University of Copenhagen

Niels Bohr Institute

Acquiring Accurate State Estimates For Use In Autonomous Flight

Erik Bærentsen

Supervisor:
Troels C. Petersen

A thesis submitted in fulfillment of the requirements for the degree of Master of Science in Physics

March 16, 2017

Abstract

Autonomous flight of unmanned aerial vehicles (UAV) demands continuous and accurate estimates of both orientation and position. This thesis presents three methods for obtaining accurate orientation estimates from angular velocity measurements obtained with sensors mounted on the UAV combined with measurements of the Earth's magnetic and gravitational fields. The difficulties stemming from these sensors' inherent noise characteristics are discussed, and methods for reducing their effects are presented. It is shown that procedures continuously estimating the bias present in measurements of the angular velocity can lead to significantly lowered estimation errors. This thesis also presents an implementation of a Kalman filter for obtaining accurate position estimates. Using the estimated orientation the filter fuses acceleration measurements with GPS position and velocity estimates to produce accurate position estimates at a high rate. This project has hereby examined a range of aspects for converting noisy and diverse measurements to useful and accurate data and thus laid the groundwork required in order to build an autopilot for UAVs.

Acknowledgements

I would like to thank my supervisor Troels C. Petersen for devoting his time to our many meetings and for his valuable guidance throughout the project. Furthermore I would like to thank my colleagues Allan Hein and Riccardo Miccini for sharing their knowledge on writing embedded software for which I am grateful. I would also like to thank Steven Friberg for giving me the opportunity to write this project.

Contents

1 Introduction

1.1 About this thesis

This thesis was done in collaboration with the company Danish Aviation Systems (DAS) located outside Copenhagen, Denmark. A few months before this collaboration was set up I was employed at DAS. The goal of my employment there was to develop, from scratch, software for an autopilot allowing autonomous flight of fixed-wing planes and multirotor crafts. As the hardware for the autopilot was to be developed alongside my work on the software, it allowed for a unique possibility of writing a statistics-oriented thesis, throughout which continuous and direct influence on the design of the experiment/sensor testbed was possible.

The goal of this thesis was hence to get an understanding of what information is needed to achieve autonomous flight, as well as investigating the advantages and disadvantages of different ways of mapping the raw sensor readings to this more abstract information.

One of the priorities during the development of the autopilot has been to write maintainable and modular code. Since part of the project has been focused on testing and comparing different methods for achieving the same goal, such as estimating orientation, modular code is advantageous as it allows for parts of the code to be easily replaced with other code.

This priority is to some degree different from similar projects in the UAV community where computational performance seems to be the dictating factor in how the software is designed. Due to the limited performance and power of the hardware, embedded software does not allow for computationally sloppy implementations and so a lot of focus has been given in this project to achieving high performance though I have been reluctant to trade good coding practices where the increase in performance was not significant.

During the development of the autopilot software several smaller pieces of code have been written to aid the rest of the work. This includes functionality for simulating, visualizing and transferring data. Throughout this thesis I have included links to repositories on my GitHub [1] where most of this code can be found.

As my knowledge on this subject was rather limited when the project started out, I have continuously throughout the project tried to keep track of how all the parts of the autopilot were going to fit together in the end. To facilitate this knowledge acquisition I have more or less been working in parallel on all the topics included in this thesis (as well as on topics not included) and tried to get familiar with the basics of each topic by doing almost all implementations myself. This means that this thesis will touch on the majority of the topics needed for a functioning autopilot, but as not all parts are working yet, not all topics will be covered from basic concepts to final results.

1.2 Introduction

A wide range of tasks with highly varying complexity can potentially be solved by autonomous **U**nmanned **A**erial **V**ehicles (UAVs). Some tasks demand a high degree of accuracy in position control and a general ability to understand the immediate surroundings - an example hereof could be delivery of packages[1] to private residences. If the package was to be delivered at a doorstep then the ability to locate a driveway at a specific address combined with position control with a precision of a few centimeters might be necessary. Other tasks might consist of only slow maneuvers at altitudes where the surroundings are just open space. Such tasks could in many cases be solved without the vehicle having the ability to recognize objects in its surroundings and with position control accuracies of just a few meters.

Tasks not involving interactions with the UAV's surroundings, other than taking off from and landing on the ground, can in many cases be solved with knowledge of position, orientation and their derivatives as functions of time.

When this project started out the goal was to get as far as possible towards having an autopilot capable of autonomous flight. The ultimate final scenario became having a multirotor UAV capable of agile autonomous flight with accurate, robust state estimates based on data from onboard sensors combined with motion information extracted from a video stream from a camera mounted on the UAV. As the project progressed it later became clear that incorporating information from a camera mainly was of interest, if it could be build onto already well-performing orientation and position estimation algorithms based on the **I**nertial **M**easurement **U**nit (IMU) and the other, relative to the camera, simpler sensors that are mounted on the autopilot board. And so, building this foundation became the core of this project. Obtaining these accurate orientation and position estimates is however, not as simple as it might seem at first as this information is generally not directly what the sensors on board the vehicle measure. The task is further complicated by the fact that the data provided by the different sensors are in some cases distored by rather severe noise and in some cases even with noise characteristics changing over time.

Acquiring accurate estimates of the orientation of the vehicle is a key component in the process of achieving autonomous flight. Part of the position estimation problem depends on accurate information about orientation, and the control algorithms that provide both stability and maneuverability of the vehicle can only do their job if they are provided with accurate orientation estimates.

A method used in this thesis in both the position and the orientation estimation algorithms in order to ensure accurate and precise estimates at sufficiently high rates, is to fuse measurements available at high rates but which result in errors accumulating over time with slower unbiased estimates.

In the position estimation algorithm the high-rate measurements are provided by an accelerometer which measures accelerations of the vehicle. These measurements can for short periods of time (few seconds) be integrated to provide estimates about position and velocity but will inevitably start to diverge due to noise in the measurements. To avoid this build-up of errors these estimates

[1] Amazon.com is currently in the test phase of a service delivering packages from their storage facilities directly to costumers.

are fused with information from a GPS that provides measurements at a much slower rate but that are in turn accurate.

The dependence on accurate orientation estimates in the position estimation algorithm comes from the accelerometer measurements that are given in a coordinate frame fixed on the vehicle. To use a measurement from the accelerometer indicating an acceleration along its x-axis we must first know which way the x-axis is pointing in the real world.

In the orientation estimation task unbiased estimates can be obtained from measuring the directions of Earth's magnetic field and gravitational field in the vehicle's coordinate frame and comparing these to their direction in a static frame of reference. These estimates are then used to correct errors that accumulate from integrating measurements of the vehicle's angular velocities measured by an angular rate sensor.

Due to a limited time frame this project is not currently at a point were the software developed for the autopilot allows for the desired flight capabilities. A few flight tests have been carried out, but the results were not yet satisfactory. The multirotor test frame or platform, see figure 2.1, was only able to fly for a few seconds and was generally unstable. It was concluded that poor orientation estimates were the main reason though additional causes might become apparent when tested again with proper orientation estimates. This conclusion has led to a shift from tests done with real data to simulated data, as the simulated data allows for easier debugging and comparison of methods. The current state of the project is that fairly thoroughly tested orientation algorithms have recently been finished. As these algorithms are prerequisites for both position estimation and UAV control, these two parts still lack equally thorough testing, though working implementations of both have been completed.

The work carried out during this thesis and the software that I have developed along side it is intended for use on UAVs and is primarily focused on rotorcrafts (such as quadrocopters) and secondly on fixed-wing aircrafts (ordinary planes). However, the majority of the material presented here is not restricted to these two platforms and could as well be used in cars, boats, phones or any other platform equipped with the required sensors.

1.3 Thesis structure

The Second chapter describes the hardware developed by DAS that has been used during this project. It also presents models of how each type of measurement is related to the physical quantities that they provide information about as well as how noise affects the measurements and some initial ways of handling this noise.

The Third and Fourth chapters concern the position estimation problem. First, a range of coordinate systems needed to describe the problem and the data are introduced, followed by a description of the steps taken in the process of estimating the position. This is concluded with a toy example demonstrating the application of the Kalman filter used.

The Fifth and Sixth chapters deal with the orientation estimation problem. First, three different quantities used to describe and work with orientation are introduced, followed by a description of methods for estimating the vehicle's orientation. These methods are then tested on simulated

data and compared to each other. Chapter five is concluded with a discussion of what problems these methods face when being used on real sensor data.

The Seventh chapter briefly describes the simulations used to generate the test data used in several of the chapters as well as for testing some of the control algorithms.

The Eighth chapter describes a method for constructing smooth trajectories through a collection of waypoints for autonomous waypoint based flight. A discussion of a problem with this method is included as well. The chapter is concluded with a brief description of a control algorithm that has been used in the flight tests.

1.4 Notation

While working on this thesis and while reading through literature from different sources, quite a bit of confusion has arisen from the different styles of notation used by the different authors. In an effort to overcome this, I have adopted a somewhat pedantic style of notation which will hopefully spare readers from similar confusion.

All vectors are written as bold lowercase letters $\mathbf{v}$ and matrices as bold uppercase letters $\mathbf{M}$. Coordinate frames are written as regular uppercase letters A. When a vector is given with respect to a specific coordinate system it will have a leading superscript denoting this frame ${}^{A}\mathbf{v}$. When a vector is written without a leading superscript, as the first example $\mathbf{v}$, it refers to the geometric object itself. Single elements of vectors will be specified with a trailing subscript ${}^{A}\mathbf{v}_i$. Single elements of matrices will be specified by two trailing subscripts $\mathbf{M}_{ij}$.

Quantities facilitating rotations[2] will have a leading and a trailing superscript. The trailing superscript is to be read as "from" whereas the leading superscript is to be read "to" and thus with the superscripts lining up ${}^{B}\mathbf{v} = {}^{B}\mathbf{R}^{A}{}^{A}\mathbf{v}$ is a transformation of the coordinate tuple of vector $\mathbf{v}$ from frame A to frame B.

When referring to actual measurements, quantities will have a leading subscript m; i.e. ${}^{A}_{m}\boldsymbol{\omega}$ is a measurement of angular velocity given in frame A.

Quantities with hats are estimated values, i.e. ${}^{S}\hat{\mathbf{R}}^{E}$ is an estimate of orientation represented by a rotation matrix.

[2]For now it will suffice to think of such quantities as ordinary rotation matrices, though the unit quaternion, which will be introduced in chapter 5, will be the object behind most of the work on orientation estimation.

2 Hardware And Measurements

Working with the autopilot board and multirotor platform has been a big part of this project. Since the project has relied mostly on simulated data, as explained in section 1.2, the hardware will not be referenced as predominantly as expected at the outset of the project, in the following chapters. Instead a brief description of the autopilot board and the multirotor test frame is given here. Thus this chapter is intended to give the reader an idea of the characteristics of the data that is at the core of a fully functional autopilot.

Several revisions of the hardware have been developed during this project with the newest revision depicted in figure 2.2 and second newest depicted in figure 2.1 where it is mounted in the multirotor platform. These boards host a series of on-board sensors as well as connections for additional external ones. The on-boards sensors are:

- **MPU9250** [2]. 9 degrees of freedom IMU. The chip contains a three-axis accelerometer, angular velocity (angular rate) sensor and magnetometer.
- **LSMDS3** [3]. 6 degrees of freedom IMU containing a three-axis accelerometer and angular rate sensor.
- **LIS3MDL** [4]. Three-axis magnetometer.
- **BMP280** [5]. A barometric pressure sensor.

At the time of writing two additional external sensors have been part of the development process:

- **LOCOSYS MC-1513** [6]. GPS.
- **LIDAR-Lite 2** [7] A one-dimensional laser rangefinder.

An aspect of significant importance for flying vehicles is robustness, which can partly be achieved through system-wide redundancy. Designed towards this and in an effort to handle some of the difficulties described in the following, the autopilot board has been equipped with two sets of angular rate sensors, accelerometers and magnetometers. However, incorporating redundant sensors in the estimation algorithms is not a trivial task as will be detailed in the following. The visualizations below of data from the accelerometer, the angular rate sensor and the magnetometer has been collected using the MPU9250 chip only.
The accelerometer and angular rate sensor in the MPU9250 chip can be sampled at 1000 Hz whereas the magnetometer can be sampled at 100 Hz. The GPS can only be sampled at 10 Hz.

2.1 Sensors and Measurement Models

Before going into detail of the estimation tasks, the following will introduce the data that will be used during actual flight, what information is contained in the data and what difficulties and

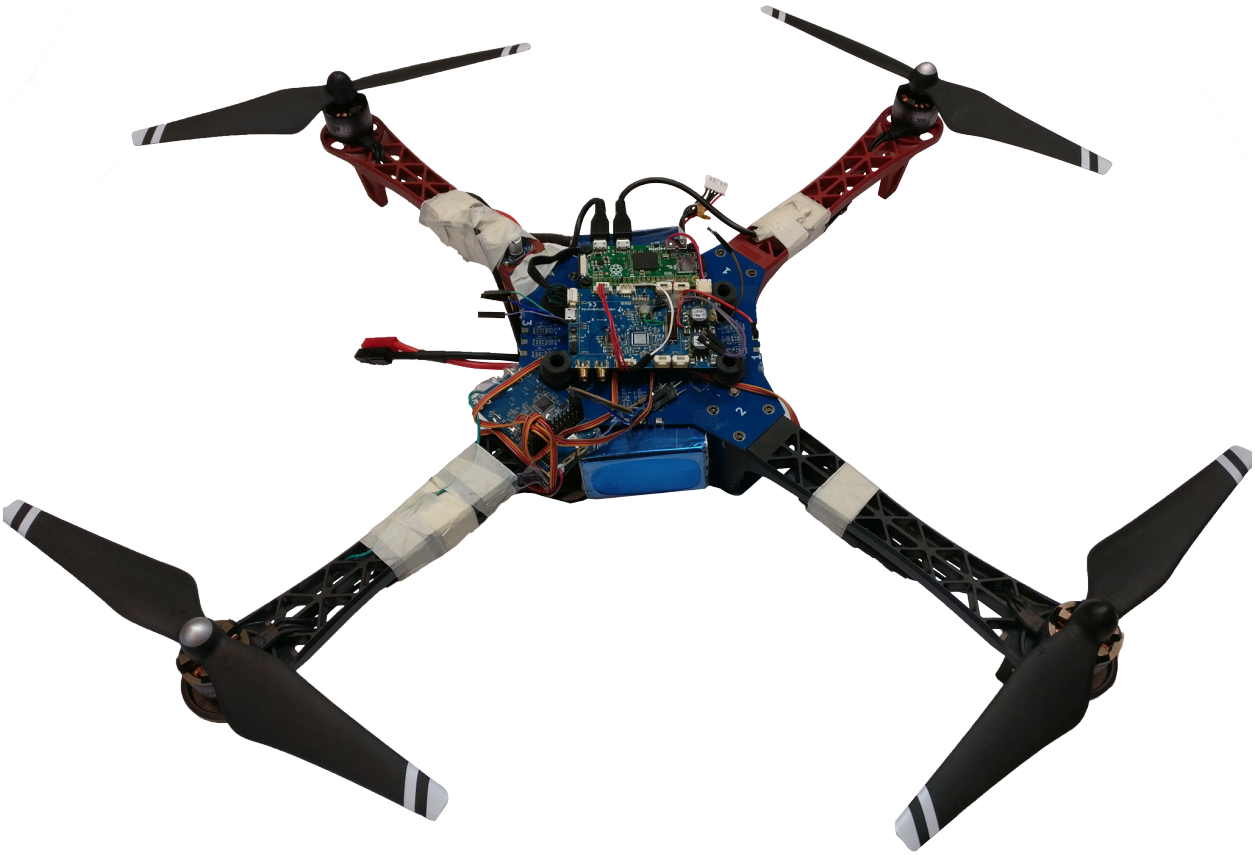

FIGURE 2.1
Multirotor test frame used in flight tests. Mounted on the platform is the second newest version of the autopilot board (blue, center) running the estimation algorithms, control algorithms etc. A Raspberry Pi Zero (green, rear) allowing for transfer of data from the autopilot over wifi for monitoring sensor data and algorithm outputs as well as acting as a programmer board. A radio link (blue, front left) used for receiving inputs from a joystick for manual, stabilized flight is also seen.

problems that needs to be taken into account and handled before the data can be used.
The distinction between coordinate frames is important in the following as we are generally interested in how measurements made in each sensor's coordinate frame S is related to the physical quantities represented in the earth frame E. A more detailed definition of the different coordinate frames and their connections will be given in the following chapters.

2.1.1 Accelerometer

Ideally the accelerometer provides perfect information about accelerations from all forces applied to the platform. If this was the case and assuming this information to be given in the Earth frame it would be a simple task to estimate the position of the platform at any time, t, given an initial position $\mathbf{r}_0 = \mathbf{r}(t = 0)$. Of course, even in this ideal case, the data from an accelerometer is the sum of accelerations leading to change in velocity and the effect of gravity, thus the gravitational

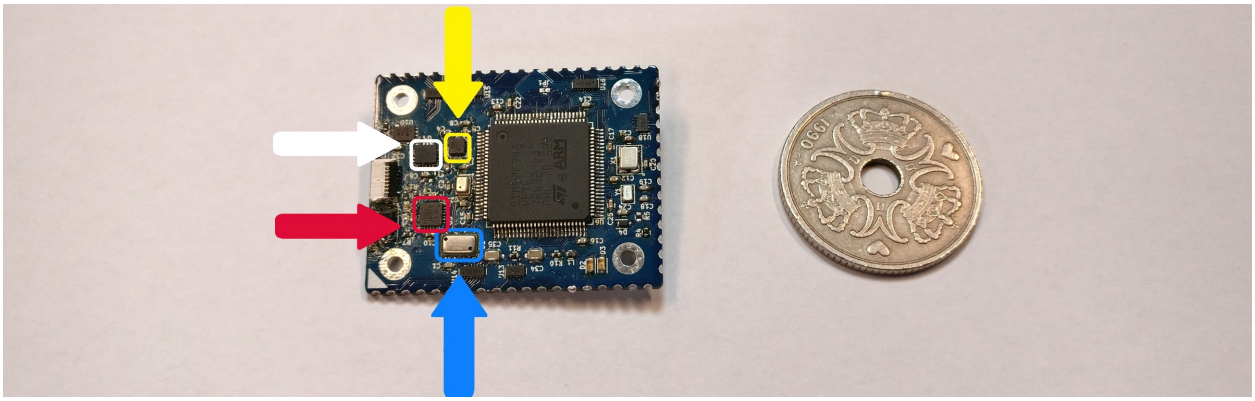

FIGURE 2.2
Newest revision of the autopilot hardware. This picture is included in order to give the reader an idea of the dimensions of the sensors used in this project. Yellow: LSMDS3. White: LIS3MDL. Red: MPU9250. Blue: BMP280. The big chip in the middle is the processor.

component needs to be known and removed before velocity and position estimates can be done. We can model this measurement as:

$$\overset{S}{m}\ddot{\mathbf{r}}(t) = {}^{S}\mathbf{R}^{E}(t)\left({}^{E}\ddot{\mathbf{r}}(t) + {}^{E}\mathbf{g}\right) \tag{2.1}$$

where $\overset{S}{m}\ddot{\mathbf{r}}(t)$ is the measured acceleration at time t in the sensor frame S, ${}^{S}\mathbf{R}^{E}(t)$ is a rotation matrix representing the orientation of the sensor relative to the Earth frame E, ${}^{E}\ddot{\mathbf{r}}(t)$ is the true acceleration in the Earth frame and ${}^{E}\mathbf{g}$ is the gravitational acceleration in the Earth frame.

The thing to note here which is explicitly captured in this formula is that an accelerometer that is completely still, with no change in velocity, measures an acceleration of $1g$ in some direction due to gravity and not just $0\,\mathrm{m/s^2}$.

A model that better captures the characteristics of the data provided by the accelerometer in the MPU9260 chip is given below. Here the measurements are affected by a time-varying bias ${}^{S}\boldsymbol{\mu}(t)$, a time-varying scale factor for each axis $\mathbf{K}(t) = \mathrm{diag}\{k_x(t), k_y(t), k_z(t)\}$, and a term of zero-mean normally distributed noise ${}^{S}\boldsymbol{\xi}(t)$:

$$\overset{S}{m}\ddot{\mathbf{r}}(t) = \mathbf{K}(t)\ {}^{S}\mathbf{R}^{E}(t)\left({}^{E}\ddot{\mathbf{r}}(t) + {}^{E}\mathbf{g}\right) + {}^{S}\boldsymbol{\mu}(t) + {}^{S}\boldsymbol{\xi}(t) \tag{2.2}$$

A visualization of a collection of measurements from the accelerometer can be found in figure 2.3. From the figure it is clear that the noise characteristics are varying with time. It is however not directly possible to establish whether the changes in measurements are caused by change in scaling factors, bias or both.

2.1.2 Angular Velocity Sensor

Similarly, if the angular rate sensor could provide perfect information about the angular velocity, the task of estimating the orientation of the setup would be relatively easy. As in the accelerometer case however, the angular rate sensor needs to be modeled in a way that takes into account a time-varying bias and scale factor. The angular rate measurements are in general of interest only in the sensor frame S which is why the true rate ω is here given in the sensor frame as opposed to the true

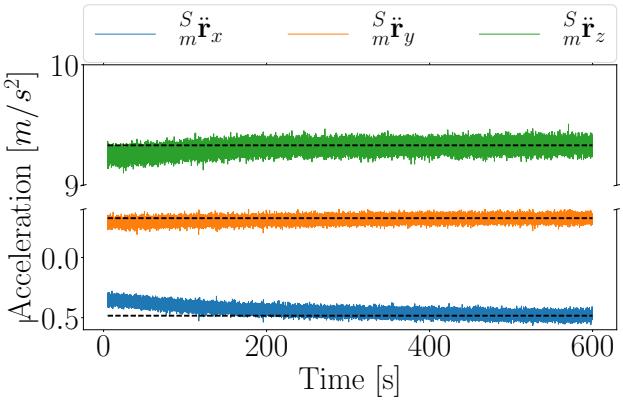

FIGURE 2.3
Samples of all 3 axes of the accelerometer over a period of 10 minutes while the sensor has not moved. Though the measurements should be constant it is clear that they change over time. This is largely due to the sensor and board heating up after being turned on. Note the broken axis visualization i.e. there is a gap from 0.4 to 9.0 on the y-axis. The values has been converted from Least Sigificant Bits (LSB) per g to m/s^2 using the value specified in the sensor's datasheet.

