Multirotor Design Optimization: The Mechatronic Approach

Øyvind Magnussen

Multirotor Design Optimization: The Mechatronic Approach

DOCTOR OF PHILOSOPHY AT THE FACULTY OF ENGINEERING AND SCIENCE, SPECIALIZATION IN MECHATRONICS

University of Agder

Faculty of Engineering and Science
2015

Doctoral Dissertation by the University of Agder 107
ISBN: 978-82-7117-793-5
ISSN: 1504-9272

Printed in the Printing Office, University of Agder
Kristiansand

”It’s not about moving mountains,
it’s about making progress”

Geir Hovland, 2011

Acknowledgements

It was an honour being a PhD student at the University of Agder, which in my opinion is the most up-and-coming university in Norway at the moment. It is filled with wisdom and experience, a superb laboratory and a pleasant staff group. Among those, the first person I want to thank is my principal supervisor Professor Geir Hovland who has been a perfect mentor for this project. His theoretical skills and determinism have been highly appreciated.

I will also like to thank my second supervisor Morten Ottestad. Morten has had exceptional ides and he has been a driving force for the project.

There has been a lot of experimental work with my PhD and the guys at the lab have been helping a lot. I would like to thank Eivind, Roy and Kalle for the help during my three years.

Also, I would like to thank my colleagues, and in particular those from my office, Ilya and Yulin and especially Knut and Magnus. We have discussed a lot of interesting topics (and some with no meaning). Thanks to you guys those three years have been among my best!

Finally I will thank my family for the support the last month of writing the thesis. It has been many late evenings, and I am very happy you have been there for me.

Øyvind Magnussen
Grimstad, Norway
May 2015

Abstract

Multirotors such as the more famous quadcopter have been a favoured research object the last years. It is widely used as a flying platform for the hobby enthusiasts, but recently also used more and more by the industry. The multirotor has complex dynamics and requires sensors and a control system in order to fly. To get the desired flight characteristics batteries, motor and the propeller have to be chosen wisely as different combinations create different properties. The usual design approach is to test different combinations of motors and propellers, and based on experience select components that will be closest to the desired flight properties.

This thesis presents an optimization method that calculates what hardware to use in order to get closest to the demanded properties. The method will only select from a given database, hence not returning a diameter and pitch of a propeller that are not available. A wide range of criteria can be optimized, examples are dynamics of the motor/propeller, flight dynamics, flight time, payload etc. The optimization routine will also calculate if the better choice is a quadcopter with four propellers, a hexacopter with six or an octocopter with eight propellers.

The optimization is not trivial due to the non-linear characteristics of the propeller. A lot of experimental work was done to test the response of the propeller, both for acceleration and deceleration. Theory and experimental work show that the thrust response of the propeller can be more or less equal if the electronic speed controller controls the motor in a special mode. This mode also makes the response of the motor faster than normal.

The new design is tested with a new approach for attitude estimation, and a

controller operating directly with the result of the estimator. Most of the multirotors use a microcontroller with limited resources as the control system, hence the attitude and controller were designed specifically without time consuming trigonometric functions such as the sine and cosine.

Overall, the methods and results presented in this thesis will aid the engineer when designing a multirotor system consisting of control system, mechanical frame, battery, actuators and propellers.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Unmanned aerial vehicles (UAVs) in general and multirotors such as quadcopters in particular have received a tremendous amount of attention in recent years, both in academia and in industry. A multitude of concepts, designs and solutions have appeared. Most of these designs and solutions have been specialised towards a specific application or a specific study of interest. For example, a multirotor may be designed for agility and fast system dynamics without consideration of maximum allowed flying time due to the battery capacity.

In this thesis *the mechatronic approach* has been introduced to the design of multirotors. The mechatronic way is a multi-disciplinary approach combining possibilities and benefits from the mechanical, electrical/electronics and control engineering disciplines. Whereas a designer in a specific engineering discipline may take the designs related to other disciplines as given, a mechatronic design engineer may modify the design in one discipline to achieve simplicity or advantages in another discipline. One concrete example of the mechatronic way presented in this thesis is the redesign of the Electronic Speed Controller (ESC). With the standard firmware of the ESC, the system dynamics often becomes highly non-linear and a control engineer is forced to apply techniques from advanced control theory. However, with a redesign of the switching logic in the ESC, it is possible to linearise the system dynamics and hence make the control design easier and more robust.

Existing UAV designs are typically not optimized against a set of common criteria and constraints. Examples of criteria are to minimise component cost and maximise flying time, while examples of constraints are a lifting capacity of at least 1kg and actuator acceleration of at least $1 rad/s^2$. Moreover, the designs are normally constrained to use a limited, discrete set of existing components and sub-systems as specified in suppliers' datasheets. Hence, a multi-disciplinary approach to design optimization using only a discrete set of available components would add significant value to the UAV community.

1.2 Organization of the Dissertation

This thesis is a collection of papers, where the published papers appear in the appendices. A total of 4 papers have been approved, published and presented at conferences. The last appendix, Paper E, is presented in a journal.

Chapter 1 gives a short introduction to the concepts of multirotors, their basic design and operating principles. A literature review is then presented regarding topics relevant for this thesis, and what topics are not covered in literature. Then the objectives of the dissertation are discussed.

Chapter 2 presents the theory used in the appended papers. It starts with coordinate systems, which are used throughout the thesis. Then theory about the hardware and sensors relevant for multirotors are presented, followed by modelling of the multirotor. The last part is the design optimization. The basic theory is presented and an example showing how a mixed integer programming model is formulated.

Chapter 3 presents the conclusion and future work.

1.3 Multirotor Basics

A multirotor is a drone where the motors are pointing in the same direction. Some alternatives with an odd number of rotors, typically 3, exist but are not covered in this thesis. This thesis is only focused on an even number of motor pairs as illustrated in Fig. 1.1.

The multirotor has 6 degrees of freedom, translation along three axes, X, Y

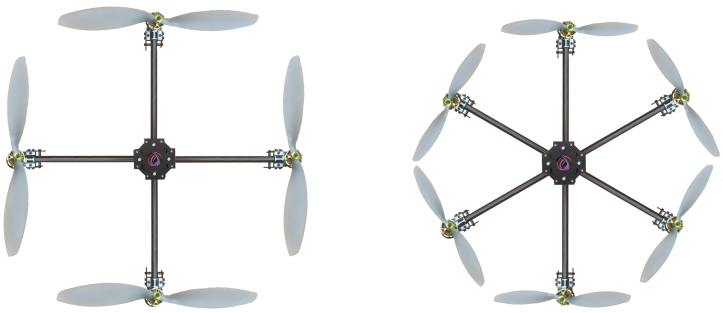
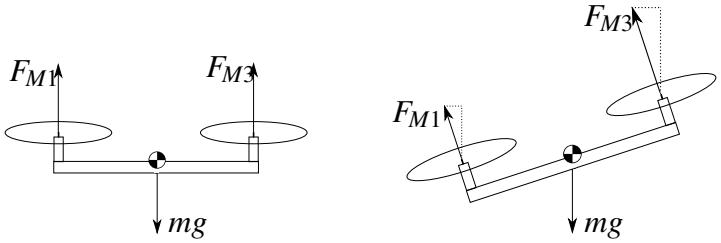


Figure 1.1: *Two typical designs of a multirotor, a quadcopter and a hexacopter.*

and Z and rotation, roll (ϕ), pitch (θ), yaw (ψ), about the same three axes. The multirotor only actuates four of the six degrees of freedom, that is roll, pitch, yaw and translation along the local Z axis thus the multirotor is under-actuated. The unactuated degrees of freedom can however be controlled by manipulating the actuated degrees of freedom giving the multirotor full manoeuvrability in 3D space. To move horizontally the multirotor must differentiate the thrust of one motor pair, or a combination of motor pairs, to tilt towards the desired direction, Fig. 1.2. Then the thrust must be increased to maintain altitude, and the multirotor will move along the desired direction. This motion applies for the roll and pitch angle.



a) The multirotor is hovering b) The multirotor is pitched.

Figure 1.2: *Example of how to actuate one of the last two degrees of freedom. In a) the two forces are of equal length resulting in no moment around the centre of gravity point. In b) the force F_{M3} is larger than F_{M1} resulting in a moment around centre of gravity and the multirotor pitches positively. This results in a positive force along the body X-axis.*

The propeller pairs of the multirotor are mounted in either clockwise (negative) or counter-clockwise (positive) rotation. The positive rotation results in a negative

torque and vice versa. In order to rotate with a positive rotation around the local z-axis (yaw motion), the net positive torque must exceed the negative. Increasing the revolutions per minute (RPM) of the propellers with positive torque while reducing the RPM of the negative ones, results in a positive net torque while keeping the thrust constant, shown in Fig. 1.3.

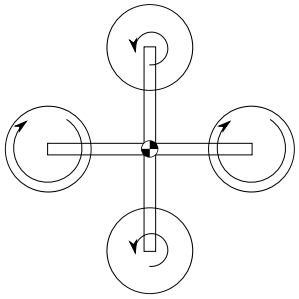


Figure 1.3: *Yaw motion induced by increasing the propeller RPM on the two clockwise rotating propellers. The result is a counter-clockwise rotation of the multirotor frame.*

1.4 Literature Review

1.4.1 Sensors and Attitude Estimation

Typical sensors needed to stabilize and control the multicopter are gyroscopes and accelerometers. The gyroscope provides angular rates in three dimensions, and the accelerometer acceleration in three dimensions, known as an inertial measurement unit (IMU). Those sensors are usually packed in a small micro chip, and is commercially available off-the-shelf. The sensors are used in a lot of electronic devices, such as mobile phones and remotes for TVs and consoles. The wide usage and high production rates are two of the reasons the sensors are becoming smaller (a few mm in size) and cheaper (a few dollars) with increased performance. Magnetometers are usually used to measure the multirotor heading. Magnetometers are also small, cheap and a three axis magnetometer is packed in a small chip.

Some high end IMUs also provide the attitude angles together with the sensor data. The angles are a result of filtering the sensor data into one combined attitude. Filtering methods used in the high end components are not always publicly known. However the manufacturer Microstrain is using different types of Kalman Filters in their sensors. Attitude estimation of the multirotor requires sophisticated methods. The propellers induce a lot of vibration of the frame, and hence affecting the sensor measurements. Especially the accelerometer which is more sensitive to vibrations than the gyroscope. The magnetometer is not affected by the vibration, but reacts to ferrous metals. There are many published papers about attitude estimation of the multirotor. In [Mahony et al., 2005] a quaternion based attitude estimation is proposed. The estimation is based on a complementary filter of the sensor data and computes an attitude quaternion. The estimation also calculates the gyroscope bias which is an issue for the gyroscopes. [Hall et al., 2008] presents a method using Multiplicative Extended Kalman Filter to estimate the attitude.

1.4.2 Control Systems

There has been a lot of research of different control strategies for the multirotors. In [Mokhtari et al., 2005] a non-linear system model is developed and a GH_∞ method is introduced to compute the controller which is mixed with robust feedback linearisation to control the non-linear system. Also less advanced techniques simplify the system and use the classical PID-controller [Li and Li, 2011]. Using PID-controller for the attitude is very common, and one of the most common controllers to use. Especially open source controllers use the PID, as they are easy to tune and provide good performance [Tayebi and McGilvray, 2006]. Position control of the multirotor is mainly done indoor with motion capture cameras which estimate the attitude and position of the multirotor very accurately at high update rates. Those systems make it possible to control a swarm of multirotors [Davis et al., 2013] and for acrobatic flights, [Brescianini et al., 2013], with an inverted pendulum on board, and throwing it from one multirotor to another.

1.4.3 Multirotor Design

The multirotor usually consists of 4, 6 or 8 actuators, where a fixed propeller is mounted on top of the motor, without any gearing system. The standard design of the multirotor has several drawbacks, especially when it comes to dynamic response. Changing the thrust of the fixed propeller requires an acceleration of the propeller which has a slow response. Instead of fixed propellers, variable pitched propellers have been proposed in [Cutler et al., 2011]. The variable pitched propellers run at high RPM regardless of the thrust, and vary the pitch of the propeller to change its thrust. With variable pitched propellers the dynamic response of the propeller is mainly limited to the speed of the actuator changing its pitch. Some variable pitched propellers also produce negative thrust, making inverted flights possible. In general the variable pitched propellers will increase the response significantly. The drawback however is the mechanical setup which is more complex than for the fixed propeller. The complex mechanical setup makes them a second choice for most applications. This paper will only discuss about a setup with fixed propellers.

To increase the dynamic performance of the actuators, it is important that the propellers are as stiff and light as possible. Increased stiffness reduces the propeller blade flapping. Blade flapping is an effect that occurs when the multirotor has a velocity relative to the air. The advancing blade has a different thrust than the retreating blade because of the differences in the relative air flow, the result is that the blade starts to flap up and down. This reduces the thrust of the propeller, and also induces roll and pitch moments on the propeller [Hoffmann et al., 2007]. With carbon propellers the stiffness is increased and the weight is reduced. The more costly carbon propellers are the preferred choice, but the selection is still limited in a market where price is an issue.

Using more than four actuators provides some system redundancy in case of an actuator fault. In [Freddi et al., 2014] a Thau observer is tested to provide estimation of the system states for the fault detection. In case of an actuator fault, it will then be possibly to compensate for it and avoid a crash. In [Sidea et al., 2014] a system with a variable number of actuators is presented. The work presented shows how the number of actuators can become a variable for the multirotor model and

the control system.

The selection of motors and propellers have significant impact of the multirotor properties. One configuration will make the multirotor fly large distances, while another will have a faster dynamic response. Most of the research of the multirotor topic have been attitude estimation and control systems for the hardware already built. To the author's knowledge there are no published papers regarding a design optimization of the multirotor aircraft.

1.5 Summary of Publications

The following papers are appended and are printed in their originally published state except for changes in format and minor errata.

1.5.1 Paper A - Calibration Procedure for an Inertial Measurement Unit Using a 6-Degree-of-Freedom Hexapod

Summary: The sensors are important for UAVs and especially multirotors in order to fly. There is a need to calibrate the sensors as un-calibrated sensors will not provide good enough measurement data. Calibration of the sensor is one issue, mounting and positioning it is another. If the sensor coordinate system is rotated relative to the onboard coordinate system, the readings will be faulty even if the sensor is calibrated. In this paper a calibration procedure is presented using a hexapod. The hexapod runs different motions while logging its own position and attitude. Readings from an IMU is compared with the "real ones" from the hexapod. From the different motions the algorithm calculates the bias, scaling, misalignment, and translational and rotational offset relative to the body frame, which is used to calibrate the sensor from where it is mounted.

Background and contribution: Difficulties to maintain a stable flight of a multirotor made it necessary to verify the sensor data. This approach was established and the sensor was calibrated. We also demonstrate a new application of the hexapod, and its wide usage.

This paper has been published as:

Øyvind Magnussen, Morten Ottestad and Geir Hovland. *International Conference on Unmanned Aircraft Systems (ICUAS).* Philadelphia, USA, June 12-15, 2012.

1.5.2 Paper B - Experimental Validation of a Quaternion-based Attitude Estimation with Direct Input to a Quadcopter Control System

Summary: Multirotors need a control system in order to fly. For automatic hover the control system has to estimate the attitude of the multirotor and feed the attitude in to the control loop. The control loop calculates new setpoints for the motor thrust and updates the motors accordingly. The microcontroller has limited processor and memory resources. This paper presents a new approach using quaternions to estimate the attitude completely without trigonometric functions. The estimator is compared with a high-end IMU which internally calculates the attitude data. The quaternion is directly used as a setpoint in the attitude controller which is able to maintain a stable flight of the multirotor.

Background and contribution: To gain experience of the multirotor, both theoretical and experimental, establishing a control system is an excellent approach. The system was tested in simulation before it was implemented on the multirotor. The paper presents a new attitude estimator, based on shortest rotation arc, and a simple method to implement the quaternion to the control system.

This paper has been published as:

Øyvind Magnussen, Morten Ottestad and Geir Hovland. *IEEE International Conference on Unmanned Aircraft Systems (ICUAS).* Atlanta, USA, May 28-31, 2013.

1.5.3 Paper C - Experimental Study on the Influence of Controller Firmware on Multirotor Actuator Dynamics

Summary: The multirotor control system updates the setpoint to the electronic speed controller (ESC) which controls the brushless DC motors. Updating the ESC with a high frequency rate is essential for stability. Standard ESCs are not designed for this rate and special ESCs for multirotors are needed. There are many control strategies for the ESC to use when controlling the motor. This paper presents a method that shows increased actuator dynamics compared to the standard method. The presented method also makes the response of increased and decreased thrust very similar. A linearised actuator model is then presented based on the theory and experimental data.

Background and contribution: From the previous paper the multirotor was able to maintain a stable flight. It was however discovered that increasing the thrust was much faster than decreasing the thrust. This extra level of complexity might be interesting to solve, but the best would be to eliminate the problem. This paper shows how this is done, and in addition increasing the response of the actuator.

This paper has been published as:

Øyvind Magnussen, Simon Kirby, Morten Ottestad and Geir Hovland. *IEEE International Symposium on Robotic and Sensors Environments (ROSE).* Timisoara, Romania, October 16-18, 2014.

1.5.4 Paper D - Multicopter UAV Design Optimization

Summary: Designing a multirotor is a complex task, especially if there are some performance needs. There are many different motors and propellers to chose from and each configuration will have different properties. This paper presents a method to calculate which hardware to select, based on a given set of datasheets for motors, propellers and batteries. The system is set up as a Mixed-Integer Linear Program (MILP) system and solved with the IBM CPLEX solver. For simplification the

solver is run one time for each number of actuators (four, six and eight). The system can optimize for different requirements such as flight time, actuator response, cost, power etc. Even though the system consists of many thousands equations, it only takes the solver a few seconds to get the optimal solution.

Background and contribution: With an accurate actuator model from the previous paper it was possible to establish a system to design an optimal multirotor from a given set of hardware and demands.

This paper has been published as:

Øyvind Magnussen, Geir Hovland and Morten Ottestad. *IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA).* Senigallia, Italy, September 10-12, 2014.

1.5.5 Paper E - Multicopter Design Optimization and Validation

Summary: Based on the work from the previous paper a validation of the optimization routine was needed. This paper presents a detailed method how to establish the optimization algorithm. Mathematical equations used are converted to equalities and inequalities using different rules, also described in this paper. The paper covers three cases to optimize, flight time with and without payload and actuator dynamics. The hardware chosen for each test was experimentally tested to verify the flight time and dynamics. The actuator response was tested with a step of $\pm 30\%$ of the force required to hover. This is a relative large step, but still the response of the motor system was quite similar to increase and decrease of the thrust.

Background and contribution: This paper completes the research of the multirotor design optimization. The paper covers the entire design optimization process and validates the calculations done with experimental testing. With this software tool the design procedure of the multirotors can change. The customer no longer has to ask what properties the multirotors have before buying, now the customer can come with demands.

This paper has been published as:

Øyvind Magnussen, Morten Ottestad and Geir Hovland. *Modeling, Identification and Control, Volume 36, Number 2, Pages 67-79* Publisher: Norwegian Society of Automatic Control.

Chapter 2

Research Methodology

2.1 Coordinate Systems

Two different coordinate systems define the position and orientation of the multirotor, the body frame and the navigation frame, Fig. 2.1. The body frame coordinate system denoted *b*, has a fixed position in the centre of gravity (CG) of the multirotor body. This coordinate system moves and rotates with the multirotor. The navigation frame denoted *n*, is in a fixed position relative to the earth. Both of the coordinate systems have positive Z-direction upwards and the right hand rule applies to both of the coordinate systems.

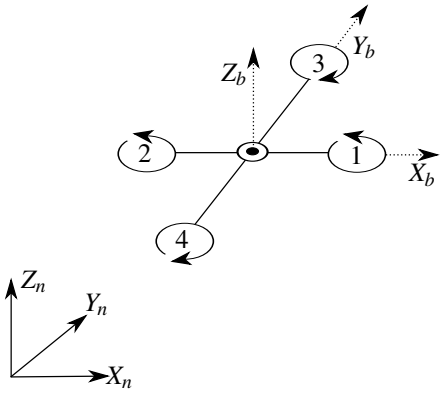


Figure 2.1: *Quadcopter coordinate systems, motor number and positive motor torque.*

Opposite to an aeroplane the multirotor does not have a distinct forward di-

rection due to its symmetric design. For simplification of equation of motion the forward direction of the multirotor is defined to be along positive X-axis.

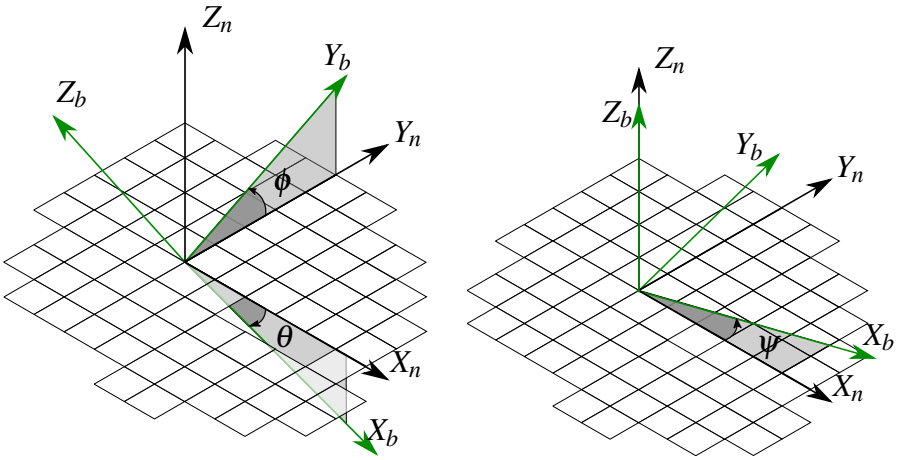
2.1.1 Euler Angles

Leonard Euler introduced a method to describe an orientation of a rigid body with three angles, known as the Euler angles. By three successive rotations around different axes of a body any orientation can be achieved [Dunn, 2011]. Euler described the method by three rotations around two different axes, Z-Y-Z. Other methods use three different rotation axes, this yields a total of twelve different rotation sequences. Since only one rotation sequence should be selected and used at a time, confusion is often a part of the method. One of the primary advantages of the Euler method is that the connection between the transformation matrix and the physical motion between frames of reference is readily envisioned and written. Euler angles suffer from singularity when the second angle is zero.

2.1.2 Tait-Bryan Angles

Tait-Bryan angles are a modified version of the Euler angles, where the rotation occurs once around each axis in the order X-Y-Z in the navigation frame. Rotating around X-Y-Z axes in the navigation frame yields the same angles as when rotating around Z-Y-X in the body frame. The three resulting angles for the X-Y-Z rotation sequence are known as roll (ϕ), pitch (θ) and yaw (ψ), Fig. 2.2. The roll angle is the angle between Y_b and the X_nY_n-plane, pitch is the angle between X_b and the X_nY_n-plane and the yaw is the angle between X_b and the X_nZ_n-plane. The Tait-Bryan angles suffer from singularity when pitch is $\pm\pi/2\,rad$. This is one of the reasons *quaternions* are often used to describe the orientation of a rigid body.

When using Tait-Bryan or Euler angles as attitude representation there is a risk of gimbal lock. Gimbal lock occurs when two or more axes are aligned and it results in a loss of one of the degrees of freedom. An aeroplane experiences gimbal lock when pitched 90 degrees. At this position the roll axis is aligned with the yaw axis, and the plane has lost a degree of freedom.



Positive roll and pitch angle applied

Positive yaw angle applied

Figure 2.2: *Roll, pitch and yaw angles. Note that the angles are not the angle from axis-to-axis but from the navigation plane to the body axis.*

2.1.3 Vector Rotations

The propeller forces are fixed in the body frame coordinate system, while the gravity is fixed in the navigation frame. To calculate the required propeller thrust to compensate for the gravity the propeller thrust vector as seen from the body coordinate system must be rotated to the navigation frame with a 3×3 rotation matrix and compared with the gravity vector. To calculate the rotation matrix the relationship for a single rotation around each of the three axes must be established. A positive rotation around the X-axis with the angle ϕ shown in Fig. 2.3. The black unit vectors represents the axes of the navigation frame, the green vectors are the body unit vectors and the blue vectors are the body vector represented by the navigation frame vectors. Applying the same for a rotation only around the Y axis and then the Z axis

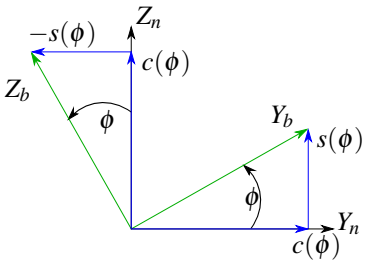


Figure 2.3: *Rotation only around the X-axis.*

results in three matrices [Sciavicco, 1996].

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c(\phi) & -s(\phi) \\ 0 & s(\phi) & c(\phi) \end{bmatrix} \quad (2.1)$$

$$R_y = \begin{bmatrix} c(\theta) & 0 & s(\theta) \\ 0 & 1 & 0 \\ -s(\theta) & 0 & c(\theta) \end{bmatrix} \quad (2.2)$$

$$R_z = \begin{bmatrix} c(\psi) & -s(\psi) & 0 \\ s(\psi) & c(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.3)$$

The rotation matrix, R_b^n, rotating a body vector to a representation in the navigation frame is derived from the multiplication of the three matrices R_x, R_y and R_z.

$$R_b^n = R_x R_y R_z = \begin{bmatrix} c(\theta)c(\psi) & -c(\theta)s(\psi) & s(\theta) \\ c(\phi)s(\psi)+c(\psi)s(\phi)s(\theta) & c(\phi)c(\psi)-s(\phi)s(\theta)s(\psi) & -c(\theta)s(\phi) \\ s(\phi)s(\psi)-c(\phi)c(\psi)s(\theta) & c(\psi)s(\phi)+c(\phi)s(\theta)s(\psi) & c(\phi)c(\theta) \end{bmatrix} \quad (2.4)$$

where 's' and 'c' represent the sine- and cosine-function respectively. The rotation matrix R_b^n rotates any vector represented in the body frame to the navigation frame. Multiplying a vector in the navigation frame with the transposed rotation matrix R_n^b, which for the rotation matrix is the same as the inverse, the vector is represented in the body frame coordinate system.

$$R_n^b = (R_b^n)^{-1} \quad (2.5)$$

The thrust F_{th} of the multirotor is fixed in the Z-direction of the body frame $[0\ 0\ F_{th}]^T$. Multiplying the thrust vector with the rotation matrix, the thrust for a given rotation is given as forces in the navigation frame:

$$F_{th,n} = R_b^n \begin{bmatrix} 0 \\ 0 \\ F_{th} \end{bmatrix} = F_{Th} \begin{bmatrix} s(\theta) \\ -c(\theta)s(\phi) \\ c(\phi)c(\theta) \end{bmatrix} \quad (2.6)$$

2.1.4 Quaternions

Another way of describing the orientation of a rigid body is by use of quaternions. Quaternions are a set of complex numbers describing an orientation in 3D-space. They were first described by Sir William Rowan Hamilton in 1843. Since the quaternions do not involve any kind of singularity when defining angles, and that they also might reduce the dynamic equations, the quaternions are the desired method for the multirotor. Tait-Bryan angles are defined with three rotations around three fixed axes, either in the body frame, or the navigation frame. The quaternions however describe one rotation around three non-fixed axes, but fixed for a specific rotation. The quaternion is a vector in a four dimensional hyper plane. It consists of a scalar (s) and a vector part of imaginary numbers (aj, bk, cl) where a,b,c describe the magnitude of the vector and j,k,l describe its direction. The imaginary numbers span over a 3 dimensional vector space V, which is perpendicular to the real part s. When quaternions are used for rotations it is the *unit* quaternion whose length equals one, $(s^2+a^2+b^2+c^2=1)$, which is used. The rotations described are rotations with three DOF (Degrees Of Freedom). Constraining the length of the four DOF quaternion to a unit quaternion reduces the number of freedoms to three. Every quaternion referred to in this document is a unit quaternion, if not stated otherwise, written as ”quaternion” for simplicity.

Drawing a four dimensional plane on a two dimensional paper is not possible, but it can be visualized as in Fig. 2.4

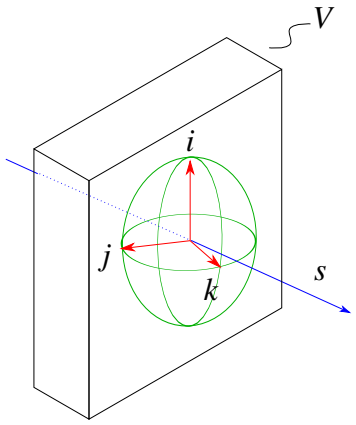


Figure 2.4: *2D representation of a 4D quaternion hyperplane with four perpendicular axes, s,j,k,l, and the vector space V.*

The complex conjugate of a quaternion is the same quaternion with opposite signs of the vector part. When multiplying quaternions the laws presented in Table 2.1 apply.

Table 2.1: *Rules for multiplication of the scalar and imaginary numbers, [Vince, 2010]*

	s	j	k	l
s	s^2	js	ks	ls
j	sj	-1	$-l$	k
k	sk	l	-1	$-j$
l	sl	$-k$	j	-1

The quaternion is to a three dimensional space the same as complex numbers are to a two dimensional complex plane. To understand quaternions it is important to understand the two dimensional complex mathematics. A complex vector $z = a + bi$ whose real part is a and imaginary part is b is illustrated in Fig. 2.5. Multiplying the z-vector with another complex unit vector whose length equals to one only adds a rotation to the z-vector. In Fig. 2.5 the z-vector is multiplied with itself and hence rotated to the new vector z^2. The commutative property of complex numbers makes the product the same regardless of the multiplication order. This

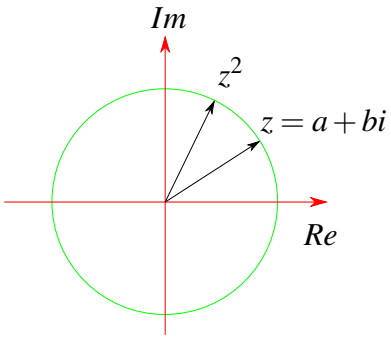


Figure 2.5: *2D complex plane with real part Re and imaginary part Im.*

can also be visualized for the quaternions. The four dimensional space is visualized as two 2-dimensional planes. The first plane spans from the real part $s = 1$ and the unit vector j. The second plane spans from the unit vectors k and l, illustrated in Fig. 2.6.

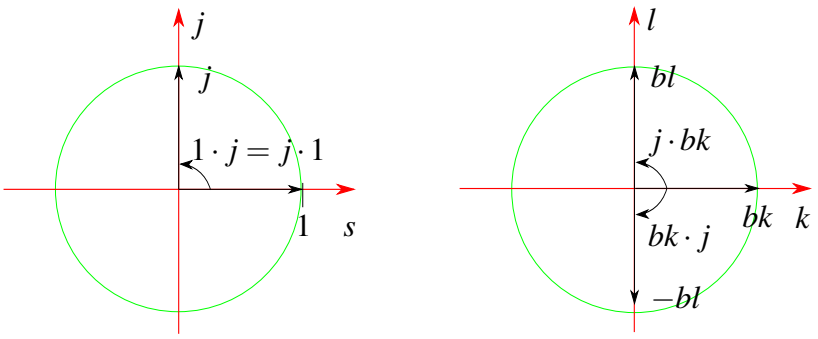


Figure 2.6: *The 4D quaternion space visualized as two 2D planes.*

In Fig. 2.6 it is shown that either a left of right multiplication of j for the sj-plane rotates the vector (1) a half turn (to j). Multiplying with the complex conjugate ($-j$) the vector is rotated a negative half turn (to $-j$). For the kl-plane a left multiplication of j to the bk-vector is also a half turn, but a right multiplication is a negative half turn, and opposite when multiplying with the complex conjugate. A multiplication of one rotation (j), gives two rotations, one rotation in the sj-plane, and one rotation in the kl-plane. To rotate only in one plane, two multiplications are needed, left multiplication with the rotation vector (j), and right multiplication with the conjugate of the vector ($-j$):

$$(j)(s+aj+bk+cl)(-j) = (-a+sj-ck+bl)(-j) = s+aj-bk-cl \qquad (2.7)$$

The result is a 180 degrees rotation, which is twice the rotation angle applied. The rotation is illustrated in Fig. 2.7

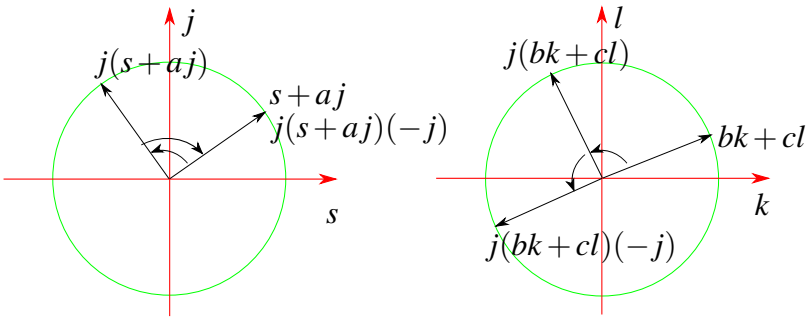


Figure 2.7: *Quaternion rotation in two planes. There is no rotation in the sj-plane, and 180 degrees rotation in the kl-plane.*

2.1.5 Quaternion Rotation of Vectors

To rotate a three dimensional vector A to a new vector A', it needs to be left-multiplied with the quaternion q representing the rotation and right-multiplied with the complex conjugate of the same quaternion $\bar{q}$. Setting the scalar part to zero, the three dimensional vector A becomes the four dimensional vector $B = [0;A]$ and can be rotated with a quaternion. The new rotated vector is the vector part of the multiplication, neglecting the scalar part.

$$B' = qB\bar{q} = q[0;A]^T\bar{q} \tag{2.8}$$

Often the symbol "$\otimes$" is used to represent the quaternion multiplication of vectors being multiplied with a quaternion without adding a leading zero to the vector being rotated. The operation is performed as shown in equation (2.8), but the notation is simplified:

$$A' = q \otimes A \otimes \bar{q} \tag{2.9}$$

2.1.6 Quaternion Operation Rules

Quaternion addition and subtraction is commutative. However the rules from Table 2.1 make the general quaternion products non-commutative. There are situations where the commutative of two quaternions exists, but that is not the general case.

The commutative, non-commutative and associativity properties of the quaternions are shown for the given three quaternions q_1, q_2 q_3:

$$q_1 + q_2 = q_2 + q_1 \tag{2.10}$$
$$(q_1 + q_2) + q_3 = q_1 + (q_2 + q_3) \tag{2.11}$$
$$q_1 \cdot q_2 \neq q_2 \cdot q_1 \tag{2.12}$$
$$(q_1 \cdot q_2) \cdot q_3 = q_1 \cdot (q_2 \cdot q_3) \tag{2.13}$$

The two quaternions q_1 and q_2 with its vector components are represented

$$q_1 = [s_1 + v_1] = [s_1 + a_1 j + b_1 k + c_1 l] \tag{2.14}$$

$$q_2 = [s_2 + v_2] = [s_2 + a_2 j + b_2 k + c_2 l] \tag{2.15}$$

By using the laws from Table 2.1 the product $q_1 q_2$ is given by:

$$\begin{aligned} q_1 q_2 =& (s_1 s_2 - a_1 a_2 - b_1 b_2 - c_1 c_2) + (s_1 a_2 + s_2 a_1 + b_1 c_2 - b_2 c_1) j \\ & (s_1 b_2 + s_2 b_1 + c_1 a_2 - c_2 a_1) k + (s_1 c_2 + s_2 c_1 + a_1 b_2 - a_2 b_1) l] \end{aligned} \tag{2.16}$$

$$q_1 q_2 = [(s_1 s_2 - v_1 \cdot v_2), (s_1 v_2 + s_2 v_1 + v_1 \times v_2)] \tag{2.17}$$

where $(s_1 s_2 - v_1 \cdot v_2)$ is the scalar and $(s_1 v_2 + s_2 v_1 + v_1 \times v_2)$ is the vector.

It can also be shown that a unit quaternion can be derived from a unit vector u represented in the navigation space and the angle θ to rotate:

$$q = \begin{bmatrix} \cos(\frac{\theta}{2}) \\ u_1 \sin(\frac{\theta}{2}) \\ u_2 \sin(\frac{\theta}{2}) \\ u_3 \sin(\frac{\theta}{2}) \end{bmatrix} \tag{2.18}$$

2.1.7 Quaternion Derivative

From the angular velocities $\omega = [\omega_x \ \omega_y \ \omega_z]^T$ in the body frame it can be proven that the time derivative of the quaternion, and its Euler integration are defined as [Kelly, 2013]:

$$\dot{q}_k = \frac{1}{2} \begin{bmatrix} 0 \\ \omega \end{bmatrix} q_k \tag{2.19}$$

$$q_{k+1} = q_k + \Delta t \cdot \dot{q}_k \tag{2.20}$$

Note that the derivative of the quaternion is not a unit quaternion, depending on the angular velocity. The same is the case of the result of the time integration due to the length of $\dot{q}_k$ and q must be normalized to represent the orientation.

2.1.8 Quaternion from Two Vectors

The quaternion represents a rotation from one coordinate system to another. The quaternion consists of a scalar part, telling how much to rotate, and a vector part defining the axis to rotate about. A quaternion from one vector (a_1) to another (a_2) is not unique as there is at least two rotations, positive and negative rotation. However the shortest rotation arc is unique, except for the case where $a_2 = -a_1$. This exception is a 180 degrees rotation around any axis perpendicular to a_1. The quaternion representing the rotation from the vector a_1 to a_2 is illustrated in Fig. 2.8.

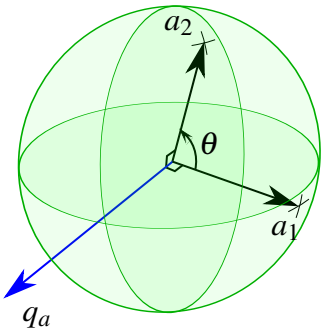


Figure 2.8: *The vector part of the quaternion q_a rotating a_1 to a_2 is perpendicular to the two vectors.*

The dot product ($\bullet$) and cross product ($\times$) of the vectors a_1 and a_2 are defined as:

$$a_1 \bullet a_2 = ||a_1|| \cdot ||a_2|| \cdot \cos(\theta) \tag{2.21}$$

$$a_1 \times a_2 = ||a_1|| \cdot ||a_2|| \cdot \sin(\theta) \tag{2.22}$$

where $||\ldots||$ denotes the magnitude of a vector.

