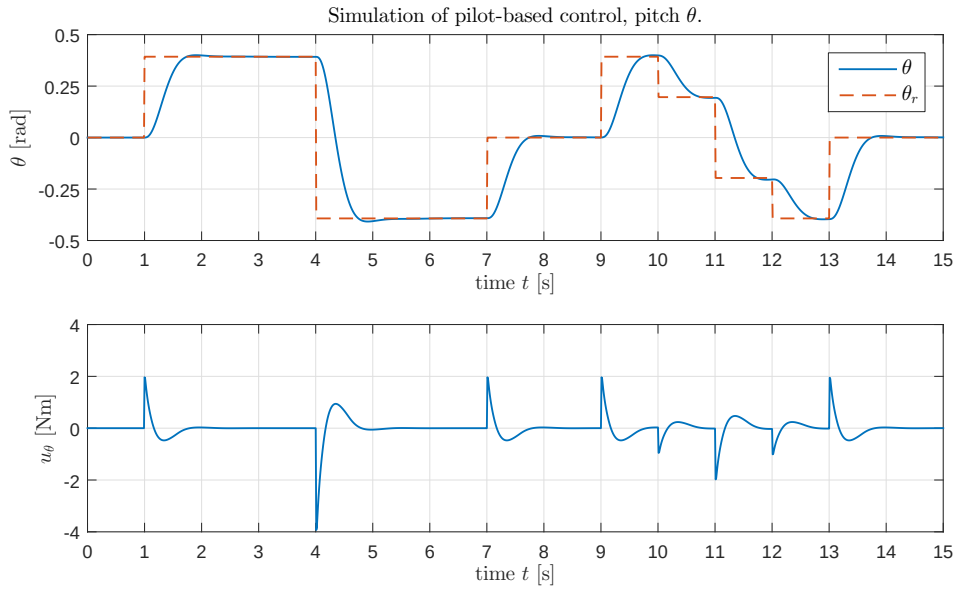

Figure 5.4: Simulation of pilot based-control in pitch θ with corresponding control input u_θ. Without noise, delay and quantization.

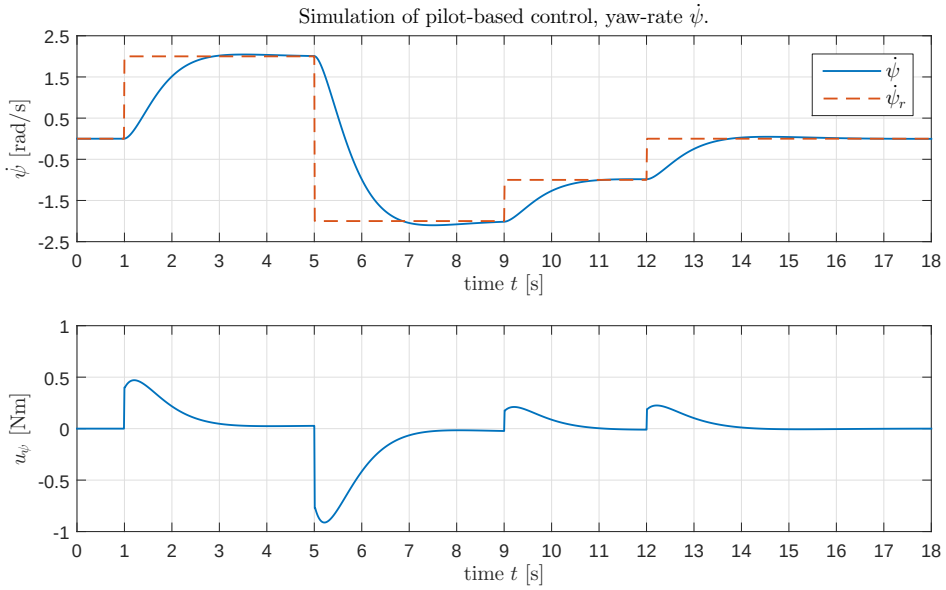

Figure 5.5: Simulation of pilot based-control in yaw-rate $\dot{\psi}$ with corresponding control input u_ψ. Without noise, delay and quantization.

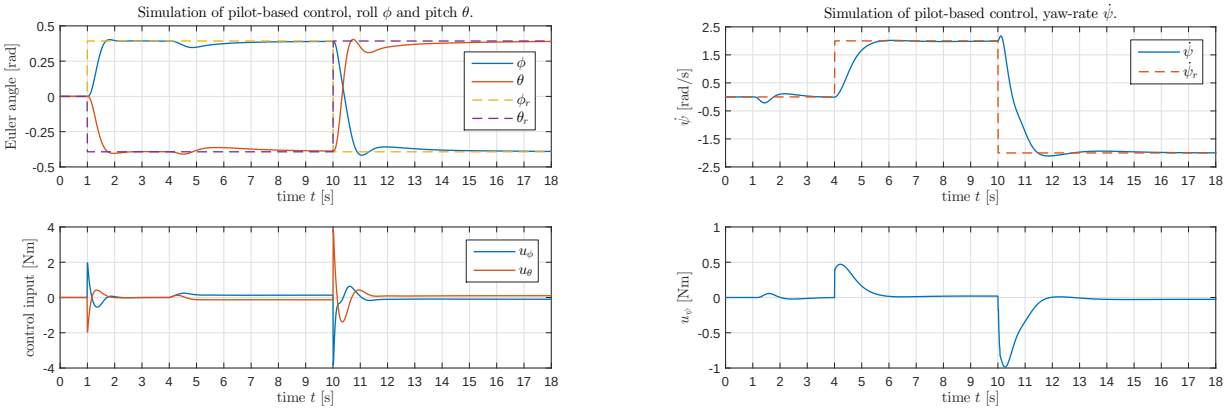

Figure 5.6: Simulation of pilot-based control in roll ϕ, pitch θ and yaw-rate $\dot{\psi}$ with corresponding control inputs. Without noise, delay and quantization.

Before commenting the simulations, note that the two subplots in Figure 5.6 are from the same simulation. From the Figures 5.3-5.5 it can be seen that the linear pilot-based control method performs very well in comparison to the linear simulations in section 4.2, even with all nonlinear effects acting. From the Figures it can also be seen that the control inputs are kept within their boundaries. In Figure 5.6 multiple reference inputs are given. Coupling effect between the states can be observed, nevertheless, the reference inputs are tracked well. At $t = 10$ s a step in all states is applied, which corresponds to the maximum allowed change in step for all states. When doing so the control input in yaw u_ψ is slightly reduced which is due to the use of the priority algorithm proposed in section 4.1.3. Hence, even for demanding reference inputs the system manages to track the desired references.

5.1.2 Attitude Control

Here simulations using the attitude controller are carried out. Since the proposed attitude controller uses the same controls in roll ϕ and pitch θ as the pilot-based controlled, simulations of these states individually, are not performed again, see section 5.1.1. The simulations are illustrated in Figures 5.7-5.8.

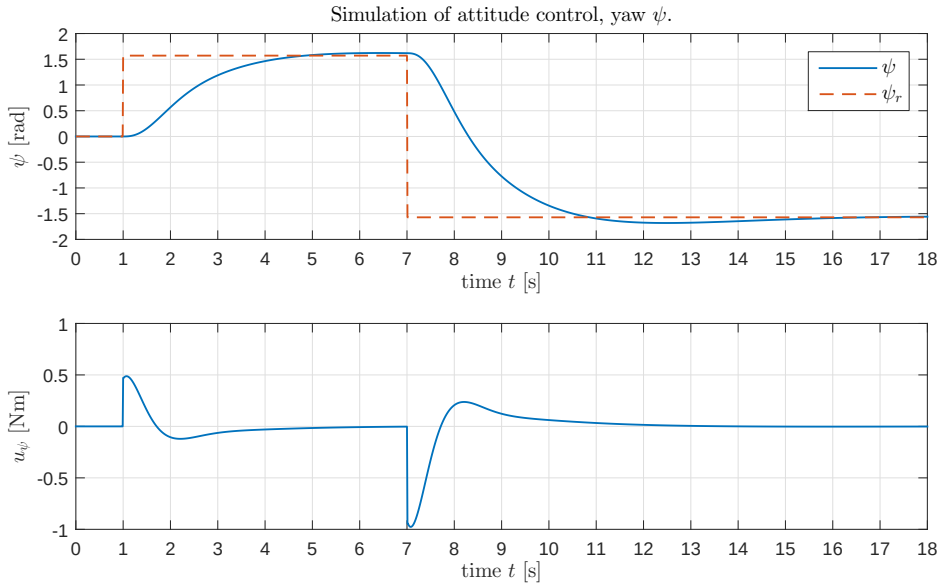

Figure 5.7: Simulation of attitude control in yaw ψ with corresponding control input u_ψ. Without noise, delay and quantization.

