The Design and Implementation of an Adaptive Controller for a Quadcopter

by

Edgar Andries Niit

Thesis presented in partial fulfilment of the requirements for the degree of Master of Engineering (Mechatronic) in the Faculty of Engineering at Stellenbosch University

Supervisor: Dr. W.J. Smit

December 2017

The financial assistance of the National Research Foundation (NRF) towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not necessarily to be attributed to the NRF.

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2017

Abstract

The Design and Implementation of an Adaptive Controller for a Quadcopter

E.A. Niit

Department of Mechanical and Mechatronic Engineering,
University of Stellenbosch,
Private Bag X1, Matieland 7602, South Africa.

Thesis: MEng (Mech)

December 2017

The Solar Thermal Energy Research Group (STERG) from Stellenbosch University is attempting to reduce the cost of Concentrated Solar Power (CSP) plants. Introducing robotics into such CSP plants can assist in reducing the cost. An optimized calibration method currently being investigated by STERG is using a pair of quadcopters to calibrate the heliostats. The system requires that the quadcopter has a stable hover, despite the presence of disturbances. The aim of this project was to design and implement an advanced controller for a quadcopter. The advanced controller should improve on the existing controller and ultimately allow for a more stable hover.

Standard control laws have unique parameters that yield a certain response based on the model on which they are implemented. Changes in the model will result in changes in the response, yielding the need for new ideal parameters. Adaptive controllers have the advantage of reducing the number of control parameters to be tuned. Reducing the number of parameters can be beneficial, as obtaining the ideal parameters can become a time-consuming process. Model reference adaptive control (MRAC) is the control approach that is considered in this project. This approach has previously been implemented on a quadcopter by Achtelik [*Adaptive Control of a Quadcopter in the Presence of large/complete Parameter Uncertainties*, (2011)].

It was desired to implement the adaptive controller on the Pixhawk flight controller. The Pixhawk flight controller was chosen due to its capabilities when considering research projects. It also runs PX4 firmware which is part of an open source project. The designed controller should integrate well with the existing PX4 firmware to allow users still to be able to use the flight controller as before. In order to integrate the adaptive controller with the PX4 firmware some modifications to the approach followed by Achtelik *et al.* (2011) was required. This report focuses on the implementation of MRAC in PX4 firmware. This required the use of quaternions in the control loop as opposed to the common Euler angles. The mixer was also extracted from the adaptive law. The mixer refers to the part of the controller which translates moment commands to motor commands, according to the airframe being used.

From simulations it could be seen that quaternions showed a significant improvement in reference tracking when it came to simultaneous pitch, roll and yaw inputs. The adaptive controller was first evaluated against other controllers in simulation before testing it in practice. In practical flight, it was again evaluated against other controllers. Specifically devised tests were evaluated to test the reference tracking and disturbance rejection of the different controllers. The adaptive controller showed the largest improvement, when compared to the other controllers, in the disturbance rejection tests. Finally, an autonomous mission was flown with the newly designed adaptive controller and also with the original PX4 controller. This showed successful integration of the adaptive controller with the existing firmware. An improvement in reference tracking for the adaptive controller, as opposed to the PX4 controller, was also found.

Uittreksel

E.A. Niit
Departement Meganiese en Megatroniese Ingenieurswese,
Universiteit van Stellenbosch,
Privaatsak X1, Matieland 7602, Suid Afrika.

Tesis: MIng (Meg)
Desember 2017

Stellenbosch Universiteit se termiese sonkrag energie navorsingsgroep (STERG) probeer om die koste van gekonsentreerde sonkragaanlegte te verminder. Die gebruik van robotika in gekonsentreerde sonkragaanlegte kan help om die koste geassosieer met sulke aanlegte te verminder. STERG is tans besig om n alternatiewe kalibrasie metode na te vors om die proses te optimiseer en sodoende koste te verminder. STERG be-oog om twee kwadrotuie vir hierdie kalabrasie proses te gebruik. ’n Vereiste is dat die kwadrotuie baie stabiel in vlug moet wees. Dit moet ook steurings kan weerstaan. Die doel van hierdie projek was daarom om ’n gevorderde beheerstelsel vir ’n kwadrotuig te ontwerp en te implementeer. Die gevorderde beheerder moet op die bestaande beheerder verbeter en ’n meer stabiele vlug tot gevolg hê.

Standaard beheer wette het unieke parameters wat ’n sekere respons vir die stelsel bepaal, gebaseer op die model waarvoor die beheerder ontwerp is. Enige veranderinge in die model sal ’n verandering in die optrede van die stelsel tot gevolg hê. Nuwe beheer parameters word daarom benodig. ’n Voordeel van ’n aanpasbeheerder is dat dit die hoeveelheid beheer parameters wat aangepas moet word verminder. Dit kan voordelig wees om die aantal parameters te verminder aangesien dit ’n tydrowende proses is om die ideale waardes te bepaal. Model verwysing aanpasbeheer (MRAC) is die beheer benadering wat oorweeg word. Achtelik [*Adaptive Control of a Quadcopter in the Presence of large/complete Parameter Uncertainties*, (2011)] het voorheen hierdie beheer benadering op ’n kwadrotuig geimplementeer. Daar was die begeerte om MRAC op die Pixhawk vlugbeheerder toe te pas. Die Pixhawk vlugbeheerder was gekies omdat dit ’n ideale vlugbeheerder is vir navorsings projekte. Die rede hiervoor was omdat dit PX4 sagteware, wat oopbron is, kan proseseer.

Die aangepaste beheerder moes dus goed inskakel met die bestaande PX4 sagteware om te verseker dat die vlugbeheerder steeds soos van te vore gebruik kon word. Sekere aanpassings aan Achtelik *et al.* (2011) se benadering was dus nodig om suksesvolle integrasie te bereik. Die projek fokus op die implementering van MRAC in die PX4 sagteware. Dit het die gebruik van quaternions in plaas van Euler hoeke vereis. Die menger was ook uit die beheer wet gehaal. Die menger verwys na die deel van die beheerder wat die moment bevele omskakel in motor bevele afhangend van die raam opset.

Simulasie het 'n beduidende verbetering in verwysingsvolging getoon in die geval van gelyktydige rol, hei en gier bevele. Die beheerder was eers teen ander beheerders vergelyk in simulasie voor daar praktiese toetse gedoen was. Vir die praktiese toetse was die ontwerpte beheerder weereens met ander beheerders vergelyk. 'n Outonome missie was uitgevoer met die nuut onwerpte beheerder asook die PX4 beheerder. Die toets het gedui op suksesvolle integrasie met die PX4 sagteware. 'n Verbetering in verwysingsvolging was ook waargeneem toe die aanpasbeheerder met die PX4 beheerder vergelyk was.

Acknowledgements

I would like to express my sincere gratitude to all those who assisted throughout the duration of this project. Firstly I would like to thank Dr. Willie Smit from the Department of Mechanical and Mechatronic Engineering Department at Stellenbosch University. As supervisor he provided valuable and constructive suggestions during my MEng study.

Secondly, I would like to thank my family for constant support. Special gratitude is extended to Cyril Niit for hardware related construction, Ronelle Niit for filming of video's and Lizelle Niit for technical writing advice. Kayla Nell also assisted in filming on occasion.

Lastly, I would like to thank ARMSCOR and the National Research Fund (NRF) for providing financial assistance over the first and second year of my MEng studies, respectively.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

3DR	3D robotics
CSP	Concentrated Solar Power
DOF	Degree of freedom
ESC	Electronic speed controller
FC	Flight controller
FPV	First person view
GPS	Global positioning system
HIL	Hardware in the loop
IR	Infra-red
IDE	Integrated development environment
Lipo	Lithium-ion polymer
LQR	Linear quadratic regulator
MIT	Massachusetts Institute of Technology
MRAC	Model reference adaptive control
NDI	Non-linear dynamic inversion
PID	Proportional-integral-derivative
PWM	Pulse width modulation
RX	Receiver
TX	Transmitter
UAV	Unmanned areal vehicle
VRFT	Virtual Reference Feedback Tuning
VTOL	Vertical take-off and landing

Constants

g =	$9.81\,\mathrm{m/s^2}$
$\pi =$	3.142

Scalars

e	Tracking error	[−]
e_a	Error between desired and measured output	[−]
e_h	Hedging error	[rad/s]
$e_{R,cos}$	Cosine of quaternion angle error	[−]
$e_{R,sin}$	Sine of quaternion angle error	[−]
e_Φ	Angle loop reference model error	[rad]
I_{xx}	Quadcopter inertia around body frame x-axis	[kg.m^2]
I_{yy}	Quadcopter inertia around body frame y-axis	[kg.m^2]
I_{zz}	Quadcopter inertia around body frame z-axis	[kg.m^2]
J	Cost function	[−]
J_{TP}	Inertia of motor rotating component	[kg.m^2]
k_d	Derivative gain	[−]
k_p	Proportional gain	[−]
k_i	Integral gain	[−]
L	Angular momentum	[kg.m^2/s]
m	Quadcopter mass	[kg]
mc	Motor command	[−]
p	Angular velocity around x-axis in body frame	[m/s]
q	Angular velocity around y-axis in body frame	[m/s]
q_{0-3}	Quaternion values	[−]
r	Angular velocity around z-axis in body frame	[m/s]
t	time	[s]
T_d	Adaptive reference model time constant	[−]
k_a	PID attenuation factor	[−]
u	Velocity along x-axis in body frame	[m/s]
u_i	Control signal	[−]
U_i	Movement signal	[−]
v	Velocity along y-axis in body frame	[m/s]
w	Velocity along z-axis in body frame	[m/s]
x^B	Coordinate in body frame	[m]
x^E	Coordinate in earth frame	[m]
y^B	Coordinate in body frame	[m]
y^E	Coordinate in earth frame	[m]
z^B	Coordinate in body frame	[m]
z^E	Coordinate in earth frame	[m]
θ	Roll angle in earth frame	[rad]

λ	Adaptive rate gain	$[-]$
σ	Sigma modification gain	$[-]$
ϕ	Pitch angle in earth frame	[rad]
ψ	Yaw angle in earth frame	[rad]
ψ_w	Yaw weight	$[-]$
Ω_i	Individual propeller speed	[rad/s]
Ω_Σ	Combined propeller speed scalar for gyroscopic force	[rad/s]

Vectors and Tensors

A_d	Desired state space representation, A matrix
A_p	Plant state space representation, A matrix
B_d	Desired state space representation, B matrix
B_p	Plant state space representation, B matrix
C^B	Coriolis-centripetal matrix for body frame
C^H	Coriolis-centripetal matrix for H-frame
$CF_{\phi,\theta,\psi}$	Configuration vector
d	Disturbance vector
e_a	Error between desired and measured output
e_h	Hedging error
e_R	Error in quaternion rotation error
$e_{R,axis}$	Quaternion error vector scaled be the sine of the angle error
$e_{R,axis}$	Cross product matrix of $e_{R,axis}$
E^B	Movement signal coefficient matrix
f	Plant state space representation, non-linearities vector
F	Thrust vector
F^B	Thrust vector in the body frame
F^E	Thrust vector in the earth frame
G^B	Gravitational force vector in body frame
G^H	Gravitational force vector in H-frame
I_{xyz}	Inertia matrix
I	3x3 identity matrix
i_{vec}	Vector of ones
J	Combine rotation and transfer matrix
J_Φ	Combined rotation and transfer matrix
K_d	Derivative gain matrix
K_{ff}	Feed forward gain matrix
K_h	Hedging error gain matrix

K_i	Integral gain matrix
K_p	Proportional gain matrix
M^B	System inertia matrix
O^B	Gyroscopic propeller matrix for body frame
O^H	Gyroscopic propeller matrix for H-frame
P	Symmetric positive definite matrix
Q	Symmetric positive definite matrix
q_{lB}	Euler to quaternion matrix
$q_{\dot{r}+V}$	Error in pitch and roll rates
r	Reference angular rates in body frame
R	Rotation matrix
R_a	Rotation matrix describing actual rotation
R_d	Rotation matrix describing desired rotation
R_r	Rotation matrix after pitch-roll only rotation
R_{rp}	Rotation matrix after pitch-roll only rotation
$R_{\dot{r}+V}$	Rotation matrix for reference rates prior to hedging
R_{ss}	Rotation matrix from rates of zero
R_x	Rotation matrix from measured angular rates
$R_{x,d}$	x component of R_d
$R_{x,rp}$	x component of R_{rp}
$R_{z,d}$	z component of R_d
$R_{z,a}$	z component of R_a
T	Angle loop reference model time constant matrix
T_Φ	Transfer matrix from body to earth frame
u	Control command vector
U	Movement signal vector/Combined moments and forces in body frame
V	Linear velocity body frame vector
V_h	Pseudo control hedging
V_r	Pseudo control angle rates
W	Combined linear and attitude angle rates in the body frame
x_d	Desired output angular rates in body frame
x_p	Plant output angular rates in body frame
α	Plant state space representation, non-linearities inertia matrix
Γ	Position of body frame w.r.t earth frame
ζ	H-frame velocity vector
Θ	Adaptive matrix gain

Θ_d	Adaptive matrix gain for disturbances
Θ_r	Adaptive matrix gain for reference rates
Θ_x	Adaptive matrix gain for measured rates
Θ_α	Adaptive matrix gain for non-linearities rates
Λ	Combine thrust and moment vector
ξ	Control signal and movement command relationship
τ_B	Moment vector in body frame
τ_E	Moment vector in earth frame
Υ	Rate of adaption matrix
Φ	Attitude angles in the earth frame
Φ_r	Reference model angles in the earth frame
Φ_{sp}	Attitude angle set points in the earth frame
$\dot{\Phi}_{ss}$	Zero rates set points
Φ_x	Measured attitude angles in the earth frame
Ψ	Combine position and attitude angles in the earth frame
ω	Angular velocity in the body frame
ω_r	Angular velocity reference values in body frame
ω_{sp}	Angular velocity set point values in body frame
ω_x	Measured angular velocities in body frame
Ω_v	Motor rotational speed vector

Operators

$\times$	Cross product
$d(.)$	Derivative
$\circ$	Dot product
$S(.)$	Skew-symmetric operator
$SF(.)$	Mixer
Tr	Trace operator

Chapter 1

Introduction

In a changing world where technological advancement is having an increasing effect in all spheres of society, autonomous vehicles are becoming a reality. Aerial vehicles are often favoured over ground vehicles due to their ability to fly over objects. This allows access to areas previously inaccessible to ground vehicles. They also utilise the previously unused space between ground-based structures and that used by airlines.

One such areal vehicle is quadcopters. Quadcopters are increasingly being used to automate tasks previously done manually. This improves efficiency and reduces cost. Quadcopters are naturally unstable and it is difficult to stabilize a quadcopter by manually controlling the motor speeds. A control system is therefore required to determine these motor speeds for the pilot based on how the pilot wants the vehicle to behave with regards to position or attitude. The control system used by quadcopters is the focus of this project. More specifically, the project focuses on the part of the control system that controls the attitude.

Many successful control systems for quadcopters have already been designed and implemented, the most popular being proportional-integral-derivative (PID) control. This is the most popular classical approach and has proven to be very successful. It is favourable due to its simplicity and effectiveness. The PID controllers on commercial quadcopters are, however, unique to a specific model. There are therefore no perfect single set of PID parameters that will allow for successful control of all quadcopters. Specific PID parameters need to be chosen and refined based on desired performance for a specific quadcopter. This process can become time-consuming; additionally, many novice pilots are not capable of making these refinements to their specific vehicle.

A method to reduce the number of control parameters to be chosen by the pilot is therefore desirable. An adaptive control system can achieve this. Having a controller whose parameters adapt to their ideal values, based on desired performance, eliminates the need for parameter tuning for different models.

Various adaptive approaches exist, some of which have previously been implemented on quadcopters. All currently existing adaptive approaches are still experimental and specific to particular problems. Code for these approaches is also not yet widely available. It is therefore desirable to design such a controller for use in one's own research group, and to integrate this controller with the existing firmware used by the group. The PX4 autopilot firmware, which runs on a Pixhawk flight controller, is used by the group. This project was primarily based on the work of Achtelik *et al.* (2011). Limited information about the controller was, however, available.

This chapter gives some additional background to the project and a more specific motivation. A brief overview of quadcopters is also provided. The objectives are given followed by an overview of the report layout.

1.1 Background

The uses of autonomous vehicles are manifold. This project can therefore have many applications. The original motivation for the project is, however, internal to a research group. One of the aims of the Solar Thermal Energy Research Group (STERG) from Stellenbosch University is to introduce autonomous vehicles into concentrated solar power (CSP) plants. Some background about the research group and its application of autonomous vehicles is therefore discussed.

Climate change is a global issue. Many countries rely on fossil fuels for electricity generation, however, burning of fossil fuels emits large amounts of green house gasses. For example, about one third of the greenhouse gasses that the US emits, comes from the generation of electricity (United States Environmental Protection Agency, 2017). This is one of the reasons why most countries are moving towards renewable electricity generation. Concentrated solar power is an attractive method of electricity generation, because it can store energy for many hours after the sun has set. CSP plants, however, have large costs associated with them. For example, the heliostats require high precision actuators and gearboxes in order to keep accurate mirror orientation and position. These contribute largely to the high installation cost.

STERG is attempting to reduce the cost of CSP plants (STERG, 2017). One possibility to reduce the cost of a heliostat is by using more affordable actuators and gearboxes on the heliostats and also by placing the heliostats on cheaper foundations. This reduction in cost, however, comes with a loss in precision. The loss in precision will have to be compensated for by calibrating the heliostats more often. The heliostat calibration is a lengthy process. An aim of STERG is therefore to optimise the calibration process by using a pair of quadcopters to calibrate the heliostats (Lock *et al.*, 2015).

Quadcopters, Figure 1.1, form part of a group of multirotor helicopter aircraft which generates thrust by means of spinning propellers. As can be deduced from the name, a quadcopter has four such propellers. The propeller speeds are adjusted to achieve its desired orientation. Quadcopters are already popular among hobbyists, but they are increasingly being applied in automation. Common uses include aerial photography, delivery of food or online orders, and assisting in rescue missions.

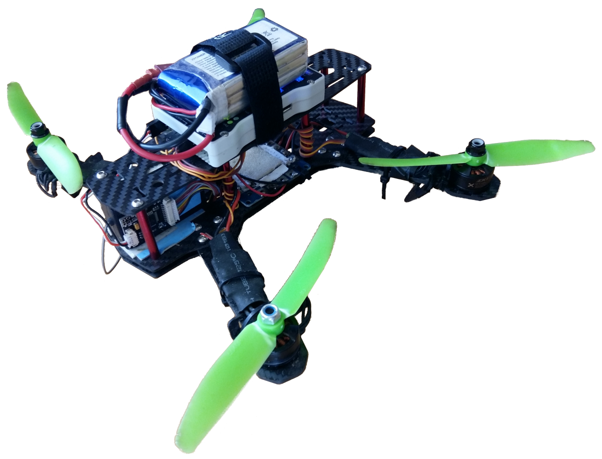


Figure 1.1: Basic quadcopter

1.2 Motivation

The control system implemented in the PX4 firmware is PID control, which uses fixed control parameters that do not change or adapt with time. This is the case with all open source flight controller software. For such control systems, the parameters need to be chosen specifically for a certain quadcopter to achieve desired performance. Manually obtaining these ideal values can become time-consuming, especially in research applications where the quadcopter model is regularly modified due to attachments such as lasers, gimbals and cameras. A self learning controller will therefore be beneficial.

Problems that arise when using quadcopters in the proposed calibration approach are that the quadcopters' hover needs to be very still and robust to disturbance rejection. The position of the quadcopter is also required to high accuracy. An additional benefit of an adaptive controller for a quadcopter is to improve on the existing PID controller's performance and ultimately allow for a more stable hover, even in the presence of disturbances.

1.3 Objectives and Scope

As mentioned in the motivation, it is required to improve on the existing PID controller implemented on the flight controller. The main area for improvement is to design a controller which removes the need for tedious parameter tuning. Improvements in disturbance rejection and tracking is also of importance. In order to ensure that the above is achieved, clearer objectives and boundaries need to be set. It was decided to focus on improving the attitude controller of the quadcopter. Since the position controller uses the attitude controller it will also benefit from improvements in the attitude controller.

The primary objective of the project is therefore that the controller should work on any quadcopter without the need for prior parameter tuning. Sub-objectives would then be that: the new controller should at least show some improvement when compare to the PX4 controller; the new controller should also seamlessly integrate with the existing firmware.

From the primary object it clear that it is desired to reduce the need for parameter tuning. The quadcopter should therefore be capable of flight, despite initial parameter values deviating from the ideal values. A second quadcopter, of different size, will also be used to test the new controller on different models. The same control system should therefore be capable of achieving flight on both quadcopters. The performance of the designed attitude controller should also be evaluated against the performance of the original attitude controller. In order to have a fair comparison the existing PX4 controller needs to be specifically tuned for the quadcopter it is to control. Ideal PID parameters for the PX4 controller therefore first need to be obtained in practical flight before the controllers can be compared. It is difficult to compare different controllers in general practical flight. Specific tests are therefore to be conducted to evaluate the success of the designed controller when compared to the original PX4 controller. A 20° step input is to be given around the roll axis. Secondly, an internal disturbance should be applied around the roll axis. Due to symmetry, improvement in the roll axis should be indicative of improvement around the pitch axis. A disturbance, or sudden change in model, that affects all axes is the final test to evaluate the controller success. The intention of the tests is to illustrate that the new controller improves on the existing, tuned, controller. The system plots can be used together with the root mean square error value to evaluate the performance of the different controllers.

An additional objective is that the designed controller should integrate well with the existing software. All the features of the existing software such as available flight modes should still be present after implementation of the designed controller. An autonomous mission and acrobatic flight will be flown in order to test for seamless integration with the existing software.

As mentioned it is desired that the designed controller will reduce the number of control parameters to be chosen by the pilot. This project therefore solely focuses on adaptive control approaches. Additionally, the project focuses on the attitude control of the quadcopter since this allows improvement in most applications. The reason for this is discussed later in the report.

1.4 Report Structure

Chapter 2 starts by explaining the basics of quadcopter dynamics. It then gives an overview of some control approaches followed by a review on previous studies done concerning quadcopter control in general and more specifically, adaptive control. Theory required for this project is also given. Chapter 3 briefly gives an overview of the quadcopter hardware used in this project, as well as the software used. The detailed quadcopter dynamics are given in Chapter 4. The control system design is given in Chapter 5, along with the simulation and practical results. Finally, Chapter 6 concludes the report.

Chapter 2

Literature Review

This chapter starts by discussing the history of quadcopters. Next an overview of how quadcopters work is provided along with the basic structure of the model. Chapter 4 considers the in depth mathematical modelling of the quadcopter. Quadcopters have become popular in recent years and since their introduction many different control approaches have been implemented and tested on them. Their instability also make them prime candidates to test the success of new control approaches. The following chapter therefore also informs the approaches taken to develop some of the existing controllers. The chapter concludes with the theory required for the adaptive controller design.

2.1 Quadcopter History

The invention of helicopters introduced the advantage of vertical take-off and landing (VTOL). Helicopters, however, require a tail rotor to counter the reaction torque created by the main rotor, and keep the aircraft facing forward. The additional rotor is inefficient since it does not provide any forward thrust or lift. This inefficiency is not present with quadcopters, which led to quadcopter research in the early 1900s. The first attempt was made by Jacques and Louis Breguet in 1907. Despite managing to take off, their quadcopter was very unstable. George de Bothezat built a quadcopter for the US army in the early 1920s. The system was complex and control difficulties required large efforts from the pilot, which led to the project being scrapped. In 1924 Étienne Oehmichen managed to fly a distance of 360 m with his quadcopter (Krossblade Aerospace, 2016; Quadcopter Arena, 2017). His quadcopter is depicted in Figure 2.1.

The above mentioned prototypes were built in a time before electric motors. A single mechanical engine was therefore required to drive the propellers of the first quadcopters. This resulted in a belt system. The individual propeller speeds could therefore not be varied. Running 4 propellers at the same speed

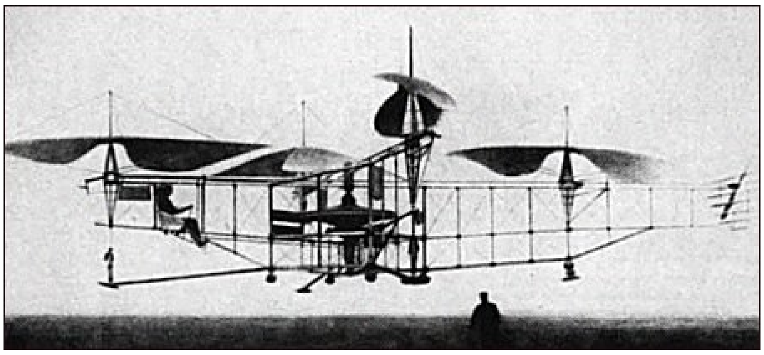


Figure 2.1: Oehmichen Quadcopter (Krossblade Aerospace, 2016)

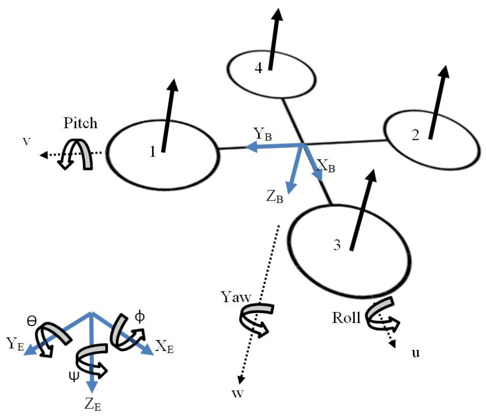
is not stable due to irregularities. Control therefore needed to be done by the pilot, which is very difficult for an unstable system. The difficulties associated with the control led to research on quadcopters being abandoned and the single rotor helicopter was favoured.

The recent improvement in electric motors and electronics allowed the concept of quadcopters to resurface around 2008. Each propeller is driven by its own motor allowing the speeds to be adjusted irrespective of one another. Sensors allow a microcontroller to constantly measure the quadcopter's orientation. These measurements are then processed by a control system which adjusts the motor speeds accordingly. The control system therefore eliminates the need for pilot stabilisation (Krossblade Aerospace, 2016).

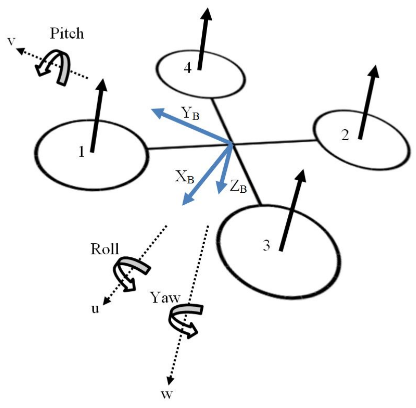
2.2 Basic Quadcopter Dynamics

This section provides a basic look into the quadcopter dynamics. The in-depth derivation of the system model can be found in Chapter 4. Figure 2.2 shows the basic structure of a quadcopter. The frame is composed of two perpendicular bars forming a symmetric cross shape. A motor is attached to the end of each bar and is used to drive the propellers which generate thrust. The electronics such as the controller, sensors and electronic speed controllers (ESCs) are also housed on the frame. Two coordinate systems are superimposed on Figure 2.2 (a) and are used to describe the quadcopter orientation.

The coordinate system on the quadcopter is called the body frame coordinate system while the coordinate system to the bottom left is called the earth frame coordinate system. The manner in which the body coordinate system is placed on the quadcopter can differ. If the axes align with the arms of the quadcopter as in Figure 2.2 (a), it is referred to as the plus-configuration. The coordinate system can also be rotated 45° around the z-axis of the body frame yielding the cross configuration; see Figure 2.2 (b). The cross configuration is more common and is also the configuration which will be used in this project.



(a) Plus configuration



(b) Cross configuration

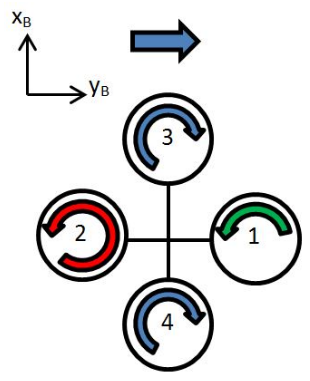
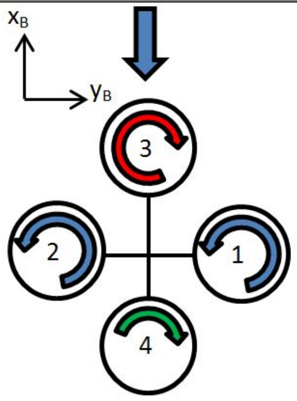
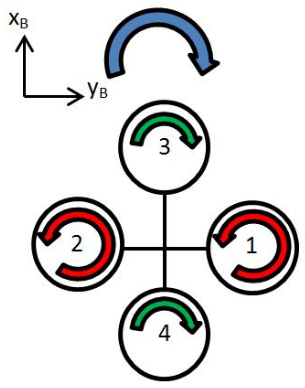
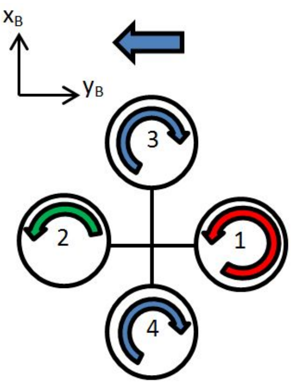
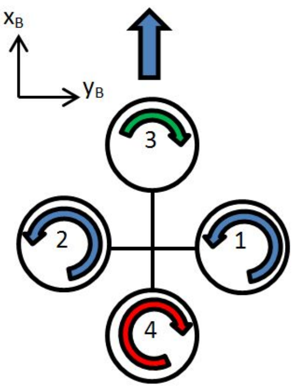
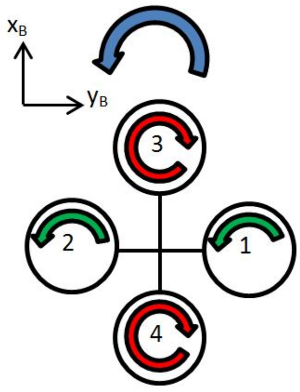
Figure 2.2: Quadcopter with coordinate systems

The plus configuration is, however, easier to understand and is therefore used to explain the basics of how quadcopter motion is achieved (Yilmaz, 2014).

The convention used in this explanation is the same as that used in the PX4 firmware (PX4, 2016*b*). Due to the coordinate system being attached to the quadcopter the thrust of all four motors will only exist in the z_B-direction. The centre of gravity will always be approximately in the centre of the quadcopter and the thrust from each motor will result in moments about the centre of gravity. Opposing motors therefore need to generate the same thrust in order to maintain the quadcopter orientation. For instance, the thrust from motor 1 and 2 should be equal in order to avoid rotation around the x_B-axis (roll). Rotating the quadcopter around the x_B and/or y_B axes also allows for translation. When the motor thrusts do not align with the z_E-axis, the thrust generated from the motors also have components along the x_E and y_E axes that push the quadcopter in that direction. For instance to move a quadcopter in the positive x_B direction a negative net moment is required around the x_B-axis. This is achieved if the thrust from motor 4 exceeds the thrust of motor 3.

A problem is that the torque applied to each propeller will result in a reaction torque, in the opposite direction, being applied to the quadcopter. This will cause rotation of the quadcopter about the z_B-axis (Hobden, 2015). The problem is solved by letting neighbouring motors rotate in opposite directions so that the torques applied to the quadcopter cancel one another out, in the case where they are of equal magnitude. If the thrust from neighbouring motors differ, rotation around the z_B-axis can be achieved. The motions described above are summarised in Table 2.1. The size of the arrows on the quadcopter represents the magnitude of the propeller speeds.

Table 2.1: Basic quadcopter dynamics summary

	Roll (around x-axis)	Pitch (around y-axis)	Yaw (around z-axis)
Positive			
Negative			

This project focuses on the control of the quadcopter attitude. Three moments control the attitude dynamics of the quadcopter. The dynamics are summarised in Figure 2.3. The propeller speeds, Ω, are related to thrusts which cause moments, τ, around the axes of the quadcopter. The goal of the controller is to calculate the moments in order to achieve the desired attitude. The moments cause angular acceleration, $\dot{\omega}$, and are related by means of the attitude dynamics. Angular acceleration causes angular rates, which in turn cause attitude angles, Φ. The angular rates and attitude angles are related by means of attitude kinematics.

$$\begin{tikzcd}
{} \arrow[r, "\Omega"] & \boxed{\text{Propeller Dynamics}} \arrow[r, "\tau"] & \boxed{\text{Attitude Dynamics}} \arrow[r, "\dot{\omega}"] & \boxed{\text{Attitude Kinematics}} \arrow[r, "\Phi"] & {}
\end{tikzcd}$$

Figure 2.3: Block diagram of dynamics breakdown

2.3 Control Approaches

The control of a quadcopter can be approached in different ways. The most common approach is to construct a mathematical model offline. An offline controller can then be designed for the mathematical model. The obtained controller is unique to the model, and therefore also unique to the quadcopter on which the model is based. An example of such an approach on quadcopters is a simple PID controller as done in the PX4 firmware (Ardupilot, 2016). After control implementation, a model can also be constructed online and an online controller is then designed based on the online model. These approaches are referred to as model based approaches. Indirect MRAC is an example of an online model based approach.

Measured data can also be used to directly find a controller without a model. This approach is referred to as a data-driven approach and can also be either online or offline. In the online cases the controller is designed in real time. An example of an online data-driven approach is Virtual Reference Feedback Tuning (VRFT) as done by Invernizzi *et al.* (2016). Combinations of the above two approaches also exist. Consider direct MRAC. A model is used to design a basic controller. The real time data is then used to update the control parameters directly without a model. The different approaches can be summarised by Figure 2.4. More information regarding the model based and data-driven approaches can be found in Hou and Wang (2013) and Hou and Xu (2009).

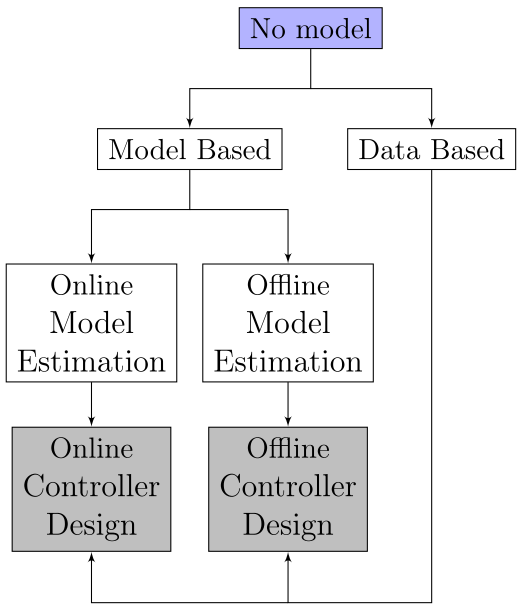


Figure 2.4: Overview of different control approaches (De Azevedo, 2014)

Simple offline controller design is discussed first. Simple PID control is considered due to it being the initial control approach used on quadcopters. The control approach implemented in the original PX4 firmware is also discussed. Some of the more advanced controllers are then discussed next, specifically adaptive controllers.

2.3.1 PID Control

The classic PID controller has the form given by equation 2.1. The error term, $e(t)$, refers to the difference between the set points and the measured outputs. The control variable, $u(t)$, refers to the output of the controller and input to the plant. It is adjusted based on the error in order to achieve the desired response.

$$u(t) = k_p e(t) + k_i \int e(t)dt + k_d \frac{de}{dt} \tag{2.1}$$

The integral and derivative of the tracking error is calculated as in the second and third term on the right hand side of equation 2.1, respectively. Different constants are used to give the terms different weightings. The constants are therefore altered to adjust the effect of each term. More information on PID control can be found at Control Tutorials (2016). The effect of the three terms on the closed loop system are summarized in Table 2.2.

Table 2.2: Effect of PID gains (Control Tutorials, 2016)

Constant	Rise time	Overshoot	Settling time	Steady-state error
P	Decrease	Increase	Small change	Decrease
I	Decrease	Increase	Increase	Eliminate
D	Small change	Decrease	Decrease	No change

The approach followed by Bresciani (2008) is used to illustrate the use of PID control on a quadcopter. Four of the degrees of freedom (DOF) are to be controlled: pitch, roll, yaw and height. The model used is given by equations 2.2-2.5. They were obtained by making two assumptions in order to simplify the full model; see Chapter 4 for a full model derivation. The first assumption neglects the effect of gyroscopic forces. The second assumption is that the T_Φ matrix approximates an identity matrix, which is true if the quadcopter is near its hover orientation.

$$\ddot{Z} = -g + (\cos\theta \cos\phi)\frac{U_1}{m} \tag{2.2}$$

$$\ddot{\phi} = \frac{U_2}{I_{xx}} \tag{2.3}$$

$$\ddot{\theta} = \frac{U_3}{I_{yy}} \tag{2.4}$$

$$\ddot{\psi} = \frac{U_4}{I_{zz}} \tag{2.5}$$

where g refers to gravitational acceleration in m/s^2. U_1 refers to the total propeller thrust in N, while U_2, U_3 and U_4 refer to the moments about the roll, pitch and yaw axes respectively.

The control is broken up into four steps. Firstly, the error in the attitude and height are used to estimate acceleration commands, by means of a PID controller, which are used to calculate the movement signals, u_i. Secondly, these movement signals are passed to a mixer which calculates the required propeller speeds. The motor dynamics are then used to relate the propeller speeds to motor voltages. Lastly the motor voltages are scaled to obtain pulse width modulation (PWM) values, that serve as motor commands (mc).

Figure 2.5 illustrates these steps with a typical feedback control loop, where step 1 is represented by the PID block and steps 2-4 are represented by the mixer block.

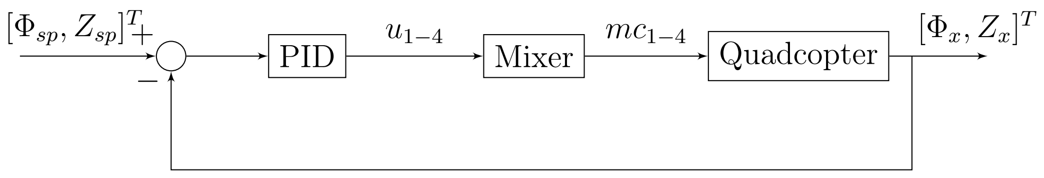


Figure 2.5: Block diagram of Bresciani's PID control approach

The first step is discussed further. Separate controllers are designed for all four states that need to be controlled. As mentioned the PID controller takes the error in a state to estimate an acceleration command. The commands are used to determine the movement signals (u) in a similar manner as which the actual accelerations are related to the actual moments/forces (U) in equations 2.2-2.5. The control of the four states is now discussed.

<u>Roll and Pitch Control</u>

As shown in Figure 2.6 the difference between the set point roll angle and the measured roll angle gives the error in the roll angle. PI control is then applied to the error, where K_I represents the integral gain and K_P the proportional gain. Derivative control is only applied on the plant output. This is done in order to avoid saturation of the motors when a step in the set point is given. The blue "SAT" block represents the saturator that is used to limit the integral value. This avoids excessive integral wind-up. The inertia around the

x-axis, I_{xx}, is used to relate the acceleration estimate to the movement signal, u_2. This relationship is similar to that of equation 2.3.

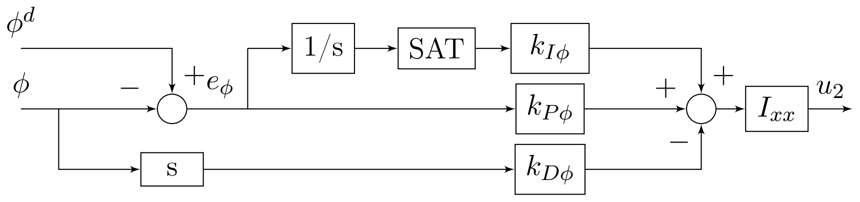


Figure 2.6: Detailed PID block for roll axis control (Bresciani, 2008)

Figure 2.7 shows that pitch control has the same structure as roll control. The only difference between the two is that the control signal is multiplied by the inertia around the y-axis, I_{yy}.

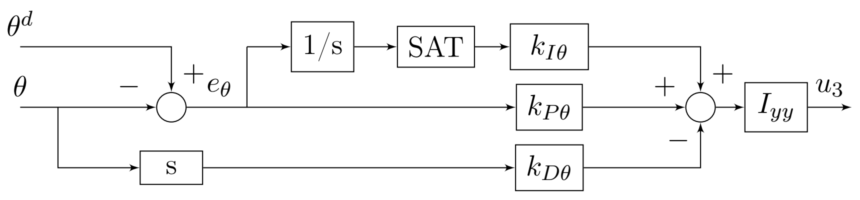


Figure 2.7: Detailed PID block for pitch axis control (Bresciani, 2008)

Yaw Control

As illustrated in Figure 2.8, the difference between the desired yaw angle and the measured yaw angle gives the error in the yaw angle. The desired yaw angle is, however determined by integrating a desired yaw angle velocity. This is done since it will not be possible to make more than one full rotation with an angle set point. The "FOLDER" block takes the discontinuity of the yaw angle from π to - π into account. The "SIGN COHERENCE" block is also used to avoid an error during the transition from π to - π. Further it is exactly the same as for the roll angle, except that the control signal is multiplied by the inertia around the z-axis, I_{zz}.

Height Control

The difference between the desired height and the measured height, as shown in Figure 2.9, gives the error in the height. The measured height used is obtained by fusing the measurements from the sonar and infra-red (IR) sensors. Further it is exactly the same as the attitude control, except that gravity is added to the control signal. The sum is then divided by the quadcopter mass and multiplied by the cosines of the roll and pitch angles as shown in equation 2.2.

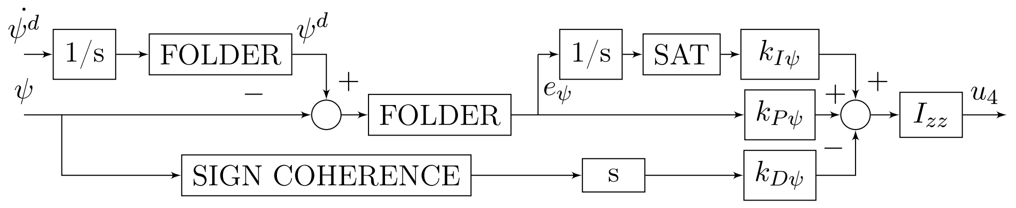


Figure 2.8: Detailed PID block for yaw axis control (Bresciani, 2008)

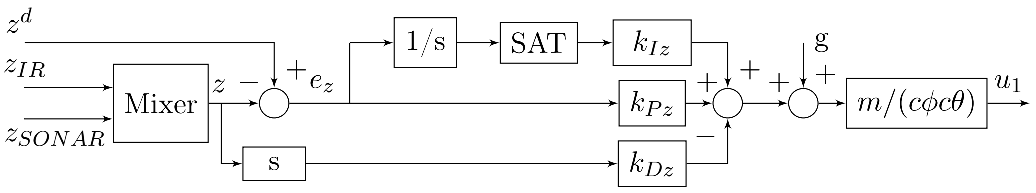


Figure 2.9: Detailed PID block for height control (Bresciani, 2008)

2.3.2 PX4 Attitude Control

The existing attitude controller in the PX4 firmware is cascaded. Two control loops are therefore used to determine a moment command to be sent to the mixer. The first, referred to as the angle loop, receives angles and determines angular rates. The second, referred to as the angle rate loop, receives angular rates and determines moment commands to be sent to a mixer. The two control loops are discussed separately. An overview of quaternions and rotation matrices is, however, required first.