The rotation angle in equation (2.18) is calculated from the dot product of the vectors, and the vector part is the unit vector of the cross product.

$$q_a^* = \left[\cos\left(\frac{\theta}{2}\right), \sin\left(\frac{\theta}{2}\right) \cdot \frac{a_1 \times a_2}{||a_1 \times a_2||}\right] \tag{2.23}$$

From calculus it is shown that

$$\sin(\theta) = 2\sin\left(\frac{\theta}{2}\right)\cdot\cos\left(\frac{\theta}{2}\right) \tag{2.24}$$

$$||a_1 \times a_2|| = ||a_1|| \cdot ||a_2|| \cdot |\sin(\theta)| \tag{2.25}$$

The expression from equation (2.23) becomes

$$q_a^* = \left[\cos\left(\frac{\theta}{2}\right), \frac{a_1 \times a_2}{2 \cdot ||a_1|| \cdot ||a_2|| \cdot |\cos\left(\frac{\theta}{2}\right)|}\right] \tag{2.26}$$

The rotation is always positive, hence $|\cos(\theta)| = \cos(\theta)$. Multiplying the quaternion with the denominator of the vector part, the quaternion becomes

$$q_a^* = \left[\cos\left(\frac{\theta}{2}\right)^2 \cdot 2 \cdot ||a_1|| \cdot ||a_2||, a_1 \times a_2\right] \tag{2.27}$$

The cosine function in the scalar part can be re-written by applying that

$$\cos\left(\frac{\theta}{2}\right) = \sqrt{\frac{1+\cos(\theta)}{2}} \tag{2.28}$$

The scalar part is then a constant 1 plus the dot product of the two vectors.

$$q_a^* = [1 + a_1 \bullet a_2, a_1 \times a_2] \tag{2.29}$$

Depending on the vectors a_1 and a_2 the resulting quaternion is not ensured to be a unit quaternion used to represent a rotation. The quaternion is therefore normalized.

$$q_a = \frac{q_a^*}{||q_a^*||} \tag{2.30}$$

2.2 Sensors and Hardware

2.2.1 Drive System Model

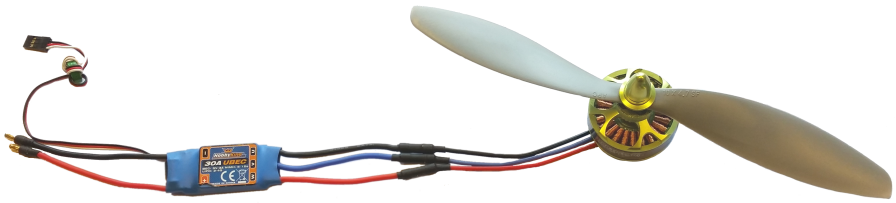


Figure 2.9: *Typical multicopter actuator system, consisting of a propeller, motor and electronic speed controller.*

2.2.1.1 Propellers

The propeller is like a rotating wing producing thrust and drag in the same manner. The shape of the cross section of the propeller is known as an airfoil, Fig. 2.10. The angle of attack of the propeller, known as pitch, is the angle between the centre line of the airfoil (chord) and the direction of the wind stream. The lift vector F_{Th} (which for a propeller is the same as thrust) of the airfoil is perpendicular to the direction of the wind stream, and the drag vector F_d is parallel to it. The aerodynamics of

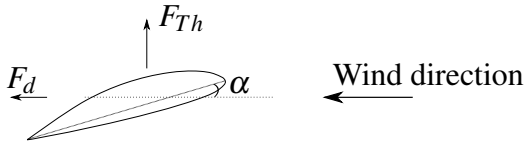


Figure 2.10: *An airfoil producing lift F_{th} and drag F_d.*

the propeller is very complex, especially when also considering that the tip of the propeller rotates faster than the propeller at its centre. This is why most of the propellers are curved, chagning its pitch from the centre to tip.

To understand how the airfoil generates lift, Bernoulli's principle can be used. Bernoulli's principle states that the pressure of a fluid (air in this case) decreases if the velocity of the fluid increases, and vice versa. The airfoil is specially shaped to increase the velocity of the air on the top side, hence decreasing the pressure. The higher pressure below the airfoil presses it upwards. In Bernoulli's equation the

pressure is a square function of the wind velocity. From this the thrust, torque and power of a propeller are described as [Phillips, 2004]:

$$F_p = C_t \rho n^2 D^4 \tag{2.31}$$

$$T_p = C_l \rho n^2 D^5 \tag{2.32}$$

$$P_p = C_p \rho n^3 D^5 \tag{2.33}$$

where C_t, C_l and C_p are the propeller constants for a given propeller and velocity.

2.2.1.2 DC Motor Operating Principle

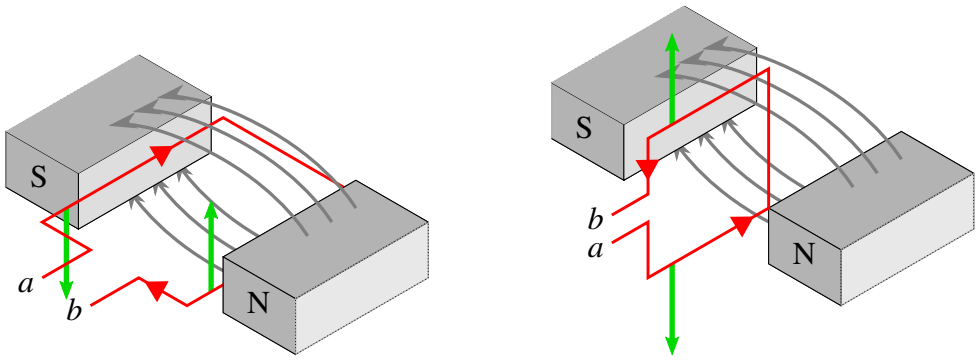
The standard design of the multirotor consisting of a frame with a fixed propeller on-top of a motor on each arm is very simple and robust, especially compared to the driving mechanism of a swash-plate helicopter. This is also one of the reasons that the frame with fixed propellers is the main design of the multirotor. However some designs with pitched propellers exist. This design is not that simple and robust, but the dynamics and the manoeuvrability of the system can be increased. This thesis is focused on the direct drive from the DC motor to the propeller.

An electrical motor is converting the electrical power from the battery into mechanical power to spin the propeller. The operating principle of the motor, illustrated in Fig. 2.11, is based on the principle that when a conductor carrying a current is placed in a magnetic field, it experiences a mechanical force whose direction is given by Fleming's left hand rule [Whitaker, 2005], and a magnitude F_{DC} given by:

$$F_{DC} = BIl \tag{2.34}$$

where F_{DC} is the force in Newton, B is the magnetic field in Wb/m^2, I is the current in Ampere and l is the length of the coil in meters. Note that on the figure the two arrows are pointing in the opposite direction. This is because the current has changed direction through the magnetic field.

When the motor in Fig. 2.11 has rotated 90 degrees from its original position the distance from the conductor to the north and south pole are equal, and the moment trying to rotate the coil is zero. If the coil had any inertia and velocity when reaching



a) Initial position, full moment about the axis b) 90 degrees rotation, zero moment about the axis.

Figure 2.11: *Fleming's left hand rule applied where the index finger is placed in the direction of the current in the coil (red wire), the middle finger placed in the direction of the magnetic force (grey from north (N) to south (S)). The direction of the mechanical force (green arrow) is given by the direction of the thumb.*

this point, the force would "push" the coil back to steady state position at 90 degrees (no moment), or oscillate around this position.

If the coil in Fig. 2.11 were to spin continuously, the current has to change direction once the coil passes the 90 degrees point. This results in a change of direction of the forces illustrated in the figure, and the motor will continue to spin. It is also important to note that for this to work, the motor has to pass the 90 degrees point only with kinetic energy, as the electro-mechanical energy is zero around this point.

This change of current direction is known as "commutation". There are two methods of motor commutation, with brushes or without brushes (brushless motor BLDC). The brushed DC motor is using the simplest operating principle of the two. It is using a purely mechanical commutation principle shown in Fig. 2.12. The coil is forced to spin in the direction shown by the arrows on the commutator. Once the coil passes the 90 degrees turn the current changes direction in the coil. The change of current direction changes direction of the force and the coil experiences a moment that spins the motor continuously.

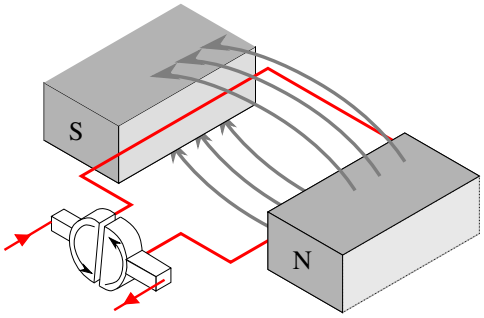


Figure 2.12: *Mechanical commutation of the DC motor using brushes.*

2.2.1.3 BLDC Operating Principle

The operating principle of the BLCD is similar to the brushed one, it needs to change the direction of the current in order to rotate. As opposed to the mechanical commutation of the brushed DC motor, the BLDC's commutation is electrical. The windings are wound around a ferro core and when the current flows through the wire, it becomes an electromagnet, illustrated in Fig. 2.13. From the right hand rule, when the fingers curve in the direction of the current the magnetic north is in the direction of the thumb. The electromagnet also works the other way around, if a non-energized coil is put in a magnetic field, it will induce a current to prevent the change, resulting in a voltage drop across the internal resistance of the coil. Once the coil is fully magnetized the current will stop flowing, it only reacts to a change of the field potential. The magnetic principal that spins the motor is that for equal

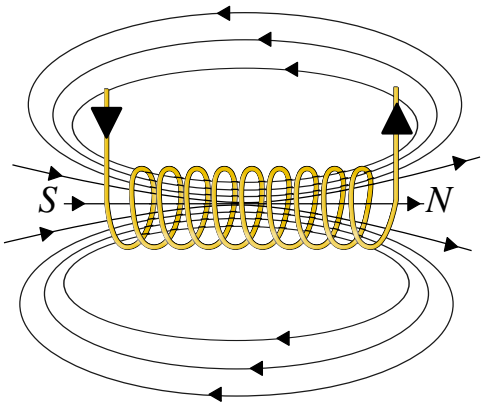
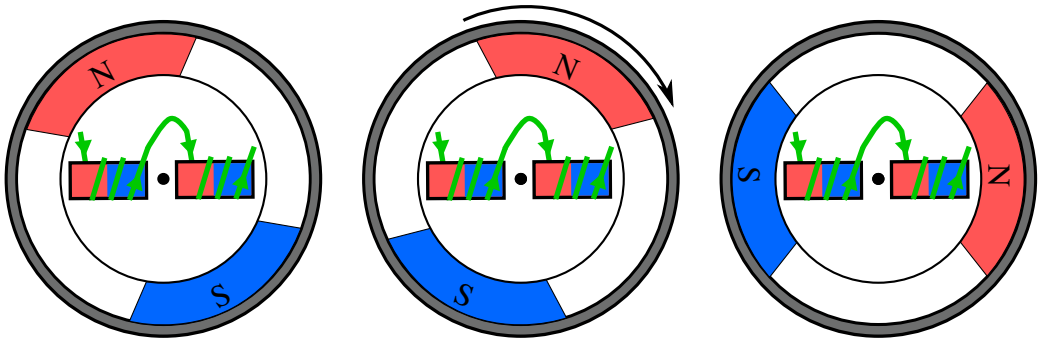


Figure 2.13: *The principle of an electromagnet. Current flows through a coil producing a magnetic field as shown.*

polarity of two magnets a force is pushing the magnets from each other, and two magnets with reversed polarity are drawn together, illustrated in Fig. 2.11.

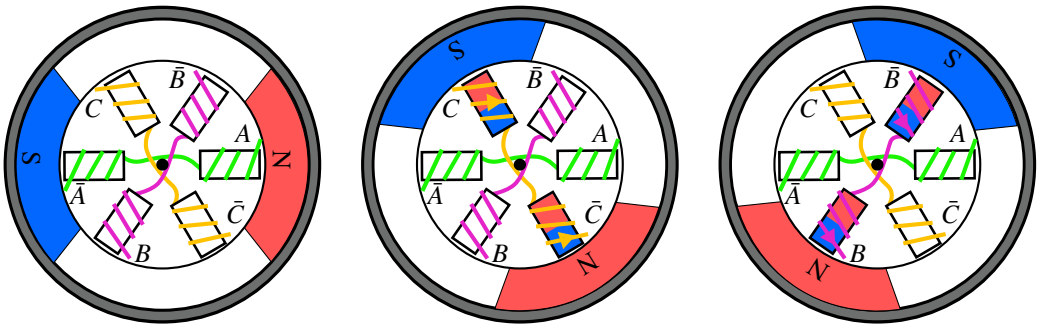


a) Initial position b) Clockwise rotation c) Steady state position

Figure 2.14: *From the initial position a), the north pole of the stator pushes the north pole of the rotor away from itself, and the south pole of the stator pushes the south pole of the rotor. The rotor starts to rotate, and in the b) position the south pole is pulling the north pole and vice versa, until it reaches c), the steady state position without movement.*

As shown in Fig. 2.14 c), the motor has come to a steady state position; the south pole of the rotor is lined up with the north pole of the stator, and vice versa. If the current in the stator is reversed in this position, the rotor becomes unstable and the direction of rotation is non-deterministic, it can either go clockwise or counter-clockwise. To overcome this two new set of stators are added to the system, shown in Fig. 2.15. The steady state position is shown in Fig. 2.15 a), with the stator not being energized. To continue the clockwise rotation positive current has to be put in to phase C. The positive current generates a magnetic field that pulls the stator in the clockwise direction. The next step to continue the rotation is to move the current out of phase C and into phase $\bar{B}$, which is the same as negative current into phase B.

If every phase is energized with a positive current and at the same time the rotor is rotated with some external force, the motor generates a torque as shown in Fig. 2.16. The torque curve shown is an ideal curve, and impossible to achieve in a physical motor. The trapezoidal shape is defined by how the stator is wound. Some motors with a sinusoidal shape also exist, but is not covered in this thesis. A torque resulting in a clockwise rotation of the rotor is defined as a positive torque.



a) Initial position b) Clockwise rotation c) Steady state position

Figure 2.15: *From the initial position a), the north pole of the stator pushes the north pole of the rotor away from itself, and the south pole of the stator pushes the south pole of the rotor. The rotor starts to rotate, and in the b) position the south pole is pulling the north pole and vice versa, until it reaches c), the steady state position without movement.*

The positive rotation should also result in a positive torque. Inverting the current where the torque is negative reverses the torque and the result is a positive torque throughout the rotation. For the regions where the torque is transitioning from high to low, or low to high torque, the current is turned off. This results in two phases contributing to positive torque throughout the entire rotation, shown in Fig. 2.17. On the figure the current is shown as steps, and the voltage is shown as the three upper trapezoidal graphs. The voltage, also referred to as back-emf (electromotive force) is partly from the current magnetizing the coil, but also from the magnet passing by, when non-energized.

The electromechanical torque T is defined as:

$$T = (4NBlr) \cdot I \tag{2.35}$$

$$T = k_t \cdot I \tag{2.36}$$

where N is the number of armature coils and r is the radius of the rotor. For BLDC used on RC-equipment such as multirotors, the motor constant is given as $k_v[rpm/volt]$. k_v is converted to $k_e = k_t$:

$$k_e = \frac{2\pi}{60} \cdot \frac{1}{k_v} \tag{2.37}$$

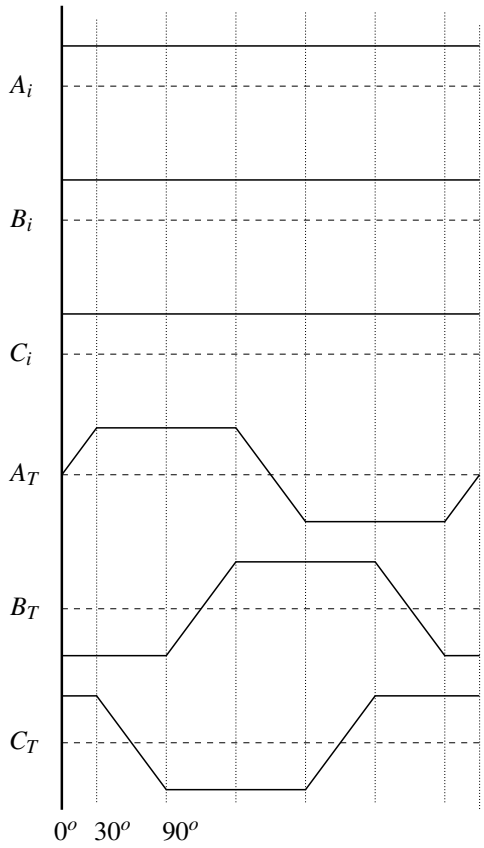


Figure 2.16: *The resultant torque (A_T, B_T, C_T) when the rotor is forced a clockwise rotation with every phase positively energized (A_i, B_i, C_i).*

As shown in Fig. 2.17 one phase is always positive another is negative and the last phase has zero current flowing. The wire configuration is reordered to simplify the setup. Each of the three wires $\bar{A}$, $\bar{B}$, $\bar{C}$ are directly connected together in a neutral point shown as the purple ring in Fig. 2.18. A positive flow of current in to phase A will flow through the coil and enter $\bar{B}$ (negative current in B) and leave the motor from phase B. From Fig. 2.17 at the $30^o - 90^o$ region this is resulting in a positive torque from both phases. With the new electrical design of the motor there is only three wires to control A, B and C. An equivalent electrical motor circuit of the BLDC is shown in Fig. 2.19

The BLDC shown in the previous figures has two magnets, also referred to as two poles. With two poles the stator has to change its current 6 times (six commutation states) in order to rotate the rotor one mechanical rotation. Most of the BLDCs

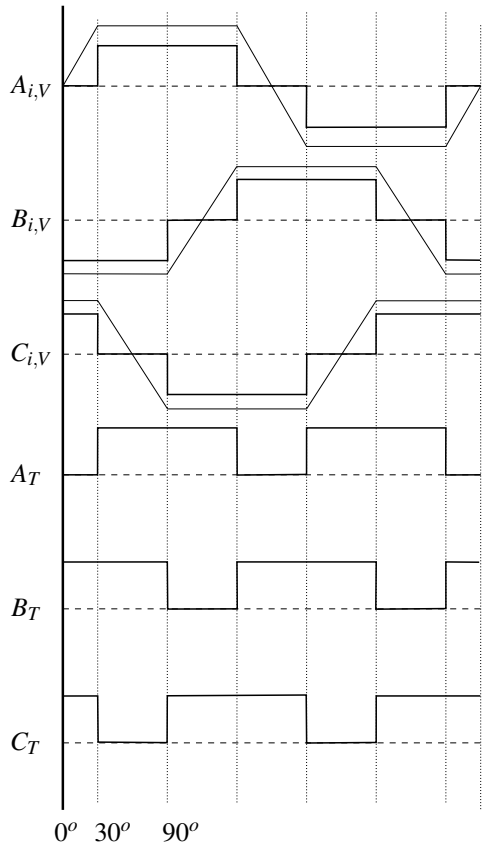


Figure 2.17: *The phase current (step shape) changes direction resulting in two phases generating positive torque. The phase where the torque is transitioning is switched off. The back-emf (trapezoidal shape) is plotted on top of the current graph.*

used on multicopters have more than two poles. The typical range is 12-22 poles and a range from 6-24 stators. With more than 2 poles, the mechanical rotation of the rotor becomes less than the electrical rotation of the motor. For instance with 4 poles the stator has to rotate 2 electrical rotations per mechanical rotation.

From $30-150$ degrees on Fig. 2.17 phase A has a positive current flowing through the coil producing a north pole at the rotor side, and pulling the south pole of the rotor clockwise. At the 150 degrees point, the current is switched off, and resulting in a decreasing back-emf. Even if the coil is turned off the next pole passing the stator (a north pole) will pull the back-emf below zero volt to the negative limit. Once it reaches this limit, the coil is enabled with a negative current keeping the voltage negative.

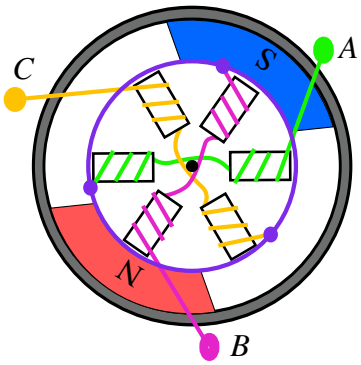


Figure 2.18: *The resultant torque (A_T, B_T, C_T) when the rotor is forced a clockwise rotation with every phase positively energized (A_i, B_i, C_i).*

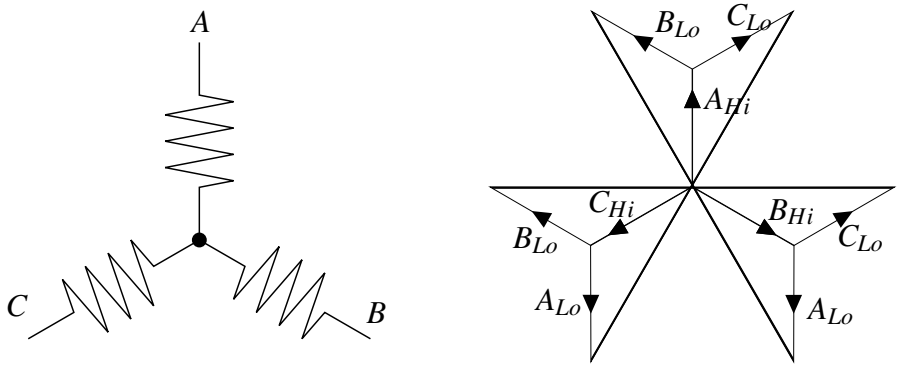


Figure 2.19: *BLDC electrical equivalent circuit and the phasor diagram showing the six commutation steps (bold vectors).*

The commutation of the brushed DC motor is fairly simple. It is taken care of by the mechanical design and it only needs current flowing through its windings (applied voltage) to rotate. The BLDC is not that simple because it needs to know the rotor position in order to commutate correctly. Industrial BLDC and high precision application uses external sensors to determine the position of the rotor. A simpler approach is to measure the back-emf and use this information to tell the position of the rotor. The most common way to use the back-emf for rotor position measurements is to measure the zero-crossing of the voltage. The zero-crossing is detected with 30 electrical degrees before the commutation time. The phase is compensated for in the controller.

Using the back-emf is a way to measure the rotor position when the rotor is rotating. When the rotor is not rotating, there is no change of the magnetic field

and the back-emf voltage is zero. This means that during start-up and at very slow rotation velocities the speed controller has no information of the rotor position. However the propellers of the multicopters are never rotating so slowly that there is no back-emf present, except during start up of the motor. At low velocities the controller does not care about the rotor position, it only sets up a slow rotating electrical field, and hopes that the motor will start to rotate along with the field. Once the motor has sufficient velocity the motor controller can measure the back-emf and control the velocity of the motor.

The electrical motor diagram is shown in Fig. 2.20. The external load for this

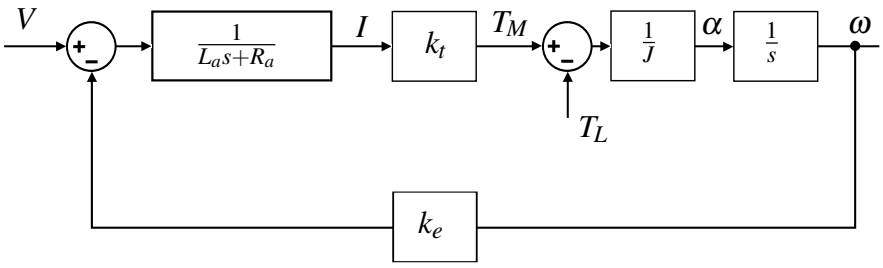


Figure 2.20: *Motor block diagram, with torque feedback from propeller. Where L_a is the motor inductance, R_a is the resistance of the windings and motor leads, k_t and k_e is the motor torque- and voltage constants respectively, J is the inertia of the motor and propeller, and T_L is the external load.*

setup is the propeller torque (2.32) and the viscous damping. The viscous damping is not taken into consideration in this thesis, leaving the external load $T_L = T_p$. The system shown has two different time constants, electrical (τ_e) and mechanical (τ_m). The electrical time constant is equal to $\tau_e = L_a/R_a$. The inductance and resistance of the motor highly depends on the choice of motor, but is typically in the range of ~$100\mu H$ and ~0.1Ω which gives an electrical time constant of $\tau_e = 1ms$. Assuming that the electrical time constant is much smaller than the mechanical it can be neglected from the system by setting $L_a = 0$. The mechanical time-constant is then found from the entire transfer function of the system $\omega(s)/V$.

$$\frac{\omega(s)}{V} = \frac{k_t}{\frac{JR_a}{k_t k_e + DR_a}s + 1} \tag{2.38}$$

where D is the function of the external load from the angular velocity. The mechanical time constant is therefore dependent of the angular velocity of the system.

A variety of propellers, both large and small, were tested in Paper D. The lowest time constant measured was $43ms$. The assumption of neglecting the electrical time constant to simplify the system is therefore valid.

When the motor is starting up at for instance 50% duty cycle, the back-emf equals zero. This results in a high current and a high torque. The propeller is not producing any torque at zero RPM, and the motor torque divided by the inertia results in an acceleration of the motor, where the angular velocity is the integral of the acceleration. As the motor starts spinning it starts to produce back-emf and the effective voltage applied to the motor is reduced. The propeller starts producing torque and the net torque accelerating the motor is therefore reduced. The motor will accelerate as long as the net torque is greater than zero. If it is below zero, the propeller brakes the motor.

Compared to the brushed motors the brushless motors have many advantages and few disadvantages. It requires less maintenance (no wear on brushes) and thus has a longer life span. The efficiency is increased as it has no voltage drop over the brushes. Also the output power with respect to the size is higher and for the flying multirotor it is essential to keep the weight low. One of the reasons is that it has the windings on the stator which are connected to the casing allowing the heat to flow to external hardware. Also, on most of the designs, the rotor is the outer part of the motor and is usually designed with special shaped holes allowing for more air to flow through the motor. With increased heat dissipation the characteristics of the motor can be increased. The electric noise is lower for the BLDC. For the brushed motor arcs are generated when the brushes cross the commutator causing electromagnetic interference (EMI) to the nearby equipment. This could be fatal if the noise is affecting the sensors such as the accelerometer or gyroscope. The BLDC also has flat speed/torque characteristics. On the downside the BLDC costs more than the brushed motors, it is more complex and requires a more complex controller. The advantages are considered significantly higher than the disadvantages and the BLDC is the preferred motor choice.

2.2.1.4 Electronic Speed Controller

The electronic speed controller (ESC) is controlling the phase current of the BLDC. The RC-industry is a big actor in this field, and off-the-shelf products are good enough to control the motors on a multicopter. The RC equipment has for many years, long before the multicopters became famous, used PWM-signals as input to the ESC. The PWM signal has a frequency of 50Hz, with a high duration of 1-2ms where 1ms is the lowest speed command, and 2ms is the maximum speed command. A multicopter is not able to fly with such a low update frequency of the motors, so the PWM frequency has for many ESCs been increased to 490Hz (2.04 ms), which is the maximum frequency since the high pulse is 2ms. Different manufacturers have different update frequency, it ranges from 150Hz to 490Hz. 150Hz is the lowest frequency possible in order to maintain a stable flight.

The ESC is controlling the motor speed by commutating the coils. Each of the three phases, A, B and C, are controlled separately with a pair of transistors, one transistor on the high side (supply side) and one on the low side (ground), in a total of six transistors, shown in Fig. 2.21. Enabling A_{Hi} and B_{Lo} the current will flow through phase A, and out of phase B.

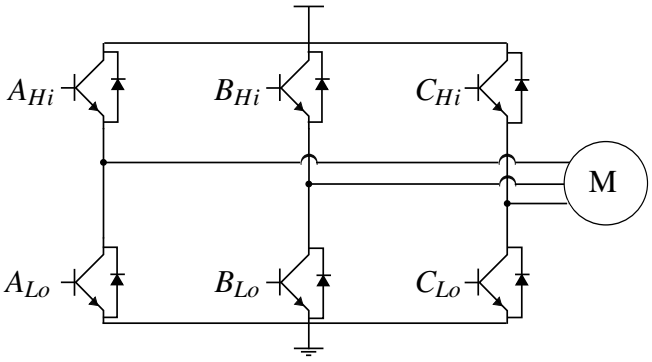


Figure 2.21: *Transistor arrangement of an ESC.*

At full speed the ESC is opening the transistors fully throughout the ON-period. This allows for the maximum voltage possible to the motor. To vary the motor velocity the output of the ESC is a PWM-signal. The transistors are fully on or fully off in these two periods. This reduces the power loss over the transistors keeping the efficiency high. The motor which can be seen as a low pass filter, will keep a

constant speed even if its input signal is modulated. This is because the modulation frequency at ~18kHz is high enough to be filtered out.

The current flows from the high side to the low side of the ESC. This means that it is only the high side that needs to be pulse width modulated. The low side can be fully open at the specified period. A typical commutation method is shown in Fig. 2.22. The figure shows the commutation sequence for one electrical rotation.

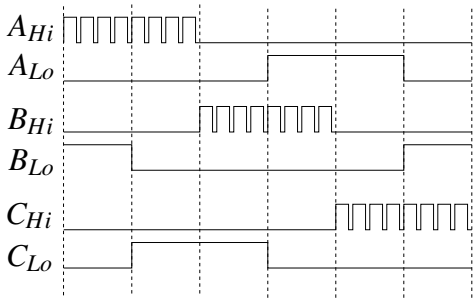


Figure 2.22: *Commutation sequence for one electrical rotation of a BLDC driven by an electronic speed controller.*

When the high side is off there is no power applied and the motor is coasting, also known as free-wheeling. The current cannot change immediately over an inductor. This means that even though the high side is off, there is current flowing through the motor (free wheeling current). Looking at the first region of Fig. 2.22 phase A_{Hi} is pulsing and phase B_{Lo} is kept high. During A_{Hi}'s off period the current flows through phase A, and out of phase B whose transistor is open. A_{Lo} is switched off, but the current flows through the diode and into phase A shown in Fig. 2.23. The resistance over the diode is significant and thus reducing the current flowing through the phases. The low current is braking the motor slowly and the propeller is the one braking the motor the most. This means that there are two time constants for the actuator system, one when the motor is accelerated and one where the motor is decelerated, making the system non-linear.

It is not only the input or output frequency of the ESC that is important when controlling a multicopter, the commutation method is also important. The method shown in Fig. 2.22 is only one out of many commutation methods. When the high side is off, the motor is coasting, and the free-wheeling current is limited because of the diode. If A_{Lo} were open during A_{Hi}'s off period, the resistance will be reduced

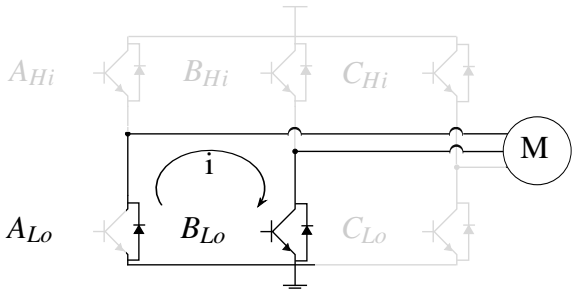


Figure 2.23: *When coasting the current flowing through the motor is limited due to high resistance over the fly-back diode.*

and the current increased. This commutation method is known as complementary PWM switching since the low side is complementary to the high side during the pulsating state, shown in Fig. 2.24.

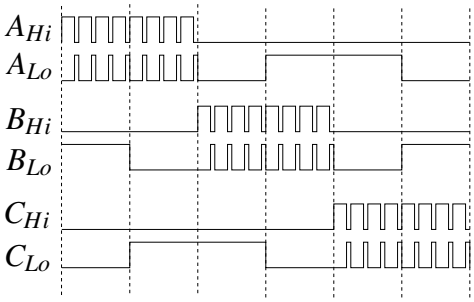


Figure 2.24: *Complementary PWM-switching of the transistor.*

The complementary commutation method has less resistance in the off-period due to the low resistance of the transistor resulting in more current flowing through the windings, as shown in Fig. 2.25. With high current the motor brakes faster. In fact the current braking the motor is larger than the braking torque from the propeller, leaving the ESC as the main braking source of the system. With the high current braking the motor the system the time constant for accelerating and decelerating the actuator becomes more linear with a faster dynamic response.

One down-side with the complementary PWM mode, is that not every transistor is able to handle negative current. Another drawback is that there is a possibility that the high side and low side are open (partly open) at the same time. This short-circuits the ESC and it will break. Thus some dead times are added to the timing of the transistors in order to be sure that the high-side is off before the low-side is

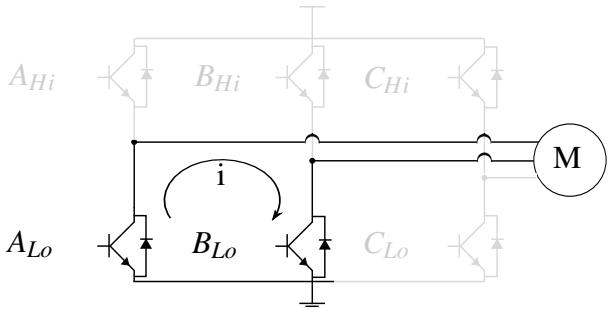


Figure 2.25: *With complementary PWM-switching the current is flowing* .

enabled.

2.2.2 Inertial Measurement Unit

The inertial measurement unit (IMU) is an essential sensor for the multirotor. The accelerometer provides acceleration data and the gyroscope angular velocity data, both for three dimensions.

2.2.2.1 Accelerometer

There are many methods to measure the acceleration. Piezoelectric accelerometers produce a small voltage when stressed by a small mass. The voltage is measured, and converted to acceleration. Other methods measure the bending of a beam, change of capacitance or even hot air bubbles. The chips are small in size, measuring only a few millimetres. The accelerometer measures the acceleration of the device, that includes the gravity. At rest the three axis accelerometer will measure an acceleration vector with a length of 1g, depending on the orientation of the device.

The accelerometer can be used to measure orientation if gravity is the only acceleration sensed by the device. Aligning the Z-axis of the accelerometer with the unit vector going straight into the earth, Z-axis will measure 1g and X and Y will measure 0g. Since the accelerometer is only measuring acceleration it does not matter in what direction the X-axis is pointing, the measurements will still be the same. The accelerometer can therefore only be used to measure roll and pitch angles, not yaw.

Mounted on a multirotor the accelerometer will measure more than just the gravity. As the multirotor flies around the accelerometer will detect some of the multirotors accelerations. The accelerations from the motion of the multirotor is considered very small compared to the relatively large accelerations of the gravity. The onboard systems also knows that if the length of the total acceleration vector is different from 1g, the multirotor is performing aggressive manoeuvres and the measured acceleration shold be avoided in the control loop.

The accelerometer is also sensitive to noise, especially from the multirotor. The propellers are inducing a lot of vibration in the airframe, making the signal noisy. This noise is high frequent and applying a low pass filter to the accelerometer will reduce the noise. Digital accelerometers usually have inbuilt filters that the user can configure. Even though the accelerometer might have issues with offsets, the orientation estimation based on the accelerometer will never drift as the estimation is not based on integration of the signal.

2.2.2.2 Rate Gyro

There are also many methods to measure angular velocity. Ring lasers provide the highest accuracy of measurements and are not affected by vibrations or accelerations. Fibre optic gyros measure the interference of light which has rotated in opposite direction of a fibre optic cable. Those gyros are also very accurate but are relatively large in size compared to the inertial gyros. The inertial gyros measure rotational velocity of the Coriolis effect. The gyros consist of vibrating beams, typical crystals. When the gyro is rotating the beams will move due to the Coriolis effect and the motion is detected.

The rate gyros, or just gyros, are measuring the rate of change of the orientation, not the orientation itself. To measure attitude from the gyroscope the rate of change has to be integrated over time. However the gyros can only estimate the orientation from its initial position, not relative to earth. The gyros are also affected by bias, which in turn will accumulate error of the integrated attitude estimation as a low frequency signal. Prior to flight and while the multirotor is stationary, this bias can be measured and compensated for. It will however change over time depending on temperature etc. The gyroscope is not as sensitive to the high frequency vibrations

of the airframe. The integrator used to estimate attitude is also a low pass filter that reduces the noise.

The accelerometer and gyroscope are two complementary sensors, in terms of measuring different quantities, and in terms of sensitivity to high frequent noise and low frequent signals.

2.3 Multirotor Modelling Concepts

A mathematical model of the multirotor can be used in a simulation model for computer experiments. The local forces on the multirotor have to be converted to forces in the body frame, either by the rotation matrix, or quaternions. The approach is more or less the same, but only described for quaternions in this thesis.

A quadcopter with reference frames is shown in Fig. 2.1. The body frame is denoted with a subscript 'b' and the navigation frame is a fixed frame denoted with a subscript 'n'. Positive axes and motor torques for the quadrotor are also shown in the same figure.

Recall from Chapter 2.1.5, that a vector $v = [v_x\, v_y\, v_z]^T$ in 3D space can be rotated to the vector v' by a quaternion with the quaternion operator $\otimes$. A vector v_b in the body frame is rotated to a new vector v_n the navigation frame using the same multiplication rules:

$$v_n = q \otimes v_b \otimes q^{-1} \tag{2.39}$$

where the new rotated vector v_n is the last three elements of the multiplication result.

The mathematical model is described in two parts, summing the forces, and summing the moments.

2.3.1 Forces

Forces acting on the multirotor are the gravity F_{ng} in the Z-direction of the navigation frame, propeller thrust F_{bz} in the body Z-direction and the gyroscopic force from the angular velocity ω_b. Blade flapping, wake interaction and any other effects caused by the translational velocity are ignored. Summing forces F_b in the

body coordinate system yields:

$$\sum F_b = m\dot{V}_b$$

$$\dot{V}_b = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ F_{bz} \end{bmatrix} + q^{-1} \otimes \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \otimes q - \omega_b \times V_b \tag{2.40}$$

$$\dot{V}_n = q \otimes \dot{V}_b \otimes q^{-1} \tag{2.41}$$

where m is the mass of the multirotor, $\dot{V}$ denotes the acceleration and V denotes the velocity in the defined frames.

2.3.2 Moments

Moments acting on the multirotor are the ones from different motor thrusts, rotational torque from the motor due to the propeller and the gyroscopic force. The induced motor torque from motor acceleration and other moments are considered very small compared to the moments described and are neglected. Summing the moments M_b in the body coordinate system yields:

$$\sum M_b = I_b \dot{\omega}_b$$

$$\dot{\omega}_b = I_b^{-1}\left(\begin{bmatrix} M_{bx} \\ M_{by} \\ M_{bz} \end{bmatrix} - \omega_b \times (I_b \omega_b)\right) \tag{2.42}$$

where I_b is the inertia of the quadcopter and M_b is the moment about each body coordinate axis from the propeller thrust:

$$\begin{bmatrix} M_{bx} \\ M_{by} \\ M_{bz} \end{bmatrix} = \begin{bmatrix} 0 & 0 & d_m & -d_m \\ -d_m & d_m & 0 & 0 \\ T_q & T_q & -T_q & -T_q \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{2.43}$$

where d_m is the distance from the quadcopters center of gravity to the center of the motors. There is a linear relationship between the thrust of the propellers and the torque which is produced, also seen from equation 2.31-2.32 , T_q is the coefficient describing this relationship.

With the two equations of accelerations the simulation model is realised as shown in Fig. 2.26. Conversion to roll, pitch and yaw is not necessary as the quaternion also represents the angle. The conversion makes it however easier to visualize the response or attitude of the system.