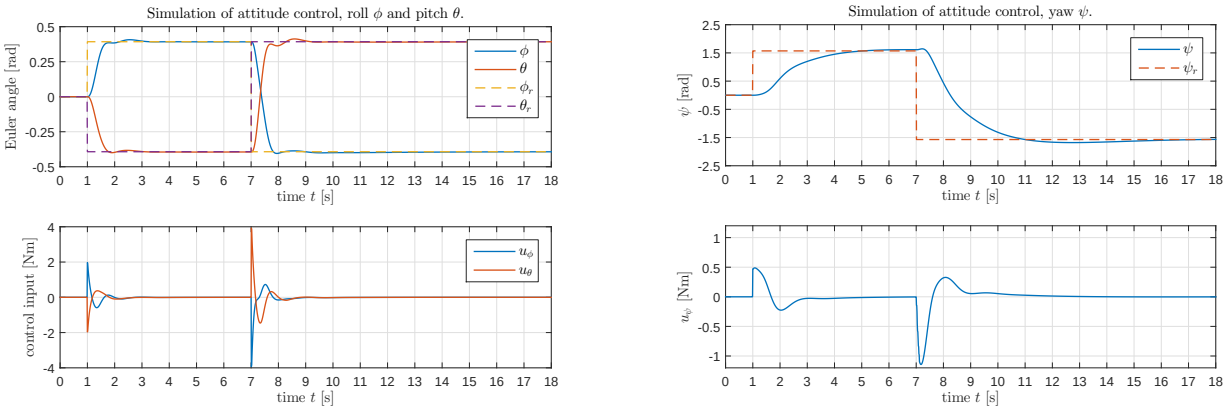

Figure 5.8: Simulation of attitude control in roll ϕ, pitch θ and yaw ψ with corresponding control inputs. Without noise, delay and quantization.

From the Figure 5.8 it can once again be seen that the linear control method performs very well in comparison to the linear simulations in section 4.3. From the Figure it can also be seen that the control input u_ψ is kept within its boundaries. In Figure 5.8 multiple reference inputs are given. Coupling effect between the states can again be observed, nevertheless, the reference inputs are tracked well. At $t = 7$ s a step in all states is applied which corresponds to the maximum allowed change in step for all states. When doing so the control input in yaw u_ψ is slightly reduced which is due to the use of the previously proposed priority algorithm.

5.1.3 Position Control

Now a simulation for the position controller is performed. In section 4.4, step responses using the full nonlinear system were carried out. Therefore, it is decided to now only simulate the system for a continuous reference input. One of the main purposes of a position controller is to be able to fly autonomously, therefore it is reasonable to do so by applying continuous reference inputs. It is decided to use the following reference inputs $x_r = A(\cos(\frac{2\pi}{T}t) - 1)$, $y_r = A\sin(\frac{2\pi}{T}t)$ and $z_r = -kt$, where $A = 5$ m, $T = 15$ s and $k = 1$. In other words, the octocopter should move in a circular motion with radius 5 m with period 15 s about the xy-plane whilst the altitude is increasing with 1 m/s. Before the simulation is performed, it is quite obvious that there is going to be some delay between the reference input and the actual state since there exists considerable rise times in the controller. Nevertheless, optimally the octocopter should track the desired reference. It is determined to plot the reference states and inputs, and all control signals i.e., position, converted position and attitude control inputs, see Figure 5.9.

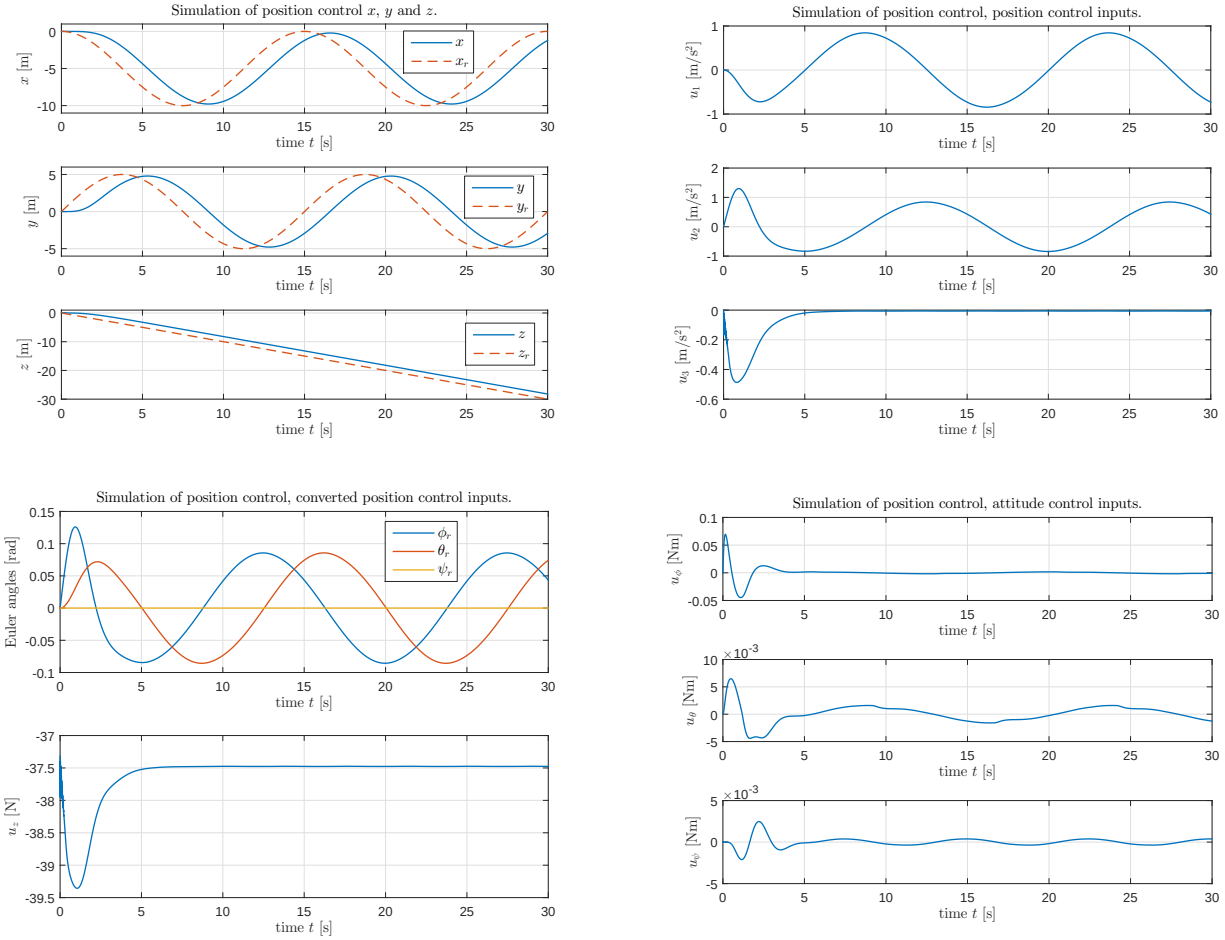

Figure 5.9: Simulation of position controller with continues reference inputs in x, y and z. Without noise, delay and quantization.

From the above Figure, it can be seen that the position controller causes the octocopter to move in the desired manner. But, there exists some delay of the actual state to the reference input, ≈ 2 s. This is due to the previously mentioned rise times in the position controller and due to the fact that when the position controller is designed it is assumed that e.g., $\dot{x}_r = 0$. By inspecting the different control inputs, one can see that no boundaries are hit, compare to Figures 4.24-4.26. Furthermore, it can be seen that it is important to have a controller of yaw even if $\psi_r = 0$, which is due to mentioned nonlinearities and disturbances which will cause ψ to change.

5.2 With Disturbances

The simulations following in this section do experience disturbances in form of signal noise, system delays and quantization of the motor control input. The simulations are carried out for the controllers: pilot-based, attitude and position control.

5.2.1 Pilot-based Control

The same simulations as in section 5.1.1 are carried out but now with disturbances, see Figures 5.10-5.13.

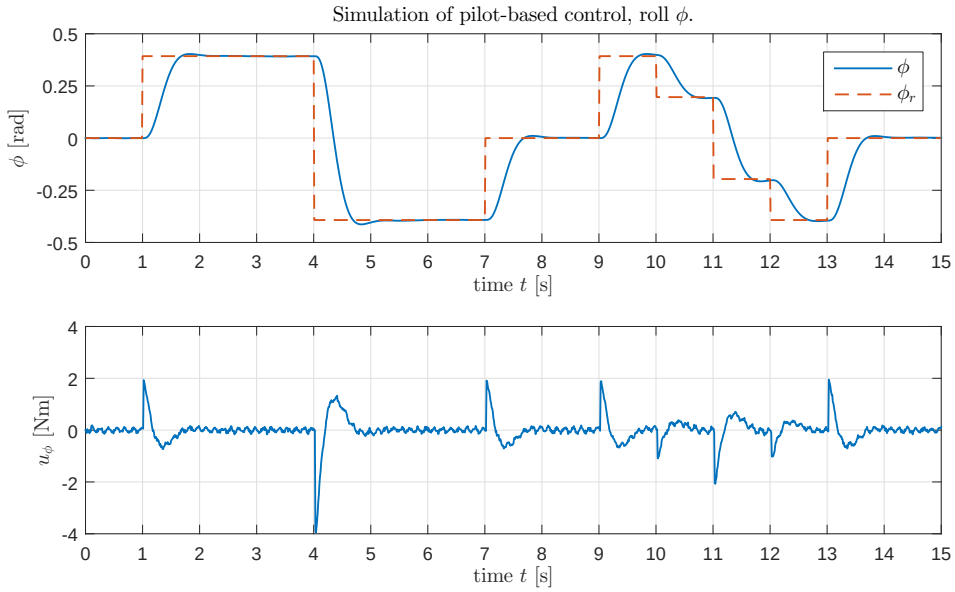

Figure 5.10: Simulation of pilot based-control in roll ϕ with corresponding control input u_ϕ. With noise, delay and quantization.