Quaternions

The attitude measurements are passed to the controller as quaternions. A quaternion can be described by considering a rotation, α, around an axis. This is shown in Figure 2.10. The rotation axis, shown in blue, is described by three angles: β_x, β_y and β_z. The quaternion associated with this rotation is given by equations 2.6-2.9.

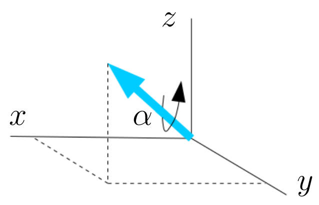


Figure 2.10: Quaternion illustration

$$q_0 = \cos\left(\alpha/2\right) \tag{2.6}$$

$$q_1 = \sin(\alpha/2)\cos\beta_x \tag{2.7}$$

$$q_2 = \sin(\alpha/2)\cos\beta_y \tag{2.8}$$

$$q_3 = \sin(\alpha/2)\cos\beta_z \tag{2.9}$$

In order to use quaternions to specify the attitude of a quadcopter, three rotations are considered: one around the z-axis (yaw), one around the y-axis (pitch) and one around the x-axis (roll). This order is known as the Body 3-2-1 sequence (Henderson, 1977). Applying equations 2.6-2.9 on each of the three axes of rotation in the above-mentioned order yields the relation between quaternions and Euler angles:

$$q_{IB} = \begin{bmatrix} \cos(\psi/2) \\ 0 \\ 0 \\ \sin(\psi/2) \end{bmatrix} \begin{bmatrix} \cos(\theta/2) \\ 0 \\ \sin(\theta/2) \\ 0 \end{bmatrix} \begin{bmatrix} \cos(\phi/2) \\ \sin(\phi/2) \\ 0 \\ 0 \end{bmatrix} \tag{2.10}$$

Applying quaternion multiplication yields (Mathworks, 2017a)

$$q_{IB} = \begin{bmatrix} \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\ \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \end{bmatrix} \tag{2.11}$$

Angle Loop

The attitude controller attempts to get two rotation matrices to match. A rotation matrix is used to relate one coordinate system to another, for further details on rotation matrices see Chapter 4. The Euler attitude set points are used to create a rotation matrix relating the earth frame to the desired body frame. The quaternion vector, describing the quadcopter orientation, is used to create a rotation matrix relating the earth frame to the actual body frame (Shoemake, n.d.).

The aim of this control loop is to align the actual and desired coordinate frames. Due to roll and pitch responses being faster than that of yaw, the controller tries to align the z-axes first by means of the shortest path. The pitch and roll change to achieve this is calculated. Now that the z-axes are aligned the yaw correction is determined. This is the angle between the two x-axes. This will not contribute much to a yaw error in manual mode since the yaw set point is set equal to the current yaw angle. The error will therefore simply be to maintain the existing yaw angle. Position control, however, generates a yaw set point. The reference rates are determined by applying proportional control on the error vector. Since the user input for yaw is still a rate, the yaw rate command is superimposed on the yaw rate determined in order to align the rotation matrices. The rates are constrained to remain within certain bounds. These rates are then used by the next control loop.

One advantage of the quaternion approach is that no discontinuity exists when the yaw angle goes from $-\pi$ to π. An improved response, when compared to the Euler approach, is also noticed in the case of simultaneous pitch, roll and yaw inputs.

Angle Rate Loop

The rates determined by the angle loop, ω_r, together with the measure rates, ω_x, are used to calculated an error vector. PID control is then applied according to equation 2.12.

$$u = K_p(\omega_r - \omega_x)k_a + K_d\omega_x s + K_i(\omega_r - \omega_x)\frac{1}{s} + K_{ff}\omega_r \tag{2.12}$$

The first term is the product of the errors, the proportional gains and an attenuation factor, k_a. The attenuation factor is used to adjust the proportional gains according to the current throttle value. After a certain throttle value the k_a value linearly decreases, from unity, at a constant gradient. The derivative control is applied only to the measured rates and not the error in the rates. The integral control term is removed if the term exceeds predetermined limits. The term only becomes active again once the values again lie within the limits.

A feed-forward term is also added which is simply gains multiplied by the rate set points. These values are set to zero by default for multirotors. The three moments are mixed together with the thrust command to obtain the motor commands. For more information see the mixer section in Appendix C.

2.3.3 Adaptive Control

Adaptive control can best be understood by looking at the biological equivalent (Narendra and Annaswamy, 2012). The dictionary defines adaption, in a biological context, as "a form or structure modified to fit a changed environment" (Dictionary.com, 2017). Extending this analogy to man made systems, an adaptive control system allows control parameters to be modified in order to function in a changing environment. This subsection looks at the history of adaptive control before discussing the basic concept further. The implementation of adaptive control on quadcopters concludes the subsection.

History of Adaptive Control

The following information is based on Ioannou and Fidan (2006). Adaptive controller research was first introduced in the 1950s. The motivation behind this research was the design of autopilot systems for agile aircraft such as fighter jets. The change in dynamics of these aircraft during flight could not be managed by simple traditional feedback systems. A controller that would be able to adjust for these changes in dynamics was therefore required. MRAC was the proposed solution. Online predictors and adaption laws were required in order to realise these adaptive controllers. Initially they were designed by means of the MIT rule and sensitivity methods. These design methods were, however, not supported by any theory, and stability could not be proven. The dangers associated with aircraft testing along with the lack of knowledge therefore led to adaptive control research to falter. The introduction of state space techniques and Lyapunov stability theory in the 1960s, however, allowed adaptive control research to be reintroduced. The adaption laws based on the MIT rule was redesigned by applying Lyapunov stability theory (Parks, 1966). These breakthroughs, along with advancements in technology, led to an increase in adaptive applications in the 1970s. Egardt (1979) found that the adaptive controllers of that time could easily become unstable in the presence of small disturbances. Research then took a new direction in the 1980s in order to achieve robust adaptive control. This led to Praly (1984) proving global stability despite the presence of of unmodelled dynamics. Research has since continued, with the latest advancement being that of L1 adaptive control (Hovakimyan and Cao, 2010).

Overview of Adaptive Control

Different types of adaptive controllers exist. MRAC will be discussed here. MRAC can be either be direct or indirect. The direct approach, considered here, directly determines the controller. The direct MRAC approach can be broken up further into two architectures, direct MRAC and direct MRAC with state predictor (Hovakimyan and Cao, 2010). They lead to the same error dynamics despite having a slight implementation difference. Both are discussed for comparative purposes. Simple direct MRAC is discussed first. The structure of such a controller is given by Figure 2.11.

An adaptive law is used to adjust the controller. The adaptive law uses the error in the actual model outputs and the reference model outputs together with the reference inputs to adjust the controller. The difference between the reference model and the actual model outputs should converge to zero. An online controller is therefore designed using a reference model determined offline.

The next method uses a state predictor instead of a basic constant reference model. Figure 2.12 shows the modification to Figure 2.11 when adding a state

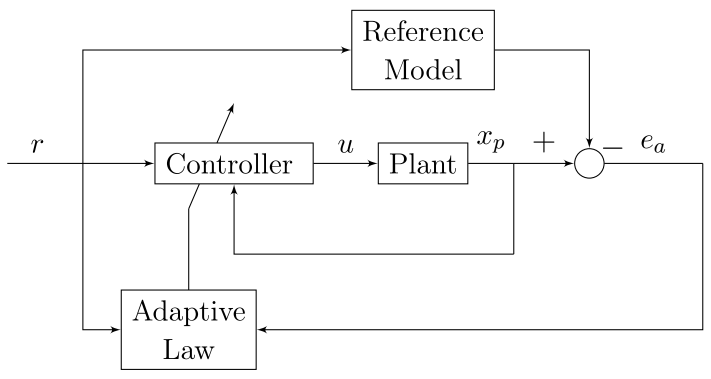


Figure 2.11: Direct MRAC without state predictor block diagram (Achtelik *et al.*, 2011)

predictor. The only difference is that the state predictor takes the controller output and adaption law outputs to predict the state whereas the reference model only takes the reference to determine the reference state (Hovakimyan and Cao, 2010). Since the same adaption outputs are used in the state predictor and in the controller, a similar structure is obtained. This is discussed in more detail in Chapter 5.

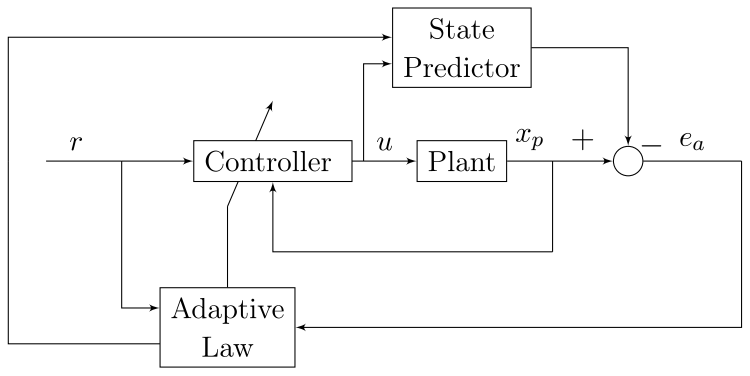


Figure 2.12: Direct MRAC with state predictor block diagram

Implementation of Adaptive Control on Quadcopters

With the improvement of electronics, quadcopters became a reality. Since their introduction they have become prime candidates for the testing of different control techniques. This is due to their low cost and small size. Their natural instability also provides a good test on the capability of the designed control system. It is difficult to determine an accurate model for a quadcopter. Varying weather conditions and other disturbance such as motor power loss can further aggravate this problem. The implementation of adaptive controllers on quadcopters is therefore ideal. This section will briefly discuss the type of adaptive controllers that have previously been implemented on quadcopters.

As was mentioned previously the attitude angle model of a quadcopter is non-linear and second order. Many of the adaptive controllers studied - Achtelik *et al.* (2011), Yilmaz (2014) and Wang (2015) - therefore make use of non-linear dynamic inversion (NDI) in order to simplify the control of the system. A more in-depth look at NDI, also known as feedback linearisation, can be found in Slotine and Li (1991). Wang (2015) also provides a good explanation of NDI.

The use of NDI yields a cascade controller. Figure 2.13 shows how the proposed controller links to the model discussed earlier in the chapter. The NDI loop takes the angle set points and determines angular reference rates required to achieve the set point angles. The angular rates are then controlled by the inner control loop. The momentum dynamics used to relate the angular rates to momentum inputs are single order. This simplifies the control. From Chapter 4 it can be seen that the attitude kinematics treated by the NDI loop do not contain any parameter uncertainties (Achtelik *et al.*, 2011). All parameter uncertainties are contained in the momentum dynamics: the inner control loop will therefore contain the adaptive component in order to compensate for these uncertainties.

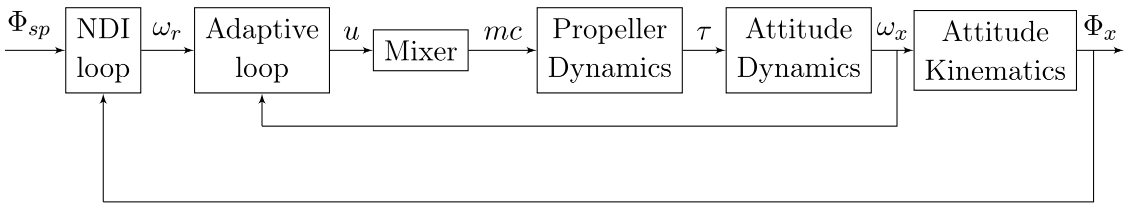


Figure 2.13: Closed loop block diagram to illustrate cascaded controller

The NDI loop used in Achtelik *et al.* (2011) and Wang (2015) have a similar structure to above. Yilmaz (2014), however, uses a second order reference model as opposed to a first order reference model. The reference model therefore also yields acceleration, but only the velocity is passed to the next loop.

Next, consider the adaptive law used within the angular rate loop. MRAC laid the foundation for L1 adaptive control and is therefore discussed first. Achtelik *et al.* (2011) uses Lyaponuv stability theory to design an adaptive controller for this loop. A similar approach was followed in this project, with modification to allow for integration with the PX4 firmware. Yilmaz (2014) replicates the work of Achtelik *et al.* (2011) in simulation. Modifications applied include the use of a second order reference model in the NDI loop and also using rate errors as opposed to angle errors as a correction term in the NDI loop.

In 2015 one of the co-authors of Achtelik *et al.* (2011) extended the concept for his PhD research, by looking at L1 adaptive control (Wang, 2015). For more information on L1 adaptive control see Hovakimyan and Cao (2010). Bertelsen and Thomsen (2014) also uses L1 adaptive control, but not with a cascaded approach as the others; a second order reference model is used to drive the adaptive law.

Additional adaptive control examples can be found in Michini and How (2009) and Michini (2007) which apply L1 adaptive control on indoor vehicles with assistance from Hovakimyan and Cao (2010); and Dydek (2010*a*) which uses MRAC on multiple quadcopters to achieve formation flight. Omidshafiei (2013) uses another learning method and uses a simplified quadcopter model, as in Chapter 4. Another approach that can be used for a learning method is neural networks as done in Nicol (2008). The above approaches require a plant estimate. One can also create an controller without the need of a mathematical model, as done in De Azevedo (2014) with VRFT. Additional literature concerning learning control approaches on quadcopters can be found in Komer (2015); Huynh *et al.* (2014); Dydek (2010*b*); Mohammadi and Shahri (2013); Huang and Liao (2015); Emran and Yesildirek (2014).

2.4 Theory

Two approaches for MRAC are considered in this report, the MIT rule approach and the Lyapunov approach. Although the Lyapunov approach will be used in the final controller due to being based on stability theory, the MIT rule approach will be designed alongside for comparative purposes. This section will look at the theory required for both approaches, first for the MIT rule approach, followed by the theory for Lyapunov approach. Most of the theory discussed in this section can be found in Astrom and Wittenmark (1994) and Dumont (2013).

2.4.1 MIT Rule Theory

MRAC orignated at MIT during the 1960s (Dumont, 2013). The initial adjustment of the adaptive parameters were done according to a rule, coined the MIT rule. As discussed previously, a reference model is used to generate the desired response of the system for a given set-point. The output of this reference model, the desired output (x_d), is compared to the plant output (x_p). The error between these values drives the adaption of the controller.

$$e_a = x_p - x_d \tag{2.13}$$

It is desirable to minimize the error. This will occur if the cost function in equation 2.14, which is a function of adaptive parameters (Θ), is minimized.

$$J(\Theta) = \frac{1}{2}e_a^2 \tag{2.14}$$

We want the value of the function to decrease as we adjust the parameters with time. The adjustment of the parameters with time should therefore yield a negative gradient of J.

$$\frac{d\Theta}{dt} = -\lambda\frac{\delta J}{\delta\Theta} \tag{2.15}$$

It is desired to know how J varies with time. J depends on the error, the error depends on Θ and Θ depends on time. The chain rule is therefore required twice.

$$\frac{dJ}{dt} = \frac{dJ}{de_a}\frac{de_a}{dt} = e_a\frac{de_a}{dt} \tag{2.16}$$

with

$$\frac{dJ}{de_a} = e_a \tag{2.17}$$

Applying the chain rule again to get the derivative of the error with respect to time yields

$$\frac{de_a}{dt} = \frac{\delta e_a}{\delta\Theta}\frac{d\Theta}{dt} \tag{2.18}$$

Substituting equation 2.15 into 2.18 yields

$$\frac{de_a}{dt} = -\frac{\delta e_a}{\delta\Theta}\lambda\frac{\delta J}{\delta\Theta} \tag{2.19}$$

In order to get the derivative of J, equation 2.14, with respect to the adaptive parameters the chain rule is required once more, since J depends on the error and the error depends on Θ.

$$\frac{\delta J}{\delta\Theta} = \frac{dJ}{de_a}\frac{\delta e_a}{\delta\Theta} = e_a\frac{\delta e_a}{\delta\theta} \tag{2.20}$$

Substituting equation 2.20 into 2.19 yields

$$\frac{de_a}{dt} = -\lambda\ e_a(\frac{\delta e_a}{\delta\Theta})^2 \tag{2.21}$$

Substituting equation 2.21 into 2.16 yields the desired derivative of J with respect to time. From equation 2.22 it can be seen that the derivative of J will be negative given a positive λ.

$$\frac{dJ}{dt} = -\lambda\ e_a^2(\frac{\delta e_a}{\delta\Theta})^2 \tag{2.22}$$

In order to determine the adaptive law substitute equation 2.20 into equation 2.15.

$$\frac{d\Theta}{dt} = -\lambda e_a \frac{\delta e_a}{\delta \Theta} \tag{2.23}$$

The $\frac{\delta e_a}{\delta \Theta}$ term will depend on the structure of of the reference model and controller.

2.4.2 Lyopunov Theory

The MIT rule does not guarantee a stable closed loop system, it is therefore necessary to introduce Lyapunov stability theory. A few definitions are required before the theorem can be discussed. These definitions, along with their proofs can be found in Astrom and Wittenmark (1994).

Consider the non-linear differential equation

$$\frac{dx}{dt} = f(x) \tag{2.24}$$

with initial conditions $f(0) = 0$. This initial condition yields $x(t) = 0$ as a possible solution to the equation. For a more detailed explanation see Appendix E.

A few assumptions regarding $f(x)$ are required in order to ensure that a unique solution exist. One such assumption is that $f(x)$ is locally Lipschitz in the neighbourhood of the origin. The Lipschitz definition is given below (Astrom and Wittenmark, 1994).

DEFINITION 1 Locally Lipschitz
A function is locally Lipschitz if

$$||f(x_2) - f(x_1)|| \leq L||x_2 - x_1|| \quad L > 0 \tag{2.25}$$

This essentially means that the absolute value of the gradient joining two points of the function, x_1 and x_2, should remain inside a given range.

$$0 < ||\frac{\Delta f(x)}{\Delta x}|| \leq L \tag{2.26}$$

In order to determine whether the solution to the differential equaiton is stable with respect to perturbation the following stability concept is required (Astrom and Wittenmark, 1994).

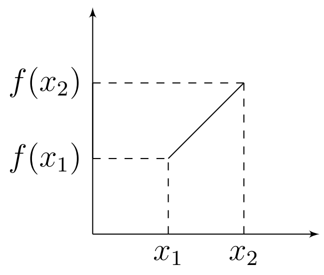


Figure 2.14: Lipschitz function illustration

DEFINITION 2 Lyapunov stability
$x(t) = 0$ is stable if for a given $\epsilon > 0$ there exists a number $\delta(\epsilon) > 0$ such that all solutions with initial condition

$$||x(0)|| < \delta \tag{2.27}$$

have the property

$$||x(t)|| < \epsilon \quad for \quad 0 \leq t < \infty \tag{2.28}$$

The solution is unstable if it is not stable. The solution is asymptotically stable if it is stable and δ can be found such that all solutions with $||x(0)|| < \delta$ have the property that $||x(t)|| \to 0$ as $t \to \infty$.

A graphic to assist in understanding the above definition is given by Figure 2.15. Consider a surface with variable height depending on location, x_1 and x_2 [1]. If a marble is let go somewhere on this surface it will roll around until an equilibrium position is obtained. Now consider the definition above. $x(0)$ is the location where the marble is let go and $x(t) = 0$ is the solution. If the marble is let go within an area specified by δ, containing the solution $x(t) = 0$, and the marble remains within an area defined by ϵ as time goes to infinity, then the solution is stable. If the solution is stable and if any starting point within δ causes the marble to reach the solution position as time goes to infinity, then it is asymptotically stable.

A function with unique properties can be used to investigate stability. Such a function is a positive definite function, defined as below (Astrom and Wittenmark, 1994; Al-Hokayem and Gallestey, 2015).

[1]The surface varying in height is purely used for illustration since it is an easy visual method to show what drives the change in x, this can also be done according to a differential equation.

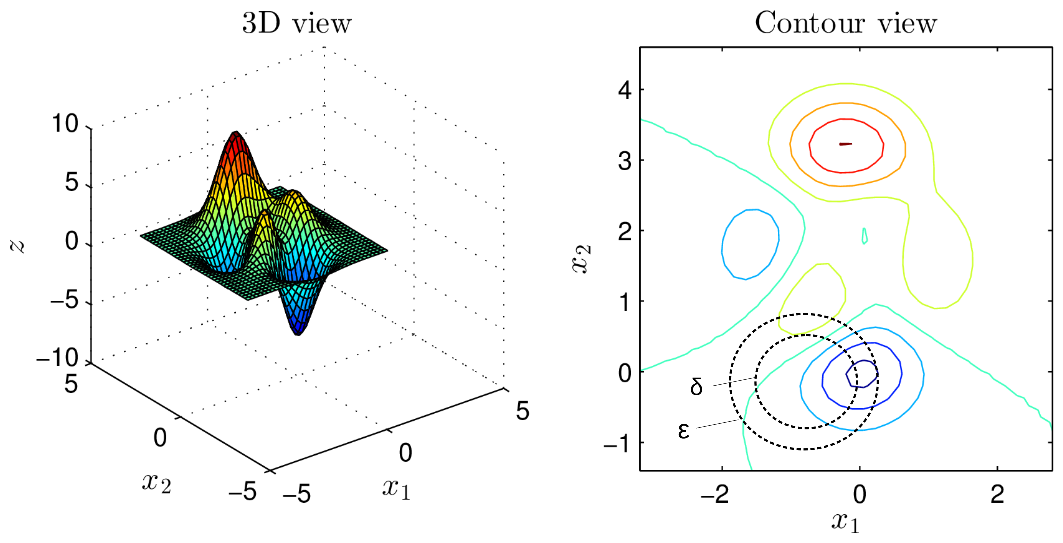


Figure 2.15: Graphical representation of Lyapunov stability

DEFINITION 3 Positive definite and semi-definite functions
A continuously differential function $V : R^n \rightarrow R$ is called positive definite in a region $U \subset R^n$ containing the origin if
1. $V(0) = 0$
2. $V(x) > 0,\ x \in U and\ x \neq 0$
A function is called positive semi-definite if condition 2 is replaced by $V(x) \geq 0$.

In order to visualise a positive definite function a top view of a function, where the curves represent equal values of the function, is given in Figure 2.16. The level curves of a positive definite function will enclose the origin and the smaller valued curves will be enclosed by larger value curves.

Now consider the differential equation as a vector. If $\frac{dx}{dt}$ always points towards the interior then the solution, x, that starts inside a certain level curve will remain inside that level curve, since it can only move closer to the origin with time; $\dot{x}$ therefore always points towards the origin. This yields the next theorem (Astrom and Wittenmark, 1994).

THEOREM 1 Lyapunov's stability theorem
If there exists a function $V : R^n \rightarrow R$ that is positive definite such that its derivative along the solution of the differential equation

$$\frac{dV}{dt} = \frac{\delta V^T}{\delta x}\frac{dx}{dt} = \frac{\delta V^T}{\delta x} f(x) = -W(x) \tag{2.29}$$

is negative semi-definite, then the solution $x(t) = 0$ to the differential equation is stable. If $\frac{dV}{dt}$ is negative definite, then the solution is also asymptotically stable. The function V is called a Lyapunov function for the differential equa-

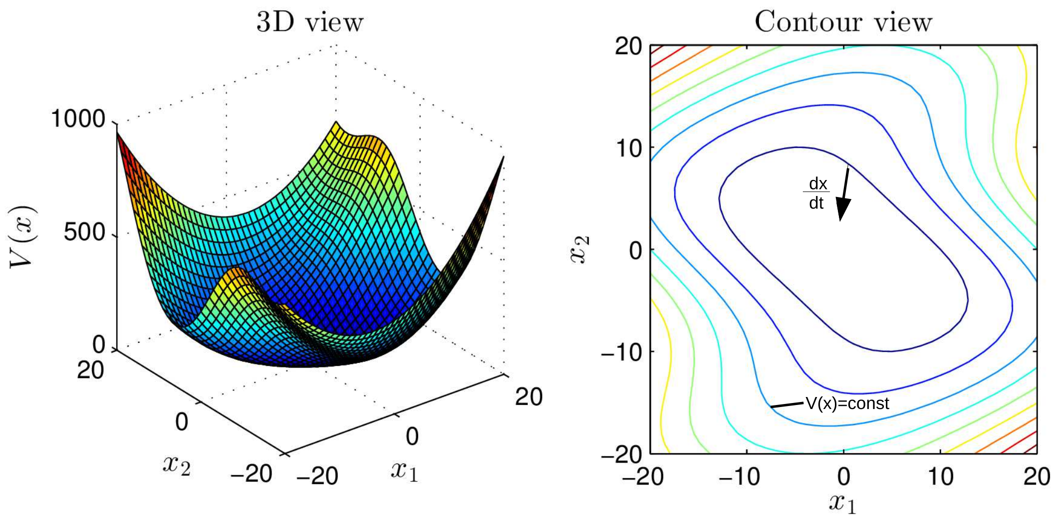


Figure 2.16: Graphical representation of a positive definite function

tion. Additionally if $\frac{dV}{dt} < 0$ and $V(x) \to \infty$ when $||x|| \to \infty$ then the solution is globally asymptotically stable.

A Lyapunov function is therefore required to analyse the stability of a system. A problem that arises is that there is no ecumenical procedure to obtain a Lyapunov function for a stable system. Some guessing is therefore involved. Quadratic functions generally provide a good starting point for possible choices. The next theorem can, however, be used to obtain a Lyapunov function in the case of linear systems (Astrom and Wittenmark, 1994).

THEOREM 2 Lyapunov functions for linear systems
Assume that the linear system

$$\frac{dx}{dt} = Ax \tag{2.30}$$

is asymptotically stable. Then for each symmetric positive definite matrix Q there exists a unique symmetric positive definite matrix P such that

$$A^T P + PA = -Q \tag{2.31}$$

Additionally

$$V(x) = x^T P x \tag{2.32}$$

is a Lyapunov function for equation 2.31.

Theorem 2 can therefore be used to obtain an Lyapunov function in the case of linear systems. Again, proof of these theorems can be found in Astrom and Wittenmark (1994).

Chapter 3

System Hardware and Software

Two quadcopter setups are used in this project. This chapter briefly looks at the hardware used on both quadcopters with a focus on the flight controller used since it runs the control system. The chapter then concludes with a discussion of the software used in conjunction with the flight controller.

3.1 Hardware

The control system was implemented on two quadcopters. The reason for this was to check the robustness of the controller and see whether the change in model could be accommodated. One of the objectives of this project is that the designed controller should integrate well with the existing software. Flying an autonomous mission is a good method to see how the newly designed control loops integrate with the unchanged control loops. This requires a GPS, which cannot be added to the smaller quadcopter giving another reason for the use of a second quadcopter. The different components of both quadcopters are summarised in Table 3.1. The most important component for this project is the flight controller. It is therefore discussed in a bit more depth than the other components. Additional information about the hardware can be found in Appendix A.

How the different components are connected is also important. Figure 3.1 shows the typical wiring of quadcopter A. Due to size restrictions Quadcopter B does not include a GPS & Magnetometer, buzzer, telemetry, safety switch and power module. The ESCs are connected directly to the battery and contain 5V regulators that are used to power the Pixhawk flight controller. The main supply voltage is also 12.6 V as opposed to 16.8 V.

Table 3.1: Hardware summary of two quadcopters

No.	Component	Quadcopter A	Quadcopter B
1	Frame	s500	ZMR250
2	Motors	emax 3506 650kv	emax MT1806 2280kv
3	ESCs	Hobbywing Platinum 30A	BLheli 12A
4	Propellers	Quanum Carbon Fiber 1238	Nylon GF 5030
5	Battery	turnigy 5000mah 4s	Power 1300mah 3s
6	GPS	3DR GPS & Magnetometer	None
7	Controller	Pixhawk	Pixhawk
8	Transmitter	Fly Sky FS-TH9X	Turnigy 9x
9	Power Module	3DR Power Module	None
10	Telemetry	3DR radio	None
11	PPM Encoder	JD-PPM Encoder	JD-PPM Encoder
12	Receiver	Fly Sky 2.4Ghz 8ch RX	Fly Sky 2.4Ghz 8ch RX

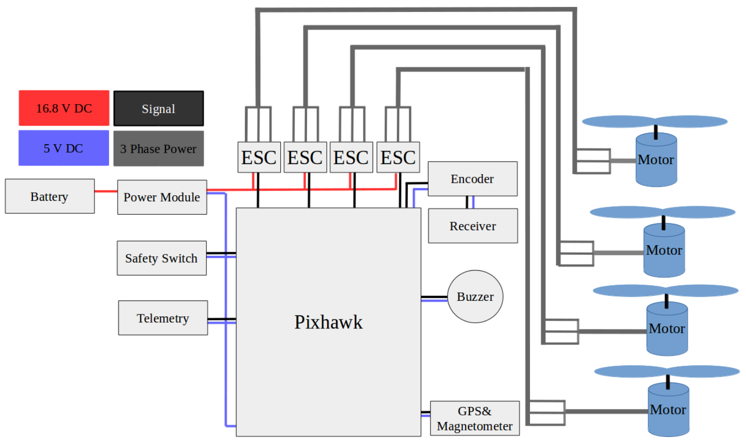


Figure 3.1: General quadcopter wiring

3.1.1 Flight Controller

Various flight controllers are available and they all have their own strengths and weaknesses when it comes to the different flying purposes. It is therefore necessary to choose the best flight controller for the project. Some possible controllers are summarised in Table 3.2.

Figure 3.2: Pixhawk flight controller (PX4 Autopilot, 2016*a*)

Table 3.2: Summary of possible flight controllers (My First Drone, 2016)

Controller Name	Flying Type	Cost (Low to High)
Flyduino KISS	Racing or Freestyle	2
Lumenier LUX	Racing or Freestyle	1
DJI NAZA-M V2	General Use or Photography	4
DJI A3	Pro Aerial Photography	5
3DR Pixhawk	DIY and Autonomous projects	3

From Table 3.2 it can be seen that the 3DR Pixhawk, shown in figure 3.2, is the best suited controller for this project. It is ideal for research projects due to the fact that its firmware is open-source.

3.2 Software

PX4 Pro is an open source autopilot system supporting various model aircrafts (PX4, 2016*b*). Pixhawk is included under the supported boards making it an applicable system for this project. The source code is developed and made available to users by a large community of developers, although the project originated at ETH Zürich (ETH Zürich, 2017). *Qgroundcontrol* is a ground station that supports the PX4 autopilot (Qgroundcontrol, 2016). It can be used to load the firmware onto the Pixhawk, set up the vehicle for flight, monitor values during flight, change parameters and pass commands to the vehicle. Although the firmware can easily be used to load the standard firmware onto the Pixhawk, this project requires custom firmware to be loaded to the board. The PX4 toolchain allows for the PX4 code to be developed (PX4, 2016*a*). An integrated development environment (IDE) simplifies the modification of the source code and provides an option to build the code directly. The IDE used in this project is *QtCreator* (QtCreator, 2016).

3.2.1 Different Flight Modes

Different flight mode options are available in the PX4 firmware. Since the PX4 autopilot system is to be used the different flight modes available in this software will be discussed (PX4, 2017*b*). Considering Figure 3.3 it can be seen that the flight mode being used simply determines where in the control sequence the user inputs enter and the controller takes over. The arrows show where the user inputs enter the cascaded control structure for each of the respective flight modes. [1]

[1]The four arrows in Figure 3.3 are not to be confused with the four categories which group the flight modes.

Manual Flight Modes

Manual control modes refer to modes where the user inputs are passed directly to the attitude controller. Three types of manual flight modes are present: angle mode, acrobatic mode and r-attitude mode. In all manual cases, the throttle is passed directly to the mixer.

Angle mode is the default mode used for flight. In *Qgroundcontrol* it is also referred to as "manual" mode. The transmitter inputs result in angle commands for pitch and roll; and angular rate for yaw. Yaw rate is used instead of yaw angle.

Acro mode, short for acrobatic mode, is for users who want a more dynamic flight where the transmitter inputs result in angular rates for pitch, roll and yaw. Besides allowing users to control the quadcopter angular rates, it also allows for pitch and roll angles exceeding 180°, required for flips and rolls.

R-attitude mode is a combination of the above two modes. The multirotor is in angle mode when the set points fall within the mode's thresholds. When the setpoints exceed these thesholds it switches to acro mode.

Assisted Flight Modes

Assisted flight modes make flight slightly easier for the pilot since more control is done by the control system itself and the user concentration required is reduced. Assisted flight modes include altitude mode and position mode.

In Altitude control mode the attitude is controlled as in manual mode. The throttle stick, however, now controls an increase or decrease in altitude. If no change in throttle input is given, the quadcopter will attempt to keep the same altitude, by means of barometer sensor readings (PX4 Autopilot, 2016*b*).

In the case of position control mode, the transmitter inputs result in left-right and forward-backward speeds. Yaw input is treated the same as in manual mode (angular rate). Throttle is treated the same as in altitude control mode.

Auto Flight Modes

Auto flight modes eliminate the need for a pilot and include: auto hover mode, auto return mode and auto mission mode. With auto hover mode the multirotor holds position and altitude.

Auto return mode will result in the multirotor returning to its starting position in a straight line. The altitude at which the return occurs depends on the altitude of the quadcopter at the instant when return to home is requested. If the current altitude is above a specified loiter altitude, the quadcopter returns at the present altitude. If the current altitude is below the loiter altitude the quadcopter will rise to the loiter height before returning. When at its home position, the quadcopter will automatically land.

Auto mission mode allows a mission to be sent to the aircraft by means of a ground control station. The quadcopter therefore moves to predefined checkpoints in 3D space.

Off-board Flight Modes

This mode allows one to set vehicle set points from another computer or microcontroller and is not included in Figure 3.3. It can be used for robotics, such as for obstacle avoidance where an additional microcontroller commands set points to the Pixhawk flight controller.

The above flight modes are achieved by means of a sequence of three controllers. The difference in the modes are simply which controllers are bypassed.

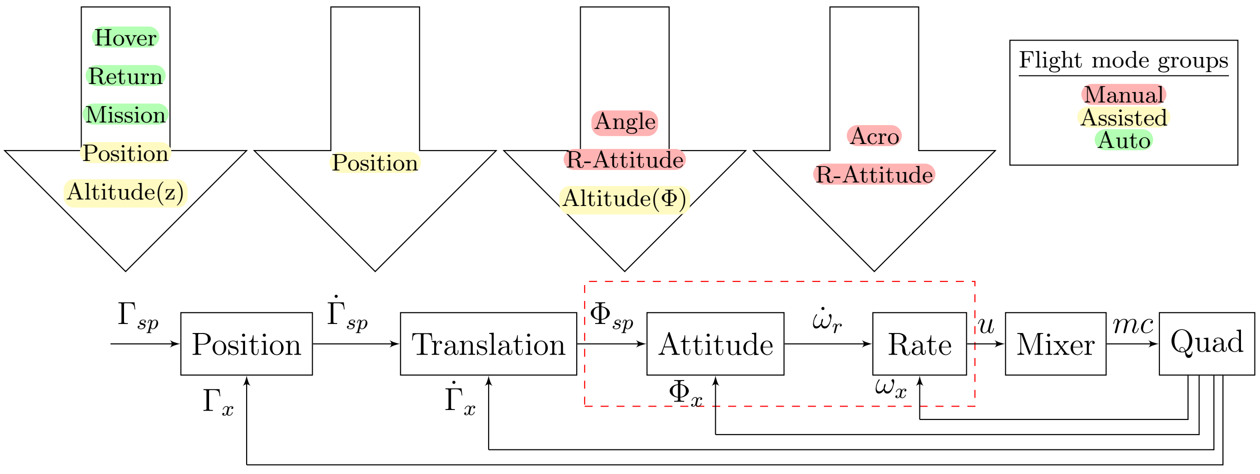


Figure 3.3: Block diagram of system to illustrate entry of different flight modes

From Figure 3.3 it is evident that all flight modes use the rate controller. In order to achieve full attitude control of the quadcopter the two innermost controllers are used. Improving on these two control loops will therefore benefit all the flight modes. This project focuses on improving these two control loops. Any flight mode besides acro mode and r-attitude mode can therefore be used for full evaluation of the newly implemented controllers.

Chapter 4

System modelling

Despite requiring the structure of the mathematical model for the control system design, a detailed mathematical model is also required to simulate the designed controller before implementation. This chapter derives the full mathematical model and also looks at some simplifications that can be considered for simpler control system designs. The chapter concludes with a discussion of the MATLAB simulation used. Additional information on parameter identification can be found in Appendix B.

4.1 Mathematical Model of Quadcopter

A quadcopter has six degrees of freedom (DOF): three are the attitude rotation angles and three are the translation positions. The mathematical model used depends on which degrees of freedom one wants to control. One can also control the rate of change of the degrees of freedom mentioned; see the flight modes discussed in Chapter 2. The model studied in this section is similar to that proposed by Bresciani (2008). Other literature, such as Hartman *et al.* (2014), provides a similar model. The work of Bresciani (2008) is considered here due to a clear derivation. It considers all DOFs and results in a complete model. A different convention is, however, used in order to remain consistent with the convention used in the PX4 firmware. The convention used is as given by figure 2.2 in Chapter 2.

Figure 4.1 summarised the modelling of a quadcopter and is a more detailed version of figure 2.3. It now includes the position model, and the integrator blocks are extracted from the dynamics and kinematics blocks for clarity and to fit better with the derived equations. The rotation of the propellers causes forces and moments. These forces and moments yield accelerations which ultimately result in a change in orientation and position. The blocks in figure 4.1 are discussed from right to left. The model is first evaluated according to the kinematic equations, which relate the body frame to the earth frame, followed by evaluation of the dynamic equations. These equations remain the same irrespective of the quadcopter configuration used (plus or cross), due to the assumption of symmetry. The difference, however, arises when considering the propeller dynamics.

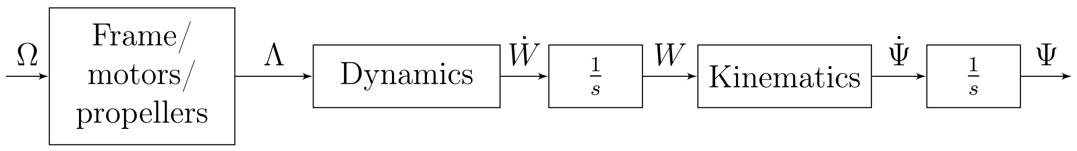


Figure 4.1: Expanded block diagram of dynamics breakdown

4.1.1 Quadcopter Kinematics

As mentioned previously, two reference frames are defined: an earth inertial reference and a body-fixed reference. The earth frame is used to define the linear position and angular orientation of the quadcopter while the body frame is used to define the linear velocity, angular velocity, forces and torques. The vector between the origins of the two frames gives the linear position of the quadcopter and is defined as in equation 4.1.

$$\Gamma^E = [x^E \, y^E \, z^E]^T \tag{4.1}$$

The difference in orientation between the two frames give the angular orientation, Φ, of the quadcopter. The orientation of the frames with respect to one another is defined by three rotations, one around each of the main axes. These three angles yield an attitude vector as shown by equation 4.2.

$$\Phi^E = [\phi \, \theta \, \psi]^T \tag{4.2}$$

A rotation matrix is defined to relate the orientation of the two reference frames to one another by means of the three rotation angles. For more information on how the rotation matrix is obtained see Appendix E.

The linear and angular velocities are given by equation 4.3 and 4.4, respectively.

$$V^B = [u \, v \, w]^T \tag{4.3}$$

$$\omega^B = [p \, q \, r]^T \tag{4.4}$$

Combining equations 4.1 and 4.2 yields equation 4.5, while combining equations 4.3 and 4.4 yields equation 4.6.

$$\Psi = [\Gamma^E \, \Phi^E] = [x^E \, y^E \, z^E \, \phi \, \theta \, \psi]^T \tag{4.5}$$

$$W = [V^B \, \omega^B] = [u \, v \, w \, p \, q \, r]^T \tag{4.6}$$

The linear velocities in the earth frame can be obtained from the linear velocities in the body frame by means of the rotation matrix given in equation E.9.

$$\dot{\Gamma}^E = R_\Phi V^B \tag{4.7}$$

The angular velocities in the body and earth frame can be related in a similar manner by means of a transfer matrix (Chin, 2009). For more information on how the transfer matrix is obtained see Appendix E. Equation 4.8 therefore relates the body and frame angular velocities by means of the transfer matrix.

$$\dot{\Phi}^E = T_\Phi \omega^B \tag{4.8}$$

Combining equations 4.7 and 4.8 yields equation 4.9.

$$\dot{\Psi} = \begin{bmatrix} R_\Phi & 0 \\ 0 & T_\Phi \end{bmatrix} W = J_\Phi W \tag{4.9}$$

4.1.2 Quadcopter Dynamics

For this analysis an assumption is made. It is assumed that the origin of the body frame coincides with the centre of mass of the quadcopter. Newton's second law then yields

$$m\ddot{\Gamma}^E = F^E \tag{4.10}$$

$$m\dot{V}^E = F^E \tag{4.11}$$

Using equation 4.7 to convert to the body frame velocities and forces, we get

$$m\frac{d}{dt}(R_\Phi V^B) = R_\Phi F^B \tag{4.12}$$

Applying the chain rule for derivatives yields

$$m(R_\Phi \dot{V}^B + \dot{R}_\Phi V^B) = R_\Phi F^B \tag{4.13}$$

The derivative of the rotation matrix is given by equation 4.14 (Hughes, 2004).

$$\dot{R}_\Phi V^B = R_\Phi \omega^B \times V^B \tag{4.14}$$

Using equation 4.14, equation 4.13 can be written as 4.15.

$$mR_\Phi(\dot{V}^B + \omega^B \times V^B) = R_\Phi F^B \tag{4.15}$$

Cancelling the rotation matrices on either side of equation 4.15 results in

$$m(\dot{V}^B + \omega^B \times V^B) = F^B \tag{4.16}$$

Similarly, using Newton's second law of rotation yields

$$I_{xyz}\ddot{\Theta}^E = \tau^E \tag{4.17}$$

$$I_{xyz}\dot{\omega}^E = \tau^E \tag{4.18}$$

where I_{xyz} is a 3x3 matrix containing the quadcopter inertias: I_{xx}, I_{yy} and I_{zz}. Using equation 4.8 to convert to the body frame angular velocities and moments yields

$$I_{xyz}\frac{d}{dt}(T_\Phi \omega^B) = T_\Phi \tau^B \tag{4.19}$$

Applying the chain rule for derivatives yields

$$I_{xyz}(T_\Phi \dot{\omega}^B + \dot{T}_\Phi \omega^B) = T_\Phi \tau^B \tag{4.20}$$

Similar to equation 4.14, the derivative of the transfer matrix is given by equation 4.21 (Hughes, 2004).

$$\dot{T}_\Phi (I_{xyz}\omega^B) = T_\Phi \omega^B \times (I_{xyz}\omega^B) \tag{4.21}$$

Using equation 4.21, equation 4.20 can be written as

$$T_\Theta(I_{xyz}\dot{\omega}^B + \omega^B \times (I_{xyz}\omega^B)) = T_\Theta \tau^B \tag{4.22}$$

Cancelling the transfer matrices on either side of equation 4.22 results in

$$I_{xyz}\dot{\omega}^B + \omega^B \times (I_{xyz}\omega^B) = \tau^B \tag{4.23}$$

Combining equation 4.16 and 4.23 yields

$$\begin{bmatrix} mI & 0 \\ 0 & I_{xyz} \end{bmatrix} \begin{bmatrix} \dot{V}^B \\ \dot{\omega}^B \end{bmatrix} + \begin{bmatrix} \omega^B \times (mV^B) \\ \omega^B \times (I_{xyz}\omega^B) \end{bmatrix} = \begin{bmatrix} F^B \\ \tau^B \end{bmatrix} \tag{4.24}$$

where I is an 3x3 identity matrix.