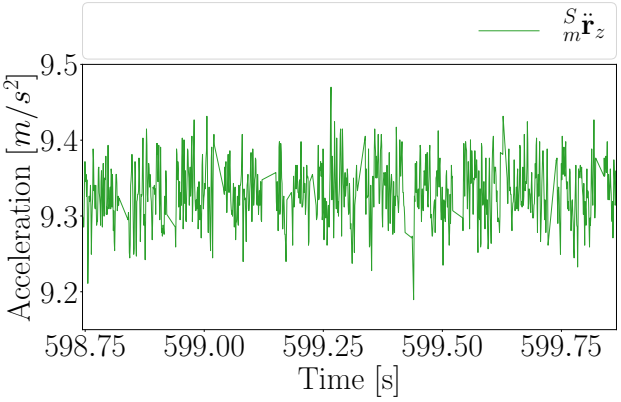

FIGURE 2.4
The last 1000 samples of the data to the left showing the random noise in the data. Every 0.1 seconds some data is missing as the process of writing data to the board SD card on the autopilot takes a while. This does not happen in flight.

the acceleration above given in the Earth frame.

$$ {}^{S}_{m}\boldsymbol{\omega}(t) = \mathbf{K}(t)\,{}^{S}\boldsymbol{\omega}(t) + {}^{S}\boldsymbol{\mu}(t) + {}^{S}\boldsymbol{\xi}(t) \tag{2.3} $$

A visualization of raw readings from the angular rate sensor can be found in figure 2.5. Again, the time varying characteristics are apparent.

2.1.3 Magnetometer

The magnetometer measures magnetic fields along 3 axes. These measurements are used in the orientation estimation algorithms, as measurements of the direction of Earth's magnetic field in the sensor frame can be used to infer the orientation of the platform by comparing the measured direction to the direction in a reference frame. These measurements are however generally affected by biases often several times larger than the actual signal. The model for magnetometer measurements presented here is similar to the one used in [37]:

$$ {}^{S}_{m}\mathbf{b}(t) = \mathbf{S}(t)\,{}^{S}\mathbf{R}^{E}(t)\,{}^{E}\mathbf{b}(t) + {}^{S}\mathbf{h}(t) + {}^{S}\boldsymbol{\xi}(t) \tag{2.4} $$

If samples of just Earth's magnetic field (no distortions or noise) were collected uniformly for all possible directions then these would lie uniformly on the surface of a sphere centered at the origin with radius equal to the field strength. However, actual measurements lie roughly on an ellipsoid (a potentially rotated and unevenly stretched sphere) translated away from the origin. The translation is, in the model above, described by the vector ${}^{S}\mathbf{h}(t)$. This bias originates from the platform that the sensor is attached to, where magnetized iron and electrical currents will distort measurements

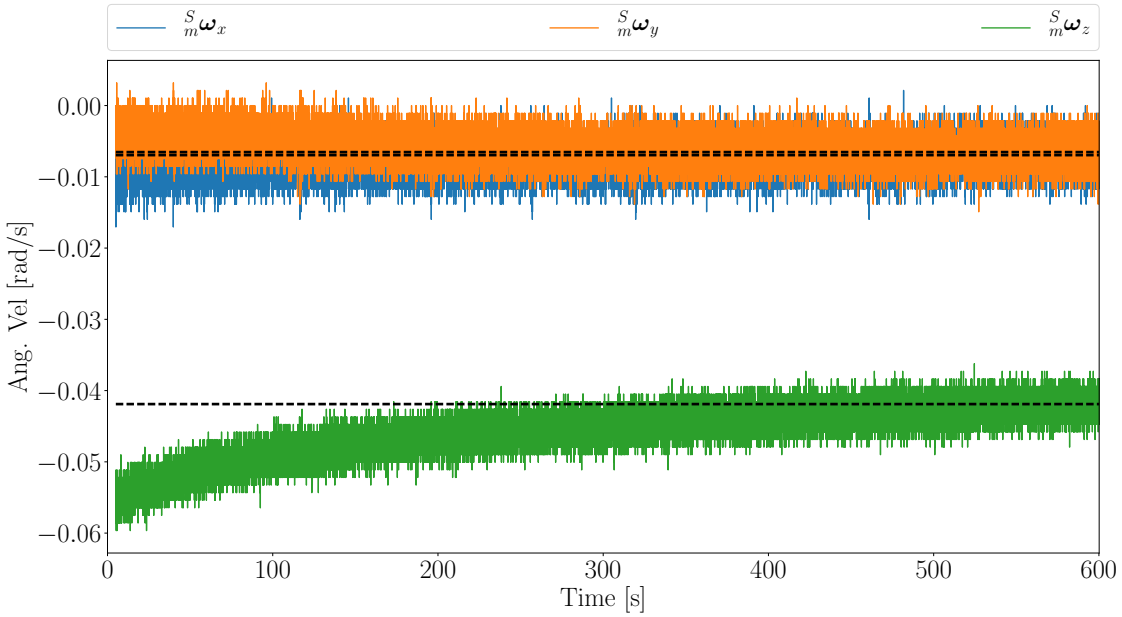

FIGURE 2.5
Samples of all 3 axes of the angular rate sensor over a period of 10 minutes while the sensor has not moved. The values has been converted from LSB/°/s using the value specified in the sensor's datasheet.

of Earth's magnetic field [39, 35]. The distortion of the sphere to an ellipsoid is captured by $\mathbf{S}(t)$ which is caused by magnetic materials also fixed in the platform.[1] The sensor will generally also have an internal time-varying change in scale and a bias which is also contained in $\mathbf{S}(t)$ and $\mathbf{h}(t)$ respectively. A visualization of a collection of measurements over time can be seen in figure 2.6.[2] A 3D-visualization is given in figure 2.8 in the next section on calibration.

2.1.4 GPS

The position as measured by the GPS is given in global position coordinates (latitude, longitude, height). These coordinates will, depending on the formulation of the position estimation problem, have to be converted to Cartesian coordinates to be related to measurements done on the platform. This relationship will be explained in detail in chapter 3 where the coordinate systems relevant in this conversion is presented. Figure 2.7 visualizes position data collected over a period of two days[3] converted to a local Cartesian coordinate system.

[1] Generally both types of distortions can just as well be caused by material not fixed to the platform but the task of compensating for these is well outside the scope of this thesis.

[2] The 50 µT mentioned in the plot caption is from an illustration of the intensity of the magnetic field across the Earth, which together with figures of the magnetic field's inclination and declination beautifully visualizations the variations in Earth's magnetic field. The intensity, inclination and declination illustrations respectively can be found at [8, 9, 10]

[3] The data was collected by an employee at DAS for another project and did not use the same GPS as the one used with the autopilot. The data should not differ between the two however.

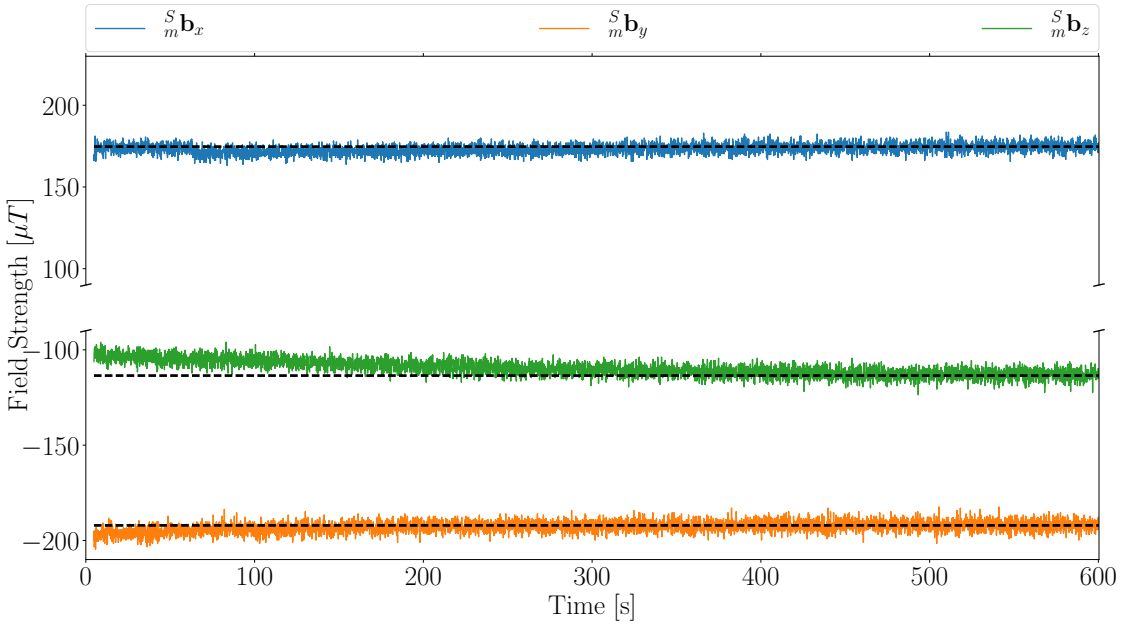

FIGURE 2.6
Samples of all 3 axes of the magnetometer over a period of 10 minutes while the sensor has not moved. Note that the strength of the magnetic field as measured by the sensor is far from the 50 µT that is the average strength in Denmark.

2.2 Sensor Calibration

As will be clear in the following chapters, the task of calibrating the sensors plays an important role in the effort towards acquiring both accurate and precise estimates. In general the calibration procedures can be labeled as being one of two types which will be denoted static versus dynamic calibrations or one-time calibrations versus continuous calibrations. Each of the two has their own advantages and disadvantages. The first type of method refers to calibrations done before each use of each sensor, e.g. once for a given platform or before each take off of a UAV. These are generally simpler than the second type of method. A problem is that these methods are unable to take into account the possible time dependence of a proper calibration. The second method refers to ongoing calibration procedures, where the sensors are calibrated while they are being used. Not surprisingly these methods can be more difficult to get right and can even lead to even poorer sensor data. Often, in practice, a combination of static and dynamics methods will be used since a good initial estimate of calibration parameters can improve the dynamic methods considerably.

Here, some of the static methods will be introduced, while the dynamic methods will be given in the later chapters as they are generally closer coupled to the estimation algorithms themselves.

2.2.1 Accelerometer

Estimates of the accelerometer bias and scaling factor can be found by aligning each of the sensors axes parallel and anti-parallel with the direction of gravity. When measurements of what should be

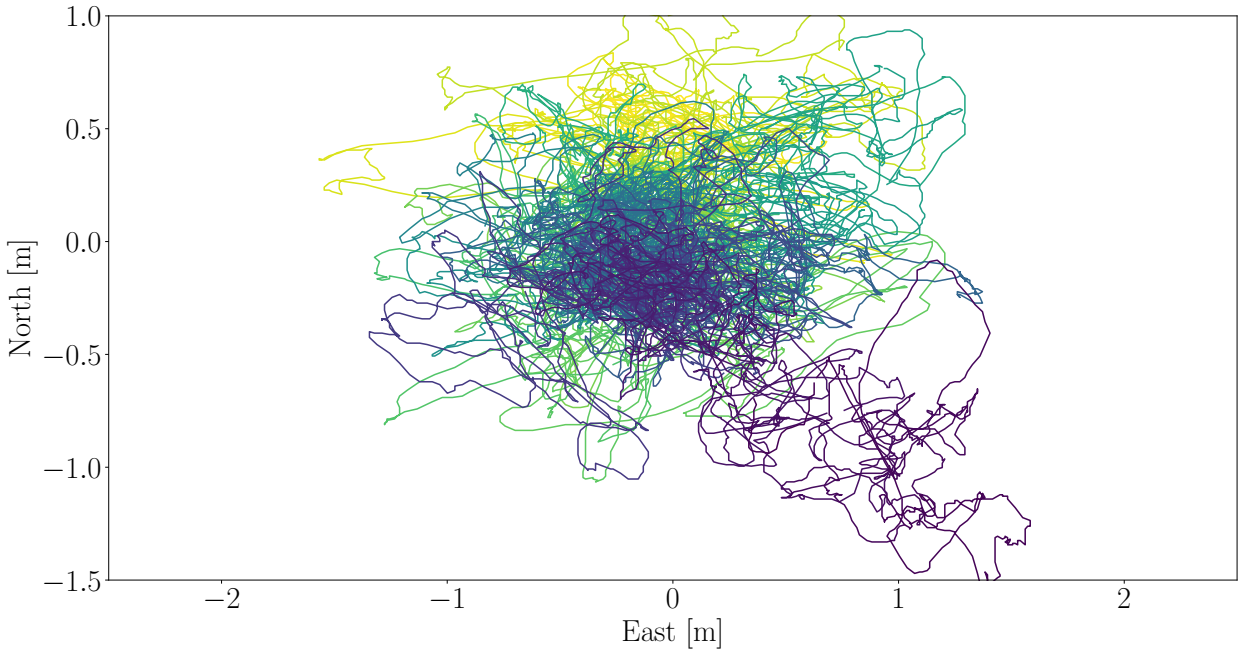

FIGURE 2.7
Position measurements from GPS visualized in a local Cartesian frame. The points are colored according to the time they were acquired, with the color gradually changing with time so that points similar in color correspond to measurements close in time. The origin is chosen to be the average of all the measurements. It is clear that measurements close to each other in time have significant correlation.

$\pm g$ has been collected, the scale factor $\mathbf{K}$ can be calculated as:

$$\begin{aligned}\hat{\mathbf{K}}_{ii} &= \frac{{}_m\ddot{\mathbf{r}}_i^+ - {}_m\ddot{\mathbf{r}}_i^-}{2|\mathbf{g}|} \\ &= \frac{\mathbf{K}_{ii}(|\mathbf{g}| + \boldsymbol{\mu}_i) - \mathbf{K}_{ii}(-|\mathbf{g}| + \boldsymbol{\mu}_i)}{2|\mathbf{g}|} \\ &= \mathbf{K}_{ii},\end{aligned} \tag{2.5}$$

where ${}_m\ddot{\mathbf{r}}_i^+$ is the i'th element of the measurement of gravity with the i'th axis anti-parallel to $\mathbf{g}$ (the i'th axis points upwards[4]) and ${}_m\mathbf{r}_i^-$ is the equivalent but with the axis parallel to $\mathbf{g}$.
When the scale factor estimates have been obtained the bias ${}^s\boldsymbol{\mu}$ can be found as:

$$\begin{aligned}\hat{\boldsymbol{\mu}}_i &= \frac{{}_m\ddot{\mathbf{r}}_i^+ + {}_m\ddot{\mathbf{r}}_i^-}{2\mathbf{K}_{ii}} \\ &= \frac{\mathbf{K}_{ii}(|\mathbf{g}| + \boldsymbol{\mu}_i) + \mathbf{K}_{ii}(-|\mathbf{g}| + \boldsymbol{\mu}_i)}{2K_{ii}} \\ &= \boldsymbol{\mu}_i,\end{aligned} \tag{2.6}$$

[4]Initially, one might think that ${}_mg_i^+$ is negative, as it points opposite the direction of gravity, but what the sensor measures in this situation is the normal force, which is pointing upwards.

These procedures are made difficult by the fact that it is somewhat difficult to determine when the axes of the sensor are perfectly aligned with gravity and what the correct values actually are here, as the measurements are easily corrupted by accelerations from sources other than gravity itself. This is evident when the sensor is held and turned around, which adds a lot of both high and low-frequency noise to the measurements.

2.2.2 Angular Rate Sensor

A static estimate of the angular rate sensor's bias is definitively the easiest of the calibration parameters to obtain. As the quantity measured is the rate of change of orientation, simply sampling the sensor while there is no change, i.e it is laying perfectly still, gives fairly reliable estimates. Similarly to the bias of the accelerometer, the angular rate sensor measurement model assumes that the bias is scaled by the scaling factor and thus the angular rate bias estimate obtained in this way is only close to the true value if the scaling factor is close to 1.

Estimating the angular rate sensor's scaling factor is however more difficult than that for the accelerometer, as no easily available non-zero reference can be used for comparison as gravity can be for the accelerometer. One possibility then is to establish a reference by for example placing the sensor on a record player. Here the sensor would then need to sample the angular velocity of the record player in both directions, $+\Omega$ and $-\Omega$. For a record player this is most easily achieved by sequencially aligning the axes of the sensor with the axis of rotation and then repeting the process upside down afterwards. When this data has been collected the scaling factor can be estimated as:

$$\begin{aligned}\hat{\mathbf{K}}_{ii} &= \frac{{}_m\boldsymbol{\omega}_i^+ - {}_m\boldsymbol{\omega}_i^-}{2|\boldsymbol{\Omega}|} \\ &= \frac{\mathbf{K}_{ii}(|\boldsymbol{\Omega}| + \boldsymbol{\mu}_i) - \mathbf{K}_{ii}(-|\boldsymbol{\Omega}| + \boldsymbol{\mu}_i))}{2|\boldsymbol{\Omega}|} \\ &= \mathbf{K}_{ii}.\end{aligned} \tag{2.7}$$

2.2.3 Magnetometer

A first approach in determining the parameters describing the relation between the true field vector and the measurements, eq. 2.4, would be to find a best fit to a collection of data points covering all or at least part of the ellipsoid that the measurements map out. This however is not readily possible as there is no simple way of calculating the distance of a point to an ellipsoid [29], let alone an expression describing the distance as a function of a general combination of ellipsoid parameters. Iterative minimization algorithms for least squares fitting do exist however [30, 53], though they have not been used or tested in this project.

Instead of the geometric distance between a point and an ellipsoid, the general ellipsoid equation below can be used in an, as termed in the literature, algebraic distance minimization:

$$(\mathbf{x} - \mathbf{v})^T \mathbf{A} (\mathbf{x} - \mathbf{v}) = 1. \tag{2.8}$$

In this work however, collections of magnetometer measurements, have qualitatively indicated that the most predominant deviations from the ideal case of a sphere at the origin has been a translation

from the origin, followed by different scaling factors along each axis, with no apparent rotation of the ellipsoid. This allows for the simpler form of the ellipsoid equation:

$$\frac{(x-c_x)^2}{r_x^2} + \frac{(x-c_y)^2}{r_y^2} + \frac{(x-c_z)^2}{r_z^2} = 1 \tag{2.9}$$

$$ax^2 + bx + cy^2 + dy + ez^2 + fz = 1, \tag{2.10}$$

where all the coefficients in the second equation and their relation to the parameters describing the ellipsoid is given in table 2.1.
Now, if all n collected data points are grouped together in a $n \times 6$ matrix $\mathbf{X}$ with each row being (x, y, z, x^2, y^2, z^2) and the coefficients from a to f are grouped together in a vector $\mathbf{c}$, we can find the coefficients by solving the equation $\mathbf{Xc} = \mathbf{1}$ which can be done by calculating the (pseudo-)inverse of $\mathbf{X}$ and summing the rows. The ellipsoid parameters can then be extracted from these coefficients according to table 2.1.

a	b	c	d	e	f	g	r_x	r_y	r_z	c_x	c_y	c_z
$\frac{1}{r_x^2 g}$	$\frac{-2c_x}{r_x^2 g}$	$\frac{1}{r_y^2 g}$	$\frac{-2c_y}{r_y^2 g}$	$\frac{1}{r_z^2 g}$	$\frac{-2c_z}{r_z^2 g}$	$1 - \frac{c_x^2}{r_x^2} - \frac{c_y^2}{r_y^2} - \frac{c_z^2}{r_z^2}$	$\sqrt{\frac{g}{a}}$	$\sqrt{\frac{g}{c}}$	$\sqrt{\frac{g}{e}}$	$-\frac{b}{2a}$	$-\frac{d}{2c}$	$-\frac{f}{2e}$

TABLE 2.1
Collection of relations between the coefficients in the ellipsoid equation 2.10 and the ellipsoid parameters. Additionally we have $g = 1 + \frac{1}{2}gb + \frac{1}{2}gd + \frac{1}{2}gf \Leftrightarrow g = 2(2 - b - d - f)^{-1}$.

For this procedure to be done on-board the autopilot, the data needs to be collected and stored first. But as will be mentioned below, the processor has a very limited amount of RAM and so one has to be careful in the data collection process not to use more than the available memory.

This method of estimating the measurement model parameters has proven to work well on datasets consisting only of two full rotations about the sensor's x- and y-axis, respectively.

2.3 Programming Microprocessors

Writing code for microprocessors, such as the STM32F745VG [11] used in this project, presents a number of challenges. Several tasks that on ordinary desktop or laptop computers would not need much thought has proven on the embedded system to demand considerable attention. A few examples of what difficulties one might encounter and will need to find solutions for are given here.

One difficulty in this project has been doing ordinary matrix calculations. The widespread matrix library Eigen that, due to different circumstances at DAS, was desirable to use over other common matrix libraries, proved to be difficult to use in combination with ChibiOS[5], the operating system running on the autopilot board. To avoid being stuck in a situation where even the simplest matrix operations were out of reach, I developed a library of my own.[6] But even with a working

[5] ChibiOS [12], which is a real time operating system, has throughout the work on the autopilot been under heavy development.

[6] Though implementing a matrix library certainly is a good exercise and a common one too actually using one's own implementations is generally disadvised as it will almost certainly lack both the performance and robustness of the common ones available. For the interested however, the library developed here, which will almost certainly run on even the simplest hardware, can be found at [13].

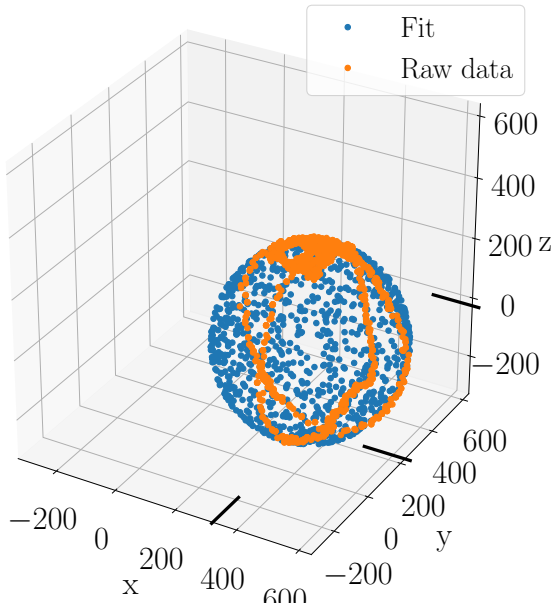

FIGURE 2.8
Static magnetometer calibration. The orange dots are measured points that map out two orthogonal ellipses. The blue dots are random samples from the ellipsoid characterized by the fit parameters. The fit has estimated the center of the ellipsoid to be at $\mathbf{r} = (337, 200, -13)$ with radii $\mathbf{k} = (286, 290, 296)$

matrix library, extra care still has to be taken, as only 240KB of memory is available for all the code and data related to the autopilot[7] on the board and thus matrices has to be kept relatively small in order not to take up all the available memory.