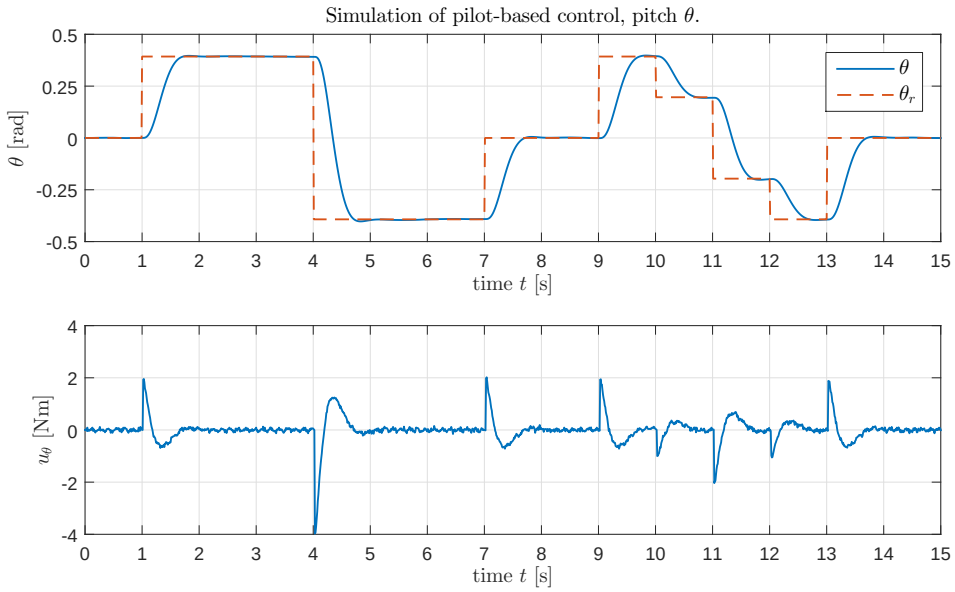

Figure 5.11: Simulation of pilot based-control in pitch θ with corresponding control input u_θ. With noise, delay and quantization.

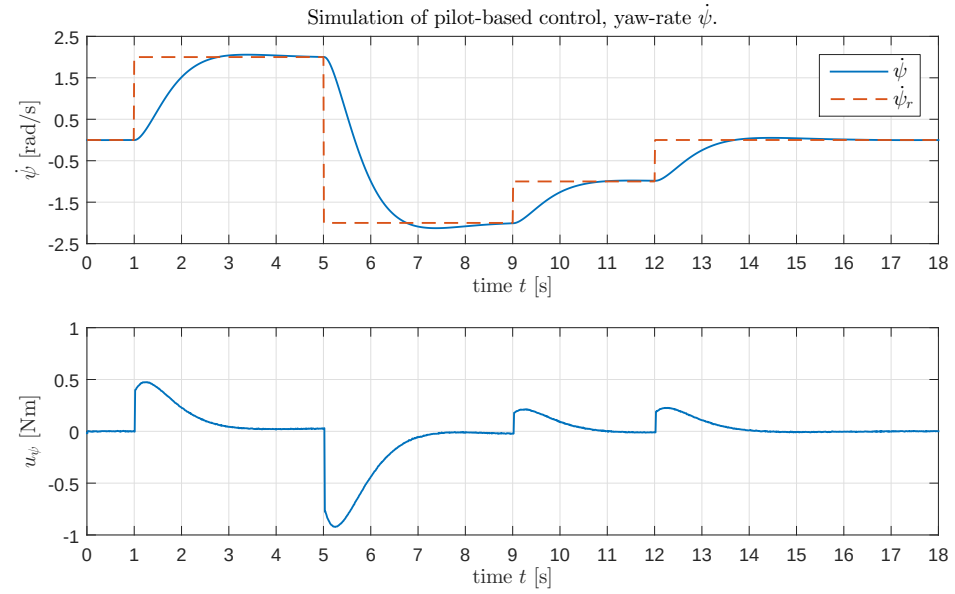

Figure 5.12: Simulation of pilot based-control in yaw-rate $\dot{\psi}$ with corresponding control input u_ψ. With noise, delay and quantization.

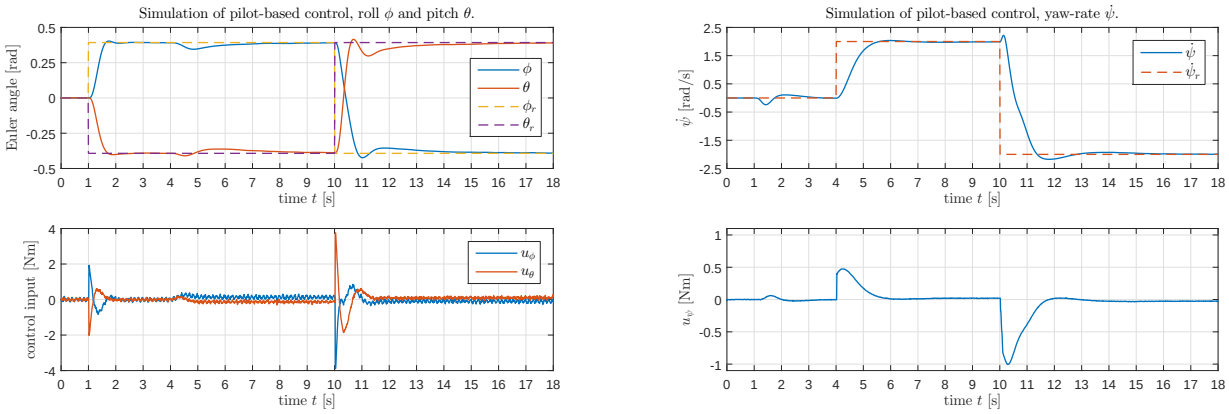

Figure 5.13: Simulation of pilot-based control in roll ϕ, pitch θ and yaw-rate $\dot{\psi}$ with corresponding control inputs. With noise, delay and quantization.

The discussions held about the step responses in section 5.1.1 can also be applied here. From Figures 5.10-5.13 it can be seen that with the now added disturbances, the control signals due fluctuate and some minor oscillations in the states can be observed. But, even with the added disturbances, the reference input is tracked well and the disturbances seam to be attenuated.

5.2.2 Attitude Control

The same simulations as in section 5.1.2 are carried out but now with disturbances, see Figures 5.14-5.15.

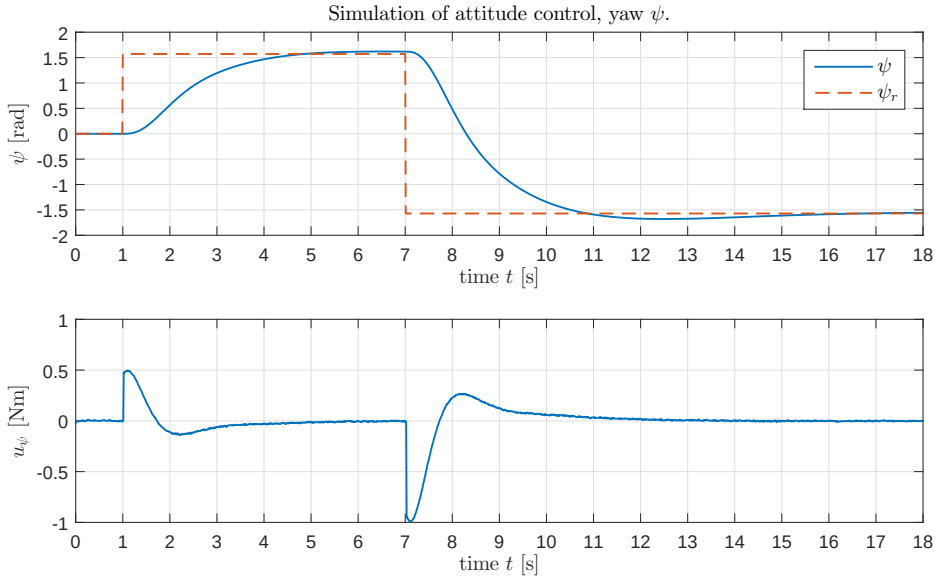

Figure 5.14: Simulation of attitude control in yaw ψ with corresponding control input u_ψ. With noise, delay and quantization.