The right-hand side vector from equation 4.24 can be defined by a force-moment vector as given by equation 4.25.

$$\Lambda = \begin{bmatrix} F^B \\ \tau^B \end{bmatrix} = [F_x \; F_y \; F_z \; \tau_x \; \tau_y \; \tau_z]^T \tag{4.25}$$

The system inertia matrix can be defined as

$$M^B = \begin{bmatrix} mI & 0 \\ 0 & I_{xyz} \end{bmatrix} \tag{4.26}$$

The second term in equation 4.24 can be written as the Coriolis-centripetal matrix. Consider the cross product of two vectors. The cross product equivalent can be written using a skew symmetric matrix (Baker, 2016). S(.) is defined as the skew-symmetric operator.

$$\begin{aligned} A \times B &= \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = - \begin{bmatrix} 0 & -b_3 & b_2 \\ b_3 & 0 & -b_1 \\ -b_2 & b_1 & 0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \\ &= S(A)B = -S(B)A \end{aligned} \tag{4.27}$$

The vector containing the two crossproducts can therefore be written as

$$\begin{bmatrix} \omega^B \times (mV^B) \\ \omega^B \times (I_{xyz}\omega^B) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & mw & -mv \\ 0 & 0 & 0 & -mw & 0 & mu \\ 0 & 0 & 0 & mv & -mu & 0 \\ 0 & 0 & 0 & 0 & I_{zz}r & -I_{yy}q \\ 0 & 0 & 0 & -I_{zz}r & 0 & I_{xx}p \\ 0 & 0 & 0 & I_{yy}q & -I_{xx}p & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \end{bmatrix} \tag{4.28}$$

The matrix on the right hand side of equation 4.28 is the Coriolis-centripetal matrix and can be simplified as

$$C^B(W) = \begin{bmatrix} 0 & -mS(V_B) \\ 0 & -S(I_{xyz}\omega^B) \end{bmatrix} \tag{4.29}$$

Substituting equations 4.25, 4.26 and 4.29 into equation 4.24 yields

$$M^B\dot{W} + C^B(W)W = \Lambda \tag{4.30}$$

The force-moment vector can be divided into three components: a gravitational component ($G_B(\Psi)$), a component due to the gyroscopic effects ($O_B(W)\Omega_\Sigma$) and lastly a component due to the forces and torques directly produced by the propellers ($U_B(\Omega_v)$). Appendix E discusses how each of these three components are obtained. Summing the three components and substituting it into equation 4.30 yields

$$M^B\dot{W} + C^B(W)W = G^B(\Psi) + O^B(W)\Omega_\Sigma + U^B(\Omega_v) \tag{4.31}$$

where

$$\Omega_\Sigma = \Omega_1 + \Omega_2 - \Omega_3 - \Omega_4 \tag{4.32}$$

$$\Omega_v = [\Omega_1\ \Omega_2\ \Omega_3\ \Omega_4]^T \tag{4.33}$$

Equation 4.31 can be simplified and broken up into each degree of freedom in terms of the body axis.

$$\dot{u} = (vr - wq) + g\sin\theta \tag{4.34}$$

$$\dot{v} = (wp - ur) - g\cos\theta\sin\phi \tag{4.35}$$

$$\dot{w} = (uq - vp) - g\cos\theta\sin\phi + \frac{U_1}{m} \tag{4.36}$$

$$\dot{p} = \frac{I_{yy} - I_{zz}}{I_{xx}}qr - \frac{J_T}{I_{xx}}q\Omega_\Sigma + \frac{U_2}{I_{xx}} \tag{4.37}$$

$$\dot{q} = \frac{I_{zz} - I_{xx}}{I_{yy}}pr + \frac{J_T}{I_{yy}}p\Omega_\Sigma + \frac{U_3}{I_{yy}} \tag{4.38}$$

$$\dot{r} = \frac{I_{xx} - I_{yy}}{I_{zz}}pq + \frac{U_4}{I_{zz}} \tag{4.39}$$

As mentioned in Appendix E, the configuration used determines $U_B(\Omega_v)$. Equations E.21 and E.22 are grouped as in equation 4.40.

$$\begin{bmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b(\Omega_1^2+\Omega_2^2+\Omega_3^2+\Omega_4^2) \\ lb(\Omega_2^2-\Omega_1^2) \\ lb(\Omega_3^2-\Omega_4^2) \\ d(-\Omega_1^2-\Omega_2^2+\Omega_3^2+\Omega_4^2) \end{bmatrix}_+ \text{ or } \begin{bmatrix} 0 \\ 0 \\ b(\Omega_1^2+\Omega_2^2+\Omega_3^2+\Omega_4^2) \\ lb\sin 45^\circ(\Omega_3^2+\Omega_2^2-\Omega_1^2-\Omega_4^2) \\ lb\sin 45^\circ(\Omega_1^2+\Omega_3^2-\Omega_4^2-\Omega_2^2) \\ d(-\Omega_1^2+\Omega_2^2-\Omega_3^2+\Omega_4^2) \end{bmatrix}_\times \tag{4.40}$$

The values of b and d are constants dependent on the motor-propeller combination being used (Greitzer *et al.*, 2007) and l is the distance from the motor axes to the centre of mass.

The above equations are written with respect to the body frame. Although it is ideal to have the rotation angles with respect to the body frame, it is ideal to have the translation positions with respect to the earth frame. A hybrid H-frame is therefore defined. The new velocity vector with respect to the H-frame is then given by

$$\zeta = [\dot{\Gamma}^E \; \omega^B] = [\dot{x}\; \dot{y}\; \dot{z}\; p\; q\; r] \tag{4.41}$$

Equation 4.31 then becomes

$$M^B\dot{\zeta} + C^H(\zeta)\Psi = G^H + O^H(\zeta)\Omega_\Sigma + U^H(\Omega_v) \tag{4.42}$$

The system inertia matrix and the gyroscopic propeller matrix remains the same. The new Coriolis-centripetal matrix, gravitational vector and moment matrix are given in Appendix E by equation E.24, E.25 and E.26, respectively. Breaking equation 4.42 up into each degree of freedom yields equations 4.43-4.48.

$$\ddot{x} = (\sin\psi\sin\theta + \cos\psi\sin\theta\cos\phi)\frac{U_1}{m} \tag{4.43}$$

$$\ddot{y} = (\cos\psi\sin\theta + \sin\psi\sin\theta cos\phi)\frac{U_1}{m} \tag{4.44}$$

$$\ddot{z} = -g + (\cos\theta\cos\phi)\frac{U_1}{m} \tag{4.45}$$

$$\dot{p} = \frac{I_{yy}-I_{zz}}{I_{xx}}qr - \frac{J_TP}{I_{xx}}q\Omega_\Sigma + \frac{U_2}{I_{xx}} \tag{4.46}$$

$$\dot{q} = \frac{I_{zz}-I_{xx}}{I_{yy}}pr + \frac{J_TP}{I_{yy}}q\Omega_\Sigma + \frac{U_3}{I_{yy}} \tag{4.47}$$

$$\dot{r} = \frac{I_{xx}-I_{yy}}{I_{zz}}pq + \frac{U_4}{I_{zz}} \tag{4.48}$$

The U_i and Ω_Σ values remain the same as before. It is noteworthy that the controller does not output a RPM command, but a PWM command. This relationship is discussed in more depth in Appendix B.

4.2 Simplified Quadcopter Model

Neglecting the effect of gyroscopic forces in equations 4.46, 4.47 and 4.48 removes the cross-coupling of degrees of freedom. Another assumption to be made is that the T_Θ matrix approximates the identity matrix; this is true if the quadcopter is near its hover position. This is essentially similar to the small angle approximation. By making the small angle approximation in equation 4.43, it also becomes decoupled. The simplified equations are given by equations 4.49, 4.50, 4.51 and 4.52.

$$\ddot{Z} = -g + \frac{U_1}{m} \tag{4.49}$$

$$\ddot{\phi} = \frac{U_2}{I_{xx}} \tag{4.50}$$

$$\ddot{\theta} = \frac{U_3}{I_{yy}} \tag{4.51}$$

$$\ddot{\psi} = \frac{U_4}{I_{zz}} \tag{4.52}$$

4.3 MATLAB Simulation

The simulation in MATLAB is achieved by means of a Simulink project, "quad-sim" created by Drexel University (Hartman *et al.*, 2014). The project is set up with a simple PID controller. Modifications in the control block were therefore required to simulate the response of the different controllers. Other minor adjustments were required due to the convention differences between the PX4 firmware and the "quad-sim" program. These changes are discussed in Appendix B.

Quad-sim does not only provide a model to simulate the response of the system with a PID controller, but it also provides information on how to obtain the quadcopter parameters. Once these parameters are obtained the model can be stored using a simple graphical user interface (GUI). The initial conditions can also be set from a GUI. The software supports both plus and cross configurations. Documents on the derivation of the mathematical model is also given. The set points can easily be changed by means of step input blocks. Both attitude and position control is available. A predefined path can also be specified. Only the attitude modelling will, however, be utilised in this project.

After the simulation is run, it gives plots of all the states as well as the motor commands and speed. A 3D animation can also be viewed. The different blocks can be seen in Figure 4.2.

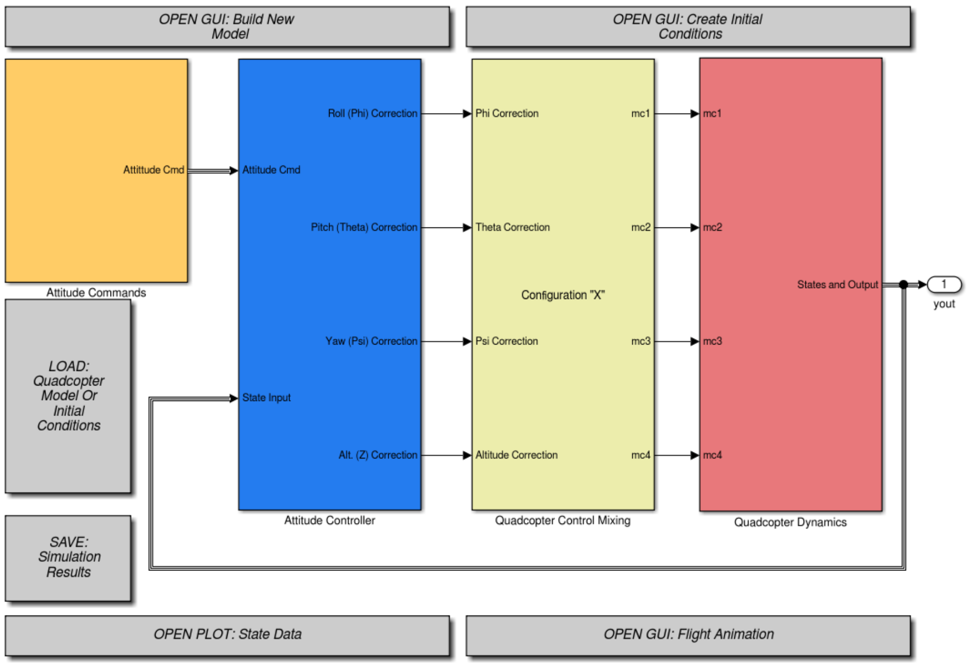


Figure 4.2: Quadsim simulink blocks (Hartman *et al.*, 2014)

The grey blocks perform additional features while the sequence of colourful blocks perform the simulation. The desired attitude angles together with the measured attitude angles and angular rates are sent to the controller block. The controller block calculates a correction factor which is then sent to the control mixing block which takes these correction commands and calculates the motor outputs to achieve the desired moments. The motor commands are then sent to the quadcopter dynamics block which determines the quadcopters' response according to the motor outputs. Each of these blocks, along with the modifications required, are discussed in more detail in Appendix B.

Chapter 5

Control

Chapter 2 showed how a cascade controller can be used on a quadcopter. Despite improved control, the use of a cascade controller will also allow integration with the flight modes currently available in the PX4 software. The approach followed is therefore similar to that of Achtelik *et al.* (2011). Other approaches such as neural networks were neglected due to concerns of the processing power of the Pixhawk flight controller. The chapter starts with an overview of the proposed controller followed by a discussion of each of the control loops. The NDI loop is discussed first together with the additions of some terms. After the concept is covered in Euler angles, the concept is extended to quaternions. The rate loop is then designed, first for the non-adaptive case followed by the adaptive case. The final design will be based on Lyapunov theory. The adaptive law is, however, first evaluated according to the MIT rule. This is done to illustrate the difference between the two approaches. Some modifications were then applied. The designed controllers are then simulated before moving on to the practical results.

5.1 Controller Overview

The cascaded method uses two control loops to control the attitude of the quadcopter. Figure 5.1 again shows the cascaded control structure. The first loop takes the commanded Euler angles in the earth frame, and uses a reference model to generate desired attitude angle rates in the body frame. The rate control loop then uses the desired angular rates together with the measured angular rates to generate the moment commands which are sent to the mixer. The mixer determines the speed command sent to the motors. Both control loops are discussed separately in more detail below.

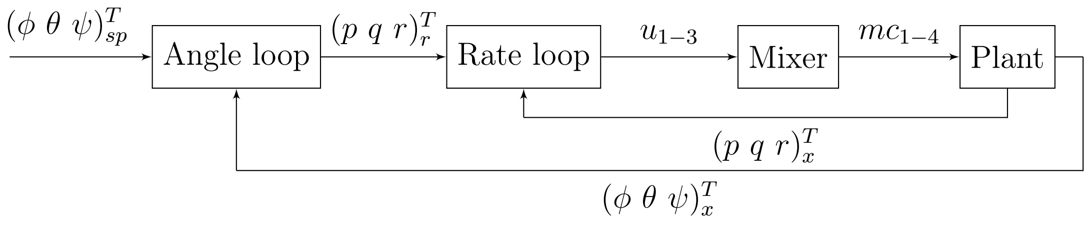


Figure 5.1: Closed loop block diagram of proposed cascaded controller

5.2 Angle Reference Loop Design

Non-linear dynamic inversion was also used in (Achtelik *et al.*, 2011). This project, however, adds an error term for the hedging. The Euler angle approach is also converted to a quaternion approach. The concept is first explained in Euler angles for clarity.

A reference model is used to generate desired angular rates (in the earth frame) which will result in a gradual increase of the reference angles to the desired attitude set point values. The reference model is given by equation 5.1.

$$\dot{\Phi}_r(t) = \frac{1}{T}(\Phi_{sp} - \Phi_r) \tag{5.1}$$

T determines the rate at which Φ_r increases. Smaller T values will yield a faster response while a larger T value will yield a slower response. Figure 5.2 (a) shows how the reference increases for a step input command of unity around the roll axis. Figure 5.2 (b) shows the reference rate required to achieve the desired change in the reference angle.

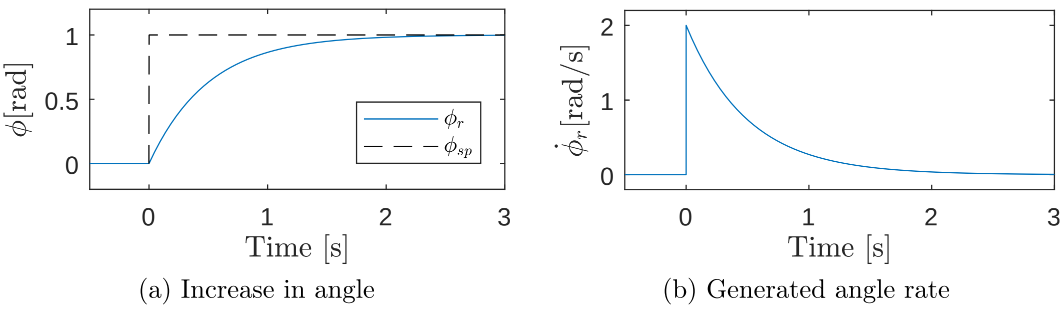


Figure 5.2: Attitude reference model response

In practice the actual attitude angles will not follow the reference model's attitude angles perfectly. An error term is therefore added. The error is given by the difference in the reference angle and the measured earth angle at that point as shown by equation 5.2.

$$e_\Phi = \Phi_r - \Phi_x \tag{5.2}$$

An integral term of this error is also added in order to avoid steady state error. The final pseudo control is then given by equation 5.3.

$$V_r = \dot{\Phi}_r + K_p e_\Phi + K_i \int e_\Phi dt \tag{5.3}$$

The pseudo control is angular rates in the earth frame. The angular rates used in the controller are, however, in the body frame. The pseudo controls are therefore multiplied by the rotation matrix as in equation 5.4.

$$\omega_r = \begin{bmatrix} p_r \\ q_r \\ r_r \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix} \begin{bmatrix} \dot{\phi}_{pc} \\ \dot{\theta}_{pc} \\ \dot{\psi}_{pc} \end{bmatrix} = T_\Phi^{-1} V_r \tag{5.4}$$

The block diagram of the adaptive reference loop is given in Figure 5.3.

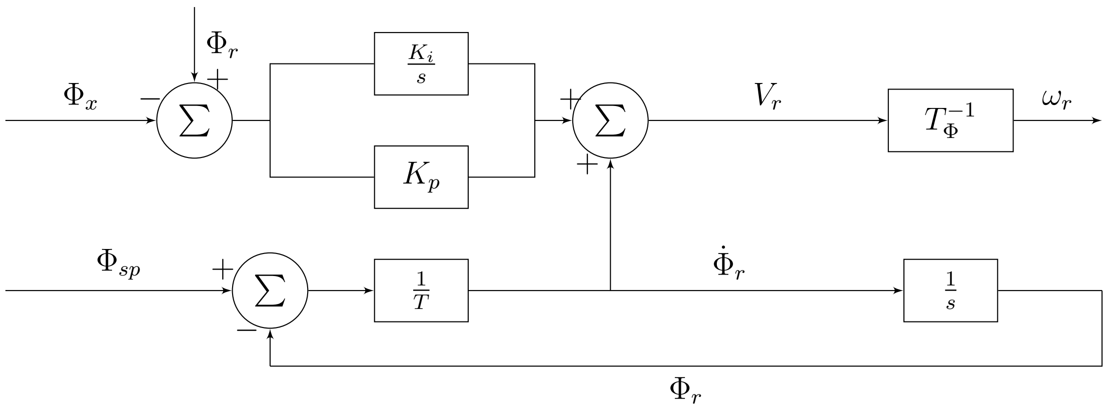


Figure 5.3: Inner workings of attitude control block

5.2.1 Modifications

Some modifications can be done in order to improve the workings of the angle control loop. The modification discussed is that of pseudo control hedging and the implementation of quaternions as opposed to Euler angles.

Pseudo Control Hedging

The outer loop requests a rate from the inner loop. The inner loop then attempts to achieve this rate. The inner loop can therefore be seen as an actuator that is being driven by the outer loop. As is the case with actuators such as motors, the motor does not instantaneously achieve its desired output. For instance the velocity achieved from the motor is not instantaneous, the motor accelerates to the desired velocity. Similarly the velocity requested from the inner loop is not instantaneously possible. The difference between the measured rate and the desired rate results in a hedging term.

$$V_h = V_r - T_\Phi \omega_x \tag{5.5}$$

This hedging term is used to slow down the reference model. Equation 5.6 shows how the hedging term is integrated into the existing reference model, equation 5.1.

$$\dot{\Phi}_r(t) = \frac{1}{T}(\Phi_{sp} - \Phi_r) - V_h \tag{5.6}$$

The updated pseudo control equation then becomes

$$V_r = \dot{\Phi}_r + V_h + K_p e_\Phi + K_i \int e_\Phi dt \tag{5.7}$$

An additional benefit of pseudo control hedging is that it accounts for motor saturation. If the motors are saturated the desired angular rate will not be achievable. The hedging accounts for this saturation by slowing down the reference model. The diagram in Figure 5.4, updated from Figure 5.3, shows where the pseudo control hedging term enters the controller.

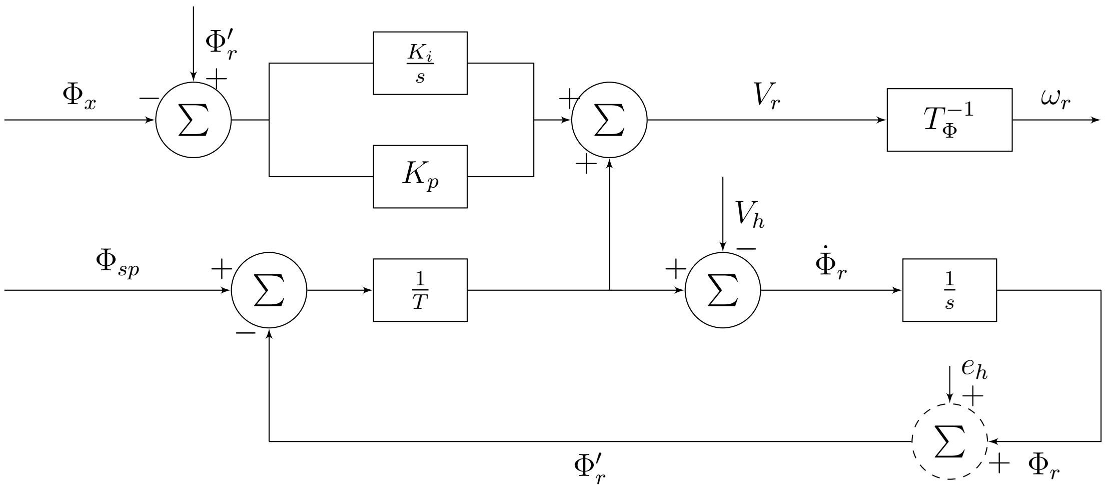


Figure 5.4: Inner workings of attitude control block with addition of hedging

The addition of the hedging term, however, introduces a steady state error, especially when adaption is switched off. When a disturbance is introduced, or simply when the quadcopter's centre of mass is slightly off-centre, a moment command is required to counter this. When adaption is switched off the only way to achieve a non-zero moment at steady state is if a non-zero angular rate is commanded ($V_r \neq 0$). Steady state of the reference model is achieved when $\dot{\Phi}_r$ is zero. Without the hedging term this occurs when Φ_r is equal to Φ_{sp}.

Adding the hedging term changes this, see equation 5.8 which is equation 5.6 with $\dot{\Phi}_r(t) = 0$.

$$V_h = \frac{1}{T}(\Phi_{sp} - \Phi_r) \tag{5.8}$$

Equation 5.8 is required to achieve steady state of the reference model. From this it is evident that to achieve perfect tracking when the reference model is at steady state, the hedging term needs to be zero. From equation 5.5 it can be seen that at steady state $V_h = V_r$. V_h can therefore not be zero with adaption switched off. This means that a steady state error between Φ_{sp} and Φ_r is always present. When adaption is switched on the idea is that the disturbance is countered by the adaptive loop's disturbance term. V_r can then be allowed to go to zero, in turn allowing V_h to go to zero and allowing Φ_{sp} and Φ_r to be equal. In theory the addition of hedging does not pose steady state problems when adaption is switched on. In practice the disturbance term does not perfectly eliminate the applied disturbances, meaning that V_r is not exactly zero. Hedging therefore still poses a steady state problem in practice.

When $\Phi_{sp} \neq \Phi_r$ the proportional and integral error term, $e_\Phi = \Phi_r - \Phi_x$, will not correct the steady state error since these error terms simply drive the measured angle to that of the reference model. Φ_r must therefore be corrected. An error term $e_h = \Phi_{sp} - \Phi_x$ is added to the reference signal fed back.

$$\Phi_r' = \Phi_r + K_h e_h \tag{5.9}$$

This helps to limit the drift of Φ_r from Φ_{sp}. This term is introduced to improve tracking at steady state. It should therefore only affect the system once it is approximately at steady state. The term is therefore only introduced once the error, e_Φ, is less than 0.08 radians. With adaption off the error at steady state is usually larger when a large disturbance is introduced. This limit, however, proved satisfactory when adaption was switched on since the adaptive disturbance term accounted for the majority of the disturbance. Figure 5.5 shows how a small disturbance affects the reference model with this error term added and without. This test was done with adaption off since with adaption on the ideal case occurs and the reference model does not drift due to the Θ_d term discussed in Section 5.4.

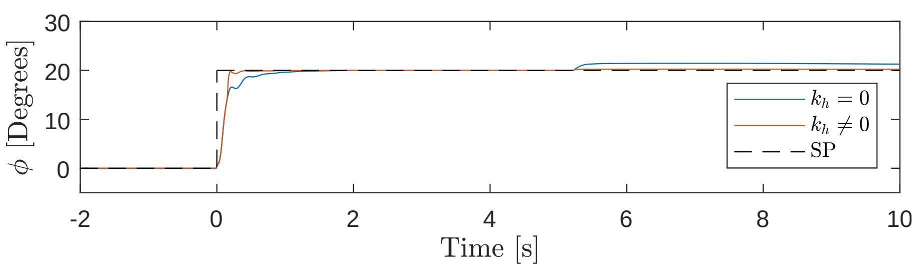


Figure 5.5: Effect of hedging error term, e_h

Addition of Quaternions

Working with Euler angles in the angle loop does pose a problem. A discontinuity occurs at the transition from $-\pi$ to π, which is problamatic for yaw rotation. It is uncertain how Achtelik *et al.* (2011) handled this discontinuity. For the Euler angle case it was therefore decided to simply control the yaw rate, which is passed directly as a command. The yaw rate sent to the adaptive loop is therefore directly commanded and not calculated using equation 5.6, as is the case for pitch and roll. This does not pose a problem for manual mode, since some drift in the yaw angle is not catastrophic in manual mode. [1] However, in order to allow the controller to be successfully integrated with the other control modes in the PX4 firmware, yaw angle control is required. This is the instance with assisted and auto control modes where the yaw orientation of the quadcopter is important. The use of quaternions removes the discontinuity issue. Using quaternions in the angle loop is, however, not as simple as it is for the original PX4 firmware case, since a reference model (equation 5.6) is used. The reference model is used to create a reference vector, z-axis, which moves towards the set point vector, z-axis. Angular pitch and roll rates are calculated in order to follow this reference vector, neglecting yaw rotation. The error between the reference z-axis and measured z-axis is used to eliminate errors since the actual vector will not perfectly follow the reference vector. It is also used to correct the yaw error by aligning the x-axes. The quaternion approach is discussed in more detail below.

For the quaternion approach, in manual mode, yaw rates are still commanded by the pilot. The yaw angle set point is, however, updated by integrating this command. The yaw angle is therefore controlled to prevent drift when no yaw rate is commanded and also to eliminate yaw error introduced during the first z-axis alignment. During missions yaw angles are commanded and not yaw rates. Fig. 5.6 shows how the above approach can be implemented using quaternions. Four rotation matrices are created. One from the measured angles, one from the calculated reference angles, one from the calculated reference rates and one for the steady state rates of zero.

The first two terms of equation 5.7 is achieved by aligning the two rate rotation matrices. Rotation matrices actually represent angle orientation and not angle rates. It was, however, found that creating a rotation matrix, with the reference rates as angles, to be aligned with a rotation matrix, with zero rates as angles, yielded good results. This step can, however, be neglected and the generated reference rates (equation 5.6), neglecting the hedging term, can be converted to body frame rates using T_{Φ}^{-1}. Due to no yaw rate, the main advantage of quaternions is also lost in this case.

[1]For more information on the different flight modes available see Chapter 2.

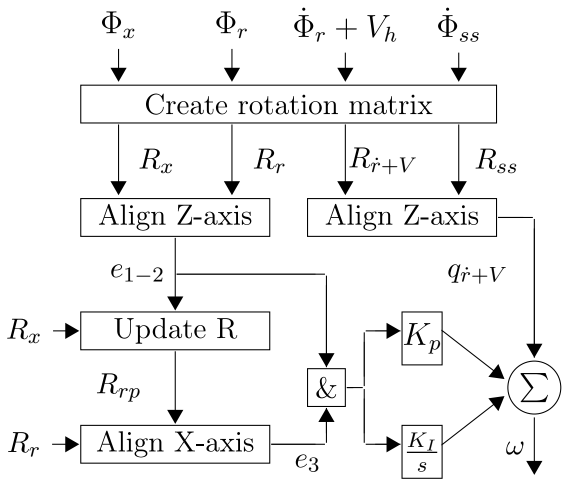


Figure 5.6: Angle loop quaternion breakdown

For the last two terms in equation 5.7 the error between the reference model and measured angles is required. The two rotation matrices considered are that of the measured orientation and that of the expected orientation at that instant in time due to the reference rates. Aligning the z-axes of these two coordinate frames result in a pitch and roll error. The corrected measured matrix, however, still has a yaw error. The x-axes are then aligned to obtain this yaw error. This error vector is then used to calculated the integral and proportional terms in equation 5.7. There is no need to convert the result to the body frame since quaternions are used. The main blocks of Figure 5.6 is discussed next. The alignment of the matrices follows a similar approach to that in the PX4 firmware Ardupilot (2016) [2].

The first block creates a rotation matrix. For more information on how a rotation matrix is obtained refer to Appendix E. The second important block aligns the z-axes. The z-vector of the desired and actual rotation matrix is extracted for both matrices as in equations 5.10-5.11.

$$R_{z,d} = [R_d^{(1,3)}, R_d^{(2,3)}, R_d^{(3,3)}] \tag{5.10}$$

$$R_{z,a} = [R_a^{(1,3)}, R_a^{(2,3)}, R_a^{(3,3)}] \tag{5.11}$$

Next, determine a vector which will define the axis around which the vector must rotate, as well as the angle of rotation. This vector is given as in equation 5.12 and represents the error between the two z-axes.

$$e_R = [i_x \sin\alpha, i_y \sin\alpha, i_z \sin\alpha]^T \tag{5.12}$$

where i is the unit vector for the axis of rotation and α is the angle of rotation around this axis. Since the rotation axis lies within the x-y plane, i_z will be

[2]Another rotation method is used if the z-axis rotation required is too large. This case is not discussed here.

zero. The e_R vector as in equation 5.12 is determined from the R_z vectors as in equation 5.13.

$$e_R = R_d^T \mu_z \tag{5.13}$$

with

$$\mu_z = \begin{bmatrix} R_{z,a}^{(2)} R_{z,d}^{(3)} - R_{z,a}^{(3)} R_{z,d}^{(2)} \\ R_{z,a}^{(3)} R_{z,d}^{(1)} - R_{z,a}^{(1)} R_{z,d}^{(3)} \\ R_{z,a}^{(1)} R_{z,d}^{(2)} - R_{z,a}^{(2)} R_{z,d}^{(1)} \end{bmatrix} \tag{5.14}$$

Now that the error between the two z-axes is known, the new rotation matrix after applying the e_R rotation is required. First, determine the cosine and sine of the angle error as in equations 5.15 and 5.16 respectively. Equation 5.16 is determined by taking the dot product of $R_{z,a}$ and $R_{z,d}$.

$$e_{R,sin} = \|e_R\|_2 \tag{5.15}$$

$$e_{R,cos} = R_{z,a} \circ R_{z,d} \tag{5.16}$$

Next a yaw weight is determined as

$$\psi_w = R_d^{(3,3)} R_d^{(3,3)} \tag{5.17}$$

The axis angle representation is given by

$$e_{R,axis} = \frac{e_R}{e_{R,sin}} \tag{5.18}$$

The rotation matrix for roll/pitch only rotation is given by

$$R_{rp} = R_a(I + [e_{R,axis}]_x e_{R,sin} + [e_{R,axis}]_x [e_{R,axis}]_x (1 - e_{R,cos})) \tag{5.19}$$

where $[e_{R,axis}]_x$ is the cross product matrix defined as

$$[e_{R,axis}]_x = \begin{bmatrix} 0 & -e_{R,axis}^{(3)} & -e_{R,axis}^{(2)} \\ e_{R,axis}^{(3)} & 0 & -e_{R,axis}^{(1)} \\ -e_{R,axis}^{(2)} & e_{R,axis}^{(1)} & 0 \end{bmatrix} \tag{5.20}$$

The last step is to align the x-axis (correct the yaw error). Similar to the z-axis case, extract the x-vector for R_{rp} and R_d as in equations 5.21-5.22.

$$R_{x,rp} = [R_{rp}^{(1,1)}, R_{rp}^{(2,1)}, R_{rp}^{(3,1)}] \tag{5.21}$$

$$R_{x,d} = [R_d^{(1,1)}, R_d^{(2,1)}, R_d^{(3,1)}] \tag{5.22}$$

The yaw error can then be determined as

$$e_R^{(3)} = \tan^{-1}(\mu_x \circ R_{z,d}, R_{x,rp} \circ R_{x,rp})\psi_w \tag{5.23}$$

with μ_x being similar to μ_z, equation 5.14, except for using $R_{x,rp}$ instead of $R_{x,a}$.

Figure 5.7 shows the improvement in tracking when using quaternions as opposed to Euler angles when the system is excited around all three axes simultaneously. The set points around the pitch and roll axes are square waves. Initially the tracking in all three axes is relatively similar for the two approaches. The quaternion approach fairs slightly better, but the pitch and roll tracking error still remain within 2° for the Euler approach. The advantage of the quaternion approach arises when a yaw rate is commanded at 12 seconds. In the case of the Euler approach, the measured pitch and roll angle deviate significantly from their set points from this point. A slight offset in the yaw angle is also present for the Euler case.

5.3 Non-adaptive Rate Loop Design

The three reference rates together with the measured rates (in the body frame) are used to generate the moment commands around the three axes. The ω_r vector is multiplied by a 3x3 matrix with positive values on the diagonal and zero values otherwise. The x_p vector is multiplied by a 3x3 matrix with negative values on the diagonal and zero values otherwise. The results are added to give the moment command vector. This block essentially forms a linear quadratic regulator (LQR) controller.

$$u = \Theta_x x_p + \Theta_r r \tag{5.24}$$

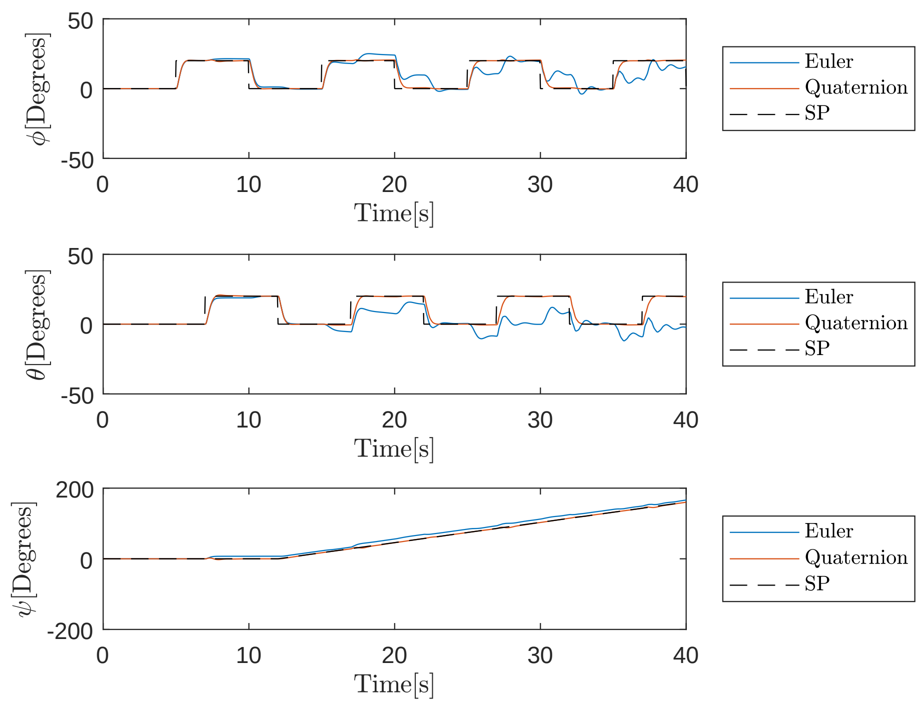


Figure 5.7: Advantage of using quaternions as opposed to Euler angles

5.4 Adaptive Rate Loop Design

Achtelik *et al.* (2011) made use of Lyapunov theory in the design of the adaptive law, the MIT rule is, however, also considered here to illustrate the difference between the two methods. Achtelik *et al.* (2011) allows adaption of values for each motor, where this project only allows for adaption for each axis of rotation. The former therefore includes the mixer with the controller, where the latter separates the two. This is in order to better integrate with the existing PX4 firmware. The control loop to which adaptive control will be added will be the rate control loop. The angle reference loop will remain unchanged since no parameter uncertainties are present, because the relation between the attitude angles and body angular rates are only trigonometric. The rate control loop will, however, become the adaptive loop since the relation between the moment commands and angular rates includes uncertain parameters such as the inertia around the different axes and the motor/propeller parameters.

The equation relating the angular body rates and the moment commands, equations 4.46-4.48, have the form of equation 5.25.

$$\dot{x_p} = A_p x_p + B_p U + \alpha f(x_p) + d \tag{5.25}$$

where A_p is zero in this case. B_p is the inverse inertia matrix, equation 5.26. $f(x_p)$ represents the non-linearities and is given in equation 5.27. α is given in equation 5.28. x_p refers to the plant angular rates in the body frame.

$$B_p = \begin{bmatrix} I_{xx} & 0 & 0 \\ 0 & I_{yy} & 0 \\ 0 & 0 & I_{zz} \end{bmatrix}^{-1} \tag{5.26}$$

$$f(x_p) = [rq\ rp\ pq]^T \tag{5.27}$$

$$\alpha = \begin{bmatrix} \frac{-1}{I_{xx}(I_{zz}-I_{yy})} & 0 & 0 \\ 0 & \frac{-1}{I_{yy}(I_{xx}-I_{zz})} & 0 \\ 0 & 0 & \frac{-1}{I_{zz}(I_{yy}-I_{xx})} \end{bmatrix} \tag{5.28}$$

The d-term represents any disturbances that might enter the system and also accounts for additional uncertainties such as the gyroscopic forces.

To add the adaptive component to the rate control loop two more terms are added to the initial control law. Vector $f(x_p)$ is multiplied by an adaptive matrix, which attempts to remove the non-linearities. A unity vector, representing disturbances, is multiplied by another adaptive matrix which attempts to remove disturbances. The original matrices, being multiplied with the reference vector and measured vector, are also allowed to adapt. These four matrices therefore change with time in order to obtained a desired response. The adaptive control law is given by equation 5.29.

$$u = \Theta_x x_p + \Theta_r r + \Theta_\alpha f + \Theta_d i_{vec} \tag{5.29}$$

The moment commands and the actual moments experienced by the quadcopter are not exactly the same. This is to be expected: for instance the exact same throttle command will not yield the same thrust for two different types of motors. The relationship is approximated by equation 5.30. The relationship is in fact quadratic, but is assumed linear as an approximation. This is due to the relationship between the force command and the motor command, which is linear to RPM, in the PX4 firmware being linear, whereas the RPM to force relationship is in fact quadratic. This yields an additional uncertainty to be adapted for. The uncertainty can be removed by taking the square root of the motor commands, although this goes against the objective of not editing the mixer. From simulation and practice this was, however, seen as not necessary, meaning that the assumption was valid.

$$U = \xi u \tag{5.30}$$

The desired response is given by another reference model, called the adaptive reference model. This reference model is given by equation 5.31. This reference model is used to adjust the adaptive matrices in equation 5.29.

$$\begin{bmatrix} \dot{p}_d \\ \dot{q}_d \\ \dot{r}_d \end{bmatrix} = \begin{bmatrix} -1/T_d & 0 & \\ 0 & -1/T_d & 0 \\ 0 & 0 & -1/T_d \end{bmatrix} \begin{bmatrix} p_d \\ q_d \\ r_d \end{bmatrix} + \begin{bmatrix} 1/T_d & 0 & \\ 0 & 1/T_d & 0 \\ 0 & 0 & 1/T_d \end{bmatrix} \begin{bmatrix} p_{sp} \\ q_{sp} \\ r_{sp} \end{bmatrix} \tag{5.31}$$

$$\dot{x}_d = A_d x_d + B_d r \tag{5.32}$$

The next step is to determine how the four adaptive matrices will adapt. Two methods are studied in this report, one which uses the MIT rule and another which uses Lyopunov theory. They follow a similar route at first, but at some point they start to deviate from one another. From this point the two methods are discussed separately.

Substituting the control law, equation 5.29, into equation 5.30 and substituting the result into 5.25 yields

$$\dot{x_p} = (A_p + B_p \xi \Theta_x) x_p + B_p \xi \Theta_r r + (B_p \xi \Theta_\alpha + \alpha) f(x_p) + B_p \xi \Theta_d i + d \tag{5.33}$$

We want the values of x_p and $\dot{x}_p$ to match those of the desired reference model. This will occur in the case of ideal adaption matrices, Θ_x^*, $\Theta_r^*, \Theta_\alpha^*$ and Θ_d^*. Matching the coefficients of equation 5.32 and 5.33 yield equations 5.34- 5.37.

$$A_d = A_p + B_p \xi \Theta_x^* \tag{5.34}$$

$$B_d = B_p \xi \Theta_r^* \tag{5.35}$$

$$B_p \xi \Theta_\alpha^* + \alpha = 0 \tag{5.36}$$

$$B_p \xi \Theta_d^* + D = 0 \tag{5.37}$$

which implies

$$\Theta_x^* = \frac{A_m - A_p}{B_p \xi} \tag{5.38}$$

$$\Theta_r^* = \frac{B_m}{B_p \xi} \tag{5.39}$$

$$\Theta_\alpha^* = \frac{-\alpha}{B_p \xi} \tag{5.40}$$

$$\Theta_d^* = \frac{-d}{B_p \xi} \tag{5.41}$$

The output of the reference model is compared to the actual measured values. The error between these values drive the adaption of the controller.

$$e_a = x_p - x_d \tag{5.42}$$

A state predictor can also be used instead of using a reference model. It is included since it is important when considering L1 adaptive control which is another adaptive method. Studying the difference between using a reference model and a state predictor can therefore be beneficial. Consider again equation 5.25 and rewrite it as in equation 5.43.

$$\frac{d\hat{x}}{dt} = A_d \hat{x} + (\hat{A}_p - A_d)\hat{x} + \hat{B}_p \xi u + \alpha \hat{f} + \hat{di} \tag{5.43}$$

where the hats represent uncertainties that are predicted. Rearranging equation 5.43 gives

$$\frac{d\hat{x}}{dt} = A_d \hat{x} + \hat{B}_p \xi (u + \frac{(\hat{A}_p - A_d)\hat{x}}{\hat{B}_p \xi} + \frac{\alpha \hat{f}}{\hat{B}_p \xi} + \frac{\hat{di}}{\hat{B}_p}) \tag{5.44}$$

The uncertainties can be estimated by the adaptive parameters, Θ_x, Θ_α and Θ_d.

$$\frac{d\hat{x}}{dt} = A_d \hat{x} + \hat{B}_p \xi (u - \Theta_x \hat{x} - \Theta_\alpha - \Theta_d) \tag{5.45}$$

Choosing the same control law as previously will allow the terms to cancel, yielding a similar equation to that of the reference model.

$$\frac{d\hat{x}}{dt} = A_d \hat{x} + \hat{B}_p \xi \Theta_r r \tag{5.46}$$

If B_p is viewed as a matrix with no uncertainties then the reference gain matrix, Θ_r, does not need to adapt, as the ideal parameter then contains no uncertainties, $\Theta_r^* = \Theta_r = \frac{B_d}{B_p \xi}$. Equation 5.46 then yields the exact same equation as that of the reference model, equation 5.32. The B_p matrix is, however, viewed as an uncertainty leading to a slight difference between the two approaches. This is due to the fact that the reference model only relies on changes in references whereas the state predictor relies on changes in references and changes in the reference adaptive parameter, Θ_r. The method using the reference model is considered further.