Another example from this project that can help describe some of the obstacles one might encounter was what seemed to be a bug in the software after switching to the newest version of ChibiOS. Here, a few seconds after start, the LEDs on the board indicating that the code was running would suddenly stop. If the board was rebooted they would again blink for a few seconds and then stop. After a long time of debugging, my colleagues and I noticed that the code would actually start running again if we put our hand close to the board, not even touching it, just close would make it continue running. The cause of the problem was in how ChibiOS handles interrupts, and so a detailed understanding of its implementation was necessary to overcome this obstacle.

A third thing that has been a factor in making real-time tests difficult is the problem of having only a relatively slow single core processor. One method that has been a great tool for analyzing the code as it was running, was to send sensor data and estimates over a serial port to a raspberry pi [14] connected to the autopilot, that would then send it over wifi to a desktop computer[8] to be visualized and analyzed. The process of sending the data over the serial port, however, takes up such a significant amount of processor power that all other code is slowed down, making the estimation algorithms run at significantly lower rates and by that changing their performance.

It should be noted however, that the processor used here is quite powerful compared to ones used in similar applications. With sufficiently well-written code, and when not performing intensive

[7]The processor is an ARM Cortex M7 with 320KB RAM in all and runs a 216MHz.
[8]The tool for sending and plotting the data was developed during the project as well and can be found at [15].

debugging the processor is able to run all the needed algorithms even with room for handling additional tasks.

2.4 Remarks

The bias and scaling factor of the sensors depend on temperature. As an example, the datasheet for the MPU9250 specifies that this dependency can be described as a linear correlation. However, as is nicely illustrated in [37], this dependency can exhibit higher order terms and even hysteresis.[9]

The noise inherent to the sensors and the distortions caused by the sensors' surroundings are the reason that the elaborate estimation methods presented in the following are necessary. A fair question to ask at this point is whether simply acquiring better sensors, which certainly do exist, would not be a better solution than to spend time developing and implementing the following estimation methods. Most of the problems would however not be solved by better sensors. Estimates might accumulate errors at a slower rate, but they would still accumulate errors. And distortions from surroundings, such as magnetic fields from electrical currents in wires, would of course not be solved by better sensors.

Another aspect to consider is that the sensors used here comes at a fraction of the price of sensors with better performance characteristics.

Finally, there is the fundamental aspect of scientific inquiry; the ability to extract accurate and robust information from data distorted by all sorts of adverse effects is both fascinating and desirable.

[9] The sensor used in the test in [37], is not entirely the same sensor as used here but a previous version of it the MPU6050.

3 Working With Position

The task of estimating the position of the platform, especially if it is to be determined in a global sense, involves several different coordinate frames and transformations between them. Before starting on the topic of how to estimate position we will go through some of the tools needed to solve this task.

The combined problem of determining both position and orientation of the vehicle relative to some reference frame is well described as the problem of determining the origin and orientation of a coordinate frame rigidly attached to the vehicle. Thus the problem of determining the position will be solved as the problem of determining the origin of a continuously translating coordinate system.

For vehicles constrained to move within a small space, for example a UAV flying indoors, or a UAV maneuvered by hand in the user's field of view, it might be satisfactory to only keep track of where the UAV is at at time t relative to where it started out. If, on the other hand, what is needed is the UAV's global position, i.e. where on the Earth it is exactly, the task becomes a bit more complicated as will be detailed here.

3.1 Coordinate Frames

Following along the lines of [26] we start with a short list of the different coordinate systems used in the estimation of position as well as in the estimation of orientation of the vehicle later on. Each coordinate system is given an abbreviation, which quantities represented in this coordinate system will have as their leading superscript.

- B - **B**ody fixed frame. Its origin is at the UAV's center of mass and its axes are aligned with the inertial principal axes (the set of orthogonal axes for which the inertia matrix is diagonal) of the UAV's body.
- VN - **V**ehicle carried north, east, down (**N**ED) frame. This frame also has its origin at the center of mass of the vehicle, but has its x-, y- and z-axes aligned with north, east and down, respectively. This is the frame generally used as reference when estimating the orientation of the vehicle as this frame and frame B are related by a rotation only.
- E - **E**arth fixed frame, with origin where the UAV starts and axes following the NED convention.
- WGS84 - Earth centered, Earth fixed ellipsoidal coordinate system. The abbreviation is from **W**orld **G**eodetic **S**ystem revision **84**. Its origin is at the center of mass of the Earth and

coordinates given as latitude ϕ, degrees from the equatorial plane, longitude λ, degrees from IERS Reference Meridian[1] and height h, distance above the ellipsoid surface[2].

- ECEF - **E**arth **c**entered **E**arth **f**ixed cartesian frame. With its origin at the center of mass of the earth, z-axis along the Earth's spin axis (towards the north pole), x-axis intersecting the reference meridian and y-axis such that the system is right handed.
- S_i - Frames fixed in each **s**ensor with directional measurements (such as the accelerometer), with origin and axes aligned with the sensor's internal axes.

It will throughout the rest of the thesis, for the sake of simplicity, be assumed that all the S_i frames coincide with the body fixed frame B. Thus, all measurements will be given in frame B, even though there might be some constant non-identity transformations on an actual platform relating frame S_i and B.

In the local position estimation problem we want to know the position of the vehicle with respect to the E frame, that is, we want to know where it is relative to where it took off. This is given by the quantity ${}^{E}\mathbf{r}$. However, measurements from the accelerometer is given in the body frame ${}^{B}_{m}\ddot{\mathbf{r}}$ and measurements from the GPS is given in the WGS84 coordinate system ${}^{WGS84}_{m}\mathbf{r}$, so we need to know how to transform these measured quantities into the E frame. Alternatively we might be more interested in determining the vehicles global position directly, described by ${}^{WGS84}\mathbf{r}$ for which we need to know how to transform from the B frame and VN into the WGS84 frame.

The estimation algorithm in the next chapter is build on the first idea, of estimating the position in the E frame, so most of the following will be geared towards this goal.

To simplify things a bit more, the problem of estimating position will be handled completely separately from the problem of estimating orientation, so in this chapter and the next it will be assumed that the orientation of the vehicle is known, so that quantities can be transformed between the body frame and the Earth frame. That is, the rotation matrix ${}^{B}\mathbf{R}^{E}(t)$, for which ${}^{B}\mathbf{v}(t) = {}^{B}\mathbf{R}^{E}(t){}^{E}\mathbf{v}(t)$, will be given. The problem of estimating this rotation is handled in chapters 5 and 6.

Difference between VN and E

The VN and E frames will be almost identical in orientation when at short distances from each other, but as the distance between the two is increased, the difference starts to become non-negligible. The difference can be illustrated by placing two NED coordinate frames at ${}^{WGS84}(0, 0, 0)$, the intersection between the meridian and equatorial plane at zero altitude, then moving one of them along the equator to the opposite side of the Earth. As both frames are NED frames, they will both have their y-axes pointing eastwards, but these will now be pointing in opposite directions relative to each other.

Taking this into account is important if the vehicle is to move over considerable distances, but in the current position estimation implementation it is assumed that the two coordinate systems are aligned. For the same reason, quantities that in practice are given in the VN frame will instead be written as given in the E frame.

[1] The International Earth Rotation and Reference Systems Service (IERS) reference meridian passes ∼100 meters east of the Royal Observatory, Greenwich [33].

[2] The latitude and longitude describes the direction of the normal at a point on the reference ellipsoid. The height is along the normal.

3.1.1 Transformations

Conversion between WGS84 and ECEF

The ECEF frame is used as an intermediate frame when relating coordinates in an NED frame to global WGS84 coordinates. The relation is [26]

$$\begin{aligned} {}^{ECEF}x &= (N(\phi)+h)\cos(\phi)\cos(\lambda) \\ {}^{ECEF}y &= (N(\phi)+h)\cos(\phi)\sin(\lambda) \\ {}^{ECEF}z &= (N(\phi)(1-e^2)+h)\sin(\phi) \end{aligned} \tag{3.1}$$

where $N(\phi) = R_{EA}/\sqrt{1-e^2\sin^2(\phi)}$, with R_{EA} being the Earth semi-major axis and e being the eccentricity (a measure of how much the Earth deviates from being spherical).
An implementation of the considerably more complicated conversion from ECEF to WGS84 can be found in [50].

Global coordinates in NED frame

Finding the coordinates of a point given in WGS84, ${}^{WGS84}\mathbf{x}$, in an NED frame with origin at ${}^{WGS84}\mathbf{o}$ can be solved in two steps:
First we find the vector from the origin of the NED frame to the point. This is done in the ECEF frame:

$${}^{ECEF}\mathbf{d} = {}^{ECEF}\mathbf{x} - {}^{ECEF}\mathbf{o} \tag{3.2}$$

where the ECEF coordinates of the point and the NED frame origin can be found using the equations 3.1.
Next we need to rotate this vector into the NED frame:
An NED frame located at $\phi = 0,\ \lambda = 0$ is orientated similarly to the ECEF frame, except for a rotation of $\frac{\pi}{2}$ around their parallel y-axes. To find the orientation of an NED frame relative to the ECEF frame at any coordinate pair (ϕ, λ)[3], we note that rotating the NED frame located at $(0, 0)$ around its x-axis i.e. its north-axis (the $ECEF$ z-axis) by an angle of λ followed by a rotation around its y-axis by ϕ, brings it to the configuration that we need. The composition of these three consecutive rotations become:

$${}^{NED}\mathbf{R}^{ECEF} = \mathbf{R}_y(\phi)\mathbf{R}_x(\lambda)\mathbf{R}_y(\pi/2) \tag{3.3}$$

$$= \begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\phi)\sin(\lambda) & \cos(\phi) \\ -\sin(\lambda) & \cos(\lambda) & 0 \\ -\cos(\phi)\cos(\lambda) & -\cos(\phi)\sin(\lambda) & -\sin(\phi) \end{bmatrix} \tag{3.4}$$

And thus, if the NED frame of interest is the E frame, we have that the coordinates of a point ${}^{WGS84}\mathbf{x}$ in this frame is

$${}^{E}\mathbf{x} = {}^{E}\mathbf{R}^{ECEF} \cdot {}^{ECEF}\mathbf{d} \tag{3.5}$$

[3]The height coordinate is not involved in the orientation of the E as it represents a translation of a NED frame origin along the ellipsoid normal at (ϕ, λ).

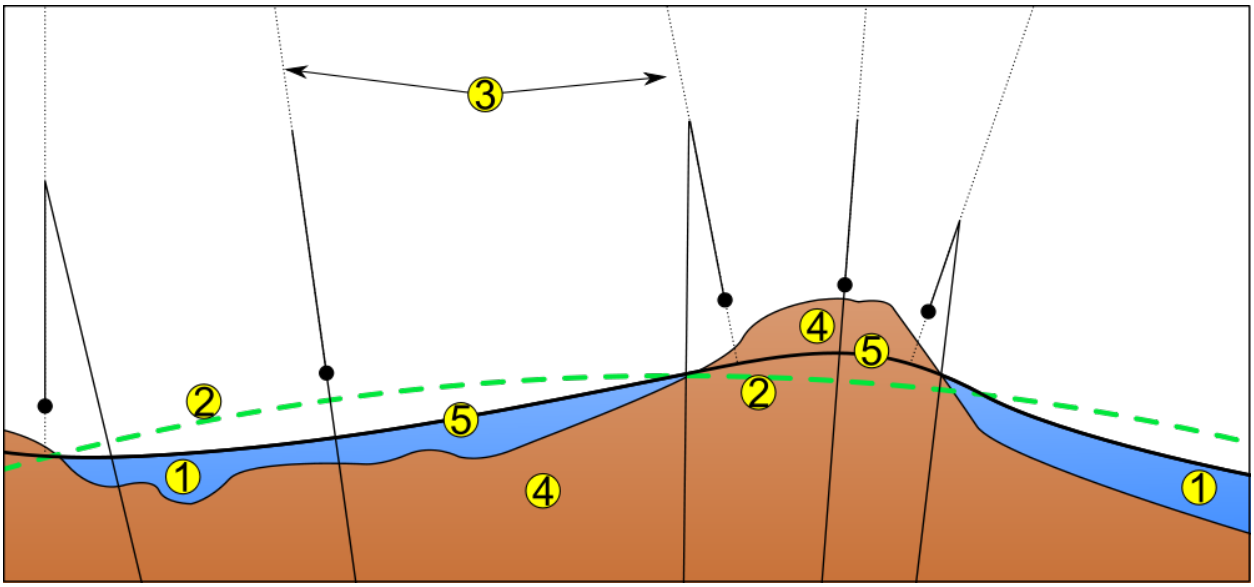

FIGURE 3.1
Illustration of the water and landmass distribution relative to the reference ellipsoid [16]. 1. Ocean, 2. Reference ellipsoid, 3. Local plumb line, 4. Continent, 5. Geoid.

3.2 Height and Altitude

The notion of the height of a point is a little ambiguous. Usually it is implicitly assumed to be the vertical distance from the point of interest to some reference point that will depend on the context. Often, in a more general setting, height is thought of as being above sea level or mean sea level, but this poses a problem when far from the ocean, as it is not simple to determine the sea level at some point hundreds or thousands of kilometers from the coast. This, however, is what is described by the geoid. It is a mathematical model of an equipotential, i.e where the surface of the oceans would be if they could reach any point while still being influenced by the mass distribution at that point [47]. As mentioned above, the WGS84 coordinates gives height with reference to the ellipsoid surface, but this might be misleading as the height difference between the geoid and reference ellipsoid vary from -105m to 80m [46].
As shown in figure 3.1, the direction of gravity is perpendicular to the equipotential surface, which means that it is not perpendicular to the reference ellipsoid, and thus introduces errors when using the direction of gravity to estimate orientation. This difference in direction is not taken into account in any of the following and should only contribute negligibly to the errors already accumulating from other sources[4].

Altitude is by [34] defined to be *"height where the chosen reference surface is mean sea level"*. Whereas height is defined as *"distance of a point from a chosen reference surface measured upward along a line perpendicular to that surface"*.

[4] I have not been able to find a source that states the order of magnitude for this deviation.

4 Position Estimation

The goal for the position estimation algorithm is to make use of the accelerometer, GPS, barometer and laser rangefinder's different advantages and use the redundancy in information about translational dynamics provided by these to correct for their individual shortcomings.

4.1 Position from acceleration

As mentioned earlier, in the case of perfect information about acceleration, the problem of estimating the position of the setup could be solved solely by double integration of a continuously measured acceleration combined with a known starting position. However, for sensors similar to the ones used here, the estimated position quickly builds up a large error. An incorrectly estimated bias ruins the position estimate right away as can be seen in figure 4.3, but zero mean Gaussian noise alone will also over time become a problem as described below.

If we assume the measured acceleration to be related to the true acceleration according to the measurement model in equation 2.2, and if the accelerometer is placed at the center of mass, so that the measured accelerations are related only to translational motion, then the position can be estimated as

$$\begin{aligned} {}^{E}\hat{\mathbf{r}}(t) &= \int_0^t \int_0^t {}^{E}\ddot{\mathbf{r}}(t')dt' \\ &= \int_0^t \int_0^t {}^{E}\mathbf{R}^{B}(t')\mathbf{K}^{-1}(t') \left({}^{B}_{m}\ddot{\mathbf{r}}(t') - {}^{B}\boldsymbol{\mu}(t') - {}^{B}\boldsymbol{\xi}(t') \right) - {}^{E}\mathbf{g}\ dt' \end{aligned} \tag{4.1}$$

This means in order to calculate the position exactly we need to know the scaling factor $\mathbf{K}^{-1}(t)$, the bias $\boldsymbol{\mu}(t)$, the noise $\boldsymbol{\xi}(t)$ and orientation ${}^{E}\mathbf{R}^{B}(t)$.
All this knowledge is not available. The sensor only provides discrete samples, and some of the sensor measurement characteristics are not readily available. In practice, the position estimate from acceleration measurements can be calculated as

$$\begin{aligned} {}^{E}\hat{\mathbf{r}}_i &= \sum_{i=1}^{N} {}^{E}\hat{\dot{\mathbf{r}}}_i \Delta\hat{t}_i \\ {}^{E}\hat{\dot{\mathbf{r}}}_i &= \sum_{i=1}^{N} \left({}^{E}\hat{\mathbf{R}}_i^{B}\hat{\mathbf{K}}^{-1} \left({}^{B}_{m}\ddot{\mathbf{r}}_i - \hat{\boldsymbol{\mu}}_i \right) - {}^{E}\mathbf{g} \right) \Delta\hat{t}_i \end{aligned} \tag{4.2}$$

where the subscript i refers to the value of the quantity at the time the i'th calculation was done[1]. Thus $\Delta\hat{t}_i = {}_m t_i - {}_m t_{i-1}$ is the time between the i'th and $i-1$'th calculation as measured by the microprocessor[2]. N is the number of estimates between $t = 0$ to $t = t_N$.
The simplifications in this implementation are:

- The normally distributed noise term has been left out. The MPU9250 sensor has a build-in low-pass filter, which helps minimize the noise term. The low-pass filter can not completely remove the noise, so the position estimate will still build up some error over time. Figure 4.1 visualizes the effect of having non-zero noise in the integrated values. The noise term should not change the expectation value of the estimate but, only increase its uncertainty as time progresses. This is confirmed by figure 4.1 where the distributions of position estimates along each axis is centered at zero.
- The scale matrix is assumed to be constant. For a quantitative answer to whether this simplification is reasonable, a more in depth analysis than has been done in this project is needed.

The estimation of the bias term $\hat{\boldsymbol{\mu}}_i$ will be handled in the following section.

This method of integrating the acceleration measurements works well for short periods of time. For unbiased but relatively noisy measurements the estimated position and velocity comes very close to the true values as can be seen from figure 4.2, where actual measurements has been integrated. The same estimate without bias correction can be seen in figure 4.3 where it is clear that the estimate quickly diverges. The effect of the zero-mean noise is however not completely neglectable. As visualized in figure 4.1, the estimated position is about 1.5 m, on average, from the true position after one minute with no actual movement.

4.1.1 Accelerometer not at center of mass

In the position estimate given above it is assumed that the accelerometer is placed at the center of mass. This, however, might be far from true in practice,[3] which will lead to measurements differing from the one assumed above.

Taking the time derivative of a vector-valued function in a rotating reference frame gives an additional term related to the changing axis unit vectors of the frame. Applying the differentiation twice to the position vector of a particle, as seen in the rotating frame, gives an equation for the acceleration on the form

$$\ddot{\mathbf{r}}_r = \ddot{\mathbf{r}}_i - 2\boldsymbol{\omega} \times \dot{\mathbf{r}}_r - \boldsymbol{\omega} \times (\boldsymbol{\omega} \times \mathbf{r}) - \dot{\boldsymbol{\omega}} \times \mathbf{r} \tag{4.3}$$

where $\ddot{\mathbf{r}}_i$ is the acceleration as seen from an inertial frame, $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ are the angular velocity and angular acceleration of the rotating frame relative to the fixed one, and $\dot{\mathbf{r}}_r$ is the velocity of the

[1] This is a bit of an idealization as some delay will be introduced from the motion occured to the measurement of it is used in the calculations.

[2] The microprocessor increments an integer every n clock cycles, where n clock cycles is intended to take 1 µs. As the time of 1 clock cycle can vary over time (for example due to change in temperature), the time, as estimated by the microprocessor, will vary from the true time as well.

[3] In some of the quadrotor and fixed-wing plane designs at DAS the planned motor controller(s) has an additional accelerometer built in. These are placed at distances up to ~ 0.5 m from the center of mass.

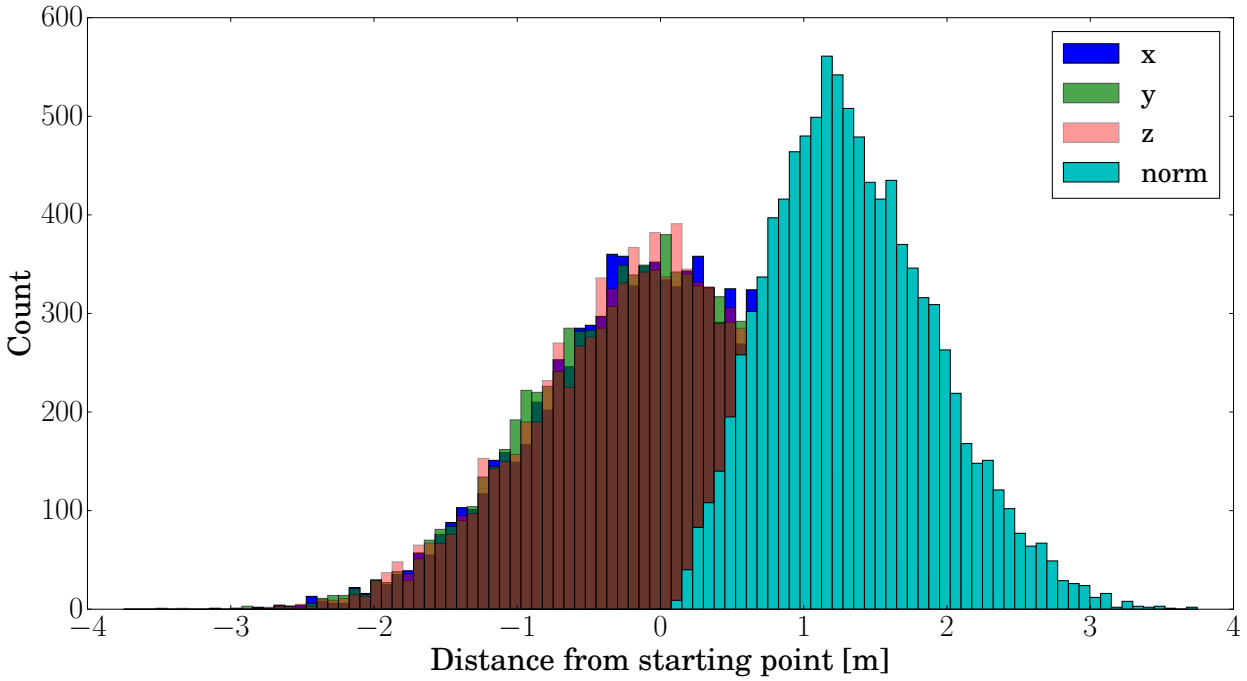

FIGURE 4.1
Distribution of estimated position from double integrated discrete acceleration values. Each value has been drawn from a $\mu = 0$, $\sigma = 0.1$ normal distribution. Each dimension is shown separately, as well as the distribution of the euclidean distance from the starting point. The plot contains 10^4 trials simulating 60 seconds each.

particle as seen from the rotating frame. As we are interested in a sensor fixed in the rotating frame, the velocity vector $\dot{\mathbf{r}}_r$ will be zero, and the position vector will be constant $\mathbf{r} = \mathbf{r}_s$, and instead we have, with time dependence explicitly included

$$\ddot{\mathbf{r}}_r(t) = \ddot{\mathbf{r}}_i(t) - \boldsymbol{\omega}(t) \times (\boldsymbol{\omega}(t) \times \mathbf{r}_s) - \dot{\boldsymbol{\omega}}(t) \times \mathbf{r}_s \tag{4.4}$$

The first additional term is the centripetal acceleration, and the second term is the linear acceleration a mass element of a rigid body undergoing angular acceleration will experience.
These additional terms originating from the rotation of the vehicle can be seen as distorting the ideal measurements, or alternatively as additional information that is usable in other parts of the full state estimation task. For example if the value of $\mathbf{r}_s$ is known, the measured accelerations can be used for improving the estimates of angular velocity. Alternatively, measurements of the angular velocity from the angular rate sensor might be used to estimate $\mathbf{r}_s$. Knowing $\mathbf{r}_s$ allows acceleration measurements from other places than at the center of mass to be corrected for the effects of rotation after which they can be used in the position estimation task.
In the current implementation these additional terms are not handled, though plans of using several accelerometers distributed over the body of the vehicle has influenced the software development process. Work on using these rotation related acceleration terms can be found in [37].