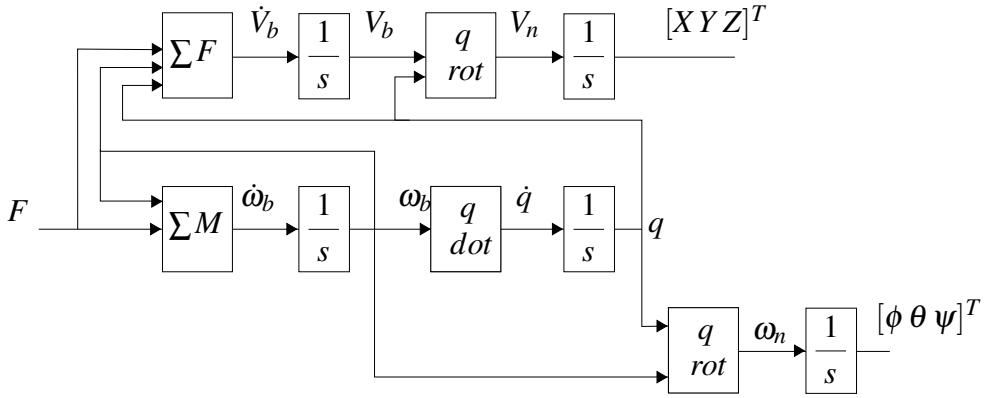


Figure 2.26: *Simulation model of the multirotor.*

2.4 Design Optimization Using Mixed-Integer Programming

A quadratic program is defined as follows:

$$\min \quad \mathbf{x^T Q x + g^T x} \tag{2.44}$$

$$\text{subject to :} \quad \mathbf{A x \leq b} \tag{2.45}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector to be solved, $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is a positive semi-definite penalty matrix, $\mathbf{g} \in \mathbb{R}^n$ is a penalty vector, $\mathbf{A} \in \mathbb{R}^{n \times n}$ is the constraint matrix while $\mathbf{b} \in \mathbb{R}^n$ is the constraint vector. When the penalty matrix $\mathbf{Q} = \mathbf{0}$, eqs. (2.44-2.45) reduce to a linear program. When some elements x_i where $i \in [1, \cdots, n]$ are constrained to be integer variables, eqs. (2.44-2.45) are called mixed integer quadratic

program (MIQP) or, when $\mathbf{Q} = \mathbf{0}$, mixed integer linear program (MILP).

By constraining the integer variables further to accept only boolean values (0 or 1), logical constraints can be incorporated and linked with the continuous variables in the optimization problem. The following example rules are taken from [Bemporad and Morari, 1999] and [Mignone, 2002]. In the examples below the following constants are defined:

$$\begin{aligned} m &= \min_{x \in X} f(x) \\ M &= \max_{x \in X} f(x) \\ \varepsilon &\quad \text{denotes a small, real, positive constant, typically the machine precision.} \end{aligned}$$

It should be noted that equality constraints of the type $ax = b$ must be converted to two inequality constraints $ax \leq b$ and $-ax \leq -b$ to satisfy eq. (2.45). However, the solver used (CPLEX) allows specification of both equality and inequality constraints.

Rule 1

Product of boolean variable and function of continuous variables.

$$z = \delta \cdot f(x)$$

or

$$\begin{aligned} &\text{IF } [\delta == 1] \text{ THEN} && z = f(x) \\ &\qquad\qquad \text{ELSE} && z = 0 \end{aligned}$$

is equivalent to

$$\begin{aligned} -M\delta + z &\leq 0 \\ m\delta - z &\leq 0 \\ -m\delta + z &\leq f(x) - m \\ M\delta - z &\leq -f(x) + M \end{aligned}$$

Rule 19a

Implication.

$$[f(x) \leq 0] \rightarrow [\delta = 1]$$

is equivalent to

$$(m - \varepsilon)\delta \leq f(x) - \varepsilon$$

Rule 19b

Implication.

$$[f(x) \geq 0] \rightarrow [\delta = 1]$$

is equivalent to

$$-(\varepsilon + M)\delta \leq -f(x) - \varepsilon$$

2.4.1 Example

To illustrate the development of a MILP problem an example is shown. In this example it is shown how the **A**-matrix and the vectors $\mathbf{x}, \mathbf{b}, \mathbf{g}$ are expanded as new elements are implemented into the system. The example is quite familiar to how the MILP was constrained in the optimization of the multirotor system.

An electric mobile robot Fig. 2.27 designed for inspection of a plant has to run as long as possible before charging the batteries.

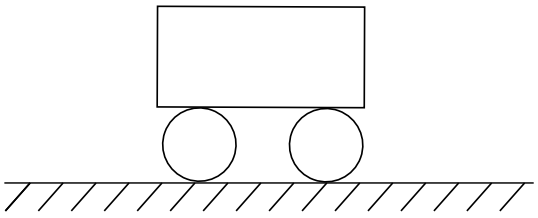


Figure 2.27: *Electric mobile robot.*

The current consumption C is equal to

$$C = (w_r + w_b)\,[kg] \cdot 2 \left[\frac{A}{kg}\right] \tag{2.46}$$

where $w_r = 2kg$ is the weight of the mobile robot and w_b is the weight of the battery. The following batteries can be chosen:

Table 2.2: *Battery selection*

Battery:	Capacity [Ah] (Cap)	Weight [kg] (W)	Price (P)
# 1	10	1	100
# 2	20	2	200
# 3	30	3	300
# 4	40	4	400

Task The motor with an internal resistance $R_a = 1\Omega$ is rated to $E = R_a I^2 = 30W$. Use MILP to chose a battery that gives the longest run time, without exceeding the power limitation. Define the system using MILP, and chose the battery that gives the longest run time.

Solution Solving such an example using the MILP requires to determine the state variables. The state variables are split in two sections, the ones that are integers, and the continuous ones. The variables are shown in Table 2.3 and Table 2.4.

Table 2.3: *Boolean decision variables. If a δ variable equals 1, then the corresponding component/region is selected. If $\delta = 0$, the component/region is not selected.*

Integer variables	**Variable name**	**Description**
$x_1 \ldots x_4$	$\delta_1 \ldots \delta_4$	Set of boolean for the chosen battery
$x_5 \ldots x_{14}$	$\delta_{s1} \ldots \delta_{s10}$	Used to calculate the square of the current I^2

Table 2.4: *Continuous variables used in the design optimization.*

Continuous variables	**Variable name**	**Description**
x_{15}	W	Weight of vehicle
x_{16}	P	Cost of vehicle
x_{17}	C	Current drain
x_{18}	R	Runtime
$x_{19} \ldots x_{28}$	$C_1^2 \ldots C_{10}^2$	Set of current^2
x_{29}	C^2	Current^2
x_{30}	E	Power consumption [W]

2.4.1.1 Boolean Variables

The first elements of the x-vector is a boolean variable which represents if the current batty is selected or not. Either the battery is selected or not, hence the four variables are boolean variables.

$$x = [\delta_1; \delta_2; \delta_3; \delta_4; ...]; \tag{2.47}$$

where $\delta_1 - \delta_4$ are Boolean variables one for each of the four batteries. The Boolean values are integer values of either 0 or 1 and are constrained by:

$$1 \cdot \delta \leq 1 \tag{2.48}$$

$$-1 \cdot \delta \leq 0 \tag{2.49}$$

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots \\ -1 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 0 & \cdots \\ 0 & -1 & 0 & 0 & \cdots \\ \vdots & & & & \end{bmatrix} \end{array} \cdot \begin{array}{c} \\ \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \vdots \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \leq \begin{array}{c} \\ \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \vdots \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \tag{2.50}$$

The same applies for $\delta_3 - \delta_{14}$ from row 5 to 28.

The optimizer has to select one of the batteries. Hence the sum of δ_1-δ_4 must be equal to one:

$$\delta_1 + \delta_2 + \delta_3 + \delta_4 = 1 \tag{2.51}$$

This results in the corresponding A matrix and x and B vectors:

$$\begin{array}{c} \\ 29 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & & \end{array} \\ \begin{bmatrix} \vdots & & & & & \\ 1 & 1 & 1 & 1 & \cdots & \\ \vdots & & & & & \end{bmatrix} \end{array} \cdot \begin{array}{c} \\ \begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \vdots \end{bmatrix} \end{array} \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} = \begin{array}{c} \\ \begin{bmatrix} \vdots \\ 1 \\ \vdots \end{bmatrix} \end{array} \begin{array}{c} \\ 29 \end{array} \tag{2.52}$$

Only one battery can be selected, and the corresponding δ-value is set to one. The rest of the δ-values are zero. This simplifies the calculation of the battery weight. The battery weight W_{bat} is calculated as the sum of the δ's multiplied with the corresponding battery weight. To convert this to a linear constraint, every product is moved to the left side of the equation, leaving the value of $B = 0$:

$$W_{bat} - W_1\delta_1 - W_2\delta_2 - W_3\delta_3 - W_4\delta_4 = 0 \tag{2.53}$$

$$
\begin{array}{c} \\ 30 \end{array}
\begin{array}{c}
\begin{array}{ccccccc} 1 & 2 & 3 & 4 & & 15 & \end{array} \\
\begin{bmatrix} \vdots & & & & & & \\ -W_1 & -W_2 & -W_3 & -W_4 & \dots & 1 & \cdots \\ \vdots & & & & & & \end{bmatrix}
\end{array}
\cdot
\begin{array}{c}
\begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \vdots \\ W_{bat} \\ \vdots \end{bmatrix}
\end{array}
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ \\ 15 \\ \end{array}
=
\begin{array}{c}
\begin{bmatrix} \vdots \\ 0 \\ \vdots \end{bmatrix}
\end{array}
\begin{array}{c} \\ 30 \end{array}
\tag{2.54}
$$

The price, current drain and run time are constrained as in equation 2.54.

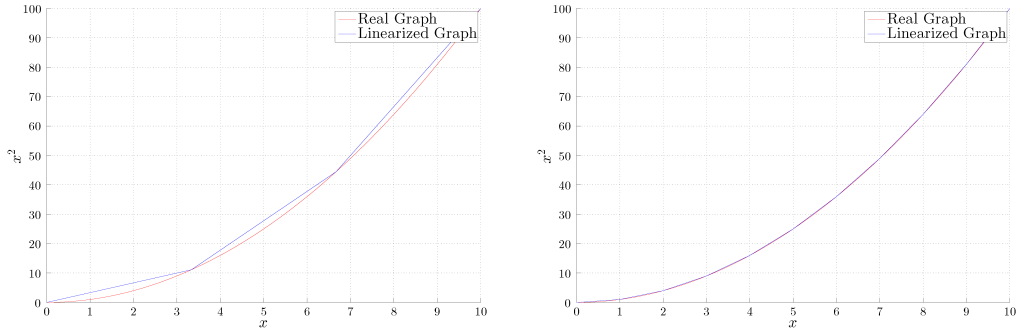
$$
\begin{array}{c} \\ 31 \\ 32 \\ 33 \end{array}
\begin{array}{c}
\begin{array}{cccccccccc} 1 & 2 & 3 & 4 & & 15 & 16 & 17 & 18 & \end{array} \\
\begin{bmatrix} \vdots & & & & & & & & & \\ -P_1 & -P_2 & -P_3 & -P_4 & \dots & 0 & 1 & 0 & 0 & \cdots \\ -C_1 & -C_2 & -C_3 & -C_4 & \dots & 0 & 0 & 1 & 0 & \cdots \\ -R_1 & -R_2 & -R_3 & -R_4 & \dots & 0 & 0 & 0 & 1 & \cdots \\ \vdots & & & & & & & & & \end{bmatrix}
\end{array}
\cdot
\begin{array}{c}
\begin{bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \vdots \\ W \\ P \\ C \\ R \\ \vdots \end{bmatrix}
\end{array}
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ \\ 15 \\ 16 \\ 17 \\ 18 \\ \end{array}
=
\begin{array}{c}
\begin{bmatrix} \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}
\end{array}
\begin{array}{c} \\ 31 \\ 32 \\ 33 \end{array}
\tag{2.55}
$$

where the price P_n of each battery is given from Table 2.2. The current C_n is calculated as in 2.46. The run time R in hours is the current capacity of the battery divided by the current drain:

$$
R_n = \frac{Cap_n[Ah]}{C_n[A]} \tag{2.56}
$$

2.4.1.2 Power function

It is not possible to multiply variables in the x-vector with each other when using linear constraints, and therefore not possible to calculate I^2 directly in order to calculate the power consumption. The optimization algorithm calculates the exponential value of variables using linearisation. The power function is divided into linear regions of equal length on the X-axis with the function of a straight line $y = ax + b$ and is not optimized with respect to induced error. The power function can be of any order within a limited range. The more regions used to divide the curve into the better the result, but it also gives a more complex system with more variables. Fig. 2.28 illustrates a power of 2 conversion, from power of 1 to power of 2, for both three and ten regions. Each line region is the linear curve from the first to the next point on the non-linear curve to be linearised. The a and b values of the linearisation



3 regions 10 regions

Figure 2.28: *Comparison of the linearization of the conversion from x to x^2 for 3 and 10 regions.*

are calculated as:

$$a = \frac{X_2^p - X_1^p}{X_2 - X_1} \tag{2.57}$$

$$b = X_1^p - aX_1 \tag{2.58}$$

where X_1 is the lower point of the region, X_2 is the upper point and p is the exponent value of the linearisation.

Each region has its own upper and lower limits, slope coefficient, offset and boolean value, shown in Table. 2.5 for three regions.

Table 2.5: *Linearisation values*

L_{min}	L_{max}	**a**	**b**	δ
0.00	3.33	3.33	0.00	δ_{s1}
3.33	6.66	10.00	-22.22	δ_{s2}
6.66	10.00	16.67	-66.67	δ_{s3}

If δ_{s1} is selected, the linear function equals $y_1 = 3.33x + 0$. This is also the function of the first line shown in Fig. 2.28. The task uses the described power function to linearise the square of the current used to calculate the power output. For each of those regions there is a boolean variable

Each region has a boolean variable which is true if the region is active, that is if the value is within the limits:

$$L_{min} \leq f(x) \leq L_{max} + \varepsilon \rightarrow [\delta_{sn} = 1] \tag{2.59}$$

where ε is the machine precision and is added to avoid two regions being set at the same time.

The two parts in equation (2.59) are split in two and using a slightly modified version of Rule 19 the optimizer clears the boolean if the value $f(x)$ is less than the limit L_{min}:

$$\begin{aligned} [f(x) < L_{min}] &\rightarrow [\delta_{sn} = 0] \\ \delta M - f(x) &\leq -L_{min} + M \end{aligned} \tag{2.60}$$

Also, the boolean variable is cleared if the value of the function $f(x)$ is greater or equal to the limit L_{max}:

$$\begin{aligned} [f(x) > L_{max}] &\rightarrow [\delta_{sn} = 0] \\ \delta M + f(x) &\leq L_{max} + M + \varepsilon \end{aligned} \tag{2.61}$$

The two constraints from (2.60) and (2.61) will only clear the boolean variables which represent a value outside of the limits. This will leave the boolean of the

region within the limit unconstrained. To set this boolean, the sum of the region boolean are set equal to one, similar as in (2.52).

$$
\begin{array}{c}
\\
34 \\
35 \\
36 \\
37 \\
\\
\end{array}
\begin{array}{c}
\begin{array}{ccccccc} & & 5 & 6 & & 17 & \end{array} \\
\begin{bmatrix}
\vdots & & & & & & \\
\dots & M & 0 & \dots & 1 & \cdots \\
\dots & M & 0 & \dots & -1 & \cdots \\
\dots & 0 & M & \dots & 1 & \cdots \\
\dots & 0 & M & \dots & -1 & \cdots \\
\vdots & & & & & &
\end{bmatrix}
\end{array}
\cdot
\begin{array}{c}
\\
\begin{bmatrix} \vdots \\ \delta_{s1} \\ \delta_{s2} \\ \vdots \\ C \\ \vdots \end{bmatrix}
\end{array}
\begin{array}{c}
\\
5 \\
6 \\
\\
17 \\
\\
\end{array}
=
\begin{array}{c}
\\
\begin{bmatrix} \vdots \\ -L_{min1} + M \\ L_{max1} + M + \varepsilon \\ -L_{min2} + M \\ L_{max2} + M + \varepsilon \\ \vdots \end{bmatrix}
\end{array}
\begin{array}{c}
\\
34 \\
35 \\
36 \\
37 \\
\\
\end{array}
\tag{2.62}
$$

where $M = 6[A]$ which is the highest current drain possibly for this task, $\varepsilon = 10^{-5}$ which is a small arbitrary number. 10 regions are used in the linearisation of the power function, giving $L_{min1} = 0$, $L_{max1} = 0.6$, $L_{min2} = 0.6$, $L_{max2} = 1.2$. These sets of inequalities make sure that the only δ_{sn} to be set is the one that represents the linear region of the actual current drain of the motor. Rule 1 calculates a value of a linear function of the selected state variable if a given boolean value is set. From the linearisation above there is one boolean for each region with a slope and offset used to calculate the square of the current.

$$C_n^2 = a_n \cdot C + b_n \tag{2.63}$$

Rule 1 generates four rows of data for each value. With a total of 10 regions this

results in 40 rows added to the system, where the first four rows become:

$$\begin{array}{c} \\ 38 \\ 39 \\ 40 \\ 41 \\ \\ \end{array} \overset{\begin{array}{ccccccc} & 5 & & 17 & 18 & 19 & \end{array}}{\begin{bmatrix} \vdots & & & & & & \\ \dots & -M & \dots & 0 & 0 & 1 & \cdots \\ \dots & m & \dots & 0 & 0 & -1 & \cdots \\ \dots & -m & \dots & -a_n & 0 & 1 & \cdots \\ \dots & M & \dots & a_n & 0 & -1 & \cdots \\ \vdots & & & & & & \end{bmatrix}} \cdot \begin{bmatrix} \vdots \\ \delta_{s1} \\ \vdots \\ C \\ R \\ C_1^2 \\ \vdots \end{bmatrix} \begin{array}{c} \\ 5 \\ \\ 17 \\ 18 \\ 19 \\ \\ \end{array} \leq \begin{bmatrix} \vdots \\ 0 \\ 0 \\ b_n - m \\ -b_n + M \\ \vdots \end{bmatrix} \begin{array}{c} \\ 38 \\ 39 \\ 40 \\ 41 \\ \\ \end{array} \tag{2.64}$$

where $M = 36$, which is the square of the maximum current possible. The slope equals $a_1 = 0.6$ and the offset for this region equals $b_1 = 0$, which also is the smallest possible value of the calculation, hence $m = 0$.

The set of inequalities from (2.67) results in a value of C_1^2 of either 0 ($\delta_{s1} = 0$), or a value of the linearisation of C^2 (δ_{s1}=1). Since only one of the $\delta_{s1} - \delta_{s10}$ are set, only one of the $C_1^2 - C_{10}^2$ are different from 0. The square value of the current is found by adding every value of $C_1^2 - C_{10}^2$, just as in 2.55.

The power E is calculated from the square of the current and the motor resistance $E = C^2 R_a$

$$\begin{array}{c} \\ 96 \\ \\ \end{array} \overset{\begin{array}{ccc} & 29 & 30 \end{array}}{\begin{bmatrix} \vdots & & \\ \dots & R_a & -1 \end{bmatrix}} \cdot \begin{bmatrix} \vdots \\ C^2 \\ E \end{bmatrix} \begin{array}{c} \\ 29 \\ 30 \end{array} = \begin{bmatrix} \vdots \\ 0 \end{bmatrix} \begin{array}{c} \\ 96 \end{array} \tag{2.65}$$

where $R_a = 1[\Omega]$

And the last inequality is to limit the total power of the system.

$$\begin{array}{c} \\ 97 \\ \end{array} \overset{\begin{array}{c} 30 \end{array}}{\begin{bmatrix} \vdots & \\ \dots & 1 \end{bmatrix}} \cdot \begin{bmatrix} \vdots \\ E \end{bmatrix} \begin{array}{c} \\ 30 \end{array} \leq \begin{bmatrix} \vdots \\ E_{max} \end{bmatrix} \begin{array}{c} \\ 97 \end{array} \tag{2.66}$$

where $E_{max} = 30[W]$.

The task is to optimize for the longest run-time of the mobile robot. Since the g-vector is a penalty vector the value has to be negated:

$$g = \begin{bmatrix} 0 \\ \vdots \\ -1 \\ \vdots \\ 0 \end{bmatrix} \begin{matrix} 1 \\ \\ 18 \\ \\ 30 \end{matrix} \tag{2.67}$$

For this task there was only one variable to optimize for, hence its value does not matter, only its sign.

Solving the set of inequalities takes only a few seconds. The solution of the MILP problem is battery #3. With this battery the weight of the mobile robot is calculated to $5kg$. The current drain $C = 5A$, this combined with the 30Ah battery, the optimizer calculates a runtime of 6 hours. The IBM CPLEX optimizer calculates the power drain $E = 25.08W$. This is not the correct answer, as the real value should be $E = I^2 \cdot R_a = 5^2 \cdot 1 = 25W$. The reason for this error of the power calculation is the linearisation of the square function used in the calculation. Increasing the number of regions allows for a lower error, but larger calculation time of the system.

Removing the power limitation of the system, the IBM CPLEX solver selects battery # 4 with a runtime of 6.6 hours. However this battery is excluded from the solution since the power equals $E = 36W$ which is above the given limit.

Chapter 3

Concluding Remarks

3.1 Conclusion

The background of the project was to optimize the design of the multirotor from a mechatronic perspective. The study of mechatronics is about *system knowledge* and the *collaboration* of the different hardware and a control system. To optimize the design it requires knowledge about the multirotor, what are the weaknesses and what are the strong sides. It was experimented with the sensors of the multirotor to gain knowledge about the entire system. As a result a calibration method was proposed to get more correct information from the IMU. With working sensors, a control system with attitude estimation was proposed. The attitude estimation and attitude controller were able to stabilise the multirotor, even at relatively aggressive manoeuvres. From the testing of the controller it was found that the actuators of the multirotor were relatively slow. From a control perspective, the slow rate of change in thrust and torque of the actuator makes it a weak spot. It is also non-linear as thrust is a square function of the rotation velocity. On the other hand the design is very robust and from the mechanic design perspective it is one of the strongest sides of the multirotor, especially compared to a relatively complicated helicopter. Based on those findings it was decided to investigate the actuator further.

The actuator system was examined theoretically and experimentally. It was found that the thrust response was increased when the motor was run in a complementary mode. This mode also made the motor accelerate and decelerate almost

identical, thus linearising the system. The findings made it possible to linearise the actuator system for given thrust range. With a linear actuator model a multirotor model could be represented as a MILP problem. The optimization solver, that solves the MILP system, does not optimize the actuators directly, but selects the optimal motor, propeller and battery combination for a given design-criteria from the available hardware. The solver also gives the number of actuators to use in order to get the best performance (quad, hexa, octo). Three different design cases were optimized with different outcomes. The hardware chosen by the solver was tested to verify the system response in terms of actuator dynamics and flight time. The calculations and experiments were quite similar thus verifying the system.

Overall, the methods and results presented in this thesis will aid the engineer when designing a multirotor system consisting of control system, mechanical frame, battery, actuators and propellers.

3.2 Future Work

There are some improvements that should be incorporated to the optimization routine. For a 3-cells battery the fully charged voltage is $12.6V$. This voltage will decrease below $11V$ when discharged. The optimization routine is using a simplified version of the battery, where the voltage is set to constant $12V$. A battery model should be implemented to calculate the parameters more accurately. Especially the flight time is affected by this, but also other variables such as the actuator dynamics, payload capacity etc. In addition to a battery model, the optimizer should be able to calculate the battery capacity needed in terms of multiples of a given small battery instead of choosing batteries from a list. This is possible since there is a linear relationship between the weight of the battery and its capacity. The optimizer is using a fixed value of the propeller constants (C_p, C_t) even though they change with RPM. The varying constants were considered to not change significantly and thus used as a single constant for the entire RPM range. A more complex improvement is to implement a dynamic response of the multirotor. The present model is assumed to be more or less static. This assumption holds for slow hovering flight, but not for more dynamic motions.

The attitude estimation does not estimate the yaw angle (heading) of the multirotor. This should be implemented if autonomous flight is desired. Yaw estimation was implemented and tested in simulations, but is not a part of this thesis.

REFERENCES

[Bemporad and Morari, 1999] Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427.

[Brescianini et al., 2013] Brescianini, D., Hehn, M., and D'Andrea, R. (2013). Quadrocopter pole acrobatics. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3472–3479. IEEE.

[Cutler et al., 2011] Cutler, M., Ure, N. K., Michini, B., and How, J. P. (2011). Comparison of fixed and variable pitch actuators for agile quadrotors. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, Portland, OR. (AIAA-2011-6406).

[Davis et al., 2013] Davis, E., Nizette, B., and Yu, C. (2013). Development of a low cost quadrotor platform for swarm experiments. In *Control Conference (CCC), 2013 32nd Chinese*, pages 7072–7077.

[Dunn, 2011] Dunn, F. (2011). *3D math primer for graphics and game development*. A K Peters/CRC Press, Boca Raton, FL.

[Freddi et al., 2014] Freddi, A., Longhi, S., Monteriu, A., and Prist, M. (2014). Actuator fault detection and isolation system for an hexacopter. In *Mechatronic and Embedded Systems and Applications (MESA), 2014 IEEE/ASME 10th International Conference on*, pages 1–6.

[Hall et al., 2008] Hall, J., Knoebel, N., and McLain, T. (2008). Quaternion attitude estimation for miniature air vehicles using a multiplicative extended kalman filter. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pages 1230 –1237.

[Hoffmann et al., 2007] Hoffmann, G. M., Huang, H., Waslander, S. L., and Tomlin, C. J. (2007). Quadrotor helicopter flight dynamics and control: Theory and experiment.

[Kelly, 2013] Kelly, A. (2013). *Mobile robotics : mathematics, models and methods*. Cambridge University Press, New York, NY, USA.

[Li and Li, 2011] Li, J. and Li, Y. (2011). Dynamic analysis and pid control for a quadrotor. In *Mechatronics and Automation (ICMA), 2011 International Conference on*, pages 573–578.

[Mahony et al., 2005] Mahony, R., Hamel, T., and Pflimlin, J.-M. (2005). Complementary filter design on the special orthogonal group so(3). In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 1477–1484.

[Mignone, 2002] Mignone, D. (2002). The really big collection of logic propositions and linear inequalities. Technical Report AUT01-11, ETH Zurich.

[Mokhtari et al., 2005] Mokhtari, A., Benallegue, A., and Daachi, B. (2005). Robust feedback linearization and gh-infinity controller for a quadrotor unmanned aerial vehicle. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 1198–1203.

[Phillips, 2004] Phillips, W. (2004). *Mechanics of flight*. Wiley, Hoboken, N.J.

[Sciavicco, 1996] Sciavicco, L. (1996). *Modeling and control of robot manipulators*. McGraw-Hill Companies, Inc, New York.

[Sidea et al., 2014] Sidea, A. G., Brogaard, R. Y., Andersen, N. A., and Ravn, O. (2014). General model and control of an n rotor helicopter. *Journal of Physics: Conference Series*, 570(5):052004.

[Tayebi and McGilvray, 2006] Tayebi, A. and McGilvray, S. (2006). Attitude stabilization of a vtol quadrotor aircraft. *Control Systems Technology, IEEE Transactions on*, 14(3):562–571.

[Vince, 2010] Vince, J. (2010). *Mathematics for computer graphics*. Springer, London New York.

[Whitaker, 2005] Whitaker, J. (2005). *The electronics handbook*. CRC Press, Boca Raton, FL.

Paper A

Calibration Procedure for an Inertial Measurement Unit Using a 6-Degree-of-Freedom Hexapod

Øyvind Magnussen, Morten Ottestad and Geir Hovland.

This paper has been published as:

Øyvind Magnussen, Morten Ottestad and Geir Hovland.*International Conference on Unmanned Aircraft Systems (ICUAS).* Philadelphia, USA, June 12-15, 2012.

Calibration Procedure for an Inertial Measurement Unit Using a 6-Degree-of-Freedom Hexapod

Øyvind Magnussen, Morten Ottestad and Geir Hovland.
Department of Engineering
Faculty of Engineering and Science, University of Agder
Jon Lilletunsvei 9, 4879 Grimstad, Norway.

Abstract — **In this paper a calibration procedure for an Inertial Measurement Unit (IMU) mounted on an Unmanned Aerial Vehicle (UAV) is presented. Calibration of the sensor when it is mounted on the UAV is attractive because it combines calibration of the internal sensor parameters with the translational and rotational offsets of the IMU relative to the body frame of the UAV. A stepwise calibration procedure is presented, based on motion profiles and measurements generated by a 6-degree-of-freedom hexapod. The experimental results demonstrate that the linear acceleration and angular velocity measurements of the IMU sensor ADIS16400 can be calibrated to typically 0.17% and 0.25% of the ADIS16400 measurement range for linear acceleration and rotational velocity, respectively.**

1 Introduction

Unmanned aerial vehicles (UAVs) are often designed with an inherent instability and such systems can only be stabilized by feedback controllers using accurate sensors. In order to achieve high performance and robustness of such systems, it is essential that the sensor measurements are as accurate as possible. A typical sensor used in UAVs is an Inertial Measurement Unit (IMU) which measures linear accelerations and angular velocities in three dimensions. The IMU calibration is considered as one of the main challenges in inertial navigation. Even if highly accurate IMUs are available from sensor manufacturers it is still desirable to perform an independent calibration and verification of the performance against a more accurate measurement device. It is also desirable to calibrate the IMU after it has been mounted on the UAV, since the actual location and orientation of the sensor relative

to the body frame of the UAV are additional error sources in the models and the feedback control algorithms. To ease the attitude calculations the IMU is often put in the center of gravity (CG) of the UAV with alignment of the sensor axes with the body frame. But UAVs, especially quadrotors, have limited space available and it can be difficult to place the IMU at the desired position.

In this paper an approach for calibration and verification of an IMU mounted on a UAV is presented. The approach uses the motion and sensing capabilities of a 6-degree-of-freedom (6-DOF) hexapod. By using a hexapod, it is possible to generate pure translational or rotational motion in selected directions, and this feature is exploited in a stepwise calibration procedure presented in this paper. Sensor parameters such as offsets, scaling factors, the alignment of the sensor's internal coordinate axes as well as the translational and rotational transformation relative to the UAV's body frame are identified.

Many authors have proposed different methods for IMU calibration, see [Sahawneh and Jarrah, 2008] and the references therein. Common for these methods is the utilization of the fact that ideally the norm of the measured output of the accelerometer and gyro cluster is equal to the magnitude of the applied force and rotational velocity. As noted in [Sahawneh and Jarrah, 2008] the main drawback of this approach is that not all the sensor parameters of the IMU are observable. This in turn implies that these uncalibrated parameters must be taken into account in the integration of the IMU in the UAV, for example with advanced filtering algorithms. There are also papers describing the process of calculating the new CG of the UAV if its shifted to a new location during operation [Mellinger et al., 2011]. Algorithms aligning two IMU in two reference frames has also been established [Shortelle et al., 1998]. But this does not count for misalignment of the internal axis-to-axis or scaling of the IMU. The research topic is not new, and old articles of how to automatically align an IMU relative to the earth [Hung and White, 1975] is also developed.

The main benefit of the work presented in this paper, is the flexibility of the hexapod to generate motions in selected directions. Hence, all the parameters of the IMU can be calibrated and the need for complex, online filtering techniques in the UAV real-time controller is reduced.

2 System Description

2.1 Inertial Measurement Unit

An IMU, ADIS16400 from Analog Devices, measures acceleration a_I and angular velocity ω_I for the origin of the IMU frame for the three axes X_I, Y_I, Z_I as illustrated in Fig. A.1. The subscripts I refer to the IMU's coordinate axes. In this paper, if the subscript does not begin with I, the variable is always referred to the body frame. In

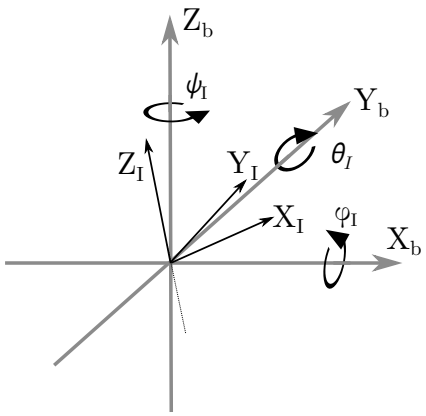


Figure A.1: *IMU frame relative to the body frame*

general, depending on how the IMU is mounted on the UAV, there will be rotations of the IMU's coordinate axes relative to the body axes (ϕ_I, θ_I, ψ_I), illustrated in Fig. A.1.

In Fig. A.2 the origin of the body frame is located at CG of the UAV. This is also the center of rotation for any rotational motion. The IMU frame is located off-center.

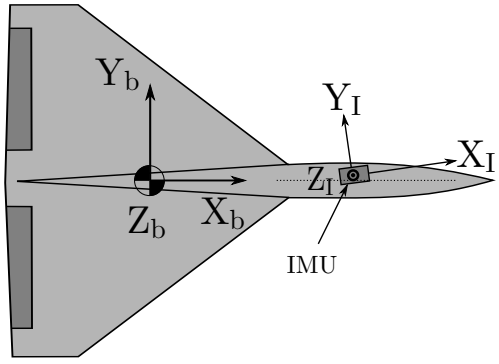


Figure A.2: *Rotation and translation of IMU frame relative to body frame.*

A off-center rotation of the IMU relative to the body frame, as in Fig A.3, will result in additional accelerations perpendicular to the rotational axis, i.e. a radial acceleration a_R and a tangential acceleration a_T.

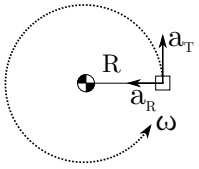


Figure A.3: *Radial and tangential accelerations when IMU is mounted off-center compared to body frame.*

These accelerations specified in the body frame are described by [Ardema, 2005] and [Shabana, 1998]:

$$a_R = \omega_b \times (\omega_b \times R) \tag{A.1}$$

$$a_T = \alpha_b \times R \tag{A.2}$$

where $\times$ is the vector cross product, $R = [R_x\ R_y\ R_z]^T$ is the distance from the origin of the body frame to the IMU frame, $\omega_b = [\omega_x\ \omega_y\ \omega_z]^T$ is the angular velocity of the body frame and $\alpha_b = [\alpha_x\ \alpha_y\ \alpha_z]^T$ is the angular acceleration of the body frame. The relationship between the various accelerations are

$$a_b + a_R + a_T + a_g = K_a \cdot a_I \tag{A.3}$$

$$a_I = \widetilde{a_I} - a_{I,n} - a_{I,o} \tag{A.4}$$

where a_I is the raw acceleration measurement from the IMU, including noise $a_{I,n}$ and bias $a_{I,o}$ in the IMU frame, $K_a \in \mathbb{R}^3$ is a gain and cross correlation matrix, a_b is the translation acceleration of the body frame, a_g is the gravity vector in the body frame. Ideally, the matrix K_a is a pure rotational matrix [Beard and McLain, 2012] given by the three angles ϕ_I, θ_I, ψ_I. However, if the IMU's coordinate axes X_I, Y_I, Z_I have an axis-to-axis misalignment, or the scaling of at least one of the axes are different than the others K_a will no longer be a proper rotational matrix
[Greval and Andrews, 2008].
The IMU's gyroscopes measure the angular rates ω_I, which are not affected by the

location of the IMU relative to the CG, only its orientation given by ϕ_I, θ_I, ψ_I.

$$\omega_b = K_\omega \cdot \omega_I \tag{A.5}$$

$$\omega_I = \widetilde{\omega_I} - \omega_{I,o} - \omega_{I,n} \tag{A.6}$$

where K_ω is the gain and cross correlation matrix ω_b is the angular rate of the body frame, $\omega_{I,n}$ is the noise of the gyroscope, $\omega_{I,o}$ is the gyroscope bias and $\widetilde{\omega_I}$ is the gyroscope measurement including noise and bias.

2.2 Stewart Platform

A Stewart Platform shown in Fig. A.4 is used to calibrate the IMU. The Stewart Platform has six degrees of freedom and can manipulate rotations and translations. The Stewart Platform is also able to set the center of rotation to a specific location. This makes it possible to mount any UAV on top of it, and to align the Stewart Platforms center of rotation with the UAV's CG. Different types of motion profiles with different frequencies and amplitudes were generated. Six degrees of freedom data were recorded for positions, velocities and accelerations in the fixed world frame. These measurements were transformed to the UAV body frame.

2.3 Laser Tracker

To verify the motion of the Stewart platform a FARO laser tracker as shown in Fig. A.5 was used to verify its position. In the verification it was found that the accuracy of the internal measurements of the Stewart Platform were comparable with the laser tracker, typically in the range 10-30 μm. Hence, in the experimental results presented in this paper, only the internal measurements from the Stewart Platform were used.

3 Calibration Concept

The UAV is mounted on top of the Stewart Platform with aligned axes and CG, shown in FIG A.6.

Figure A.4: *Stewart Platform used for IMU calibration experiments.*

The calibration consists of three different stages.

A) Remove offsets

B) Calculate scaling and correlation matrices

C) Find the location of the inertial measurement units R_x, R_y and R_z positions

Figure A.5: *FARO laser tracker used to verify the internal measurements of the Stewart Platform.*

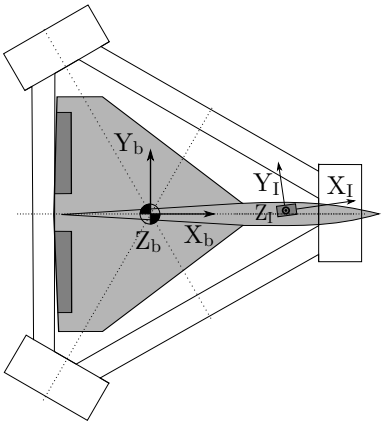


Figure A.6: *UAV mounted for calibration*

3.1 Offset Calibration

The gyroscope bias are removed by calculating the average output over a period of time, with the gyroscope in a fixed position. The bias is subtracted from the gyroscope as given by (A.6). Acceleration bias can be calculated by taking two measurements, where one measurement is taken while the sensor is turned 180 degrees compared to the first measurement, this is however not done in this paper.

3.2 Scaling and Correlation

In order to calculate the two matrices K_a and K_ω, two separate tests had to be done, where the first test calculates K_a and the second calculates K_ω. The series of measurement data were synchronized in time for both the Stewart Platform and the

IMU. The motion to calculate K_a consists of rapid translative motions, combined with a slow rotation around every body axis. The translative motions manipulate the accelerometer directly, while the slow rotations makes the gravity affect each axis of the IMU dynamically. The Stewart Platform has a force limit for each actuator which in turn limits the maximum acceleration. Adding the rotating gravity vector relative to the IMU increases the acceleration measured by the accelerometer and hence the the signal-noise ratio. The rotation frequency of 0.01Hz is very low, as a result a_g in eq.(A.3) dominates a_R and a_T which can be neglected from the equations.

The motion to calculate K_ω was a pure rotation about the body axes X_b, Y_b, Z_b. The amplitude and frequencies were higher than for K_a resulting in larger rotational velocities. Typical frequencies generated by the Stewart Platform were in the range 0.6-0.7 Hz with an amplitude of 10 degrees.