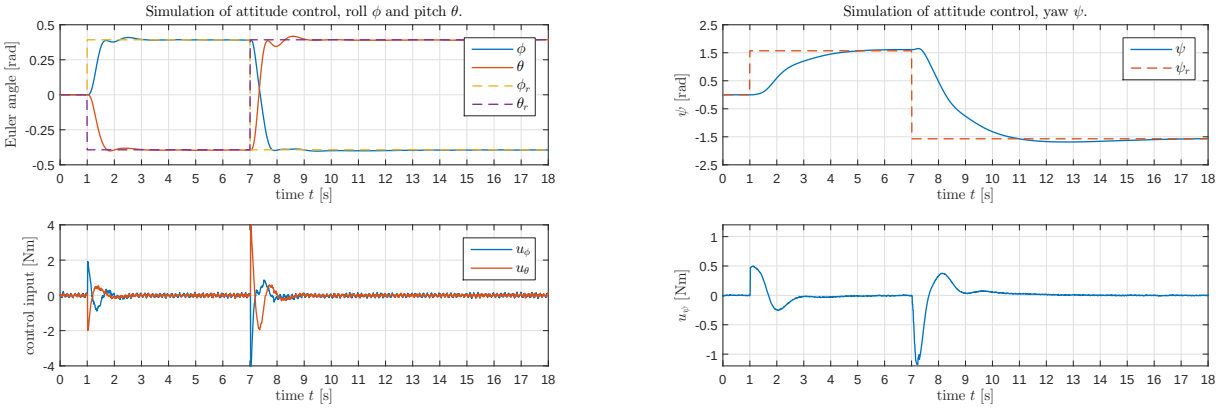

Figure 5.15: Simulation of attitude control in roll ϕ, pitch θ and yaw ψ with corresponding control inputs. With noise, delay and quantization.

Aside from the discussion held in section 5.1.2, it can again be seen that the added disturbances have caused the control signals to fluctuate and caused some minor oscillations in the states in need of control. Otherwise, the reference input is still tracked well and the disturbances seam to be attenuated.

5.2.3 Position Control

Now the same simulation as in section 5.1.1 is carried out but with disturbances, see Figure 5.16.

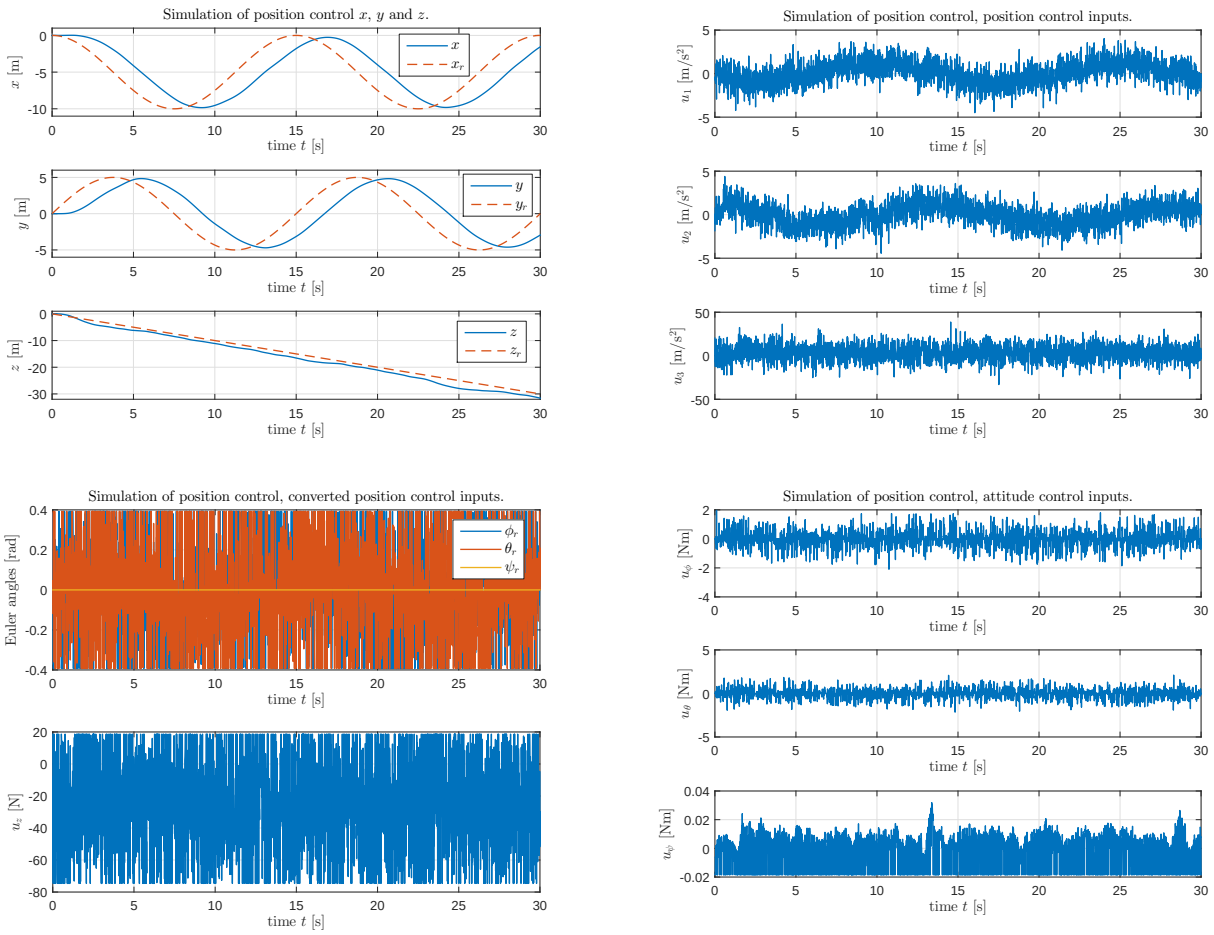

Figure 5.16: Simulation of position controller with continues reference inputs in x, y and z. With noise, delay and quantization.

Firstly, from Figure 5.16 it can be seen that despite the added disturbances the system manages to maneuver the octocopter in a fairly desired manner. Secondly, from the control input plots it can be seen that the added disturbance has a large impact on the control inputs. This large propagation of the disturbance on the control inputs is due to the large variances set on the noise added on the translational states, see section 2.1.3. From the converted attitude reference input plot one can now observe how the reference inputs hits their defined boundaries set in section 4.4. Nevertheless, by using these noise levels the controller manages to track the desired path in an acceptable manner.

Chapter 6

Evaluation of Flying Data

An important and crucial part in this thesis is the testing of the developed octocopter and the proposed control strategies. As mentioned in section 2.1.2, the octocopter runs on C code which has been generated from Simulink. The final Simulink model includes the possibility to log data during the running stage of the system, which allows for great possibilities to analyze real flying data. As mentioned in chapter 1, the thesis project is conducted in collaboration with another technologist, Markus Mikkelsen. We, together with Saab Dynamics, decided that the first implemented controller for testing should be the pilot-based controller. This mainly to reduce the control chain and the probability for errors, e.g., the positions controller is more complex.

Before flying, the implemented controller was tested without any mounted propellers to ensure that proper motor commands from the controller were sent and to further test implemented functions. It was decided to conduct the first ever flight at an open secluded football field. A flight schedule was made, which divided the flying time equally between me and Markus. The schedule was roughly divided into two main parts, flying without and with the integrating parts in the controller. This division was made since it was concluded that the integrating part could cause problems. If the vehicle would end up in a position where the integrating part would start to integrate, large control inputs could be generated resulting in unexpected behavior. It is noteworthy to mentioned that both controllers had implemented anti-windup and other fail safe features. It was decided to test my controller first, without the integrating states. The octocopter lifted and was able to stabilize itself in roll and pitch, see Figure 6.1.

Figure 6.1: First flight, using the pilot-based controller without integrating states.

Before the testing with integrating states was conducted, the whether conditions changed and it started to rain, therefore at this point the testing was ended. During the flight, windy conditions ruled. Also, even though the octocopter was flyable, the control in yaw-rate was somewhat poor, causing the vehicle to drift. When the code later was inspected, it was found that a single parameter for the yaw-rate controller had a shift in the decimal point. Furthermore, the logging of the data experienced some smaller delays due to the high sample rate. This was fixed by using multiple threads in the GC Raspi. Because of these minor bugs in the code and the windy conditions it is decided not to illustrate the data logged from the first flight.

Before the second flight occasion, the above software problems were solved and future testing of the octocopter were performed. In order to carry out tests with mounted propellers, the vehicle was fixed by elastic threads to a 25 kg weight. The second flight was conducted in a hangar at Saab, thus no wind would interfere. Once again the flying time was divided between me and Markus. Since the octocopter already had shown that it was flyable, it was decided that the pilot was allowed to fly and test the vehicle to greater extends this time. Indoors, in the hangar without any integrating part, the octocopter showed good performance and the pilot was able to maneuver the vehicle as desired. Furthermore, by trimming the yaw-rate, the drift in yaw experienced during the first flight, was gone. With the integrating states enabled the octocopter performed very well, the integrating states made the controller more responsive and the before required trimming of the yaw-rate was now not needed. Due to photographic restriction, no footage from the flights are available.

Unfortunately, when Markus controller was implemented, the octocopter started to oscillate in roll, pitch and yaw, and the pilot was not able to control the vehicle. The vehicle flew up in the ceiling and was after this point beyond recovery. The octocopter crashed and suffered severe damage. Since the flight was carried out in the final stages of the thesis project, it was not manageable due to lack of time to start any repairs. Nevertheless, the data from my flights were successfully logged and could be downloaded. Relevant data are presented in sections 6.1-6.2.