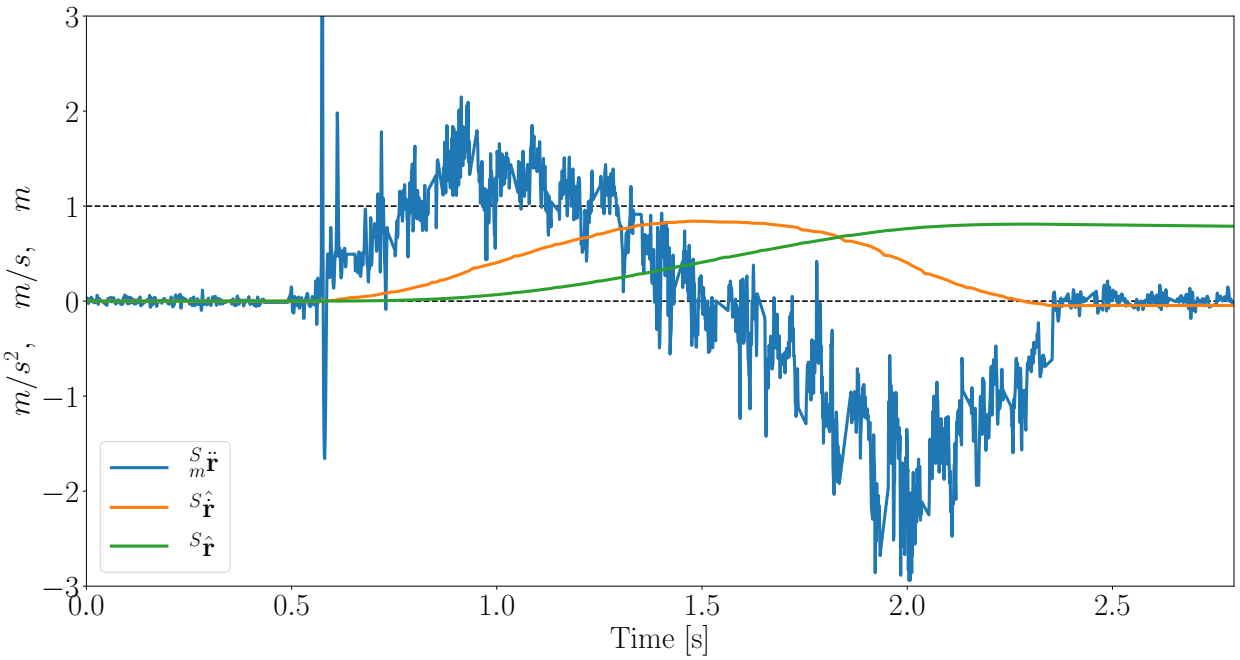

FIGURE 4.2

Position and velocity estimated from acceleration measurements. The samples were collected while the sensor was slided roughly 1 m along a plane surface in the direction parallel to the x-axis of the sensor. As is correctly estimated, the sensor had zero velocity at the start and at the end of the movement. The first 0.5 seconds of the data has been used to estimate the bias of the measurements. This bias has been subtracted from all data points before the estimations were carried out.

4.2 Filtering

To correct for the accumulation of errors over time, the acceleration based estimates need to be combined with other types of data. As mentioned in chapter 2, the platform used in this project is equipped with a GPS, providing measurements of position in 3 dimensions, and a barometer, and a laser rangefinder providing 1-dimensional measurements. Other common sources for position information may include ultrasound distance sensors, cameras mounted on the vehicle, external cameras or other local positioning systems. All these sensors and the laser rangefinder provide position information relative to the immediate surroundings, whereas the GPS and barometer provide global information.

With information about the position of the vehicle obtained from different sources we are faced with the problem of combining them all into a single improved estimate. This is generally called a filtering problem. Filtering problems can be grouped into linear and non-inear problems. There are well established practical methods of solving the linear problems. As described in [52], the general non-linear problem can be stated as

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{w}) \tag{4.5}$$

$$\mathbf{y} = h(\mathbf{x}, \mathbf{v}) \tag{4.6}$$

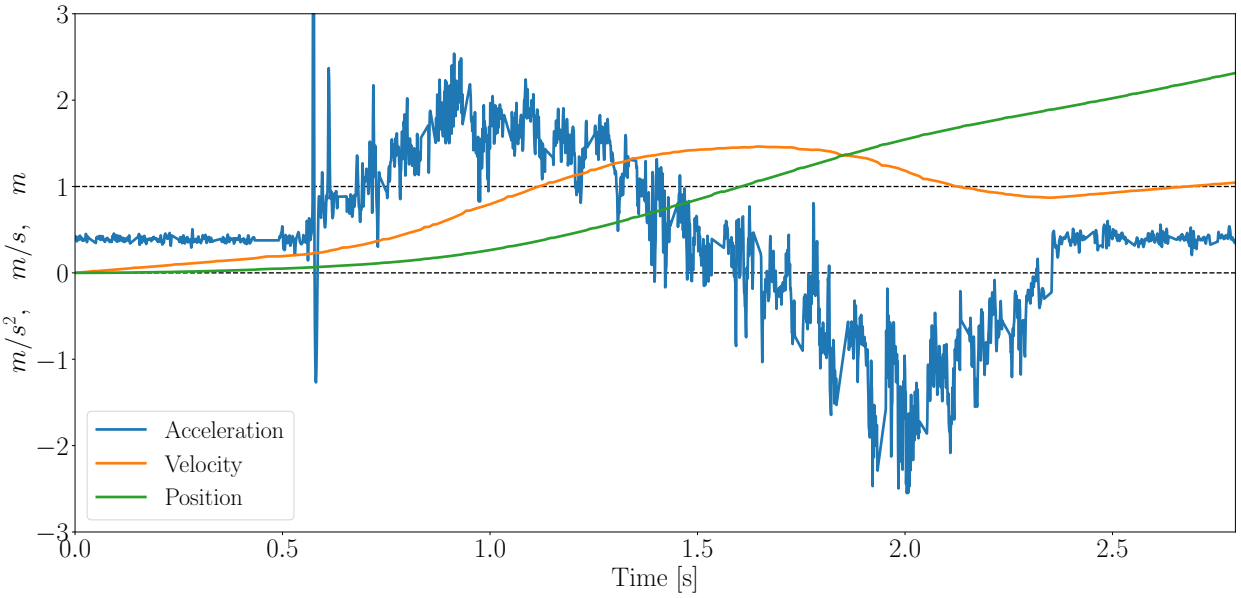

FIGURE 4.3
Position and velocity estimated from acceleration measurements. The samples were collected while the sensor was slid roughly 1 m along a plane surface in the direction parallel to the x-axis of the sensor. The sensor was lying still before and after it was slided. The considerable bias in the measurements leads to both velocity and position estimates quickly diverging from the true values.

where $\mathbf{x}$ is the state of the system: a set of variables capturing the status of the system. $\mathbf{u}$ is the input to the system and is in general the variable of the system that we have control over. $\mathbf{w}$ is white noise with covariance $\mathbf{Q}$ that is inherent to the dynamics of the system. $\mathbf{y}$ is how we observe the system. $\mathbf{v}$ is white noise related to the observations with covariance $\mathbf{R}$. For linear systems this simplifies to

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} + \mathbf{w} \tag{4.7}$$

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{v} \tag{4.8}$$

4.2.1 System model

For the accelerometer based position estimation above, in a situation where additional information about position is available through GPS measurements, and with no input to the system, an estimate using both the accelerometer and the GPS could be formulated as

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \\ \dddot{\mathbf{r}} \end{bmatrix} \qquad \mathbf{A} = \begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{r} \\ \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

$$\mathbf{y} = \begin{bmatrix} \mathbf{r} \\ \ddot{\mathbf{r}} \end{bmatrix}$$

Note that the state vector here has 9 elements and $\mathbf{A}$ is thus a 9×9 matrix.
An alternative to this model, as we here only model changes in acceleration through the noise term $\mathbf{w}$, a simpler formulation or at least a formulation with a smaller state vector (which in some situations is preferable) could look like

$$\begin{aligned} \dot{\mathbf{x}} &= \begin{bmatrix} \dot{\mathbf{r}} \\ \ddot{\mathbf{r}} \end{bmatrix} & \mathbf{A} &= \begin{bmatrix} \mathbf{0} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \\ \mathbf{x} &= \begin{bmatrix} \mathbf{r} \\ \dot{\mathbf{r}} \end{bmatrix} & \mathbf{B} &= \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} \\ \mathbf{u} &= \ddot{\mathbf{r}} & \mathbf{C} &= \begin{bmatrix} \mathbf{1} & \mathbf{0} \end{bmatrix} \\ \mathbf{y} &= \mathbf{r} \end{aligned} \tag{4.9}$$

In this model our state is position and velocity, which is what we want to estimate. By using the accelerometer measurements as input $\mathbf{u}$, the prediction step can be performed at the same rate as they are available. The update step can then done when GPS measurements become available, which is at a much slower rate. A slightly modified version of this model is used in the toy example in section 4.2.4.

There is generally quite a bit of freedom in choosing a system model, though there is some limit to what can be estimated from the available observations. This relation is described by a system's observability [52].

A simple addition to the model above for multirotors that has proven effective in improving the position estimate[4] for "not-too-aggressive" flight,[5] takes the tilt of the vehicle into account. If we for a moment assume the thrust vector from the propellers of a steadily hovering UAV to be constant, then a tilt of the UAV in the direction $\hat{\mathbf{e}}$ (where $\hat{\mathbf{e}}$ is in the horizontal xy-plane) by an angle θ will result in an acceleration of

$${}^{E}\ddot{\mathbf{r}} = |\mathbf{g}| \begin{bmatrix} \sin(\theta)\hat{\mathbf{e}} \cdot \hat{\mathbf{x}}_E \\ \sin(\theta)\hat{\mathbf{e}} \cdot \hat{\mathbf{y}}_E \\ -\cos(\theta) \end{bmatrix} \tag{4.10}$$

where $\hat{\mathbf{i}}_E$ is the unit vector along the i'th axis of the Earth frame. This addition has not yet been included in the current implementation.

[4]The idea is based on a conversation with Benjamin Vedder [17]1 where he described some work on multirotors he had done [18].

[5]Situations where the multirotor is close to horizontal and the thrust from its propellers are close to $m\mathbf{g}$.

4.2.2 The Kalman Filter

In this thesis the Kalman filter has been the filter of choice. For this method we first rewrite the general linear equations as a sampled-data system [52], which is a discretization of an otherwise continuous time system:

$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{F}_{k-1}\mathbf{x}_{k-1} + \mathbf{G}_{k-1}\mathbf{u}_{k-1} + \mathbf{w}_{k-1} \tag{4.11}$$

$$\mathbf{y}_k = \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \tag{4.12}$$

where $\mathbf{F}_k = \exp(\mathbf{A}\Delta t)$, $\mathbf{G}_k = \int_{t_{k-1}}^{t_k} \exp(\mathbf{A}(t_k - \tau))\mathbf{B}_k(\tau)d\tau$, $\mathbf{H}_k = \mathbf{C}$, $\Delta t = t_k - t_{k-1}$. Furthermore the covariance for $\mathbf{w}_{k-1}$ is approximated as $\mathbf{Q}_k \approx \mathbf{Q}(t_k)\Delta t$.

The Kalman filter uses this discretized dynamics model combined with an iterative least squares method to estimate the state $\mathbf{x}$ as well as the covariance $\mathbf{P}$ of the state estimate. The algorithm has two steps: a prediction step, where the system dynamics model is used to propagate the state and covariance estimate one time step ahead, followed by an update step, where measurements are used to correct or update the state and covariance estimates based on information provided by the observations.
Sticking with the notation in [52], the prediction and update step respectively are

$$\hat{\mathbf{x}}_k^- = \mathbf{F}_{k-1}\hat{\mathbf{x}}_{k-1}^+ + \mathbf{G}_{k-1}\mathbf{u}_{k-1} \tag{4.13}$$

$$\mathbf{P}_k^- = \mathbf{F}_{k-1}\mathbf{P}_{k-1}^+\mathbf{F}_{k-1}^T + \mathbf{Q}_{k-1} \tag{4.14}$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \tag{4.15}$$

$$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k(\mathbf{y}_k - \mathbf{H}_k\hat{\mathbf{x}}_k^-) \tag{4.16}$$

$$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\mathbf{P}_k^-(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)^T + \mathbf{K}_k\mathbf{R}_k\mathbf{K}_k^T \tag{4.17}$$

The matrix $\mathbf{K}$ is called the Kalman gain. The Kalman filter is unbiased for any value of the Kalman gain, but is chosen such that the uncertainty of the estimate is minimized [52]. Here, we also see why we are interested in having only a small state vector: Because the update step involves a matrix inversion we are interested in keeping a small state vector, as it will otherwise become too computationally expensive to perform on our microprocessor.[6]

4.2.3 Current Implementation

Going back to explicitly labeling all quantities by the coordinate frames they are given in, the state, input and observation of the current implementation[7] of the Kalman filter are

[6] The matrix inversion can be avoided if specific assumptions about correlations of the noise can be made [52].

[7] The specific model of a Kalman filter given here has been based on work done by others previously working at DAS.

$$\mathbf{x} = \begin{bmatrix} {}^{E}\mathbf{r} \\ {}^{E}\dot{\mathbf{r}} \\ \mu_{bar} \\ \boldsymbol{\mu}_{acc} \end{bmatrix} \qquad \mathbf{u} = \begin{bmatrix} {}^{B}\ddot{\mathbf{r}} \\ {}^{E}g \end{bmatrix} \qquad \mathbf{y} = \begin{bmatrix} {}^{E}\mathbf{r} \\ {}^{E}\dot{\mathbf{r}} \\ \tilde{z}(\mathbf{r}) \end{bmatrix}$$

The last term in the observation vector is the altitude or height above sea level provided by a barometer. This term is affected by local pressure variations and might therefore deviate from the altitude provided by the GPS. A bias μ_{bar} is added to the system model to account for this possible difference. The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{F}, \mathbf{G}, \mathbf{R}, \mathbf{Q}$ can be found in appendix B written out element by element.

As the orientation estimation problem, described in the following chapters, took precedence, this implementation still needs additional testing. Qualitative test on live data has indicated that the implementation behaves as intended, but no comparable test to ground truth data has been performed yet.

4.2.4 Toy Example

The full position estimation part of the autopilot still needs additional testing, thus no visualization of its application is included. However, a small 1-dimensional toy example is included here in order to demonstrate the effects of how the Kalman filter uses the system dynamics model and observations.

Using the simulation framework, described in chapter 7, a dataset was generated. This dataset is used both for generating observations and as a ground truth for comparison. The dataset was generated by simulating a UAV starting at position ${}^{E}\mathbf{r} = (0, 0, 0)$ with a desired position ${}^{E}\mathbf{r}_D = (0, 0, 1)$. As the UAV flies towards the desired position the distance between the UAV and the desired position at some point goes below some predefined limit and the desired position is changed to ${}^{E}\mathbf{r}_D = (0, 0, 0)$. This causes the UAV to start slowly descending towards the lower position. When the UAV is again sufficiently close to the low position the desired position is moved up again. This is repeated a few times. The simulated behavior resulting from this is illustrated in figure 4.4.

For the Kalman filter a model similar to that in equations 4.9 with accelerometer bias added as a state variable and velocity added as an observation is used. To illustrate the filter's ability to estimate accelerometer bias a large bias of $-1\,\mathrm{m/s^2}$ was added to the true acceleration along with normally distributed noise. Simulated noisy position and velocity measurements[8] were used at 1/100 the rate of the accelerometer measurements, and thus the measurements are simulated to be available at the same rate as from the actual sensor on the platform at 1000 Hz and 10 Hz respectively. The estimated position, velocity and accelerometer bias are visualized in figure 4.5. Here we see that, for the first part of the simulation, the many prediction steps continuously result in velocity estimates towards larger and larger negative values. This is due to the accelerometer bias estimate not having converged to the true value yet. Each time a new velocity measurement is available, the erroneously estimated velocity is corrected towards the measurement. As better and better estimates of the bias are obtained, both the position and velocity estimates in between observations improve considerably.

[8] The noise added to the true position, velocity and acceleration is not chosen in order to mimic the true values but only to demonstrate their effect.

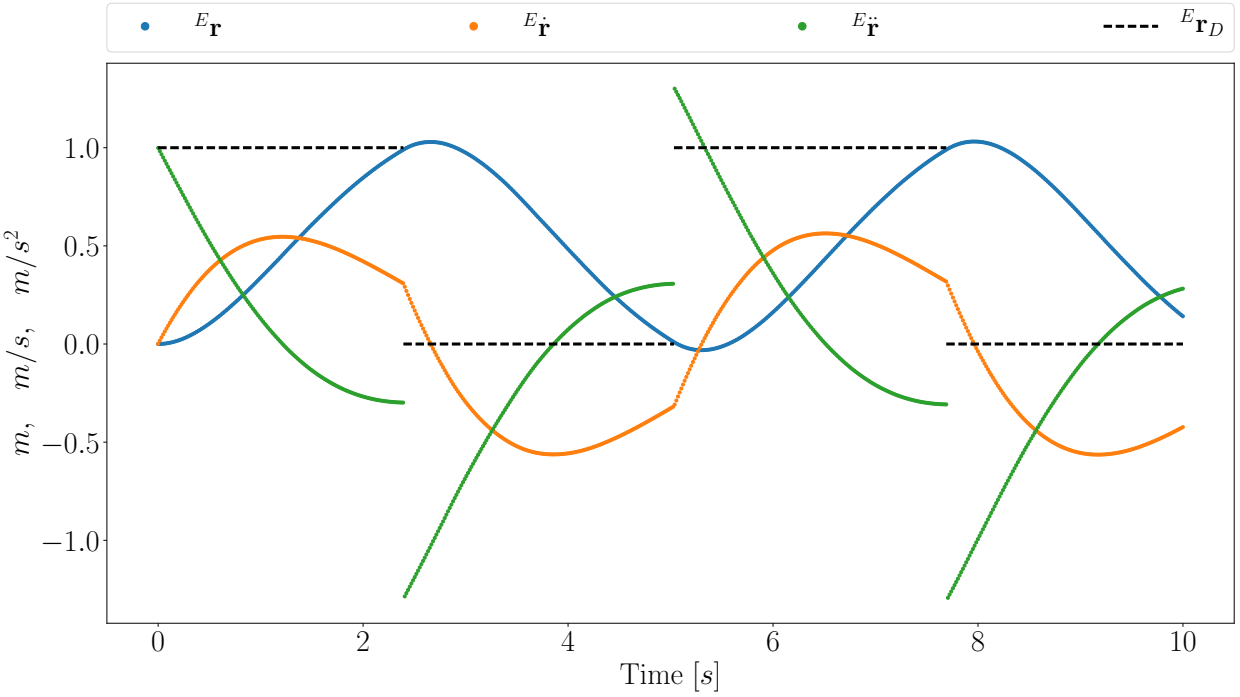

FIGURE 4.4
Position, velocity and acceleration of a simulated UAV flying between two waypoints in 1 dimension. The UAV is controlled using the controller described in chapter 8. No noise is included in the simulation.

4.2.5 Non-linear Formulations

As mentioned in the previous chapter, a filter could instead be designed to estimate the position of the vehicle in the WGS84 frame. This however involves non-linear models for relating the vehicle's velocity expressed in the VN frame to the derivatives of position in the WGS84 frame [26]. The ordinary Kalman filter is inadequate for this problem. However, several filters exist for handling non-linear filter problems with common examples being, the Extended Kalman filter, the Unscented Kalman filter, and the particle filter [52]. A few experiments with these filters have been carried out during this project, but were not, in the preliminary tests, designed towards estimating the state of a UAV and will not be discussed further.

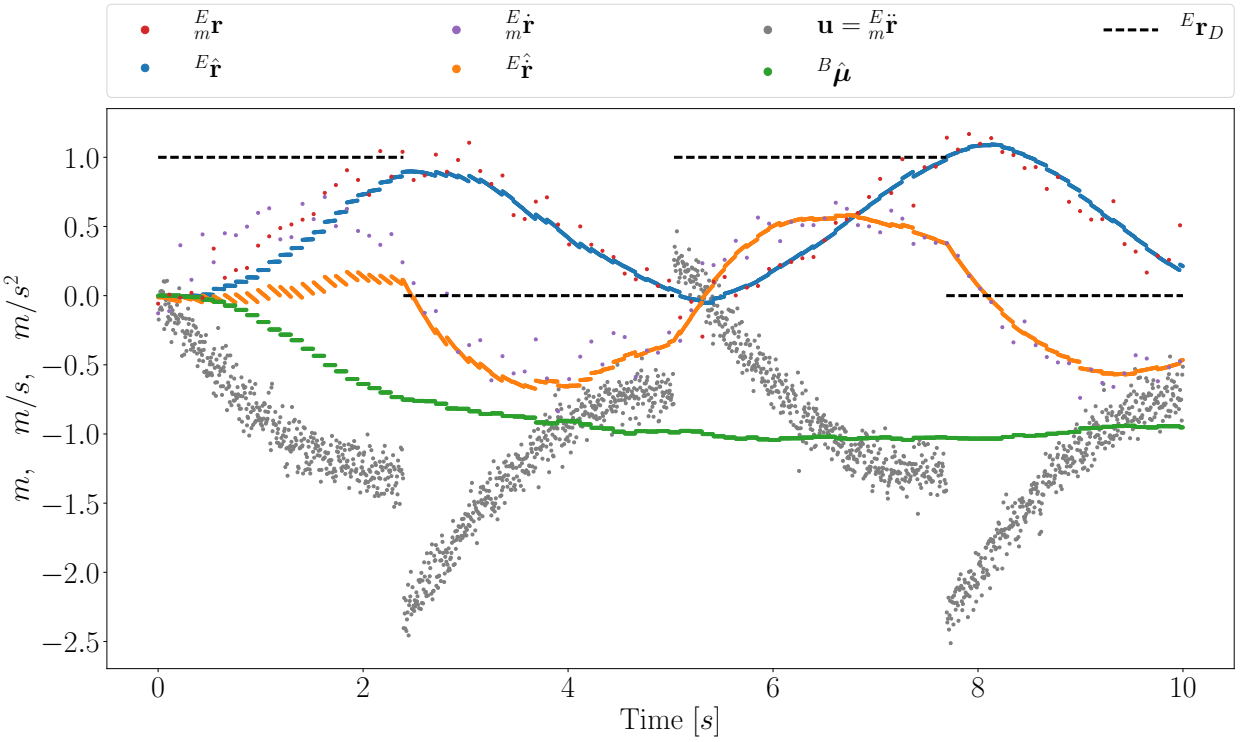

FIGURE 4.5
Position, velocity and accelerometer bias as estimated by the Kalman filter. The true values of position and velocity can be seen in figure 4.4. The noisy and sparse dots are measurements of position, velocity and acceleration.