For both the accelerometer and gyroscope the problem can be written on the form:

$$\Omega_b = K \cdot \Omega_I \tag{A.7}$$

where Ω_b is a $3 \times n$ matrix with the true datasets from the Stewart Platform, $K \in \mathbb{R}^3$ is the correlation and scaling matrix, Ω_I is a $3 \times n$ matrix with the measured data, either acceleration or rotational velocity, from the IMU and n is the number of measurement samples. The problem is to find the best fit for K over the data sets. As noted in [Sayed, 2012], several optimization criteria have been used for estimation purposes over the years, but the most important ones, at least in the sense of having had the most applications, are criteria that are based on quadratic cost functions. The most common among these is the linear least-squares criterion.

Each column in (A.7) represents a data set, X, Y and Z respectively. The objective of the least-squares problem is to determine the vectors in the column space of K that is closest to the vectors in Ω_b in the least-squares sense.

$$\min_{\Omega_I} ||\Omega_b - K\Omega_I||^2 \tag{A.8}$$

The final solution to the problem is:

$$\begin{aligned} \Omega_b &= K \cdot \Omega_I \\ \Omega_b \cdot \Omega_I^T &= K \cdot \Omega_I \cdot \Omega_I^T \\ K &= \Omega_b \cdot \Omega_I^T \cdot (\Omega_I \cdot \Omega_I^T)^{-1} \\ K &= \Omega_b \Omega_I^+ \end{aligned} \tag{A.9}$$

where Ω_I^+ is the Moore-Penrose pseudoinverse and will exist and be unique for any Ω_I [Golub and Loan, 1996]. This approach (A.9) is used in order to calculate the two matrices K_a and K_ω.

3.3 Location of the Inertial Measurement Unit

The last step is to calibrate the position $R = [R_x\ R_y\ R_z]^T$ of the IMU relative to CG. The distance vector R can be calculated from (A.1) and (A.2) by rotating around one axis at a time. Rotating around the body frame x-axis yields:

$$\begin{aligned} a_{b,x} &= 0 \\ a_{b,y} &= -\alpha_{b,x} \cdot R_z - \omega_{b,x}^2 \cdot R_y \\ a_{b,z} &= \alpha_{b,x} \cdot R_y - \omega_{b,x}^2 \cdot R_z \end{aligned}$$

and in matrix form:

$$\begin{bmatrix} a_{b,x} \\ a_{b,y} \\ a_{b,z} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ -R_z & -R_y \\ R_y & -R_z \end{bmatrix} \cdot \begin{bmatrix} \alpha_{b,x} \\ \omega_{b,x}^2 \end{bmatrix} \tag{A.10}$$

Solving the equations using the approach from eq.(A.9) resolves two sets of R_y and R_z, because of the structure of the matrix in eq.(A.10). Limitations of the Stewart Platform results in a very low value of ω^2 compared to α. This makes the estimates of R_y, R_z which are multiplied with the angular acceleration α_x more trustworthy than the same estimates multiplied with the angular velocity ω_x^2. Because of this result the estimates multiplied with the accelerations are kept as the final calibrations.

The resulting equations are:

$$\begin{bmatrix} a_{b,x} \\ a_{b,y} \\ a_{b,z} \end{bmatrix} = \begin{bmatrix} 0 \\ -R_z \\ R_y \end{bmatrix} \cdot \alpha_{b,x} \tag{A.11}$$

Doing this for each of the three axes results in two calculations for each distance. If the two datasets are identical it is a verification of the test method of the IMU.

4 Experiments

4.1 Remove Bias

The first experiment was to remove the gyro offsets. The offset was calculated as the mean value of 10.000 samples:

$$\omega_{I,o} = \begin{bmatrix} -0.0043 \\ 0.0010 \\ 0.0048 \end{bmatrix} \frac{rad}{sec} \tag{A.12}$$

4.2 Scaling and Correlation

The scaling and cross correlation matrix of the gyroscopes were then calculated. For the method to be able to separate the data the test has to be as complex as possible. Running a sine function at each body axis with a the same frequency, only phase shifted, will not result in a correct calculation of the K-matrix. For instance doing a test with the same frequency for each body axis only with a $\frac{\pi}{2}$ rad phase on the y-axis will not result in correct calculations. Assuming that the IMU reference frame is perfectly aligned with the body frame, and the IMU is perfectly scaled and has perpendicular axes, the method will not be able to separate the rotation around

the x and the y-axis. The resultant K-matrix will be

$$\begin{bmatrix} \omega_{b,x} \\ \omega_{b,y} \\ \omega_{b,z} \end{bmatrix} = \begin{bmatrix} 0.5 & 0.5 & 0 \\ 0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \omega_{I,x} \\ \omega_{I,y} \\ \omega_{I,z} \end{bmatrix}$$

and if the phase difference of $\omega_{b,x}$ and $\omega_{b,y}$ is π rad, the rotation of the X_b- and Y_b-axis will be opposite and a negative sign will occur in the correlation matrix:

$$\begin{bmatrix} \omega_{b,x} \\ \omega_{b,y} \\ \omega_{b,z} \end{bmatrix} = \begin{bmatrix} 0.5 & -0.5 & 0 \\ -0.5 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \omega_{I,x} \\ \omega_{I,y} \\ \omega_{I,z} \end{bmatrix}$$

In order to overcome this, a sine-function with different frequencies were used.

$$\begin{aligned} \omega_{b,x} &= A \sin(2\pi f_x t) \\ \omega_{b,y} &= A \sin(2\pi f_y t) \\ \omega_{b,z} &= A \sin(2\pi f_z t) \end{aligned}$$

where the amplitude $A = 4$ degrees and the frequency $f_x = 0.6$ Hz, $f_y = 0.65$ Hz, $f_z = 0.7$ Hz. Using the approach from (A.9) with a number of 10.000 sampled datasets from the Stewart Platform and the IMU gives:

$$K_\omega = \begin{bmatrix} 0.95 & 0.29 & 0.01 \\ -0.29 & 0.95 & 0.01 \\ -0.01 & -0.01 & 1.00 \end{bmatrix} \tag{A.13}$$

The same approach was followed for the accelerometer. The resultant K_a matrix is:

$$K_a = \begin{bmatrix} 0.97 & 0.27 & 0.02 \\ -0.30 & 0.97 & -0.01 \\ 0.01 & 0.01 & 1.00 \end{bmatrix} \tag{A.14}$$

A result of the measured angular velocity w_I multiplied with the correction matrix K_ω is shown in Fig. A.7, and for the accelerometer in Fig. A.8. The figures are for

a different dataset than for the calculation of K_ω and K_a.

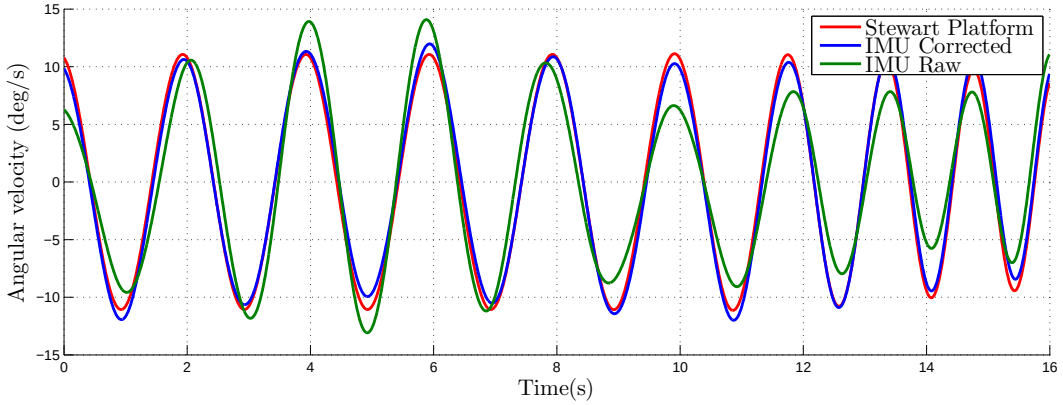


Figure A.7: *Gyroscope scaling and correction for $\omega_{b,x}$, $\omega_{I,x}$ corrected and $\omega_{I,x}$ raw.*

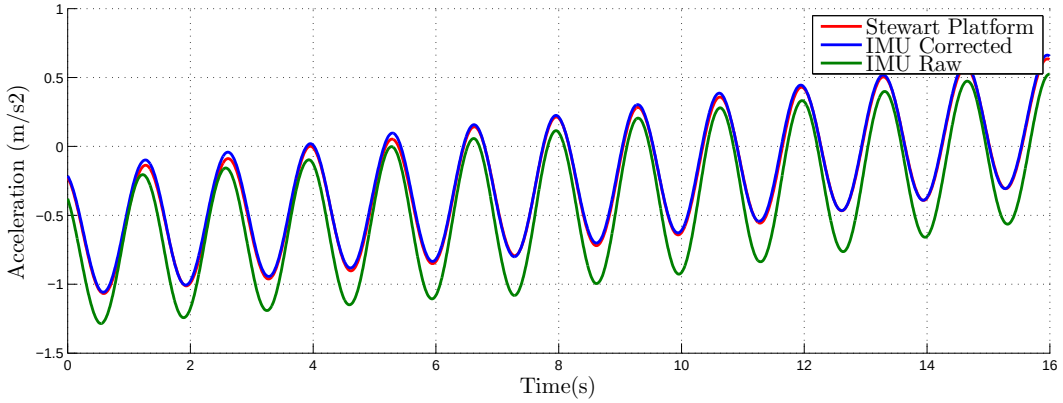


Figure A.8: *Accelerometer scaling and correction for $a_{b,x}$, $a_{I,x}$ corrected and $a_{I,x}$ raw.*

The IMU's orientation was measured to be:

$$\phi_I = 0\ deg$$
$$\theta_I = 0\ deg$$
$$\psi_I = 17\ deg$$

Assuming the IMU is ideal (scaling is perfect and the axis-to-axis misalignment is zero) then the resultant K-matrices would be a rotation matrix. A XYZ rotation

matrix is defined as [Siciliano et al., 2011] and [Genta, 2012]:

$$C = \begin{bmatrix} c\theta_I c\psi_I & s\phi_I s\theta_I c\psi_I - c\phi_I s\psi_I & c\phi_I s\theta_I c\psi_I + s\phi_I s\psi_I \\ c\theta_I s\psi_I & s\phi_I s\theta_I s\psi_I + c\phi_I c\psi_I & c\phi_I s\theta_I s\psi_I - s\phi_I c\psi_I \\ -s\theta_I & s\phi c\theta_I & c\phi_I c\theta_I \end{bmatrix} \tag{A.15}$$

where C is the rotation matrix from the IMU frame to the body frame, c is the cosine function and s is the sine function. Inserting the measured values in the rotation matrix C yields:

$$C = \begin{bmatrix} 0.96 & -0.29 & 0 \\ 0.29 & 0.96 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.16}$$

The measured correction matrices are very close to the ideal one, K_ω is the one which is closest. There are many reasons why K_a is a bit off. One reason could be the scaling, and correlation are different for the accelerometer than for the gyroscope. The ADIS16400 is however a high-end IMU with less than 0.2 degrees misalignment of the internal axes. Another issue could be that the accelerometer has a range of ± 18 g, which makes it impossible for the Stewart Platform to use the whole range. The range of the excited acceleration to the IMU compared to the accelerometer noise is also significantly less than for the gyroscopes.

4.3 Location of the Inertial Measurement Unit

The result of K_a from (A.14) was used in order to correct the acceleration measurement when calculating the position of the IMU, R, as in (A.3). The Stewart Platform was run in three steps, where each step was a rotation around the X_b, Y_b, Z_b axis respectively. The amplitude was 5 degrees and with a frequency of 0.7 Hz. The same approach as for the calculation of K_a and K_ω (A.9) was used to calculate R. According to (A.11) two sets of each distance will be calculated. The results are displayed in table A.1:

Table A.1: *Result of calculation of the distances R_x, R_y and R_z, results are in meters (m)*

Rotation axis	R_x	R_y	R_z
ω_x :	–	−0.237	0.227
ω_y :	0.362	–	0.233
ω_Z :	0.362	−0.239	–

The IMU's position was measured in meters to be:

$$R = \begin{bmatrix} 0.365(m) \\ -0.235(m) \\ 0.230(m) \end{bmatrix}$$

A comparison of the measured values with the calculated ones, shows that the error range is less than 5 mm.

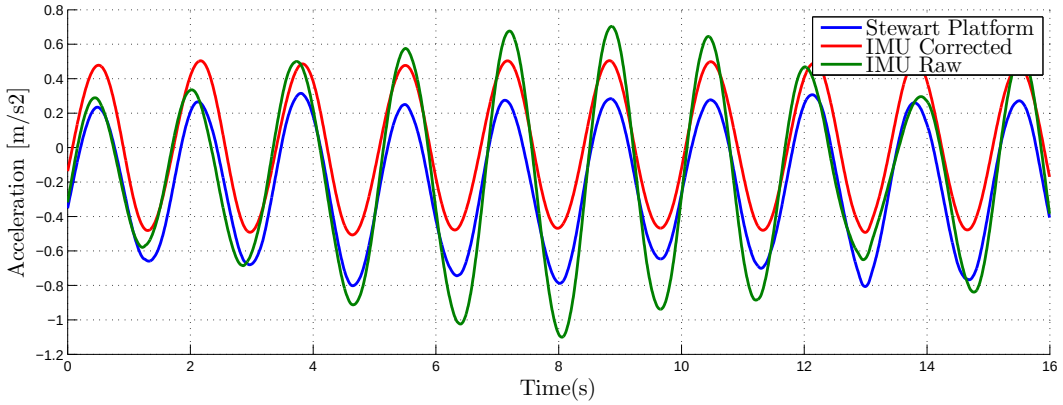
4.4 Full system test

A full system test was done in order to verify the calculated correction matrices and distances. K_a, K_ω and the calculated distances R_x, R_y, R_z was used according to (A.3) with rotation about the three body axes X_b, Y_b, Z_b:

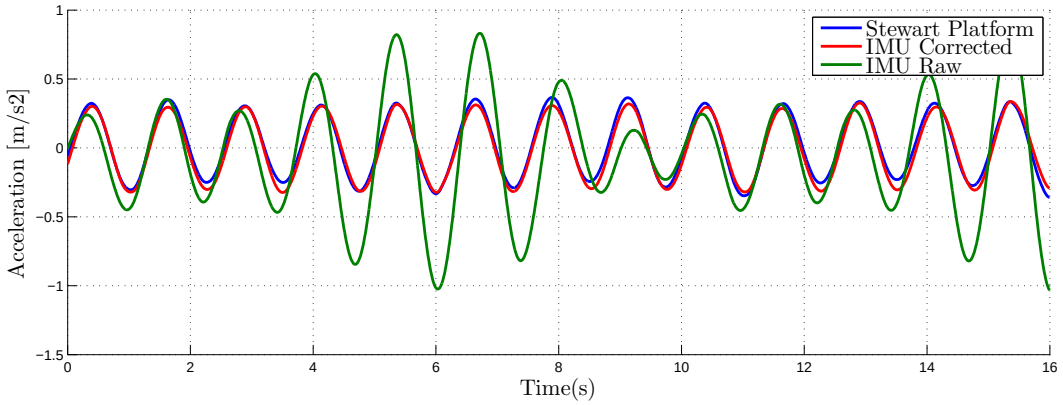
$$\omega_{IC} = K_\omega \cdot \omega_I \tag{A.17}$$

$$a_b + a_g = K_a \cdot a_I - \omega_{IC} \times (\omega_{IC} \times R) - \dot{\omega_{IC}} \times R \tag{A.18}$$

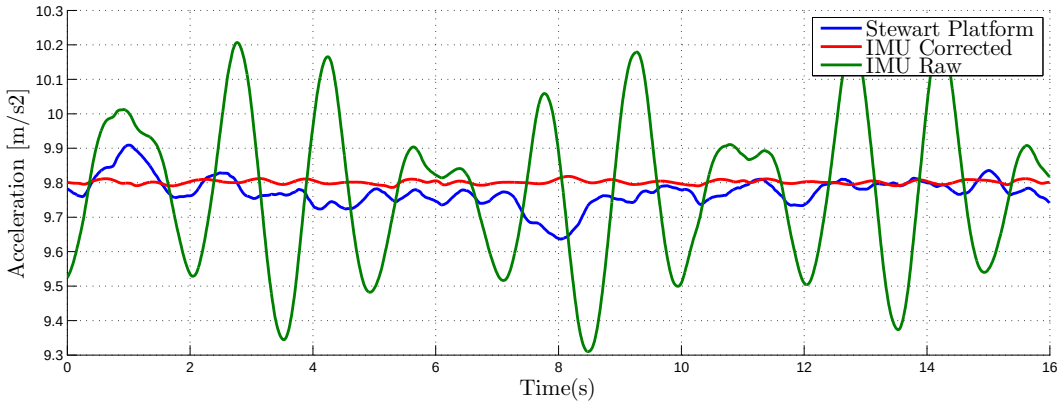
where ω_{IC} is the corrected rotational velocity measured by the gyroscope and $\dot{\omega_{IC}}$ is the time derivative of the rotational velocity.
The results are shown in Fig. A.9



(a) 1



(b) 2



(c) 3

Figure A.9: *Comparison of the acceleration from the Stewart Platform, the corrected acceleration from the IMU and the non-corrected acceleration from IMU, for a_x, a_y and a_z respectively*

The root-mean-square (RMS) of the signals is a verification of how good the estimates are as a mean. The RMS defined below is an overall verification of how close the estimated values are to the true signal. The signal from the Stewart Platform is subtracted from the signals from the IMU. The difference is squared so that the negative signals will be positive. The RMS is defined as:

$$RMS = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_I - x_J)^2} \tag{A.19}$$

where n is the number of samples, x_I is the IMU sampling data and x_J is the Stewart Platform sampling data.

Table A.2: *Results of the RMS calculation for the corrected data set*

a_x	0.264	
a_y	0.058	(m/s^2)
a_z	0.177	
ω_x	0.0123	
ω_y	0.0123	$(^o/s)$
ω_z	0.0107	

5 Conclusions

This paper has demonstrated a stepwise calibration method for an ADIS16400 IMU mounted on a UAV by using a 6-DOF Hexapod motion generator. Separate motion profiles were used to calibrate different types of IMU parameters, such as offsets, scaling, axis-to-axis misalignment as well as translational and rotational offsets relative to the body frame of the UAV. The experimental results show an achieved accuracy of typically 0.1-0.3 m/s^2 for the linear accelerations and typically 0.01-0.015 rad/s for the rotational velocities during a test motion profile with motion frequencies of 0-1Hz, 5^o angular range and 0.1m positional range. The test motion profile was independent from the motion profiles used for calibration. The achieved accuracies correspond to 0.17% and 0.25% of the ADIS16400 measurement range for linear acceleration and rotational velocity, respectively. The work presented in

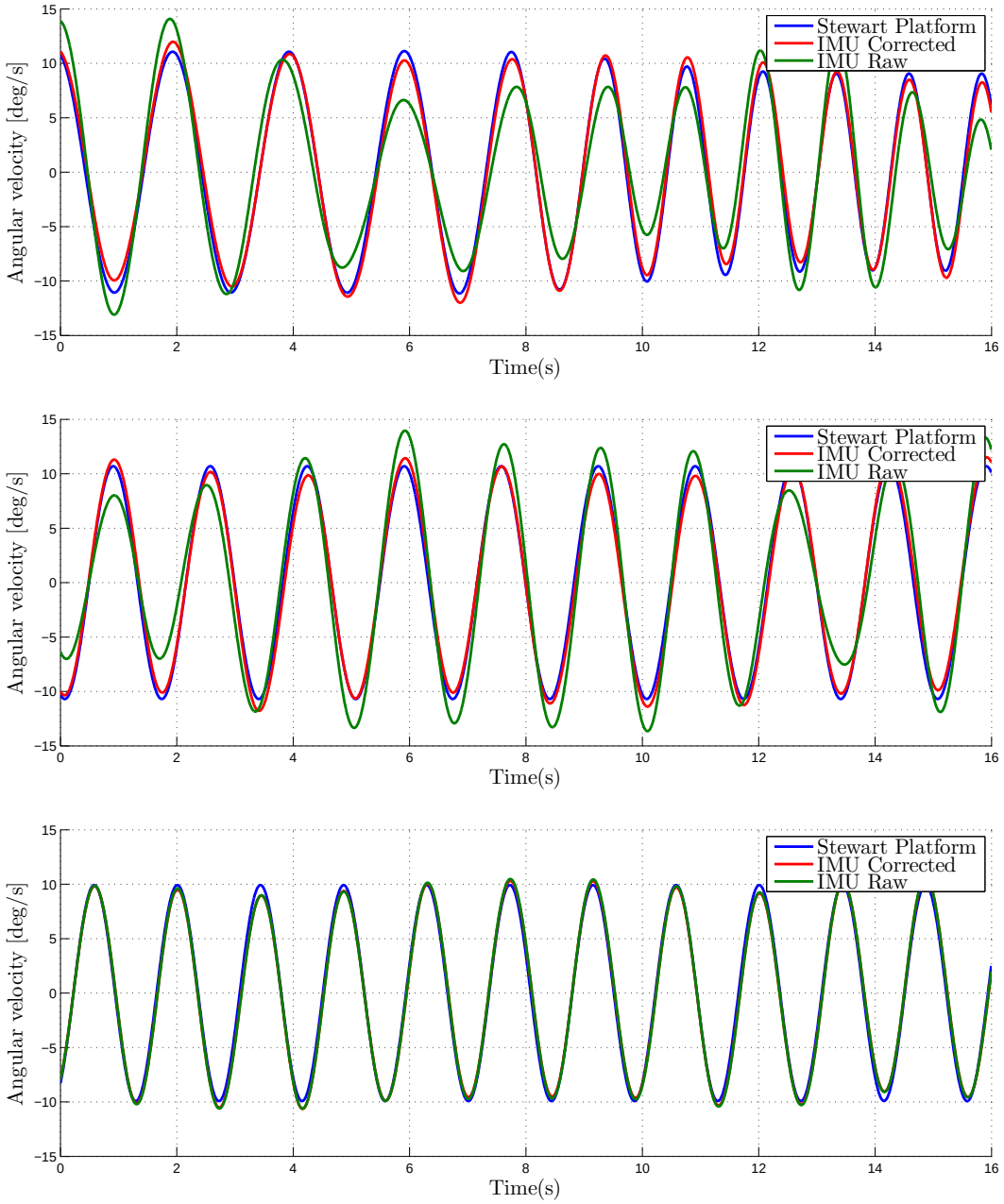


Figure A.10: *Comparison of the rotational velocity from the Stewart Platform, the corrected rotational velocity from the IMU and the non-corrected rotation velocity from IMU, for ω_x, ω_y and ω_z respectively*

this paper demonstrates that a 6-DOF hexapod is well suited for calibration of an IMU when mounted on a UAV.

REFERENCES

[Ardema, 2005] Ardema, M. (2005). *Newton-Euler Dynamics*. Springer.

[Beard and McLain, 2012] Beard, R. and McLain, T. (2012). *Small Unmanned Aircraft Theory and Practice*. Princeton University Press.

[Genta, 2012] Genta, G. (2012). *Introduction to the Mechanics of Space Robots*. Springer.

[Golub and Loan, 1996] Golub, G. and Loan, C. V. (1996). *Matrix Computations, 3rd ed.* The Johns Hopkins University Press.

[Greval and Andrews, 2008] Greval, M. and Andrews, A. (2008). *Kalman Filtering Theory and Practice using MATLAB*. Wiley-IEEE Press.

[Hung and White, 1975] Hung, J. and White, H. (1975). Self-alignment techniques for inertial measurement units. *Aerospace and Electronic Systems, IEEE Transactions on*, AES-11(6):1232 –1247.

[Mellinger et al., 2011] Mellinger, D., Lindsey, Q., Shomin, M., and Kumar, V. (2011). Design, modeling, estimation and control for aerial grasping and manipulation. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2668 –2673.

[Sahawneh and Jarrah, 2008] Sahawneh, L. and Jarrah, M. (2008). Development and calibration of low cost mems imu for uav applications. In *Mechatronics and Its Applications, 2008. ISMA 2008. 5th International Symposium on*, pages 1 –9.

[Sayed, 2012] Sayed, A. (2012). *Digital Signal Processing Fundamentals*. CRC Press.

[Shabana, 1998] Shabana, A. (1998). *Dynamics of Multibody Systems , 2nd ed.* Cambridge Unversity Press.

[Shortelle et al., 1998] Shortelle, K., Graham, W., and Rabourn, C. (1998). F-16 flight tests of a rapid transfer alignment procedure. In *Position Location and Navigation Symposium, IEEE 1998*, pages 379 –386.

[Siciliano et al., 2011] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G. (2011). *Robotics Modelling, Planning and Control*. Springer.

Paper B

Experimental Validation of a Quaternion-based Attitude Estimation with Direct Input to a Quadcopter Control System

Øyvind Magnussen, Morten Ottestad and Geir Hovland.

This paper has been published as:

Øyvind Magnussen, Morten Ottestad and Geir Hovland.*IEEE International Conference on Unmanned Aircraft Systems (ICUAS).* Atlanta, USA, May 28-31, 2013.

Experimental Validation of a Quaternion-based Attitude Estimation with Direct Input to a Quadcopter Control System

Øyvind Magnussen, Morten Ottestad and Geir Hovland.
*Department of Engineering
Faculty of Engineering and Science, University of Agder
Jon Lilletunsvei 9, 4879 Grimstad, Norway.

***Abstract* —**

This paper presents a method to calculate the attitude quaternion of a quadcopter with few calculations. The quaternion calculation is based on accelerometers and gyroscopes from an Inertial Measurement Unit (IMU). The quaternion from the accelerometer is calculated as the shortest rotation arc from the gravity vector in the navigation frame. The quaternion from the gyroscope is calculated based on equations of the quaternion derivative. A complementary filter is combining the two quaternions with a componentwise comparison. The attitude estimation is calculated without any trigonometric functions. The quaternion is directly used as an input to the attitude controller. The attitude controller is a PD controller running at 400Hz. A model of the quadcopter in Matlab verified that the control system worked as intended. The estimator was verified with a Stewart platform, by mounting the quadcopter on top of it and comparing the angles from the Stewart platform with the angles from the filter. Finally the algorithms were implemented on a quadcopter controller board, and the attitude estimator were compared with the attitude estimation from a high-end IMU from MicroStrain. The complete control system was also tested on a 8-bit microcontroller running at 16 MHz. The relatively slow processor on the microcontroller was also able to do every calculations within 2.5ms.

1 Introduction

Quadcopters are popular for hobby-enthusiasts as well as academic research in control of Unmanned Aerial Vehicle (UAVs) due to their simple and low-cost design. Industrial use is currently limited, but potential applications are for example search-and-rescue, surveillance, movie recording, etc.

The quadcopter is an interesting test-bed for academic research since the dynamics is open-loop unstable and a feedback controller with an attitude estimator is required to stabilize and operate it. Such a controller must be able to handle all six degrees of freedom (DOF). A common 6-DOF representation consists of the X,Y,Z positions as well as the roll, pitch and yaw (RPY) angles relative to ground. However, the RPY angles suffer from singularities which may become an issue with quadcopters due to the possibility of acrobatic motions with large angles. An alternative 6-DOF representation is to use the X,Y,Z positions combined with a quaternion to represent the orientation. Quaternions are also used due to the reduced number of floating-point calculations which is a benefit on low-cost, low-weight microcontroller hardware.

Quaternion estimation of the attitude is often done by a variety of Kalman filters. The attitude estimation in [Hall et al., 2008] is computed using a multiplicative extended Kalman filter. The estimation is based on information from accelerometers, gyroscopes and a GPS. In [de Marina et al., 2012] an unscented Kalman filter is proposed. Both of the filters are efficient, but also require significant of computational power.

Attitude estimation is also used in human body motion tracking [Xiaoping et al., 2008], [Yun and Bachmann, 2006]. These filters rely on slow moving objects and are not suited for UAVs.

In [Stingu and Lewis, 2009] a quaternion estimation for a quadcopter is presented. This is similar to the one presented in this paper, but involves more complicated mathematics. Their control system is also similar to the one in this paper, but involves trigonometric functions to generate the angle setpoints. A complimentary filter design for attitude estimation is also presented in [Mahony et al., 2005]. An advantage of that approach is that the bias of the gyroscope is also estimated, but the filter can be complicated to implement on a microcontroller.

In [Tayebi and McGilvray, 2004], [Tayebi and McGilvray, 2006] a new quaternion

attitude controller for a quadcopter is proposed. As noted the proposed controller is based upon the compensation of the Coriolis and gyroscopic torques and the use of a PD^2 feedback structure, where the proportional action is in terms of the vector quaternion and the two derivative actions are in terms of the airframe angular velocity and the vector quaternion velocity. The quaternion is extracted from the rotation matrix, while in this paper the quaternion is directly generated from the accelerometer and gyroscopes.

In this paper a new approach is taken, which is more computationally efficient to compared other methods. The estimator is based on the combination of sensor data from a gyroscope and an accelerometer combined with a complementary filter. The new approach for estimating the quaternion is used directly in a quaternion-based attitude controller, without the need for any trigonometric functions.

2 Quadcopter Testbed

The quadcopter used for testing is a DJI Flame Wheel F450 quadcopter, Fig. B.1. The total lifting capacity of the DJI is approximately 1600 grams. The control system was implemented MultiWii Mega, a low-cost flight controller from diymulticopter.com.

Figure B.1: *DJI quadcopter used for testing*

The flight controller is equipped with an MPU6050 IMU. The accelerometer is measuring the acceleration of the quadcopter including the gravity. It is assumed that the gravity is much greater than the body accelerations of the quadcopter, and

can be used to estimate the attitude.

The roll and pitch setpoint angles of the DJI are controlled by a RC transmitter, where the receiver is directly connected to the flight controller.

3 Modelling

The quadcopter reference frame is the body frame denoted with a subscript 'b'. The navigation frame is a fixed frame denoted with a subscript 'n'. Positive axes and motor torques are shown in Fig. B.2. The quadcopter is modeled using quaternions

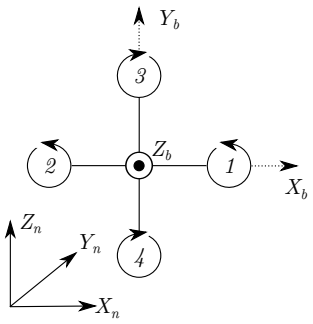


Figure B.2: *Quadcopter model with positive axes and motor torques*

described in [Sidi, 2002]. The quaternion has the benefit of fast calculations and singularity-free representation. The quaternion is defined as:

$$q = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\frac{\beta}{2}) \\ \sin(\frac{\beta}{2})\, u_1 \\ \sin(\frac{\beta}{2})\, u_2 \\ \sin(\frac{\beta}{2})\, u_3 \end{bmatrix} \tag{B.1}$$

where β is the angle to rotate, and $u = [u_1\ u_2\ u_3]^T$ is the unit vector describing the axis to rotate about. It's inverse is defined as

$$q^{-1} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \end{bmatrix}^T \tag{B.2}$$

Multiplication of two quaternions q and r is denoted with the symbol $\otimes$:

$$q \otimes r = \begin{bmatrix} r_0q_0 - r_1q_1 - r_2q_2 - r_3q_3 \\ r_0q_1 + r_1q_0 - r_2q_3 + r_3q_2 \\ r_0q_2 + r_1q_3 + r_2q_0 - r_3q_1 \\ r_0q_3 - r_1q_2 + r_2q_1 + r_3q_0 \end{bmatrix} \tag{B.3}$$

A 3×1 vector v can be rotated in 3D space to the vector v' using a quaternion by extending v to a 4×1 vector by adding a leading zero.

$$v' = q \otimes \begin{bmatrix} 0 \\ v \end{bmatrix} \otimes q^{-1} \tag{B.4}$$

where the new rotated vector v' is the last three elements of the multiplication result.

The derivative of the quaternion is defined as

$$\dot{q} = \frac{1}{2} q \otimes \begin{bmatrix} 0 \\ \omega_b \end{bmatrix} \tag{B.5}$$

where ω_b is a 3×1 vector describing angular velocities in rad/sec.

A conversion from a given quaternion to Euler Angles, roll (ϕ), pitch (θ), yaw (ψ) is defined as

$$\begin{bmatrix} \phi \\ \theta \\ \psi \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\frac{2(q_0q_1+q_2q_3)}{1-2(q_1^2+q_2^2)}\right) \\ \sin^{-1}(2(q_0q_2 - q_3q_1)) \\ \tan^{-1}\left(\frac{2(q_0q_3+q_1q_2)}{1-2(q_2^2+q_3^2)}\right) \end{bmatrix} \tag{B.6}$$

Forces acting on the quadcopter are the gravity F_g in the Z-direction of the navigation frame, propeller thrust F_{bz} in the body Z-direction and the gyroscopic force from the angular velocity ω_b. Blade flapping, wake interaction and any other effects caused by the translational velocity are ignored. Summing forces F_b in the body coordinate system yields:

$$\sum F_b = m\dot{V}_b$$

$$\dot{V}_b = \frac{1}{m}\begin{bmatrix} 0 \\ 0 \\ F_{bz} \end{bmatrix} + q^{-1} \otimes \begin{bmatrix} 0 \\ 0 \\ 0 \\ g \end{bmatrix} \otimes q - \omega_b \times V_b \tag{B.7}$$

$$\dot{V}_n = q \otimes \dot{V}_b \otimes q^{-1} \tag{B.8}$$

$$\tag{B.9}$$

where m is the mass of the quadcopter, $\dot{V}$ denotes the acceleration and V denotes the velocity in the defined frames.

Moments acting on the quadcopter are the one from different motor thrust, rotational torque from the motor due to the propeller and the gyroscopic force. The induced motor torque due to motor acceleration is neglected since it is difficult to measure. Summing the moments M_b in the body coordinate system yields:

$$\sum M_b = I_b \dot{\omega}_b$$

$$\dot{\omega}_b = I_b^{-1}\left(\begin{bmatrix} M_{bx} \\ M_{by} \\ M_{bz} \end{bmatrix} - \omega_b \times (I_b \omega_b)\right) \tag{B.10}$$

where I_b is the inertia of the quadcopter and M_b is the moment about each body coordinate axis from the propeller thrust:

$$\begin{bmatrix} M_{bx} \\ M_{by} \\ M_{bz} \end{bmatrix} = \begin{bmatrix} 0 & 0 & d_m & -d_m \\ -d_m & d_m & 0 & 0 \\ T_q & T_q & -T_q & -T_q \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{B.11}$$

where d_m is the distance from the quadcopters center of gravity to the center of the motors. There is a linear relationship between the thrust of the propellers and the

torque which is produced [Roskam and Lan, 1997], T_q is the coefficient describing this relationship.

4 Attitude Estimation

The proposed attitude estimator is only estimating the roll and pitch angles. The roll and pitch angles are also the most critical angles to control for a stable flight. The attitude estimator is a complementary filter that combines two different quaternions, one from the gyroscopes and one from the accelerometers, into one filtered quaternion.

4.1 Gyroscope Quaternion

The quaternion from the gyro, q^g, is calculated as a time integration of the quaternion derivative from (B.5)

$$q^g_{k+1} = q_k + t_s \dot{q}^g_k \tag{B.12}$$

where k denotes the discrete time step, q_k denotes the estimated quaternion from the complementary filter, t_s is the sample rate of the system and $\dot{q}^g_k$ is the time derivative of the quaternion calculated as in (B.5). The initial value of q^g_0 has to be set to a normalized quaternion, i.e. $[1\ 0\ 0\ 0]^T$, or ideally the quaternion from the accelerometer since this will result in a more correct initial value.

4.2 Accelerometer Quaternion

The accelerometer is attached to the quadcopter body frame and its values describes the quadcopter body orientation vector relative to the gravity in the navigation frame, given a steady state such as hovering. Aggressive maneuovers may affect the accelerometer with other forces like centripetal force and give a false reading of the gravity vector. For any given vectors v_1 and v_2, where $v_1 \neq -v_2$ and $||v_1|| = ||v_2|| \neq 0$, the quaternion for the shortest rotation arc between those vectors

can be described as [Roberts and Tayebi, 2011]:

$$q_0 = \frac{1}{||v_1||}\sqrt{\frac{||v_1||^2 + v_1^T v_2}{2}} \tag{B.13}$$

$$q_{1:3} = \frac{1}{||v_1||}\sqrt{\frac{1}{2(||v_1||^2 + v_1^T v_2)}} \cdot v_2 \times v_1 \tag{B.14}$$

By using the normalized accelerometer vector and the fixed gravity vector in the navigation frame $r_{nz} = [0\ 0\ 1]^T$, (B.13) and (B.14) can be used to calculate the quaternion of the quadcopter relative to the fixed navigation frame. The method which calculates the shortest rotation arc between the two vectors encounters a singularity for a 180 degrees rotation. At this situation there is no shortest rotation arc and the calculation results in dividing by zero. A workaround can be implemented using if-else statements.

Inserting the normalized accelerometer vector $\widetilde{a}_b$ and the gravity vector $r_{nz} = [0\,0\,1]^T$ in (B.13) and (B.14) yields

$$q_0^a = \sqrt{\frac{1 + \widetilde{a}_{bz}}{2}} \tag{B.15}$$

$$q_{1:3}^a = \frac{1}{\sqrt{2(1 + \widetilde{a}_{bz})}} \cdot \begin{bmatrix} \widetilde{a}_{by} \\ -\widetilde{a}_{bx} \\ 0 \end{bmatrix} \tag{B.16}$$

4.3 Filtered Quaternion

The gyroscope can provide a quaternion with fast and smooth angle updates. The quaternion from the gyroscope, q^g is calculated as the rotation from its initial position based on the angular velocity, and is not referred to a fixed reference frame. The gyroscope is also affected by a bias which means that the quaternion q^g will drift over time if not compensated for.

The accelerometer however will always have a fixed reference frame i.e. the navigation frame. But the accelerometer is sensitive to noise and airframe vibrations and will not be able to estimate the attitude smooth and fast enough in order to stabilize the quadcopter.

The proposed filter will combine q^g with q^a and compensate for both the drifting and the noise. Assume that the output quaternion is mainly based on q^g. As q^g starts to drift q^g will be significant different than q^a. q^g should then be rotated back towards q^a. Rotating q^g completely back to q^a in one operation makes q^g equal to q^a. Since q^a is has a lot of noise, q^g should only be rotated enough to compensate for the drifting. Quaternion rotations as described in (B.4) can be time consuming if implemented on a microcontroller. A much simpler way of doing it is to componentwise compare the two quaternions q^g and q^a and calculate the difference q^e:

$$q^e = q^a - q^g \tag{B.17}$$

$$q^e = \begin{bmatrix} q_0^a - q_0^g \\ q_1^a - q_1^g \\ q_2^a - q_2^g \\ q_3^a - q_3^g \end{bmatrix} \tag{B.18}$$

q_e is not a quaternion saying how much to rotate q^g in order to get to q^a. It is only the componentwise difference and is the correction needed to be added to q^g in order to become q^a. By implementing a scaling factor P_{err}, the total difference to compensate for at each cycle is reduced. This reduces the noise from the accelerometer.
The estimated quaternion q of the filter becomes:

$$\begin{aligned} q &= q^g + P_{err} q^e \\ q &= q^g + P_{err}(q^a - q^g) \\ q &= q^g \cdot (1 - P_{err}) + q^a \cdot P_{err} \end{aligned} \tag{B.19}$$

where $P_{err} \in [0,1]$ has to be set as low as possible to reduce the noise from the accelerometer, but high enough to correct for the drifting of the gyroscope. Setting $P_{err} = 0$ eliminates the accelerometer and $P_{err} = 1$ eliminates the gyroscope. The needed value of P_{err} will be dependent on factors such as gyroscope bias, cycle time of the system, accelerometer noise etc. An initial value of P_{err} can be found by monitoring the estimated quaternion with no movement. P_{err} should then be reduced until the estimated quaternion starts to drift and then increased back to no

drifting. A block diagram of the filter is presented in Fig. B.3.