The MIT rule and Lyopunov stability theory now start to deviate from one another. The MIT rule is considered further first, followed by the Lyopunov stability theory from this same point.

5.4.1 MIT Controller Design

This section builds on the theory discussed in Chapter 2. Using $p = \frac{d}{dt}$ equation 5.33 can be written as equation 5.47, considering each axis separately.

$$x_p^j = \frac{B_p^{jj}\xi^{jj}\Theta_r^{jj}r^j + (B_p^{jj}\xi^{jj}\Theta_\alpha^{jj} + \alpha^{jj})f^j(x_p) + B_p^{jj}\xi^{jj}\Theta_x^{jj}\imath^{jj} + d^j}{p - A_p^{jj} - B_p^{jj}\xi^{jj}\Theta_x^{jj}} \tag{5.47}$$

Substuting equation 5.47 into equation 5.42 and taking the partial derivatives with respect to Θ yields equations 5.48 - 5.51.

$$\frac{\delta e_a^j}{\delta\Theta_r^{jj}} = \frac{B_p^{jj}\xi^{jj}r^j}{p - A_p^{jj} - B_p^{jj}\xi^{jj}\Theta_x^{jj}} \tag{5.48}$$

$$\frac{\delta e_a^j}{\delta\Theta_x^{jj}} = \frac{B_p^{jj}\xi^{jj}[B_p^{jj}\xi^{jj}\Theta_r^{jj}r^j + (B_p^{jj}\xi^{jj}\Theta_\alpha^{jj} + \alpha^{jj})f^j(x) + B_p^{jj}\xi^{jj}\Theta_d^{jj} + d^j]}{(p - A_p^{jj} - B_p^{jj}\xi^{jj}\Theta_x^{jj})^2} \tag{5.49}$$

$$\frac{\delta e_a^j}{\delta\Theta_\alpha^{jj}} = \frac{B_p^{jj}\xi^{jj}f^j(x)}{p - A_p^{jj} - B_p^{jj}\xi^{jj}\Theta_x^{jj}} \tag{5.50}$$

$$\frac{\delta e_a^j}{\delta\Theta_d^{jj}} = \frac{B_p^{jj}\xi^{jj}\imath^j}{p - A_p^{jj} - B_p^{jj}\xi^{j}\Theta_x^{jj}} \tag{5.51}$$

Substituting equations 5.48 - 5.51 into 2.23, yields equations 5.52 - 5.55.

$$\frac{d\Theta_r^{jj}}{dt} = -\Upsilon^{jj}e_a^j\frac{B_p^{jj}\xi^{jj}r^j}{p - A_d^{jj}} \tag{5.52}$$

$$\begin{gathered}\frac{d\Theta_x^{jj}}{dt} = -\Upsilon^{jj}e_a^j\frac{x_p^j}{(p - A_d^{jj})} \\ \text{since } x_p^j = \frac{B_p^{jj}\xi^{jj}\Theta_r^{jj}r^j + (B_p^{jj}\xi^{jj}\Theta_\alpha^{jj} + \alpha^{jj})f^j(x) + B_p^{jj}\xi^{jj}\Theta_d^{jj}\imath^j + d^j}{p - A_d^{jj}}\end{gathered} \tag{5.53}$$

$$\frac{d\Theta_\alpha^{jj}}{dt} = -\Upsilon^{jj}e_a^j\frac{B_p^{jj}\xi^{jj}f^j(x)}{p - A_d^{jj}} \tag{5.54}$$

$$\frac{d\Theta_d^{jj}}{dt} = -\Upsilon^{jj} e_a^j \frac{B_p^{jj}\xi^{jj} i}{p - A_d^{jj}} \tag{5.55}$$

with $A_d^{jj} = A_p^{jj} + B_p^{jj}\xi^{jj}\Theta^{jj}$ (assuming $[\Theta^*]^{jj} = \Theta_x^{jj}$). Equations 5.52 - 5.55 are the adaption rates and are used to update the diagonal adaptive matrices.

5.4.2 Lyapunov Controller Design

The derivative of the error term in equation 5.42 is given by equation 5.56.

$$\dot{e}_a = \dot{x}_p - \dot{x}_d \tag{5.56}$$

Substituting equation 5.32 and 5.33 into 5.56 yields 5.57

$$\dot{e}_a = (A_p + B_p\xi\Theta_x)x_p + B_p\Theta_r r + (B_p\xi\Theta_\alpha + \alpha)f(x_p) + B_p\xi\Theta_d i + d - (A_d x_d + B_d r) \tag{5.57}$$

Substituting equations 5.34- 5.37 into 5.56 yields equation 5.58.

$$\begin{aligned}\dot{e}_a &= (A_d - B_p\xi\Theta_x^* + B_p\Theta_x)x + B_p\xi\Theta_r r + (B_p\xi\Theta_\alpha + \alpha)f(x) \\ &\quad + (B_p\xi\Theta_d + D)i - (A_d x_d + B_p\xi\Theta_r^* r) \\ &= A_d(x - x_d) + B_p\xi(\Theta_x - \Theta_x^*)x + B_p\xi(\Theta_r - \Theta_r^*)r \\ &\quad + B_p\xi(\Theta_\alpha - \Theta_\alpha^*)f(x) + B_p\xi(\Theta_d - \Theta_d^*)i\end{aligned} \tag{5.58}$$

The difference terms inside the parentheses represent the error between the ideal and actual matrices. This error can be represented by a $\tilde{\Theta}$ symbol. Equation 5.58 then becomes 5.59

$$\dot{e}_a = A_d e + B_p\tilde{\Theta}_x x + B_p\tilde{\Theta}_r r + B_p\tilde{\Theta}_\alpha f(x) + B_p\tilde{\Theta}_d i \tag{5.59}$$

Next we choose a Lyapunov function of the form (Narendra and Annaswamy, 2012)

$$J = \frac{1}{2}x^T P x + \frac{1}{2}Tr[\tilde{\Theta}^T\Upsilon^{-1}\tilde{\Theta}] \tag{5.60}$$

where P and Υ^{-1} are positive definite symmetric matrices.

This form is similar to that discussed in Chapter 2, except for the additional term. We want to drive the error vector to zero; the error vector therefore forms part of the first term. Additionally, we want to drive the error in the adaptive matrices to zero. The Lyapunov function needs to be a scaler, and therefore the trace operator is introduced as in the second term.

Substitute the different error terms into equation 5.60 to get

$$J = \frac{1}{2}e^T Pe + \frac{1}{2}\tilde{\Theta}_x^T\Upsilon^{-1}\tilde{\Theta}_x + \frac{1}{2}Tr[\tilde{\Theta}_r^T\Upsilon^{-1}\tilde{\Theta}_r] + \frac{1}{2}Tr[\tilde{\Theta}_\alpha^T\Upsilon^{-1}\tilde{\Theta}_\alpha] + \frac{1}{2}Tr[\tilde{\Theta}_d^T\Upsilon^{-1}\tilde{\Theta}_d] \tag{5.61}$$

For the system to be stable the derivative of equation 5.61 needs to at least be negative semi definite. Taking the derivative of equation 5.61 yields equation 5.62. For more information on how equation 5.62 was obtained see Appendix E.

$$\begin{aligned}\dot{J} = -\frac{1}{2}[e_a^T Q e_a + Tr[\tilde{\Theta}_x^T(\xi^T B_p^T P e_a x^T + \Upsilon_x^{-1}\dot{\tilde{\Theta}}_x)] + Tr[\tilde{\Theta}_r^T(\xi^T B_p^T P e_a r^T + \Upsilon_r^{-1}\dot{\tilde{\Theta}}_r)] \\ + Tr[\tilde{\Theta}_\alpha^T(\xi^T B_p^T P e_a f(x)^T + \Upsilon_\alpha^{-1}\dot{\tilde{\Theta}}_\alpha)] + Tr[\tilde{\Theta}_d^T(\xi^T B_p^T P e_a i^T + \Upsilon_d^{-1}\dot{\tilde{\Theta}}_d)]]\end{aligned} \tag{5.62}$$

Q is defined as a symmetric positive definite matrix, therefore if all but the first term is set to zero $\dot{J}$ will be negative definite as desired. Setting the last four terms to zero will help us to find how the the adaptive matrices update.

$$\tilde{\Theta}_x^T(\xi^T B_p^T P e_a x^T + \Upsilon_x^{-1}\dot{\tilde{\Theta}}_x) = 0 \tag{5.63}$$

$$\tilde{\Theta}_r^T(\xi^T B_p^T P e_a r^T + \Upsilon_r^{-1}\dot{\tilde{\Theta}}_r) = 0 \tag{5.64}$$

$$\tilde{\Theta}_\alpha^T(\xi^T B_p^T P e_a f(x)^T + \Upsilon_\alpha^{-1}\dot{\tilde{\Theta}}_\alpha) = 0 \tag{5.65}$$

$$\tilde{\Theta}_d^T(\xi^T B_p^T P e_a i^T + \Upsilon_d^{-1}\dot{\tilde{\Theta}}_d) = 0 \tag{5.66}$$

It is assumed that the ideal adaptive matrices remain constant. The change in the error term is therefore due to the change in the actual adaptive matrices $\dot{\tilde{\Theta}} = \dot{\Theta}$. The derivative of the adaptive matrices are given by equations

$$\dot{\Theta}_x = -\Upsilon_x \xi^T B_p^T P e_a x_p^T \tag{5.67}$$

$$\dot{\Theta}_r = -\Upsilon_r \xi^T B_p^T P e_a r^T \tag{5.68}$$

$$\dot{\Theta}_\alpha = -\Upsilon_\alpha \xi^T B_p^T P e_a f^T \tag{5.69}$$

$$\dot{\Theta}_d = -\Upsilon_d \xi^T B_p^T P e_a i_{vec}^T \tag{5.70}$$

These derivatives are integrated in order to update the adaptive matrices. The control law results in the moments required around all three axes for attitude control. Figure 5.8 summarises the adaptive loop.

5.4.3 Modifications

A modification can be done in order to improve the workings of the adaptive controller. The modification discussed is that of σ-modification and again follows the same approach as Achtelik *et al.* (2011). With longer flights the adaptive parameters can drift. Adding a damping term to the adaption rate equation limits the growth of the terms. The rates are decreased proportional to how big the adaptive gain is at that point. Simple σ-modification is given by equation 5.71.

$$\dot{\Theta}_x = \Upsilon_x \xi^T B_p^T P e_a x_p^T - \Upsilon_x \sigma \Theta_x \tag{5.71}$$

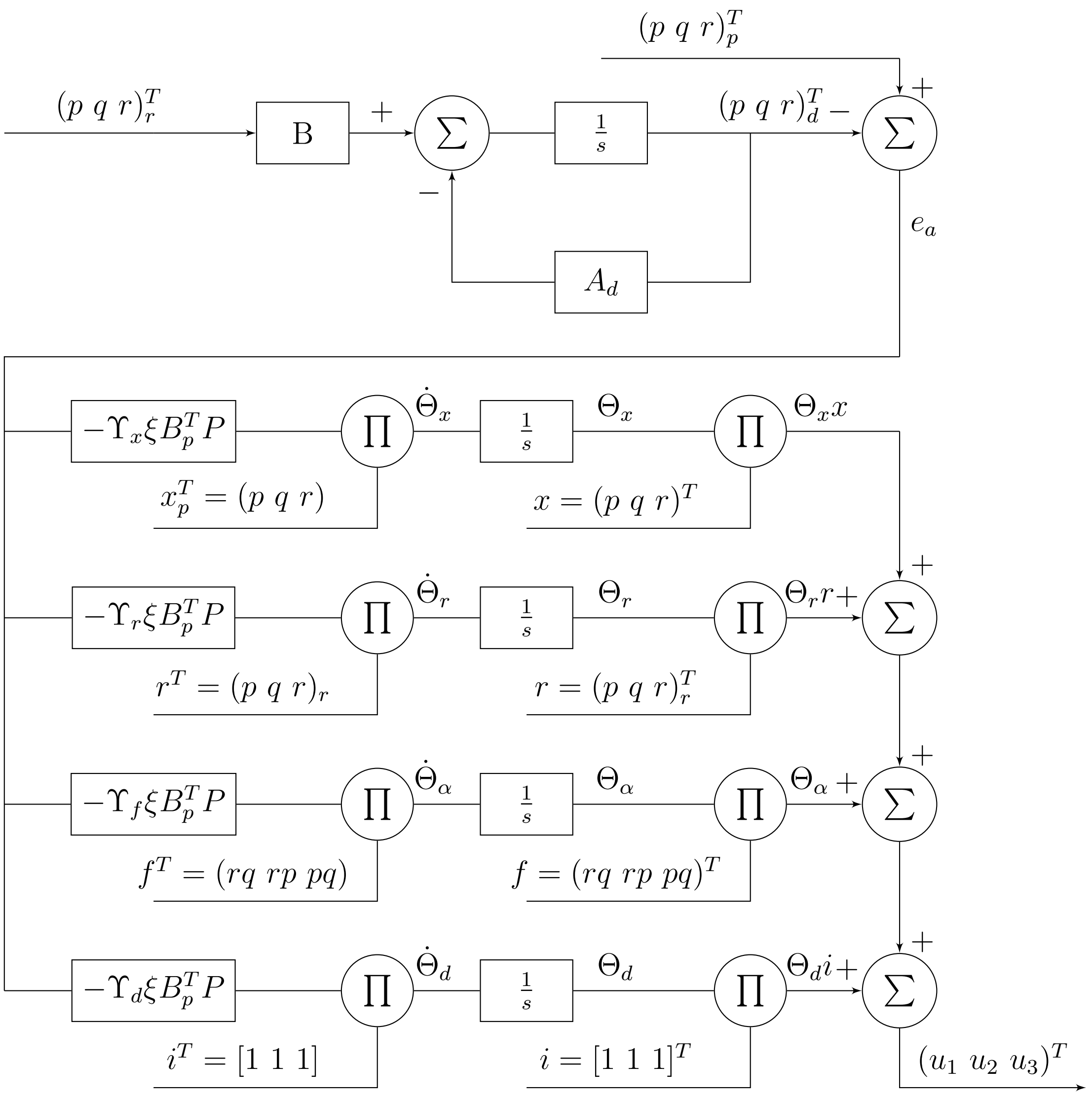


Figure 5.8: Inner workings of MRAC loop

This approach is, however, faulty if the system behaves as desired. The e_a term then approximates zero and the adaption rate equation is dominated by the second term. Since this term is negative the adaptive gain will be driven towards zero. Ideally we would like the damping term to only contribute when the actual system is not behaving like the desired system. Scaling the damping term with the norm of the error allows us to regulate the contribution of the damping term according to how well the system is behaving.

$$\dot{\Theta}_x = \Upsilon_x \xi^T B_p^T P e_a x_p^T - \Upsilon_x \sigma \Theta_x ||e||_2 \tag{5.72}$$

The above equation still poses a problem. If the error in one axis is large it will result in large derivative terms for all the axes (since the e-norm increases), even those where a small error is present. The large derivative term in the axes where the error term is small again yields the same problem as before and those adaptive gains are driven to zero. An additional scaling term is therefore added in order to decouple the axes.

$$\dot{\Theta}_x = \Upsilon_x \xi^T B_p^T P e_a x_p^T - \Upsilon_x \sigma \Theta_x ||e||_2 \begin{bmatrix} |e_x| & 0 & 0 \\ 0 & |e_y| & 0 \\ 0 & 0 & |e_z| \end{bmatrix} \tag{5.73}$$

5.5 Simulation Results

A few simulations were run before moving on to practical testing. A square wave was given to the quadcopter to track along its roll axis, with the initial adaptive parameters set to zero. The controller would then need to adapt from zero and applying a square wave allows one to see how the controller adapts until sufficient tracking of the reference model is achieved. Once the controller has reached its ideal parameters from excitation, the tests aimed to be implemented practically were also simulated. These tests are also evaluated against other control approaches. The tests include:

1. A step input around the roll axis of 20°.
2. A step disturbance in the roll moment command of 0.2.
3. The addition of a weight on one of the arms.

The control approaches to be compared are that of simple PID control, the PX4 controller, adaptive control and the adaptive structure with adaption switched off after convergence of the parameters.

5.5.1 Square Wave Learning

In simulation it is possible to set the initial adaptive gains to zero. The quadcopter is then excited by means of a square wave around its roll axis. The square wave varies between 0° and 20° with a period of four seconds. Figure 5.9 shows the angle response of the quadcopter around all three of its axes. The Figure also includes the reference model generated by the non-linear dynamic inversion loop. This would be the quadcopter response if it followed the commanded rates perfectly.

Within the second cycle, the roll angle behaves as desired. The difference in the outer reference model angle and the response angle is due to the rate set point not being instantaneously achievable, since an instantaneous step in velocity is not possible. The pseudo control hedging does assist in slowing down the reference model slightly, however, the main reason for the difference lies within the adaptive control loop. When looking at the adaptive loop reference model, it is seen that immediate tracking of the angle rates is not desired.

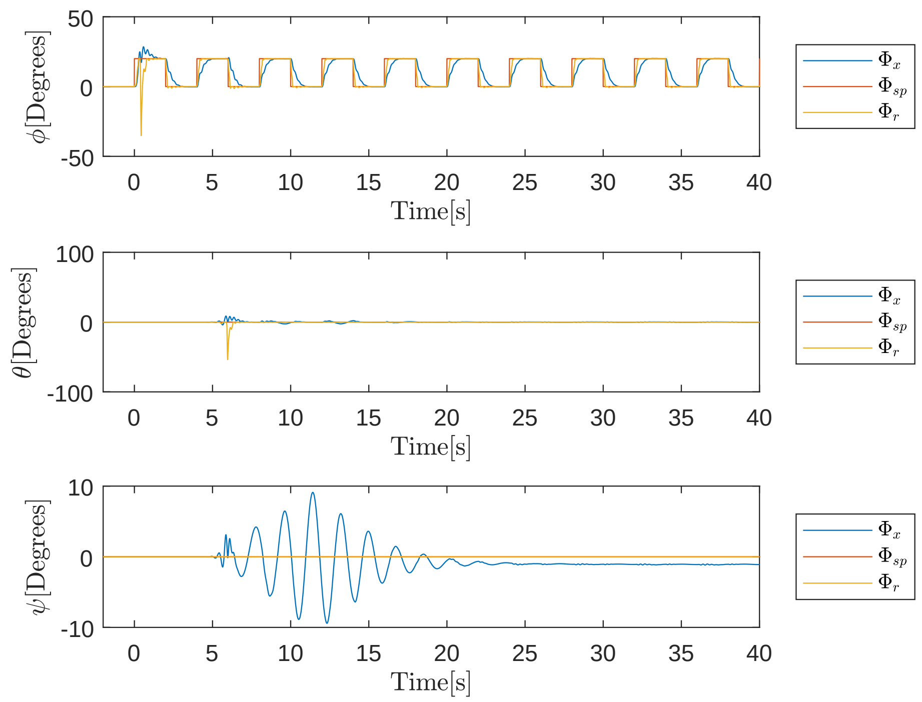


Figure 5.9: Simulation angles for square wave learning

Equation 5.32 shows how the rates should respond to a change in rate set points. The rate response is therefore gradual, not instantaneous. Figure 5.10 shows how the adaptive reference model rates are gradual, while the set point rates are instantaneous.

When considering the pitch and yaw axes, oscillations can be noticed after roughly five seconds. The reason for these delayed oscillations are due to the fact that the quadcopter is not being excited around those axes. The adaptive gains responsible for the control around those axes are still approximately zero, since no error has occurred to drive these adaptive gains. The step at zero seconds around the roll axis causes a minor movement in the pitch and yaw axis, so small that it is not yet visible without enlarging the scale of the y-axis given in the Figure by a factor of 10^8. Therefore it does not yet yield a large enough angle error around those axes to lead to adaption of the gains affecting those axes. Eventually having no correction being made, due to adaptive gains of approximately zero, the oscillations grow. Large oscillations around the pitch axis start occurring at around 7 seconds. The adaption around those axes is then initiated and the error is driven back to zero.

Figure 5.10 shows the angular rate response of the quadcopter around all three axes. The reference model of the adaptive loop along with the set point commanded from the angle loop is also included. The difference between the adaptive loop reference model and the measured angular rates is what drives the adaption.

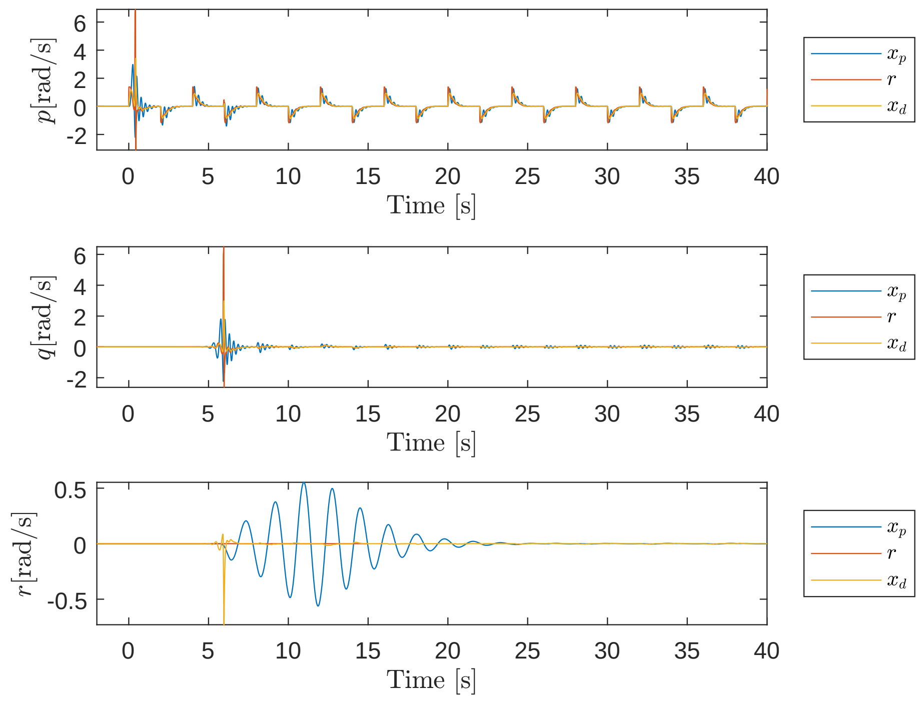


Figure 5.10: Simulation angle rates for square wave learning

Similar to the angle case, the angular rates around the pitch and yaw axes take a while to show fluctuations. After two cycles the roll rate response closely follows the adaptive roll rate of the adaptive reference model, except for some oscillation of the measured rate superimposed on the signal. The adaptive values for the roll axis are therefore close to their ideal values at this point. Slight fluctuations associated with the roll axis still occur in Θ_x, due to the superimposed oscillations in the measured value, but the mean adaptive gains for the roll axis have essentially converged. This is evident from Figures 5.11-5.14. This slight unnecessary oscillation can be removed with a dead zone as in the practical case.

The 3 subgraphs of Θ are explained as follows. The first graph contains the first row elements, the second the second row elements and third the third row elements. Each graph therefore contains the separate gains with which each angular rate needs to be multiplied. Plot 1 therefore contains the gains that effect the roll moment command. Plot 2 contains the gains that affect the pitch moment command and plot 3 the gains that affect the yaw moment commands. It is important to note that the matrix is dominated by the diagonal elements. This essentially shows that errors in pitch and yaw rates, have less of an effect on the roll command, as would be expected.

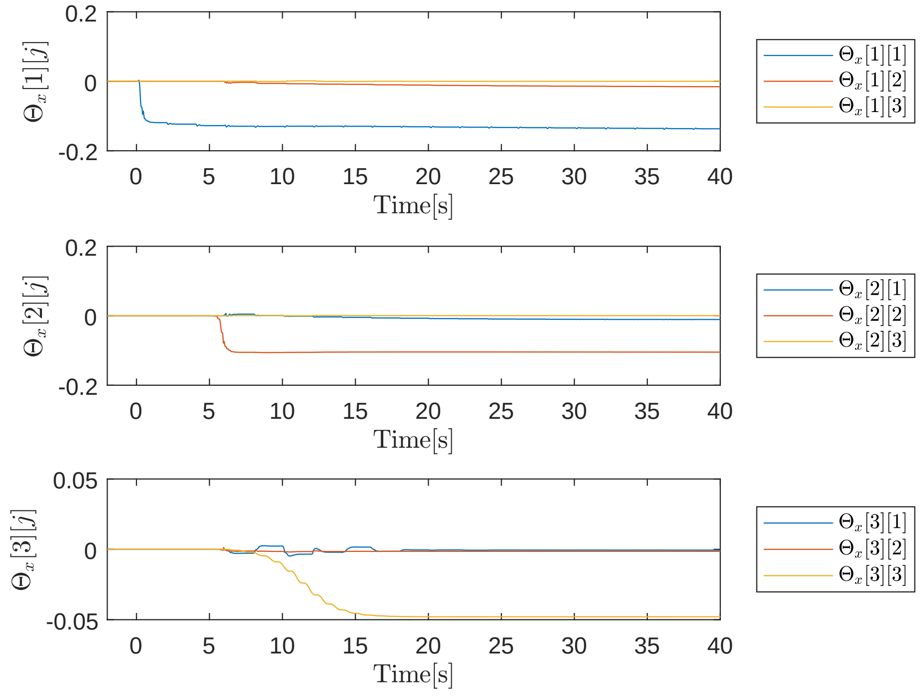


Figure 5.11: Square wave learning simulation, Θ_x

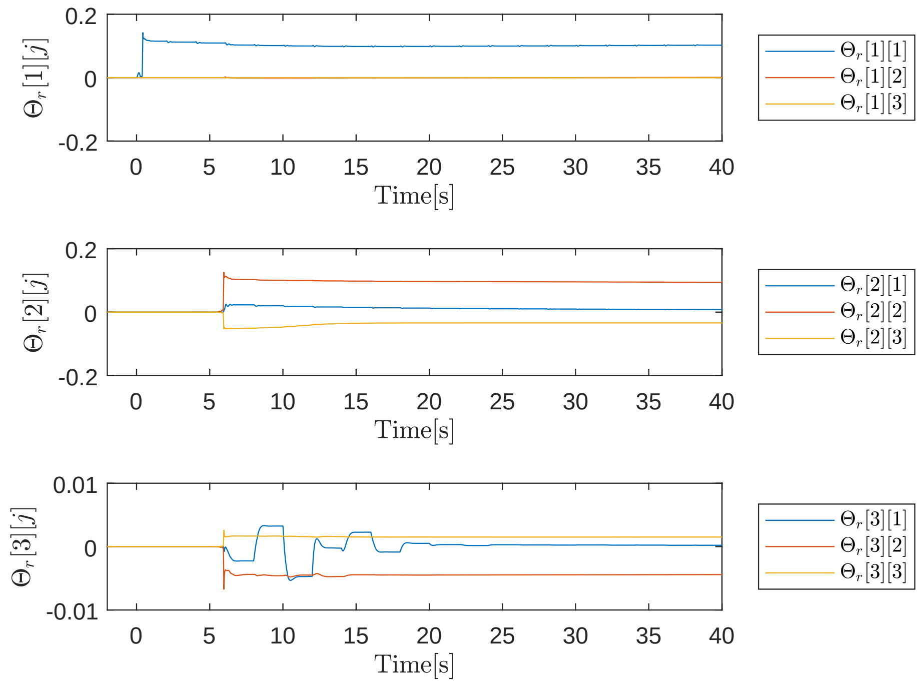


Figure 5.12: Square wave learning simulation, Θ_r

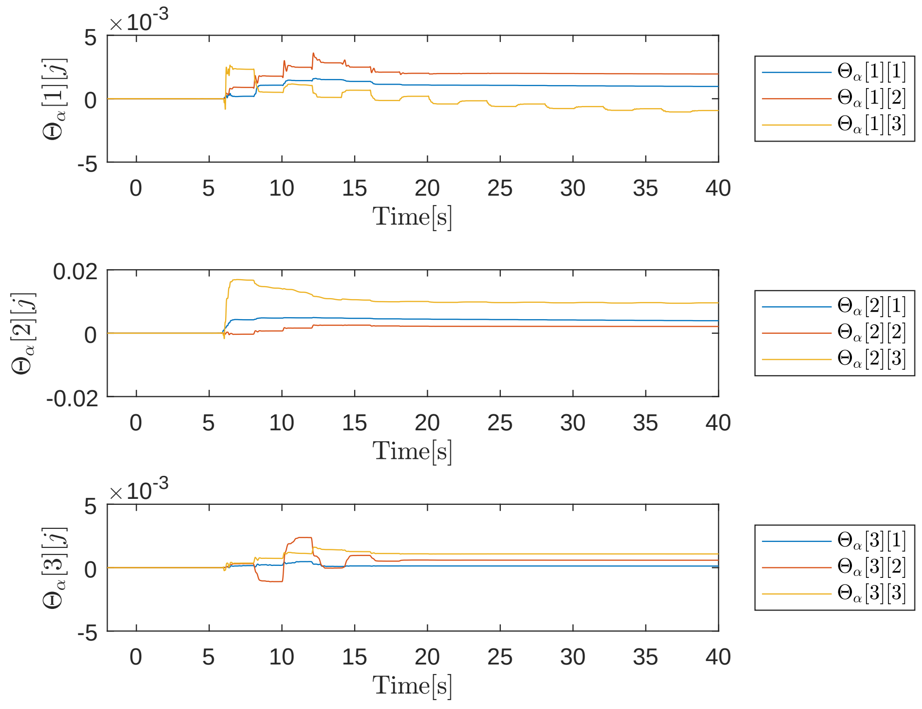


Figure 5.13: Square wave learning simulation, Θ_α

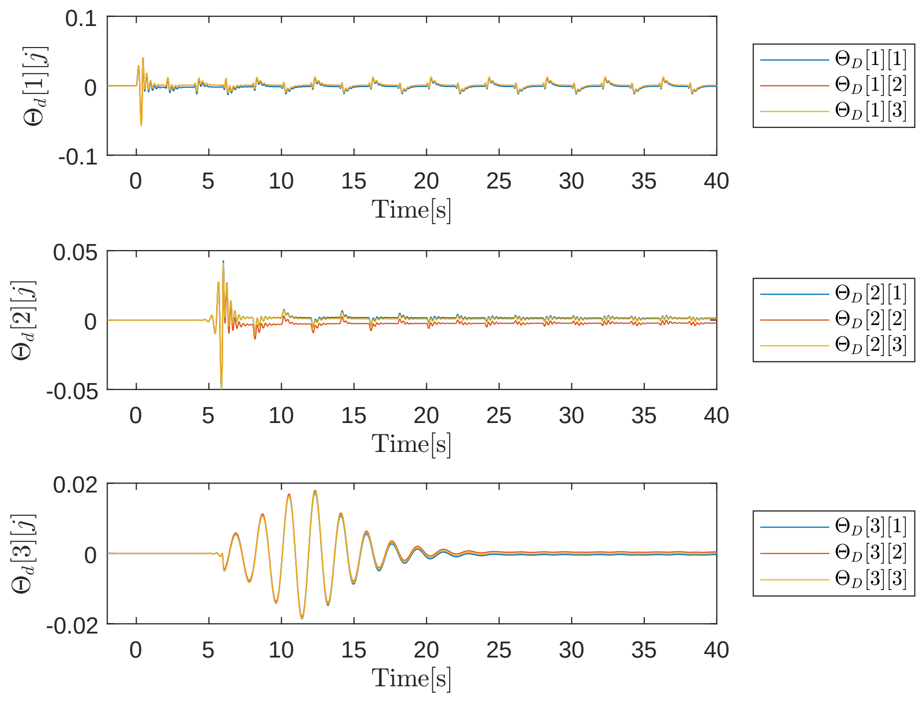


Figure 5.14: Square wave learning simulation, Θ_d

The pitch and yaw adaptive gains take longer to reach steady values. They also do not converge to their ideal values since the quadcopter is not directly excited around these axes. The alpha matrix gains are very small and almost negligible in comparison with the others. This is due to the fact that the alpha gains for an axis are driven by the product of angular velocity of the measured values of the other two axes. In most cases at least one of these are small yielding small adaptive gains. Aggressive simultaneous inputs can lead to larger alpha gains. Since different axes' rates drive adaption of another axis's adaptive gains the matrix is not diagonally dominant. All matrix elements are therefore plotted in the practical case.

The alpha gains grow very quickly once at the verge of instability since the errors in rates and rates themselves become very large. That is why the rate of adaption was not increased. The values on the rows of the disturbance matrix should be similar. This is due to a vector being multiplied by the transpose of a vector of ones (equation 5.70).

5.5.2 Step Response of 20 Degrees

After exciting the quadcopter around all three of its axes, giving the adaptive gains chance to reach their ideal values, the controller is evaluated in simulation against other controllers. The competing controllers are: a simple PID controller, the PX4 controller and the adaptive controller with adaption switched off. The controller gains chosen for the PX4 controller are the same gains that were used practically with quadcopter B. The control gains were refined practically in order to optimise the response on quadcopter B. The simple PID controller gains were optimized in simulation to optimise the response of the quadcopter. The control parameters used can all be found in Appendix F. The first test done is giving the quadcopter a step input of 20° around the roll axis. Figure 5.15 shows the angle response of the four mentioned controllers.

The PX4 controller proofs to be the best of the four according to the response shown in Figure 5.15. As mentioned later, in the practical section, the speed of the adaptive controller's response can be improved by using a faster reference model. The slightly slower reference model is, however, chosen to allow the code to be able to be easily implemented on quadcopters of different sizes. A larger quadcopter would for instance not be capable of achieving this response. A faster reference model was used in practice with the smaller quadcopter and a response similar to that of the PX4 was achieved, see Figure D.22. Table 5.1 summarises the response for the 20° step input around the roll axis.

Additional plots for the roll step input of 20° can be found in Appendix D. Figure D.1 further shows that the PX4 controller demands more from the actuators than the other controllers, since a higher angular rate response is

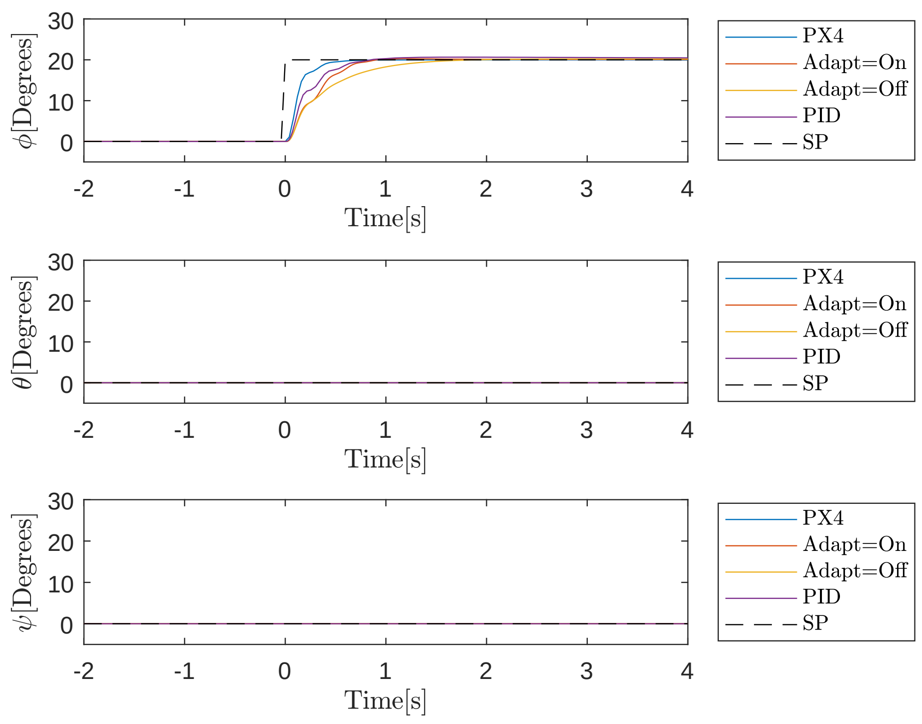


Figure 5.15: Combined simulation angles for 20 degree step input

Table 5.1: Simulation response summary for step input

Axis	Specification	PX4	Adaption On	Adaption Off	PID
Roll	Settling time (90%)	0.37 s	0.61 s	0.98 s	0.55 s
	Overshoot	0°	0°	0.6°	0°
	Steady state error	0°	0°	0°	0.6°

achieved as expected. The adaptive parameter have already converged by the time of the step input. Therefore Figures D.2- D.5, show little fluctuation and remain relatively constant. It is also noticeable that the adaptive matrices are diagonally dominant.

5.5.3 Internal Disturbance

One of the important aspects of an adaptive controller is how well it handles disturbances. Applying a disturbance to a quadcopter practically is quite difficult. As an initial test it was decided to apply a virtual disturbance in the code itself. Figure 5.16 shows where the disturbance is introduced. In reality the disturbance is introduced after the propeller dynamics block. Since the mixer and propeller dynamics blocks essentially only result in scaling by a constant factor, at a given moment command, the use of the internal disturbance should deliver an appropriate approximation. It was found that applying an external disturbance of 0.08 Nm yielded the same response as with a 0.2 moment command internal disturbance. This would, however, not be the case in practice. An additional test, as described later, was therefore done after stability was proven in this test. Success in this test was therefore desired, before being able to move on to a more dangerous test.

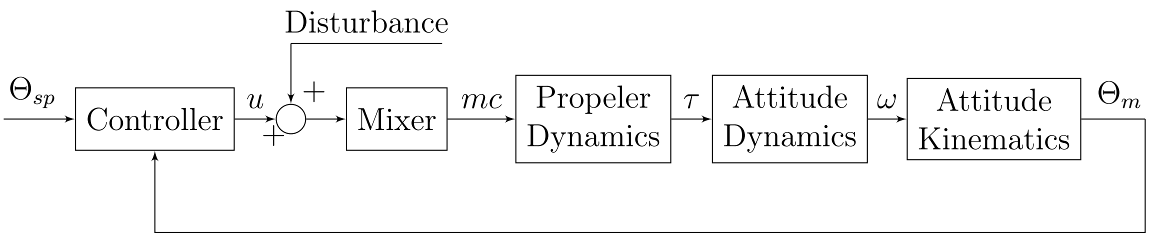


Figure 5.16: Block diagram of control approach

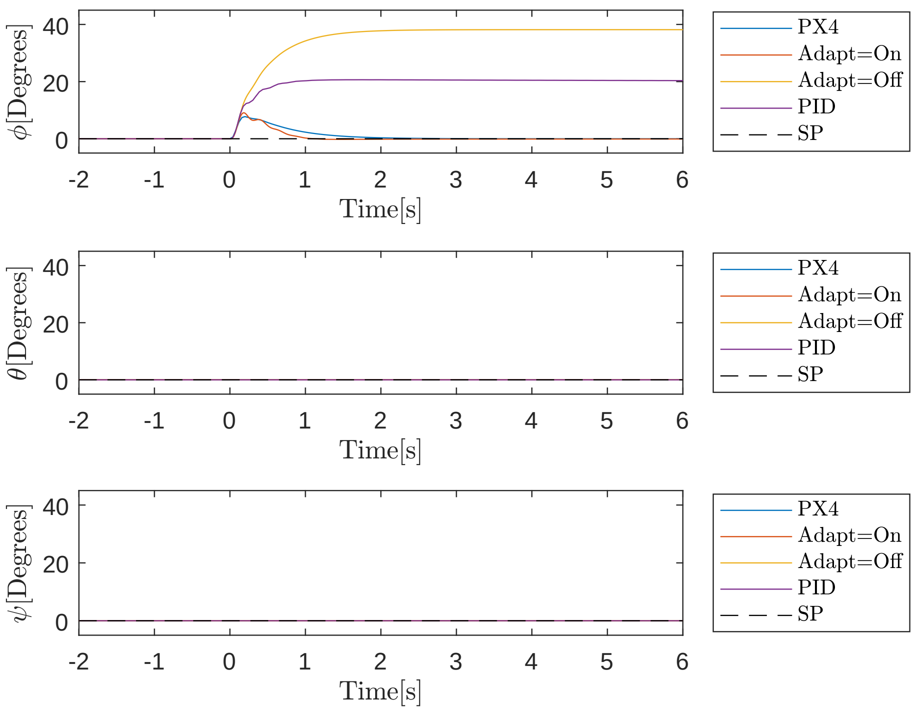


Figure 5.17: Combined simulation angles for 0.2 N.m. disturbance

A step value of 0.2 is added to the roll moment command in order to replicate a disturbance. The angle response of the system is given in Figure 5.17. The controller with adaption switched off fairs the worst. This is due to the fact that it does not have an integral component in the rate loop like the PX4 controller has, and due to the adaption being switched off, the disturbance rejection term is not active. The basic PID controller fairs far better, although the steady state error is still too large when compared to the PX4 case. The integrator reduces the steady state error, but does not eliminate it.

Making use of a cascade controller, such as in the case of the PX4 controller, significantly improves the disturbance rejection. The roll angle peaks at approximately 8° and reduces to zero within 2.5 seconds. The adaptive controller has a slightly larger peak value, approximately 9°. The adaptive controller, however, reaches steady state in approximately half the time, 1.3 seconds. The effect on the pitch and yaw axes are negligible in all the controller cases. The roll axis responses are summarised in Table 5.2.

Table 5.2: Simulation response summary for internal disturbance

Axis	Specification	PX4	Adaption On	Adaption Off	PID
Roll	Return to zero	2.5 s	1.3 s	∞	∞
	Overshoot	8°	9°	38°	19°
	Steady state error	0°	0°	38°	19°

Additional plots for the internal disturbance around the roll axis can be found in Appendix D. Figure D.6 shows the angular rates for the different controllers. The PX4 controller has a smaller fluctuation in the roll rate, this however, is the reason for the steady state error taking longer to reach zero when compared to the adaptive controller. The slight oscillation in the roll rate for the adaptive controller is to be expected. The controller itself adjusts to counter the disturbance; due to fast adaption some overshoot of the ideal parameters can occur. This can be seen in the case of the Θ_d matrix. Figures D.7-D.10 show the adaptive gain matrices. Again, the off diagonal elements of Θ_x and Θ_r are zero.