5 Working With Orientation

Several useful quantities for representing and working with orientations exist. The methods for estimating orientation described in chapter 6 rely on unit length quaternions, the control algorithm in chapter 8 is based on Euler angles and the Kalman filter in the previous chapter used rotation matrices. Before we start using quaternions and Euler angles in the following chapters, a brief description is given here, detailing how they are used to represent orientations, and how their derivatives are related to the angular velocity.

5.1 Representations of Orientation

The task of representing the orientation of a rigid object can adequately be described as the orientation of a coordinate frame fixed on the object relative to some fixed reference frame. Here we will focus on coordinate frames with common origin and thus frames only differing in orientation.

We will start by mentioning an ambiguity that can lead to confusion[1] namely what a rotation matrix actually represents. Does it represent how the axes of one coordinate system are described with respect to another coordinate system (a passive rotation) as visualized in figure 5.1, or does it describe how the coordinates of a vector changes when that vector is rotated (an active rotation) as visualized in figure 5.2?
If we think of the depiction in figure 5.1 as two coordinate systems initially coinciding before rotating the primed system by an angle θ, then the coordinates of the vector $\mathbf{r}$ in the primed system will be:

$$\begin{aligned} \mathbf{r}' &= \mathbf{R}_p\mathbf{r} \\ \begin{bmatrix} r'_x \\ r'_y \end{bmatrix} &= \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}_p \begin{bmatrix} r_x \\ r_y \end{bmatrix} \end{aligned} \tag{5.1}$$

However, if we look at it as an active rotation[2] as in figure 5.2, still having positive angles being counter-clockwise, then the primed coordinates will be given as:

$$\begin{aligned} \mathbf{r}' &= \mathbf{R}_a\mathbf{r} \\ \begin{bmatrix} r'_x \\ r'_y \end{bmatrix} &= \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}_a \begin{bmatrix} r_x \\ r_y \end{bmatrix} \end{aligned} \tag{5.2}$$

Equations 5.1 and 5.2 both yield the same result, as is also clear from the two illustrations that they should, but the general rotation matrix and the angle is defined differently.

[1] I personally found it to be very helpful to have matters cleared up, in any case.

[2] The term active rotation as opposed the a passive rotation is used in [31]. Others [55] refer to them as alibi and alias transformation respectively.

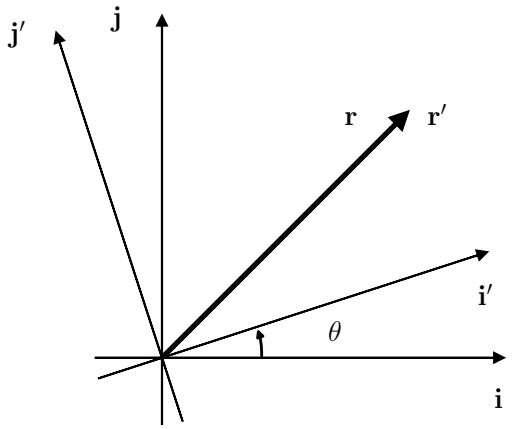

FIGURE 5.1
For passive rotations the rotation matrix expresses how the two coordinate systems are oriented relative to each other. The angle θ is defined as the angle from the unprimed system to the primed system.

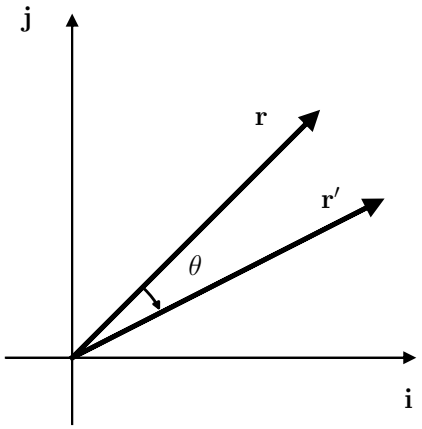

FIGURE 5.2
For active rotations the rotation matrix expresses how the quantity acted on is changed from the unprimed to the primed.

Stated in a more concise way:
For ${}^{B}\mathbf{r} = \mathbf{R}_p(\theta){}^{E}\mathbf{r}$ the rotation matrix gives the coordinates of $\mathbf{r}$ in coordinate frame B as function of how they look in frame E. Where B is rotated counter-clockwise with respect to E by an angle θ.
For ${}^{E}\mathbf{r}' = \mathbf{R}_a(\theta){}^{E}\mathbf{r}$ the rotation matrix gives the coordinates of $\mathbf{r}'$ in coordinate frame E. $\mathbf{r}'$ is the result of rotating $\mathbf{r}$ clockwise by an angle θ.
For some angle ϕ: ${}^{B}\mathbf{r} = \mathbf{R}_p(\phi){}^{E}\mathbf{r} = {}^{E}\mathbf{r}' = \mathbf{R}_a(-\phi){}^{E}\mathbf{r}$.

We wish to estimate the quantity $\mathbf{R}$ solving $\mathbf{r}' = \mathbf{R}\mathbf{r}$. What this quantity represents has two equally valid interpretations. Either it represents the orientation of the primed coordinate system with respect to the unprimed, and facilitates transforming coordinate tuples from the unprimed to the primed system. Or it presents an operator changing $\mathbf{r}$ to $\mathbf{r}'$ and thus provides the coordinates of $\mathbf{r}'$ as a function of $\mathbf{r}$.

In the following the first, passive interpretation will be used because this, as also pointed out in [31], better captures the task of describing the orientation of the vechicle.

Three representations of orientation that are commonly used in aviation and robotics [56, 26] are the direction cosine matrix, Euler angles and quaternions. I will here briefly present all three.

5.1.1 Direction Cosines

The direction cosines express exactly the first interpretation given above of what is represented by a rotation, as it directly describes the components of each primed axis with respect to the unprimed system. The direction cosine matrix is defined as [31] [56]:

$$\left({}^{B}\mathbf{R}_{p}^{E}\right)_{ij} = \left({}^{B}\mathbf{R}_{DCM}^{E}\right)_{ij} = \cos(\theta_{ij}) = \hat{\mathbf{i}}_B \cdot \hat{\mathbf{j}}_E \qquad i,j = \{x,y,z\}. \tag{5.3}$$

Well-known properties of the direction cosine matrix are

$$\left({}^{B}\mathbf{R}^{E}\right)^{-1} = \left({}^{B}\mathbf{R}^{E}\right)^{T} \tag{5.4}$$

$$= {}^{E}\mathbf{R}^{B} \tag{5.5}$$

$$\det\left({}^{B}\mathbf{R}^{E}\right) = +1. \tag{5.6}$$

Time derivative

Following the derivation in [56] of the time derivative for the direction cosine matrix for a body rotating relative to frame E with angular velocity ${}^{B}\boldsymbol{\omega}$ we take the time derivative of both sides of ${}^{B}\mathbf{r} = {}^{B}\mathbf{R}^{E} \cdot {}^{E}\mathbf{r}$:

$$\begin{aligned}
\frac{d}{dt}{}^{B}\mathbf{r} &= \frac{d}{dt}{}^{B}\mathbf{R}^{E} \cdot {}^{E}\mathbf{r} \\
{}^{B}\dot{\mathbf{r}} &= {}^{B}\dot{\mathbf{R}}^{E} \cdot {}^{E}\mathbf{r} + {}^{B}\mathbf{R}^{E} \cdot {}^{E}\dot{\mathbf{r}} \\
{}^{B}\boldsymbol{\omega} \times {}^{B}\mathbf{r} &= {}^{B}\dot{\mathbf{R}}^{E} \cdot {}^{E}\mathbf{r} \\
&= {}^{B}\dot{\mathbf{R}}^{E} \cdot {}^{E}\mathbf{R}^{B} \cdot {}^{B}\mathbf{r} \\
0 &= \left({}^{B}\dot{\mathbf{R}}^{E} \cdot {}^{E}\mathbf{R}^{B} - [{}^{B}\boldsymbol{\omega}]_{\times}\right) \cdot {}^{B}\mathbf{r} \\
{}^{B}\dot{\mathbf{R}}^{E} &= [{}^{B}\boldsymbol{\omega}]_{\times} \cdot {}^{B}\mathbf{R}^{E},
\end{aligned} \tag{5.7}$$

where

$$[{}^{B}\boldsymbol{\omega}]_{\times} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{5.8}$$

is the antisymmetric matrix that facilitates the cross-product calculation in matrix form.

Keeping track of the orientation of the vehicle using the angular velocity can be done by integrating eq. 5.7. In practice the estimated orientation could be updated each time a new measurement was available as ${}^{B}\mathbf{R}_{i}^{E} = {}^{B}\mathbf{R}_{i-1}^{E} + [{}^{B}\boldsymbol{\omega}_i]_{\times} \cdot {}^{B}\mathbf{R}_{i-1}^{E}\Delta t_i$. However, this does not in general leave ${}^{B}\mathbf{R}_{i}^{E}$ orthogonal [56]. An exact solution for the update can be calculated as ${}^{B}\mathbf{R}_{i}^{E} = \exp([{}^{B}\boldsymbol{\omega}]_{\times}\Delta t_i){}^{B}\mathbf{R}_{i-1}^{E}$ [52], for which Rodrigues' formula can be used [36] [40] to calculate the matrix exponential:

$$\exp([\boldsymbol{\omega}']) = I + \frac{[\boldsymbol{\omega}']_{\times}}{|\boldsymbol{\omega}'|}\sin(|\boldsymbol{\omega}'|) + \frac{[\boldsymbol{\omega}']_{\times}^{2}}{|\boldsymbol{\omega}'|^{2}}(1 - \cos(|\boldsymbol{\omega}'|)), \tag{5.9}$$

where $\boldsymbol{\omega}' = {}^B\boldsymbol{\omega}_i \Delta t_i$.
As the estimated matrix becomes non-orthogonal due to numerical errors or due to the first simple update step, it can be approximately "re-orthogonalized" by following the simple procedure described in [49].

5.1.2 Euler Angles

The Euler Angles consist of three angles corresponding to three consecutive rotations of a coordinate system. A convention often used within aerospace engineering and flight is the roll, pitch, yaw convention or xyz-convention[3] represented as the three angles (θ, ϕ, ψ). If the vehicle we want to describe is a plane, we fix a coordinate system to it such that the x-axis points forward, the y-axis points along the right wing and the z-axis following a right-handed coordinate system points downwards. This is illustrated in figure 5.3.
Having the body coordinate system and the fixed reference coordinate system initially coincide, the body frame can be brought to the configuration described by the three Euler angles as follows:

- Firstly the frame is rotated by the angle ψ about its z-axis giving the intermediate axes $\mathbf{a}, \mathbf{b}, \mathbf{c}$ (with $\mathbf{c}$ being parallel to $\mathbf{z}$).
- Secondly, the intermediate $\mathbf{a}, \mathbf{b}, \mathbf{c}$ frame is rotated by the angle ϕ about the intermediate b-axis giving $\mathbf{d}, \mathbf{e}, \mathbf{f}$ (with $\mathbf{b}$ being parallel to $\mathbf{e}$ and $\mathbf{d}$ now parallel to $\mathbf{x}'$).
- And lastly, a rotation by the angle θ around the d-axis brings the frame to the resultant configuration $\mathbf{x}', \mathbf{y}', \mathbf{z}'$.

The Euler angles can in this way be used to parameterize the direction cosine matrix

$$\begin{aligned} {}^B\mathbf{R}^E &= \mathbf{R}_x(\theta)\mathbf{R}_y(\phi)\mathbf{R}_z(\psi) \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\theta & s\theta \\ 0 & -s\theta & c\theta \end{bmatrix} \begin{bmatrix} c\phi & 0 & -s\phi \\ 0 & 1 & 0 \\ s\phi & 0 & c\phi \end{bmatrix} \begin{bmatrix} c\psi & s\psi & 0 \\ -s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ &= \begin{bmatrix} c\psi\, c\phi & c\phi\, s\psi & -s\phi \\ c\psi\, s\phi\, s\theta - c\theta\, s\psi & c\psi\, c\theta + s\psi\, s\phi\, s\theta & c\phi\, s\theta \\ c\psi\, c\theta\, s\phi + s\psi\, s\theta & -c\psi\, s\theta + c\theta\, s\psi\, s\phi & c\phi\, c\theta \end{bmatrix}, \end{aligned} \tag{5.10}$$

where $\cos(\theta)$ and $\sin(\theta)$ have been shortened to $c\theta$, $s\theta$ respectively and likewise for ϕ and ψ.
All 12 possible combinations of the three main rotation matrices have been generated and placed in appendix A for reference.

Time derivative, Euler rates

Again we will follow the derivation presented in [56]. We will again look at the roll, pitch, yaw sequence described above. The time derivative of the Euler angles can be considered as three

[3] Naming or labeling these rotation sequences can lead to ambiguities as the order (as in xyz) might refer to the order of the rotation matrices as they are multiplied together to construct the full matrix or it might refer to the order the rotations are performed (which for xyz if around z then y then x). In this thesis the order refers only to the order in which the matrices are written when multiplied together.

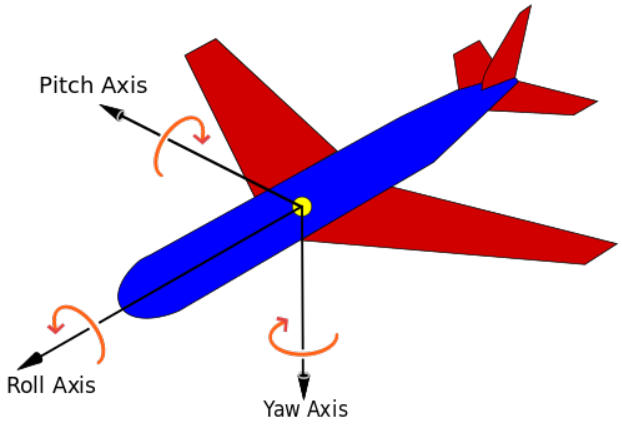

FIGURE 5.3
Illustration of coordinate system fixed on a fixed wing plane [19]. The axis are as is clear from the two illustrations to the roll, pitch and yaw convention used within.

instantaneous angular velocities along the axes $\mathbf{z}$, $\mathbf{b}$ and $\mathbf{d}$ that can be combined into a single angular velocity by the property of infinitesimal rotations [31]:

$$\begin{aligned}\boldsymbol{\omega} &= \dot{\theta}\,\mathbf{d} + \dot{\phi}\,\mathbf{b} + \dot{\psi}\,\mathbf{z} \\ &= \dot{\theta}\,\mathbf{x}' + \dot{\phi}\,\mathbf{e} + \dot{\psi}\,\mathbf{c}\end{aligned} \tag{5.11}$$

As expressed here this quantity is a bit unhandy as it is given as three nonorthogonal components. To find the components of $\boldsymbol{\omega}$ expressed in the body frame we use that eq. 5.11 can be written as

$$\boldsymbol{\omega} = \begin{bmatrix}\mathbf{x}' & \mathbf{y}' & \mathbf{z}'\end{bmatrix}\begin{bmatrix}\dot{\theta}\\0\\0\end{bmatrix} + \begin{bmatrix}\mathbf{d} & \mathbf{e} & \mathbf{f}\end{bmatrix}\begin{bmatrix}0\\\dot{\phi}\\0\end{bmatrix} + \begin{bmatrix}\mathbf{a} & \mathbf{b} & \mathbf{c}\end{bmatrix}\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} \tag{5.12}$$

$${}^{B}\boldsymbol{\omega} = \begin{bmatrix}\dot{\theta}\\0\\0\end{bmatrix} + \mathbf{R}_x\begin{bmatrix}0\\\dot{\phi}\\0\end{bmatrix} + \mathbf{R}_x\mathbf{R}_y\begin{bmatrix}0\\0\\\dot{\psi}\end{bmatrix} \tag{5.13}$$

where we have used that the matrix of the body frame basis vectors ${}^{B}\begin{bmatrix}\mathbf{x}' & \mathbf{y}' & \mathbf{z}'\end{bmatrix}$ is the identity matrix in the body frame and that $\begin{bmatrix}\mathbf{d} & \mathbf{e} & \mathbf{f}\end{bmatrix} = \begin{bmatrix}\mathbf{x}' & \mathbf{y}' & \mathbf{z}'\end{bmatrix}\mathbf{R}_x$.

Thus

$$
\begin{aligned}
{}^{B}\boldsymbol{\omega} &= \mathbf{S}_{xyz} \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} \\
&= \begin{bmatrix} 1 & 0 & -s\phi \\ 0 & c\theta & s\theta c\phi \\ 0 & -s\theta & c\theta c\phi \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} \\
\begin{bmatrix} \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix} &= \mathbf{S}_{xyz}^{-1} \cdot {}^{B}\boldsymbol{\omega} \\
&= \frac{1}{c\phi} \begin{bmatrix} c\phi & s\theta s\phi & c\theta s\phi \\ 0 & c\theta c\phi & -s\theta c\phi \\ 0 & s\theta & c\theta \end{bmatrix} {}^{B}\boldsymbol{\omega}
\end{aligned} \tag{5.14}
$$

Here we notice an inherent problem of the Euler angles, namely that the relation between the angular velocities and the Euler rates has a singularity for certain angles. For the xyz-convention it is at $\phi = \pm\pi/2$.
As measurements of the angular velocity of the body arrive, the Euler angles can be updated as

$$
\begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix}_{i} = \begin{bmatrix} \theta \\ \phi \\ \psi \end{bmatrix}_{i-1} + \mathbf{S}_{xyz}^{-1} \cdot {}^{B}\boldsymbol{\omega} \Delta t_i \tag{5.15}
$$

5.1.3 Quaternion

Quaternions[4] are numbers with one scalar part and three imaginary parts

$$
q = (q_0, \mathbf{q}) = q_0 + q_1 i + q_2 j + q_3 k \tag{5.16}
$$

invented by William Rowan Hamilton in 1843 [27].
When used to represent orientation they can be seen as being closely related to Euler's rotation theorem which states (as interpreted in [31]):

> The general displacement of a rigid body with one point fixed is a rotation about some axis.

That is, any orientation of a rigid body can be changed to any other orientation through a rotation about an axis $\hat{\mathbf{e}}$ by an angle θ. Thus any composition of two rotations about two different axes can be combined into a single rotation about one axis.
In this interpretation [56] the three imaginary components of the quaternion are related to the axis

[4]Several different styles of notation can found in literature. In this thesis we will use the notation found in [32].

by

$$\mathrm{Im}(q) = \mathbf{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix} = \sin(\theta/2)\hat{\mathbf{e}} \tag{5.17}$$

and the scalar part to the angle as

$$\mathrm{Re}(q) = q_0 = \cos(\theta/2) \tag{5.18}$$

Note that quaternions constructed like this have unit length $|q| = \sqrt{q_0^2 + q_1^2 + q_2^2 + q_3^2} = 1$.
A quaternion describing the orientation of the body following the notation used in this thesis so far is written as ${}^{B}q^{E} = {}^{B}(q_0, \mathbf{q})^{E}$. The basis vectors of the body frame are related to the basis vectors of the reference frame by

$$\hat{\mathbf{i}}_B = \mathrm{Im}({}^{B}q^{E} \otimes (0, \hat{\mathbf{i}}_E) \otimes {}^{B}q^{E*}) \tag{5.19}$$

where the superscript $*$ denotes the quaternion conjugate

$$(q_0, \mathbf{q})^* = (q_0, -\mathbf{q})^* \tag{5.20}$$

$${}^{B}q^{E*} = {}^{E}q^{B} \tag{5.21}$$

and $\otimes$ denotes the quaternion product or Hamilton product which is non-commutative and is defined from the relations for multiplication of the basis elements [27]

$$ii = jj = kk = -1 \tag{5.22}$$

$$ij = k = -ji \tag{5.23}$$

$$ki = j = -ik \tag{5.24}$$

$$jk = i = -kj \tag{5.25}$$

From this we can write out the quaternion product

$$\begin{aligned} q \otimes q' &= (q_0 + q_1 i + q_2 j + q_3 k)(q'_0 + q'_1 i + q'_2 j + q'_3 k) \\ &= q_0 q'_0 - q_1 q'_1 - q_2 q'_2 - q_3 q'_3 + \\ &\quad (q_0 q'_1 + q_1 q'_0 + q_2 q'_3 - q_3 q'_2) i + \\ &\quad (q_0 q'_2 + q_2 q'_0 + q_3 q'_1 - q_1 q'_3) j + \\ &\quad (q_0 q'_3 + q_3 q'_0 + q_1 q'_2 - q_2 q'_1) k \end{aligned}$$

Writing the components of the quaternion in a single vector $q = [q_0, q_1, q_2, q_3]^T$ allows us to write the above as

$$q \otimes q' = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} q'_0 \\ q'_1 \\ q'_2 \\ q'_3 \end{bmatrix} \tag{5.26}$$

The quaternions can be used to parameterize the direction cosine matrix as [56]

$$R(q_0, \mathbf{q}) = \begin{bmatrix} 1-2(q_2^2+q_3^2) & 2(q_1q_2+q_3q_0) & 2(q_1q_3-q_2q_0) \\ 2(q_2q_1-q_3q_0) & 1-2(q_1^2+q_3^2) & 2(q_2q_3-q_1q_0) \\ 2(q_3q_1+q_2q_0) & 2(q_3q_2-q_1q_0) & 1-2(q_1^2+q_2^2) \end{bmatrix} \tag{5.27}$$

Time derivative

From [56] we have

$$^B\dot{q}^E = \frac{1}{2}{}^Bq^E \otimes (0, {}^B\boldsymbol{\omega}) \tag{5.28}$$

The orientation at each angular velocity measurement can thus be updated as

$$^B\tilde{q}_i^E = {}^Bq_{i-1}^E + \frac{1}{2}{}^Bq_{i-1}^E \otimes (0, {}^B\boldsymbol{\omega}_i) \tag{5.29}$$

This however will not in general leave the quaternion $^B\tilde{q}_i^E$ with unit length and a normalization step is thus needed

$$^Bq_i^E = \frac{^B\tilde{q}_i^E}{|^B\tilde{q}_i^E|}. \tag{5.30}$$

5.2 Euler's Second Law

The dynamics of a freely spinning rigid object lacking the symmetries common in most textbook examples on mechanics can encompass surprising behavior. An example of this is described and visualized in chapter 7.
The equations describing these dynamics are Euler's rotation equations.

$$\begin{aligned} I_1\dot{\omega}_1 + (I_3 - I_2)\omega_2\omega_3 &= M_1 \\ I_2\dot{\omega}_2 + (I_1 - I_3)\omega_3\omega_1 &= M_2 \\ I_3\dot{\omega}_3 + (I_2 - I_1)\omega_1\omega_2 &= M_3 \end{aligned} \tag{5.31}$$

where I_i is the i'th principal moment of inertia, and ω_i and M_i are the angular velocity of the rigid body and torque along body axis i, respectively.