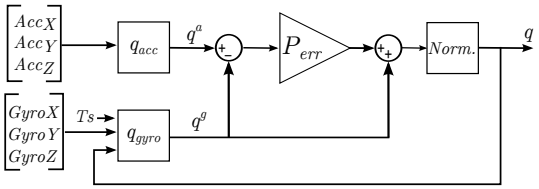


Figure B.3: *Attitude estimator*

where the normalization block is given by

$$q_{norm} = \frac{q}{||q||} \tag{B.20}$$

and the blocks q_{gyro} and q_{acc} are given by (B.12) and (B.15)-(B.16) respectively. The normalization is very important as numerical errors will make the length of the quaternion different from one over time.

5 Control Architecture

For a XYZ rotation matrix the quaternion can be written as:

$$\mathbf{q} = \begin{bmatrix} \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\cos(\frac{\psi}{2}) - \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) + \sin(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) \\ \cos(\frac{\phi}{2})\cos(\frac{\theta}{2})\sin(\frac{\psi}{2}) - \sin(\frac{\phi}{2})\sin(\frac{\theta}{2})\cos(\frac{\psi}{2}) \end{bmatrix} \tag{B.21}$$

Equation (B.21) clearly shows that each element in the quaternion q is dependent of a rotation around every of the three axes. By assuming small angles the sine- and cosine function can be linearized by the use of these formulas:

$$\cos(\sigma) = 1 \tag{B.22}$$

$$\sin(\sigma) = \sigma \tag{B.23}$$

$$\sin(\sigma)\sin(\sigma) = 0 \tag{B.24}$$

$$\cos(\sigma)\cos(\sigma) = 1 \tag{B.25}$$

Applying the rules of (B.22)-(B.25) in (B.21) yields

$$q_{1:3} = \begin{bmatrix} \frac{\phi}{2} \\ \frac{\theta}{2} \\ \frac{\psi}{2} \end{bmatrix} \tag{B.26}$$

Small angles are normally in the range of ± 15 degrees. In this case the sine and cosine is for half the angle, hence this approximation is valid for angles up to ± 30 degrees. By use of this linearization $q_{1:3}$ can be used as a control input directly. Since the attitude controller is not estimating the yaw, ψ will always be set to zero. The linearization also shows that it is possible to compare desired angle setpoints directly with the quaternion vector $q_{1:3}$ without generating a new quaternion from the angle setpoints.

The attitude of the quadcopter is controlled by angle setpoints to the controller in terms of a desired quaternion q_d. The controller is a proportional cascade controller with angle and angular velocity feedback as illustrated in Fig. B.4. The controller is often used for quadcopters, also for demanding situations [Mellinger et al., 2010].

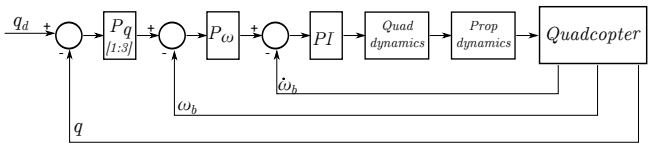


Figure B.4: *Attitude controller*

where q_d is the desired quaternion, q is the estimated attitude quaternion of the quadcopter, ω_b is the angular velocity of the quadcopter's body frame. The block $P_q[1:3]$ extracts the vector part of the quaternion and multiplies it with a gain P_q. The output of the block is the setpoint for the angular velocity. P_ω is the gain of the difference of the angular velocity setpoint and the measured value. "Quad Dynamics" converts the desired setpoint from the controller into forces for each

motor using a slightly modified version of (B.11):

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 0 & 0 & d_m & -d_m \\ -d_m & d_m & 0 & 0 \\ T_q & T_q & -T_q & -T_q \\ 1 & 1 & 1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} M_x \\ M_y \\ M_Z \\ T_h \end{bmatrix} \tag{B.27}$$

where T_h is the desired total propeller thrust. By adding T_h, the matrix in (B.11) becomes invertible and the propeller thrusts $F_{1:4}$ can be calculated as in (B.27).

6 Simulations

To verify the controller a quadcopter was modeled in Matlab/Simulink based on equation (B.7) - (B.10). The model is shown in Fig. B.5 The verification was con-

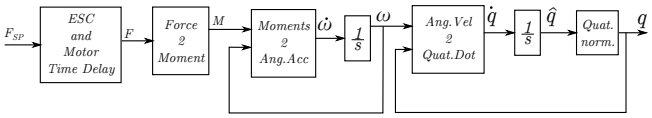


Figure B.5: *Quadcopter model in Matlab/Simulink*

ducted to see how the quadcopter would behave if the quaternion was used directly as an input, and compare the results with extracted Euler angles. The quadcopter was excited with a step input to the controller with a quaternion equal to $[20, 30, 0]^T$ degrees.

Fig. C.18 shows that there is a small deviation in the response of the two different input signals. As the angle increases the quaternion set point decreases, showed in (B.21).

The reduction of the angle is due to the linearization in (B.21)-(B.25), but is not critical for the operation of the quadcopter. Angles different from zero are normally only used in transient responses, while the steady-state situation is normally hovering where the angles are zero.

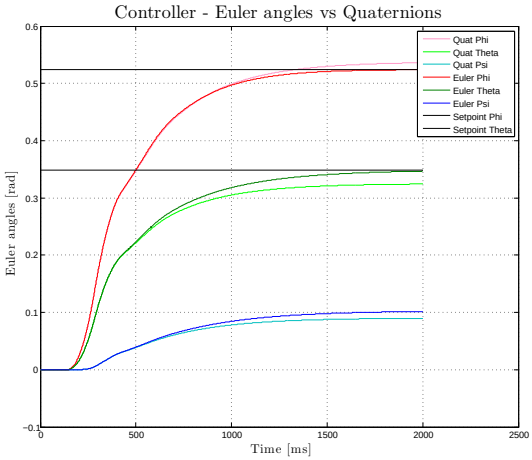


Figure B.6: *Comparison of a quaternion and Euler angle as setpoints to the controller*

7 Experiments

7.1 Stewart Platform

The attitude estimator was verified using a Bosch-Rexroth Stewart platform. The Stewart platform has previously been verified by a FARO laser tracker to have approximately 10μ meter in position error. It is therefore assumed that the angle error of the Stewart platform is also very small. Prior to the testing, the offset of the gyros were calculated and compensated for. With little gyroscope offset the value of P_q can be very small. The selected value was $P_q = 0.002$. The low value of P_q results in a smooth signal with low noise. The draw back is that the smaller P_q is, the longer it takes to correct for initial errors.

The quadcopter was placed on top of the Stewart platform, Fig. B.7 with aligned axes. The Stewart platform was set to run a sine with an amplitude of 0.17 radians and a frequency of 0.2 Hz. The frequency was set low due to restrictions of the Stewart platform.

The quadcopter attitude was initialized with aligned axes with the navigation frame, $q_0 = [1\ 0\ 0\ 0]^T$. The Stewart platform started its motions prior to starting the quadcopter to induce an offset error. The error was induced to verify that the filter could compensate for it. The estimated quaternion was converted into Euler angles for comparison using (B.6) and the result is shown in Fig. B.8.

Figure B.7: *Stewart platform used for testing*

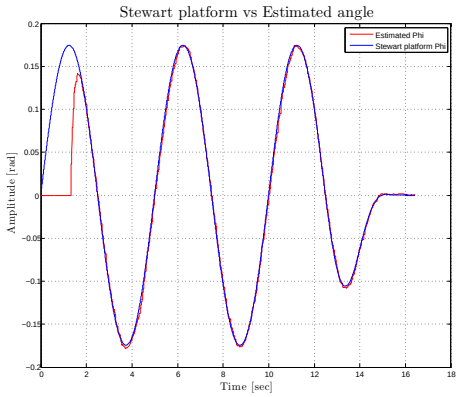


Figure B.8: *Dynamic response of the filter tested on a Stewart platform*

After approximately 500ms the estimated attitude of the quadcopter was equal to the attitude of the Stewart platform. During the whole test the estimator was able to track the attitude of the Stewart platform. It must be noted that this was a very ideal test with very small amplitudes of noise and other disturbances. The test did however verify that the filter works as intended, and gave good results.

7.2 Real flight comparison

The filter was designed to be good and simple to implement on a low cost controller board for a multicopter. Unlike the the Stewart Platform a multicopter introduces a lot of vibrations to the airframe, and hence the IMU. The IMU and especially the

accelerometers, are affected by those vibrations. The MPU6050 has an integrated filter which reduces the unwanted vibrations. A drawback of the MPU6050 is that it does not support individual filter settings for the accelerometers and gyroscopes. The bandwidth of the gyroscopes were set to 42Hz and with that setting the bandwidth of the accelerometers were automatically set to 44Hz. In general it is desired to filter the accelerometer even some more. Since this was not possible to do on the MPU6050 a second filter was implemented on the microcontroller in order to get decent values from the accelerometers. The filter is a first order low pass filter:

$$\hat{a}_{b(n)} = \hat{a}_{b(n-1)} \cdot G_a + a_{(n)} \cdot (1 - G_a) \tag{B.28}$$

where ˆ denotes the filtered accelerometer value, (n) and $(n-1)$ denotes the new and previous value and $G_a \in [0,1]$ denotes the filter gain. Increasing G_a increases the filter of the accelerometer. A more advanced filter is not desired, or needed, as this complicates the implementation on the microcontroller.

The attitude estimation filter and attitude controller algorithms were implemented on the MultiWii Mega. The efficient algorithms made the loop frequency as high as 400Hz, which includes sensor readings and scaling, attitude estimation, attitude controller and update of the motor speed.

In lack of a motion capture system, such as IR-cameras, the proposed filter was compared with one of the best high-end IMU's available, the 3DM-GX3-25 from MicroStrain. Both sensors were installed on the quadcopter and aligned as good as possible in order to get equal angle values. Aligning three axes with relative primitive equipment is almost impossible. With even a small rotation about one of any of the axes will result in different measurement. Also the two sensors were placed on foam dampers to reduce the mechanical vibrations. The dampers might also result in different movements. A log of the two sensors from a acrobatic real flight are shown in Fig. B.9.

Fig. 9 is a log from a real flight from take-off to landing. It shows that the proposed algorithm calculates the attitude close to the the 3DM-GX3-25 sensor. Even flying the quadcopter over a longer period of time shows no indication of drifting of the proposed filter.

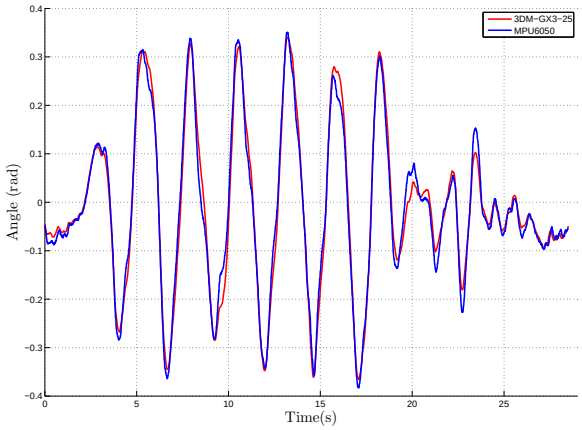


Figure B.9: *Pitch from a flight log with the proposed quaternion estimator implemented on a MultiWii Mega using MPU6050 (blue) with the high-end 3DM-GX3-25 IMU (red).*

8 Conclusions

In this paper a new method to estimate the attitude (roll and pitch) of a UAV was described. The attitude was quaternion based and a calculated quaternion from accelerometers was filtered together with the quaternion from the gyroscopes using a complementary filter. The attitude of the accelerometer was calculated using a formula for the shortest rotation arc from the gravity reference vector. The quaternion from the gyroscope was calculated using time integration of the derivative of the quaternion which is calculated by the angular velocities measured by the gyroscopes. The quaternion were completely estimated without the need of trigonometric functions. The estimated filter is able to dynamically correct errors, and was verified experimentally by a test on a Stewart platform and compared with a high-end IMU from MicroStrain. The gyroscope bias has to be calculated prior to the flight, as the attitude estimator does not estimate it. An enhancement of the proposed filter is to dynamically adapt the filter gain P_{err} for the two quaternions q^a and q^g in flight. One approach is to reduce P_{err} if the module of the sensed accelerometer vector is different from one.

The quaternion was also used for direct input to the control system. For small roll and pitch angles the quaternion vector is a representation of the Euler angles of the quadcopter, given a yaw angle equal to zero. The quadcopters are usually flying for

a position setpoint, hence this approximation should not affect the total system. If necessary the approximation can be compensated for by solving two equations from (B.21).

The proposed filter may not be the best filter when it comes to noise cancellation, but the required computational effort required to calculate the attitude quaternion is very low, and in all a good attitude estimator to stabilize a quadcopter.

REFERENCES

[de Marina et al., 2012] de Marina, H., Pereda, F., Giron-Sierra, J., and Espinosa, F. (2012). Uav attitude estimation using unscented kalman filter and triad. *Industrial Electronics, IEEE Transactions on*, 59(11):4465 –4474.

[Hall et al., 2008] Hall, J., Knoebel, N., and McLain, T. (2008). Quaternion attitude estimation for miniature air vehicles using a multiplicative extended kalman filter. In *Position, Location and Navigation Symposium, 2008 IEEE/ION*, pages 1230 –1237.

[Mahony et al., 2005] Mahony, R., Hamel, T., and Pflimlin, J.-M. (2005). Complementary filter design on the special orthogonal group so(3). In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 1477–1484.

[Mellinger et al., 2010] Mellinger, D., Michael, N., and Kumar, V. (2010). Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *Proceedings of the International Symposium on Experimental Robotics*.

[Roberts and Tayebi, 2011] Roberts, A. and Tayebi, A. (2011). Adaptive position tracking of vtol uavs. *Robotics, IEEE Transactions on*, 27(1):129 –142.

[Roskam and Lan, 1997] Roskam, J. and Lan, C. (1997). *Airplane Aerodynamics and Performance*. Airplane design and analysis. DARcorporation.

[Sidi, 2002] Sidi, M. (2002). *Spacecraft Dynamics and Control A practical engineering approach*. Cambridge Unversity Press.

[Stingu and Lewis, 2009] Stingu, E. and Lewis, F. (2009). Design and implementation of a structured flight controller for a 6dof quadrotor using quaternions. In *Control and Automation, 2009. MED '09. 17th Mediterranean Conference on*, pages 1233 –1238.

[Tayebi and McGilvray, 2004] Tayebi, A. and McGilvray, S. (2004). Attitude stabilization of a four-rotor aerial robot. In *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, volume 2, pages 1216 – 1221 Vol.2.

[Tayebi and McGilvray, 2006] Tayebi, A. and McGilvray, S. (2006). Attitude stabilization of a vtol quadrotor aircraft. *Control Systems Technology, IEEE Transactions on*, 14(3):562 – 571.

[Xiaoping et al., 2008] Xiaoping, Y., Bachmann, E., and McGhee, R. (2008). A simplified quaternion-based algorithm for orientation estimation from earth gravity and magnetic field measurements. *Instrumentation and Measurement, IEEE Transactions on*, 57(3):638 –650.

[Yun and Bachmann, 2006] Yun, X. and Bachmann, E. (2006). Design, implementation, and experimental results of a quaternion-based kalman filter for human body motion tracking. *Robotics, IEEE Transactions on*, 22(6):1216 –1227.

Paper C

Experimental Study on the Influence of Controller Firmware on Multirotor Actuator Dynamics

Øyvind Magnussen*, Simon Kirby**, Morten Ottestad* and Geir Hovland*.

This paper has been published as:

Øyvind Magnussen*, Simon Kirby**, Morten Ottestad* and Geir Hovland*.*IEEE International Symposium on Robotic and Sensors Environments (ROSE).* Timisoara, Romania, October 16-18, 2014.

Experimental Study on the Influence of Controller Firmware on Multirotor Actuator Dynamics

Øyvind Magnussen*, Simon Kirby**, Morten Ottestad* and Geir Hovland*.

*Department of Engineering

Faculty of Engineering and Science, University of Agder

Jon Lilletunsvei 9, 4879 Grimstad, Norway.

**Hostway Canada, Inc.

***Abstract* —**

In this paper the dynamic response of a propeller actuator commonly used in hobby unmanned aerial vehicles is studied experimentally. It is shown that the choice of electronic speed controller firmware has a significant effect on the overall actuator dynamics. Six different scenarios are tested: 1+2) Rising/falling step response with the standard firmware of the Hobbyking F30a, 3+4) Rising/falling step response with firmware from Simon Kirby/GitHub and 5+6) Rising/falling step response with firmware from Simon Kirby/GitHub including complementary Pulse-Width-Modulation (PWM) switching. Experimental results show a significant difference in actuator dynamics depending on the chosen firmware. By using firmware with complementary PWM switching, the rising and falling step responses are very similar. Such model symmetry is an advantage for control systems development, both in terms of robustness and performance.

1 Introduction

Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicles (UAVs) such as multi-rotor aircrafts adjust the actuators' thrusts for motion control. Most UAVs use fixed pitch propellers with varying propeller velocity as they provide a low-cost, simple and robust design compared to swash-plate mechanisms and pitch control. Because of these benefits modeling and control of UAVs have received significant attention at many research labs during the last decade. In order to achieve the best

possible control performance of a UAV, an accurate dynamic model which describes the dynamics of the actuating system is essential. In this paper, the actuating system is composed of a brushless (electronic speed) controller, a brushless motor and a propeller.

An accurate simulation model of the propeller actuation system is also of significant benefit when evaluating new UAV designs and testing new advanced control algorithms. In this paper a physical test-bench including a force sensor is used to measure thrust, current, voltage and velocity from a propeller actuator commonly used in hobby UAVs, the Hobbyking F30a.

One common type of UAV which requires accurate dynamic models for control is the quadcopter. A quadcopter is typically controlled by manipulating the speed of its four actuators. When the electronic speed controller (ESC) increases the current in the windings the motor torque will increase and the motor accelerates. The dynamic behaviour when decelarating is different from acceleration, where regenerated current and air-resistance cause the motor to slow down. With low speed both the regenerated current and the air-resistance are low, hence the deceleration will be highly speed-dependent. In other words, the dynamics of the actuator is different when accelerating and decelerating and there is a need for a dynamic model which captures this behaviour.

In [Cheron et al., 2010] a similar study to this paper was performed. An experimental testbed for a multirotor actuator was used to identify model parameters. A transfer function with a time-delay was estimated based on both positive and negative step responses from 0-1500 RPM. Applying the same transfer function on reduction of the actuator thrust yields a difference in the two systems. The authors claim that this difference was due to the aerodynamical properties and the control algorithm in the ESC that does not brake the motor.

In [Pounds et al., 2007] a transfer function model of a hobby actuator model consisting of rotor, motor, controller board and battery was estimated. An additional pole and zero for the battery model were included, while the time-delay was omitted. It was stated that *the rate of the battery voltage change is slow and we do not believe it will substantially affect the dynamics of the system*. In total, the cascaded transfer function for the actuator took the form of a gain, one zero and two poles.

Custom control boards were used because it was found that *RC hobby controllers are unsuitable due to non-linearities and bandwidth limitations*.

[Awan et al., 2011] presented an adaptive Luenberger observer for a quadcopter model. A second order model with two poles and no zeroes was assumed for the individual actuator dynamics. Experimental results were presented, but no distinction between rising and falling step responses were considered. Instead, the authors write: *In future work, we plan to extend this work to a nonlinear model, which can cover greater flight envelopes*.

In [Adigbli, 2007] different attitude and position control methods of a quadcopter model were presented. The nonlinear actuator model was linearized around an operating point which resulted in an actuator dynamics model represented as a first-order transfer function with a single pole. Time-delay and different dynamics for falling and rising step responses were not considered.

In this paper experimental validation of a motor-propeller model is established by use of Matlab and LabVIEW. The model is established from step responses for different ESC firmware, and verified by different input signals.

2 System Description

When increasing the RPM of the brushless DC motor (BLDC) the ESC increases the motor voltage by increasing the duty cycle of the pulse-width modulated (PWM) signal to the motor. This results in more current to the windings and the motor generates more torque which accelerates the motor. To reduce the RPM there are at least three different methods available. One is to let the motor coast and let the air resistance and the viscous damping brake the motor. The other option is to short-circuit the motor windings with the transistors and let the regenerated current brake the motor in addition to the air resistance. The third option is to actively brake the motor by applying a negative voltage compared to the regenerated voltage. This results in two different models, one where the motor is increasing its RPM, and one when reducing it.

For each quadcopter arm the motor system is as described in Fig. (C.1):

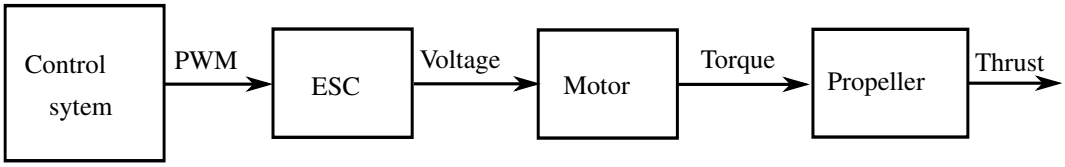


Figure C.1: *Quadcopter actuator system*

2.1 Actuator system

The actuator system used for testing is the motor, Fig. C.2, and propeller of a DJI F450 quadcopter. The quadcopter has a total weight of approximately 1.0 kg. With 4 propellers, each has to lift 250 g in hover. The quadcopter comes with two different propellers 8”x4.5” and 10”x4.5” where the first number denotes the propeller diameter, and the second number denotes the propeller pitch. Testing showed that the smaller propeller is able to change its RPM twice as fast as the larger one and with a total lift capacity of approximately 0.7kg, hence it was used for further testing. The motor K_V is 920 RPM/volt, and its maximum current is 30A. The ESC used in the experiments is Hobbyking’s F30a with an input PWM frequency of $450Hz$. The ESC is tested with standard firmware, and a firmware provided by Simon Kirby

[Kirby, 2013]. The new firmware converts the input signal directly to motor PWM, which is run at 18kHz. It is optimized for speed and every safety feature is removed in order to decrease the response time of the system.

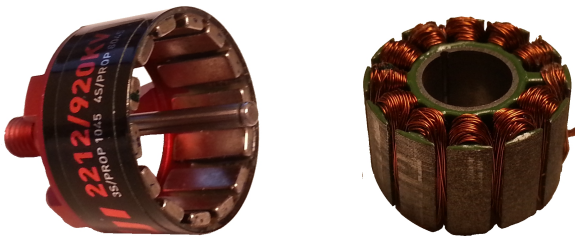


Figure C.2: *DJI* $2212/920K_V$ *motor used in testing*

2.2 Process Controller and sensors

The process controller is a cRIO-9022 from National Instruments. This unit generates the PWM control signal for the ESC using a NI-9401 high speed digital I/O. The NI-9401 also measures the RPM by sampling the output of a hall effect sensor which detects the 14 permanent magnets on the motor.
The cRIO-9022 also reads the load from a Phidget Micro Load Cell $(0-5kg)$, sampled with 25kHz. The ESC is updated with new values at $400Hz$, which gives $2.5ms$ delay between the updates. The logged thrust value is the mean value of the sampled thrust from the load cell during this time period. In addition to the load cell the RPM is also converted to thrust, thus two complementary sensors are used to ensure correct force measurement in dynamic conditions. The current is measured with an AttoPilot 90A current sensor. It is sampled the same way as the load cell.

3 Modeling

The BLDC is similar to a brushed DC motor and its block diagram can be represented as shown in Fig. C.4.
where V is the applied voltage, L_a is the motor inductance and R_a is the copper resistance. K_T is the torque constant, J is the total inertia of the motor and propeller,

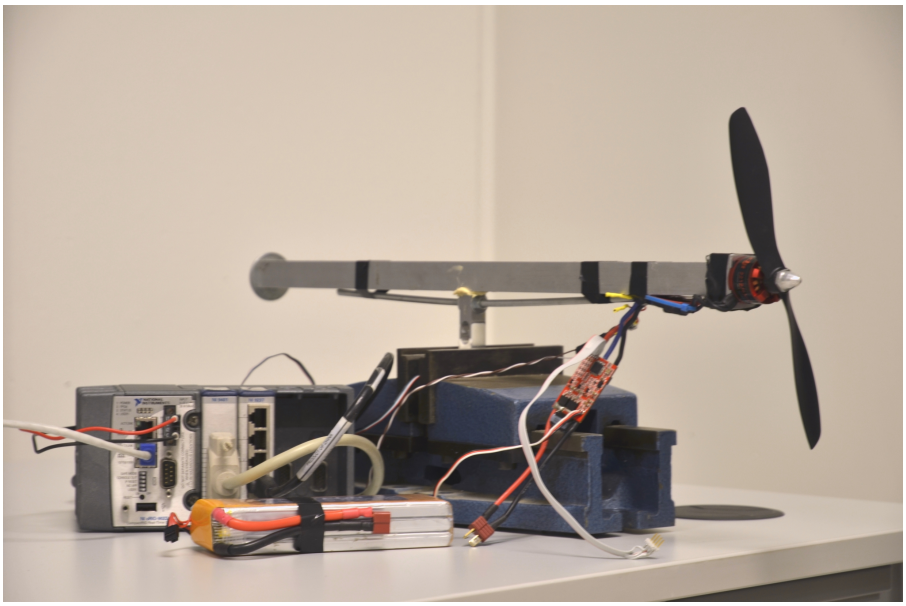


Figure C.3: *Test stand with cRIO, load cell, ESC and motor with propeller. The propeller is reversed and blowing away from the stand to reduce the turbulence.*

ω is the angular velocity and B is the viscous damping, K_E is the voltage constant, and T_L is the external load, which for this system is the propeller torque. The torque from the propeller is proportional to the generated thrust, F_{th}, which is a function of the angular velocity [Kermode and Philpott, 2012]. Due to the relationship $T = K_T I$, the external load T_L, combined with the viscous damping B, can be replaced by a function $f(w,I)$ which calculates the current for a given angular velocity. The current multiplied with K_T yields the total external torque.

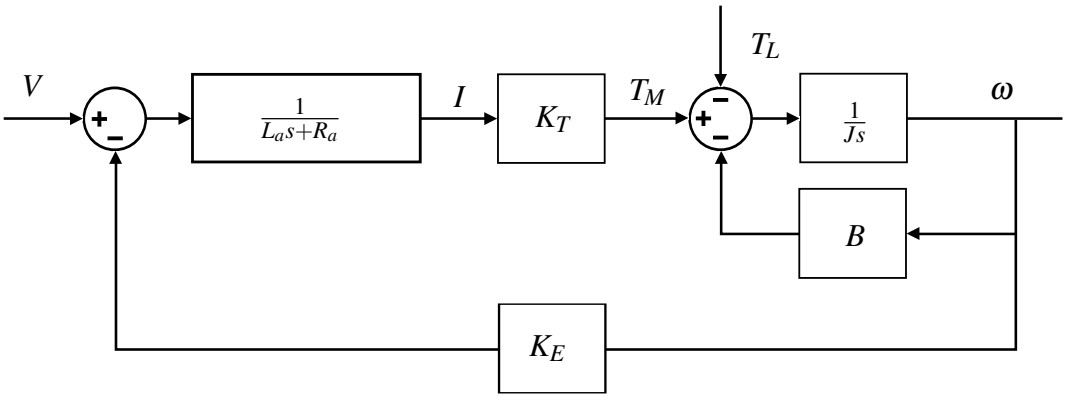


Figure C.4: *Linearized motor block diagram*

3.1 Delay and Second Order Model

Simulations show that during hover the value of $f(\omega, I)$ is approximately a constant of $0.004A/(rad/sec)$. Fig. C.5 is a linearized block diagram of Fig .C.4, where the function $f(\omega, I)$ is replaced with a constant gain $D = 0.004$.

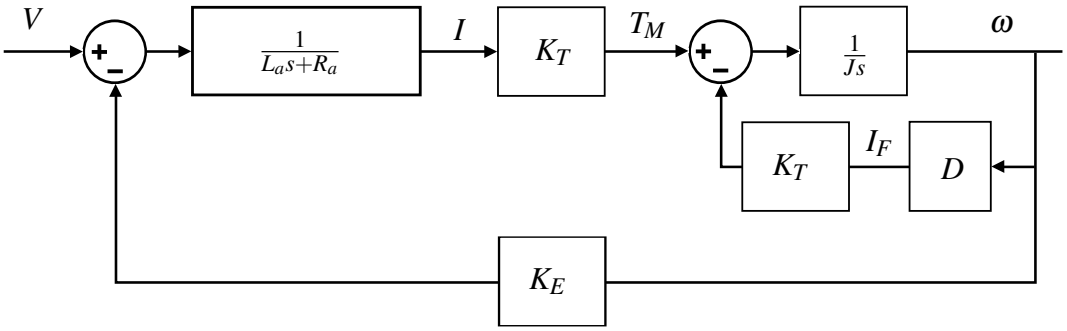


Figure C.5: *Motor block diagram*

Reducing the block diagram the transfer function $H(s)$ which converts voltage to angular velocity for the system is obtained

$$H(s) = \frac{\omega(s)}{V(s)} = \frac{\frac{K_E}{R_a D + K_E}}{\frac{L_a J}{R_a K_T D + K_T K_E} s^2 + \frac{L_a K_T D + J R_a}{R_a K_T D + K_T K_E} s + 1} \tag{C.1}$$

The transfer function for the thrust of the motor is the thrust output, F_{Th}, relative to the thrust setpoint, $\tilde{F}_{Th}$, $H_{Th}(s) = \frac{F_{Th}(s)}{\tilde{F}_{Th}(s)}$. According to the transfer function $H(s)$ the angular velocity is a function of the applied voltage. Measuring the continuously switching motor voltage V of the BLDC can be complicated, but also not needed. The total gain from thrust setpoint to actual thrust equals one, see Fig. C.6. Hence the inverse gain of the transfer function $H(s)$ multiplied with the angular velocity set point which is calculated from the thrust setpoint, is equal to the motor voltage.

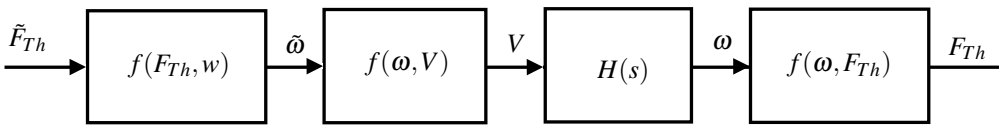


Figure C.6: *Block diagram from thrust set point to propeller thrust*

$f(F_{Th}, \omega)$ is a non-linear function converting the thrust setpoint $\tilde{F}_{Th}$ to angular ve-

locity setpoint $\tilde{\omega}$. $f(\omega,V)$ is the inverse gain of the transfer function $H(s)$ and $f(\omega,F_{Th})$ is the non-linear function converting angular velocity ω to actual propeller thrust F_{Th}.

3.2 Delay and First Order Model

If the inductance of the motor is very low, its dynamics can be neglected from the system by setting $L_a = 0H$ and the result is a simplified first order transfer function $H_f(s)$

$$H_f(s) = \frac{\frac{K_E}{R_aD+K_E}}{\frac{JR_a}{R_aK_TD+K_TK_E}s+1} \tag{C.2}$$

The time constant of the new system is the mechanical time constant τ_m with the same gain G as for the second order system $H(s)$.

$$\tau_m = \frac{JR_a}{R_aK_TD+K_TK_E} \tag{C.3}$$

$$G = \frac{K_E}{R_aD+K_E} \tag{C.4}$$

3.3 Complementary ESC mode

The driving method of a BLDC is based on a six step commutation cycle [Acarnley and Watson, 2006], [Vinatha et al., 2006], [Semiconductors, 2007], where the current is controlled with PWM of the high side field effect transistors (FETs). Updating the firmware of the ESC makes it possible to enable full complementary switching of the high and low side of the FETs. The switching occurs when the active phase is being pulse width modulated to control the current, Fig. C.7. The complementary scheme increases the current during PWM off state since the current can flow through the low side FET which has less voltage drop than the body diode. The increased current will also brake the motor more during deceleration, and in total give more responsive control of the RPM.
Complementary mode is also known as active free-wheeling or synchronous rectification. Fig. C.8.

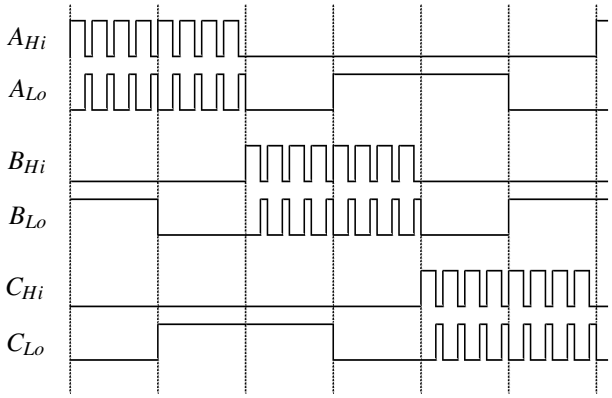


Figure C.7: *Complementary PWM scheme*

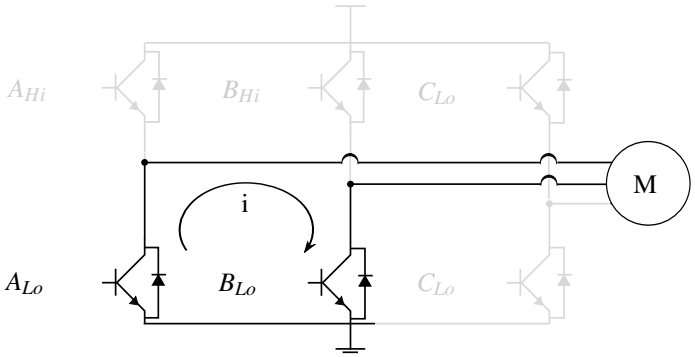


Figure C.8: *Complementary PWM scheme, where the low-side FET A_{Lo} is on when A_{Hi} is off. The current is then passing through the FET instead of the body diode, resulting in a braking effect.*

4 Experimental Results

The quadcopter weighs $1.0kg$ which for hover yields $250g$ of thrust per propeller. For the standard control margin of 30% the full thrust range of $\pm 30\%$ gives the range of $175g$-$325g$. The experiments consist of two different step inputs to the system, one for estimation and one for verification. The first input signal, $Step1$, is a standard step from $175g$ to $325g$. The second input signal $Step2$, is a signal with decreasing amplitude but with the same stationary values as $Step1$. The test results shown are the mean value of 20 unfiltered measurements of the angular velocity converted to the corresponding thrust, described in section 4.4. To ease the comparison of the increasing and decreasing thrust, the plotted values are the absolute values of the change of thrust marked with a 'Δ' on the graph. The decreasing

thrusts are marked with a dashed line.

4.1 Standard ESC vs. Updated Firmware

The standard ESC firmware is not optimized for speed and modifications. By implementing Simon Kirbys own firmware the ESC responds faster for both rising and falling step, Fig. C.9 and Fig. C.10. Note that the delay of the system is also decreased in addition to the rise time.

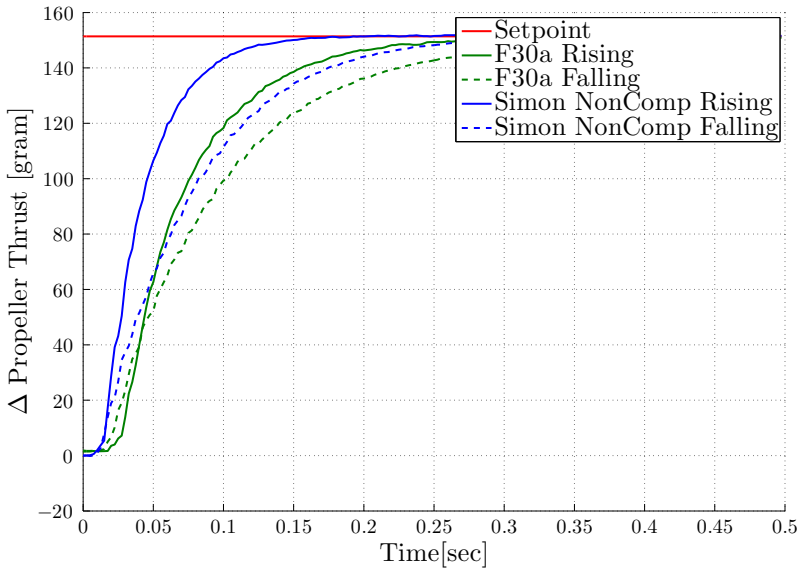


Figure C.9: *Step*1 *input to the standard firmware (green), and the one provided by Simon Kirby (blue) for both rising (solid) and falling step (dashed).*

4.2 Updated Firmware Mode Comparison

Turning on the complementary switching of the FETs provides even faster response, especially when decreasing the thrust, shown in Fig. C.11 and Fig. C.12.

4.3 Result Comparison

First order transfer functions of the logged data from *Step*1 were generated using System Identification Toolbox from LabVIEW and Matlab. Both of the programs

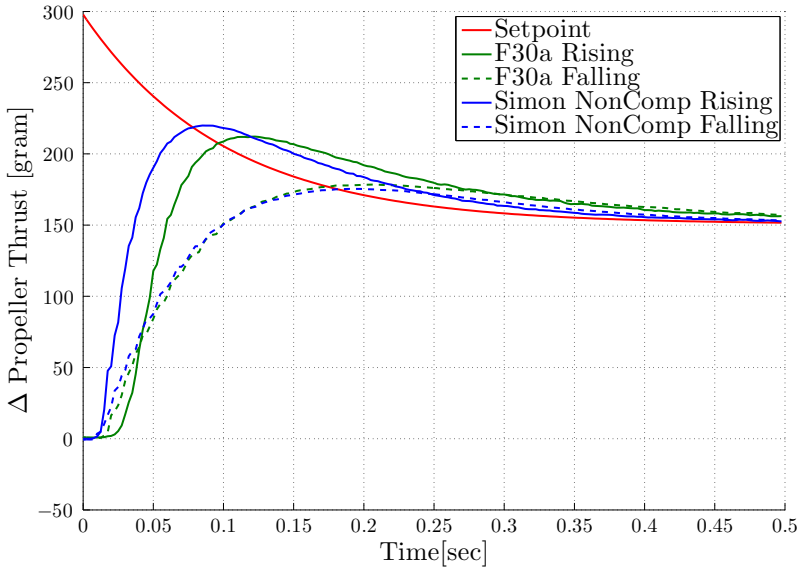


Figure C.10: *Step2 input to the standard firmware (green), and the one provided by Simon Kirby (blue) for both rising (solid) and falling step (dashed).*

gave the same results. The results and the mechanical time constant from (C.3) with data from table C.2 are presented in table C.1.

Table C.1: *Experimental results as first order system with delay*

ESC:	delay [ms]	τ_{up}[ms]	τ_{dn}[ms]	RMS
Standard	20	50	78	96
Simon	15	29	63	279
SimonCmp	15	30	35	11
From Parameters	N/A	32	32	0

where RMS is calculated as the difference of the rising, R, and falling, F, thrust for a $500ms$ time range:

$$RMS = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(R_i - F_i)^2}$$

where N is the number of samples.