6.1 Pilot-based Control without Integrating States

As above mentioned, the second flight occasion was divided into two parts, without and with the integrating states in the pilot-based controller. The data logged from the flight without the integrating states is now going to be presented. It is decided that the most relevant data logged are the states which are controlled ψ, θ and $\dot{\psi}$, the respective reference inputs and the corresponding control inputs. The flight lasted 132 s and the data is presented in Figures 6.2-6.4.

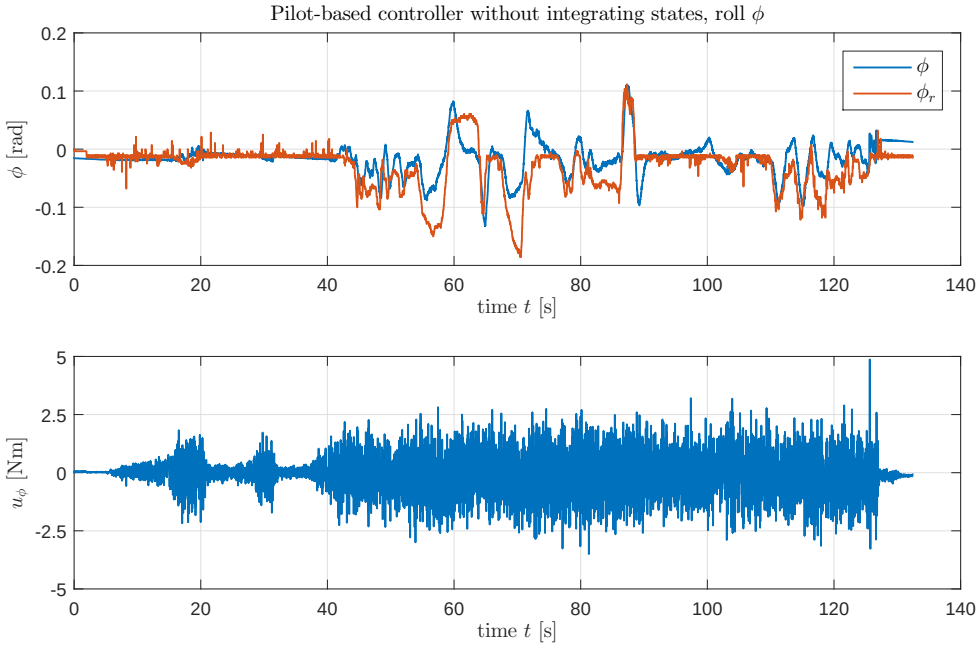

Figure 6.2: Flying data logged from second flight. Pilot-based control without integrating states, roll ϕ.

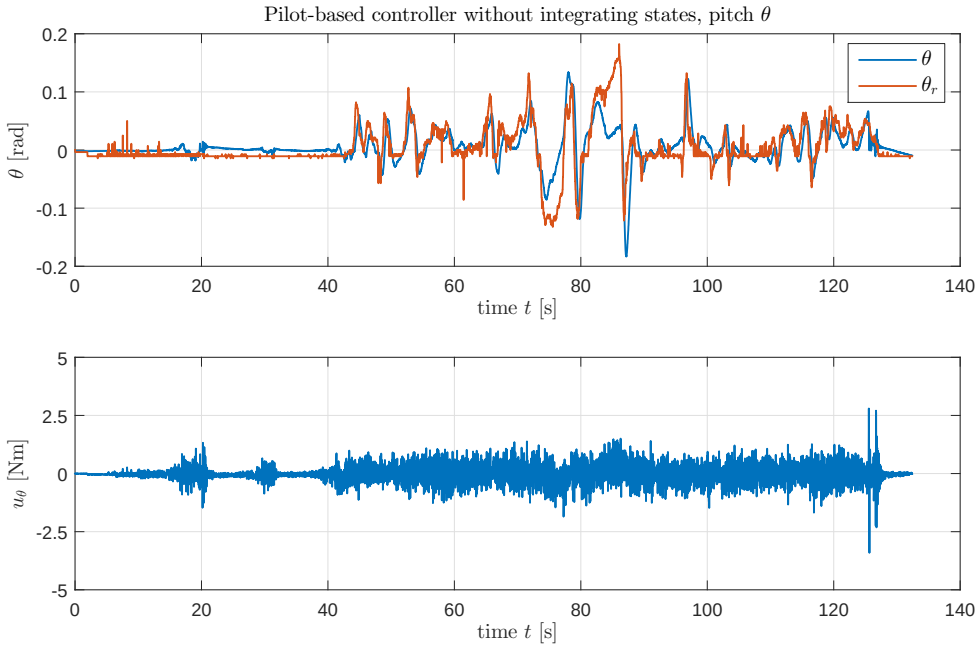

Figure 6.3: Flying data logged from second flight. Pilot-based control without integrating states, pitch θ.

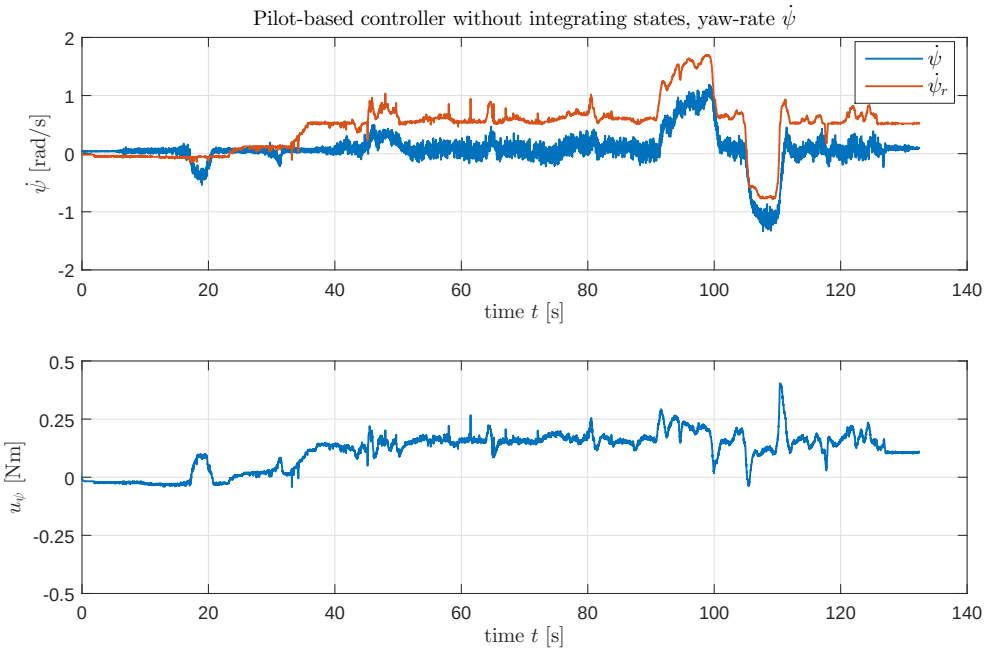

Figure 6.4: Flying data logged from second flight. Pilot-based control without integrating states, yaw-rate $\dot{\psi}$.

Firstly, from the above Figures one can see that vehicle lifts at about 40 s. From Figures 6.2-6.3 it can be seen that the controller is able to accomplish roll and pitch commands. Although, at some point e.g., $t = 60$ s the desired roll reference is not fully followed, see Figure 6.2. By inspecting Figure 6.4, one can see that without the integrating states, the yaw-rate reference needs trimming before the octocopter is stable in yaw. After the trimming, the yaw-rate did follow the reference input quite well, note the offset. Also, all the control inputs are kept within their defined boundaries, hence the implemented priority algorithm is not engaged during the flight. Furthermore, the control input is noisy which is mainly due to the noise in the estimated angular-rates e.g., $\dot{\psi}$, see section 2.1.3. Also, it can be seen the reference signal does experience some smaller disturbance, which probably is due to the quantization caused when converting the PPM signal from the radio transmitter to a byte, see section 2.1.9. Besides these disturbances, the vehicle did perform very well and the pilot was able to maneuver the octocopter as desired.

6.2 Pilot-based Control with Integrating States

The second flight occasion with the integrating states, i.e., the complete pilot-based controller lasted 152 s and the data is now going to be presented, see Figures 6.5-6.7.