Figure D.7 shows that the gain associated with roll measurement $\Theta_x[1][1]$ has a small increase in magnitude when the disturbance occurs. Due to the disturbance, the system moves faster than expected. The controller therefore tries harder to slow down the system, yielding an increase in the gain associated with roll measurement $\Theta_x[1][1]$.

The values of the alpha matrix are negligible when compared to the other matrices. This is expected since the system only experiences one angular velocity. The non-linearities are therefore minimal.

The majority of the disturbance is absorbed by the disturbance matrix, as is desired. This can be seen by multiplying the matrix by a vector of ones as done in the control law. The roll element of the vector yields a value of close to 0.2, which is equal to the disturbance value given in the code.

5.5.4 Addition of Off-centre Weight

The internal disturbance yielded an easy and safe method to test the disturbance rejection of the different controllers practically. This, however, gives the ideal case. After seeing how the system behaves to the internal disturbance practically a more realistic test could be devised. One of the quadcopter's arms is extended. A weight is then added to this extension. This yields an moment around the pitch and roll axes. Figure 5.18 shows how the weight is attached to the quadcopter by means of a top view.

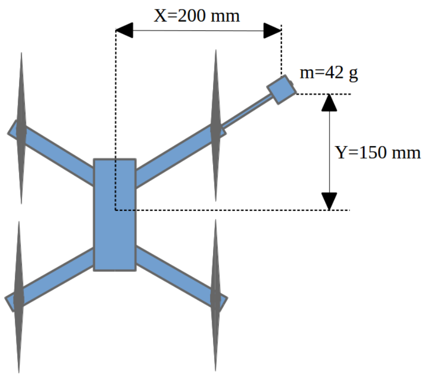


Figure 5.18: Quadcopter with off-centred weight

The moment around the roll axis is determined as 0.0824 N.m. and the moment around the pitch axis is determined as 0.062 N.m. The response of the system is simulated to these two moments as simultaneous disturbances. Figure 5.19 shows the attitude angle response.

This is a good test since all three axes are affected. The roll axis shows the largest peak, which is to be expected since it corresponds to the larger moment disturbance. The different controllers compare similarly to how they compared with the virtual disturbance. The controller with adaption switched off and the basic PID controller fair poorly. The adaptive controller and the PX4 controller again prove to be close. These two are therefore discussed further. In the case of the roll and pitch angles the peak values of the two controllers are indistinguishable. The settling time of the adaptive controller is, however, faster. Around the yaw axis the adaptive controller is superior to all the others. It has the smallest overshoot and steady state error. The basic PID controller fairs better around the yaw axis while the controller with adaption switched off still fairs poorly. The result of the angle responses for the off-centre weight disturbance is summarised in Table 5.3.

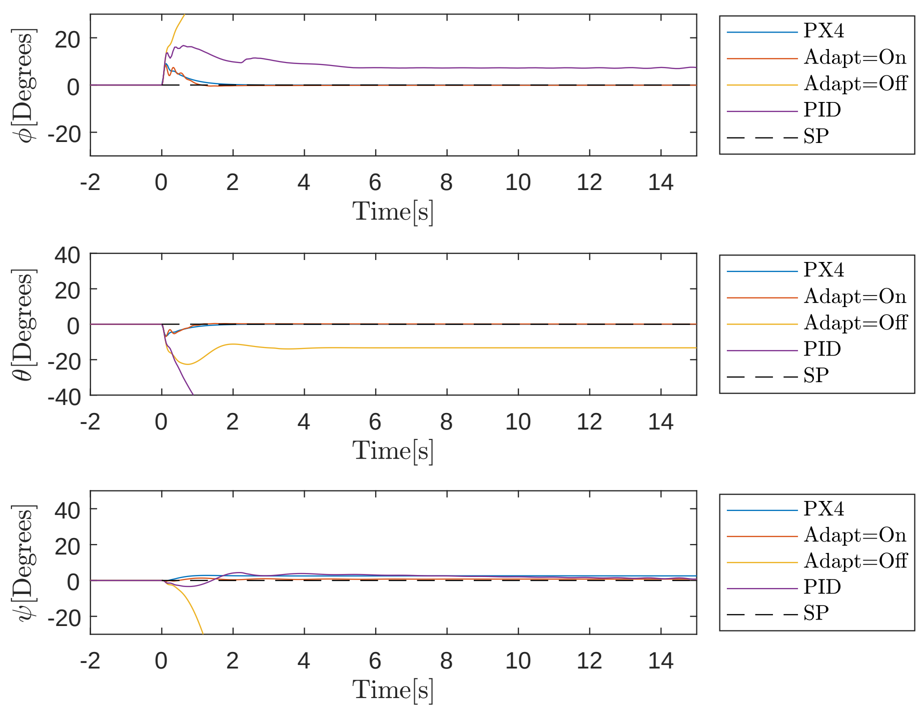


Figure 5.19: Combined simulation angles for off-centre weight

Table 5.3: Simulation response summary for off-centre weight

Axis	Specification	PX4	Adaption On	Adaption Off	PID
Roll	Return to zero	1.4 s	1 s	∞	∞
	Overshoot	9.3°	8.6°	36°	16.8°
	Steady state error	0°	0°	36°	7°
Pitch	Return to zero	1.5 s	0.8 s	∞	∞
	Overshoot	6.8°	6.6°	23°	86°
	Steady state error	0°	0°	13°	86°
Yaw	Return to zero	0.5 s	0 s	∞	2.5 s
	Overshoot	2.5°	1°	∞	4.7°
	Steady state error	2.4°	0.5°	∞	0.5

Additional plots for the off-centred weight test can be found in Appendix D. Figure D.11 shows the angular rates for the different controllers. Similar to the internal disturbance case, the PX4 controller has smaller fluctuations in the angle rates; this, however, is the reason for the steady state errors taking longer to reach zero when compared to the adaptive controller. The slight oscillations in the angular rates for the adaptive controller are again due to fast adaption. Figures D.7-D.10 show the adaptive gain matrices. In this case the off-diagonal elements do not remain zero due to simultaneous motion around all three axes. They, however, remain diagonally dominant. Figure

D.7 also shows that the gains associated with roll measurement $\Theta_x[2][2]$ and pitch measurement $\Theta_x[1][1]$ have a small increases in magnitude when the disturbances occur. The reason for this is the same as what it was in the internal disturbance case. The off-diagonal elements $\Theta_x[1][2]$ and $\Theta_x[2][1]$ also show an increase.

Again, the values of the alpha matrix are negligible when compared to the other matrices. The majority of the disturbances are absorbed by the disturbance matrix Θ_d, as is desired. This can be seen by multiplying the matrix by a unity vector as done in the control law. The roll element of the vector yields a value of 0.173 while the pitch element yields a value of 0.131. These values are not equal to the moment disturbances given around the respective axes. This is due to the differences that exist between the moment command values and the actual moments that were applied. The values differ by a factor of approximately 2.1.

5.5.5 Effect of Change in Model

One of the advantages of an adaptive controller is that the gains converge to their ideal values. This removes the need for manually adjusting the control parameters when a change in model occurs. The parameters which can be chosen and their effect are important. The proportional and integral error gains of the angle loop are additional components and not primary control parameters. They can therefore remain unchanged. As mentioned previously, parameters defining the reference models are chosen so to be achievable by most quadcopters and therefore also remain unchanged. That leaves Υ, which determines the rates of adaption. Looking at equations 5.67-5.70 it can be seen that the derivative of adaption depends on B_p and ξ, which are unique for different quadcopters. Larger quadcopters have larger motors and longer lever arms yielding to a larger scaling factor between the moment command and achieved moment. Larger quadcopters, however, also have larger inertias, yielding smaller values for the B_p matrix. The increase in ξ and decrease in B_p therefore oppose another, yielding similar adaptive gains despite a change in model. The same values of Υ are therefore used again. To view how the exact same controller fairs with two different models, a square wave was given as reference roll angle. This was done with quadcopter A and B. The simulation results are given in Appendix E, Figures D.16-D.21.

The larger quadcopter shows more oscillation in the angle response which could be indicative of adaption being too fast. This needs to be taken into account in the practical case. The large quadcopter takes longer to experience oscillation around the other two axes. This is due to the larger inertias. The values of Θ_x and Θ_r for quadcopter A and B are similar to one another, supporting the fact that the changes in B_p and ξ opposes one another. The difference in adaptive gain matrices might look small, although this difference can drastically effect flight. For instance the PX4 controller, specifically tuned for quadcopter A, was practically tested on quadcopter B and resulted in a very shaky flight.

5.6 Practical Implementation

5.6.1 Firmware Modification

The PX4 firmware was already discussed in Chapter 3. Before the firmware could be modified in order to implement the designed controller the overall code needed to be studied. A brief overview of the PX4 code structure is therefore required and given in Appendix C. The only part of the PX4 code which is extensively modified is the “mc_attitude_control" file. The original and modified code are discussed. A brief discussion of the mixer file is also given since understanding this file is important in understanding the code.

5.6.2 Testing Procedure

The PX4 software provides a hardware in the loop (HIL) simulation feature. This allows one to test the firmware on the real processor by means of a simulator (PX4, 2017*c*). HIL simulation provides a good starting point before moving on to practical flight tests.

Initial practical testing of any control system is dangerous since the system can be unstable. This is even more the case with of quadcopter since it is not fixed and has spinning blades. A test rig was therefore designed and built for initial testing to at least first check for stability. The design and construction of the test rig is discussed in Appendix A. Once positive results were seen on the test rig, testing could be done on the unrestricted quadcopter. It is noteworthy that the gains showing positive results on the test rig were far larger than the gains showing positive results in free flight. This is due to restrictions imposed due to the test rig. The adaption could not be tested successfully on the test rig, since the restrictions placed on the quadcopter did not allow the adaptive gains to converge to a value. The adaptive component was therefore tested freely.

The first goal for practical testing was to achieve stable flight with adaption switched off. Initial values for the adaption matrices were therefore required. These can be achieved in two ways: calculated from inertias (ideal gains assuming inertias are perfectly known) or using a LQR controller design approach. Before moving on to practical testing, even before testing on the test rig, a good understanding of how the different parameters will affect the flight of the quadcopter is required. The effects of the different gains are therefore discussed first.

5.6.3 Effect of Different Gains

It is important to understand how the different parameters will affect the flight of the quadcopter. The parameters for the angle control loop and angular rate control loop with adaption switched off are summarised in Table 5.4.

Table 5.4: Summary of control parameters with adaption switched off

Control loop	Parameter	Description
Angle	$\frac{1}{T}$	Reference model time constant
	K_y	Rate reference elimination
	K_p	Error in angle
	K_i	Error in angle intergral
Rate	Θ_x	Measured rate gain
	Θ_r	Slow changing ref rate gain
	Θ_{der}	Derivative in rate error
	Θ_{int}	Intergral in rate error)

The control equations are repeated below with slight modifications which are purely for illustrative purposes in this section. Equation 5.1 remains the same.

$$\dot{\Phi}_r(t) = \frac{1}{T}(\Phi_{sp} - \Phi_r) \tag{5.74}$$

A coefficient K_y is added to the first term of equation 5.3. This coefficient is either a one or zero in order to pass or eliminate the first term.

$$v = K_y\dot{\Phi}_r + K_p e + K_i\frac{e_{angle}}{s} \tag{5.75}$$

Considering the rate loop, a derivative term is added along with an integrator term.

$$u = \Theta_x x_p + \Theta_r \omega_r + \Theta_{der} x_p s + \Theta_{int}\frac{e_{rate}}{s} \tag{5.76}$$

The effect of each of these parameters is discussed. Increasing $\frac{1}{T}$ will increase the speed of the response of the reference model. Increasing the value results in an increase in the required rate to achieve the desired fast response. The quadcopter therefore needs to be capable of attaining the generated rates. Decreasing the value will effectively increase the damping of the system to set points.

K_y is added in order to be able to eliminate the rate term generated by the reference model. Eliminating this term will not affect simple hover, but it will reduce the desired rate in order to achieve set points. An increase in K_p will therefore be required. In the ideal case the actual quadcopter will follow the reference model perfectly and K_p can be zero. Slight corrections are, however, required. Increasing the parameter too much can lead to over correction which leads to instability. The integral term, K_i is simply to eliminate steady state error.

Θ_x attempts to slow the quadcopter down by generating an opposing torque to which direction it is rotating. It is important to slow down the quadcopter, although if the term becomes too large the counter moment can become excessive and cause oscillation. Θ_r is multiplied by the reference rates in order to produce moment commands to achieve these reference rates. Increasing Θ_r excessively leads to overshoot. This can be countered by also increasing Θ_x although this can lead to saturation of the motors. Sensor noise then also has a more prominent effect. As previously mentioned Θ_{der} provides additional damping. Θ_{int} is again to eliminate steady state error, but this time in the angular rates.

The PX4 controller can be mimicked if: $\Theta_x = \Theta_r = K_p(\text{PX4})$, $\Theta_{der} = K_d(\text{PX4})$, $\Theta_{int} = K_i(\text{PX4})$, $K_y = 0$, $K_i = 0$ and $K_p = K_p(\text{PX4 angle loop})$.

5.6.4 Modifications Due to Practical Observation

In the case of the Lyapunov controller it was found that adding an additional derivative term to the control law added additional damping that led to a more stable hover. It was found not to have much of an negative effect on the response time of the quadcopter.

When the quadcopter is stationary, on the ground, a small error in the attitude will be present due to an uneven surface. This error will cause the integrator in the angle control loop to grow. In order to avoid this integrator wind-up, the integrator error is only updated once the thrust set point exceeds the thrust required for take-off. Updates are also postponed when:

1. one of the motor commands exceeds the upper or lower limits (before adding yaw to mixer),
2. one of the motor commands exceeds the upper or lower limits (after adding yaw to mixer),
3. the integrator itself reaches its limits,
4. or the moment command is excessively large (larger than the thrust command).

This is the same approach followed in the original PX4 controller for its inner rate loop. The adaptive matrices pose the same problem. The failure of the stationary quadcopter to match the reference model will lead to excessive growth of the adaptive matrices. The adaption is only activated once the thrust set point exceeds the thrust required for take-off. The adaptive matrices as well as the integrators are reset upon arming of the quadcopter.

In order to guard against instability due to excessive gains, they should be limited in the practical system. Practical testing was used to see how large the gains can be made before the quadcopter became unstable and are summarized in Table F.1.

To avoid unnecessary adaption, when the quadcopter is behaving close to its ideal case, the adaption is switched off. This occurs when the error driving the adaption, as defined in equation 5.42, falls within a specified tolerance. This "dead zone" also helps to stop the adaptive matrices from adapting due to sensor noise. The quadcopter was allowed to hover in ideal conditions. The fluctuations in the error was measured and used to determine the tolerance as ± 0.5 rad/s. The practical parameters used for control loops, as depicted in equations, 5.1, 5.3 and 5.29, are summarised in Table F.1 in Appendix F.

5.7 Practical results

The final control system is evaluated against the existing PX4 controller and also to the controller itself when adaption is switched off. Different tests are used to measure the controller success. The tests implemented are as listed below. It is noteworthy that both these disturbances are instantaneous and not oscillatory.

1. Initial parameter convergence
2. A step response to 20° roll input
3. An internal disturbance
4. Addition of an off-centred weight
5. Acrobatic flight
6. An autonomous mission

The first four tests are similar to the tests simulated in the simulation section. The last 2 tests are to check whether the implemented controller integrates well with the existing PX4 software and that all flight modes are still possible. The majority of these tests are again done on multiple controllers for comparative purposes. The PID controller used in the simulation section is neglected in the practical section since it proved to be the worst of the existing controllers. The first test only considers the adaptive controller since the purpose of the test is to show parameter convergence which are not applicable in the other controllers. Tests 2 and 3 are done with the PX4 controller, adaption-on controller and adaption-off controller. Test 4 neglects the adaption off controller due to safety concerns. Test 5 and 6 are not tests to compare controllers, but show integration with the existing PX4 code. It is also difficult to replicate the exact flight conditions in test 5. The adaptive controller is therefore the only controller considered. Test 6 is easy to replicate, it is therefore done with the adaption on and PX4 controller to serve as an additional comparable test.

5.7.1 Initial Parameter Convergence and Flight Test

In the ideal case the adaptive controller is able to take off with the adaptive gains set to zero. Once the quadcopter is off the ground, errors in the angular rates occur and the adaptive controller drives the adaptive gains in order to minimise the error in the angular rates. In the practical case setting the adaptive gains to zero is not possible. The adaptive matrices can only converge to their ideal values once the quadcopter is in the air. The quadcopter therefore effectively needs to take off without any control system since the gains are set to zero. Also, an error is introduced if the take-off surface is even slightly uneven. The adaptive matrix values' sign can then change, increasing instability. In an attempt to counter this, adaption only starts once a certain thrust value is specified. In the case of the larger quadcopter the quadcopter can still be close enough to the ground and due to the design of the legs it can get hooked in the grass leading to unnecessary adaption. Achtelik *et al.* (2011) proposed holding the quadcopter by hand during take off. This approach was, however, dismissed due to safety concerns. In the practical case, small initial values are chosen just to get the quadcopter off the ground. The diagonal elements of the Θ_x and Θ_r matrices were therefore set to initial values of -0.01 and 0.01, respectively. The quadcopter was given a throttle command and allowed to stabilise, after which it was excited around the three axes. Figure 5.20 shows the angle response of the system for this test.

After about 4 seconds it seems that the quadcopter achieves stable hover. It is then excited around the roll axis. The measured response shows good tracking of the set point. The other two angles also remain within close range of their set points, proving that excitation around one axis does not have a negative effect on the other axes. Similar results are shown for excitation around the pitch axis, at fifteen seconds. Two rotations are then given around the yaw axis at twenty seven seconds. A slight overshoot in the yaw axis occurs. Since the yaw rates are commanded by the pilot, the change in rate set points are determined by the pilot and not the controller. The change in yaw rate can therefore possibly be higher than the rates for the roll and pitch axes. Figure 5.21 shows that the decrease in yaw rate is, however, not faster than with the other axes, the overshoot in this case is therefore due to the inertia around the z-axis being larger.

The angle rates are given in Figure 5.21. The initial high-frequency oscillations of the measured rates are due to the instability before the adaptive parameters converge closer to their ideal values. After five seconds the reference and measured rates are virtually indistinguishable on the shown scale.

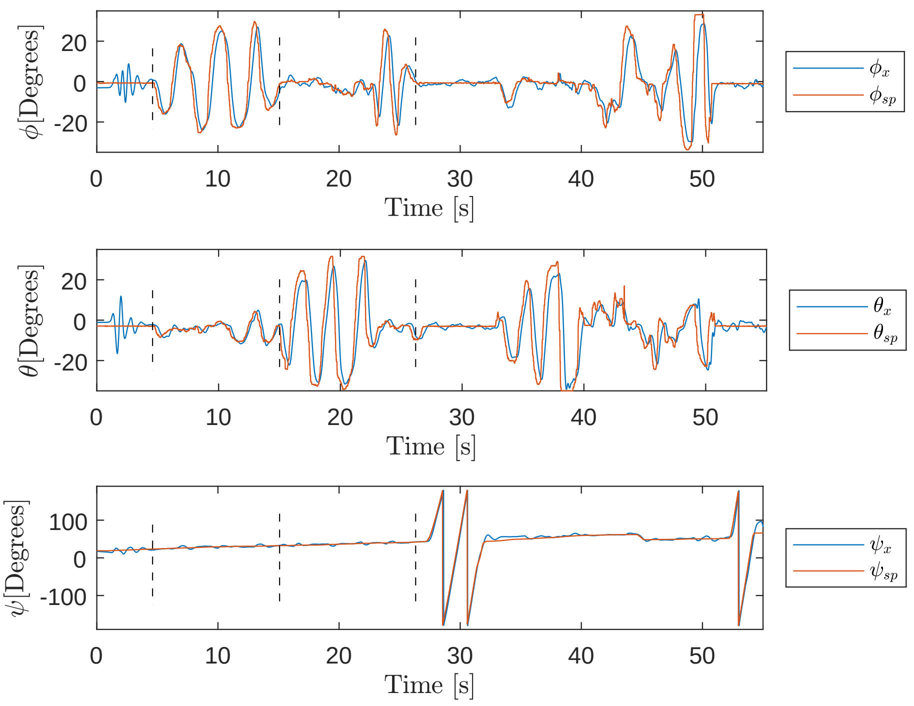


Figure 5.20: Practical angles for initial parameter convergence and flight test

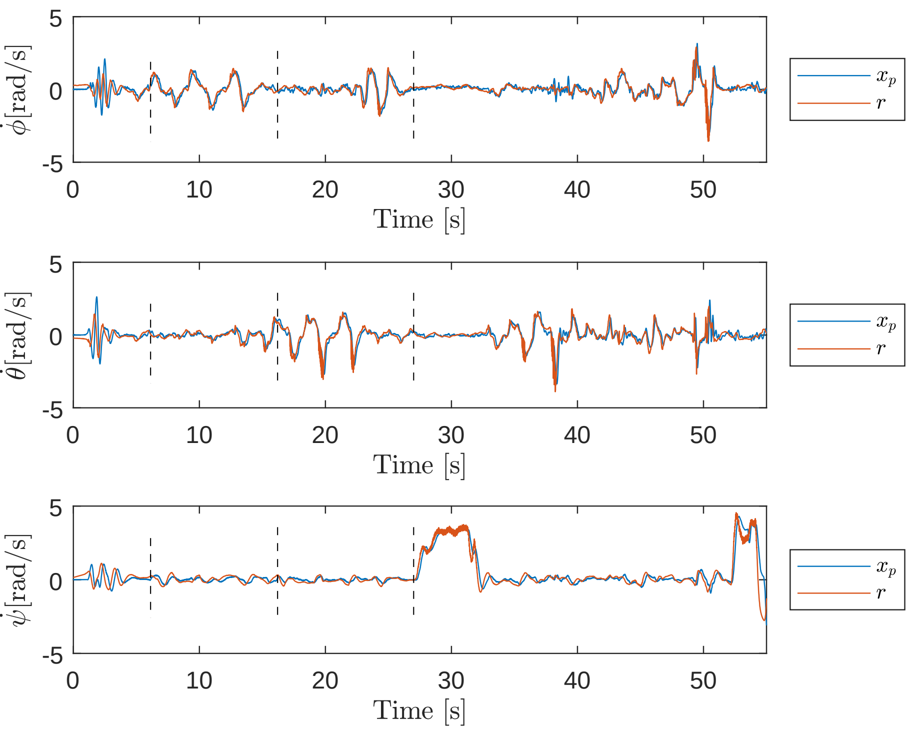


Figure 5.21: Practical angle rates for initial parameter convergence and flight test

The adaptive gains are given in Figures 5.22-5.25. Only the diagonal element of Θ_x and Θ_r are plotted, since as seen in the simulation section, these matrices are diagonally dominant. The adaptive gain values come close to their ideal values within 4 seconds. This explains why the angle response of the system also shows stable hover after approximately 4 seconds. The slight fluctuations in the Θ_x and Θ_r occur as the quadcopter is excited around that axis. For instance the gains associated with yaw show a slight spike in both of these matrices when the yaw command is given at 27 seconds. After 4 seconds the adaptive gains have not yet reached their exact ideal values. They are, however, close enough to their ideal values that the error between the adaptive reference model rates and the measured rates fall within the dead zone. Adaption is then switched off. Excitation around the axes allow the error to become larger than the specified dead zone and the parameters adapt further. This explains the slight increases of the adaptive parameters with time. The Θ_α values are larger than those experienced in simulation. This is due to a stronger link between axes. Motion about one axis has a larger effect on the other two axes in practice than in simulation. Reasons for this are: the quadcopter is not perfectly symmetric, the center of mass is not exactly in the middle, motor non-linearities and the motor thrust to PWM relationship.

The Θ_d parameters fluctuate around zero with no markable peaks, due to no direct disturbance. Contribution from the disturbance parameters can be seen in the two disturbance tests.

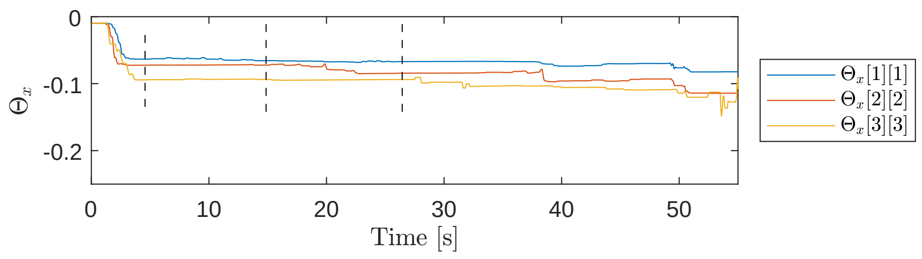


Figure 5.22: Practical initial parameter convergence and flight test, Θ_x

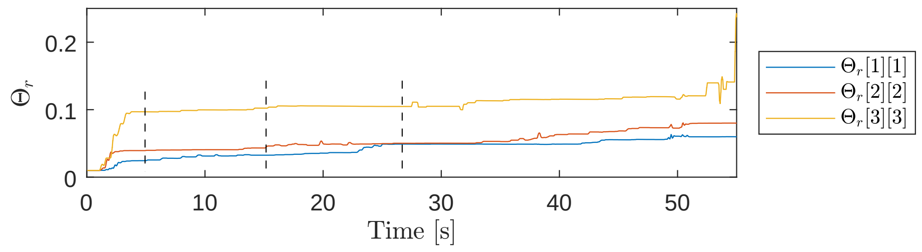


Figure 5.23: Practical initial parameter convergence and flight test, Θ_r

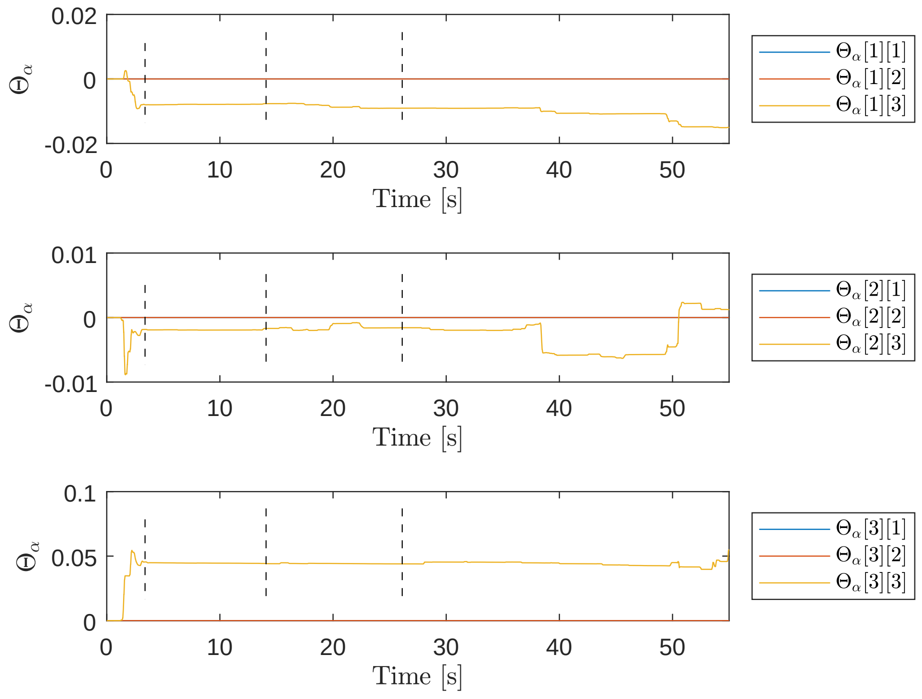


Figure 5.24: Practical initial parameter convergence and flight test, Θ_α

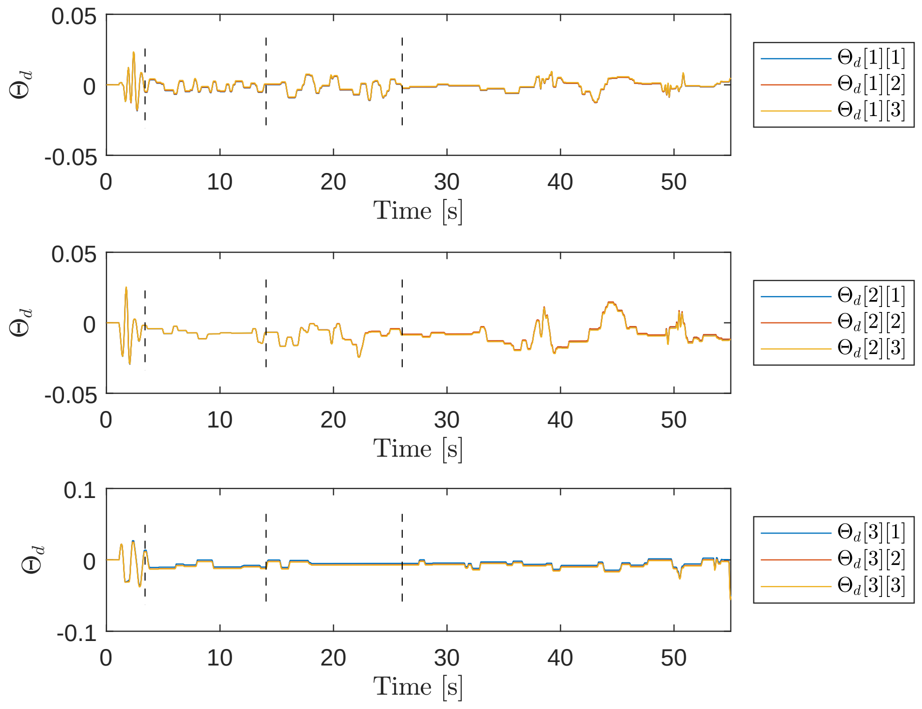


Figure 5.25: Practical initial parameter convergence and flight test, Θ_d

5.7.2 Step Response of 20 Degrees

The angle response of the system to a 20° step input around the roll axis is shown in Figure 5.26. As was the case in simulation, the PX4 controller yields the faster response. As mentioned, the response of the adaptive controller can be improved by using a faster reference model in the angle loop. Figure D.22 in Appendix D shows that a response similar to that of the PX4 controller can be achieved. The controllers with adaption on and adaption off have a similar initial response. At steady state the adaptive controller removes the steady state error. The reason for this error is described along with the hedging modification in section 5.2.1. The PX4 controller also has a small steady state error. When looking at the other axes it is seen that the pitch axis is relatively unaffected in the case of all three controllers. A slight steady state yaw error is, however, introduced in the case of the PX4 and adaption off controller. The results for the 20° step input test are summarised in Table 5.5.

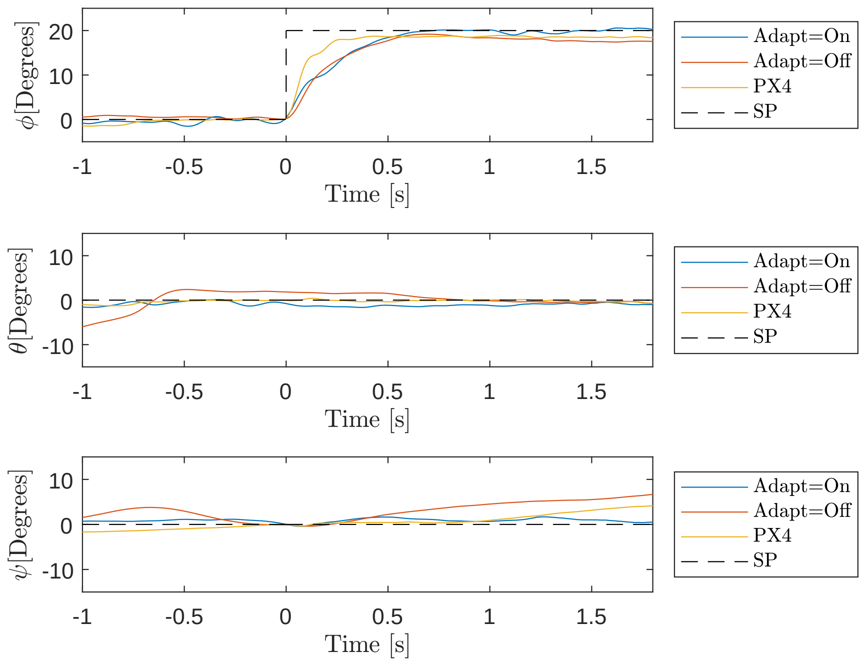


Figure 5.26: Combined practical angles for 20° step input

Additional plots for the roll step input of twenty degrees can be found in Appendix D. The angle rates of the system to a 20° step input around the roll axis is shown in Figure D.23. Again, the practical results are similar to the simulation results. The PX4 controller's fast response time requires fast rates and acceleration, which can lead to actuator saturation in the case of larger

Table 5.5: Practical response summary for 20° step input

Axis	Specification	PX4	Adaption On	Adaption Off
Roll	Settling time (90%)	0.23 s	0.5 s	0.54 s
	Overshoot	0°	0°	0°
	Steady state error	1.5°	0°	2°
	RMSE	2.64°	2.49°	2.58°

quadcopters. Only minor fluctuations occur in the adaptive matrices, since convergence of parameters have already occurred.

5.7.3 Internal Disturbance

The angle response of the system to an internal disturbance of 0.2 around the roll axis is shown in Figure 5.27. Again, the practical results are similar to those predicted by the simulation. The adaptive controller has a slightly larger deflection than the PX4 controller, but the time it takes the adaptive controller to return back to zero is less. With adaption switched off, the disturbance could not be rejected and a large steady state error occurs. The pitch axis is relatively unaffected, while the disturbance does introduce a slight steady state error in the case of the PX4 and adaption off controller. The results for the internal disturbance test are summarised in Table 5.6.

Table 5.6: Simulation response summary for 0.2 N.m disturbance

Axis	Specification	PX4	Adaption On	Adaption Off
Roll	Return to zero (±1°)	2.3 s	0.6 s	1 s
	Overshoot	10°	14°	25°
	Steady state error	0°	0°	22°
	RMSE	2.48°	2.07°	9.84°

Additional plots for the internal roll disturbance can be found in Appendix D. The angle rates of the system to an internal disturbance of 0.2 around the roll axis is shown in Figure D.28. The adaptive controller's roll rate has oscillations, as mentioned in the simulation section; this is a result of the fast adaption.

Similar to the simulation case, Figure D.29 shows that the gain associated with roll measurement ($\Theta_x[1][1]$) has a small increase in magnitude when the disturbance occurs. Figure D.30 also shows the same decrease in the gain associated with the roll set point ($\Theta_r[1][1]$). As in the simulation case, the values

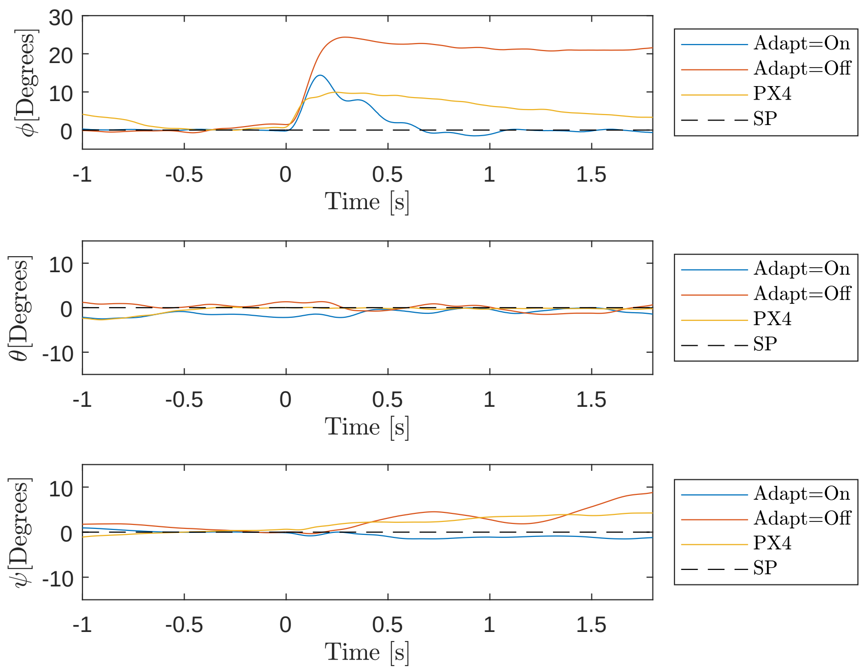


Figure 5.27: Combined practical angles for 0.2 N.m. disturbance

of the alpha matrix are still negligible when compared to the other matrices. They are, however, much more prevalent due to unmodelled non-linearities and uncertainties. Again, the majority of the disturbance is absorbed by the disturbance matrix, as is desired. This can be seen by multiplying the matrix by a unity vector as done in the control law. The roll element of the vector yields a value of close to 0.184, which is similar to the disturbance value given in the code. Although it is not as close as in simulation, it is still the majority of the disturbance.

5.7.4 Addition of Off-centre Weight

The angle response of the system to the addition of an off-centre weight is shown in Figure 5.28. The addition of the off-centre weight introduces larger fluctuations around the pitch and roll axes for the adaptive controller. The measured values, however, return back to zero more quickly. This faster return coupled with the slight overshoot in the opposite direction, due to the fast adaption, results in less of a positional drift. The yaw angle for the PX4 case also continued to gradually drift away from its set point value. The results for the off-centre weight test are summarised in Table 5.7.

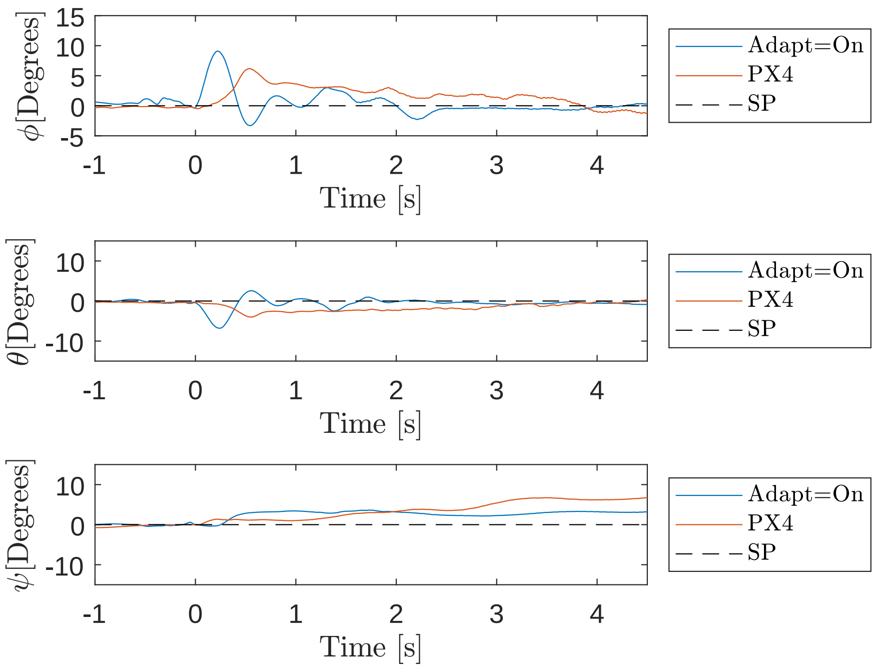


Figure 5.28: Combined practical angles for off-centre weight

Table 5.7: Simulation response summary for off-centre weight

Axis	Specification	PX4	Adaption On
Roll	Return to zero ($\pm 1°$)	∞	2.3 s
	Overshoot	6°	9°
	Steady state error	1°	0°
	RMSE	0.91°	0.65°
Pitch	Return to zero ($\pm 1°$)	3.1 s	1.7 s
	Overshoot	4°	7°
	Steady state error	0°	0°
	RMSE	1.1°	0.85°
Yaw	Return to zero ($\pm 1°$)	∞	∞
	Overshoot	7°	3°
	Steady state error	6°	3°
	RMSE	6.8°	0.51°

Additional plots for the off-centre weight test can be found in Appendix D. When looking at the angular rates in Figure D.33 it can be seen that the addition of the off-centre weight introduces a high frequency oscillation onto the measured rates. A vibration is therefore introduced into the system. This is true for all 3 axes with both controllers. The PX4 controller's, however, has a larger frequency and amplitude. Although not visible from the angular response, this vibration was also audibly noticeable when the practical test was done with the PX4 controller. The adaptive controller dampens the vibration better.

The first three adaptive matrices do not show any major adaption, meaning that the majority of the disturbance was compensated for by Θ_d. Multiplying the matrix by a unity vector as done in the control law yields a roll element of -0.15 in the vector and a pitch element of 0.88 in the vector. These values are the respective moment roll and moment pitch commands required to counter the actual moments of the off-centre weight.

5.7.5 Acrobatic Mode

As mentioned, one of the objectives of this project was that the new control law should integrate well with the other existing PX4 software. One way of testing this, was to see whether the different flight modes available with the PX4 software were still possible. The first mode tested was acrobatic mode. The angle control loop is, therefore, bypassed and the pilot directly provides the angle rates. Figures D.38-D.42, showing an acrobatic flight test, can be found in Appendix D.

5.7.6 Autonomous Mission

The second flight mode test to show successful integration is that of an autonomous mission. Flying an autonomous mission allows for the test to be replicated easily, not as was the case with the acrobatic case. The mission could therefore be done with two controllers for comparison. An autonomous mission also tests all the different control loops, making it the best test to prove proper integration. Figures D.43-D.53, showing an autonomous mission test, can be found in Appendix D. Note that the unchanged adaptive controller used for quadcopter B was again used for quadcopter A. The PX4 controller, however, required separately tuned parameters for each quadcopter.

Chapter 6

Conclusion

The flight controller used within the research group is the Pixhawk flight controller running PX4 firmware. It was therefore decided to use this flight controller for the control system design. It was also desired to keep the firmware outside of the attitude controller as untouched as possible and also that the changed controller can integrate well with the existing functions and modes. A adaptive controller was decided upon due to its capability to eliminate the need for tedious parameter tuning. This is especially true for a research group that make use of quadcopters of different size, and since research is done using these quadcopters, their model is often changed by the addition of hardware. Having one controller that can be used on all the different setups was therefore desired.

Different adaptive control approaches were researched. The model reference adaptive control approach followed by Achtelik *et al.* (2011) and the L1 adaptive control approach followed by Wang (2015) stood out due to the controllers having a similar cascade control structure as that of the PX4 controller.

Once the choice of control approach was decided upon, the controller needed to be designed and simulated. A model of the quadcopter was therefore required. The modelling and simulation was achieved with assistance of "quadsim" (Hartman *et al.*, 2014). The designed controller was evaluated against competing controllers, mainly the PX4 controller since this would be the controller the adaptive controller attempts to replace. Different tests were simulated in order to evaluate the controller. Once promising results were achieved in simulation the controller could be implemented practically.

Before moving on to test the controller practically, two addtional steps were, however, done. The PX4 software provides a method of simulating the PX4 code on the Pixhawk flight controller itself. This enables one to see whether the implementation of the controller in the software is correct. After suitable flight was achieved in the simulator the quadcopter was placed on a test rig to make sure the system was stable before moving on to practical flight testing. Testing on the test rig was limited to tests with adaption switched off. Practical take-off was then done with adaption switched off, with initial estimates for the ideal parameters. Once suitable flight was achieved with adaption switched off, adaption was switched on in flight. Suitable adaptive rates and other parameters were obtained empirically. After this final testing and controller comparison could be done.