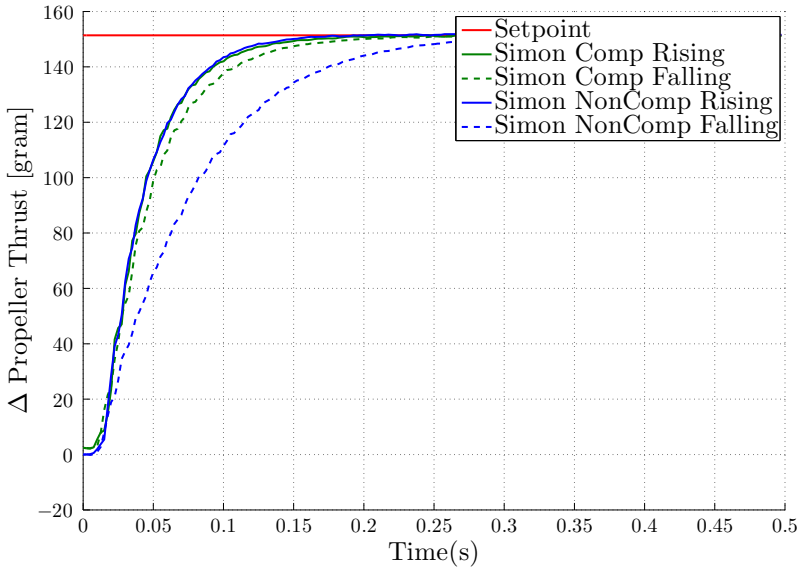


Figure C.11: *Step*1 *input to the firmware provided by Simon Kirby with complementary driven PWM enabled (green) and disabled (blue) for both rising (solid) and falling step (dashed).*

4.4 Parameters

The model described in section 4.1 requires the function generating desired thrust to angular velocity and vice versa, $f(F_{Th}, \omega)$ and $f(\omega, F_{Th})$ respectively. The function was found by increasing the RPM slowly to reduce the significance of the system dynamics, while logging the thrust and angular velocity. From the sampled data a set of polynomials was generated by curve fitting algorithms. The polynomials were also tested experimentally by setting different thrust set points and measuring the output. A log of the angular velocity and the thrust is shown in Fig. C.13.

The function converting angular velocity to current feedback, $f(\omega, I)$, was generated in the same way as for the thrust and angular velocity. Results are shown in Fig. C.14

The motor inductance was measured to $80\mu H$ with an Amprobe 37XR-A. The resistance was measured using a TTi EX354RD dual power supply. The current was set to $1.000A$ and the voltage was read to 0.220 volt which gives a resistance of 0.220Ω. The inertia of the motor and propeller were calculated using standard equations, and the results were compared with the results from a CAD model. According to

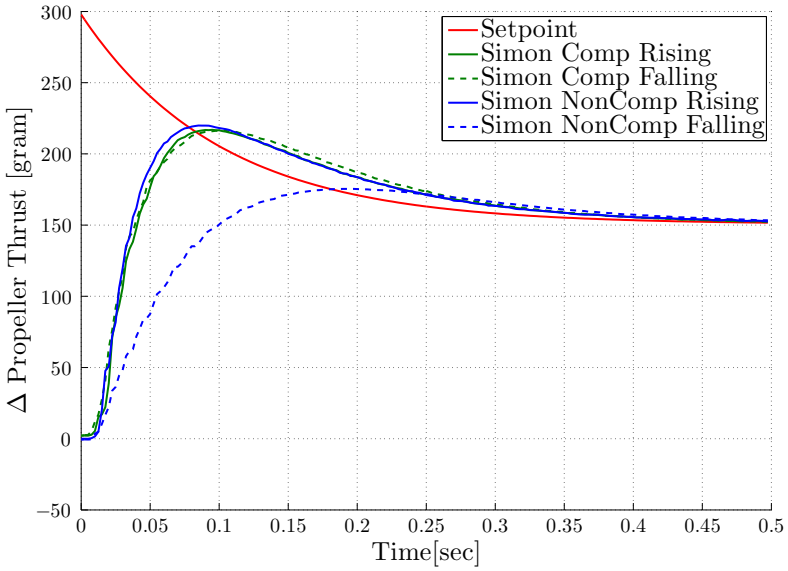


Figure C.12: *Step2 input to the firmware provided by Simon Kirby with complementary driven PWM enabled (green) and disabled (blue) for both rising (solid) and falling step (dashed).*

Table C.2: *Motor data from manufacturer and calculated*

Inductance	L_a	$= 80\mu H$
Resistance	R_a	$= 0.22\Omega$
RPM constant	K_V	$= 920 RPM/Volt$
Torque constant	K_T	$= 0.0104\frac{Nm}{A}$
Volt constant	K_E	$= 0.0104\frac{V}{rad/sec}$
Inertia, motor + propeller	J	$= 17.2 \cdot 10^{-6} kgm^2$
Damping factor	D	$\approx 0.004\frac{A}{rad/sec}$

the manufacturer the motor K_V is $920 RPM/V$. This results in $K_E = 0.0104\frac{V}{rad/sec}$. According to literature [Hughes and Drury, 2013] K_T has approximately the same value as K_E which gives $K_T = 0.0104\frac{Nm}{A}$. The damping factor D is not used in simulations, it is presented to show its approximate value. Even though the damping factor D varies with the RPM, simulations show that for this setup it is close to $0.004\frac{A}{rad/sec}$. The system parameters are presented in Table C.2.

The electrical time constant of the battery was tested to verify its significance. The

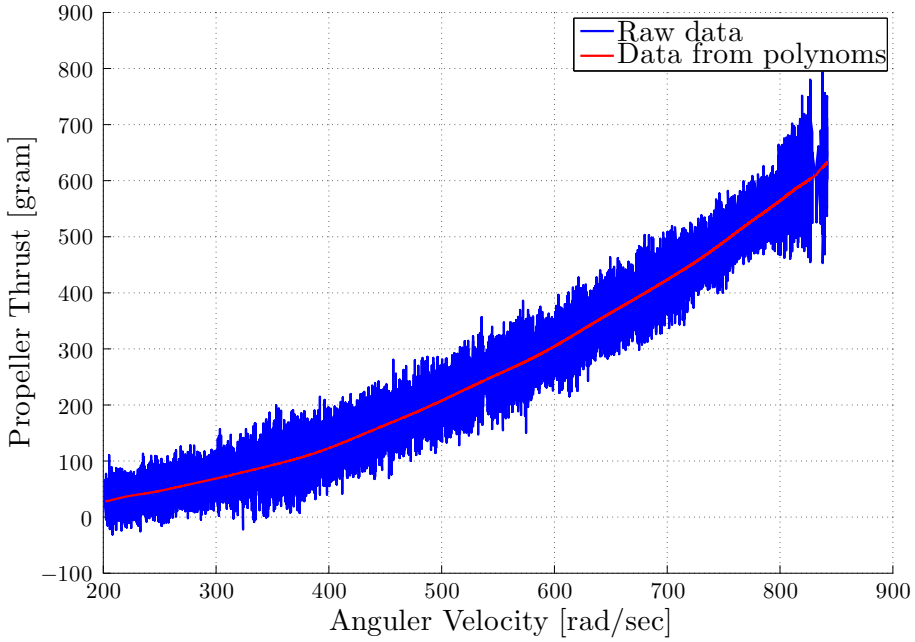


Figure C.13: *Plot of propeller thrust with respect to rad/sec for both the measurement and the estimated function*

response shown in Fig. C.15 is the mean value of 50 step inputs with an amplitude of 30A. LabVIEW generated an estimated first order transfer function of the response. The estimated transfer function is not a good match of the system, but the battery bandwidth is many times faster than the mechanical system and can be neglected.

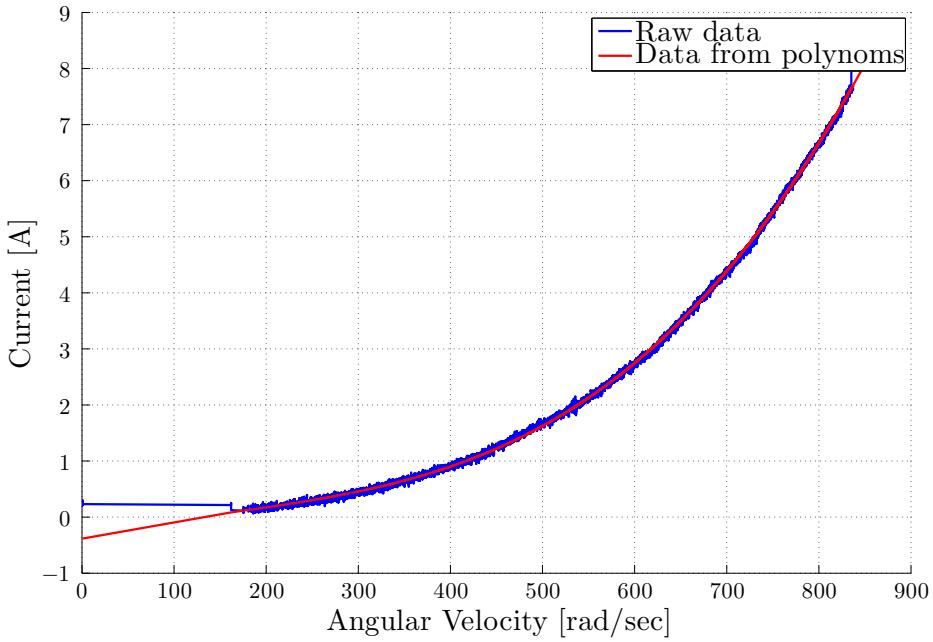


Figure C.14: *Comparison of the angular velocity with respect to current for measured data and estimated function.*

5 Model Comparison

5.1 Delay and First Order Model

In the experimental section, six transfer functions of the thrust were generated from the input signal $Step1$. That is one set (increase and decrease) for each of the three different ESC settings. The difference of the measured and simulated system response with $Step1$ as input will be small since the first order model is based on the same input signal which generated the transfer function. If the actuator system is a close match to a first order system the response of the measured and simulated thrust response should also be a close match for the second input signal $Step2$. A first order transfer function was used to simulate the two input signals for both increase and decrease of thrust, with time constant shown in Table C.1. When the two time constants for increase and decrease of thrust differs significantly a switching mechanism needs to be implemented in order to switch between the two time constants respectively, this mechanism is not described in this paper. For this to work

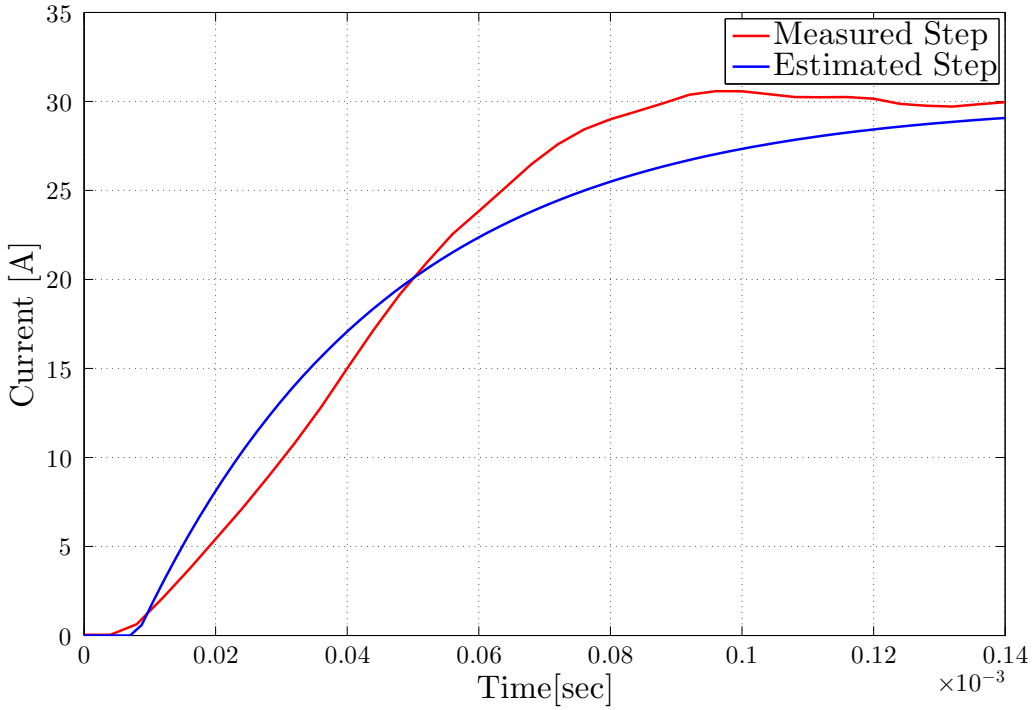


Figure C.15: *Step response of the battery used in testing, with measured data and estimated first order function.*

the simulated and measured response must be close to each other.

In Fig. C.16 the measured and simulated response of the transfer function for the standard ESC setting is shown.

For the step input the simulated increase and decrease of the thrust response are a close match to the measured thrust of the system. This is also expected since the transfer function is based on this data set. Applying the second input signal to the transfer function the differences are more prominent. When increasing the thrust the measured and simulated curves have a different shape and amplitude signaling that the system is not of first order. Decreasing the thrust the curves are more close to each other, but with slightly less amplitude.

For the updated ESC with non complementary switching the results are a bit different. When increasing the thrust the estimated transfer function is a very good match for the two inputs. For the decrease in thrust the model is not a very good match for the second input, shown in Fig. C.17. The system cannot be described as a first

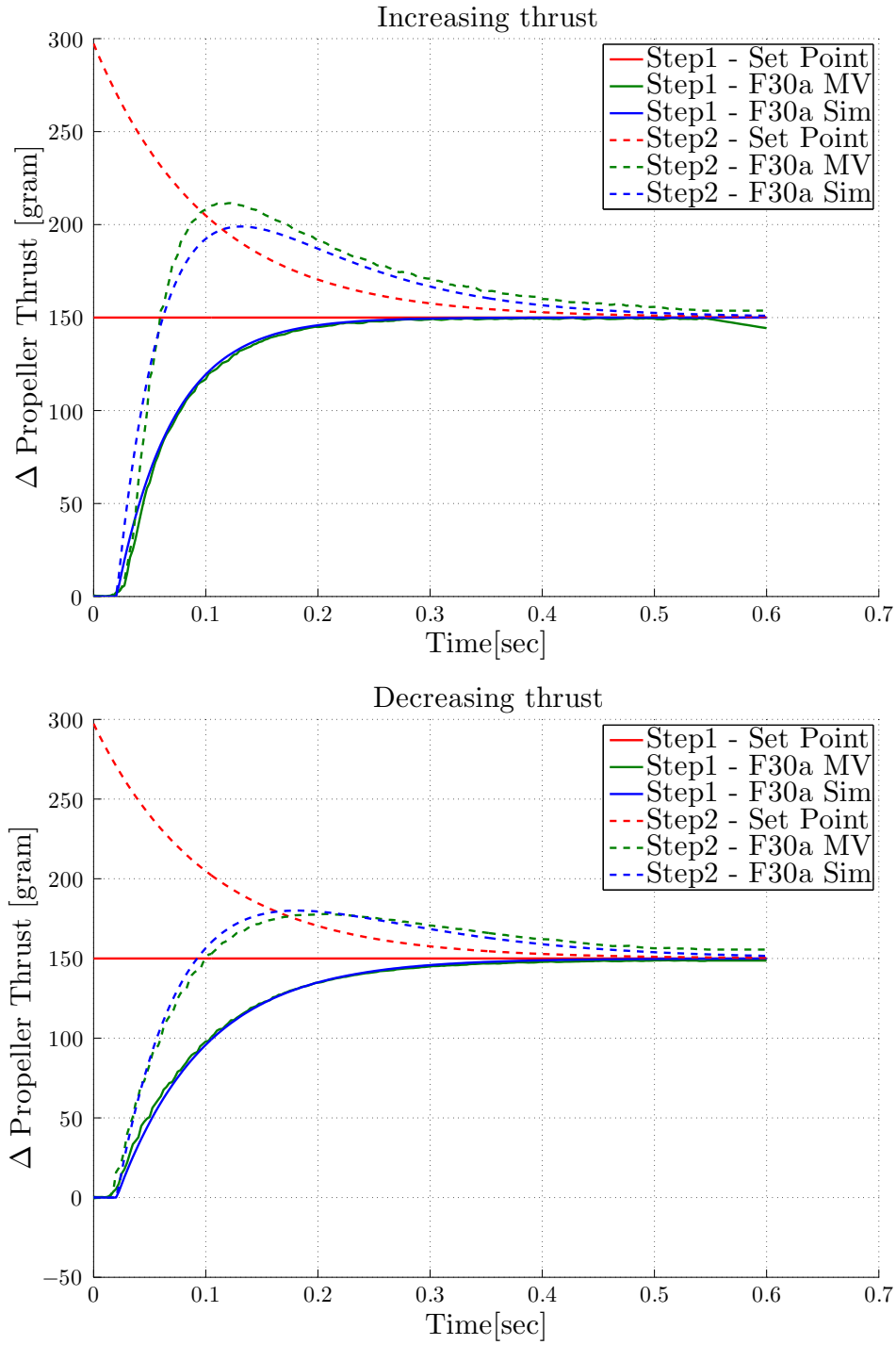


Figure C.16: *Simulation result for F30a for rising (top) and falling (bottom) step*

order model.

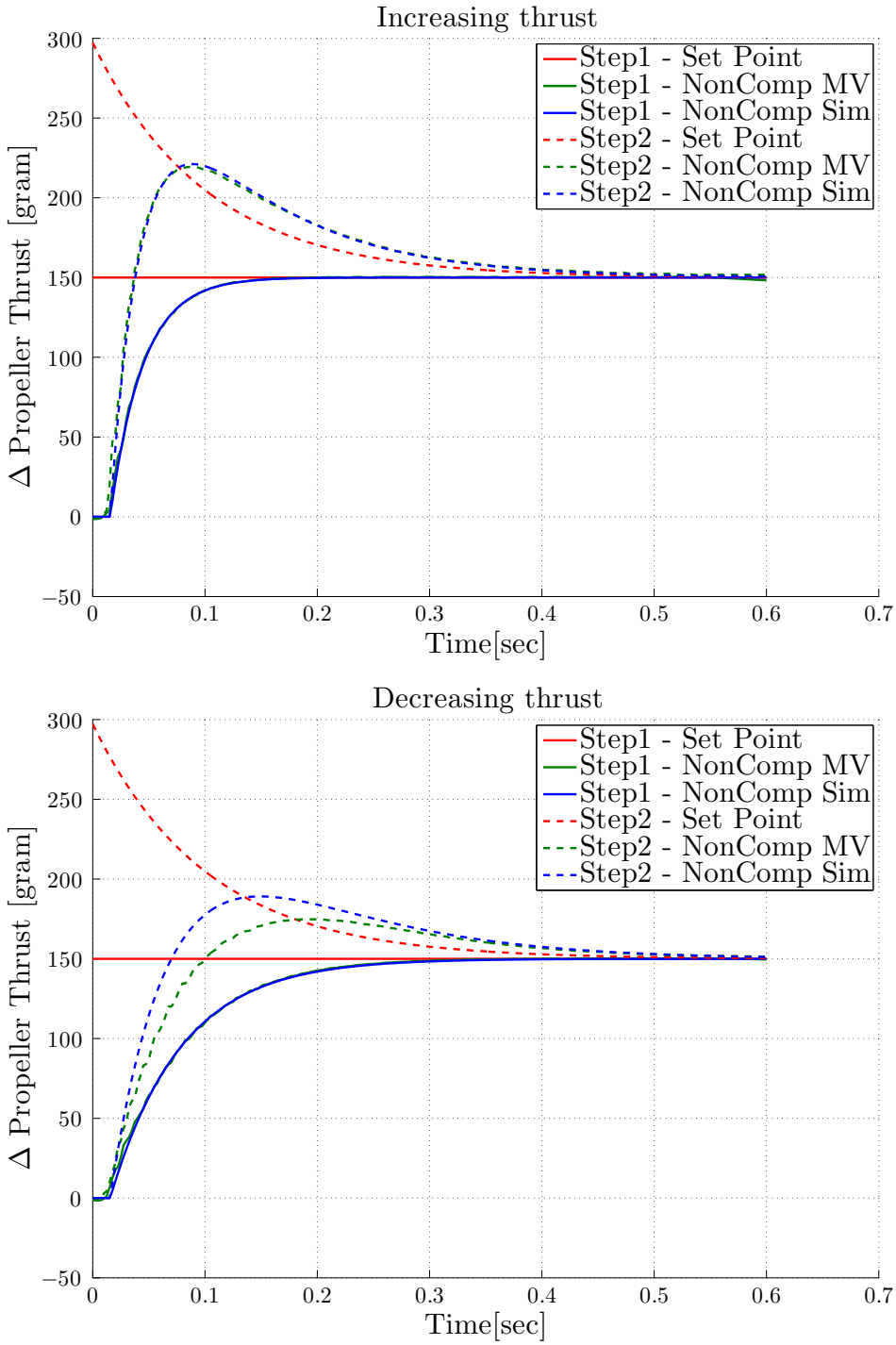


Figure C.17: *Simulation result for updated firmware with non-complementary PWM switching for rising (top) and falling (bottom) step.*

Running the ESC in complementary mode the simulation is almost an exact match of the experimental results, for both of the inputs for an increase and decrease of the thrust, Fig. C.18. In addition to being a good match the two estimated transfer functions are almost identical, and the need for a switching nonlinear model is not required in order to describe the system since it consists of a single time constant.

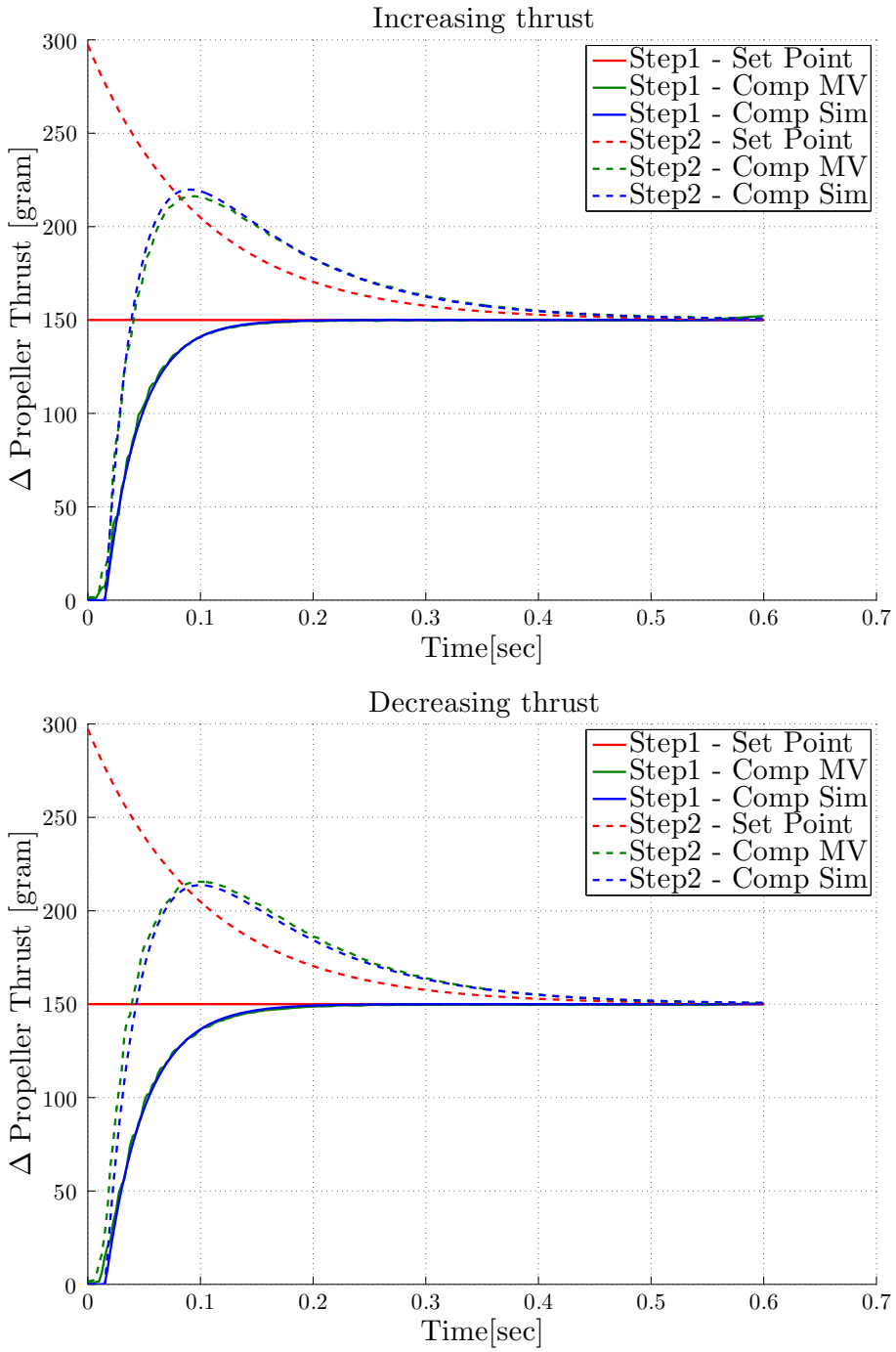


Figure C.18: *Simulation result for Simon Comp for rising (top) and falling (bottom) step*

5.2 Delay and First Order Model With Nonlinear Load

In the previous subsection the simulated system was a first order transfer function with linear load. The propeller torque is a nonlinear function of the angular velocity and thus induces nonlinearities to the system. The nonlinear system is realized by replacing the constant D from Fig. C.5 with the function $f(\omega, I)$ which graph is shown in Fig. C.14. Using the measured actuator data from Tab. C.2 and the experimental data shown in Fig. C.13 and Fig. C.14 the nonlinear motor model in Fig. C.6 is realized. This model includes the nonlinearities from the current feedback, which also includes viscous damping. It is based on measured and given component data, unlike the model in the previous subsection which was based on estimation of measured thrust data. The system model is only using one set of parameters which eliminates the need for a switching mechanism for an increase or decrease of thrust. Because of this it is only the response of the complementary ESC settings which is compared with the simulation results since this was the only ESC setting that gave almost identical results for both increasing and decreasing thrust.

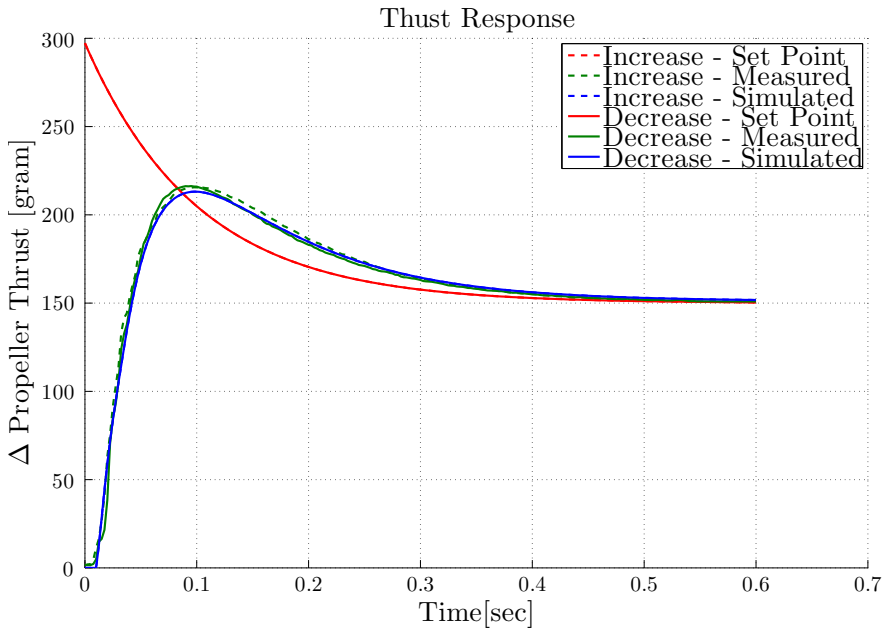


Figure C.19: *Thrust response of the simulated data and the complementary ESC settings with nonlinear load.*

The comparison of the simulated and measured response are shown in Fig. C.19. Running the ESC on complementary switching of the FET's gives nearly equal results for the measured and estimated system.

6 Conclusions

In this paper we have shown that a single transfer function run on a standard mode ESC does not describe the behavior of the thrust response of the propeller due to its significant differences when increasing and decreasing the thrust. Those differences are drastically reduced when the ESC is run in complementary PWM mode which also was the fastest mode tested. By using firmware with complementary PWM switching, the rising and falling step responses are very similar, hence the system can be described as a single system, either a linear transfer function or a first order system with non linear load. Such model symmetry is an advantage for control systems development, both in terms of robustness and performance. The electrical time constant is also shown to be neglectable since the mechanical time constant is much higher.

REFERENCES

[Acarnley and Watson, 2006] Acarnley, P. and Watson, J. (2006). Review of position-sensorless operation of brushless permanent-magnet machines. *Industrial Electronics, IEEE Transactions on*, 53(2):352–362.

[Adigbli, 2007] Adigbli, P. (2007). Nonlinear attitude and position control of a micro quadrotor using sliding mode and backstepping techniques. In *Proc. 3rd US-European Competition and Workshop on Micro Air Vehicle Systems (MAV07) & European Micro Air Vehicle Conf. and Flight Competition (EMAV2007)*, Toulouse, France.

[Awan et al., 2011] Awan, A., Park, J., and Kim, H. (2011). Thrust estimation of quadrotor uav using adaptive observer. In *11th Intl. Conf. Control, Automation and Systems*, KINTEX, Gyeonggi-do, Korea.

[Cheron et al., 2010] Cheron, C., Dennis, A., Semerjyan, V., and Chen, Y.-Q. (2010). A multifunctional HIL testbed for multirotor VTOL UAV actuator. In *Mechatronics and Embedded Systems and Applications (MESA), 2010 IEEE/ASME International Conference on*, pages 44–48.

[Hughes and Drury, 2013] Hughes, A. and Drury, B. (2013). *Electric Motors and Drives: Fundamentals, Types and Applications, 4th Edition*. Newnes.

[Kermode and Philpott, 2012] Kermode, A. C. and Philpott, D. (2012). *Mechanics of Flight*. Pearson Education Limited.

[Kirby, 2013] Kirby, S. (2013). Github repository. https://github.com/sim-.

[Pounds et al., 2007] Pounds, P., Mahony, R., and Corke, P. (2007). System Identification and Control of an Aerobot Drive System. In *Proc. Information, Decision and Control*, Adelaide, Australia.

[Semiconductors, 2007] Semiconductors, F. (2007). Three-Phase BLDC Motor Sensorless Control Using MC56F8013/23. Design Reference Manual DRM070, Rev 02.

[Vinatha et al., 2006] Vinatha, U., Pola, S., and Vittal, K. P. (2006). Recent developments in control schemes of bldc motors. In *Industrial Technology, 2006. ICIT 2006. IEEE International Conference on*, pages 477–482.

Paper D

Multicopter UAV Design Optimization

Øyvind Magnussen, Geir Hovland and Morten Ottestad.

This paper has been published as:

Øyvind Magnussen, Geir Hovland and Morten Ottestad.*IEEE/ASME 10th International Conference on Mechatronic and Embedded Systems and Applications (MESA).* Senigallia, Italy, September 10-12, 2014.

Multicopter UAV Design Optimization

Øyvind Magnussen, Geir Hovland and Morten Ottestad.
*Department of Engineering
Faculty of Engineering and Science, University of Agder
Jon Lilletunsvei 9, 4879 Grimstad, Norway.

***Abstract* —**
Designing and selecting hardware for a multirotor can be challenging in order to get the best flight performance out of the system. In addition to selecting the hardware, the number of actuators can also be altered. For a 4 actuator (quadrotor) setup, one set of hardware can give the optimal design, while for a 6 actuator setup (hexarotor) the same hardware may not necessary give the same response. In this paper we present a design optimization process of a multirotor, where the hardware is selected from a set of low-cost off-the-shelf standard RC hobby parts. Constraining the problem to a given hardware ensures existence of the selected hardware, and the design can be implemented. Also the system equations will remain linear when selecting from a set of given data. Hence the problem is defined as a mixed integer linear program (MILP) and solved with the Cplex solver. The Cplex solver is fast and the solution is close to the optimal solution of the problem. In this paper the multirotor is designed for 4, 6 and 8 actuators, and the design with the lowest value of the objective function is the optimal design. As shown, optimizing the design for a given set of hardware and payload results in a 11 minutes longer time of flight, than just using the biggest propeller, motor and battery available.

1 Introduction

During the last decade many research papers have been published on the topic of control strategies and generation of trajectories [Mellinger et al., 2012], [Culligan et al., 2007] for multirotors, and especially quadrotors. The multirotor

Figure D.1: *Examples of two different multirotors used in the design optimization process, a quadrotor and a hexarotor.*

has a simple construction design and consists mainly of a frame with a given number of arms, motors, propellers and a battery in addition to the control system, see Fig. D.1. Different hardware components give different flight characteristics, and for any given task, there exists an optimal design. For instance increasing the diameter of the propeller gives more thrust from the propeller for a given RPM, but the increased inertia makes the system less responsive.

Selecting hardware from experience and making sure the selected hardware is powerful enough by validating the components from web calculators, is a common way of designing a multirotor. Designing a multirotor without a computer aided solver makes it difficult to optimize for more than just a few criteria and constraints as the problem expands very fast. Using MILP in the design process there are no limits of linear criteria to implement. As shown in this paper the optimization is able to increase the flight time, and to increase the flight dynamics.

The optimizer used in this paper is the IBM ILOG CPLEX, simply referred to as Cplex. Commercial solvers such as the Cplex outperforms open source solvers. Cplex is able to optimize large scale MILP problems and is known for its fast calculations [Meindl and Templ, 2012]. The Cplex optimizer uses the branch and cut algorithm to solve the problem [IBM, 2013]. The branch and cut method is the most promising technique available for proving optimality [Mitchell, 2009].

Design optimization has been used for full size aircraft for many years. In [Mukesh et al., 2012] the aerodynamics of an aircraft airfoil were optimized to in-

crease the coefficient of lift. Several optimization algorithms were tested in the search for the optimal design.

In [Lee et al., 2012] robust design method with statistical constraints coupled to multi-objective evolutionary algorithms has been demonstrated for a Unmanned Aerial System. Numerical results show that the solutions obtained by the robust design approaches with statistical constraints have improvement on both aerodynamic and structural design quality in terms of their reliable performance and its robustness with respect to uncertainty design parameters.

Even though the design optimization is quite common for the full scale aircrafts, to our knowledge there are only a few published papers available, on the topic of design optimization of multirotors. [Ng and Leng, 2007] optimizes the multirotor by using the genetic algorithm. The genetic algorithm may have the tendency to converge towards a local optima, rather than the global optimum of the problem [Melanie, 1999].

This paper begins with a description of parameters to be constrained or optimized, followed by a set of assumptions for the process. Then equations and parameters used in the calculations are described. Next the optimization problem formulation is presented. Included in this is the functions used in the optimization process and how non-linear problems are linearized. Two case studies are presented to demonstrate the effectiveness of the optimal design process.

2 System Parameters

The following items can either be optimized or set to constraints.

- Payload capacity
- Dynamic performance
- Flight time
- Cost/complexity
- Propeller RPM

The design optimization process is free to change the following variables:

- Number of actuators
- Propeller type
- Motor type
- Battery
- Propeller RPM

The propellers, motors and batteries are selected from datasheets. The optimization process is based on a set of assumptions:

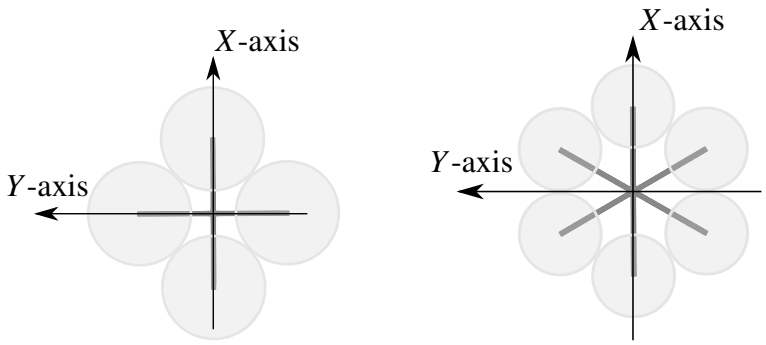
1. The multicopter inertia is increased with an increase of the frame size. Frame diameter is therefore kept as small as possible only to keep the propeller tip to tip distance equal to Δ, a small positive number.

2. The frame weight is proportional to the propeller diameter with a scalable weight pr length factor.

3. The calculated dynamic performance is a measurement of the rotational attitude performance of the multicopter, not the translational velocity or position. It is based on static propeller tests and system inertia, hence variable airflow through the propeller during flight is not taken into consideration. The payload and batteries will not affect the dynamic performance since they are modeled as a point mass in the center of the rotation.

4. It is assumed an even number of motors, where the minimum number is 4 and maximum number is 8, examples shown in Fig. D.2.

5. Each motor and propeller is equal.



a. Quadcopter example b. Hexacopter example

Figure D.2: Examples of different solutions, the first is a quadcopter with larger propeller, second is a hexacopter with smaller propellers

2.1 Battery Capacity

The batteries are selected from a set of datasheets, with weight, price, mAh, number of cells and discharge rate available. Fully charged the total battery voltage is 4.2V times the number of cells, and at critical voltage 3.2V times the number of cells. During this design optimization the battery voltage is kept constant at 4.0V pr cell.

2.2 Propeller Performance

Low pitch propellers yield less speed, but more torque. Such propellers are desirable in an optimization problem. Typical sizes vary from 8" to 14", with a pitch of 3.8" to 6". The propeller thrust equation used is the equation given by the propeller manufacturer APC [Propellers, 2013].

$$F_{Th} = \rho C_t n^2 D^4 \tag{D.1}$$

where ρ $[kg/m^3]$ is the density of the air, n $[rev/s]$ is the propeller angular velocity, C_t is the propeller thrust coefficient given by the datasheet and D $[m]$ is the diameter

of the selected propeller. C_t is not a constant for any RPM or velocity, but the variations are so small they are neglected. The selected C_t is for zero velocity and at 1000 RPM.
The power P_p $[W]$ drawn from the propeller is calculated as

$$P_p = \rho C_p n^3 D^5 \tag{D.2}$$

where C_p is the power coefficient for the propeller at a given RPM and velocity. The power coefficient is also changing with change of RPM and velocity, but considered neglectable. The selected C_p is from zero velocity and 1000 RPM.
The propeller torque T_q $[Nm]$ is calculated from the propeller power and the angular velocity.

$$T_q = \frac{P_p}{\omega} \tag{D.3}$$

where ω $[rad/s]$ is the angular velocity of the propeller.