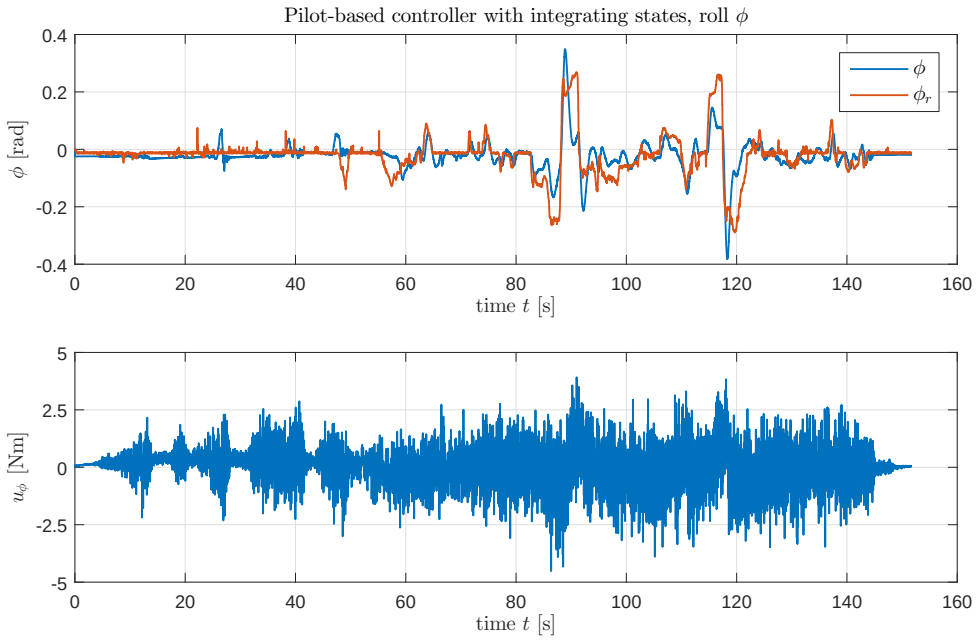

Figure 6.5: Flying data logged from second flight. Pilot-based control with integrating states, roll ϕ.

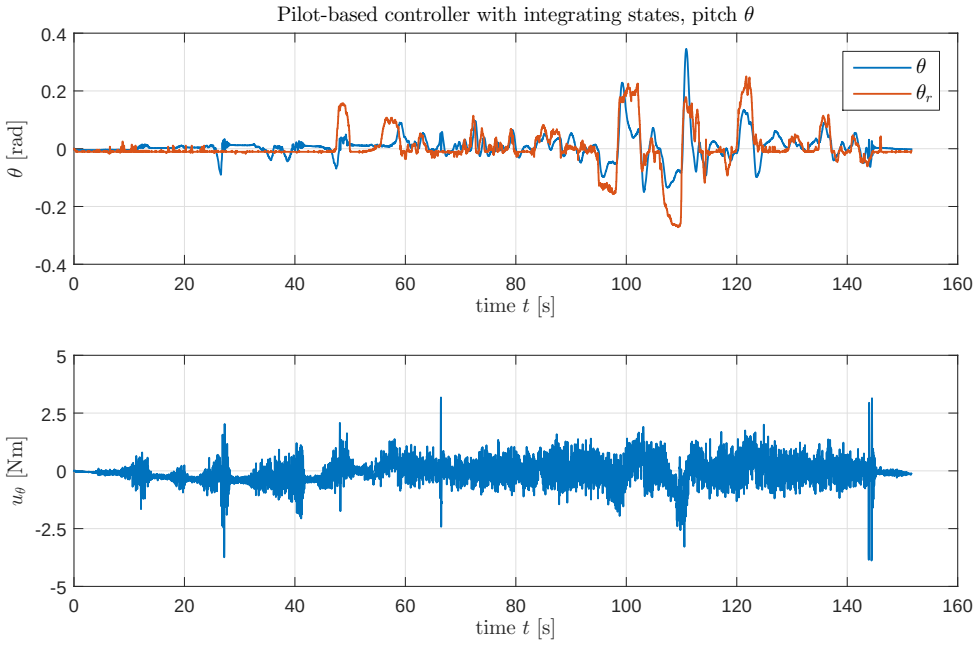

Figure 6.6: Flying data logged from second flight. Pilot-based control with integrating states, pitch θ.

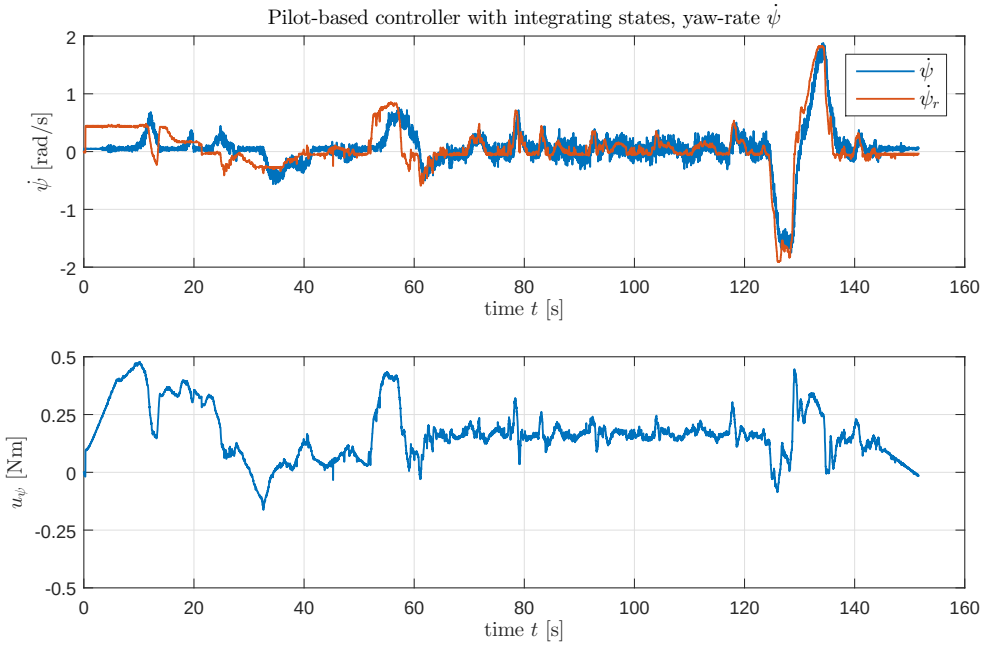

Figure 6.7: Flying data logged from second flight. Pilot-based control with integrating states, yaw-rate $\dot{\psi}$.

With the integrating states enabled, it can be seen from the above Figures that the control input for all states has increased in size which is due to the integrating state. Also, at the test occasion, it could be observed that the octocopter performed more responsive with the integrating states. Besides the mentioned noises and disturbances in section 6.1, one can see that now the yaw-rate does follow the reference input without the need for any trimming, compare to Figure 6.4. From Figure 6.7 at about $t = 0 - 40$ s it can be seen how the reference input in yaw-rate is trimmed down to a neutral level. Also, the control inputs are kept within their defined boundaries.

Chapter 7

Discussion and Future Work

During the thesis report, analyses and discussions are continuously presented. In this chapter, these discussions are supplemented by future thoughts and hypothesis. The discussion held here also includes and proposes future work and solutions to difficulties and problems encountered during the thesis project. The discussion is divided into sections covering the chapters 2-6, including relevant topics.

7.1 The Octocopter

During the thesis project, extensive hardware modifications and implementations where carried out, both mechanical and electrical. The greatest difficulties encountered where the implementation and programming of the Arduino and the GC Raspi, sections 2.1.1-2.1.2. One of the greater challenges when implementing the Arduino was to establish a stable connection via I2C to the octocopters PDB, section 2.1.7. This difficulty partially arose due to lack of insight in how the PDB distributes the I2C busses. Furthermore, the hardware platform from which the project based on, the Okto XL 4S12 [18], was not ideal. The platform is quite expensive and uncommonly used in these types of projects. This makes it more difficult to get hold of spare parts and to gather insight in how the platform is designed. Also, the Okto XL 4S12 is quite large compared to other multirotors, a smaller platform would allow for easier handling and testing of the vehicle. In conclusion, this type of project would benefit from using a smaller, inexpensive and popular used open-sourced multirotor platform.

One of the main design features of the platform is the ability to generate C code from Simulink, responsible for the control of the octocopter. Quite a few problems arose when configuring the GC Raspi to enable this feature. Without going into detail about specific problems, the main issue was the poor insight in how Simulink converts a Simulink model to corresponding C code. The concept of generating C code from a high level programming language such as Simulink is great, but it also reduces the insight and understanding of what exactly is generated and run on the platform. Better understanding in how Simulink generates code is required, otherwise own written C code is preferable.

When deriving control strategies that are meant to be implemented on real hardware and not just on paper, it is of importance to have good knowledge and insight in the system and its properties. In order to measure and estimate these properties, it is crucial to use well motivated methods and carry out accurate measurements. The estimation of the motor constants c_T and c_Q is crucial, section 2.1.10. They should be accurately measured in order to maximize the performance of the tuned controllers. The used estimations of the motor constants where the ones determined during the last year's thesis. When comparing the required (analytically determined) and the actual motor commands for lift off, it was found that there was a significant difference. This raises suspicion that the estimated motor constants are off and perhaps require better estimation. Nevertheless, it would be beneficial to use a platform were some of these important properties are known or at least are well documented e.g., a data sheet corresponding to the motors.

7.2 Modeling

The modeling of the octocopter is crucial and has to be handled with care. The use of Euler angles and quaternions is quite common when expressing rotations in 6DoF. Euler angles are more intuitive than quaternions, but by using quaternions, the possibility for singular points can be eliminated. It would be interesting to investigate how it would be possible to use quaternions when controlling a vehicle in 6DoF. In theory, this would allow the vehicle to have any rotational position without worrying about singularities. The presented method in this thesis of using quaternions could be applied to directly control on quaternions, which could be of interest in future projects.