5.3 Interpretations

Figures 5.4 to 5.6 visualize the same dataset containing information about the motion of a rigid object spinning two full revolutions around a constant axis of rotation $\hat{\mathbf{e}} = \mathbf{e}/|\mathbf{e}|$ where $\mathbf{e} = [1, 2, 3]^T$. For this specific dataset it is probably the axis angle representation that is the easiest to interpret, whereas it would be quite difficult to guess what motion produced the values of the Euler angles. This easier interpretation of the axis angle representation and to some degree the quaternions due to Euler's rotation theorem is the basis of one of the orientation estimation methods developed in chapter 6.

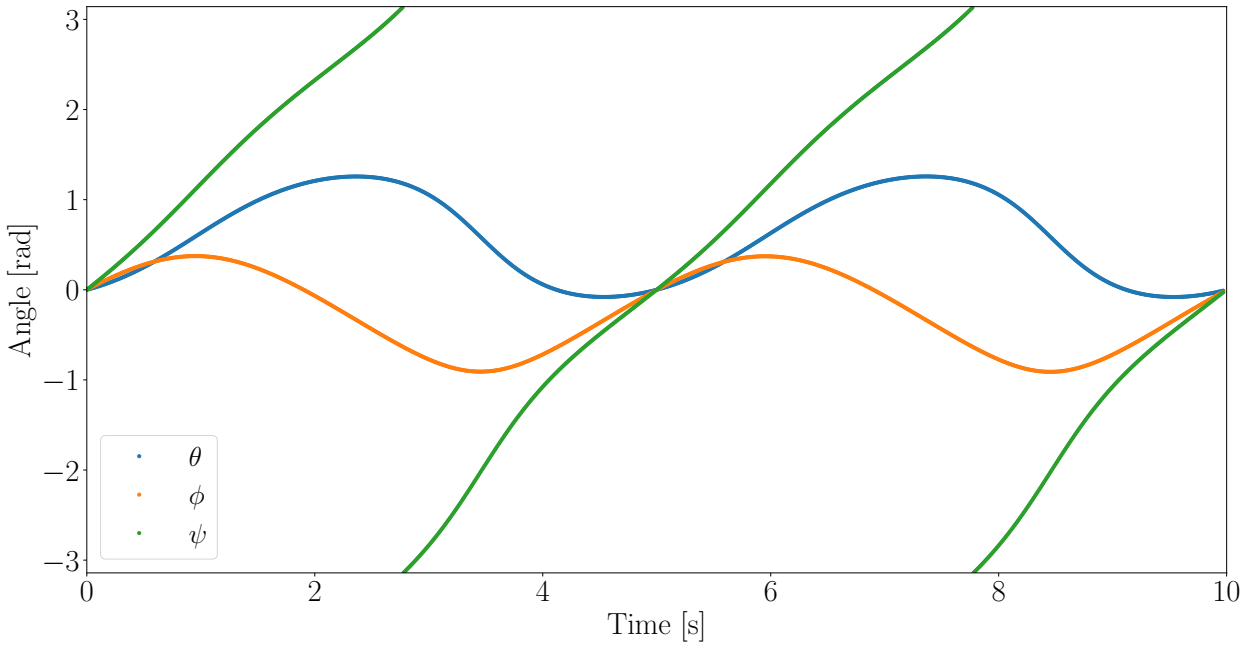

FIGURE 5.4
Visualization of the orientation of a rigid body spinning with constant angular velocity ${}^{E}\boldsymbol{\omega} = \frac{4\pi}{10\cdot\sqrt{14}}[1,2,3]^{T}\frac{rad}{s}$ represented as euler angles.

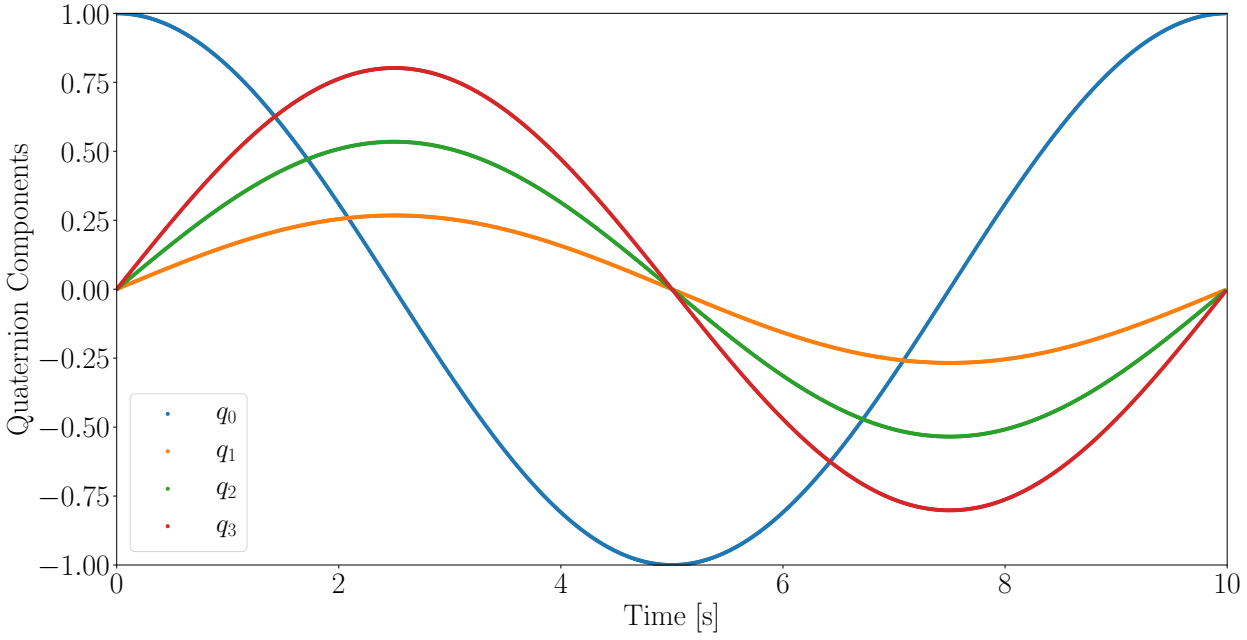

FIGURE 5.5
Visualization of the orientation of a rigid body spinning with constant angular velocity ${}^{E}\boldsymbol{\omega} = \frac{4\pi}{10\cdot\sqrt{14}}[1,2,3]^{T}\frac{rad}{s}$ represented as a quaternion.

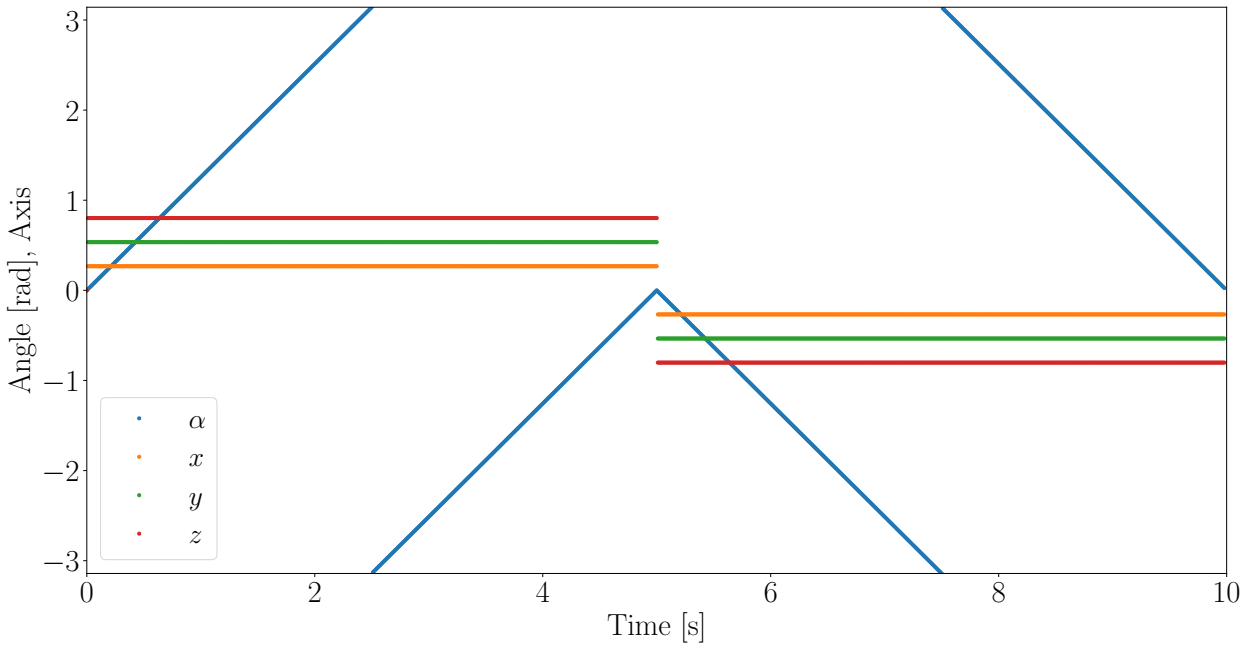

FIGURE 5.6
Visualization of the orientation of a rigid body spinning with constant angular velocity ${}^{E}\boldsymbol{\omega} = \frac{4\pi}{10\cdot\sqrt{14}}[1,2,3]^{T}\frac{rad}{s}$ represented as an angle α and an axis of rotation $[x,y,z]$.

6 Orientation Estimation

The orientation estimates play an important role in both the position estimation problem and in the control of the UAV. Here a thorough description how to obtain accurate orientation estimates is given.

If we knew the angular velocity ${}^B\boldsymbol{\omega}(t)$ and the orientation ${}^BR^E(t=0)$ exactly, the task of estimating the orientation would be the task of solving 5.7, 5.14 or 5.28. Likewise, for perfect information about the Earth's gravitational field and magnetic field we can calculate ${}^BR^E(t) = {}^BR^E({}^B\mathbf{g}(t), {}^B\mathbf{b}(t), {}^E\mathbf{g}, {}^E\mathbf{b})$. The details of how to do this calculation will be handled below.
However, as we only have access to noisy discrete measurements, the estimation problem is better described as the problem of obtaining a combined estimate based on the field measurements and the angular rate measurements together.

6.1 Estimating Orientation Using Vector Observations

If we for a moment assume perfect measurements, that is unbiased and noise free, then for such measurements of a single field, say gravity, given in the body frame for which we know the coordinate tuple in the reference frame (the Earth frame) we can formulate the orientation estimation task as:
Given the coordinates of the vector $\mathbf{r}$ in the two frames B and E, find $\mathbf{R}_s$ such that ${}^B\mathbf{r} = \mathbf{R}_s{}^E\mathbf{r}$.
A solution to this equation will be one of infinitely many, as a unique solution can not be completely determined from only one measurement. We can see this from the fact that if we rotate the body frame around an axis parallel to ${}^B\mathbf{r}$ this would not change the measurement. The most general form solving the above equation is therefore $\mathbf{R} = \mathbf{R}({}^B\mathbf{r}, \theta_1)\mathbf{R}(\hat{\mathbf{e}}, \phi)\mathbf{R}({}^E\mathbf{r}, \theta_2)$ for any θ_1, θ_2 where $\mathbf{R}(\hat{\mathbf{e}}, \phi)$ is a rotation about some axis $\hat{\mathbf{e}}$ by some angle ϕ that alone solves the equation.

For two perfect measurements where a unique solution exist the formulation of the problem becomes:
Given the coordinates of two vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ in each frame find $\mathbf{R}$ such that ${}^B\mathbf{r}_1 = \mathbf{R}^E\mathbf{r}_1$ and ${}^B\mathbf{r}_2 = \mathbf{R}^E\mathbf{r}_2$ and thus $\mathbf{R} = {}^B\mathbf{R}^E$.

6.1.1 Common Axis Quaternion

The following presents a method for determining the quaternion that represents the orientation of the body frame that corresponds to the two representations ${}^B\mathbf{g}$ and ${}^B\mathbf{b}$ in the body.
This method is presented here as the first example of an estimation method, as it takes a pretty intuitive approach to the problem of estimating orientation. However, as we develop this method, it becomes evident that is has its disadvantages.

In the following it makes sense to adopt the active rotation interpretation, as the math involves

functions of the coordinates of the same vector in different frames. By the active rotation interpretation it can be seen as being functions between different vectors. The vectors $\mathbf{g}$ and $\mathbf{b}$ and thus their representation in each frame will be assumed to have unit length to obtain clearer equations. Taking Euler's theorem 5.1.3 as a starting point, we will as a first step find an axis of rotation that can be used to rotate the coordinates of one of the vectors in the Earth frame into the coordinates in the body frame (we use $\mathbf{g}$, but could just as well have used $\mathbf{b}$). Two choices of axes are readily, possible namely the normal vector to the plane spanned by the two vectors ${}^B\mathbf{g}$, and ${}^E\mathbf{g}$ and the bisector of the two vectors

$$\hat{\mathbf{e}}_n = \pm\frac{{}^B\mathbf{g} \times {}^E\mathbf{g}}{|{}^B\mathbf{g} \times {}^E\mathbf{g}|} = \pm\frac{{}^B\mathbf{g} \times {}^E\mathbf{g}}{\sqrt{1-({}^B\mathbf{g} \cdot {}^E\mathbf{g})^2}}, \quad \text{for } |{}^B\mathbf{g} \times {}^E\mathbf{g}| \neq 0 \tag{6.1}$$

$$\hat{\mathbf{e}}_b = \pm\frac{{}^B\mathbf{g} + {}^E\mathbf{g}}{|{}^B\mathbf{g} + {}^E\mathbf{g}|} = \pm\frac{{}^B\mathbf{g} + {}^E\mathbf{g}}{\sqrt{2(1+{}^E\mathbf{g} \cdot {}^B\mathbf{g})}}, \quad \text{for } {}^B\mathbf{g} \neq -{}^E\mathbf{g} \tag{6.2}$$

with corresponding angles given by

$$\cos(\theta_n) = {}^B\mathbf{g} \cdot {}^E\mathbf{g} \tag{6.3}$$

$$\cos(\theta_b) = -1 \tag{6.4}$$

Any vector in the plane spanned by these two vectors, i.e. $\hat{\mathbf{e}}_l = \cos(\phi)\hat{\mathbf{e}}_n + \sin(\phi)\hat{\mathbf{e}}_b,\ \forall\phi \in [0, 2\pi]$ can be used as a rotation axis between frames B and E. This can be seen by noting that the component of ${}^B\mathbf{g}$ and ${}^E\mathbf{g}$ parallel to the axis of rotation are left unchanged, while the length of the component of ${}^B\mathbf{g}$ perpendicular to the axis is equal to the length of the component of ${}^E\mathbf{g}$ perpendicular to the axis for all axes in this plane. The angle for a rotation about the axis $\hat{\mathbf{e}}_l$ is given by $\cos(\theta_l) = {}^E\hat{\mathbf{g}}_{\perp_{\hat{\mathbf{e}}_l}} \cdot {}^B\hat{\mathbf{g}}_{\perp_{\hat{\mathbf{e}}_l}} = {}^E\hat{\mathbf{b}}_{\perp_{\hat{\mathbf{e}}_l}} \cdot {}^B\hat{\mathbf{b}}_{\perp_{\hat{\mathbf{e}}_l}}$, where ${}^E\hat{\mathbf{g}}_{\perp_{\hat{\mathbf{e}}_l}}$ is the unit vector pointing in the direction of the component of ${}^E\mathbf{g}$ perpendicular to $\hat{\mathbf{e}}_l$.

Here a problem with this method becomes clear, namely that for ${}^B\mathbf{g} = -{}^E\mathbf{g}$ no of the two vector spanning the plane of possible axes of rotation are defined. It is however possible to choose any axis in the plane having $\pm{}^B\mathbf{g} = \mp{}^E\mathbf{g}$ as normal vector.

$$\hat{\mathbf{e}}_{any} = \frac{{}^E\mathbf{g} \times \mathbf{v}_{random}}{|{}^E\mathbf{g} \times \mathbf{v}_{random}|}, \quad \text{for } |{}^E\mathbf{g} \times \mathbf{v}_{random}| \neq 0 \tag{6.5}$$

where $\mathbf{v}_{random}$ is any vector not parallel to ${}^E\mathbf{g}$ or ${}^B\mathbf{g}$. The angle of rotation will be $\cos(\theta_{any}) = -1$. That $\hat{\mathbf{e}}_{any}$ is in the plane defined by ${}^E\mathbf{g}$ can be seen from the identity $\mathbf{a}\cdot(\mathbf{b}\times\mathbf{c}) = \mathbf{b}\cdot(\mathbf{c}\times\mathbf{a}) = \mathbf{c}\cdot(\mathbf{a}\times\mathbf{b})$ with e.g. $\mathbf{a} = \mathbf{b} = {}^E\mathbf{g},\ \mathbf{c} = \mathbf{v}$ as ${}^E\mathbf{g} \cdot ({}^E\mathbf{g} \times \mathbf{v}) = \mathbf{v} \cdot ({}^E\mathbf{g} \times {}^E\mathbf{g}) = 0$.

Obtaining a plane of possible axes in the same way for $\mathbf{b}$ we can find a common axis $\hat{\mathbf{e}}_c$ and angle θ_c that can be used to rotate both vectors between B and E. The common axis is given by the intersection of the two planes which can be found by taking the cross product of the normal vectors of the two planes. The common angle θ_c can be calculated in the same way as θ_l above.

The common axis and associated angle can thus be calculated as

$$\hat{\mathbf{e}}_c = \pm\frac{\mathbf{n_g}\times\mathbf{n_b}}{|\mathbf{n_g}\times\mathbf{n_b}|} \tag{6.6}$$

$$\propto (\mathbf{e}_n\times\mathbf{e}_b)_\mathbf{g}\times(\mathbf{e}_n\times\mathbf{e}_b)_\mathbf{b} \tag{6.7}$$

$$\propto ({}^B\mathbf{g}-{}^E\mathbf{g})\times({}^B\mathbf{b}-{}^E\mathbf{b}) \tag{6.8}$$

$$\cos(\theta_c) = \frac{{}^E\mathbf{g}_{\perp\mathbf{e}_c}\cdot{}^B\mathbf{g}_{\perp\mathbf{e}_c}}{|{}^E\mathbf{g}_{\perp\mathbf{e}_c}||{}^B\mathbf{g}_{\perp\mathbf{e}_c}|} \tag{6.9}$$

$$= \frac{{}^E\mathbf{b}_{\perp\mathbf{e}_c}\cdot{}^B\mathbf{b}_{\perp\mathbf{e}_c}}{|{}^E\mathbf{b}_{\perp\mathbf{e}_c}||{}^B\mathbf{b}_{\perp\mathbf{e}_c}|} \tag{6.10}$$

where the identity $\mathbf{a}\times(\mathbf{b}\times\mathbf{c}) = \mathbf{b}(\mathbf{a}\cdot\mathbf{c}) - \mathbf{c}(\mathbf{a}\cdot\mathbf{b})$ has been used in going from line 6.7 to line 6.8.

Using eq. 5.17 and 5.18 the quaternion corresponding to the axis and angle can be calculated as

$$q = \begin{bmatrix}\cos\left(\frac{\theta_c}{2}\right)\\ \hat{\mathbf{e}}_c\sin\left(\frac{\theta_c}{2}\right)\end{bmatrix} \tag{6.11}$$

$$= \begin{bmatrix}\pm\sqrt{\frac{1+p}{2}}\\ \pm\hat{\mathbf{e}}_c\sqrt{\frac{1-p}{2}}\end{bmatrix} \tag{6.12}$$

where $p = \cos(\theta_c) = \frac{{}^E\mathbf{g}_{\perp\mathbf{e}_c}\cdot{}^B\mathbf{g}_{\perp\mathbf{e}_c}}{|{}^E\mathbf{g}_{\perp\mathbf{e}_c}||{}^B\mathbf{g}_{\perp\mathbf{e}_c}|}$.

Except for the undetermined signs and the slightly more complicated expression 6.12 this method is equivalent to R. G. Reynolds; direct quaternion method [41] described below.

6.1.2 Wahba's Problem

The problem of estimating the orientation of a rigid body using vector observations is in the literature most commonly posed as a least square error minimization problem, originally formulated by Grace Wahba [54]. The minimization problem is given as a minimization over $\mathbf{R}\in\mathrm{SO}_3$

$$L(\mathbf{R}_{opt}) = \min_{\mathbf{R}}\left(\frac{1}{2}\sum_i w_i\left|{}^B\mathbf{r}_i - \mathbf{R}\cdot{}^E\mathbf{r}_i\right|^2\right) \tag{6.13}$$

where the sum runs over the different vector observations (in the case of the sensors used here the sum is over the magnetic and gravitational field vectors) and w_i are non-negative weights applied to each vector observation [41].
Several methods for estimating $\mathbf{R}_{opt}$ or instead q_{opt} exists, as well as methods that approximate them. Below is given a list of methods common in the literature [41, 42, 43, 44]. The methods are named in a similar way as it is done in [41].
Implementations of these methods did not seem to be readily available, this combined with the desire to get a better understanding of the estimation problem, encouraged me to implement them myself. These implementation can be found at [20, 21]. These methods have separate advantages,

Matrix solutions

- First solutions
- Unconstrained Least Square
- FOAM
- TRIAD
- SVD

Quaternion Solutions

- QUEST
- ESOQ 1 and 2
- Direct Quaternion
- Davenports q Method

such as including uncertainties (one version of the SVD method calculates the orientation error covariance [41]), however, only Davenport's q Method and the direct quaternion method have been used in the following.