Tests were done to evaluate the tracking and response time of the controller. Although the response proved to be slower to step inputs, it could be modified to give a similar response. It was, however, decided against this in order to allow the same code to run on larger, slower quadcopters. The controller was evaluated against itself with adaption switched off and against the PX4 controller. Tests were also done to test disturbance rejection. When comparing the adaptive controller to that of the tuned PX4 controller, during simple flight, the improvement in tracking is not significant. It is, however, important to note that the PX4 controller required tuning whereas the adaptive controller reached the gains itself. A clearer sign of improvement from the PX4 controller to the Adaptive controller could be seen with disturbances. Although the adaptive controller had a slightly larger peak deviation due to the disturbance, it returned back to the initial value faster. The extra time it took the PX4 controller to return to its initial value caused significant drift in position. In some cases a disturbance even introduced a steady state error in the case of the PX4 controller. Disturbances also introduced larger vibrations in the PX4 controller's case.

It is noteworthy that the disturbances considered in this project are all instantaneous step inputs. An oscillatory disturbance such as wind is therefore not considered. High frequency disturbances can cause problems with the adaptive controller. For instance it was found that in windy conditions, where the wind direction was inconsistent and where gusts occurred, the controller became unstable. This is due to the oscillatory excitation causing excessive growth of the disturbance term. The adaption rate for the disturbance term should therefore be limited if inconsistent wind gusts are present.

In the case that the quadcopter model remains unchanged and parameter tuning is only required once, and no disturbances are expected, an adaptive controller is not necessary due to no significant improvement in this case. However, the advantage of the adaptive controller comes in when additional hardware, such as cameras and larger batteries, are attached to the quadcopter.

With the larger quadcopter, the adaptive controller shows a clearer improvement in tracking as can be seen in the case of the autonomous mission. This is due to the uncertainties being greater in the case of larger quadcopters. The effect of unbalanced propellers and deflection of the arms become larger. This is despite the PX4 controller being tuned specifically for that quadcopter.

6.1 Recommendations

In the practical case one always runs the risk of adaptive parameters becoming too large and affecting the stability of the flight. The parameters can be limited to avoid instability, this will, however, still not result in ideal flight. In some cases the basic PX4 controller might be preferred. If growth of the adaptive parameters is a concern, adaption can be switched off after convergence of parameters. From testing it was seen that this led to large steady state errors, especially with disturbances. An additional integrator term can, however, be added to improve upon this. The rate loop then becomes similar to that of the PX4 controller with the proportional term first converging to an ideal value. It also has two proportional terms, one for measured values and one for reference values instead of one solely for the error. This allows for an additional damping effect since the two proportional terms can vary with respect to each other.

6.2 Future Work

The MIT rule controller was only designed in this project and not evaluated against the Lyapunov controller in simulation or practice. Implementation of the MIT rule controller can possibly yield interesting results. Secondly, a more modern approach is the L1 adaptive controller. This control approach is based on MRAC and essentially adds a low pass filter. Design and implementation of this controller can possibly show improvement on the basic MRAC.

It is also desirable to implement the designed controllers on even more quadcopters to see how it handles the change in model. Additional tests considering high frequency disturbances can also be investigated.

It was found that the improvement in the attitude controller yielded improvement in lateral position control as well. Height control, however, remains unchanged. Take-off and landing in position mode is also currently an issue with the PX4 controller. It is suspected that this issue is present since the quadcopter altitude is measured with a barometric sensor. The quadcopter does therefore not exactly know when it is in contact with the ground. Adding a sonar sensor to measure vertical distance can possibly improve on this issue.

Appendices

Appendix A

Hardware

The hardware of quadcopter A is discussed first follow by a discussion of quadcopter B. Other hardware includes a torque stand and a test rig.

A.1 Quadcopter A

A 500 mm glass-fibre frame is used to house all the other components. The 500 mm refers to the diagonal distance between opposite motors. The frame has a built-in power distribution board which supplies power to the different components. PWM values are sent to the ESCs by the flight controller. The ESCs then powers the 3-phase brushless motors according to this PWM signal. The ESCs used are Hobbywing Platinum 30A. The motors used are Emax 3506 650kV. The 35 refers to the stator diameter in mm and the 06 to the stator height in mm. The kV value has units of RPM/V and indicates the no load rotational speed of the motor per volt applied. 12 inch carbon-fibre propellers with a 38mm pitch are spun by the motors. A 4 cell battery provides power to the system. Telemetry is used in order to monitor parameters during flight. The last important component to mention is the GPS which is used in position mode.

A.2 Quadcopter B

A smaller 250 mm carbon-fibre frame is used to house all the other components. Again, the 250 mm refers to the diagonal distance between opposite motors. A separate power distribution board is required to supply power to the different components. Smaller motors have a lower amp draw. 12A BLheli ESCs are therefore sufficient. The motors used are Emax 1806 2280kV. Similarly, the 18 refers to the stator diameter in mm and the 06 to the stator height in mm. 5 inch nylon propellers with a 30mm pitch are spun by the motors. A 3 cell battery provides power to the system. Quadcopter B does not have the additional telemetry or a GPS due to size restrictions.

A.3 Torque Test Stand

The torque test stand used for motor parameter identificaiotn is depicted in figure A.1.

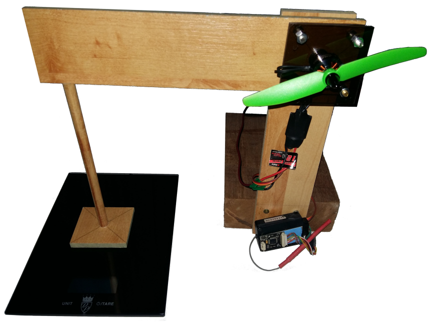
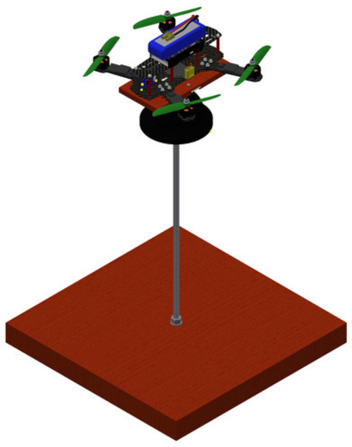


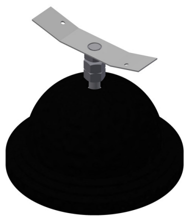
Figure A.1: Torque test stand

A.4 Test Rig

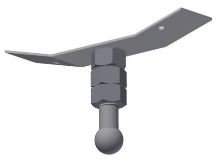
The test rig used for initial testing is shown in figure A.2. A mount was also made for s500 quadcopter.



(a) Full view



(b) Pivot mechanism assembly



(c) Pivot mechanism axle

Figure A.2: Test rig

Appendix B

Simulation

B.1 MATLAB

As mentioned, the modelling of the system is achieved by means of a Simulink project created by Hartman *et al.* (2014). The three main blocks along with some modifications are discussed below. The modifications are all applied on the controller and control mixing blocks. The inputs and outputs of the blocks remain unchanged. The attitude set points along with all the states enter the controller block. Three moment commands are then determined by the control system. These are then sent to the control mixing block. The control mixing block determines the motor commands, which is sent to the quadcopter dynamics block. The quadcopter dynamics block determines hpw the quadcopter will respond to certain motor commands.

B.1.1 Modified Control Block

As mentioned the attitude set points along with all the states enter the controller block. Figure B.1 shows that the attitude controller block can be broken up into two main sections: the angle loop and the angle rate loop. All the states and angle set points are used by the angle loop to generate angle rates as discussed in Chapter 5. Note that the yaw angle is not commanded to the angle loop, but a yaw rate. The commanded yaw rate is superimposed on the yaw rate determined by the blue angle block. The commanded yaw rate is also integrated to serve as a set point in the angle control.

The rate loop takes the rates determined by the angle loop, along with the system rates, and determines three moment commands. The rate block also outputs the change in adaptive matrices and the reference model. These changes are integrated in order to update the values in the rate block. Each moment commands passes through a saturation block to ensure that the requested command remains between -1 and 1.

The height is also controlled by means of a PID controller. For the PX4 controller case, manual mode simply passes the throttle value to the motors. In the case of other modes the original PX4 altitude controller is used. The altitude control block therefore falls outside of the scope of this project.

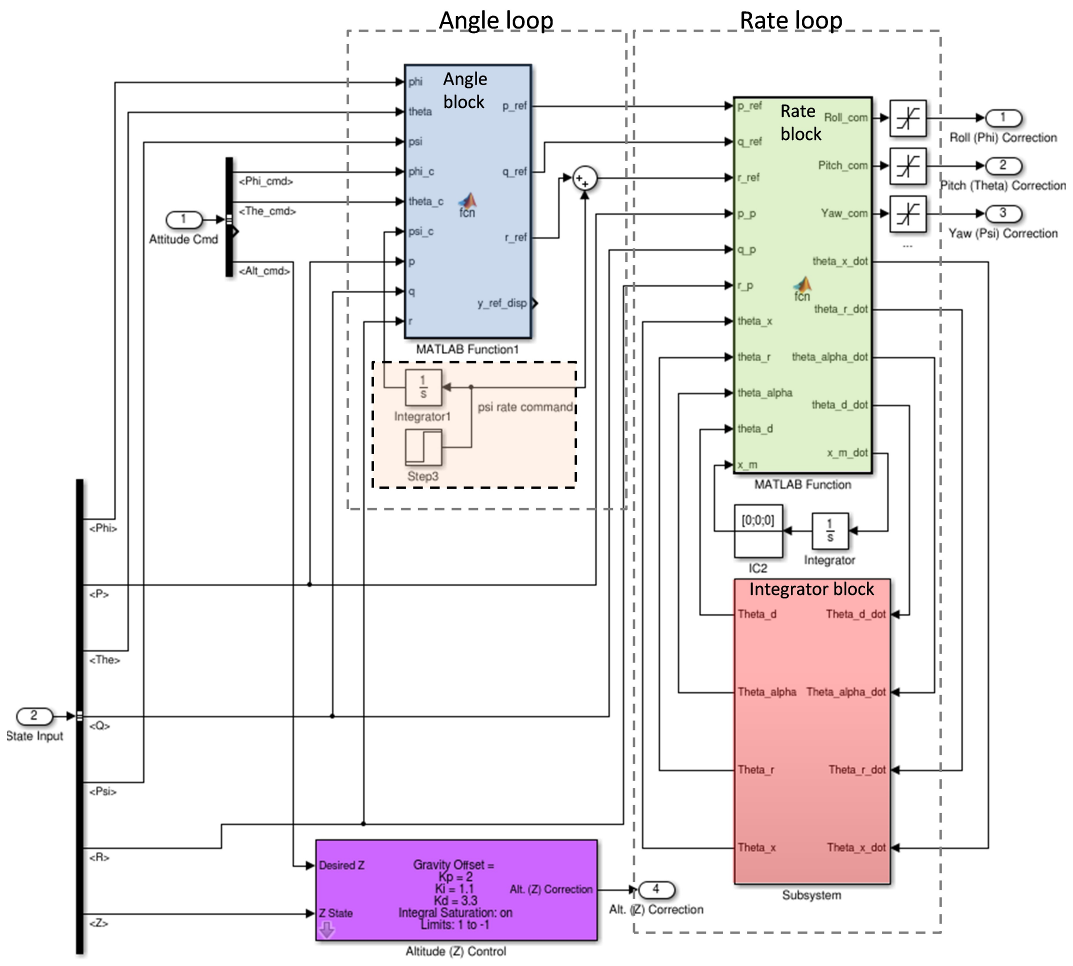


Figure B.1: Quadsim modified attitude control block

B.1.2 Quadcopter Control Mixing with Modifications

Figure B.2 shows how the three moments determined by the controller block are mixed together with the altitude command to obtain the motor outputs. A switch decides whether they are mixed according to a plus or cross configuration. The PX4 mixer adds an additional $\cos 45°$ factor, in the case of pitch and roll, when the cross configuration is used. This accounts for the change in lever arm due to the change in axes. The motor values in the simulation are given as a value between 0 - 100. A value of 50 is seen as hover giving the correction factors space to result in values between -50 and 50 at most. The

PX4 motor outputs, before scaling and adding throttle, range between -1 and 1. A scaling factor of 50 is therefore required so that the controller gains in the MATLAB simulation are equivalent to the controller gains implemented in the Pixhwak code. Further details on how the two separate configuration blocks works can be found in C.4.

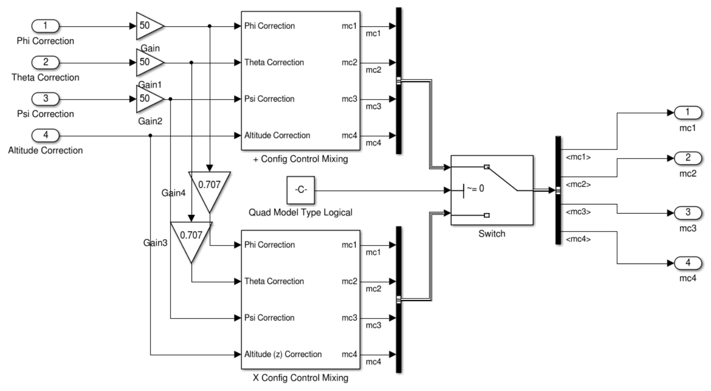


Figure B.2: Quadsim mixer

B.1.3 Quadcopter Dynamics

The quadcopter dynamics block, shown in figure B.3, is the model block of the control loop. The block determines how the quadcopter will respond to different motor commands. The motor dynamics block, shown in figure B.4,

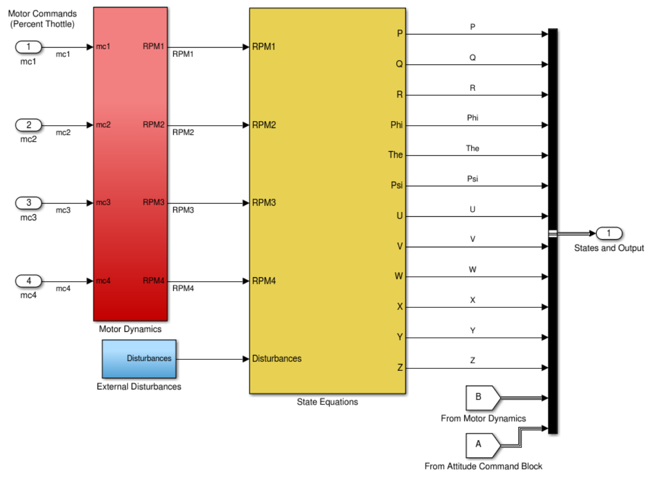


Figure B.3: Quadsim quadcopter dynamics

determines at which speed the motor will rotate given a certain motor command. The motor command values are first limited to remain within the given range (0-100). The next block represents throttle cut-off. Throttle cut-off is due to the fact that the motor does not start spinning until a minimum throttle point is reached. This introduces a jump discontinuity. The last block of the motor dynamics take into account that the steady state motor rotation speed is achieved instantly. This block simulates how the motor accelerates to its steady state value. The motor rotation speed then enter the state equation

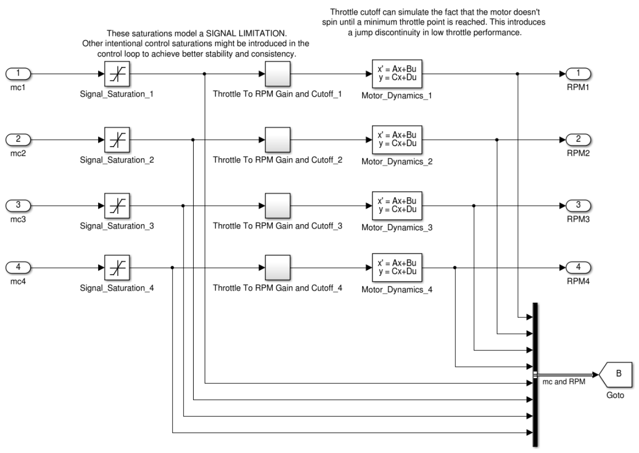

Throttle cutoff can simulate the fact that the motor doesn't spin until a minimum throttle point is reached. This introduces a jump discontinuity in low throttle performance.
These saturations model a SIGNAL LIMITATION. Other intentional control saturations might be introduced in the control loop to achieve better stability and consistency.
1
mc1
Signal_Saturation_1
Throttle To RPM Gain and Cutoff_1
x' = Ax+Bu
y = Cx+Du
Motor_Dynamics_1
1
RPM1
2
mc2
Signal_Saturation_2
Throttle To RPM Gain and Cutoff_2
Motor_Dynamics_2
2
RPM2
3
mc3
Signal_Saturation_3
Throttle To RPM Gain and Cutoff_3
Motor_Dynamics_3
3
RPM3
4
mc4
Signal_Saturation_4
Throttle To RPM Gain and Cutoff_4
Motor_Dynamics_4
4
RPM4
mc and RPM
B
Goto

Figure B.4: Quadsim motor dynamics

block. How this block determines how the quadcopter behaves according to motor rotational speeds is discussed in the next section.

B.2 Model Parameters

Different parameters unique to the quadcopter are required for simulation. These parameters include: c_r, b, min throttle, τ, c_t, c_q, inertias and weight. These values were obtained for quadcopter A in a previous project (Fourie, 2016). This was not the case for quadcopter B. Each of these parameters and how they were obtained, for quadcopter B, are discussed below. A summary of the final parameter values are given at the end of the section. Motor testing was done by (Rotorbench, n.d.) and their data are used in this section.

Throttle to rpm relationship

The first relationship required is that between throttle percentage and rpm. Figure B.5 shows practical test results relating measure RPM values to the commanded PWM value. The relationship required for the Simulink simulation is of RPM to throttle percentage and not PWM. The PWM values range from 1000 to 2000 and is linearly related to the throttle value, resulting in equation (B.1).

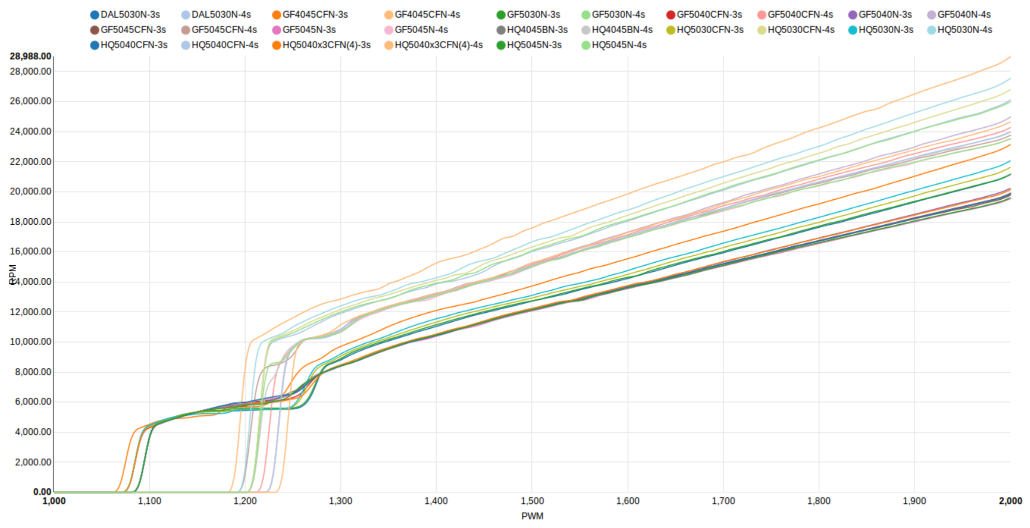


Figure B.5: RPM generated for different PWM values (Rotorbench, n.d.)

$$Throttle\% = \frac{PWM - 1000}{10} \tag{B.1}$$

Figure B.5 and equation B.1 is used to relate the RPM to the thrust percentage. This relation is a straight line with gradient, Cr, and y intercept, b.

<u>Motor time constant</u>

The RPM achieved at different PWM commands shown in figure B.5 are at steady state. The RPM was, however, not instantaneously achieved. How the motor-propeller combination accelerates therefore needs to be taken into account This is done by an time constant which is defined as the time it takes the RPM to reach 63 percent of its steady state value. Measurements of how the RPM increases to a step in throttle command is therefore required. Such measurements for the Emax mt1806 were not available at (Rotorbench, n.d.). Two options were available, build a testing device or estimate the time constant parameter. Due to the fact that large variations in the time constant has little effect on the response of the system and that accurate simulation is not the main focus of the project, it was decided to estimate the value. The typical value obtained by the creators of the simulation program is therefore used. The default value can be increased by 30% before slight oscillations are superimposed on the signal. Decreasing the time constant by 30% makes the response slightly smoother. Figure B.6 shows the effect of the change in time constants by showing the response to a 20 degree setpoint. Miniquad test bench have step input data for multiple similar motors (Harrell, 2015). Figure B.7 shows the response of different motors to a step in throttle input. The

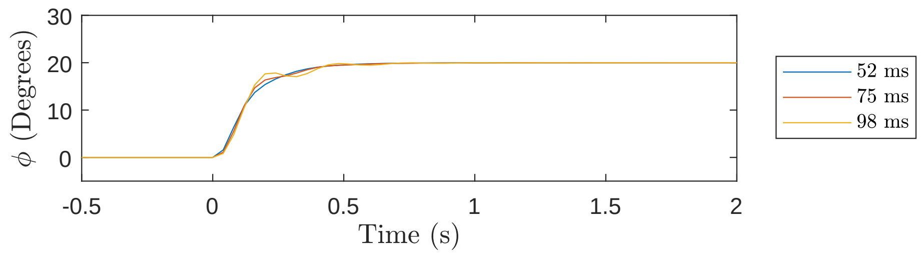


Figure B.6: Effect of different time constants on 20 degrees roll step input

throttle is stepped up from an idle throttle corresponding to 5000rpm to 50 % throttle. The time the motor takes to reach 63.2 % of the total rpm step is taken as the time constant. All these motors result in similar time constants. From this it can be seen that using a time constant of 7.5 ms is conservative.

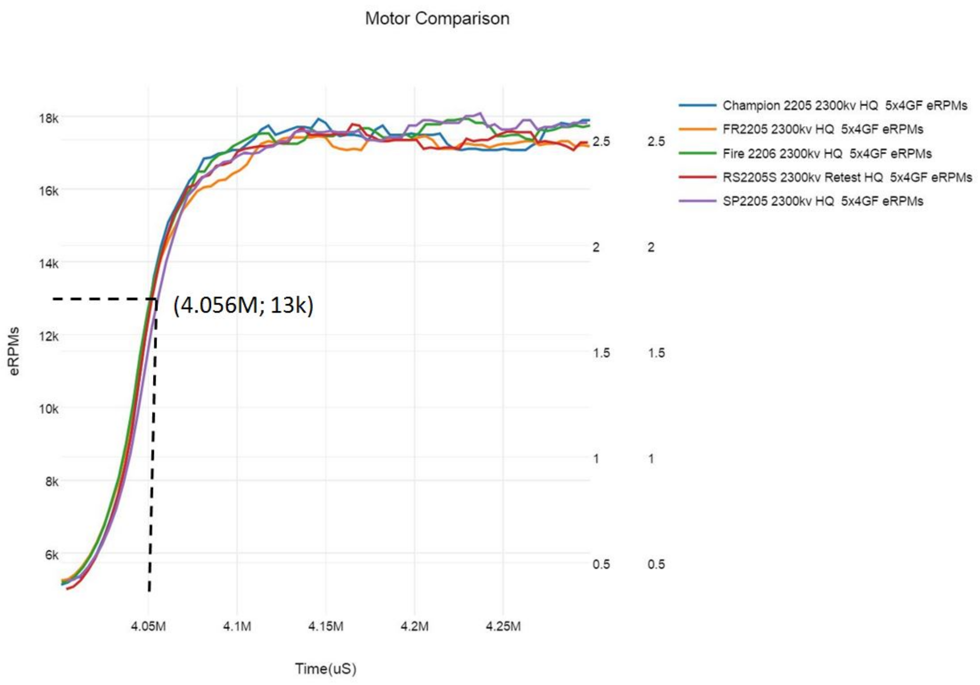


Figure B.7: Throttle step input comparison (Harrell, 2015)

<u>RPM to thrust relationship</u>

A parameter relating the RPM to the thrust generated by a propeller at steady state is required next. Figure B.8 shows this relationship. The relationship is assumed to be quadratic and is approximated by equation B.2.

$$C_t = \frac{F_y}{RPM^2} \tag{B.2}$$

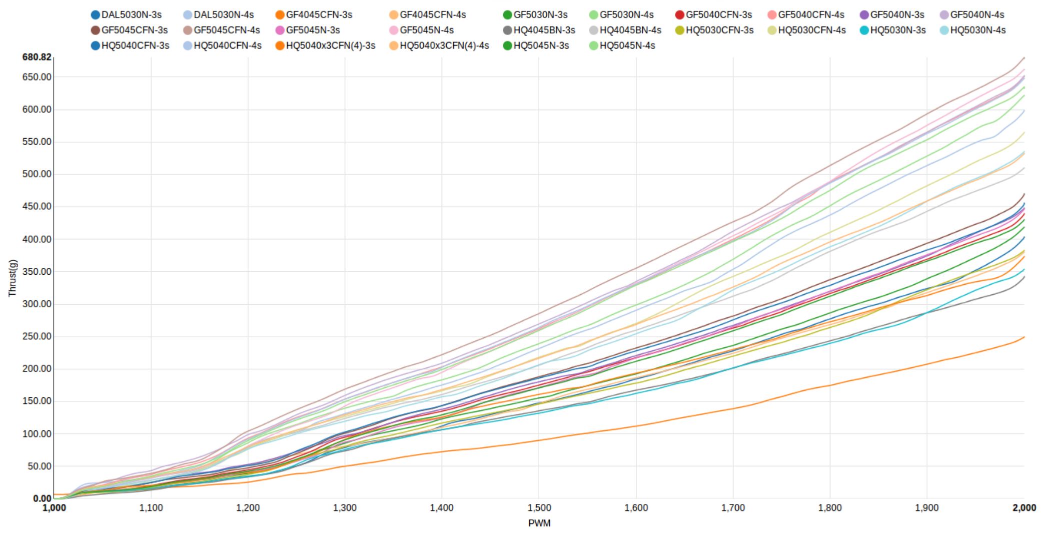


Figure B.8: Thrust generated for different PWM values (Rotorbench, n.d.)

This parameters varies slightly at different RPM's. It is therefore calculated at 50% throttle, 1500 PWM, in order to obtain a mean value.

RPM to torque relationship

Similar to the RPM to thrust relationship, a RPM to torque relationship is required. This torque refers to the reaction torque of the motors which is responsible for yaw. Measurements to obtain this relationship were not available for the Emax Mt1806 motors. The relationship is, however, also quadratic as the RPM to thrust relationship, see equation B.3 .

$$C_t = \frac{T}{RPM^2} \tag{B.3}$$

Obtaining a relationship between the torque and thrust at a given RPM, thrust command, will enable calculation c_t at the same throttle command value of 50%. In order to obtain the torque to thrust relationship, torque and thrust measurements would be required at the same throttle command. In order to measure the torque generated by the motor a test rig, similar to that proposed by Hartman *et al.* (2014), was build. Figure A.2 shows the build test rig. The thrust measurement was simply taken by directly attaching a motor to a plate. The relationship was then used to scale the c_t value.

Inertias and mass

Hartman *et al.* (2014) provides an easy method to find the inertias of the quadcopter around all three axes. The user simply has to make simple length and weight measurements and then input these measurements into the GUI interface. The net weight and inertia values are then approximated. The values, as entered into the Simulink GUI are given in table B.1. For more information on what these values refer to, consult Hartman *et al.* (2014).

Table B.1: Summary of measurements to create model in Quadsim

Unit	Symbol	Description	Quad A	Quad B	Unit
Motors	m	Mass of one motor	73	18	g
	dm	Distance: center to motor	22.25	13	cm
	h	Height of motor	3.175	2.67	cm
	r	Radius of motor	1.403	1.15	cm
ESCs	m	mass of one ESC	30	11	g
	a	Width of ESC	2.54	2	cm
	b	Length of ESC	5.715	4.2	cm
	ds	COM of ESC to center	6.255	9	cm
Central Hub	m	mass of other	800	289	g
	r	radius of hub	5.6	4	cm
	H	Height of Hub	13	8	cm
Arms	m	Mass of 1 arm	45	10	g
	r	Radius arm	3.2	1.5	cm
	L	Length	18.6	9	cm
	da	Attachment to hub center	5	2	cm

Parameter summary

Table B.2: Propeller and motor parameter calculation values, quad B

PWM	Throttle (%)	Thrust (N)	Torque (N.m)
1500	50	1.528	0.02715

Table B.3: Propeller and motor parameter summary

Property	Ct ($\frac{N}{rpm^2}$)	Cq ($\frac{Nm}{rpm^2}$)	Cr (rpm)	b (rpm)	τ	min throttle
Quad A	2.2857E-07	3.184E-8	175	3612	0.075	8%
Quad B	9.4217E-09	1.673E-10	45	1200	0.075	15%

Table B.4: Inertia summary

Property	$I_{xx}(kg/m^2)$	$I_{xx}(kg/m^2)$	$I_{xx}(kg/m^2)$
Quad A	0.010664	0.010664	0.018787
Quad B	0.0011685	0.0011685	0.0019945

Appendix C

PX4 Firmware

C.1 Architectural Overview

The PX4 code can either be developed on Linux or Mac OS. A comprehensive overview of the toolchain installation in order to develop the code can be found at PX4 (2016*a*). Once the toolchain is installed the code can either be built in the console or in a graphical user interface. Qt Creator was used to compile the firmware for this project. If a full installation is done the firmware can also be simulated on the computer (PX4, 2017*c*).

In order to modify the PX4 firmware for the purpose of the project a better overview of the firmware is required. PX4 can be broken down into two main layers (PX4, 2017*a*):

- PX4 middleware
- PX4 flight control

The middleware consists of the tools, drivers and libraries that are required for the flight control layer (Dronesmith Technologies, 2016). The PX4 flight control layer is a collection of algorithms, one of these algorithms is the control algorithm which is of importance in this project. The middleware layer remains unchanged for the purpose of this project.

Each of the blocks in figure C.1 represents a module in the flight control firmware. Communication between these modules are achieved by means of publish/subscribe calls. The attitude block is the block which requires modifications in order to implement the adaptive controller. This chapter will examine the existing attitude control module and also the modified attitude module. A brief overview of some of the modules upstream of the attitude control module and also downstream of the attitude control module are required. A “rcS" script is the first file that is executed at start up and calls all

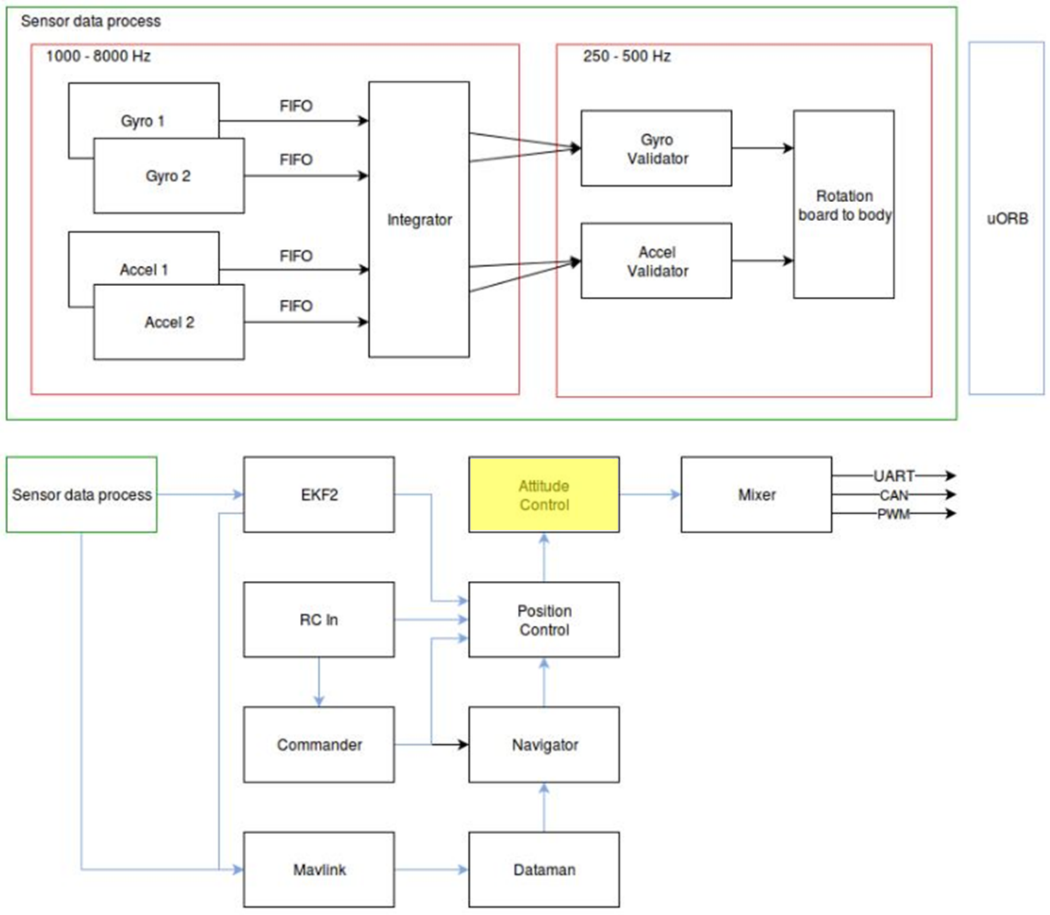


Figure C.1: Software architecture (PX4, 2017*a*)

other scripts. The configuration selection will determine which scripts need to be started.

C.2 Pre-attitude Control

The raw sensor values go to estimators where the control states used by the control loops are determined. As mentioned in Chapter 2 different flight modes exist. The flight modes define states for the system which determines which sections of the modules need to be executed (PX4, 2017*b*). The PWM values of the remote control are therefore converted to either position rate commands, angle commands or angular rate commands dependent on the flight mode selected. The commands together with the measured states enter the position controller. Depending on the flight mode selected, new angle commands are calculated or the commander values are simply passed to the attitude control loop.

C.3 Attitude Control

The module to be modified is discussed next. The original attitude control module is discussed first, followed by an overview of the modifications applied to the original attitude control module.

C.3.1 Existing Attitude Control

Figure C.2 shows the attitude control module breakdown. Once the script is started a object is constructed from the "MulticopterAttitudeControl" class. The control task is then started. A trampoline function is used as a shim for calling the task_main from the task_create. The main attitude control task can now execute and is represented by the dotted red lines.

The application starts by subscribing to all the structures that will be required and updating all the necessary control parameters. A array is created to check when a poll, signalling that new data is available, occurs. A while loop start and executes until attitude control is no longer required. The loop starts by checking whether new data is available and waits until it is available, up until 100 ms. Once new data is available an if statement is entered to start the attitude control. The time since the previous execution is determined for use in the control loop. After fetching the available data, the measured time is limited to remain within a range, 2 ms-20 ms. The flight mode is then checked. If the selected mode is that of r-attitude and the threshold on pitch or roll is exceeded then a flag is set to false to avoid running the attitude angle control loop. This flag is already set to false in the commander module if acrobatic mode is selected.

If the above flag is true then the attitude angle control is called. The rates determined from this funciton are then published as setpoints. This function is done according to the angle subsection in section 2.3.2. If the flag is set false then skip the attitude funciton and set the rate setpoints directly from commander file (RC commands). If the flag specifying whether rate control is needed is set then the rate controller is called. This will calculate the moment commands which are then published along with the throttle command which are used by the mixer. This is done according to the rate section in Chapter 2.

C.3.2 Modified Attitude Control

An additional wait is added to the wait for measurements wait. A wait of 6ms is used to allow for a more constant time sample. This is, however, not neccessary. The main modifications lie within the control attitude block and the control attitude rate blocks. Other modificaitons include the publication of adaptive gains after the control attitude rates block.

C.4 Post-attitude Control

The three moments determined above are mixed together with the thrust command to obtain the motor commands. The moment commands vary between -1 and 1, while the thrust command varies between 0 and 1. The initial motor commands vary between 0 and one before the final motor commands are offset to vary between -1 and 1. These values are scaled to PWM values.

The get_control function returns the respective moment commands for roll, pitch and yaw, and also the thrust command. These values are multiplied by scaling values, which is unity by default, before being limited to lie between -1 and 1. The mixing will depend on the configuration chosen. A table containing information, dependent on the geometry (number of rotors and axis configuration) being used, is obtained from a multi_tables file. Essentially scaling vectors are obtained from this table. The scaling vectors are used to correctly mix the moment commands according to the selected configuration. The scaling vectors for the plus and cross configuration are obtained as follows. A configuration vector, dependent on whether the plus or cross configuration, is used and is given by equation C.1.

$$\mathrm{CF} = \begin{bmatrix} 90^\circ \\ -90^\circ \\ 0^\circ \\ 180^\circ \end{bmatrix}_{+} \text{ or } \begin{bmatrix} 45^\circ \\ -135^\circ \\ -45^\circ \\ 135^\circ \end{bmatrix}_{\times} \tag{C.1}$$

This accounts for the change in lever arm due to the change in axes.The scaling vector for each moment command is then determined according to equations C.2-C.4.

$$\mathrm{SF}_\phi = \cos(\mathrm{CF} + 90^\circ i) \tag{C.2}$$

$$\mathrm{SF}_\theta = \cos(\mathrm{CF}) \tag{C.3}$$

$$\mathrm{SF}_\psi = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix} \tag{C.4}$$

As evident from these scaling vector, the PX4 mixer adds an additional cos45 factor in the case of pitch and roll when the cross configuration is used.

The combined mixing can be done illustrated by

$$mc_{1-4} = u_1 \text{ SF}_{\phi,1-4} + u_2 \text{ SF}_{\theta,1-4} + u_3 \text{ SF}_{\psi,1-4} + \text{Thrust} \tag{C.5}$$

where i is a vector of ones.