The modeled body forces and torques in section 3.4 are determined to be among the most significant ones. Other prominent effects which are of interest, are aerodynamics. As previously mentioned in section 3.4, extensive modeling regarding aerodynamics can be made. In this thesis only aerodynamic drag is modeled. In order to analyze and model other aerodynamic effects, experiments measuring properties of the vehicle have to be setup and performed. These experiments demand much care and preparation and would probably need an entire thesis project worth of time. Although, by having more knowledge of the vehicle's aerodynamics, better control algorithms could be derived.

At the end of the modeling chapter, the derived nonlinear model is linearized about hovering, see section 3.7. When applying a first order T.E, all nonlinearities disappear and are not compensated for in a linear control. A solution to this would be to try to control the full nonlinear model. But since it is quite complex, it would not be as straight forward as the application of a linear controller. To simplify control, the nonlinear model can be approximated by a simpler nonlinear model. For example, the rotation matrices contain the nonlinear trigonometric functions cos and sin. These nonlinear functions could be approximated by the small angle assumption which would simplify control, note that the resulting expression would still be nonlinear. The control of such a simplified nonlinear model would be able to compensate and reduce cross coupling effects which can be observed in chapter 5. Furthermore, it is also important to determine which nonlinearties should be compensated for. It might be desirable to leave some nonlinearites uncompensated. For example, a pilot-based controller might benefit from letting the pilot be responsible for the compensation of disturbances due to wind. Futhermore, in order to compensate for e.g., aerodynamics it is crucial to have good knowledge and insight, otherwise compensation could cause more problems than advantages.

7.3 Control

As discussed in section 4.1, it is of importance to have strategies of converting the model control inputs to corresponding motor commands, especially when developing a complete multirotor. It is crucial to define boundaries on the control inputs. The performance of a controller is strongly related to how much control can be achieved. The boundaries in this project are based on the available control about hovering. In order to increase the available control input about hovering, the force u_z can momentary be increased to allow for larger control inputs in torques. However by adjusting u_z, unexpected behaviors might follow. If for example the vehicle is descending, a rapid change of its rotational orientation will momentary increase its altitude. These types of effects have to be known by e.g., the pilot. Furthermore, from Figures 4.2-4.4 it can be seen that increasing u_z will after some point not increase the available torque, on the contrary, decrease it. Thus, it is concluded that the derived approach in section 4.1 performs well, even in critical situations and also increases robustness and safety.

The derived pilot-based and attitude linear state feedback controllers showed very good performance, even though the system in need of control is highly nonlinear. By using an observer and compensating for the rise times in the motor dynamics, performance is increased. In section 7.2 a simplified nonlinear model is discussed, which if controlled, could reduce the cross coupling effect observed in chapter 5. There are numerous nonlinear control methods which could be of interest, but how applicable they are, depend on the system in need of control. Using some Lyapunov based method such as back-stepping as already been proposed in research articles and may be of interest. It would also be of interest to investigate how a quaternion based controller could be used. A controller based on quaternions would

allow for any rotational position, hence acrobatic motions such as loopings could be performed. However, this would also require another type of platform. In conclusion, in order to improve the pilot-based and attitude controller, the use of a nonlinear control method should be investigated.

Concerning position control, the method of combining the Lyapunov based and the attitude controller showed promising results. With the current position controller, it is not possible to change yaw. In order to change yaw, the equations of converting the position control inputs to attitude references have to be altered. Since it is shown that the control of yaw is slower than roll and pitch, it is reasonable to let the yaw reference be given to the position controller, similar to the translational position references. This would also make sense if e.g., the octocopter carries a camera. Furthermore, when designing the position controller it is assumed that no rise time in the system exists. To enhance the performance of the controller, the rise time in the attitude controller could be compensated for, similar to the compensation of the motor dynamics. This would reduce the delay in the position controller observed in chapter 5. Also, to improve robustness, an integrating state can be added, compare to the attitude controller. This would then require the implementation of an anti-windup.

7.4 Simulation

To simulate the complete nonlinear system with disturbances, MATLAB's simulation tool Simulink is used. Simulink comes with a lot of predefined functions and the visual block structure, see Figure 5.1, gives a great overview of the simulation model. By using Simulink to its full capacity e.g., mask programming, flexibility is added and simulations are simplified. As mentioned in the beginning of chapter 5, a visualization of the octocopter in 6DoF is made using MATLAB's `plot` function. The vehicle moves in real time and can be controlled by e.g., using a pc gaming controller which allows for great testing and training. In order to improve the visualization, a next step would be to use a third-party software supporting Simulink which can visualize objects in a realistic environment. The Simulink model would compute and send relevant position data which the third-party software would use to visualize the octocopter in a suitable environment. This would improve training and enhance visualization.

The simulation of the observer-based states feedback controllers, pilot-based and attitude, showed promising results. The controllers behaved as desired even with added disturbances. The disturbances are well rejected and did not affect the states to greater extent. As mentioned in section 7.2, the linear controllers do not compensate for coupling effects between the states. These effects can be observed in the simulations, but even with the largest allowed changes in reference, these effects did not have any major contribution. However, it would be of interest to design a controller that could cancel or reduce these effects.

The derived position controller in section 4.4 did manage to maneuver the octocopter in a desired motion. However, during the simulation, continuous references are used which resulted in a delay between the actual position and the reference. Although the octocopter managed to produce a desirable motion, it would be of interest to design the position controller such that continuously reference inputs could better be used. The discussed compensation for the rise times in attitude controller would also help, see section 7.3. When simulating the position controller with disturbances, large levels of noise in the control inputs can be observed. Even though the controller manage to control the vehicle, these large noise levels should be reduced. More about the noise and methods for reducing them are presented in section 7.5.

During all simulations the priory algorithm derived in section 4.1 is applied. The method showed good performance and managed to handle critical situations. Thus, it can be concluded that it is not always necessary to be too conservative when deriving control boundaries, as long as there exists a solution to handle critical situations.

7.5 Evaluation of Flying Data

The possibility to test and evaluate derived control algorithms on a real system requires a lot of preparation but is also very rewarding. As mentioned in chapter 6, the last flight ended with a crash. From this experience a lot can be learned. It is important to simulate and test derived control algorithms to there limits before they can be applied on real hardware. It is also crucial to test implemented controllers and functions under controlled circumstances e.g., fixate the octocopter and test how the vehicle behaves under critical situations. Also, when testing the vehicle for the first times without any mechanical safety features, the testing location and procedure have to be evaluated with care. Open areas and only with relevant personnel is to prefer. This reduces the risk of damage to the vehicle and other objects.

When comparing the simulations to the real logged data, it is quite clear that they do not match. The real data does experience much noise and the reference is not tracked as well as in the simulations. As previously mentioned in chapter 6, the octocopter did perform well and the pilot was able to maneuver the vehicle as desired. In order to improve control, the high noise levels in the used states have to be reduced. The IMU and GPS are mounted on arms which vibrate due to the rotating motors, see section 2.1.3. A first step to reduce vibration noise would be to move the IMU and GPS to positions which are less vulnerable to vibrations. Secondly, the parameters for the EKF in the Nav Raspi are not tuned for the octocopter. For instance, filtering of high frequency content might be beneficial, i.e., if it is assumed that the octocopter only changes its angular-rate at maximal e.g., $|30|$ rad/s, content above ≈ 5 Hz could be considered as noise.

Another type of disturbance is the quantization of the reference signal. This effect can be observed when looking closely at e.g., ψ_r in Figure 6.2. Small fluctuations can be observed which are due to the quantization of the PPM signal sent from the radio controller. During the project it is decided to convert these signals to an array of bytes, one byte for each channel. By doing so, information is lost which will produce fluctuations in the reference. Even though this disturbance can be seen as rather insignificant, a larger resolution of the PPM signal would be beneficial. The quantization error can be reduced by letting e.g., two bytes represent one channel, this would enhance the resolution by a factor of 255 i.e., using a 16-bit unsigned int.

Also, estimations and measurements have to be accurate to obtain the desired performance from the controllers. Some properties are more important than others. As earlier mentioned in section 7.1, the estimations of the motor constants c_T and c_Q are off which might negatively affect performance. Therefore, it would be of interest to simulate the system were important parameters are varied. This would give more insight in the robustness of the controllers and increase knowledge of the importance of different parameters, thus requiring extra attention and accurate estimation.

Lastly, it is difficult to compare the logged data to the simulations since the references are not the same. In order to properly evaluate the true system, it would be of interest to send steps in the reference, similar to the ones in the simulation. This would either require a spacial area or a mechanical construction which e.g., would enable the octocopter to rotate about its own axes. Another alternative is to simulate the system with the reference from the radio controller and compare the simulated result to the real data.

Appendix A

Experimental Results

Experimental result of measuring the angular-rates of a rotating motor with mounted plastic propeller for different motor commands given in Table A.1.

u_M [byte]	ω_M [rpm]	u_M [byte]	ω_M [rpm]
0	0	90	3970
1	680	100	4280
2	790	110	4590
3	900	120	4910
4	1000	130	5200
5	1080	140	5480
6	1160	150	5750
7	1230	160	6030
8	1300	170	6230
9	1360	180	6530
10	1410	190	6680
20	1860	200	6850
30	2210	210	7050
40	2520	220	7400
50	2800	230	7600
60	3070	240	7750
70	3340	250	7780
80	3640	254	7950

Table A.1: Experimental result of measuring the angular-rates of a rotating motor with mounted plastic propeller for different motor commands. u_M is the sent motor command and ω_M the corresponding measured angular-rate.