6.1.3 Direct Quaternion Method

The simplest of the quaternion estimation methods listed above is the direct quaternion method by Reynolds.

$$ {}^{B}\tilde{q}^{E} = ({}^{B}\mathbf{g} \cdot {}^{E}\mathbf{b} - {}^{B}\mathbf{b} \cdot {}^{E}\mathbf{g}, ({}^{B}\mathbf{g} - {}^{E}\mathbf{g}) \times ({}^{B}\mathbf{b} - {}^{E}\mathbf{b})) \tag{6.14} $$

$$ {}^{B}q^{E} = \frac{{}^{B}\tilde{q}^{E}}{|{}^{B}\tilde{q}^{E}|} \tag{6.15} $$

If we look at the equation in detail, we see that for specific rotations the method is unable to provide a valid estimate.
If ${}^{B}\mathbf{g} - {}^{E}\mathbf{g} = 0$ corresponding to an axis of rotation $\hat{\mathbf{e}} = {}^{B}\mathbf{g} = {}^{E}\mathbf{g}$, then the method provides no valid estimate of the axis, and similarly for $\hat{\mathbf{e}} = {}^{B}\mathbf{b} = {}^{E}\mathbf{b}$. We also see that $q_0 = {}^{B}\mathbf{g} \cdot {}^{E}\mathbf{b} - {}^{B}\mathbf{b} \cdot {}^{E}\mathbf{g} = {}^{E}\mathbf{g} \cdot ({}^{E}\mathbf{b} - {}^{B}\mathbf{b}) = 0$ in these cases, and thus no valid estimate of the angle can be obtained either. In fact this is the case for all axes of rotation in the plane $\hat{\mathbf{e}} = \cos(\phi){}^{E}\mathbf{g} + \sin(\phi){}^{E}\mathbf{b}$ [42].
A way to overcome this problem is to instead estimate the quaternion ${}^{B}q^{E'}$ where E' is some other reference frame related to frame E by ${}^{E'}q^{E}$

$$ {}^{B}q^{E} = {}^{B}q^{E'}{}^{E'}q^{E} \tag{6.16} $$

An example of this method described in the literature [42, 44] is Shuster's sequential rotation method, where three additional reference frames related to E through ${}^{E'}q^{E} \in \{(0, \hat{\mathbf{x}}), (0, \hat{\mathbf{y}}), (0, \hat{\mathbf{z}})\}$ are used to ensure a good estimate.
The direct quaternion method with Shuster's sequential rotations is what is used on the autopilot and in the following in orientation estimation filters relying on full orientation estimates from vector measurements.

This method, as shown in [42], is however not an optimal solution to Wahba's problem eq. 6.13 though it does result in very accurate estimates. Figure 6.1 and 6.2 visualize the distribution of errors obtained by the robust Davenport's q method [41] and the direct quaternion method with Shuster's sequential rotations. The data has been generated by defining two reference vectors and 10^4 random quaternions used to rotated the reference vectors into the simulated body frame. Zero

mean normally distributed noise was then added to the simulated observations before being used in the estimators. Depicted in these figures are the distribution of the losses as defined by eq. 6.13 with weights equal to 1 for the quaternion obtained by each method.

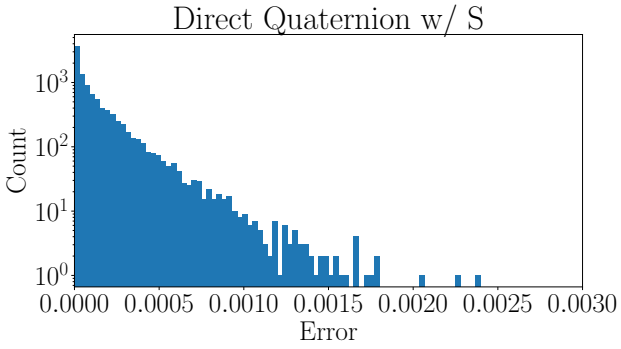

FIGURE 6.1
Distribution of estimation error as defined by eq. 6.13 for 10^4 pairs of simulated noisy vector field observations.

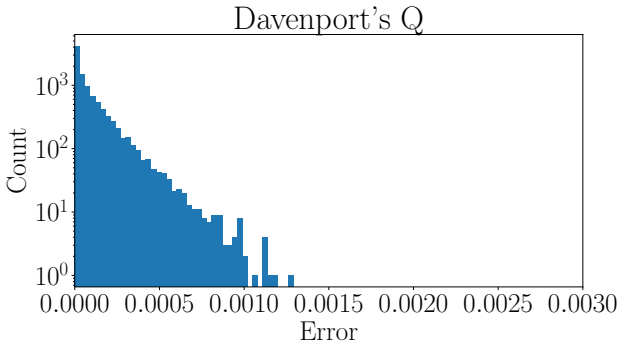

FIGURE 6.2
Distribution of estimation error as defined by eq. 6.13 for 10^4 pairs of simulated noisy vector field observations.

6.2 Combining Angular Velocity And Field Measurements

Using the angular velocity to update the orientation estimate results in the same difficulty as encountered in the position estimation problem, namely that errors accumulate over time and the estimated orientation begins to diverge as time progresses. However, for short periods of time, integrating the angular rate measurements will result in decent estimates.
As stated in the beginning of this chapter we want to use both kinds of measurements for obtaining a single improved estimate. This will allow for corrections of the low frequency error in the angular rate estimate

$$ {}^{B}q^{E}_{ang,i} = {}^{B}q^{E}_{i-1} + \frac{1}{2} {}^{B}q^{E}_{i-1} \otimes (0, {}^{B}\boldsymbol{\omega}) \Delta t_i \tag{6.17} $$

as well as the high frequency error from the field measurements estimate

$$ {}^{B}q^{E}_{vec,i} = q({}^{B}\mathbf{g}(t), {}^{B}\mathbf{b}(t), {}^{E}\mathbf{g}, {}^{E}\mathbf{b}) \tag{6.18} $$

In the following, three methods that has been investigated during this project will be described. Firstly, my own relatively simple method based on slerp (**S**pherical **L**inear int**erp**olation) used widely within computer graphics for smooth animation [51] [27]. Secondly a complementary filter by Mahony et al. [40] and lastly a steepest decent based algorithm by Madgwick [38]. All three filter contain two parameters β and ζ. The first parameter β controls the weighting between the angular rate based estimate and the field based estimate whereas ζ is related to methods acting as lowoass filters where ζ acts as a cutoff frequency.

6.2.1 Slerp

The basic idea of this filter is to make a simple weighted average of the estimates obtained from the angular rate measurements q_{ang} and the field measurements q_{vec},s respectively. But, as unit quaternions are points on the three-sphere $\mathbb{S}_3$, a linear combination of the two as $q = \beta q_{vec} + (1 - \beta)q_{ang},\ \beta \in [0, 1]$ will not in general result in a new unit length quaternion[1].
Instead, the combined estimate can be obtained as an spherical linear interpolation from q_{ang} to q_{vec} as [51]

$$q_{slerp}(\beta) = \frac{\sin(\beta\Omega)}{\sin(\Omega)} q_{vec} + \frac{\sin((1-\beta)\Omega)}{\sin(\Omega)} q_{ang} \tag{6.19}$$

where $\cos(\Omega) = q_{ang} \cdot q_{vec} = \sum_i q_{ang,i} q_{vec,i}$.
This filter allows for a very simple interpretation of how the problem is handled and of what the filter's β parameter does: if $\beta = 0$ then the filter estimate will be based only on the angular rate measurements and if $\beta = 1$ it will be based only on the field measurements. For any other value of β the filter estimate will be a weighted average of the two, weighted by β.

This does not work out of the box however. The reason is that both q and $-q$ represent the same rotation. If we take a closer look at figure 5.5, we see that the quaternion at time 0 equals the quaternion at time 10, though after 10 seconds the rigid body has undergone two full revolutions about the axis of rotation and not just one. This sign ambiguity is present in the result from the direct quaternion method as well, as can be seen in figure 6.3 where the Poinsot dataset described in chapter 7 is used.
We can handle this difficulty if we assume the current estimate $\hat{q}_{slerp}$ to be sufficiently close to the true value. As $\hat{q}_{slerp}$ and q_{QD} in this case should represent almost the same orientation, their dot product should be positive. If the sign is negative we negate q_{QD}:

$$\tilde{q}_{DQ} = \begin{cases} q_{DQ} & \text{if } q_{DQ} \cdot \hat{q}_{slerp} \geq 0 \\ -q_{DQ} & \text{if } q_{DQ} \cdot \hat{q}_{slerp} < 0 \end{cases} \tag{6.20}$$

For the dataset used in figure 6.3 the performance of the Slerp filter has been evaluated for values of β equally spaced between 0 and 1. The result is visualized in figure 6.4 where the Slerp filter is compared to the following two filters.

Expectation Value-Based Gyro Bias Estimation

For the Slerp filter to be able to handle the most likely biased data from the angular rate sensor it needs to be able to estimate this bias. To accommodate this, a method was developed (actually two methods based on the same idea) and is described here.
First we look at the quaternion derivative as a function of a angular velocity measurement with an

[1] If q_{vec} and q_{ang} are not too different, as they are not expected to be in a real application, then the ordinary linear interpolation followed by a normalization and the spherical linear interpolation produces almost identical results. For the former method the interpolation (with normalization) is not linear in β. If the extra number of computations needed to calculate the three sin functions are problematic, then the first method can be applied without losing other than the simple interpretation if how the weighting is done based on β.

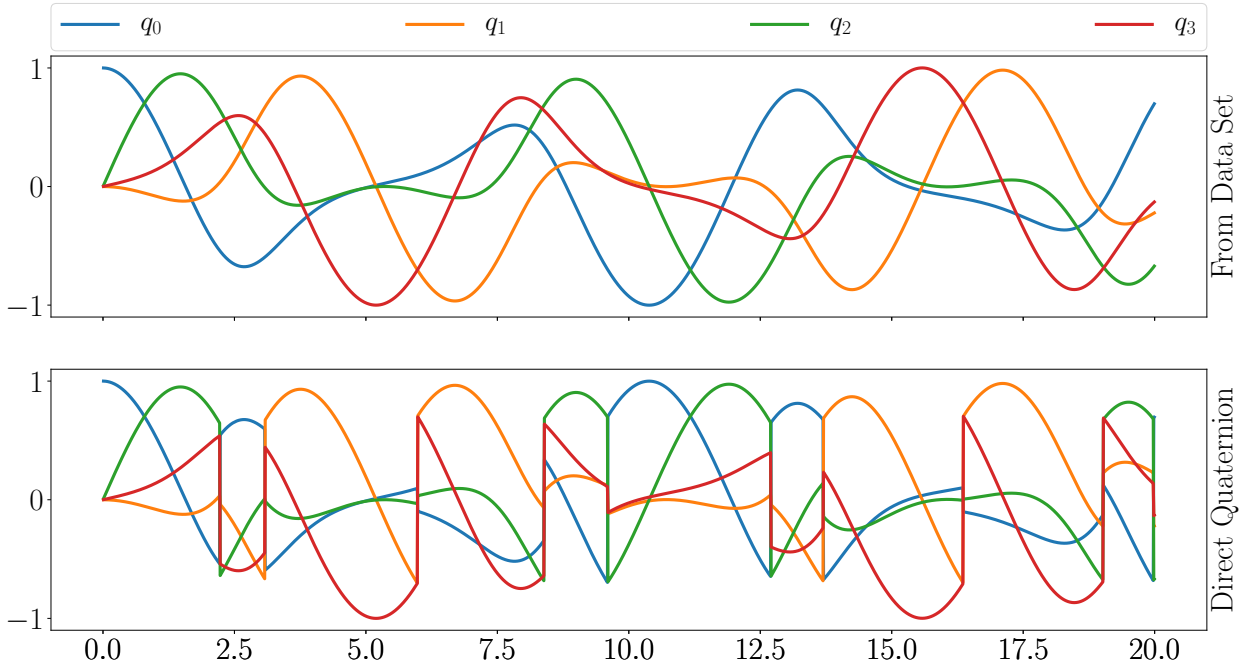

FIGURE 6.3
Visualization of the quaternion as estimated using the direct quaternion method. The method does not guarantee that the estimated quaternion from two very similar pairs of observations are themselves similar. Both q and $-q$ represents the same rotation and thus even the seemingly very different quaternions can represent very similar rotations. The data represented by the quaternion in the top plot is described in chapter 7.

additive bias

$$\dot{q}_{ang,t} = \frac{1}{2} q_t \otimes (0, {}_m\boldsymbol{\omega}_t) \tag{6.21}$$

$$= \frac{1}{2} q_t \otimes (0, \boldsymbol{\omega}_t + \boldsymbol{\mu}) \tag{6.22}$$

$$= \frac{1}{2} q_t \otimes (0, \boldsymbol{\omega}_t) + \frac{1}{2} q_t \otimes (0, \boldsymbol{\mu}) \tag{6.23}$$

$$= \dot{q}_t + \frac{1}{2} q_t \otimes (0, \boldsymbol{\mu}) \tag{6.24}$$

From this we can get

$$(0, \boldsymbol{\mu}) = 2 q_t^* \otimes (\dot{q}_{ang,t} - \dot{q}_t) \tag{6.25}$$

where if we estimate the true quaternion derivative $\dot{q}_t$ using the vector based estimate of q_t as

$$\dot{q}_t = \dot{q}_{v,t} \approx \frac{q_{v,t} - q_{v,t-1}}{\Delta t} \tag{6.26}$$

we can write

$$(0, \boldsymbol{\mu}) \approx 2q_t^* \otimes (\dot{q}_{ang,t} - \dot{q}_{v,t}) \tag{6.27}$$

In practice it is advantageous to use $q_{slerp,t-1}$ in 6.26 instead of $q_{v,t-1}$ as this results in less jittery estimates and as the true quaternion q_t is unknown we use $q_{slerp,t}$ in eq. 6.27.

Alternatively

$$q_{ang,t} = q_{t-1} + \frac{1}{2} q_{t-1} \otimes \dot{q}_{ang,t} \Delta t \tag{6.28}$$

$$= q_t + \frac{1}{2} q_{t-1} \otimes (0, \boldsymbol{\mu}) \Delta t \tag{6.29}$$

$$q_{ang,t} - q_{vec,t} = \frac{1}{2} q_{t-1} \otimes (0, \boldsymbol{\mu}) \Delta t \tag{6.30}$$

$$(0, \boldsymbol{\mu})_t = \frac{2}{\Delta t} q_{t-1}^* \otimes (q_{ang,t} - q_{vec,t}) \tag{6.31}$$

where we have assumed $q_{vec,t} \approx q_t$. This method is the one used in the later sections where the Slerp filter is tested and compared to the other filters. In these tests the vector part of eq. 6.31 is lowpass filtered in order to remove some of the otherwise very dominating high frequency noise. The lowpass filter is governed by the parameter ζ:

$$\hat{\boldsymbol{\mu}}_t = \hat{\boldsymbol{\mu}}_{t-1} + \zeta(\mathrm{Im}((0, \boldsymbol{\mu})_t) - \hat{\boldsymbol{\mu}}_{t-1}) \tag{6.32}$$

6.2.2 Passive Complimentary Filter

In [40] Mahony et al. present a method for fusing the angular rate measurements and the noisy orientation estimates obtained from the field measurements. They also include a method for continuously estimating angular rate bias. The estimated orientation and gyro bias is proven to converge exponentially to the true values.
The concluding formulas for the dynamics of the estimates are

$${}^B\dot{\hat{q}}^E = \frac{1}{2} {}^B\hat{q}^E \otimes (0, {}^B_m\boldsymbol{\omega} - \hat{\boldsymbol{\mu}} + \mathbf{c}) \tag{6.33}$$

$$\mathbf{c} = \beta \, s \, \mathbf{v} \tag{6.34}$$

$$\dot{\hat{\boldsymbol{\mu}}} = -\zeta \, s \, \mathbf{v} \tag{6.35}$$

$$(s, \mathbf{v}) = {}^B\hat{q}^{E*} \otimes {}^Bq^E \tag{6.36}$$

where ${}^B_m\boldsymbol{\omega}$ is the measured angular velocity, $\hat{\boldsymbol{\mu}}$ is the estimated bias and $\tilde{q} = (s, \mathbf{v})$ is a quaternion representing the error between the estimated and actual orientation. The quaternion error is estimated as $\tilde{q} = {}^B\hat{q}^{E*} \otimes {}^Eq^B_{vec}$.
The term $s\,\mathbf{v}$ handles the sign ambiguity of the vector based quaternion estimate in all cases and thus no extra correction is needed.

6.2.3 Madgwick

The Madgwick filter[2] takes a different approach to minimizing Wahba's loss function, as the problem is solved as an iterative steepest decent minimization problem. In contrast to the filters mentioned above no, full orientation estimate from the field measurements such as the direct quaternion estimate is needed.
The optimization problem is formulated as[3] [38, 37]

$$\mathbf{f}(q_{opt}) = \min_{q}(\mathbf{f}(q, {}^{E}\mathbf{g}, {}^{B}\mathbf{g}, {}^{E}\mathbf{b}, {}^{B}\mathbf{b})) \tag{6.37}$$

$$\mathbf{f}(q, {}^{E}\mathbf{g}, {}^{B}\mathbf{g}, {}^{E}\mathbf{b}, {}^{B}\mathbf{b}) = \begin{bmatrix}\mathbf{f_g}\\ \mathbf{f_b}\end{bmatrix} \tag{6.38}$$

$$\mathbf{f_r}(q, {}^{E}\mathbf{r}, {}^{B}\mathbf{r}) = \mathrm{Im}({}^{B}q^{E} \otimes (0, {}^{E}\mathbf{r}) \otimes {}^{B}q^{E*}) - {}^{B}\mathbf{r} \tag{6.39}$$

Resulting in an update function for the orientation estimate as

$${}^{B}q_t^{E} = {}^{B}q_{t-1}^{E} + \left({}^{B}\dot{q}_{\boldsymbol{\omega}_c,t}^{E} - \beta\frac{\nabla\mathbf{f}}{|\nabla\mathbf{f}|}\right)\Delta t \tag{6.40}$$

where $\nabla\mathbf{f}$ is the gradient of the error function to be minimized[4], ${}^{B}\dot{q}_{\boldsymbol{\omega}_c,t}^{E}$ is the quaternion derivative calculated from the bias compensated angular rate measurement. The angular rate bias is estimated as

$$\boldsymbol{\mu} = \zeta\sum_{t}\boldsymbol{\omega}_{\epsilon,t}\Delta t \tag{6.41}$$

$$\boldsymbol{\omega}_{\epsilon,t} = 2{}^{B}q_{t-1}^{E*} \otimes \frac{\nabla\mathbf{f}}{|\nabla\mathbf{f}|} \tag{6.42}$$

6.3 Controlled Tests

The first implementations of all three filters were tested on live data with the algorithms running on the autopilot board, where qualitative tests showed that all filters were all to estimate orientation. However, from time to time the filters exhibited odd behaviors, such as completely losing track of the true value, and without the possibility of quantitative comparable tests the causes for the behavior was very difficult to locate. The tests also allow for a systematic evaluation of the performance of the filters for different values of their two parameters.

[2]The filter referred to here is the first version of the orientation estimation filter in [37] which can be found in several open source projects. The original source code can be found at `http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/`. Later in [37] a revised and very different version of the filter is presented. The source code for an implementation of this newer version was recently publish by Madgwick at `https://github.com/xioTechnologies/Fusion`. This revised version has not been tested here.

[3]As written here, as well as what has been implemented is an algorithm solving for ${}^{B}q^{E}$ instead of ${}^{E}q^{B}$ as in Madgwick's own version

[4]See appendix C for how the gradient is calculated

6.3.1 Angular Rate vs. Field Measurements

The Slerp, Mahony, and Madgwick filters are all designed towards providing a combined orientation estimate that is superior to estimates based on the field measurements and angular rate measurements alone. To evaluate this desired property all three filters have been tested on a dataset based on the Poinsot dynamics described in chapter 7. Two versions of this dataset, with different noise characteristics, have been used for testing filter behavior when angular rate bias was not being estimated, that is $\zeta = 0$. The results hereof are visualized in figure 6.4 where it is clear that all 6 curves describing the estimation error have a minimum for some value of the filters' β parameter. The Slerp filter's weighting parameter β makes sense only in the range from 0 to 1. The filter has been tested for 100 different values of β uniformly chosen between 0 and 1 and thus the values of the endpoints of the Slerp filter curves give the errors resulting from using either solely the angular rate measurements ($\beta = 0$) or solely the field measurements ($\beta = 1$). The range of beta values of the Madgwick and Mahony filters are not confined to be between 0 and 1 by design, and the beta values shown on the axis has been scaled by the highest beta value of the parameter scan for each filter (these values are given in the plot caption).
The visualized error term is the mean over the entire dataset with the error compared to the ground truth at each sample, defined as

$$\begin{aligned}\text{Error}_i = &\left(\text{Im}({}^{B}q^{E}_{filter,i} \otimes (0, {}^{E}\mathbf{g}) \otimes {}^{B}q^{E*}_{filter,i}) - {}^{B}\mathbf{g}_i\right)^2 + \\ &\left(\text{Im}({}^{B}q^{E}_{filter,i} \otimes (0, {}^{E}\mathbf{b}) \otimes {}^{B}q^{E*}_{filter,i}) - {}^{B}\mathbf{b}_i\right)^2\end{aligned} \tag{6.43}$$

where ${}^{B}q^{E}_{filter,i}$ is the estimated orientation after seeing sample i and ${}^{B}\mathbf{g}_i$ and ${}^{B}\mathbf{b}_i$ are the true values of gravity and Earth's magnetic field in the body frame of the i'th sample.
As expected, all filters perform equally well when using only the angular rate measurements. These estimates only result in precise estimates when there is no consistent bias. The difference in errors for the highest values of β used in each filter should not be compared as β_{max} is to some degree arbitrarily chosen for the Madgwick and Mahony filters. It does on the other hand make sense to compare the smallest error across the filters for each dataset.

The filters behave as intended. When the redundancy in orientation information from the angular rate and field measurements can be utilized an improved orientation estimate is obtained. The error stemming from integrating biased angular rate measurements leads to an inaccuracy in the estimated orientation, as the bias contributes a non-zero mean error to the filters' orientation update, whereas the field based orientation estimates are accurate but imprecise. Fusing these sources of orientation information improves accuracy and precision.