2.3 Number of Motors

The number of motors is constrained to an equal number of counter rotating motors to eliminate the torque in the frame yaw axis. The length of the frame L, is only dependent of the propeller diameter D, and the number of actuators N_a, shown in Fig. D.3. Applying the law of sines the length of the frame is obtained.

$$\begin{aligned} \frac{\sin d}{D} &= \frac{\sin b}{B} = \frac{\sin c}{C} \\ L &= D\frac{1}{\sin(\pi/N_a)} \end{aligned} \tag{D.4}$$

2.4 Motor Performance

Parameter used in the optimization process is the motor speed constant $K_v[\frac{RPM}{V}]$, internal resistance $R_a[\Omega]$, torque constant $K_t[\frac{Nm}{A}]$, voltage constant $K_e[\frac{V}{rad/sec}]$, motor weight $[kg]$ and the continuous motor power $P_M[W]$.

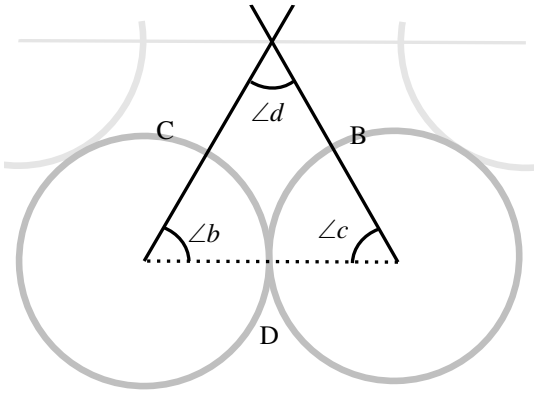


Figure D.3: *The length of the frame is only dependent of the propeller diameter and the number of actuators.*

The continuous motor power needs to be greater than power drawn from the propeller at a given angular velocity. The power drawn by the propeller is calculated as in (D.2) and the continuous power of the selected motors has to be greater than this. The mechanical time constant of the motor is given by [Magnussen et al., 2014]:

$$\tau_m = \frac{JR_a}{R_a K_t \zeta + K_t K_e} \tag{D.5}$$

where J is the total rotational inertia of the motor and propeller and ζ is the damping of the motor. All of the parameters above are known constants and can be grouped. Hence, equation (D.5) is linear in J. The propeller damping ζ is calculated as:

$$\zeta = \frac{T_q}{\omega K_t} \tag{D.6}$$

2.5 Flight Time

The power the propeller draws is calculated in (D.2). When a battery is selected the voltage and current capacities are known. From the propeller power and battery voltage the current consumption is calculated from the electrical power equation $P = UI$.

$$I = \frac{P_p}{V_{bat}} \tag{D.7}$$

$$h = \frac{I}{B_a} \tag{D.8}$$

where $I[A]$ is the current consumption, $h[hours]$ is the flight time and $B_a[Ah]$ is the battery capacity.

2.6 Multirotor Weight

From the algorithm perspective, the multicopter weight consists of separate parts such as the weight of the battery, motors, propellers, frame and the weight of the payload. The ESC's, controller board, wires etc. are a part of the payload. The datasheets give the parameters for calculating the weight for one battery, motor, and propeller. The motor and propeller parameters multiplied with the number of actuators give the total weight of the actuators. The frame weight is calculated as the length of the frame multiplied with $0.045\frac{kg}{m}$ which is the weight of a $7x9x1000mm$ carbon fiber tube.

2.7 Moment of Inertia

The moment of inertia of the multirotor is calculated as the sum of the actuator inertia and the frame inertia. Any additional components are considered as a mass in the center of gravity and not affecting the inertia. The actuator inertia I_a is calculated as the inertia for a point mass, and the frame inertia I_f as the inertia of a rod about its end.

$$I_m = m_a L_{ai}^2 \tag{D.9}$$

$$I_f = \frac{1}{3} m_f L^2 \tag{D.10}$$

where m_a is the actuator weight, L_{ai} is the perpendicular length from the rotation axis to the actuator and m_f is the weight of one frame length. The frame configuration is for some number of actuators not equal for the roll and pitch axis, illustrated in Fig. D.2. Thus making it necessary to calculate two sets of inertia, one for roll and one for pitch.

3 Design Optimization

The optimization problem is given by a set of linear constraints and an objective function.

$$\min \mathbf{g}^T \mathbf{x} \tag{D.11}$$

$$\text{subject to } \mathbf{A}\mathbf{x} \leq \mathbf{B} \tag{D.12}$$

where the matrix **A** and the vector **B** are the given constraints, and **g** is the objective function vector. The functions are written in Matlab and exported to the Cplex optimizer.

3.1 Boolean Variables

Boolean variables are used to represent the selected equipment. The Boolean variables are the first part of the **x**-vector

$$\mathbf{x} = [\delta_1, \delta_2, \delta_3, \delta_4, \delta_5 ... \delta_n, ...] \tag{D.13}$$

The Boolean values are either true or false and are constrained by:

$$\delta \leq 1 \tag{D.14}$$

$$-\delta \leq 0 \tag{D.15}$$

For instance each propeller type has a δ representing the active propeller, $\delta_p = \delta_{p1}, \delta_{p2}, ..., \delta_{pn}$. Since only one propeller can be selected, only one of the Boolean variables can be set at a time and the sum of all propeller deltas must equal one:

$$\sum_{i=1}^{N_p} \delta_i = 1 \tag{D.16}$$

where N_p is the number of propellers.

3.2 Continuous Variables

The continuous variables are the rest of the variables defining the **x**-vector.

$$\mathbf{x} = [...x_1, x_2, x_3, ..., x_n] \tag{D.17}$$

Datasheets define most of the selectable values for the process. Those values are constants and multiplying them is a linear operation. For instance multiplication of the selected propeller diameter and C_t is a linear operation. The values the optimizer generates are not constants; multiplication of those variables is a nonlinear operation, only constants are a valid multiplier in order to maintain the linear restriction. This, in addition to not ending up with non-existing equipment, are some of the benefits of using datasheets for the MILP solver. The optimizing routine only generates a few variables such as the n^2, the force to lift, the rotational inertia of the multicopter and the mechanical time constants of the actuator system.

3.3 Functions

The functions used to describe the program are simple functions containing a single instruction, and more advanced functions containing many instructions for a single operation. The simple functions used are summing a set of Boolean values or continuous variables $x(a_i)$ and assigning the result to a new variable $x(b)$.

$$x(b) - (\sum_{i=1}^{N} x(a_i)) = 0$$

Other simple functions used are scaling a value $x(a)$ to a new variable $x(b)$ with the scaling factor w_x

$$x(a)w_x - x(b) = 0$$

The optimization process uses three advanced functions, Rule 1, Rule 4 as described in [Mignone, 2002] and Rule19gt which is a modified version of Rule 19 in [Mignone, 2002]. The advanced functions implements logical inequalities based on the state of the δ values. For the following inequalities, $f(x)$ is a real linear

function defined as $f(x) = x(i_x) \cdot w_x + O$, where $x(i_x)$ is the variable and w_x is the gain of the operation and O is a constant offset. M is the maximum value of the function $f(x)$, and m is the minimum. z is the calculated value. $\widetilde{M} = max(0, -m)$ and $\widetilde{m} = min(0, M)$, ε is a very small positive number. The optimizer only knows the inequality "less or equal to", applying the ε the inequality becomes "less than".

Rule 1 will assign the value of the function $f(x)$ to z if the δ is set. Otherwise the value of z becomes zero.

$$\begin{aligned} IF\ \delta = 1\ THEN z &= f(x) \\ ELSE\ z &= 0 \end{aligned}$$

$$\begin{aligned} -M\delta + z &\leq 0 \\ m\delta - z &\leq 0 \\ -m\delta + z &\leq f(x) - m \\ M\delta - z &\leq -f(x) + M \end{aligned}$$

Rule 4 will assign the value of $f(x)$ to z only if every δ is set. Otherwise the value of z will becomes zero.

$$\begin{aligned} IF[(\bigwedge_{i=1}^{n} \delta_i) == N]\ THEN z &= f(x) \\ ELSE z &= 0 \end{aligned}$$

$$\begin{aligned} z + \widetilde{M}(\sum_{i=1}^{N} \delta_i) &\leq f(x) + N\widetilde{M} \\ z &\leq f(x) + M \\ -z - \widetilde{m}(\sum_{i=1}^{N} \delta_i) &\leq -f(x) - N\widetilde{m} \\ z &\leq -f(x) - m \\ z + \widetilde{m}\delta_1 &\leq 0 \end{aligned}$$

$$\vdots$$
$$z+\widetilde{m}\delta_i \leq 0$$
$$-z-\widetilde{M}\delta_1 \leq 0$$
$$\vdots$$
$$-z-\widetilde{M}\delta_i \leq 0$$

Rule 19gt checks the limit of the $f(x)$ value and clears the δ if the limit is out of range. This rule will not set the δ if the function value is within the limits.

$$IF\ f(x) > X_{max}\ THEN \delta = 0$$
$$IF\ f(x) < X_{min}\ THEN \delta = 0$$

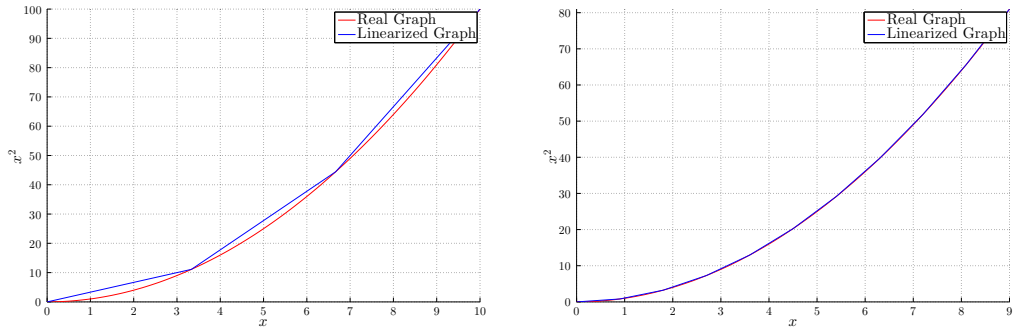
$$\delta M + f(x) + \varepsilon \leq\ X_{max} + M$$
$$\delta M - f(x) - \varepsilon \leq -X_{min} + M$$

3.4 Linearization

Many of the functions used are highly non-linear. For instance to linearize the conversion from n^2 used to calculate the thrust to n^3 used to calculate the power, the function is divided into equal segments connected with the function of a straight line $y = ax + b$. The optimizer will check the current n^2-value, and select the corresponding a and b for the given segment, illustrated in Fig. D.4. The a and b values of the linearization is calculated as:

$$a = \frac{X_2^p - X_1^p}{X_2 - X_1} \tag{D.18}$$
$$b = X_1^p - aX_1 \tag{D.19}$$

where X_1 is the lower limit of the segment, X_2 is the upper limit and p is the exponent value of the linearization. A list of the a's and b's represent the curve for the specific region and one δ value for every region is generated. Rule19gt determines which region is active and clears all the δ values which is out of the region. To set the non



3 segments 10 segments

Figure D.4: *Comparison of the linearization of the conversion from n^2 to n^3 for 3 and 10 segments.*

constrained δ value the sum of the δ values set must equal one. Once the correct δ value is set, Rule 1 linearizes the power function at the given x-value with $w_x = a$ and $O = b$.

3.5 Function implementation

Table D.1 shows which constrain is implemented for each of the functions. Some of the functions requires more than one variable to be calculated. For instance each propeller has different parameters, which for a given RPM will give different thrust. For the problem to remain linear those parameters must be treated as constants, and not to be put in a single variable. Hence calculating the propeller thrust requires one continuous variable per propeller to store the calculated thrust for that propeller with the specific RPM. Each of those calculated thrust values are multiplied with δ_p, as shown in Rule 1, and therefore only one of the values in the set of variables will be different than zero. The total thrust generated by the propeller F_{Th} is the sum of the variables in the set F_{Thi}, calculated using the simple functions.

$$F_{Th} = \sum_{i=1}^{N_p} \delta_i F_{Thi} \tag{D.20}$$

Every funcion in Table D.1 that uses the ”Sum set” function, uses a set of variables as described for the thrust. This process is more complex for the calculation of τ_m,

since this value is based on parameters of the battery, motor and propeller. Three nested for-loops were used to generate the every combination of the parts shown in Table D.2-D.4, resulting in a total of 150 variables only to calculate τ_m.

Table D.1: *Rules used for calculations*

	Rule 1	Rule 4	Rule 19gt	Sum set
Actuator weight				•
Propeller RPM	•		•	
Propeller Thrust	•			•
Frame Length	•			•
Frame weight				•
Propeller rotational inertia	•			•
Motor rotational inertia	•			•
Actuator rotational inertia				•
Actuator mechanical time constant		•		•
Roll- and pitch inertia	•		•	•
n^3 from n^2	•		•	•
Propeller power		•		•
Battery capacity [mAh]	•			•
Current consumption [A]	•			•
Flight Time inverse [1/h]	•			•

The number of actuators cannot easily be set to a variable and optimized since other parameters are dependent on this value with multiplication and division making the problem non-linear. To linearize this process the number of actuators is set as a constant, and the whole optimization process is tested with the given set of actuators. The optimization process were run multiple times with different number of actuators, the solution with the lowest objective function gives the optimal design.

4 Case Study

The propeller manufacturer APC has a free propeller database on their webpage with parameters for every propeller they produce. 6 different propellers shown in Table D.2 were chosen for the optimization problem, raging from 8-14 inches.

Table D.2: *Propeller types used in case study*

Propeller:	weight [g]	Diameter [inch]	Ct	Cp
# 1	7.1	8	0.1338	0.0897
# 2	9.1	9	0.1262	0.0837
# 3	11.9	10	0.1222	0.0797
# 4	13.9	11	0.1156	0.0746
# 5	17.9	12	0.1146	0.0727
# 6	25.0	14	0.1027	0.0630

The brush-less motor manufacturer Hyperion also provides datasheets for their products. The optimizer is free to chose motors from Table D.3, raging from 200-600 W. The K_v of the motors might be a bit too high for a multirotor application, as lower K_v means less RPM but higher torque. The principle is however the same, and later the motors can easily be replaced with a set of new ones.

Table D.3: Motor types used in case study

Motor:	weight [g]	Kv	Max Watt	Resistance [Ω]
# 1	32	3900	200	0.064
# 2	54	2640	375	0.063
# 3	54	3200	415	0.040
# 4	64	2608	430	0.048
# 5	79	1630	600	0.079

The batteries available in the optimization problem are the Zippy Compact from Hobbyking, shown in Table D.4. The maximum Ampere that the battery can output is the battery capacity [Ah] times the C-rating [1/h].

Table D.4: *Battery types used in case study*

Battery:	weight [g]	Capacity [mAh]	Voltage	C-rating
# 1	179	2200	12	25
# 2	95	1000	12	25
# 3	309	4000	12	25
# 4	618	8000	12	25
# 5	397	5000	12	25

4.1 Case 1 - Dynamic Performance

The multirotor is to lift an SLR camera at $0.5kg$ and to fly for 10 minutes. The task is to optimize the multirotor dynamics in terms of moment of inertia for the roll and pitch axis and the mechanical time constant τ_m for the actuators. The objective function is comparing inertia with time, hence the parameter gains must be set based on knowledge or experience. For this task they were all set to one.
On a standard laptop running an i7 Q740 processor @1.73GHz the solver spends 5 seconds solving the problem with 4 rotors, 8 seconds with 6 rotors and 10 seconds with 8 rotors. Results of the process is presented in Table D.5. The value of the objective function is lowest for the 4 rotor multicopter, and for this application the best suited choice. The actuator time constant is lower for the other two options, but the roll and pitch inertia are higher thus resulting in a higher value of the objective function.

4.2 Case 2 - Flight Time

If only the flight time is of interest the optimizer can be set to remove the flight time constraint, and only optimize for the longest flight time with the same payload as in Case 1. This gives the longest flight time achievable with the given hardware. The results are presented in Table D.6. Using 6 actuators it is possible to fly for 27 minutes while lifting the payload. The natural way of selecting the component for the longest flight time is to select the biggest propeller [#6], motor [#5] and battery [#4]. For 4 actuators this is the same solution as the optimizer hence the same result. For 6 actuators this configuration only gives 16 minutes of flight, compared to 27

Table D.5: *Results of case study 1*

	4 Actuators	6 Actuators	8 Actuators
Propeller [#]	5	4	5
Motor [#]	1	1	1
Battery [#]	3	1	5
τ_m [s]	0.27	0.24	0.24
Roll+Pitch Inertia $[kgm^2]$	0.04	0.09	0.19
Flight Time $[min]$	19	12	21
Number of constraints	11733	14851	17969
Number of variables	2833	3549	4265
Value of the objective function $[*1000]$	312	334	431

minutes for the optimal solution. For 8 actuators this does not give a valid solution.

Comparing the results from Case 2 with Case 1, it is shown that the actuator mechanical time constant and roll+pitch inertia is higher for Case 2, but also the flight time which defined the objective function. The value of the objective functions for Case 1 and Case 2 cannot be compared since the objective function itself is different for the two cases.

Table D.6: *Results of case study 2*

	4 Actuators	6 Actuators	8 Actuators
Propeller [#]	6	5	5
Motor [#]	5	1	2
Battery [#]	4	4	4
τ_m [s]	0.24	0.23	0.21
Roll+Pitch Inertia $[kgm^2]$	0.10	0.11	0.27
Flight Time $[min]$	23	27	23
Value of the objective function $[*1000]$	2.56	2.21	2.60

5 Conclusions

In this paper a design optimization for a multirotor has been described. The optimization problem was described as a mixed integer linear program, and solved with the Cplex optimizer. Running a design optimization process the design criteria can be extended beyond what is possible with only experience in the field. Based on a set of datasheets the algorithm chose the hardware resulting in the lowest value of the objective function. Optimal designs were generated for 4, 6 and 8 actuators, for two different cases. One to optimize the dynamics of the multirotor with a given payload and flight time, and only to optimize the flight time. The best dynamic performance was achieved using 4 actuators, while the longest flight time was achieved for 6 actuators. The design can be optimized for a wide range of constraints and objective functions and the problem can easily be extended to other functions.

REFERENCES

[Culligan et al., 2007] Culligan, K., Valenti, M., Kuwata, Y., and How, J. (2007). Three-dimensional flight experiments using on-line mixed-integer linear programming trajectory optimization. In *American Control Conference, 2007. ACC '07*, pages 5322–5327.

[IBM, 2013] IBM (Visited: March 2013). http://www.ibm.com.

[Lee et al., 2012] Lee, D., Periaux, J., Gonzalez, L., Srinivas, K., and Onate, E. (2012). Robust multidisciplinary uas design optimisation. *Structural and Multidisciplinary Optimization*, 45(3):433–450.

[Magnussen et al., 2014] Magnussen, Ø., Hovland, G., and Ottestad, M. (2014). Experimental study on the influence of controller firmware on multirotor actuator dynamics. In *Submitted to IROS*.

[Meindl and Templ, 2012] Meindl, B. and Templ, M. (2012). Analysis of commercial and free and open source solvers for linear optimization problems. *Eurostat and Statistics Netherlands within the project ESSnet on common tools and harmonised methodology for SDC in the ESS*.

[Melanie, 1999] Melanie, M. (1999). An introduction to genetic algorithms. *Cambridge, Massachusetts London, England, Fifth printing*, 3.

[Mellinger et al., 2012] Mellinger, D., Kushleyev, A., and Kumar, V. (2012). Mixed-integer quadratic program trajectory generation for heterogeneous quadrotor teams. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 477–483.

[Mignone, 2002] Mignone, D. (2002). The REALLY BIG Collection of Logic Propositions and Linear Inequalities. Technical Report AUT01-11.

[Mitchell, 2009] Mitchell, J. (2009). Integer programming: branch and cut algorithms integer programming: Branch and cut algorithms. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*, pages 1643–1650. Springer US.

[Mukesh et al., 2012] Mukesh, R., Lingadurai, K., and Karthick, S. (2012). Aerodynamic optimization using proficient optimization algorithms. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1–5.

[Ng and Leng, 2007] Ng, T. T. H. and Leng, G. S. B. (2007). Design of small-scale quadrotor unmanned air vehicles using genetic algorithms. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, 221(5):893–905.

[Propellers, 2013] Propellers, A. (Visited: March 2013). http://www.apcprop.com/v/index.html.

Paper E

Multicopter Design Optimization and Validation

Øyvind Magnussen, Morten Ottestad and Geir Hovland.

This paper has been published as:

Øyvind Magnussen, Morten Ottestad and Geir Hovland.*Modeling, Identification and Control, Volume 36, Number 2, Pages 67-79* Publisher: Norwegian Society of Automatic Control.

Multicopter Design Optimization and Validation

Øyvind Magnussen, Morten Ottestad and Geir Hovland.
*Department of Engineering
Faculty of Engineering and Science, University of Agder
Jon Lilletunsvei 9, 4879 Grimstad, Norway.

***Abstract* — This paper presents a method for optimizing the design of a multicopter unmanned aerial vehicle (UAV, also called multirotor or drone). In practice a set of datasheets is available to the designer for the various components such as battery pack, motor and propellers. The designer can not normally design the parameters of the actuator system freely, but is constrained to pick components based on available datasheets. The mixed-integer programming approach is well suited to design optimization in such cases when only a discrete set of components is available. The paper also includes an experimental section where the simulated dynamic responses of optimized designs are compared against the experimental results. The paper demonstrates that mixed-integer programming is well suited to design optimization of multicopter UAVs and that the modeling assumptions match well with the experimental validation.**

Keywords **Multicopter, multirotor, drone, UAV, mathematical modeling, design optimization, experimental validation.**

1 Introduction

The interest in multicopters (also called multirotors or drones) has increased significantly in recent years, both in academic research and in commercial applications. One example is the Prime Air multicopter by Amazon, see Fig. E.1 and [Amazon, 2015]. Other examples are the multicopters designed and developed to demonstrate acrobatic abilities, see for example [Hehn and D'Andrea, 2014] and the references therein. Despite the large number of published works, very little is published on design optimization of multicopters given an intended application and desired performance specifications.

In [Magnussen et al., 2014] a concept for design optimization of a multicopter based on mixed-integer programming was presented. In the current paper the same concept is described in more detail giving all the necessary information to allow a design engineer to repeat the design procedure. In normal practice, a set of datasheets of components such as battery pack, motors and propellers is available to the designer. Variables such as thrust per propeller or motor torque can not be designed freely, but must be chosen from a fixed set of available datasheets. The mixed-integer programming framework solves optimization problems that fall within this category. Certain variables are constrained to be discrete (for example a Boolean variable indicating selection of a particular component represented by a datasheet) while others are continuous variables (for example total weight and flight time). As demonstrated in the paper, the mixed-integer linear programming framework also allows for approximation of nonlinear functions. A nonlinear function is approximated by a set of discrete regions of the y-axis, each represented by a linear function. The accuracy of the approximation can be adjusted by changing the total number of linearized regions.

Numerical methods for nonlinear optimization normally suffer from drawbacks such as long computation times and the fact that the methods can end up in undesirable sub-optimal local minima. Examples are Newton-Raphson gradient search methods and evolutionary-based search methods, such as the Complex method, see for example [Whitney, 1969] and [Tyapin and Hovland, 2009] where the optimization algorithm took several hours to converge towards an acceptable but sub-optimal solution. A linear program (LP) on the other hand, using the interior point method, can be solved in polynomial time, see [Karmarkar, 1984]. The solution of an LP with a large number of variables can be found quickly using commercial solvers such as IBM CPLEX, see [IBM, 2015]. The solution of a mixed-integer linear program (MILP) is built on an LP solver and techniques such as branch-and-bound. The optimization solver formulated in this way can handle nonlinear function approximations and discrete design variables and returns a repeatable solution in relatively short time compared to an iterative nonlinear search algorithm where the solution can vary from run to run, depending on the initial conditions. It should be noted, however, that mixed-integer optimization problems are NP-hard, see for

example [Clausen, 1999].

The paper also contains a section presenting experimental validation of the performance of the multicopter actuators. Both the actuator rise-time constants and the total flight time used in the design optimization procedure are compared with the same parameters estimated from real measurements using the various components available to the designer. The validation compares the performance of four propellers and three motors against the simulation model in three different test cases: I) Longest possible flight time, no payload, II) Longest possible flying time, 1.5kg payload and III) Fastest possible motor response, no payload. For the three test cases, optimized solutions were calculated for multicopters with 4, 6 and 8 actuators, giving a total of 9 potentially different designs. The results demonstrate that the simulation model matches well with the experiments. The experimental validation increases the reliability of the optimization results.

The paper is organized as follows: in Section 2 a dynamic model of a multicopter actuator is presented. Section 3 gives a detailed description of the design optimization procedure. Section 4 presents results from the experimental validation based on three different test cases. Discussion and conclusions are presented in Section 5.

Figure E.1: *Example of multirotor UAV design by Amazon for transport and delivery of parcels.*

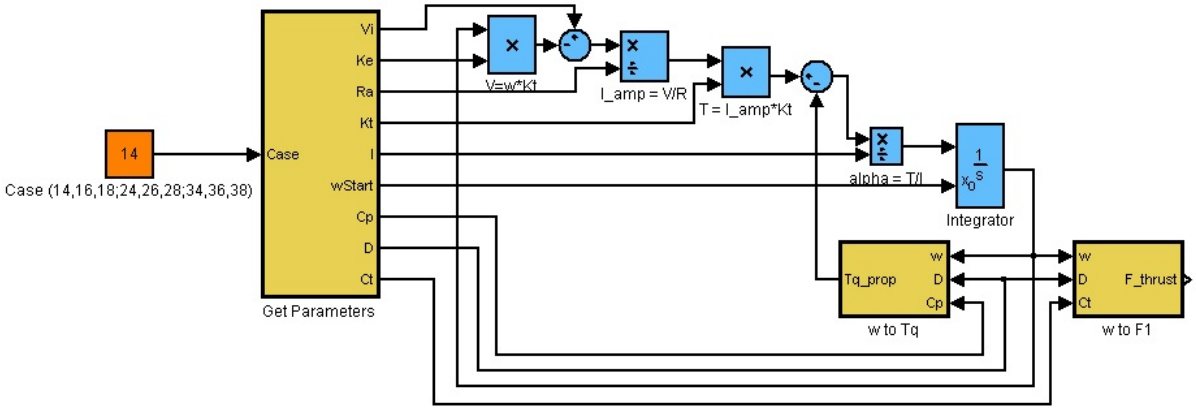


Figure E.2: *Simulink implementation of multirotor actuator dynamics.*

2 Modeling

A Simulink model of the multicopter actuator is shown in Fig. E.2. The model generates the simulated thrust response for the 9 different test cases used in this paper (test cases I, II and III with 4, 6 and 8 actuators). The propeller thrusts from the simulation model are later compared with experiments (force measurements using a load cell mounted underneath the actuator). The propeller thrust is calculated from eq. (E.1):

$$F = C_t \cdot \rho \cdot n^2 \cdot D^4 \tag{E.1}$$

where C_t is the thrust coefficient, ρ is the density of air, n is the rotor speed (rev/sec) and D is the propeller diameter. The actuator power W and torque T are calculated from eqs. (E.2)-(E.3):

$$W = C_p \cdot \rho \cdot n^3 \cdot D^5 \tag{E.2}$$

$$T = \frac{W}{\omega} \tag{E.3}$$

where C_p is the power coefficient and $w = 2\pi n$ is the rotor speed (rad/sec). The motor controller and actuator dynamics are given by the equations below:

$$V_m = V_i - K_e \omega \tag{E.4}$$

$$i_m = \frac{V_m}{R_a} \tag{E.5}$$
$$\tau_m = K_t \cdot i_m \tag{E.6}$$
$$\frac{d\omega}{dt} = \frac{1}{J}(\tau_m - T) \tag{E.7}$$

where J is the total inertia, V_m, i_m, τ_m are the motor voltage, current and torque, K_e, K_t and R_a are motor parameters and V_i is the supply voltage from the motor controller. Mechanical friction is not included in the dynamic model in eq. (E.7), since this information is difficult to obtain from the manufacturer's datasheets. Eqs. (E.1)-(E.7) are implemented in the Simulink model in Fig. E.2.

By combining eqs. (E.2)-(E.7) and using the fact that for DC motors the motor torque and back emf constants are equal, that is, $K_t = K_e$, the following transfer function can be defined:

$$\frac{\omega}{V_i}(s) = \frac{K_t}{\left(\frac{JR_a}{K_t^2 + D_\omega R_a}\right)s + 1} \tag{E.8}$$

where $D_\omega = \frac{T}{\omega} = \frac{\rho C_p n^3 D^5}{\omega^2}$. Assuming constant parameters in eq. (E.8), the time-constant would be $t_r = \frac{JR_a}{K_t^2 + D_\omega R_a}$. Since D_ω is a function of the propeller speed, this time-constant is an estimation.

3 Design Optimization

A quadratic program is defined as follows:

$$\min \quad \mathbf{x}^T\mathbf{Q}\mathbf{x} + \mathbf{g}^T\mathbf{x} \tag{E.9}$$
$$\text{subject to :} \quad \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{E.10}$$

where $\mathbf{x} \in \mathbb{R}^n$ is the state vector to be solved, $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is a positive semi-definite penalty matrix, $\mathbf{g} \in \mathbb{R}^n$ is a penalty vector, $\mathbf{A} \in \mathbb{R}^{m\times n}$ is the constraint matrix while $\mathbf{b} \in \mathbb{R}^m$ is the constraint vector. When the penalty matrix $\mathbf{Q} = \mathbf{0}$, eqs. (E.9)-(E.10) reduce to a linear program. When some elements x_i where $i \in [1, \cdots, n]$ are constrained to be integer variables, eqs. (E.9)-(E.10) are called mixed integer quadratic

program (MIQP) or, when $\mathbf{Q} = \mathbf{0}$, mixed integer linear program (MILP).

By constraining the integer variables further to accept only Boolean values (0 or 1), logical constraints can be incorporated and linked with the continuous variables in the optimization problem. The following rules 1 and 4 are taken from [Bemporad and Morari, 1999] and [Mignone, 2002]. Rules 19a and 19b are modified compared to [Mignone, 2002] ($\delta = 0$ instead of $\delta = 1$). In the rule definitions below the following notation is used:

$$\begin{aligned} m &= \min_{x \in X} f(x) \\ M &= \max_{x \in X} f(x) \\ \varepsilon &\quad \text{denotes a small, real, positive constant,} \\ &\quad \text{typically the machine precision.} \end{aligned}$$

It should be noted that equality constraints of the type $ax = b$ must be converted to two inequality constraints $ax \leq b$ and $-ax \leq -b$ to satisfy eq. (E.10). However, the solver used (CPLEX) allows specification of both equality and inequality constraints.

Rule 1: Product of δ and $f(x)$

Product of Boolean variable and function of continuous variables.

$$z = \delta \cdot f(x)$$

or

$$\begin{aligned} &\text{IF } [\delta == 1] \text{ THEN} && z = f(x) \\ &\quad \text{ELSE} && z = 0 \end{aligned}$$

is equivalent to

$$\begin{aligned} -M\delta + z &\leq 0 \\ m\delta - z &\leq 0 \\ -m\delta + z &\leq f(x) - m \\ M\delta - z &\leq -f(x) + M \end{aligned}$$

Rule 4: Product of several δ's and $f(x)$

Product of several Boolean variables and function of continuous variables.

$$z = \left(\bigwedge_{i=1}^{n} \delta_i \right) \cdot f(x)$$

or

$$\begin{aligned} &\text{IF } \left[\left(\bigwedge_{i=1}^{n} \delta_i \right) == 1 \right] \text{ THEN} && z = f(x) \\ &\quad \text{ELSE} && z = 0 \end{aligned}$$

is equivalent to

$$\begin{aligned} z + \tilde{M} \left(\sum_{i=1}^{n} \delta_i \right) &\leq f(x) + n\tilde{M} \\ z &\leq f(x) + \tilde{M} \\ -z - \tilde{m} \left(\sum_{i=1}^{n} \delta_i \right) &\leq -f(x) - n\tilde{m} \\ -z &\leq -f(x) - \tilde{m} \\ z + \tilde{m}\delta_1 &\leq 0 \\ &\vdots \\ z + \tilde{m}\delta_n &\leq 0 \\ -z - \tilde{M}\delta_1 &\leq 0 \\ &\vdots \\ -z - \tilde{M}\delta_n &\leq 0 \end{aligned}$$

where $\tilde{M} = \max(0, -m)$, $\tilde{m} = \min(0, -M)$.

Nonlinear positive function: $y = g(x)$

The scalar $y \geq 0$ equals the nonlinear function $g(\cdot)$ of the scalar $x \in [0, x_{max}]$. The set $\mathbf{z}$ approximates $g(x)$ by using a set of straight line segments $\mathbf{a} \in \{a_1, \cdots, a_N\}$, $\mathbf{k} \in \{k_1, \cdots, k_N\}$. Only one of the Boolean variables in the set $\boldsymbol{\delta} \in \{\delta_1, \cdots, \delta_N\}$ is true and specifies which element in $\mathbf{z} \in \{z_1, \cdots, z_N\}$ is used. $M = max(g(x))$. The nonlinear function approximation builds on rules 1, 19a and 19b in the equations below:

$$
\begin{aligned}
z_1 &= \delta_1(a_1 x + k_1) \\
&\vdots \\
z_N &= \delta_N(a_N x + k_N) \\
[z_1 \leq 0] &\rightarrow [\delta_1 = 0] \\
\left[z_1 \geq \frac{M \cdot 1}{N}\right] &\rightarrow [\delta_1 = 0] \\
&\vdots \\
\left[z_N \leq \frac{M \cdot (N-1)}{N}\right] &\rightarrow [\delta_N = 0] \\
[z_N \geq M] &\rightarrow [\delta_N = 0] \\
\sum_{i=1}^{N} \delta_i &= 1 \\
y &= \sum_{i=1}^{N} z_i
\end{aligned}
$$

Rule 19a: Implication of $f(x) \leq f_L$

Implication.

$$[f(x) \leq f_L] \rightarrow [\delta = 0]$$

is equivalent to

$$M\delta \leq f(x) - f_L + M + \varepsilon$$

Rule 19b: Implication of $f(x) \geq f_U$

Implication.

$$[f(x) \geq f_U] \rightarrow [\delta = 0]$$

is equivalent to

$$M\delta \leq -f(x) + f_U + M - \varepsilon$$

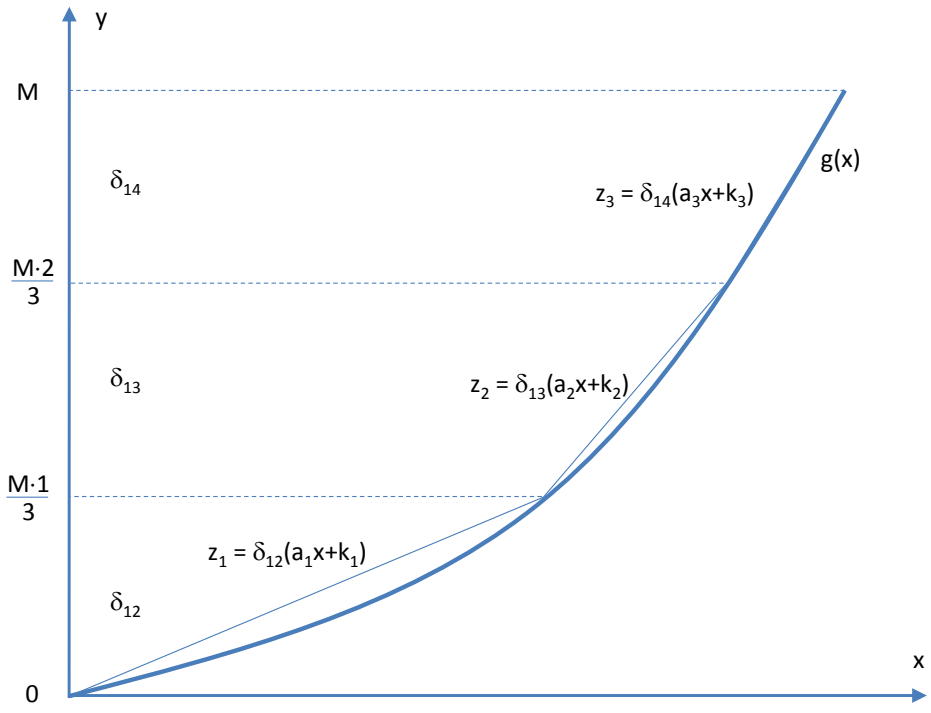


Figure E.3: *Illustration of mixed-integer linear approximation of a nonlinear positive function $g(x)$ with three regions (δ_{12}, δ_{13} and δ_{14}).*

Table E.1: *Boolean decision variables. If $\delta = 1$, then the corresponding component/region is selected. If $\delta = 0$, the component/region is not selected.*

Criterion	Objective function
Min. Time-constant	$\mathbf{g^T x} = x_{16}$
Min. Power	$\mathbf{g^T x} = x_{17}$
Min. Price	$\mathbf{g^T x} = x_3$
Max. Flight-time	$\mathbf{g^T x} = x_{21}$
Min. Inertia Roll	$\mathbf{g^T x} = x_{13}$
Min. Inertia Pitch	$\mathbf{g^T x} = x_{14}$

Table E.2: *Different candidate objective functions to be minimized/maximized depending on the requirement specifications.*

Variables	**Description**
$\delta_1 \cdots \delta_4$	Propeller 1-4 selection
$\delta_5 \cdots \delta_7$	Motor 1-3 selection
$\delta_8 \cdots \delta_{11}$	Battery 1-4 selection
$\delta_{12} \cdots \delta_{i_1}$	Used to calculate n^3
$\delta_{i_1+1} \cdots \delta_{i_2}$	Used to calculate inertia (roll)
$\delta_{i_2+1} \cdots \delta_{i_3}$	Used to calculate inertia (pitch)

Fig. E.3 illustrates how a nonlinear function can be approximated using a combination of discrete and continuous variables. In this example three regions δ_{12}, δ_{13} and δ_{14} are chosen to approximate the nonlinear function $g(x)$. The δ's are chosen to represent an equal spacing on the y-axis in the figure. In each region represented by δ_i a linear approximation $z_i = \delta_i(a_i x + k_i)$ is used.

The objective function $\mathbf{g}^T\mathbf{x}$ in eq. (E.9) can be defined as shown in Table E.2 depending on the application's requirement specifications. For payload transfer and package delivery, for example, the longest possible flight time or smallest possible power consumption may be desirable objectives. For acrobatics, on the other hand, the designer may instead want to select the solution giving the smallest possible time-constant or smallest possible roll and pitch inertias.