Experimental result from moment of inertia estimation of the motor with mounted plastic propeller about its z-axis given in Table A.2.

N	t [s]	T [s]	$I_{M_{zz}}^{\text{plastic}}$ [10^{-4} kgm^2]
10	8.7	0.87	1.16
10	8.6	0.86	1.14
15	12.8	0.85	1.12
15	12.9	0.86	1.14
20	17.0	0.85	1.11
20	17.1	0.86	1.12
25	21.2	0.85	1.11
25	21.3	0.85	1.12

Table A.2: Experimental result from moment of inertia estimation of the motor with mounted plastic propeller about its z-axis. N is the number of periods, t total time, T period length and $I_{M_{zz}}^{\text{plastic}}$ the estimated moment of inertia.

Experimental result from moment of inertia estimation of the motor with mounted carbon propeller about its z-axis given in Table A.3.

N	t [s]	T [s]	$I_{M_{zz}}^{\text{carbon}}$ [10^{-4} kgm^2]
10	8.3	0.83	1.06
10	8.4	0.84	1.09
15	12.6	0.84	1.09
15	12.6	0.84	1.09
20	16.5	0.83	1.05
20	16.6	0.83	1.06
25	20.7	0.83	1.05
25	20.7	0.83	1.05

Table A.3: Experimental result from moment of inertia estimation of the motor with mounted carbon propeller about its z-axis. N is the number of periods, t total time, T period length and $I_{M_{zz}}^{\text{carbon}}$ the estimated moment of inertia.

Experimental result from moment of inertia estimation of the octocopter about its x-axis given in Table A.4.

N	t [s]	T [s]	I_{xx} [kgm^2]
10	11.58	1.158	0.1397
10	11.42	1.142	0.1359
15	17.01	1.134	0.1340
15	16.94	1.129	0.1329
20	22.63	1.132	0.1334
20	22.61	1.131	0.1332
25	28.32	1.133	0.1337
25	28.30	1.132	0.1335

Table A.4: Experimental result from moment of inertia estimation of the octocopter about its x-axis. N is the number of periods, t total time, T period length and I_{xx} the estimated moment of inertia.

Experimental result from moment of inertia estimation of the octocopter about its y-axis given in Table A.5.

N	t [s]	T [s]	I_{yy} [kgm^2]
10	11.60	1.160	0.1282
10	11.54	1.154	0.1269
15	17.50	1.167	0.1297
15	17.59	1.173	0.1310
20	23.29	1.165	0.1292
20	23.10	1.155	0.1271
25	29.32	1.173	0.1311
25	29.20	1.168	0.1300

Table A.5: Experimental result from moment of inertia estimation of the octocopter about the y-axis. N is the number of periods, t total time, T period length and I_{yy} the estimated moment of inertia.

Experimental result from moment of inertia estimation of the octocopter about its z-axis given in Table A.6.

N	t [s]	T [s]	I_{zz} [kgm^2]
10	21.96	2.196	0.2233
10	21.81	2.181	0.2203
15	32.77	2.185	0.2210
15	33.26	2.217	0.2277
20	43.69	2.185	0.2210
20	44.25	2.213	0.2267
25	54.15	2.166	0.2173
25	54.60	2.184	0.2209

Table A.6: Experimental result from moment of inertia estimation of the octocopter about its z-axis. N is the number of periods, t total time, T period length and I_{zz} the estimated moment of inertia.

Bibliography

[1] Arduino. Ardunio Nano.
www.arduino.cc/en/Main/ArduinoBoardNano. 2015-05-27.

[2] Biltema. Biltema Digital Tachometer.
www.biltema.se/sv/Bil—MC/Verktyg-och-Verkstadsutrustning/Testinstrument-och-Elektronik/Varvtalsraknare-laser-2000023356/. 2015-05-25.

[3] Biltema. Biltema PE-braided fishing line 200 m/0.21 mm/9 kg.
www.biltema.se/sv/Fritid/Fiske/Fiskelina/Fiskelina-2000024298/. 2015-05-25.

[4] D-Link. D-link Wireless N 150 Micro USB Adapter DWA-121.
www.dlink.com/se/sv/home-solutions/connect_us/adapters/dwa-121-wireless-n-150-pico-usb-adapter. 2015-05-27.

[5] Eugene Butikov. *Precession and nutation of a gyroscope.* St Petersburg State University, St Petersburg, Russia. IOP Publishing Ltd. 2006.

[6] Graupner HoTT 2.4 Systems. Graupner GMBH & CO.KG, Henriettenstraße 94 - 96, D-73230 Kirchheim/Teck, Germany.

[7] Graupner MC-32 HOTT - Programming Manual. Graupner GMBH & CO.KG, Postfach 1242, D-73220 Kirchheim/Teck, Germany.

[8] Henrik Ohlsson and Hrvoje Corluka. *Modellering och reglering av en oktakopter.* Department of Automatic Control, Lund University, Lund, Sweden. 2014.

[9] Human Benchmark. Reaction Time Statistics.
http://www.humanbenchmark.com/tests/reactiontime/statistics. 2015-07-09.

[10] James Diebel. *Representing Attitude: Euler Angles, Unit Quaternions, and Rotation Vectors.* Stanford University, Stanford, California 94301-9010. 20 October 2006.

[11] Karl Henrik Johansson, Bo Wahlberg and Elling W.Jacobsen. *EL2620 Nonlinear Control - Lecture notes.* Dep. of Automatic Control, KTH, Stockholm, Sweden, 2014.

[12] Kristoffer Bergman and Jonatan Ekström. *Modeling, Estimation and Attitude Control of an Octorotor Using PID and L1 Adaptive Control Techniques.* Department of Electrical Engineering, Linköpings universitet, Linköping, Sweden. 2014.

[13] KTH, Royal Institution of Technology. Elements of Statistical Learning, Lecture Notes 2012.
http://www.csc.kth.se/utbildning/kth/kurser/DD3364/Lectures/KKT.pdf. 2015-08-03.

[14] Markus Mikkelsen and Oscar Oscarson. *Handover Document for Octocopter.* Saab Dynamics, Linköping, Sweden. 2015.

[15] Matt R. Jardin and Eric R. Mueller. *Optimized Measurements of UAV Mass Moment of Inertia with a Bifilar Pendulum.* AIAA Guidance, Navigation and Control Conference and Exhibit 20 - 23 August 2007, Hilton Head, South Carolina. August 2007.

[16] MikroKopter. Mikrocopter BL-Ctrl V2.0.
www.mikrocontroller.com/index.php?main_page=product_info&cPath=69&products_id=504.
2015-05-26.

[17] MikroKopter. Mikrocopter 12×3.8" CFK.
www.mikrocontroller.com/index.php?main_page=product_info&cPath=75&products_id=613.
2015-03-23.

[18] MikroKopter. Mikrocopter OktoXL 4S12.
wiki.mikrokopter.de/en/ArfOktoXL. 2015-05-26.

[19] MikroKopter. Mikrocopter EPP1245.
www.mikrocontroller.com/index.php?main_page=product_info&cPath=75&products_id=255.
2015-03-23.

[20] MikroKopter. Mikrocopter MK3638 5mm.
www.mikrocontroller.com/index.php?main_page=product_info&cPath=73&products_id=615.
2015-03-23.

[21] MikroKopter. Okto XL PDB V8.
https://www.mikrocontroller.com/index.php?main_page=product_info&cPath=77&products_id=612.
2015-05-26.

[22] MikroKopter. VISLERO LiPo 6000 mAh 4S 20 C 14.8 V.
www.mikrocontroller.com/index.php?main_page=product_info&cPath=87&products_id=533
&zenid=cvqrvntl3cif35tg7gi976f380. 2015-05-26.

[23] Ranks of matrices and the Rouché-Capelli Theorem. Number of solutions of a linear system.
http://venus.unive.it/ tolotti/RoucheCapelli_GSEM.pdf. 2015-07-09.

[24] Raspberry Pi Foundation. Raspberry Pi 2 Model B.
www.raspberrypi.org/products/raspberry-pi-2-model-b/. 2015-05-27.

[25] Torkel Glad and Lennart Ljung. *Control Theory - Multivariable and Nonlinear Methods*. CRC Press, Taylor & Francis Group, 2000.

[26] u-blox. u-blox EVK-6T.
u-blox.com/en/evaluation-tools-a-software/gps-evaluation-kits/evk-6-evaluation-kits.html. 2015-05-26.

[27] xsense. xesens MTi-100 IMU.
www.xsens.com/wp-content/uploads/2013/12/MTi-100-series2.pdf. 2015-05-26.

[28] WolframMathWorld. Quaternion.
mathworld.wolfram.com/Quaternion.html. 2015-06-07.

TRITA -MAT-E 2015:63
ISRN -KTH/MAT/E--15/63-SE