The PX4 firmware, however, does the mixing in stages to account for motor saturation. Figure C.3 shows a breakdown of the mixing procedure. The initial mixing is done without the yaw component after which the smallest and largest motor commands are determined. If the minimum motor value is saturated and the difference between the maximum and minimum is below unity a thrust boost value is calculated in order to attempt to remove this saturation. If the required boost value is too large a roll-pitch scaling factor is calculated. A similar approach is followed if the largest motor value saturates, except now a value to decrease the thrust is calculated. In the case of saturation where the difference between the maximum and minimum is above unity, a boost value and a roll- pitch scaling factor is calculated. The mixing process is then repeated with the yaw component and correction components. The motor values are again checked for saturation and the yaw value and thrust values are adjusted accordingly. Final mixing then takes place after these corrections. The final motor commands are determined according to equation C.6.

$$mc_{1-4} = \text{idle\_speed} + mc_{1-4}(1 - \text{idle\_speed}) \tag{C.6}$$

This ensure that the final motor command is a value between -1 and 1. These values are sent to the pwm_out_sim.cpp module which calculates the PWM values to send to the ESCs according to equation C.7.

$$mc_{1-4} = 1500 + 500mc_{1-4} \tag{C.7}$$

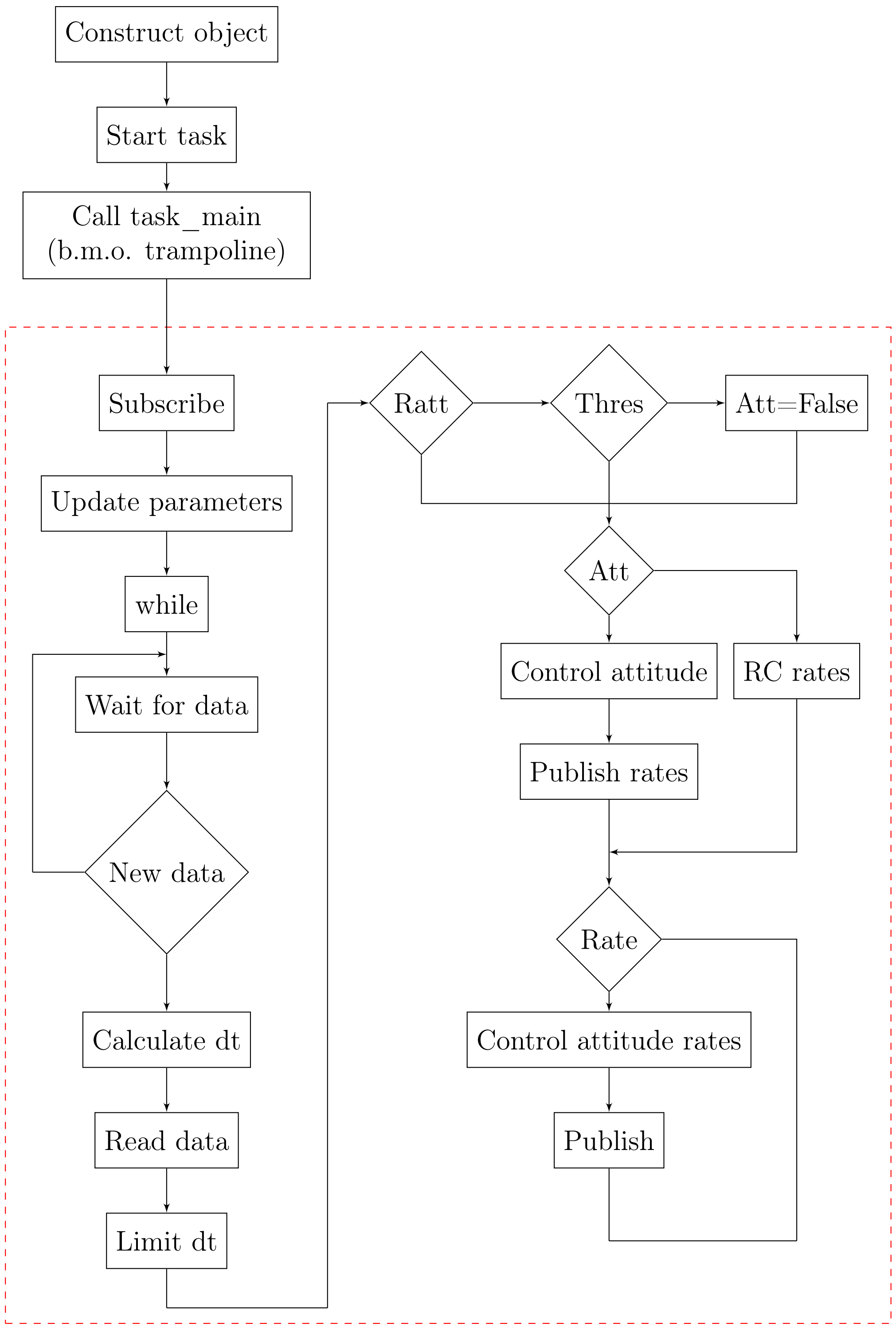


Figure C.2: Attitude control module breakdown

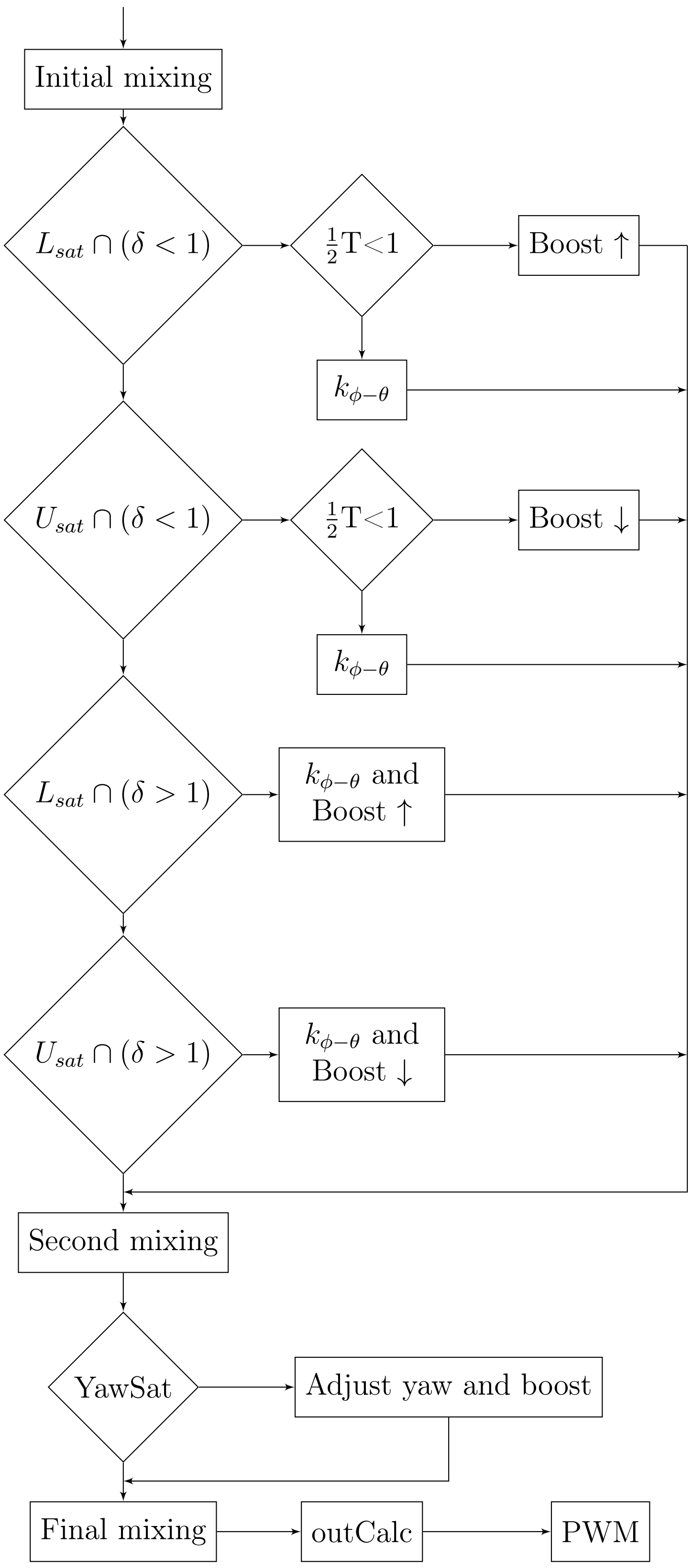

Initial mixing
$L_{sat} \cap (\delta < 1)$
$\frac{1}{2}$T<1
Boost ↑
$k_{\phi-\theta}$
$U_{sat} \cap (\delta < 1)$
$\frac{1}{2}$T<1
Boost ↓
$k_{\phi-\theta}$
$L_{sat} \cap (\delta > 1)$
$k_{\phi-\theta}$ and Boost ↑
$U_{sat} \cap (\delta > 1)$
$k_{\phi-\theta}$ and Boost ↓
Second mixing
YawSat
Adjust yaw and boost
Final mixing
outCalc
PWM

Figure C.3: Mixer module breakdown

Appendix D

Additional results

D.1 Simulation

D.1.1 20 Degree Roll Step Input

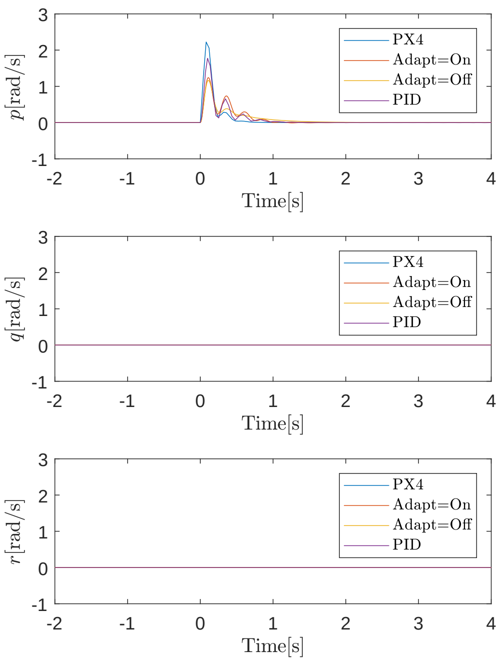


Figure D.1: Combined simulation angle rates for 20° step input

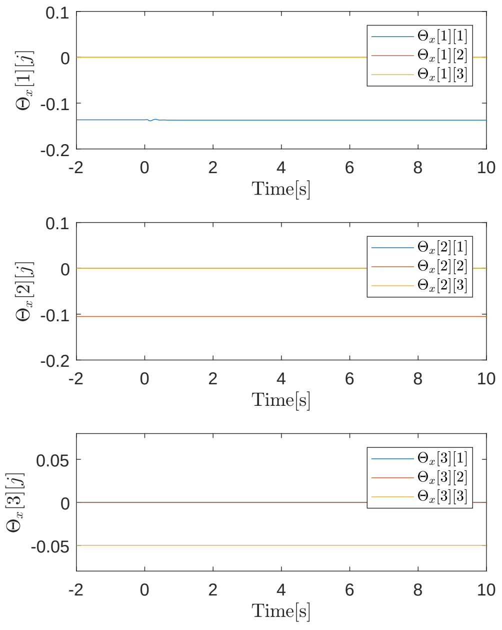


Figure D.2: Simulation 20° step input, Θ_x

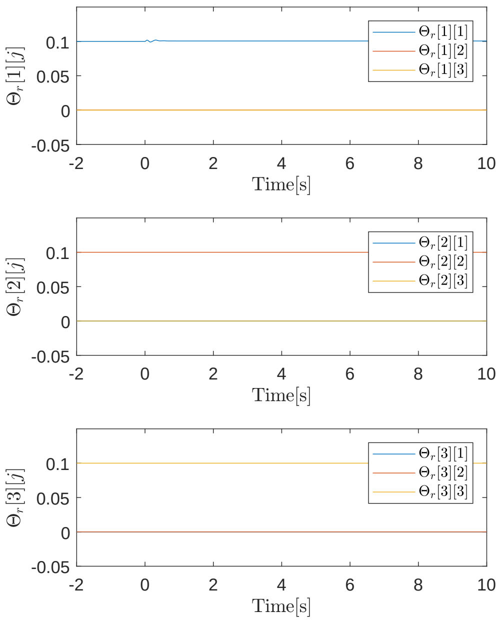


Figure D.3: Simulation 20° step input, Θ_r

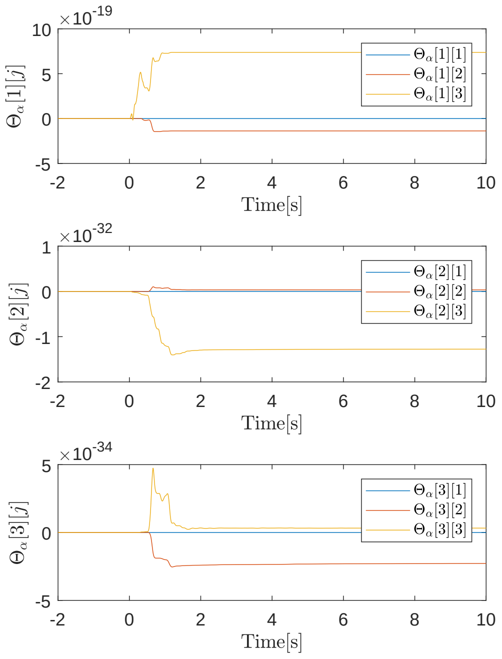


Figure D.4: Simulation 20° step input,, Θ_α

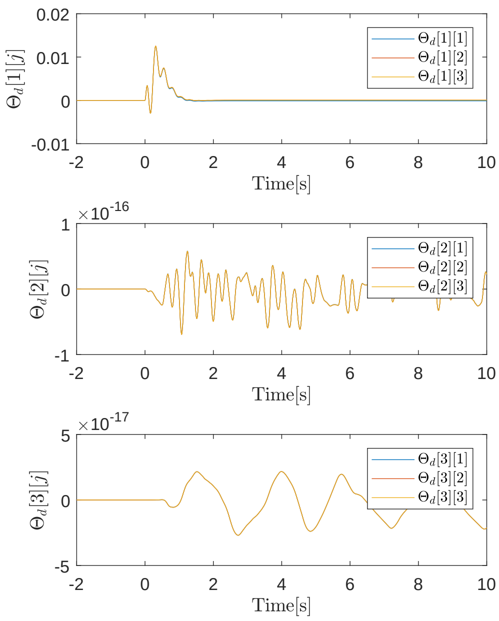


Figure D.5: Simulation 20° step input,, Θ_d

D.1.2 Internal Disturbance

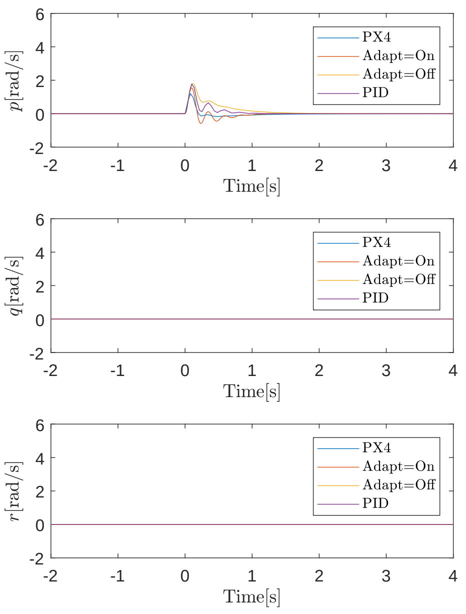


Figure D.6: Combined simulation angle rates for 0.2 N.m. disturbance

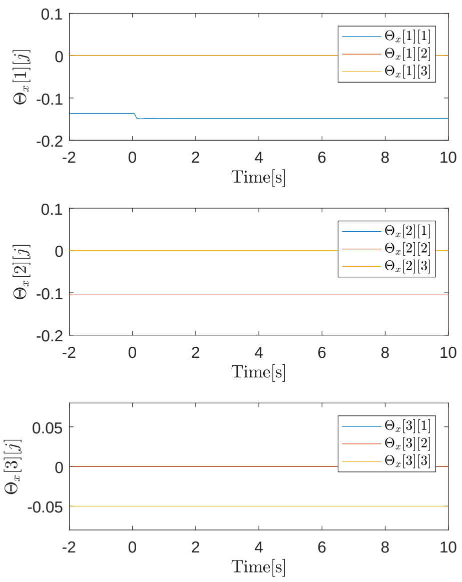


Figure D.7: Simulation 0.2 N.m. disturbance, Θ_x

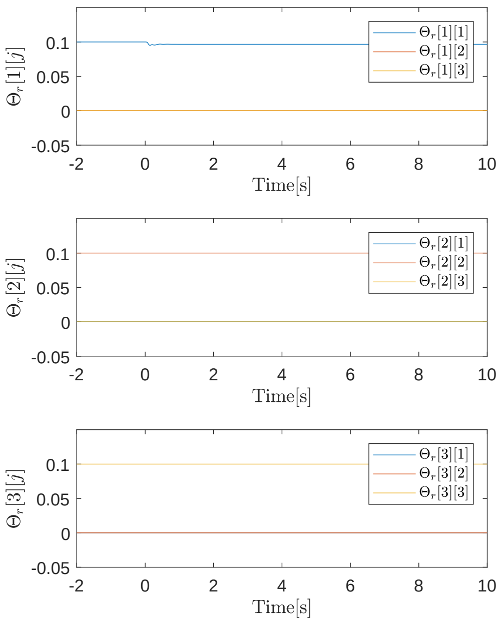


Figure D.8: Simulation 0.2 N.m. disturbance, Θ_r

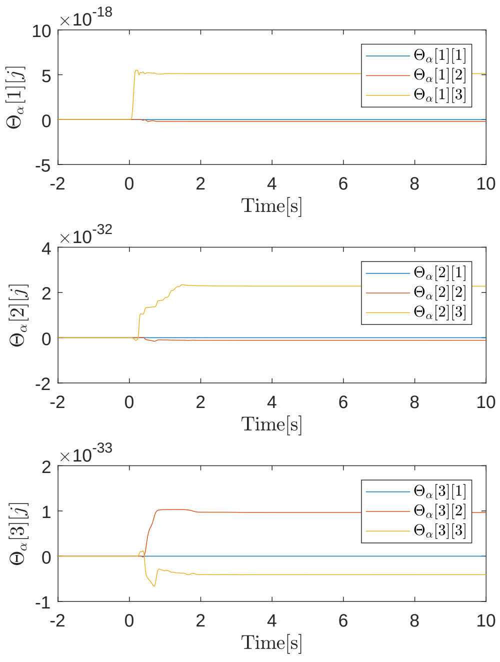


Figure D.9: Simulation 0.2 N.m. disturbance, Θ_α

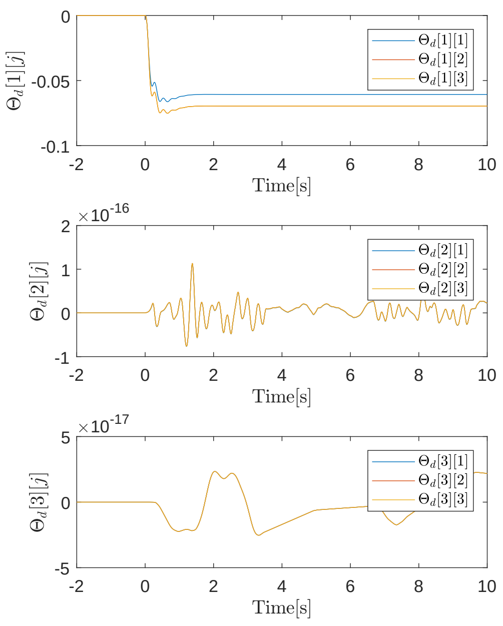


Figure D.10: Simulation 0.2 N.m. disturbance, Θ_d

D.1.3 Addition of Off-centre Weight

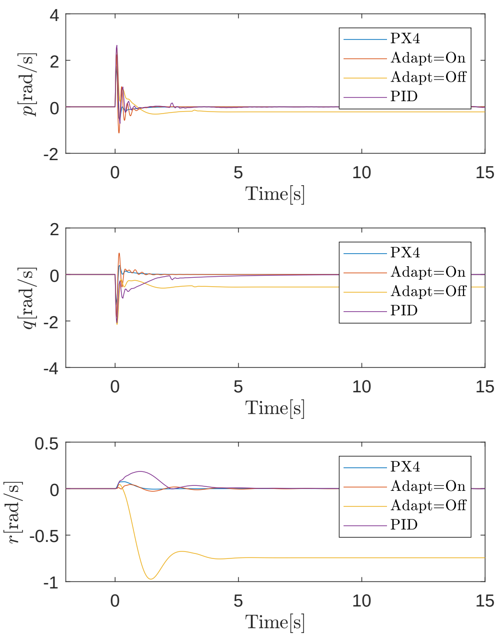


Figure D.11: Combined simulation angle rates for off-centre weight

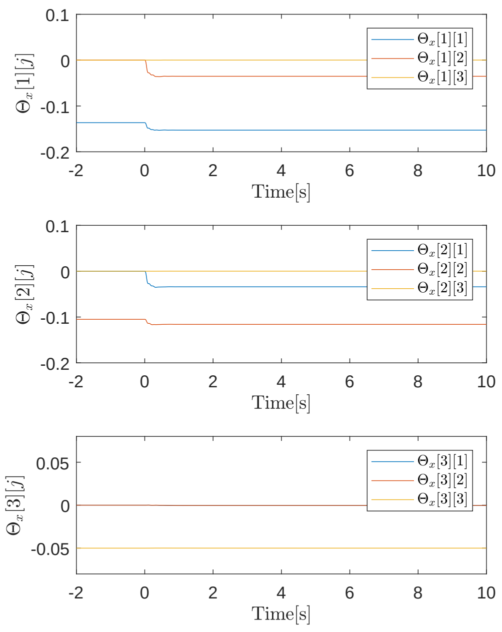


Figure D.12: Simulation off-centre weight, Θ_x

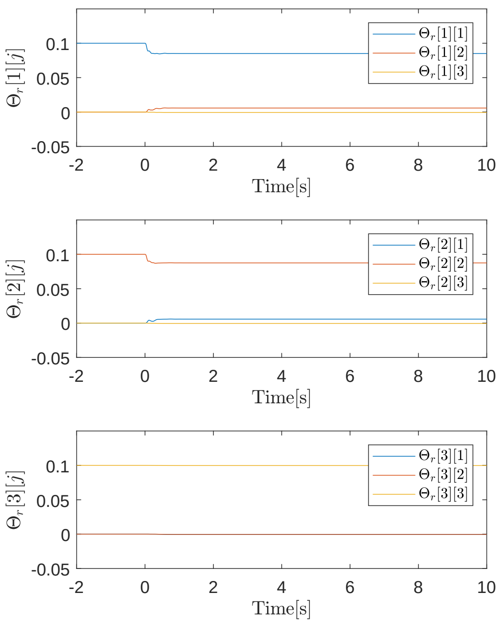


Figure D.13: Simulation off-center weight, Θ_r

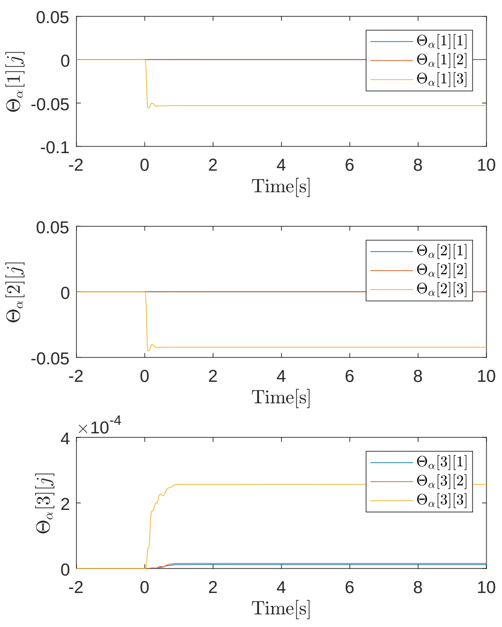


Figure D.14: Simulation off-centre weight, Θ_α

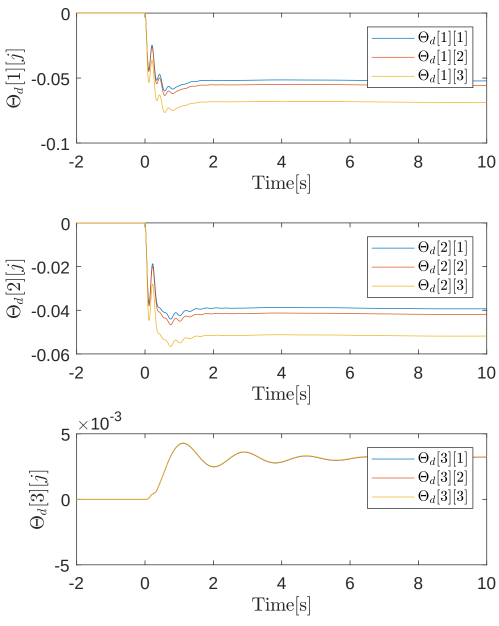


Figure D.15: Simulation off-centre weight, Θ_d

D.1.4 Change in Model

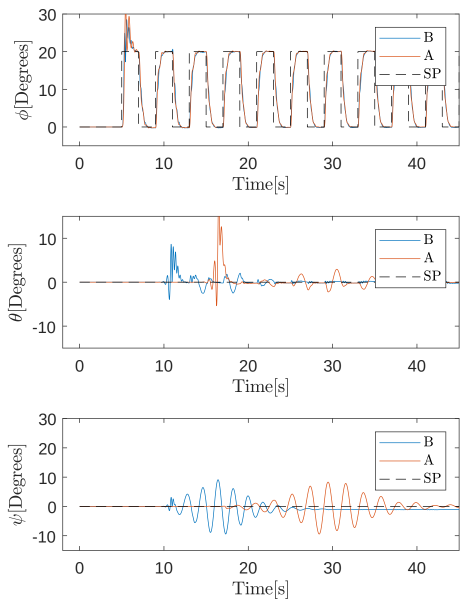


Figure D.16: Change in model simulation angles for square wave learning

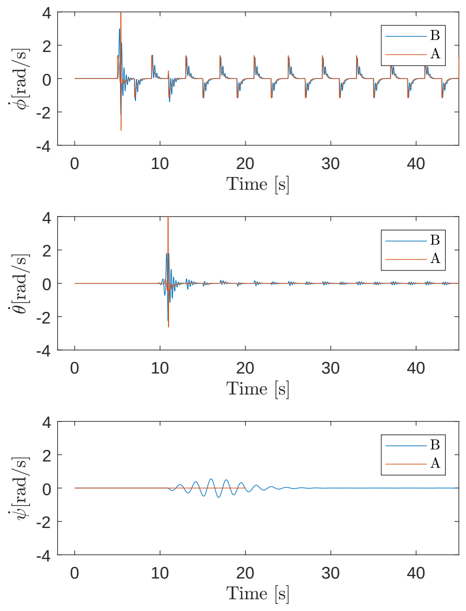


Figure D.17: Change in model simulation angle rates for square wave learning

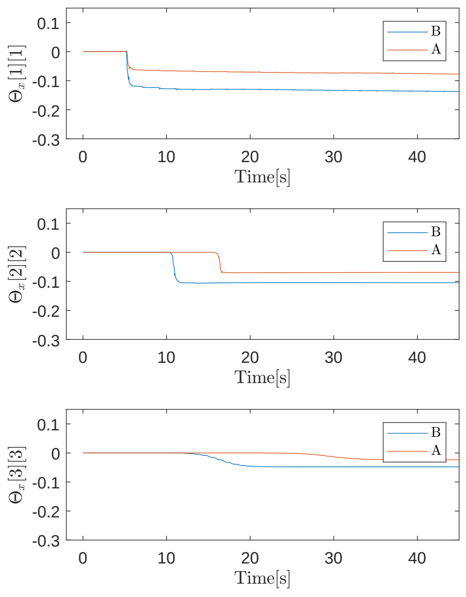


Figure D.18: Simulation of square wave learning for change in model, Θ_x

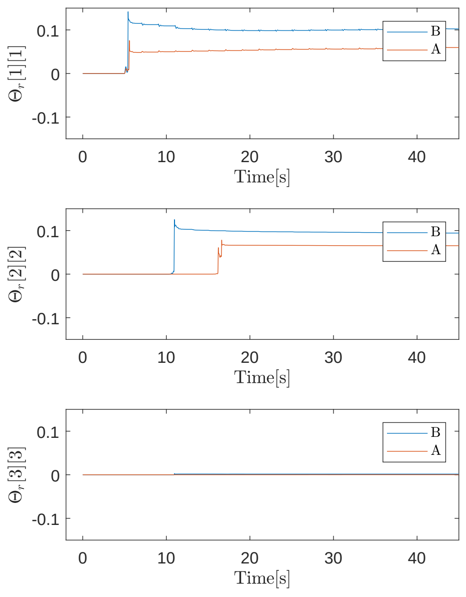


Figure D.19: Simulation of square wave learning for change in model, Θ_r

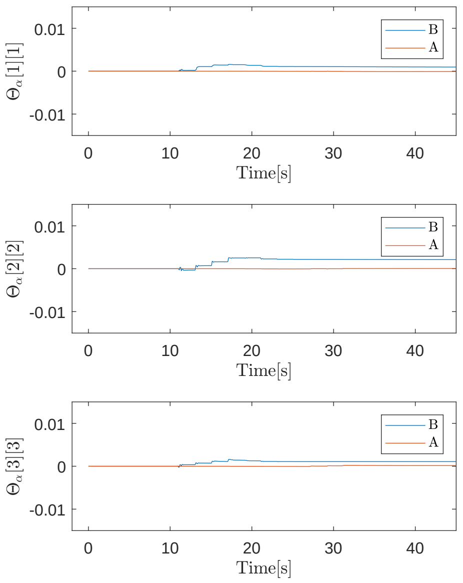


Figure D.20: Simulation of square wave learning for change in model, Θ_α

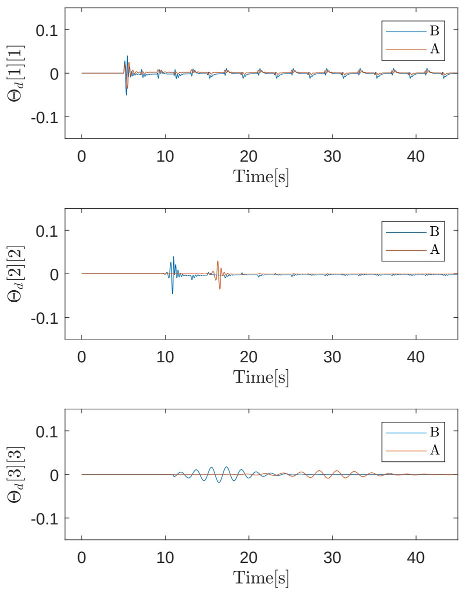


Figure D.21: Simulation of square wave learning for change in model, Θ_d

D.2 Practical

D.2.0.1 20 Degree Roll Step Input

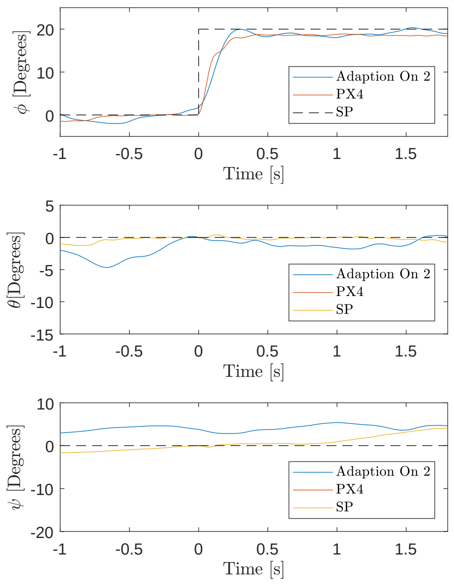


Figure D.22: Practical 20° step input angle (fast reference model)