For the multirotor design optimization, the Boolean decision variables δ_i are defined as shown in Table E.1 and the continuous variables x_i are defined as shown

Table E.3: *Continuous variables used in the design optimization. "cg perp. to" = "center of gravity perpendicular to".*

Variables	Description
x_1	Total weight
x_2	Weight of actuator (motor+prop.)
x_3	Total price
x_4	Force to lift
x_5	n^2
x_6	n^3
x_7	Length of frame
x_8	Weight of frame
x_9	Actuator roll inertia
x_{10}	Actuator pitch inertia
x_{11}	Frame roll inertia
x_{12}	Frame pitch inertia
x_{13}	Total roll inertia
x_{14}	Total pitch inertia
x_{15}	Actuator rotational inertia $J_m + J_p$
x_{16}	Actuator rise-time constant
x_{17}	Propeller power
x_{18}	Current consumption
x_{19}	Required Ah
x_{20}	Battery Ah
x_{21}	1 / Flight Time
$x_{22} \cdots x_{25}$	n^2 for each propeller type
$x_{26} \cdots x_{28}$	Set of motor inertias J_m
$x_{29} \cdots x_{32}$	Set of propeller inertias J_p
$x_{33} \cdots x_{36}$	Set of battery Ah's
$x_{37} \cdots x_{40}$	Set of power values
$x_{41} \cdots x_{44}$	Set of current values
$x_{45} \cdots x_{48}$	Set of 1 / flight times
$x_{49} \cdots x_{96}$	Set of time-constants
$x_{97} \cdots x_{j_1}$	Distance from cg perp. to roll axis
$x_{j_1+1} \cdots x_{j_2}$	Distance from cg perp. to pitch axis

$x_{j_2+1} \cdots x_{j_3}$	Distance^2 from cg perp. to roll axis
$x_{j_3+1} \cdots x_{j_4}$	Distance^2 from cg perp. to pitch ax.
$x_{j_4+1} \cdots x_{j_5}$	Set of actuator roll inertias
$x_{j_5+1} \cdots x_{j_6}$	Set of actuator pitch inertias
$x_{j_6+1} \cdots x_{j_7}$	Set of frame roll inertia
$x_{j_7+1} \cdots x_{j_8}$	Set of frame pitch inertia
$x_{j_8+1} \cdots x_{j_9}$	Set of frame lengths
$x_{j_9+1} \cdots x_{j_{10}}$	Actuator Distance^2 Roll **z**-set
$x_{j_{10}+1} \cdots x_{j_{11}}$	Actuator Distance^2 Pitch **z**-set
$x_{j_{11}+1} \cdots x_{j_{12}}$	RPM **z**-set for n^3, calc. from n^2

in Table E.3. The complete mixed-integer state vector $\mathbf{x}$ introduced in eq. (E.9) is defined as follows:

$$\mathbf{x} = \left[\delta_1 \cdots \delta_{i_3}, x_1 \cdots x_{j_{12}}\right]^T \tag{E.11}$$

The indices i and j of discrete and continuous variables are summarized in Table E.4. Three different MILPs are created. One with the number of actuators constrained to $N_a = 4$, one with $N_a = 6$ and one with $N_a = 8$. The total number of discrete and continuous variables varies between the three programs and is equal to 1795, 2444 and 3091, respectively. To avoid multiplication of several free variables in the constraints, it was decided to create three separate MILPs rather than creating one optimization program with N_a as a free variable. To find the optimal solution, the three MILPs are run separately and the solution with the lowest objective function value is chosen.

In the following, the complete set of constraints used in the multicopter design optimization is listed. The constraints make use of the rules 1, 4, 19a, 19b and the nonlinear function approximation defined earlier in this section.

Boolean variable constraints:

$$0 \leq \delta_i \leq 1 \tag{E.12}$$

$$\sum_{i=1}^{4} \delta_i = 1 \qquad \sum_{i=5}^{7} \delta_i = 1 \tag{E.13}$$

Table E.4: *Optimization indices for $N_a = 4$, 6 and 8 actuators.*

Index	$N_a = 4$	$N_a = 6$	$N_a = 8$
i_1	241	241	241
i_2	561	721	881
i_3	881	1201	1521
j_1	98	99	100
j_2	100	102	104
j_3	102	105	108
j_4	104	108	112
j_5	106	111	116
j_6	108	114	120
j_7	110	117	124
j_8	112	120	128
j_9	114	123	130
j_{10}	274	283	290
j_{11}	594	763	930
j_{12}	914	1243	1570

$$\sum_{i=8}^{11} \delta_i = 1 \qquad \sum_{i=12}^{i_1} \delta_i = 1 \tag{E.14}$$

$$\sum_{i=i_1+1}^{i_2} \delta_i = 1 \qquad \sum_{i=i_2+1}^{i_3} \delta_i = 1 \tag{E.15}$$

Total weight and price:

$$x_2 = N_a \left(\sum_{i=1}^{4} (m_{p,i}\delta_i) + \sum_{i=5}^{7} (m_{m,i}\delta_i) \right) \tag{E.16}$$

$$x_1 = x_2 + x_8 + \sum_{i=8}^{11} \left(m_{b,i}\delta_i \right) \tag{E.17}$$

$$x_3 = N_a \left(\sum_{i=1}^{4} (P_{p,i}\delta_i) + \sum_{i=5}^{7} (P_{m,i}\delta_i) \right) + \sum_{i=8}^{11} \left(P_{b,i}\delta_i \right) \tag{E.18}$$

Individual propeller thrust and n^2:

$$x_4 = (x_1 + m_L)\, g \tag{E.19}$$

$$x_{22} = \frac{1}{N_a \rho C_{t,1} D_1^4} \delta_1 x_4 \tag{E.20}$$

$$\vdots$$

$$x_{25} = \frac{1}{N_a \rho C_{t,4} D_4^4} \delta_4 x_4 \tag{E.21}$$

$$x_5 = \sum_{i=22}^{25} x_i \tag{E.22}$$

Frame length and weight:

$$x_{j_8+1} = L_f(D, N_a)\, \delta_1 \tag{E.23}$$

$$\vdots$$

$$x_{j_9} = L_f(D, N_a)\, \delta_4 \tag{E.24}$$

$$x_7 = \sum_{i=j_8+1}^{j_9} x_i \tag{E.25}$$

$$x_8 = m_F N_a x_7 \tag{E.26}$$

Rotational inertia of motors and propellers:

$$x_{26} = J_{m,1} \delta_5 \tag{E.27}$$

$$\vdots$$

$$x_{28} = J_{m,3} \delta_7 \tag{E.28}$$

$$x_{29} = J_{p,1} \delta_1 \tag{E.29}$$

$$\vdots$$

$$x_{32} = J_{p,4} \delta_4 \tag{E.30}$$

$$x_{15} = \sum_{i=26}^{32} x_i \tag{E.31}$$

Time constants:

$$F_{req} = \left(N_a(m_{p,1}+m_{m,1})+m_{b,1}+m_L\right)\frac{g}{N_a} \tag{E.32}$$

$$n = \sqrt{\frac{F_{req}}{C_{t,1}\rho D_1^4}} \tag{E.33}$$

$$D_\omega = \frac{\rho C_{p,1} n^3 D_1^5}{(2\pi n)^2} \tag{E.34}$$

$$x_{49} = \delta_1\delta_5\delta_8\left(\frac{R_{a,1}(J_{m,1}+J_{p,1})}{R_{a,1}D_\omega + K_{t,1}^2}\right) \tag{E.35}$$

$$\vdots$$

$$F_{req} = \left(N_a(m_{p,4}+m_{m,3})+m_{b,4}+m_L\right)\frac{g}{N_a} \tag{E.36}$$

$$n = \sqrt{\frac{F_{req}}{C_{t,4}\rho D_4^4}} \tag{E.37}$$

$$D_\omega = \frac{\rho C_{p,4} n^3 D_4^5}{(2\pi n)^2} \tag{E.38}$$

$$x_{96} = \delta_4\delta_7\delta_{11}\left(\frac{R_{a,3}(J_{m,3}+J_{p,4})}{R_{a,3}D_\omega + K_{t,3}^2}\right) \tag{E.39}$$

$$x_{16} = \sum_{i=49}^{96} x_i \tag{E.40}$$

$$x_{16} \leq \tau_{max} \tag{E.41}$$

It should be noted here that the time-constants calculated in eqs. (E.35) and (E.39) are estimates based on the linear transfer function in eq. (E.8). If real measurements of the time-constants were available for each combination of battery, motor and propeller, these time-constants would be used in $x_{49}\cdots x_{96}$ instead of the estimates based on the datasheet information.

Calculate square of roll & pitch lengths: These constraints use the nonlinear function approximation rules. The corresponding Boolean variables are $\{\delta_{i_1+1},\cdots\delta_{i_2}\}$ and $\{\delta_{i_2+1},\cdots\delta_{i_3}\}$. The corresponding z-value sets are $\{x_{j_9+1},\cdots x_{j_{10}}\}$ and $\{x_{j_{10}+1},\cdots x_{j_{11}}\}$.

$$x_{j_2+1} = x_{97}^2 \tag{E.42}$$

$$\vdots$$

$$x_{j_3} = x_{j_1}^2 \tag{E.43}$$
$$x_{j_3+1} = x_{j_1+1}^2 \tag{E.44}$$
$$\vdots$$
$$x_{j_4} = x_{j_2}^2 \tag{E.45}$$

Actuator Roll Inertia ($m \cdot R^2$): The factor 2 in eq. (E.48), (E.51), (E.54) and (E.57) is used because there are two actuators per frame arm.

$$x_{j_4+1} = \delta_1 \delta_5 (m_{p,1} + m_{m,1}) x_{j_2+1} \tag{E.46}$$
$$\vdots$$
$$x_{j_5} = \delta_4 \delta_7 (m_{p,4} + m_{m,3}) x_{j_3} \tag{E.47}$$
$$x_9 = 2 \sum_{i=j_4+1}^{j_5} x_i \tag{E.48}$$

Actuator Pitch Inertia ($m \cdot R^2$):

$$x_{j_5+1} = \delta_1 \delta_5 (m_{p,1} + m_{m,1}) x_{j_3+1} \tag{E.49}$$
$$\vdots$$
$$x_{j_6} = \delta_4 \delta_7 (m_{p,4} + m_{m,3}) x_{j_4} \tag{E.50}$$
$$x_{10} = 2 \sum_{i=j_5+1}^{j_6} x_i \tag{E.51}$$

Frame Roll Inertia:

$$x_{j_6+1} = \frac{1}{3} L_f(D_1, N_a) m_F \delta_1 x_{j_2+1} \tag{E.52}$$
$$\vdots$$
$$x_{j_7} = \frac{1}{3} L_f(D_4, N_a) m_F \delta_4 x_{j_3} \tag{E.53}$$
$$x_{11} = 2 \sum_{i=j_6+1}^{j_7} x_i \tag{E.54}$$

Frame Pitch Inertia:

$$x_{j_7+1} = L_f(D_1,N_a)m_F\delta_1 x_{j_3+1} \tag{E.55}$$
$$\vdots$$
$$x_{j_8} = L_f(D_4,N_a)m_F\delta_4 x_{j_4} \tag{E.56}$$
$$x_{12} = 2\sum_{i=j_7+1}^{j_8} x_i \tag{E.57}$$

Total Roll & Pitch Inertias:

$$x_{13} = x_9 + x_{11} \tag{E.58}$$
$$x_{14} = x_{10} + x_{12} \tag{E.59}$$

Calculation of n^3: This constraint uses the nonlinear function approximation rules. The corresponding Boolean variables are $\{\delta_{12},\cdots\delta_{i_1}\}$. The corresponding z-value set is $\{x_{j_{11}+1},\cdots x_{j_{12}}\}$.

$$x_6 = x_5^{1.5} \tag{E.60}$$

Propeller Power:

$$x_{37} = C_p\rho D_1^5\delta_1 x_6 \tag{E.61}$$
$$\vdots$$
$$x_{40} = C_p\rho D_4^5\delta_4 x_6 \tag{E.62}$$
$$x_{17} = \sum_{i=37}^{40} x_i \tag{E.63}$$

Battery Ah:

$$x_{33} = B_{1,Ah}\delta_8 \tag{E.64}$$
$$\vdots$$
$$x_{36} = B_{4,Ah}\delta_{11} \tag{E.65}$$
$$x_{20} = \sum_{i=33}^{36} x_i \tag{E.66}$$

Current Consumption:

$$x_{41} = \frac{N_a}{B_{1,V}}\delta_8 x_{17} \tag{E.67}$$
$$\vdots$$
$$x_{44} = \frac{N_a}{B_{4,V}}\delta_{11} x_{17} \tag{E.68}$$
$$x_{18} = \sum_{i=41}^{44} x_i \tag{E.69}$$

Calculate Ah:

$$x_{19} = \left(\frac{T_F}{60}\right) x_{18} \tag{E.70}$$
$$x_{20} = x_{19} \tag{E.71}$$

Calculate the Inverse of Flight Time = A / Ah:

$$x_{45} = \frac{x_{18}}{B_{1,Ah}}\delta_8 \tag{E.72}$$
$$\vdots$$
$$x_{48} = \frac{x_{18}}{B_{4,Ah}}\delta_{11} \tag{E.73}$$
$$x_{21} = \sum_{i=45}^{48} x_i \tag{E.74}$$

4 Experimental Results

The purpose of the experiments presented in this section is to validate the models and assumptions used in the mixed-integer optimization presented in section 3. The experimental setup is shown in Fig. E.4. The validation presented in this paper has chosen both the 63% rise time of the actuator thrust as well as the total flight time as the parameters to compare against the simulation model in Fig. E.2. The rise times are measured directly from step responses in thrust, while the total flight time is estimated via the measured current when the step responses have reached steady-state.

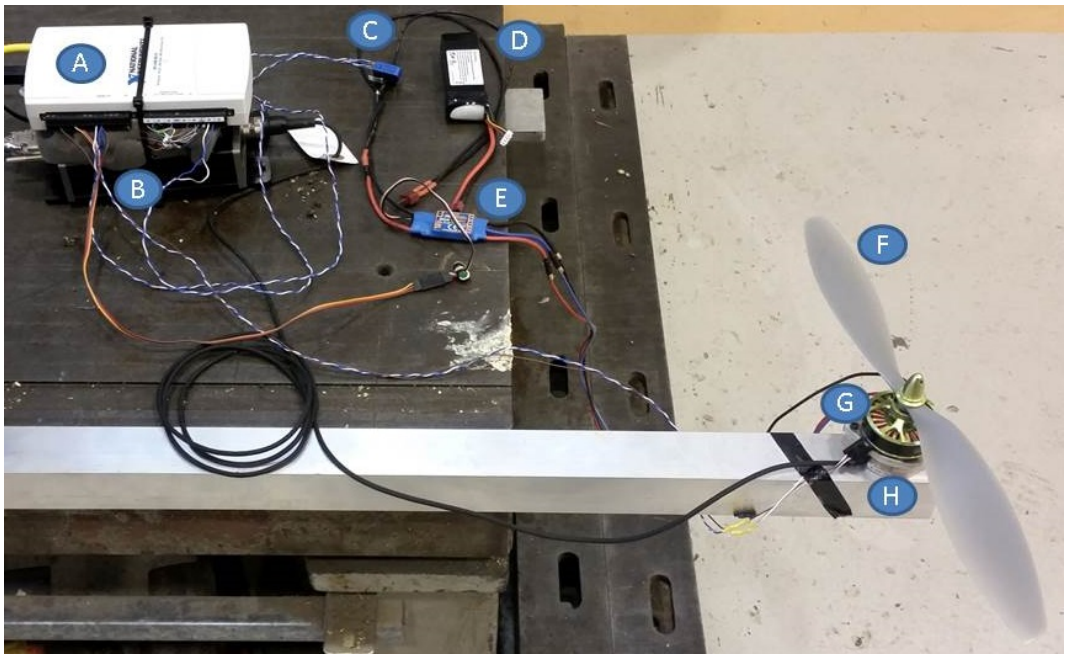


Figure E.4: *Experimental setup for validation. A: Digital/analog IO card, B: Load cell amplifier, C: Current sensor, D: Battery pack, E: Motor controller, F: Propeller, G: Motor, H: Load cell.*

Three different optimization test cases were studied, as summarized below:

- I: Longest possible flying time, no payload
- II: Longest possible flying time, 1.5kg payload
- III: Fastest possible motor response, no payload

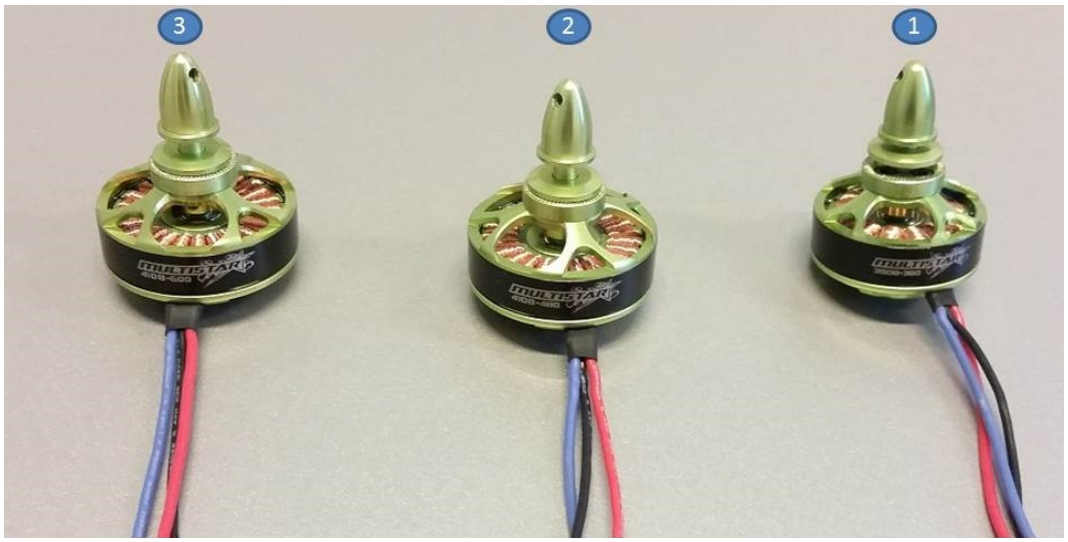


Figure E.5: *Three different motors used in the design optimization and validation tests.*

The design optimization was constrained to use either 4, 6 or 8 actuators in the multicopter. Section 4.1 contains a summary of the different components (datasheets) available for the design optimization. The experimental results for each test case are summarized in sections 4.2, 4.3 and 4.4.

4.1 Summary of Datasheets

The available datasheets are summarized in Table E.5, E.6 and E.7. In total three motors (see Fig. E.5), four propellers (see Fig. E.6) and four batteries were available (the same as in the design optimization procedure in section 3, Table E.1). Since all the batteries have a supply voltage of $11.1V$ and the experiments focused on validation of the 63% rise-time constant and flight time estimated via the measured current, the same battery could be used in all the tests, see Fig. E.7. The selection of battery in the experiments did not have an impact on the estimated time-constants and total flight time.

Figure E.6: *Four different propellers used in the design optimization and validation tests.*

Figure E.7: *Battery used in the design validation tests.*

Table E.5: *Datasheets: Motors.* $m_{m,i}$ *is the motor weight,* $D_{m,i}$ *is the motor diameter,* $S_{l,i}$,$S_{w,i}$ *are the shaft length and width,* $\rho_{m,i}$ *is the material density of the motor and* $P_{m,i}$ *is the motor price.*

Motor [#]	**1**	**2**	**3**
Model	4108-380KV Turnigy	4108-480KV Turnigy	4108-600KV Turnigy
$K_{t,i}$	380	480	600
$R_{a,i}$ (Ω)	0.222	0.148	0.123
$m_{m,i}$ (g)	111	111	111
$i_{max,i}$ (A)	17	22	26
$P_{max,i}$ (W)	360	380	400
$D_{m,i}$ (m)	0.047	0.047	0.047
$S_{l,i}$ (m)	0.012	0.012	0.012
$S_{w,i}$ (m)	0.004	0.004	0.004
$\rho_{m,i}$ (kg/m^3)	6800	6800	6800
$P_{m,i}$ ($)	31.36	31.36	31.36

4.2 Test Case I

In this test case the design was optimized for the longest possible flying time and no payload. As seen in Table E.8, propeller 4 was chosen when the design was constrained to use 4 and 8 actuators, while propeller 3 was chosen with 6 actuators. Motor 3 was chosen with 4 and 6 actuators, while motor 2 was chosen with 8 actuators. Battery 3 was chosen regardless of the number of actuators being 4, 6 or 8. Battery 3 has the highest capacity (16000 mAh), but also the highest weight. The selection of the battery with the highest capacity is not obvious. The solution with

Table E.6: *Datasheets: Batteries.* m_b *is the weight of the battery, while* P_b *is the price.*

Battery [#]	**1**	**2**	**3**	**4**
$m_{b,i}$ (g)	309	618	1236	308
$B_{i,Ah}$ (mAh)	4000	8000	16000	3300
$B_{i,V}$ (V)	11.1	11.1	11.1	11.1
$B_{i,imax}$ (A)	100	200	400	105
$P_{b,i}$ ($)	25.5	51.0	102.0	26.7

Table E.7: *Datasheets: Propellers.* $m_{p,i}$ *is the weight of the propeller,* p_i *is the pitch angle of the blade and* $P_{p,i}$ *is the propeller price.*

Propeller [#]	**1**	**2**	**3**	**4**
$m_{p,i}$ (g)	12	14	18	25
$D_{p,i}$ (m)	0.254	0.2794	0.3048	0.3556
$C_{t,i}$ (m)	4.7	4.7	4.7	4.7
$C_{p,i}$ (m)	0.1222	0.1156	0.1146	0.1027
p_i (deg)	10	11	12	14
$P_{p,i}$(\$)	4.7	5.0	5.6	7.8

4 actuators has both the longest flight time and the lowest price, so it is the prefered design in Test Case I.

Fig. E.8 illustrates the experimental results from Test Case I. The black curves show the simulated thrust response when using the Simulink model in Fig. E.2 with either 4, 6 or 8 actuators. The red curves show the measured thrust response under acceleration, while the green curves show the measured curves under deceleration. Note that the green curves are inverted about the steady-state value for easier comparison with the acceleration and the simulated response. The circles in the figure represent the values at the 63% rise time. The experimental results confirm that a linear model assumption is appropriate for the multicopter actuator, consisting of battery pack, controller, motor and propeller as shown in Fig. E.4. Tables E.11 and E.12 summarize the three different rise times (simulation, acceleration, deceleration). For Test Case I the simulated rise times are slightly faster than the measured values, ranging from 30 to 42ms faster.

4.3 Test Case II

In Test Case II the design was optimized for the longest possible flying time with a 1.5kg payload. As seen in Table E.9, propeller 4 and battery 3 were chosen regardless of the number of actuators being 4,6 or 8. Motor 3 was chosen when the design was constrained to 4 and 6 actuators, while motor 1 was chosen with 8 actuators. The solution with 8 actuators has the longest flight time. However, the price with 8 actuators is significantly higher than the price for the solutions with 4 and 6

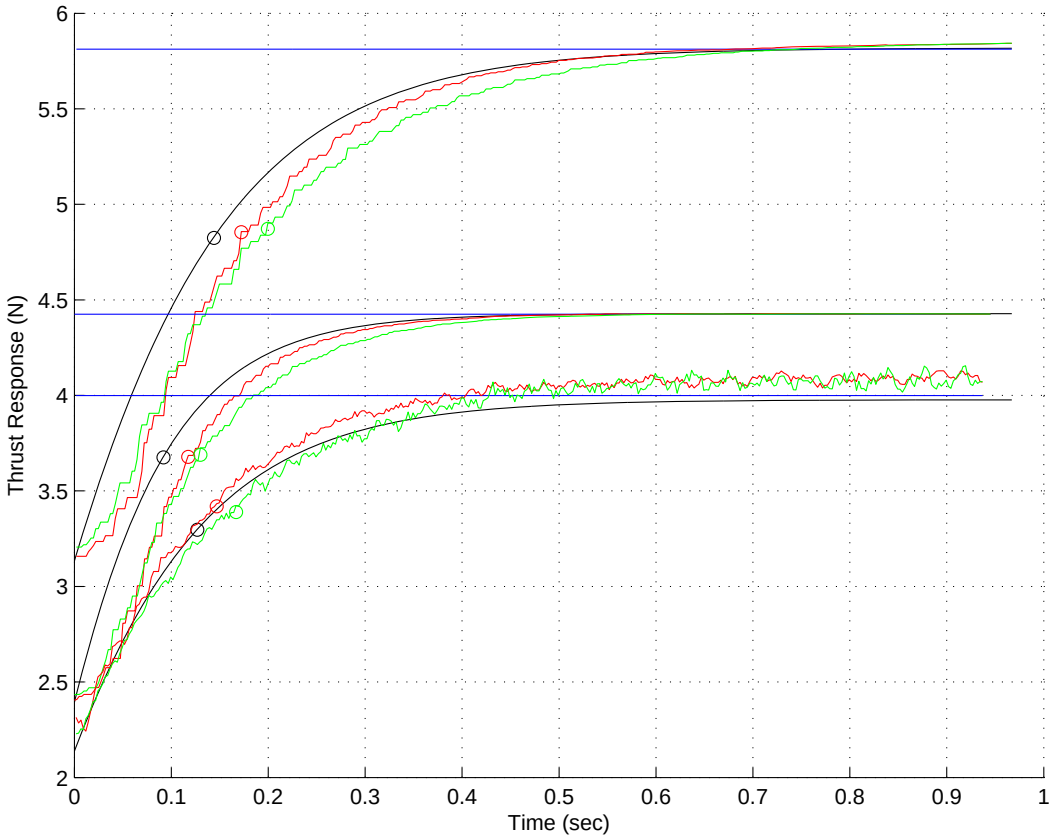


Figure E.8: *Thrust response per propeller, Test Case I. Top: 4 Actuators, Middle: 6 Actuators, Bottom: 8 Actuators. Blue: Input, Red: Acceleration, Green: Deceleration, Black: Simulation model. The circles represent the thrust at the* 63% *rise time.*

Table E.8: *Optimization Results: Test Case I.*

Number of Actuators	**4**	**6**	**8**
Propeller chosen [#]	4	3	4
Motor chosen [#]	3	3	2
Battery chosen [#]	3	3	3
Time constant (ms)	161	165	134
Flight time (min)	62.5	60.0	55.5
Price	227.4	290.2	352.9

actuators.

Table E.9: *Optimization Results: Test Case II.*

Number of Actuators	**4**	**6**	**8**
Propeller chosen [#]	4	4	4
Motor chosen [#]	3	3	1
Battery chosen [#]	3	3	3
Time constant (ms)	150	156	123
Flight time (min)	25.4	27.1	27.3
Price	227.4	290.2	352.9

Fig. E.9 illustrates the experimental results from Test Case II. The results for Test Case II are better than for Test Case I. Tables E.11 and E.12 summarize the three different rise times (simulation, acceleration, deceleration). For Test Case II the simulated rise times are slightly faster than the measured values, ranging from 10 to 15ms faster. Overall, the match between the simulation model and the experiments is good.

4.4 Test Case III

In Test Case III the design was optimized for the fastest possible motor response and no payload. As seen in Table E.10 propeller 1, motor 1 and battery 1 were chosen regardless of the number of actuators being 4,6 or 8. Since the time-constant (39ms) is the same for 4, 6 and 8 actuators, the natural choice is to use the solution

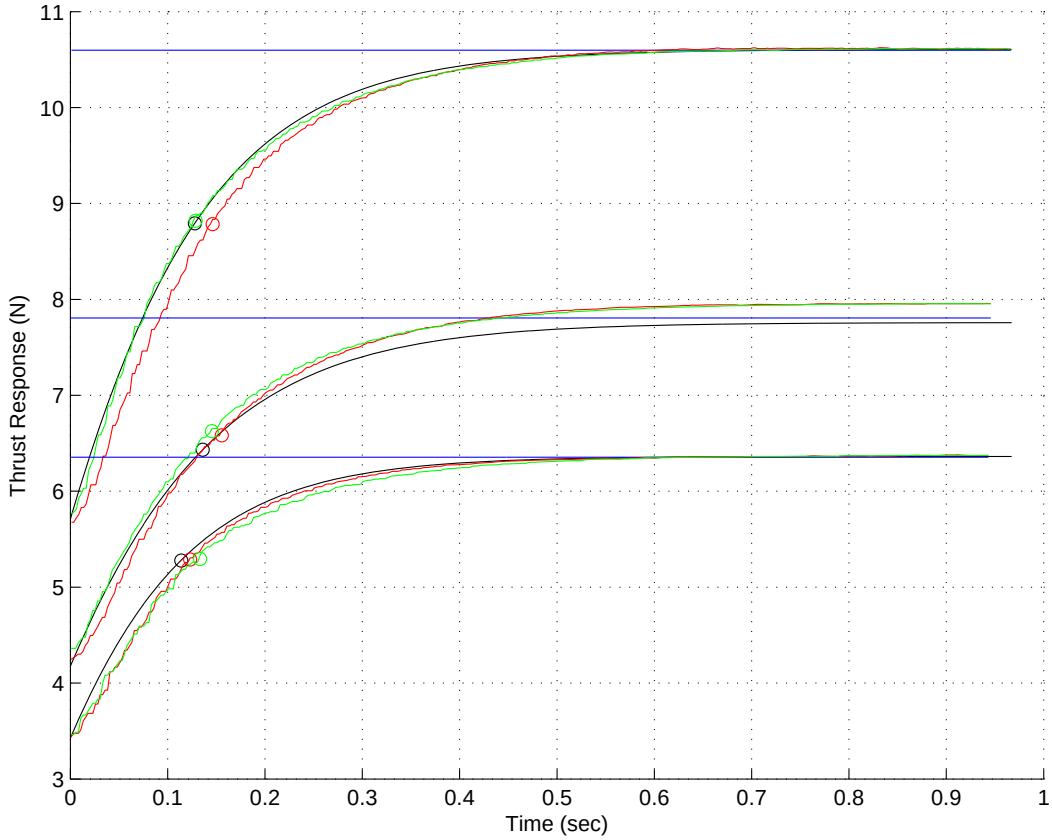


Figure E.9: *Thrust response per propeller, Test Case II. Top: 4 Actuators, Middle: 6 Actuators, Bottom: 8 Actuators. Blue: Input, Red: Acceleration, Green: Deceleration, Black: Simulation model. The circles represent the thrust at the* 63% *rise time.*

with 4 actuators since the flight time is the highest and the price is the lowest for this choice. Fig. E.10 illustrates the experimental results from Test Case III. The

Table E.10: *Optimization Results: Test Case III.*

Number of Actuators	**4**	**6**	**8**
Propeller chosen [#]	1	1	1
Motor chosen [#]	1	1	1
Battery chosen [#]	1	1	1
Time constant (ms)	39	39	39
Flight time (min)	37.2	29.4	23.8
Price	150.9	213.65	276.4

results for Test Case III are similar to Test Case II. Tables E.11 and E.12 summarize the three different rise times (simulation, acceleration, deceleration). For Test Case III the simulated rise times differ from the measured values by -21ms to +8ms.

Table E.11: *Measured rise times (63%, in ms) for the different test cases and number of propellers. T_a is for acceleration, T_d is for deceleration.*

Test	$T_{a,4}$	$T_{d,4}$	$T_{a,6}$	$T_{d,6}$	$T_{a,8}$	$T_{d,8}$
I	172	200	118	130	147	167
II	146	129	156	146	123	133
III	52	72	48	58	43	48

Table E.12: *Simulated rise times (63%, in ms) for the different test cases and number of propellers. T_i is the rise time with i propellers. ΔT_i is the time difference between T_i and the average of $T_{a,i}$ and $T_{d,i}$.*

Test	T_4	ΔT_4	T_6	ΔT_6	T_8	ΔT_8
I	144	-42	92	-32	127	-30
II	128	-10	136	-15	114	-14
III	41	-21	41	-12	53	8

Table E.13 shows the differences between the time-constants found from the experiments and the estimates based on the transfer function in eq. (E.8) and also used in

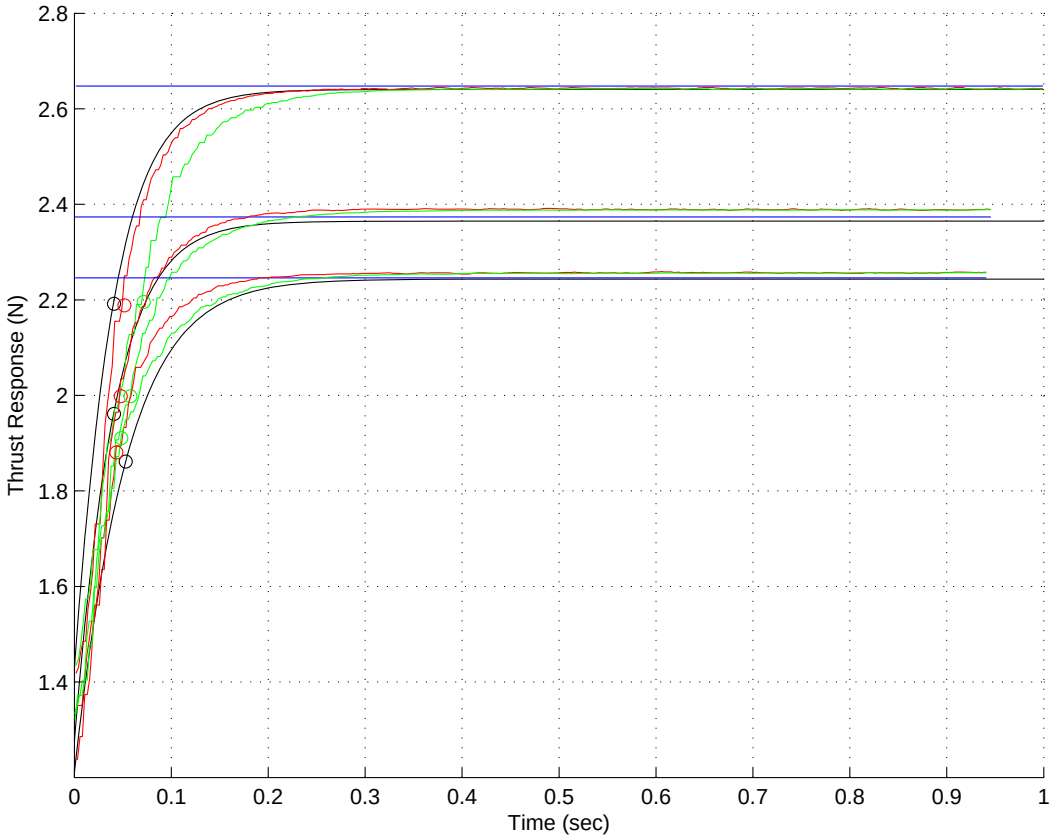


Figure E.10: *Thrust response per propeller, Test Case III. Top: 4 Actuators, Middle: 6 Actuators, Bottom: 8 Actuators. Blue: Input, Red: Acceleration, Green: Deceleration, Black: Simulation model. The circles represent the thrust at the* 63% *rise time.*

Table E.13: *Time-constants found from experiments (mean value of $T_{a,i}$ and $T_{d,i}$ in Table E.11) for the different test cases and number of propellers. ΔT_i is the time difference (in ms) between T_i and the time-constant estimates based on eq. (E.8).*

Test	T_4	ΔT_4	T_6	ΔT_6	T_8	ΔT_8
I	186	25	124	-41	157	23
II	138	-12	151	-5	128	-5
III	62	23	53	14	46	7

the optimization, eqs. (E.35), (E.39). The results are satisfactory with differences in the range -41 to +25ms. One error source is the time-constant assumption made in eq. (E.8) for a linear system.

The actual flight times can be estimated by dividing the battery capacity (Ah) by the measured current (A) when the step-responses have reached steady-state and by the number of actuators (N_a). Table E.14 shows the estimated flight times vs. the (inverted) flight times calculated in eqs. (E.72)-(E.74). The standard deviation between measured and estimated flight times in Table E.14 is 12.8%. Note that the battery voltage as stated in the datasheets is 11.1V, while the voltage when the battery is fully charged is more than 12V. In addition, the battery is not capable of keeping the voltage higher than 11.1V when discharged. Hence, both the estimated flight times and the ones found from experiments in Table E.14 probably overstate the actual flight times slightly.

Table E.14: *Estimated flight times in minutes from the experiments $T_{e,i}$ vs. flight times calculated in the optimization $T_{o,i}$ for test cases I, II and III and 4, 6 and 8 actuators.*

Test	$T_{e,4}$	$T_{o,4}$	$T_{e,6}$	$T_{o,6}$	$T_{e,8}$	$T_{o,8}$
I	66.6	62.5	57.5	60.0	62.0	55.5
II	29.4	25.4	28.5	27.1	33.2	27.3
III	33.9	37.2	25.3	29.4	20.5	23.8

5 Conclusions

This paper has presented an optimization framework for multirotors based on mixed-integer programming. The framework allows for efficient selection of optimal combinations of components from a set of available datasheets. Nonlinear functions can be approximated by using a combination of discrete and continuous variables. The designs can be optimized towards a set of different criteria, such as flight time, power consumption or dynamic performance, depending on the designer's preferences. Optimized designs with 4, 6 and 8 propellers are presented using real components available at, for example, [HobbyKing, 2015].

A simulation model of a multirotor actuator is presented and this model has been validated against experiments in three different test cases (longest possible flying time with or without payload, as well as fastest motor response). The experiments have used a step input in thrust and compared the 63% rise-time against the simulation model. The results are good with step-time differences between simulation and experiments in the range -21 to +42ms. The relatively small differences may be caused by unmodelled effects, such as motor friction and measurement delay. The difference between the estimated time-constants used in the design optimization and the time-constants estimated from the experiments have also been evaluated. The results are good with differences in the range -42 to +25ms. The total flight times have also been validated and have a standard deviation of 12.8% compared to the modeled flight times.

Overall, the results presented in this paper demonstrate that mixed-integer programming provides both a relatively accurate and efficient approach to multirotor design optimization from available datasheets. On an Intel i7-4770S 3.1GHz processor the different optimization problems were solved in typically 5-25 seconds using the IBM CPLEX solver. The experimental validation confirms that the modelling assumptions made in the design optimization formulation are reasonable and hence increase the confidence that the optimized designs actually meet the intended application's requirement specifications.

REFERENCES

[Amazon, 2015] Amazon (2015). Prime Air. http://www.amazon.com/b?node=8037720011. Accessed: 2015-04-07.

[Bemporad and Morari, 1999] Bemporad, A. and Morari, M. (1999). Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427.

[Clausen, 1999] Clausen, J. (1999). Branch and bound algorithms - principles and examples. http://www.diku.dk/OLD/undervisning/2003e/datV-optimer/JensClausenNoter.pdf. Accessed: 2015-04-07.

[Hehn and D'Andrea, 2014] Hehn, M. and D'Andrea, R. (2014). A frequency domain iterative learning algorithm for high-performance, periodic quadrocopter maneuvers. *Mechatronics*, 24(8):954–965.

[HobbyKing, 2015] HobbyKing (2015). Online shop. http://www.hobbyking.com. Accessed: 2015-04-07.

[IBM, 2015] IBM (2015). CPLEX Optimizer. http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer. Accessed: 2015-04-07.

[Karmarkar, 1984] Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

[Magnussen et al., 2014] Magnussen, Ø., Hovland, G., and Ottestad, M. (2014). Multicopter UAV Design Optimization. In *Proc. IEEE/ASME Intl. Conf. on Mechatronic and Embedded Systems and Applications*.

[Mignone, 2002] Mignone, D. (2002). The really big collection of logic propositions and linear inequalities. Technical Report AUT01-11, ETH Zurich.

[Tyapin and Hovland, 2009] Tyapin, I. and Hovland, G. (2009). Kinematic and Elastostatic Design Optimisation of the 3-DOF Gantry-Tau Parallel Kinematic Manipulator. *Modeling, Identification and Control*, 30(2):39–56.

[Whitney, 1969] Whitney, D. (1969). Optimum step size control for Newton-Raphson solution of nonlinear vector equations. *Automatic Control, IEEE Transactions on*, 14(5):572–574.