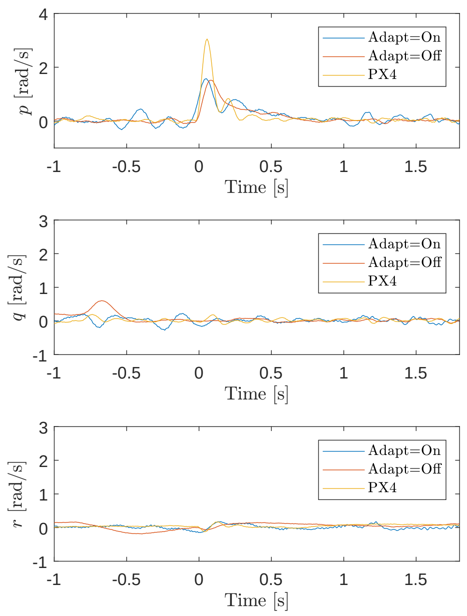


Figure D.23: Combined practical angle rates for 20° step input

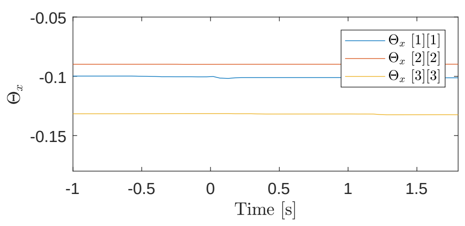


Figure D.24: Practical 20 degree step input, Θ_x

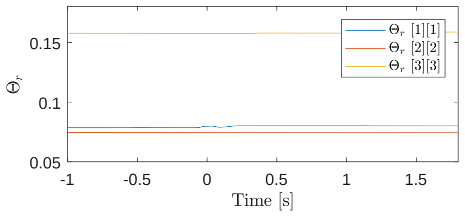


Figure D.25: Practical 20 degree step input, Θ_r

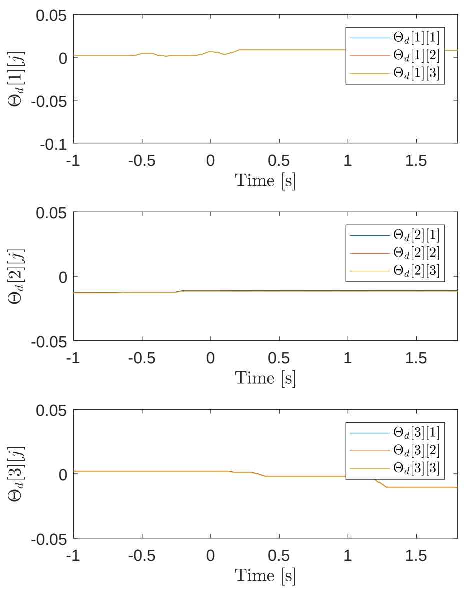


Figure D.26: Practical 20 degree step input, Θ_α

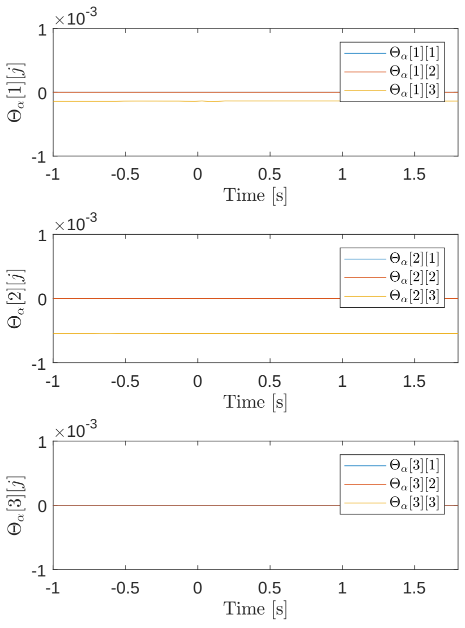


Figure D.27: Practical 20 degree step input, Θ_d

D.2.0.2 Internal Disturbance

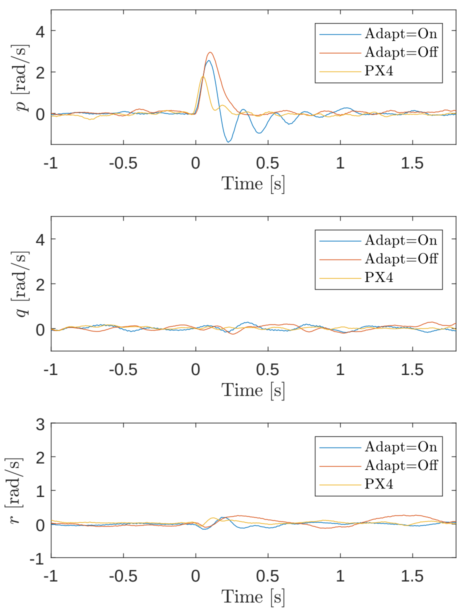


Figure D.28: Combined practical angle rates for 0.2 N.m disturbance

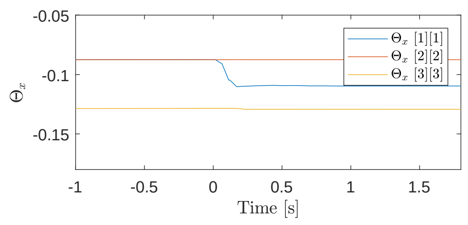


Figure D.29: Practical 0.2 N.m disturbance, Θ_x

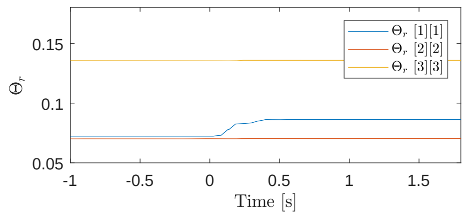


Figure D.30: Practical 0.2 N.m disturbance, Θ_R

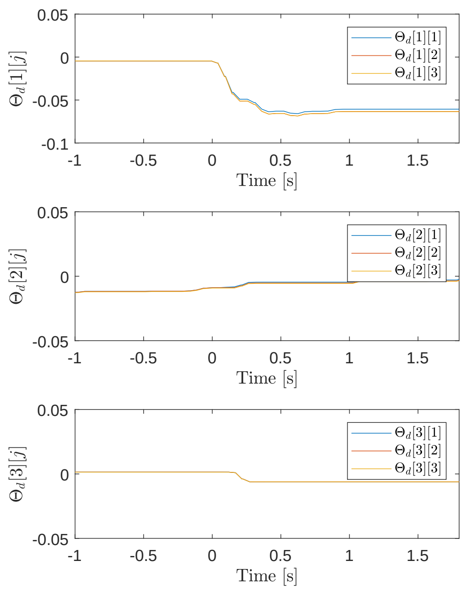


Figure D.31: Practical 0.2 N.m disturbance, Θ_α

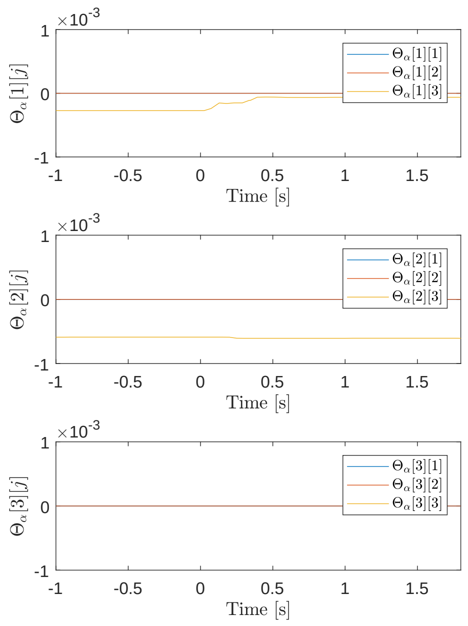


Figure D.32: Practical 0.2 N.m disturbance, Θ_d

D.2.0.3 Addition of Off-centre Weight

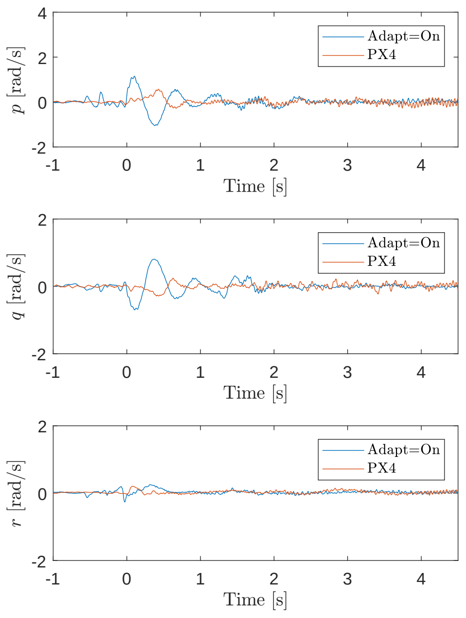


Figure D.33: Combined practical angle rates for off-centre weight

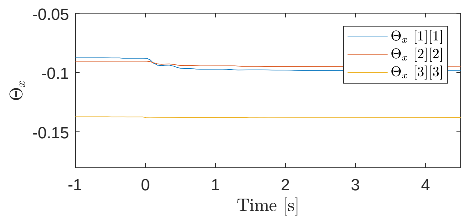


Figure D.34: Practical off-centre weight, Θ_x

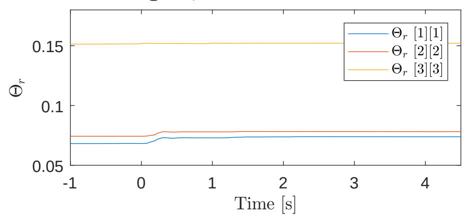


Figure D.35: Practical off-centre weight, Θ_r

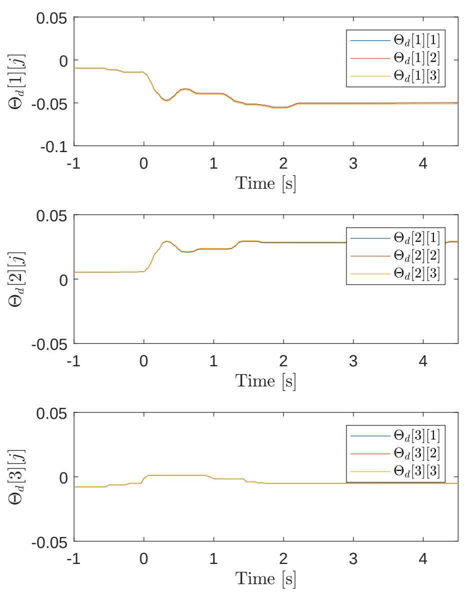


Figure D.36: Practical off-centre weight, Θ_α

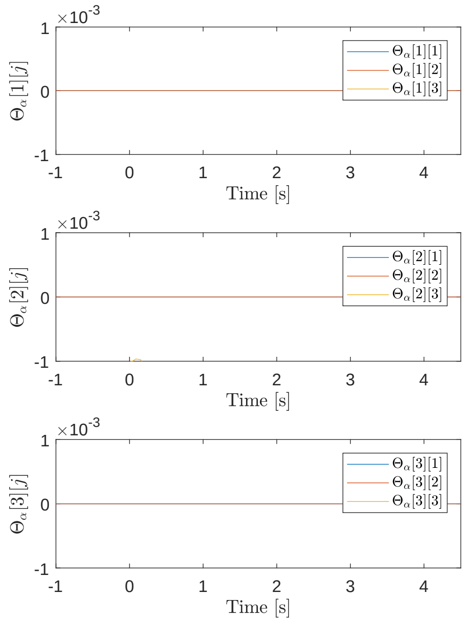


Figure D.37: Practical off-centre weight, Θ_d

D.2.1 Acrobatic Mode

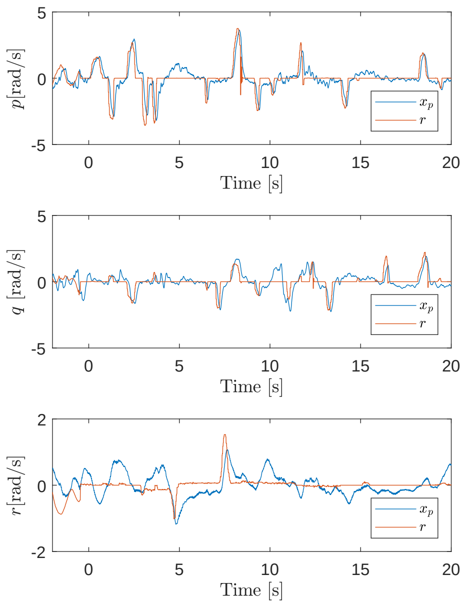


Figure D.38: Practical acrobatic flight angle rates for Adaptive Controller

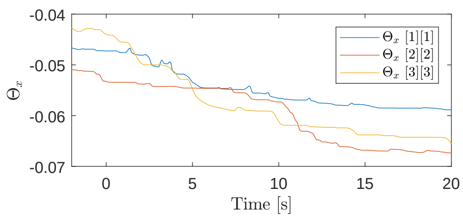


Figure D.39: Practical acrobatic flight Θ_x

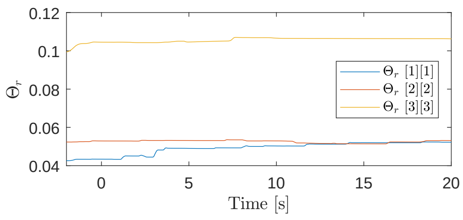


Figure D.40: Practical acrobatic flight Θ_r

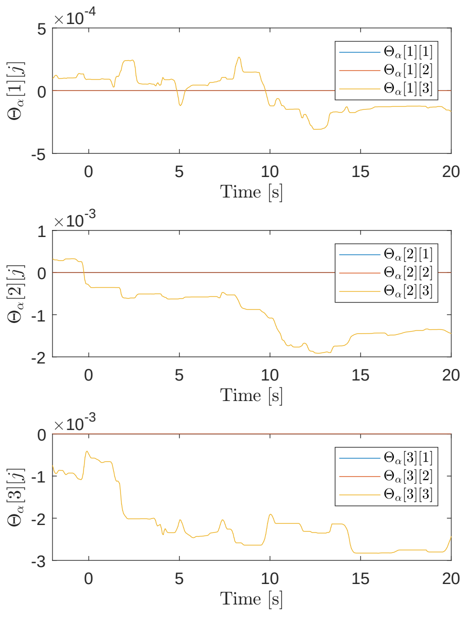


Figure D.41: Practical acrobatic flight Θ_α

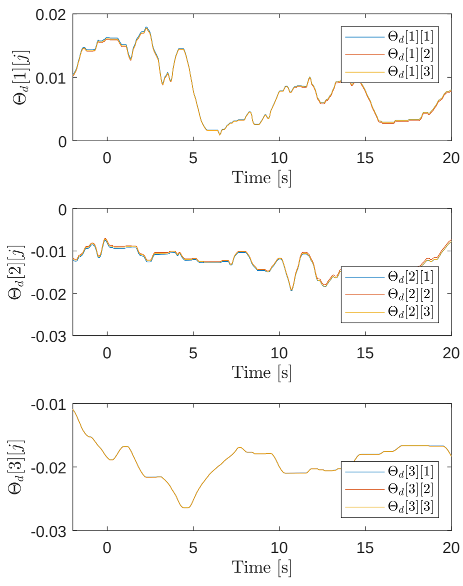


Figure D.42: Practical acrobatic flight Θ_d

D.2.2 Autonomous Mission

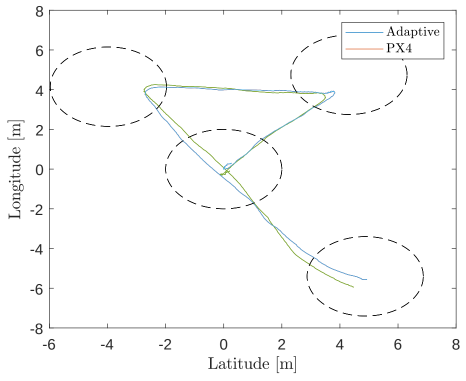


Figure D.43: Combined practical Mission position

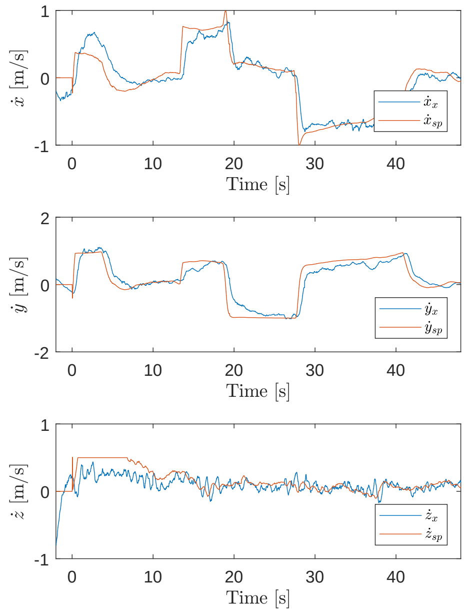


Figure D.44: Practical mission translation rates for PX4 Controller

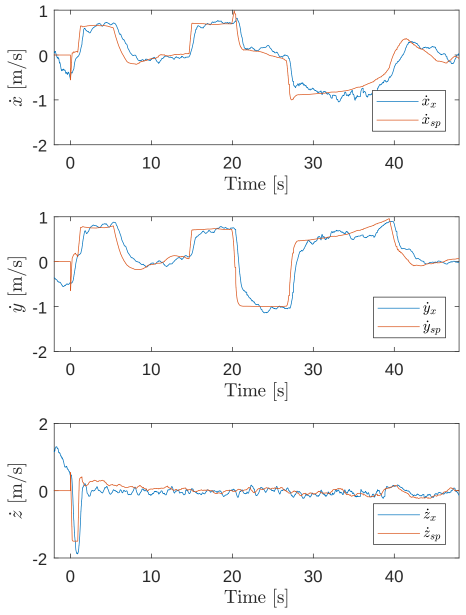


Figure D.45: Practical mission translation rates for Adaptive Controller

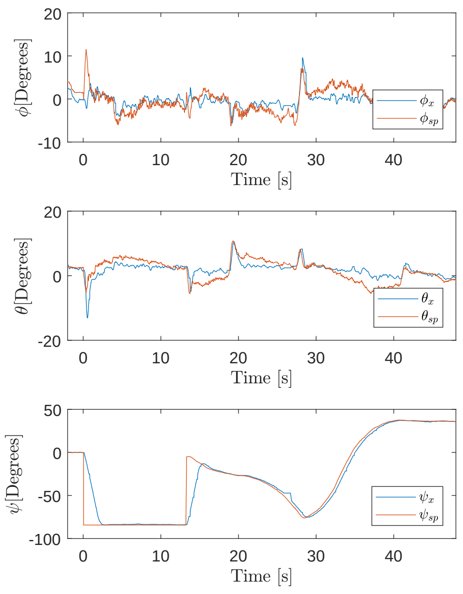


Figure D.46: Practical mission angles for PX4 Controller

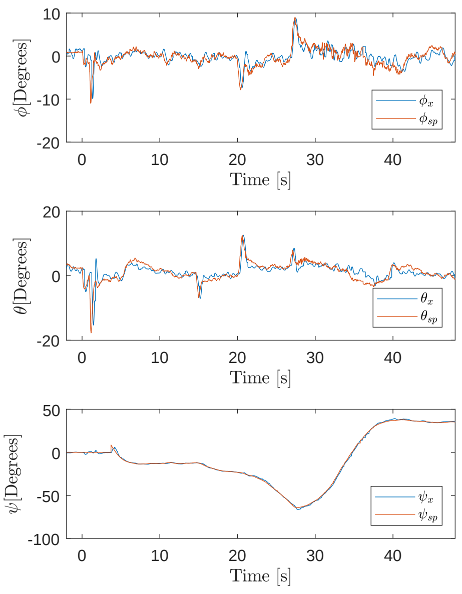


Figure D.47: Practical mission angles for Adaptive Controller

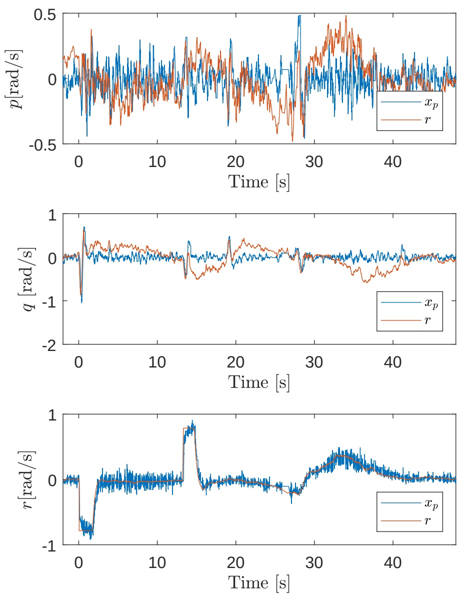


Figure D.48: Practical mission angle rates for PX4 Controller

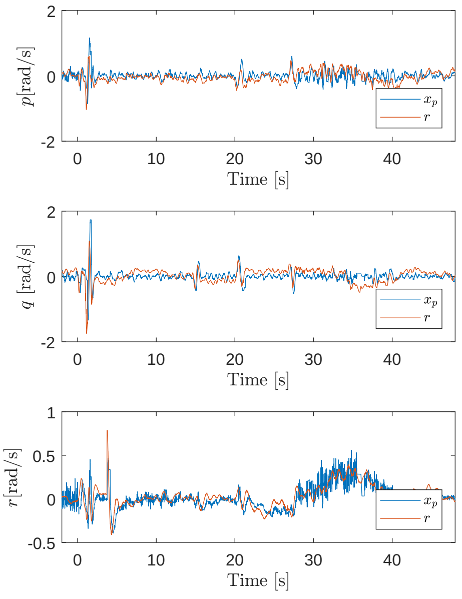


Figure D.49: Practical mission angle rates for Adaptive Controller

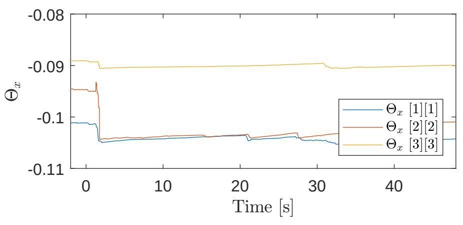


Figure D.50: Practical mission, Θ_x

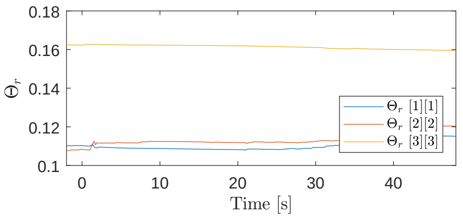


Figure D.51: Practical mission, Θ_r

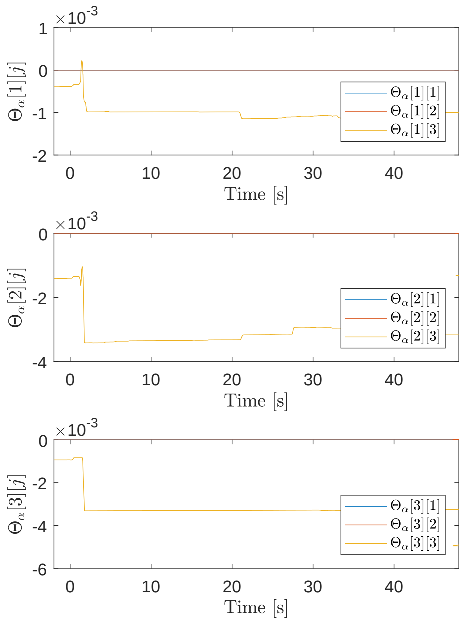


Figure D.52: Practical mission, Θ_α

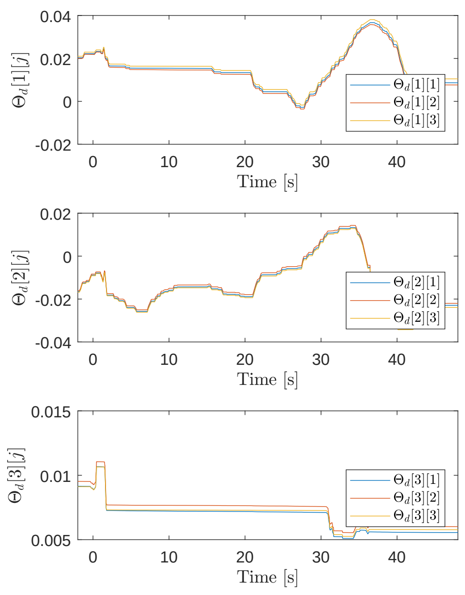


Figure D.53: Practical mission, Θ_D

Appendix E

Additional Calculations

E.1 x(t)=0 possible solution

Consider the non-linear differential equation

$$\frac{dx}{dt} = f(x) \tag{E.1}$$

with initial conditions $f(0) = 0$. This initial condition yields $x(t) = 0$ as a possible solution to the equation. A more detailed explanation is given here.

This initial condition forces $x(t)$ to have a gradient of zero at $x = 0$. $x(t) = 0$ is therefore one of the possible solutions. Consider $\frac{dx}{dt} = x + 5$. The constant will determine the gradient of $x(t)$ at $t = 0$. $x(t) = 0$ can therefore never be a solution to the equation. However, due to the initial condition $f(0) = 0$, $x + 5$ is not an appropriate function since it does not satisfy the condition, $f(0) = 0 \neq 5$. An example of a function which does satisfy the condition is $f(x) = x^2$. The possible solutions to $\frac{dx}{dt} = x^2$ are plotted in figure E.1 which shows $x(t) = 0$ as a possible solution. The differential equation solution is given by $x(t) = \frac{1}{c_1 - t}$. Choosing $c_1 = \infty$ yields $x(t) = 0$.

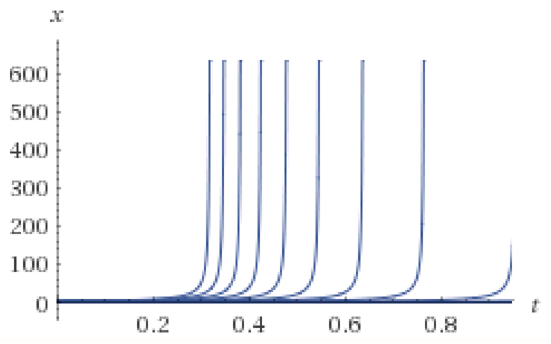


Figure E.1: Quadratic functions

E.2 Model

E.2.1 Rotation Matrix

In order to introduce the rotation matrix, first consider the simple two dimensional case as shown in Figure 2.2.

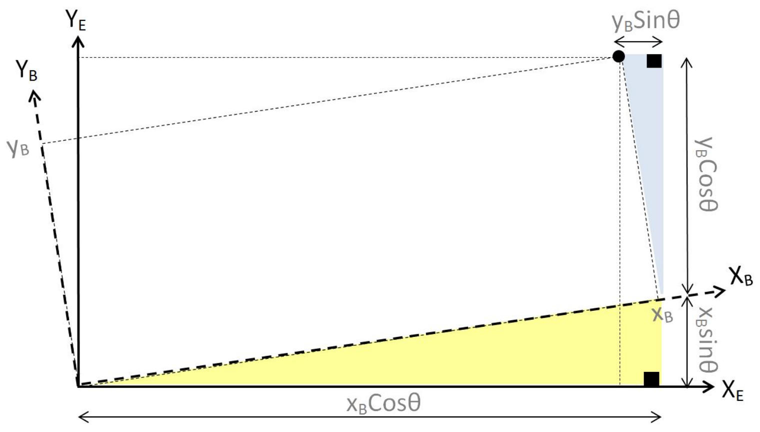


Figure E.2: Two Dimensional Rotation Matrix (Mathworks, 2017*b*; Noureldin *et al.*, 2013)

Consider point C in the body frame with x coordinate, x_{Body}, and y coordinate, y_{Body}. Point C can be given in the earth frame using trigonometry. The coordinates in the earth frame are given by equations E.2 and E.3.

$$x^E = x^B \cos\theta - y^B \sin\theta \tag{E.2}$$

$$y^E = y^B \cos\theta + x^B \sin\theta \tag{E.3}$$

The coordinates of C in the body and earth frame can therefore be related by equation E.4.

$$\begin{aligned}\begin{bmatrix} x^E \\ y^E \end{bmatrix} &= \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} x^B \\ y^B \end{bmatrix} \\ &= R \begin{bmatrix} x^B \\ y^B \end{bmatrix}\end{aligned} \tag{E.4}$$

where

$$R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{E.5}$$

The two dimensional case can be extended to the three dimensional case by considering the rotation about each main axis separately. This will result in three rotation matrices which can be post multiplied to obtain the complete rotation matrix.

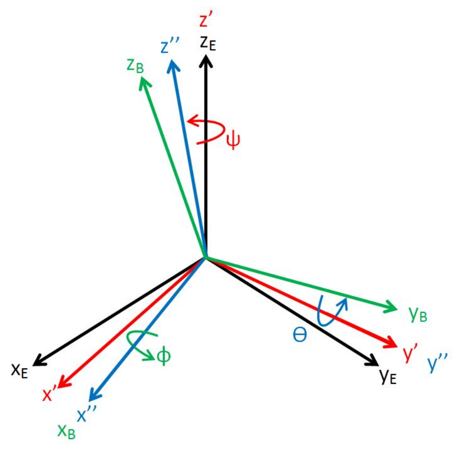


Figure E.3: Rotation about z-y-x axes

Rotation about the z-y-x axes, in that order, is shown in figure E.3. Equation E.6 gives the rotation matrix around the z axis.

$$R(\psi, z) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{E.6}$$

Equation E.7 gives the rotation matrix around the y axis.

$$R(\theta, y) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{E.7}$$

Equation E.8 gives the rotation matrix around the x axis.

$$R(\phi, x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{E.8}$$

Post multiplying the three rotation matrices above yields the rotation matrix, equation E.9, which relates the body frame orientation to that of the earth frame. The same Body 3-2-1 sequence, as mentioned in Chapter 2, is used here (Henderson, 1977).

$$\begin{aligned} R_\Phi &= R(\psi, z)R(\theta, y)R(\phi, x) \\ &= \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\theta + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix} \end{aligned} \tag{E.9}$$

E.2.2 Transfer Matrix

The angular velocities in the body and earth frame can be related in a similar manner by means of a transfer matrix (Chin, 2009). Consider the body 3-2-1 sequence once more. A yaw rotation is applied first, follow by a pitch rotation and a roll rotation. The Euler yaw rotation applied is not about the body yaw axis after the other two rotations are also completed. The pitch axis also shifts due to roll. Since roll is the last rotation no change in axis is caused. Considering this sequence of rotation and adjusting for the effect the successive rotations yields equation E.10.

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \dot{\phi} \\ 0 \\ 0 \end{bmatrix} + R(\phi, x)^{-1} \begin{bmatrix} 0 \\ \dot{\theta} \\ 0 \end{bmatrix} + R(\phi, x)^{-}1R(\theta, y)^{-1} \begin{bmatrix} 0 \\ 0 \\ \dot{\psi} \end{bmatrix} = T_{\Theta}^{-1} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{E.10}$$

Simplifying equation E.10 yields the transfer matrix E.11.

$$T_{\Phi} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\cos\theta & \sin\phi\cos\theta \end{bmatrix} \tag{E.11}$$

E.2.3 Force-moment vector body frame

The force-moment vector can be divided into three components: a gravitational component, a component due to the gyroscopic effects and lastly a component due to the forces and torques directly produced by the propellers. The gravity is only along the z-axis in the earth frame. The rotation matrix, equation E.9, is used to transform the gravitational force to the body frame.

$$G^B(\Psi) = \begin{bmatrix} F_G^B \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} R_{\Phi}^T F_G^E \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} R_{\Phi}^T \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} \\ 0_{3x1} \end{bmatrix} = \begin{bmatrix} mg\sin\theta \\ -mg\cos\theta\sin\phi \\ -mg\cos\theta\sin\phi \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.12}$$

A gyroscopic torque is created due to the simultaneous rotation of the propellers and the quadcopter itself (pitch and roll). Consider figure E.4. The propeller spins at an angular velocity of Ω. This causes an angular moment, L, perpendicular to the plain of rotation. The angular momentum is defined by equation E.13.

$$L = J_{TP}\Omega \tag{E.13}$$

where J_{TP} is the inertia of the motor's rotating component. When the quadcopter tilts around the x-axis due to roll; or y-axis due to pitch the quadcoter's orientation changes and the angular momentum vector tilts with it. The new angular momentum vector is given by L'. A change in angular momentum therefore occurred, dL. A change in angular momentum causes a torque defined by equation E.14.

$$\tau = \frac{dL}{dt} \tag{E.14}$$

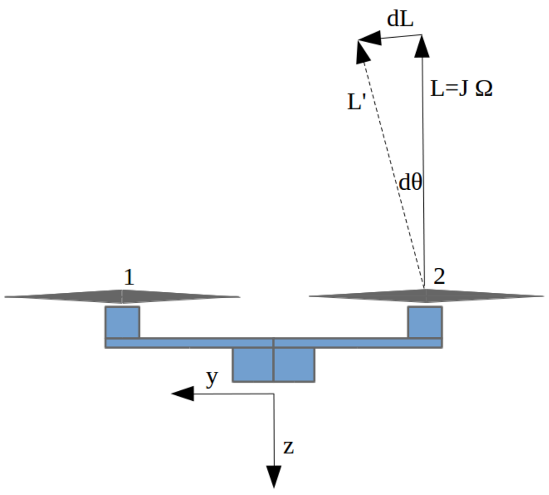


Figure E.4: Gyroscopic force on quadcopter

The tilt is caused by an angular velocity, ω.

$$\omega = \frac{d\theta}{dt} = \frac{dL}{L}\frac{1}{dt} \tag{E.15}$$

using the property $\alpha = arc/r$, with α L the change in angle, L the radius and dL the approximate arc length. Substituting equations E.14 and E.15 into E.15 and solving yields

$$\tau = \omega J_{TP}\ \Omega \tag{E.16}$$

The torque direction can be deduced from the right hand rule, where dL is in the direction of one's thumb. The torque is therefore applied around the y-axis for the case of figure E.4.
Applying the above method on all four motors and considering both pitch and roll, yields equation E.17 [1].

$$\tau_{gyro} = \begin{bmatrix} \tau_{gyro,p} \\ \tau_{gyro,q} \\ \tau_{gyro,r} \end{bmatrix} = \begin{bmatrix} -qJ_{TP}\Omega_1 - qJ_{TP}\Omega_2 + qJ_{TP}\Omega_3 + qJ_{TP}\Omega_4 \\ pJ_{TP}\Omega_1 + pJ_{TP}\Omega_2 - pJ_{TP}\Omega_3 - pJ_{TP}\Omega_4 \\ 0 \end{bmatrix} \tag{E.17}$$

In shorthand this can be written as in equation E.18 (including forces).

$$O^B(v)\Omega_v = \begin{bmatrix} 0_{3x1} \\ J_{TP} \begin{bmatrix} -q \\ p \\ 0 \end{bmatrix})\Omega_\Sigma \end{bmatrix} = J_{TP} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -q & -q & q & +q \\ p & p & -p & -p \\ 0 & 0 & 0 & 0 \end{bmatrix} \Omega_v \tag{E.18}$$

[1]The Ω_i values strictly have magnitude and not direction, therefore all are positive

where

$$\Omega_\Sigma = \Omega_1 + \Omega_2 - \Omega_3 - \Omega_4 \tag{E.19}$$

$$\Omega_v = [\Omega_1\ \Omega_2\ \Omega_3\ \Omega_4]^T \tag{E.20}$$

The moments and forces directly generated by the propellers constitute the last component. It can be shown that the square of the propeller speed is proportional to the thrust force generated (Greitzer *et al.*, 2007). The forces and moments are given with respect to the body frame. The first three elements of the vector are found by summing the forces in the body frame. The fourth and fifth elements are found by taking the moments around the x^B and y^B-axes respectively. The quadcopter configuration being used effect the equations of these two moments. The last element is the simplified net torque created around the z^B-axis due to the reaction torques of the propellers. This relationship is also related to the square of the propeller speeds (Greitzer *et al.*, 2007). Equations E.21 and E.22 shows movement vector containing the total propeller force and the three moments around the axes. U_1 and U_4 remain the same despite the configuration. Equation E.21 is for the plus configuration and equation E.22 is for the cross configuration.

$$U^B(\Omega_v) = E^B\Omega_v^2 = \begin{bmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ lb(\Omega_2^2 - \Omega_1^2) \\ lb(\Omega_3^2 - \Omega_4^2) \\ d(-\Omega_1^2 - \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{E.21}$$

$$U^B(\Omega_v) = E^B\Omega_v^2 = \begin{bmatrix} 0 \\ 0 \\ U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ lbsin(45^\circ)(\Omega_3^2 + \Omega_2^2 - \Omega_1^2 - \Omega_4^2) \\ lbsin(45^\circ)(\Omega_1^2 + \Omega_3^2 - \Omega_4^2 - \Omega_2^2) \\ d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \end{bmatrix} \tag{E.22}$$

The values of b and d are constants dependent on the motor-propeller combination being used (Greitzer *et al.*, 2007) and l is the distance from the motor axes to the centre of mass.

Summing the three components and substituting it into equation 4.30 yields

$$M^B\dot{v} + C^B(v)v = G^B(\Psi) + O^B(v)\Omega_\Sigma + E^B\Omega_v^2 \tag{E.23}$$

E.2.4 Force-moment vector H-frame

The system inertia matrix and the gyroscopic propeller matrix remains the same. The new Coriolis-centripetal matrix, gravitational vector and moment matrix are given by equation E.24, E.25 and E.26 respectively.

$$C^H(\Psi) = \begin{bmatrix} 0 & 0 \\ 0 & -S(I_{xyz}\omega^B) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & I_{ZZ}r & -I_{YY}q \\ 0 & 0 & 0 & -I_{zz}r & 0 & I_{xx}p \\ 0 & 0 & 0 & I_{yy}q & -I_{xx}p & 0 \end{bmatrix} \tag{E.24}$$

$$G^H = \begin{bmatrix} F_G^E \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -mg \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{E.25}$$

$$U^H(\Omega_v) = E^H(\Psi)\Omega_v^2 = \begin{bmatrix} R_\Phi & 0 \\ 0 & I_{xyz} \end{bmatrix} E^B\Omega_v^2 = \begin{bmatrix} (sin\psi sin\phi + cos\psi sin\theta cos\phi)U_1 \\ (-cos\psi sin\phi + sin\psi sin\theta cos\phi)U_1 \\ (cos\theta cos\phi)U_1 \\ U_2 \\ U_3 \\ U_4 \end{bmatrix} \tag{E.26}$$

E.3 Derivative of J

$$J = \frac{1}{2}e^T Pe + \frac{1}{2}\tilde{\Theta}_x^T\Gamma^{-1}\tilde{\Theta}_x + \frac{1}{2}Tr[\tilde{\Theta}_r^T\Gamma^{-1}\tilde{\Theta}_r] + \frac{1}{2}Tr[\tilde{\Theta}_\alpha^T\Gamma^{-1}\tilde{\Theta}_\alpha] + \frac{1}{2}Tr[\tilde{\Theta}_d^T\Gamma^{-1}\tilde{\Theta}_d] \tag{E.27}$$

For the system to be stable the derivative of equation needs to atleast be negative semi definite. Taking the derivative of equation E.27 by means of the product rule Petersen and Pedersen (2012) yields

$$\dot{J} = \frac{1}{2}[\dot{e}^T Pe + e^T P\dot{e}] + Tr[\tilde{\Theta}_x^T\Gamma_x^{-1}\dot{\tilde{\Theta}}_x] + Tr[\tilde{\Theta}_r^T\Gamma_r^{-1}\dot{\tilde{\Theta}}_r] + Tr[\tilde{\Theta}_\alpha^T\Gamma_\alpha^{-1}\dot{\tilde{\Theta}}_\alpha] + Tr[\tilde{\Theta}_d^T\Gamma_d^{-1}\dot{\tilde{\Theta}}_d] \tag{E.28}$$

using

$$\frac{d}{dt}Tr(\tilde{\Theta}^T\Gamma^{-1}\tilde{\Theta}) = Tr(\frac{d}{dt}\tilde{\Theta}^T\Gamma^{-1}\tilde{\Theta}) = Tr(\dot{\tilde{\Theta}}^T\Gamma^{-1}\tilde{\Theta}) + Tr(\tilde{\Theta}^T\Gamma^{-1}\dot{\tilde{\Theta}}) = 2Tr(\tilde{\Theta}^T\Gamma^{-1}\dot{\tilde{\Theta}}) \tag{E.29}$$

since the trace operator is used, only the diagonal elements are of importance. The transpose of the terms in the second trace can therefore be taken as follows.

$$\begin{aligned} Tr(\tilde{\Theta}^T\Gamma^{-1}\dot{\tilde{\Theta}}) &= Tr((\tilde{\Theta}^T\Gamma^{-1}\dot{\tilde{\Theta}})^T) = Tr(\dot{\tilde{\Theta}}^T(\tilde{\Theta}^T\Gamma^{-1})^T) \\ &= Tr(\dot{\tilde{\Theta}}^T(\Gamma^{-1})^T\tilde{\Theta}) = Tr(\dot{\tilde{\Theta}}^T\Gamma^{-1}\tilde{\Theta}) \end{aligned} \tag{E.30}$$

Next substitute equation 5.59 into the first two terms of equation 5.61.

$$e^T P\dot{e} = e^T P[A_d e + B_p\xi\tilde{\Theta}_x x + B_p\tilde{\xi}\Theta_r r + B_p\xi\tilde{\Theta}_\alpha f(x) + B_p\xi\tilde{\Theta}_d i] \tag{E.31}$$

$$\dot{e}^T Pe = [A_d e + B_p\xi\tilde{\Theta}_x x + B_p\xi\tilde{\Theta}_r r + B_p\xi\tilde{\Theta}_\alpha f(x) + B_p\xi\tilde{\Theta}_d i]^T Pe \tag{E.32}$$

Adding equation E.31 and E.32 and solving

$$\begin{aligned}\frac{1}{2}[e^T P\dot{e} + \dot{e}^T Pe] =& \frac{1}{2}e^T(PA_d + A_d^T P)e + x^T\tilde{\Theta}_x^T\xi^T B_p^T Pe \\ &+ r^T\tilde{\Theta}_r^T\xi^T B_p^T Pe + f(x)^T\tilde{\Theta}_\alpha^T\xi^T B_p^T Pe \\ &+ i^T\tilde{\Theta}_d^T\xi^T B_p^T Pe\end{aligned} \tag{E.33}$$

Equation E.33 needs to be substituted into equation E.28 to find the complete derivative of the Lyapunov function.

$$\begin{aligned}\dot{J} =& \frac{1}{2}e^T(PA_d + A_d^T P)e + x^T\tilde{\Theta}_x^T\xi^T B_p^T Pe + r^T\tilde{\Theta}_r^T\xi^T B_p^T Pe \\ &+ f(x)^T\tilde{\Theta}_\alpha^T\xi^T B_p^T Pe + i^T\tilde{\Theta}_d^T B_p^T Pe + Tr[\tilde{\Theta}_x^T\Gamma_x^{-1}\dot{\tilde{\Theta}}_x] \\ &+ Tr[\tilde{\Theta}_r^T\Gamma_r^{-1}\dot{\tilde{\Theta}}_r] + Tr[\tilde{\Theta}_\alpha^T\Gamma_\alpha^{-1}\dot{\tilde{\Theta}}_\alpha] + Tr[\tilde{\Theta}_d^T\Gamma_d^{-1}\dot{\tilde{\Theta}}_d]\end{aligned} \tag{E.34}$$

In order to achieve a negative semi-definite $\dot{V}$ we substitute equation E.35 into the first term.

$$A^T P + PA = -Q \tag{E.35}$$

where Q is symmetric positive definite. We then want to eliminate the remaining terms. Substituting equation E.35 and using the following relationships

$$x^T\tilde{\Theta}_x^T\xi^T B_p^T Pe = Tr[\tilde{\Theta}_x^T\xi^T B_p^T Pex^T] \tag{E.36}$$

$$r^T\tilde{\Theta}_r^T\xi^T B_p^T Pe = Tr[\tilde{\Theta}_r^T\xi^T B_p^T Per^T] \tag{E.37}$$

$$f(x)^T\tilde{\Theta}_\alpha^T\xi^T B_p^T Pe = Tr[\tilde{\Theta}_\alpha^T\xi^T B_p^T Pef(x)^T] \tag{E.38}$$

$$i^T\tilde{\Theta}_d^T\xi^T B_p^T Pe = Tr[\tilde{\Theta}_d^T\xi^T B_p^T Pei^T] \tag{E.39}$$

$\dot{J}$ becomes

$$\begin{aligned}\dot{J} =& \frac{1}{2}e^T(PA_d + A_d^T P)e + Tr[\tilde{\Theta}_x^T\xi^T B_p^T Pex^T] \\ &+ Tr[\tilde{\Theta}_r^T\xi^T B_p^T Per^T] + Tr[\tilde{\Theta}_\alpha^T\xi^T B_p^T Pef(x)^T] \\ &+ Tr[\tilde{\Theta}_d^T\xi^T B_p^T Pei^T] + Tr[\tilde{\Theta}_x^T\Gamma_x^{-1}\dot{\tilde{\Theta}}_x] \\ &+ Tr[\tilde{\Theta}_r^T\Gamma_r^{-1}\dot{\tilde{\Theta}}_r] + Tr[\tilde{\Theta}_\alpha^T\Gamma_\alpha^{-1}\dot{\tilde{\Theta}}_\alpha] + Tr[\tilde{\Theta}_d^T\Gamma_d^{-1}\dot{\tilde{\Theta}}_d]\end{aligned} \tag{E.40}$$

Grouping the terms together as follows

$$\begin{aligned}\dot{J} =& -\frac{1}{2}[e^T Qe + Tr[\tilde{\Theta}_x^T(\xi^T B_p^T Pex^T + \Gamma_x^{-1}\dot{\tilde{\Theta}}_x)]Tr[\tilde{\Theta}_r^T(\xi^T B_p^T Per^T + \Gamma_r^{-1}\dot{\tilde{\Theta}}_r)] \\ &+ Tr[\tilde{\Theta}_\alpha^T(\xi^T B_p^T Pef(x)^T + \Gamma_\alpha^{-1}\dot{\tilde{\Theta}}_\alpha)] + Tr[\tilde{\Theta}_d^T(\xi^T B_p^T Pei^T + \Gamma_d^{-1}\dot{\tilde{\Theta}}_d)]\end{aligned} \tag{E.41}$$

Appendix F

Control parameter summary

Table F.1: Controllers' parameters summary for quadcopter B

Controller	Loop	Parameter	Simulation (ϕ, θ, ψ)	Practical (ϕ, θ, ψ)
PID	-	P	20,20,20	-
		I	3,3,3	-
		D	5.2,5.2,10	-
PX4	Angle	P	6.4,6.4,6.4	6.4,6.4,6.4
	Rate	P	0.152,0.152,0.152	0.152,0.152,0.152
		I	0.1,0.1,0.1	[0.1,0.1,0.1
		D	0.0044,0.0044,0.0044	0.0044,0.0044,0.0044
		ff	0,0,0	0,0,0
		K_a	0.75,0.75,0.75	0.75,0.75,0.75
Adaptive	Angle	$\frac{1}{T}$	4,4,0	4,4,0
		K_p	3,3,3	6,6,6
		K_i	0.1,0.1,0.1	1.5,1.5,1.5
	Rate	$\Upsilon_{x,r,\alpha}$	4.8,4.8,4.8	1.9,1.9,1.9
		Υ_d	12,12,12	6.4,6.4,6.4
		$\Theta_d er$	0.003,0.003,0.003	0.003,0.003,0.003
		σ	0.06,0.06,0.06	0.06,0.06,0.06
		A_p	-15,-15,-9	-15,-15,-9
		B_p	15,15,9	15,15,9
		Q	0.001,0.001,0.001	0.001,0.001,0.001
		Θ_x(limit)	-0.3 to -0.005 (all)	-0.3 to -0.005 (all)
		Θ_r(limit)	0.005 to 0.3 (all)	0.005 to 0.3 (all)
		Θ_d(limit)	-0.2 to 0.2 (all)	-0.2 to 0.2 (all)
		Θ_α(limit)	-0.2 to 0.2 (all)	-0.2 to 0.2 (all)

List of References

Achtelik, M., Bierling, T., Wang, J., Höcht, L. and Holzapfel, F. (2011). Adaptive control of a quadcopter in the presence of large/complete parameter uncertainties. In: *AIAA Meeting Papers: Infotech@Aerospace*. Available at `https://doi.org/10.2514/6.2011-1485`.

Al-Hokayem, P. and Gallestey, E. (2015 March). Lyapunov stability notes. [Online]. Spring 2017, 227-0207 Nichtlineare Systeme und Regelung (Nonlinear Systems and Control), Available at `http://people.ee.ethz.ch/~apnoco/Lectures2015/03_Lyapunov.pdf`.

Ardupilot (2016). *PX4 Pro Autopilot Software*. [Online]. Available at: `https://github.com/PX4/Firmware`, [2016, September 25].

Astrom, K. and Wittenmark, B. (1994). *Adaptive Control*. Dover Publications.

Baker, M. (2016). *Skew Symmetric Matrix*. [Online]. Available at: `http://www.euclideanspace.com/maths/algebra/matrix/functions/skew/`, [2017, August 12].

Bertelsen, A. and Thomsen, M. (2014). *Modelling and L1 Adaptive Control of a Quadcopter*. Masters, Technical University of Denmark.

Bresciani, T. (2008). *Modelling, Identification and Control of a Quadrotor Helicopter*. Masters, Lund University.

Chin, H. (2009). Kinematics of moving frames. [Online]. MIT Open Courseware, Fall 2009, 2.017J Design of Electromechanical Robotic Systems, Available at `https://ocw.mit.edu/courses/mechanical-engineering/2-017j-design-of-electromechanical-robotic-systems-fall-2009/course-text/MIT2_017JF09_ch09.pdf`.

Control Tutorials (2016). *Introduction: PID Controller Design*. [Online]. Available at: `http://ctms.engin.umich.edu/CTMS/index.php?example=Introduction\&section=ControlPID`, [2016, November 16].

De Azevedo, F. (2014). *Complete system for quadcopter control*. Undergraduate thesis, University of Rio Grande do Sul.

Dictionary.com (2017). *Adaption*. [Online]. Available at: `http://www.dictionary.com/browse/adaption`, [2016, January 14].

Dronesmith Technologies (2016). *Why We Chose PX4 (vs APM) as Luci's Default Firmware.* [Online]. Available at: `https://medium.com/@Dronesmith/why-we-chose-px4-vs-apm-as-lucis-default-firmware-ea39f4514bef`, [2017, August 25].

Dumont, G. (2013 January). Model reference adaptive control - an overview. [Online]. Workshop slides, Available at `http://www.cds.caltech.edu/archive/help/uploads/wiki/files/140/IEEE_WorkShop_Slides_Lavretsky.pdf`.

Dydek, Z. (2010*a*). *Adaptive Control of Unmanned Aerial Systems.* Doctoral, Massachusetts Institute of Technology.

Dydek, Z. (2010*b*). *Adaptive Control of Unmanned Aerial Systems.* Doctoral, Masachusetts Institute of Technology.

Egardt, B. (1979). *Stability of Adaptive Controllers.* Lecture Notes in Control and Information Sciences. Springer.

Emran, B. and Yesildirek, A. (2014). Robust nonlinear composite adaptive control of quadrotor. *International Journal of Digital Informaiton and Wireless Communications*, vol. 4, pp. 213–225.

ETH Zürich (2017). Eth softwre to become a standard for drones. Available at: `https://www.ethz.ch/en/news-and-events/eth-news/news/2016/01/eth-software-to-become-standard-for-drones.html`, [2017, October 29].

Fourie, D. (2016). *Test rig for electric motor and propeller.* Undergraduate thesis, University of Stellenbosch.

Greitzer, E.M., Spakovszky, Z.S. and Waitz, I.A. (2007). *Thermodynamics and Propulsion.* [Online]. Available at: `http://web.mit.edu/16.unified/www/FALL/thermodynamics/notes/node86.html`, [2017, June 5].

Harrell, R. (2015). *Miniquad Test Bench.* [Online]. Available at: `http://www.miniquadtestbench.com`, [2016, October 3].

Hartman, D., Landis, K., Mehrer, M., Moreno, S. and Kim, J. (2014). *Quadcopter Dynamic Modeling and Simulation (Quad-Sim).* [Online]. Available at: `https://github.com/dch33/Quad-Sim`, [2016, July 29].

Henderson, D. (1977). *Euler Angles, Quaternions, and Transformation Matrices.* NASA.

Hobden, A. (2015). *Quadcopters: Yaw.* [Online]. Available at: `https://hoverbear.org/2015/05/27/quadcopters-yaw/`, [2016, September 9].

Hou, Z. and Wang, Z. (2013). From model-based control to data-driven control: Survey, classification and perspective. *Information Sciences*, vol. 235, pp. 3–35.

Hou, Z. and Xu, J. (2009). On data-driven control theory: the state of the art and perspective. *Acta Automatica Sinica*, vol. 35, pp. 650–667.

Hovakimyan, N. and Cao, C. (2010). *L1 Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation.* Society for Industrial and Applied Mathematics.

Huang, C. and Liao, Y. (2015). Gui pid self-tuning system for quadcopters. In: *13th International Conference on Control, Automation, Robotics and Vision*, vol. 24, pp. 103–108. ISBN 978-969-670-025-8.

Hughes, P. (2004). *Spacecraft Attitude Dynamics*, chap. 2, pp. 22–23. Dover Publications.

Huynh, M., Zhao, W. and Xie, L. (2014). L1 adaptive control for quadcopter: Design and implementation. In: *13th International Conference on Control, Automation, Robotics and Vision.*

Invernizzi, D., Panizza, P., Riccardi, F., Frmentin, S. and Lovera, M. (2016). Data-driven attitude control law of a variablepitch quadrotor: a comparison study. In: *Automatic Federation of Automatic Control*, pp. 236–241.

Ioannou, P. and Fidan, B. (2006). *Adaptive Control Tutorial.* Society for Industrial and Applied Mathematics.

Komer, B. (2015). *Biologically Inspired Adaptive Control of Quadcopter Flight.* Masters, University of Waterloo.

Krossblade Aerospace (2016). *History of quadcopters and other multirotors.* [Online]. Available at: `http://www.krossblade.com/history-of-quadcopters-and-multirotors/`, [2017, February 9].

Lock, J., Smit, W. and Treurnicht, J. (2015). An investigation into multi-dimensional prediction models to estimate the pose error of a quadcopter in a csp plant setting. In: *AIP Conference Proceedings.* AIP Publishing.

Mathworks (2017*a*). *Quaternion Multiplication.* [Online]. Available at: `https://www.mathworks.com/help/aeroblks/quaternionmultiplication.html`, [2017, February 5].

Mathworks (2017*b*). *Rotation Matrix.* [Online]. Available at: `http://mathworld.wolfram.com/RotationMatrix.html`, [2017, February 5].

Michini, B. (2007). *Modelling and Adaptive Control of Indoor Unmanned Aerial Vehicles.* Masters, Massachusetts Institute of Technology.

Michini, B. and How, J. (2009). L1 adaptive control for indoor autonomous vehicles: Design process and flight testing. In: *AIAA Guidance, Navigation, and Control Conference.* American Institute of Aeronautics and Astronautics.

Mohammadi, M. and Shahri, A. (2013). Adaptive nonlinear stabilization control for a quadrotor uav: Theory, simulation and experimentation. *Journal of Intelligent and Robotic Systems.* DOI 10.1007/s10846-013-9813-y.

My First Drone (2016). *Best flight controllers and why.* [Online]. Available at: `http://myfirstdrone.com/tutorials/buying-guides/best-flight-controllers/`, [2016, July 25].

Narendra, K. and Annaswamy, A. (2012). *Stable Adaptive Systems.* Dover Publications.

Nicol, C. (2008). Robust neural network control of a quadrotor helicopter. In: *Canadian Conference on Electrical and Computer Engineering*, pp. 1233–1237. doi 10.1109/CCECE.2008.4564736.

Noureldin, A., Karamat, T. and Georgy, J. (2013). *Fundamentals of Inertial Navigation, Satellite-based Positioning and their Integration.* Springer.

Omidshafiei, S. (2013). Reinforcement learning-based quadcopter control. Available at: `http://www.mit.edu/~shayegan/files/quadcopter\_control\_rl.pdf`.

Parks, P. (1966). Lyapunov redesign of model reference adaptive control systems. *IEEE Trans. Automatimation Control*, vol. 11, no. 3, pp. pp. 362–367. doi 10.1109/TAC.1966.1098361.

Petersen, K. and Pedersen, M. (2012). *The Matrix Cookbook.* IMM.

Praly, L. (1984). Robust model reference adaptive controllers, part 1. In: *The 23rd IEEE Conference on Decision and Control.*

PX4 (2016*a*). *Development Environment.* [Online]. Available at: `https://dev.px4.io/en/setup/building\_px4.html`, [2017, June 16].

PX4 (2016*b*). *Introducing Copter.* [Online]. Available at: `http://px4.io/`, [2017, August 12].

PX4 (2017*a*). *Architectutal Overview.* [Online]. Available at: `https://dev.px4.io/en/concept/architecture.html`, [2017, August 25].

PX4 (2017*b*). *Flight modes.* [Online]. Available at: `https://dev.px4.io/en/concept/flight\_modes.html`, [2016, November 16].

PX4 (2017*c*). *Simulation.* [Online]. Available at: `https://dev.px4.io/en/simulation/`, [2016, September 26].

PX4 Autopilot (2016*a*). *Atmospheric Hardware.* [Online]. Available at: `https://pixhawk.org`, [2017, August 29].

PX4 Autopilot (2016*b*). *Atmospheric Pressure.* [Online]. Available at: `https://pixhawk.org/dev/know-how/atmospheric\_pressure`, [2017, August 29].

Qgroundcontrol (2016). *Qgroundcontrol.* [Online]. Available at: `http://qgroundcontrol.com/`, [2016, September 12].

QtCreator (2016). *QtCreator.* [Online]. Available at: `https://www.qt.io/ide/`, [2016, September 26].

Quadcopter Arena (2017). *The history of drones and quadcopters.* [Online]. Available at: `http://quadcopterarena.com/the-history-of-drones-and-quadcopters/`, [2017, February 9]. Accessed: 2017-08-12.

Rotorbench (nd). *Emax 1806 2280kv.* [Online]. Available at: `http://rotorbench.com/`, [2016, October 3].

Shoemake, K. (nd). Quaternions. Available at `http://www.cs.ucr.edu/~vbz/resources/quatut.pdf`.

Slotine, J. and Li, W. (1991). *Applied Nonlinear Control.* Prentice-Hall.

STERG (2017). *About.* [Online]. Available at: `http://sterg.sun.ac.za/about/`, [2017, September 2].

United States Environmental Protection Agency (2017). *Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990-2015.* [Online]. Available at: `https://www.epa.gov/ghgemissions/inventory-us-greenhouse-gas-emissions-and-sinks`, [2017, August 1].

Wang, J. (2015). *Novel Control Approaches to Quadrotors Inspired by Dynamic Inversion and Backstepping.* Doctoral, University of Munich.

Yilmaz, E. (2014). *Adaptive Robust Attitude Controller Design for a Quadrotor Platform.* Masters, Middle East Technical University.