6.3.2 Bias Estimation

As presented above all three filters have a way of estimating angular rate bias while simultaneously estimating the orientation represented as the quaternion ${}^{B}q^{E}$. A correctly estimated bias should allow for smaller values of β to be used and with that less jittery orientation estimates.

To investigate the filters' ability to estimate bias as a function of ζ as well as the effect of a correctly estimated bias, a grid scan over both β and ζ with 200 values for each has been done. The results here do not directly translate to how the filters should be tuned in a real world application, but they do work as a way to investigate whether the filters behave as desired.

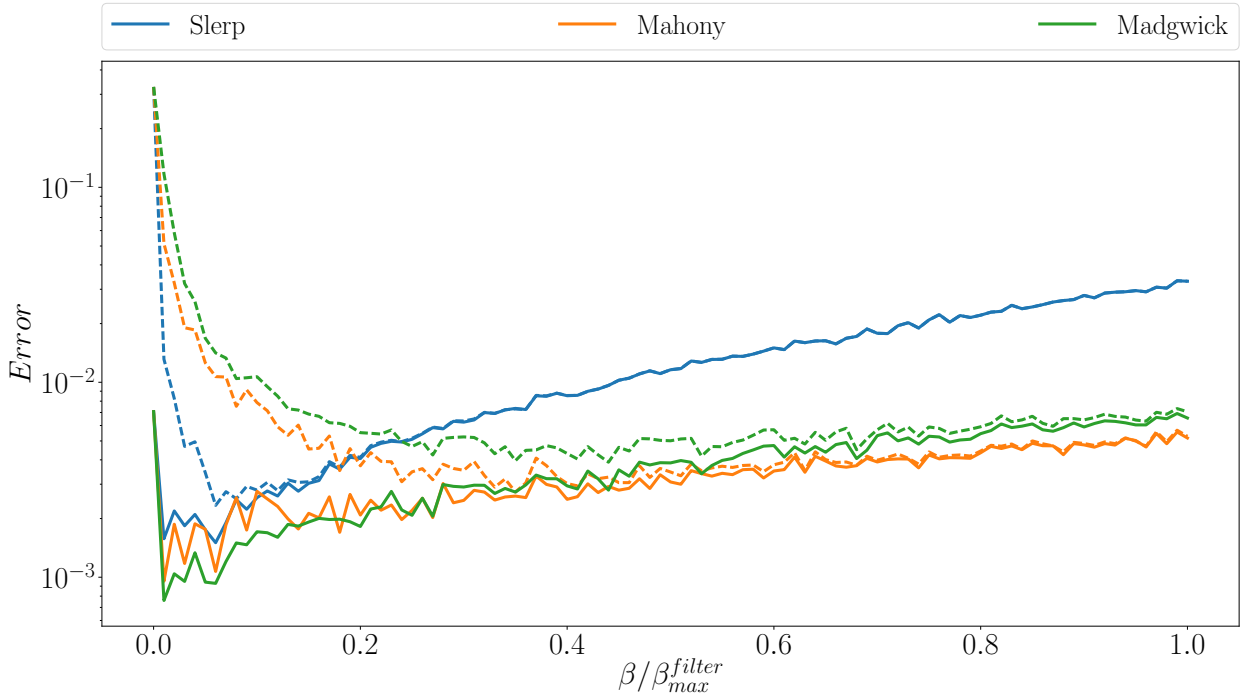

FIGURE 6.4

Average error as function of β for the three filters. It is clear that the estimate using both kinds of measurements is superior to the estimates based on only a single kind.
The two curves for each filter represents the result from tests using the same dataset but with different noise characteristics. The noise is normally distributed with mean μ and standard deviation σ, $\mathcal{N}_{type}(\mu, \sigma)$: Solid line: $\mathcal{N}_{vec}(0, 0.1)$, $\mathcal{N}_{ang}(0, 0.1)$. Dashed line: $\mathcal{N}_{vec}(0, 0.1)$, $\mathcal{N}_{ang}(0.1, 0.1)$.
The range of beta values tested goes from 0 to β_{max}^{filter} with max values being $\beta_{max}^{slerp} = 1$, $\beta_{max}^{Mahony} = 50$, $\beta_{max}^{Madgwick} = 2$. $\zeta = 0$ for all 6 curves.

The dataset used in this test is the same as the one used to produce the dashed curves in figure 6.4 and the same as visualized in figure 6.3. The results of this test are visualized in figure 6.5. The left hand side plots visualize the same error term as in figure 6.4 and thus visualize how well the orientation has been estimated throughout the dataset. The right hand side plots visualize how well the bias has been estimated with the error term given as the mean over the entire dataset of the squared norm of the difference between the estimated bias and the true bias.

$$\text{Bias Error}_i = |\hat{\boldsymbol{\mu}}_{filter,i} - \boldsymbol{\mu}_{true}|^2 \tag{6.44}$$

In both plots the minimum error value in each column has been colored bright yellow. These dots visualize the trend of the optimal value for β as ζ is increased. The trend is as desired, i.e. as the estimated bias approaches the true bias, the optimal value of β decreases indicating that weighting the angular rate measurements higher improves the filters' estimate of ${}^{B}q^{E}$.
Another trend that can be seen from these plots is that there is a positive correlation between β and ζ for the minimum bias estimation error. If the estimated bias is far from the true bias, then the overall orientation estimate will accumulate errors as the integrated angular rate measurements

corrected with a wrong bias will contain some error at each sample. As the bias estimate at each update depends on the current orientation, a wrong orientation estimate will lead to a wrong update of the bias, gradually worsening both estimates. A higher value of β will counteract this as the unbiased field based estimates are weighted higher.

6.3.3 Convergence from Field Measurements

Another thing that has proven interesting to investigate is the filters' convergence characteristics as they are very different. The test for convergence counts how many update iterations the filters need to go through to go from a wrong orientation estimate to the true one

$$\text{number of iterations} = N({}^{B}q^{E}_{wrong}, {}^{B}q^{E}_{true}) \tag{6.45}$$

The filters are updated using the true value of ${}^{B}\mathbf{g}$ and ${}^{B}\mathbf{b}$ at each iteration, and are repeatedly updated until the estimation error, as defined by 6.43, goes below some preset threshold.

A very thorough investigation could test the convergence from any wrong orientation to any true orientation and in that way test all possible cases. However such a test spans 6 degrees of freedom and would thus be computationally expensive and difficult to interpret as no immediate visualization seems to be available.
Instead a test of convergence towards only a single true orientation has been done[5]. The filters are initialized with a orientation estimate with a normalized version of a quaternion with components chosen uniformly[6] between -1 and 1. The true orientation is chosen to be the identity ${}^{B}q^{E}_{true} = q_I = (1, 0, 0, 0)$.

$$N({}^{B}q^{E}_{wrong}, (0, \mathbf{0})) \tag{6.46}$$

The convergence times obtained from these tests are visualized in figure 6.6. The tests were done using values of β close the optimal values as determined from figure 6.5 for $\zeta = 0$.

The very different convergence times are not inherently good or bad, as both low and high convergence times have both advantages and disadvantages. Slow convergence times means a filter will only be affected very little from measurements completely ruined by noise or distortions as only a very small step towards these noisy observations will be taken. Conversely, rapid movements might not be captured by the angular rate sensor's measurements,[7] so the orientation estimate that because of this will contain a large error will need to converge back to an unbiased estimate. In this case fast convergence is preferred, and hence the slerp filter's behavior is preferable.

6.4 Working With Real Data

In the above tests of the filters' behavior as functions of β and ζ the data provided to the filters were corrupted only by uncorrelated normally distributed noise. This is however unlikely to be the case in real applications. The accelerometer will provide measurements of all accelerations acting

[5] Whether the full test of convergence from any orientation to any orientation would provide interesting information other that what results from this test does seems unlikely anyway.

[6] This method of generating random quaternions does not provide uniform samples from $\mathbb{S}_3$, but as uniformity is not really important in this test this simpler method is used.

[7] This effect has been observed in tests done earlier during this project. The cause for it has not been determined.

on the body and thus measurements will by due to gravity and other external forces all of which will corrupt the field measurement based orientation estimate. Combined with possibly distorted magnetic field measurements which will also worsen the orientation estimate this calls for methods for minimizing these effects.
Some especially important distortions for flying vehicles to handle are distortions resulting in a tilted (rotations about axes in the xy-plane) orientation estimate, as erroneous tilt estimates are likely to cause the vehicle to crash. A method from [37] used in the Madgwick filter makes sure that magnetic field measurements only contribute to estimating yaw, the rotation about the direction of gravity, by changing ${}^{E}\mathbf{b}$ at each sample. This method is unfortunately a little cumbersome for the direct quaternion method when using Shuster's sequential rotations, as the three additional reference vectors need to be constructed each time a new sample is received.

A more general method that was implemented early on in the project, in the autopilot software, is a simple rejection criteria based on the norm of the measurements. If the norm of the measurement exceeds some predefined threshold, i.e. $||{}^{B}_{m}\mathbf{r}|^2 - |{}^{E}\mathbf{r}|^2| > \gamma$, then the specific sample is excluded from the update step. This will only filter out the subset of all possible distortions that change the norm of the measurement. However, this method has proven in qualitative test to produce significant improvements to the robustness of the estimate. For filters using a full quaternion estimate, such as the direct quaternion method, rejecting either the acceleration or magnetic field measurement means no full estimate can be made and the filters' quaternion estimate can be updated using only the angular rate measurement. This is however not the case for the Madgwick filter, where a step along the gradient (though another gradient) is possible from only one field measurement.

The trend, seen in the figure 6.5, that the value of β can be lowered when good bias estimates are obtained is especially important for the control of for example multirotors. The less jittery orientation estimates allow for higher responsiveness in the control algorithms as rapid changes in orientation can be assumed to be due to actual changes in orientation and not just noise in the estimate. The possibility of relying more heavily on the angular rate measurements is also advantageous if the controlled vehicle performs aggressive maneuvers as these maneuvers will likely involve high accelerations which as described above makes it difficult to get good field based orientation estimates.

One of the main ideas behind the three filters presented here is that they can use field measurements to infer the orientation of the sensor taking the measurements. One of these measurements comes from the accelerometer, which aims to determine the direction of gravity in the body frame. That this measurement is actually useful in practice is however not entirely obvious.
Imagine a multirotor in 2D as in figure 7.2 with propellers spinning, hovering above ground, standing completely still. In this situation an accelerometer on board the vehicle would measure (disregarding noise for now) exactly $1g$ along the body z-axis which in this case is parallel to the Earth z-axis. Imagine instead the same vehicle still with zero velocity $\dot{\mathbf{r}}(t=0) = \mathbf{0}$ but now rotated by some angle θ of say 30 degrees counter-clockwise while still having the propellers producing the same thrust as before. Now the on-board accelerometer would measure exactly $1g$ along the body z-axis. That is, the measurement has not changed at all[8]. And how could it change? It is still only the propellers that exert a force on the body and they by construction produce a thrust along the body z-axis

[8] A calculation showing this can be found in appendix C

no matter how the vehicle is oriented relative to the world. And yet we want, in this situation, to use the measurement from the accelerometer to determine the tilt of the vehicle. Obviously, if the propellers keep spinning at the same rate, not enough thrust is produced along the Earth z-axis to counteract gravity, but even if the thrust is increased to $\frac{|\mathbf{g}|}{\cos(\theta)}$ the situation is the same, the accelerometer only measures a component along the body z-axis.
What we want to measure is ${}^{B}\mathbf{g}$, but what the accelerometer actually measures is gravity along the Earth z-axis compounded with acceleration in some direction in the Earth frame, rotated into the body frame. In the body frame, the measurement can separated into two components: acceleration due to the propellers along the body z-axis and some other acceleration:

$$\begin{aligned} {}^{B}_{m}\ddot{\mathbf{r}} &= {}^{B}\mathbf{R}^{E}\left({}^{E}\mathbf{g} + {}^{E}\ddot{\mathbf{r}}\right) = {}^{B}\mathbf{R}^{E}\left(g\hat{\mathbf{z}}_{E} + {}^{E}\ddot{\mathbf{r}}\right) && (6.47)\\ &= a_{p}\hat{\mathbf{z}}_{B} + a_{o}{}^{B}\hat{\mathbf{v}} && (6.48)\end{aligned}$$

where $\hat{\mathbf{z}}_{E}$ is the unit vector along the z-axis of the Earth frame, a_p and a_o are the accelerations originating from the **p**ropellers and from **o**ther sources and ${}^{B}\hat{\mathbf{v}}$ is some unit length vector.
Say for simplicity that ${}^{B}\mathbf{R}^{E}$ again corresponds to a rotation of 30 degrees counter-clockwise so that for ${}^{B}_{m}\ddot{\mathbf{r}}$ to equal ${}^{B}\mathbf{g}$ we must have that ${}^{E}\ddot{\mathbf{r}} = \mathbf{0}$ and thus ${}^{B}\mathbf{g} = a_{p}\hat{\mathbf{z}}_{B} + a_{o}{}^{B}\hat{\mathbf{v}}$. Again for the vehicle to have no acceleration along $\hat{\mathbf{z}}_{E}$ we must have that $a_{p} = \frac{|\mathbf{g}|}{\cos(\theta)}$ and for the vehicle to have no acceleration at all it must be the case that $a_{o}{}^{B}\mathbf{v} = \tan\theta|\mathbf{g}|\hat{\mathbf{x}}_{E}$. In words; something has to counteract the horizontal thrust component from the propellers. This "something" can in ordinary circumstances only be caused by drag from moving through the air.
We must have that ${}^{E}\ddot{\mathbf{r}} = \mathbf{0}$ to be able to measure purely the effects of gravity ${}^{B}\mathbf{g}$ in the body frame. But this can only happen, for a non-zero tilt, if the vehicle has reached what could be called terminal horizontal velocity where drag forces exactly opposes the horizontal trust component from the propellers.
If on the other hand we from other sources already know ${}^{E}\ddot{\mathbf{r}}$, we can subtract it from ${}^{B}_{m}\ddot{\mathbf{r}}$ to get ${}^{B}\mathbf{g} = {}^{B}_{m}\ddot{\mathbf{r}} - {}^{B}\mathbf{R}^{E}\,{}^{E}\ddot{\mathbf{r}}$ and in that way get the information we need for the orientation estimate.

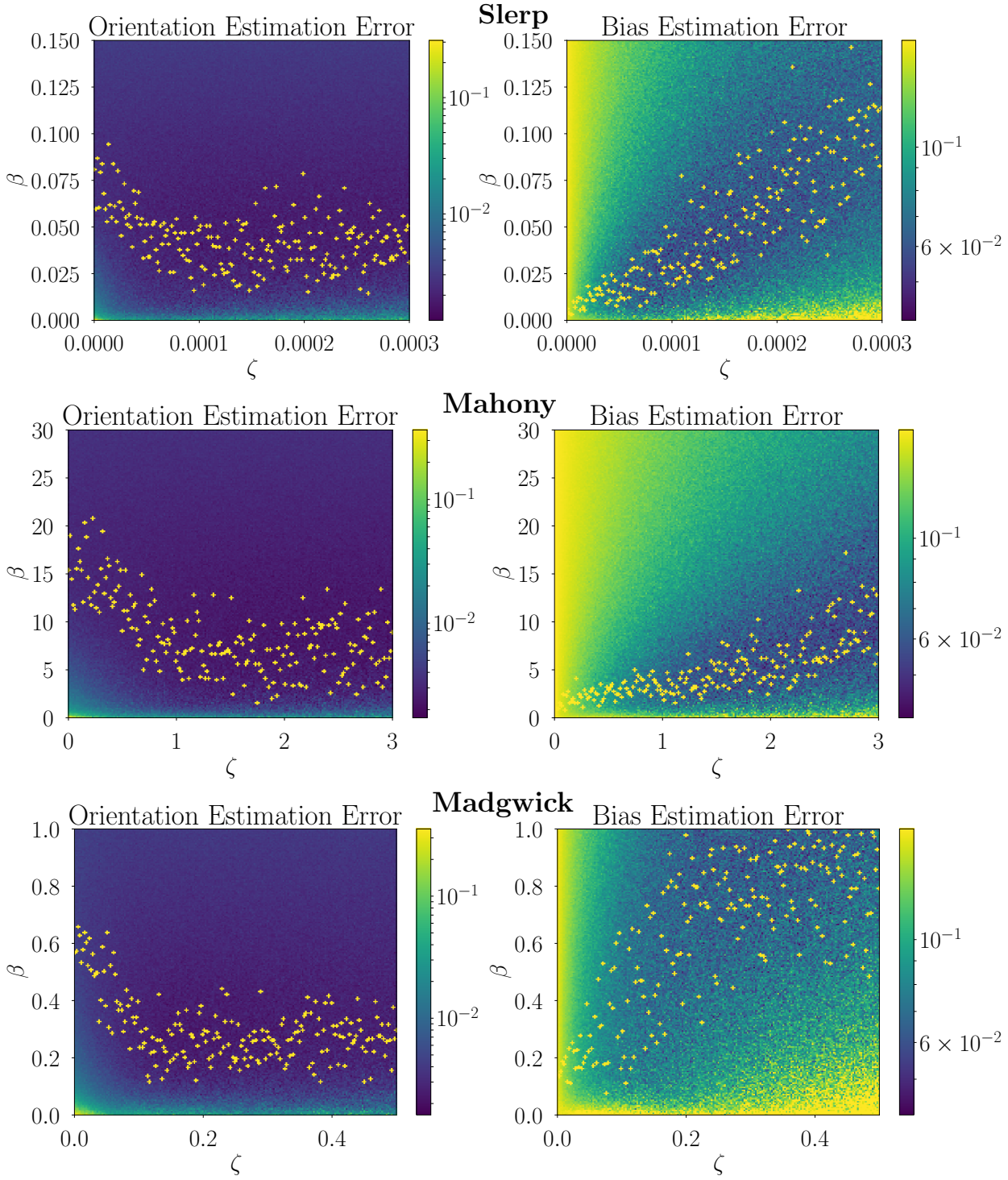

FIGURE 6.5

Visualization of the orientation estimate error and bias estimate error. Yellow dots indicate minimum error values for each value of ζ. For a full description of what is being visualized see text at the start of the section on bias estimation.

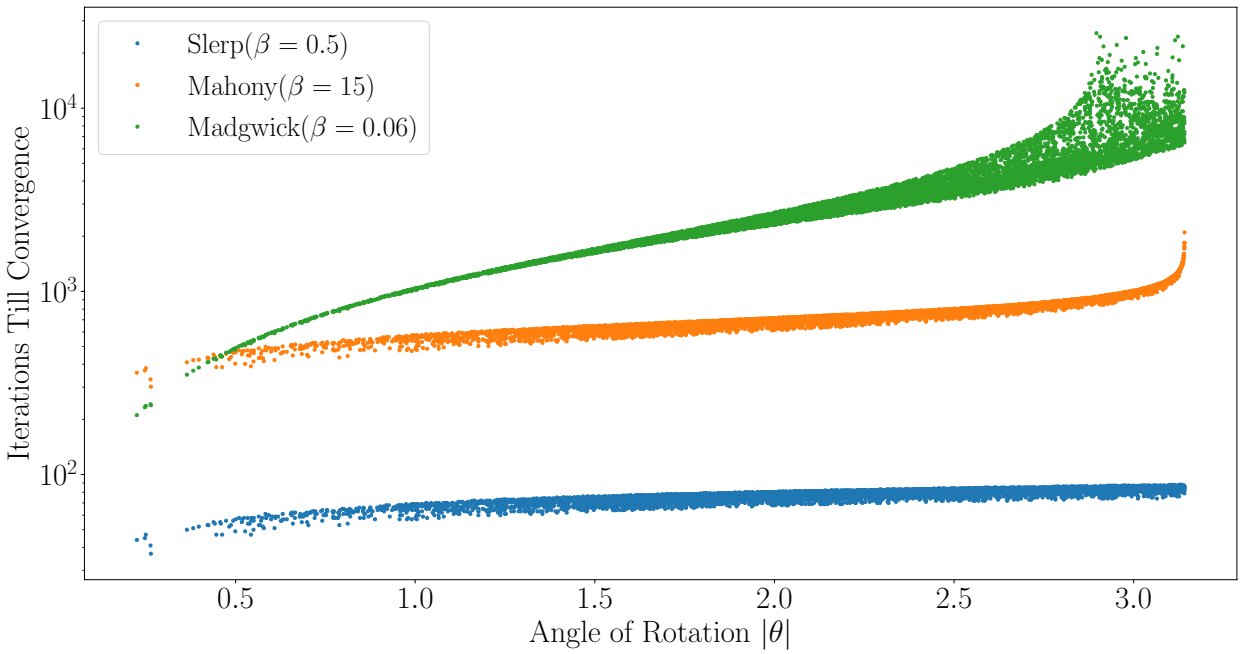

FIGURE 6.6
Number of iterations needed for the filters to converge to the true orientation from a random orientation as function of the angle between the two. See section on convergence from field measurements for a full description.

7 Simulation

A simulation was written early in the project to get a better understanding of the dynamics of rotations in 3 dimensions, and in particular to see the effects of Euler's rotation equations 5.31. This allowed for the possibility of generating datasets describing motions likely to be encountered in actual flight. But most importantly the simulation allowed for generating idealized data, that can be used to test that the filters behave properly, as their estimates can be compared to a ground truth. Furthermore, effects of perturbations from the idealized data can be investigated one at a time, which makes debugging much easier.

Simulations has also been used for testing the control algorithm along with the trajectories described in chapter 8.

The simulation was initially written in Python to allow for rapid development and to be able to make use of the plotting library matplotlib [22]. For the interested, the Python implementation can be found at [23]. The autopilot code is written in C++ and a version of the simulation has been written in C++ as well, but is still under development. The C++ version allows for much easier testing and debugging of the code running on the autopilot, as the same code can be used in both. This implementation can be found at [24]. It should noted that both simulations, while rather advanced in aspects, still need improvements and extensions in others, if they are to be used in practical and more demanding scenarios.

7.1 Spinning T-Handle

Some of the subtleties contained in rigid body rotations in 3D is surprising to many physicists, both students and experienced researchers. An example of a surprising phenomenon that recently got attention from a video clip on the internet, is that of the behavior of a spinning asymmetric object which seemingly breaks conservation of angular momentum as its angular velocity continuously change over time[1]. This effect can be seen for rigid objects with principal moments of inertia $\lambda_1 < \lambda_2 < \lambda_3$ set to spin around an axis close the principal axis with the intermediate principal moment of inertia λ_2.

This behavior has been used to test the validity of the simulation. The continuous change in angular velocity was used to qualitatively validate the simulation while conservation of angular momentum was used as a quantitative measure for the validity. The simulation is done without gravity acting on the object.

Figure 7.1 is an illustration of this behavior though the static plot does not do the actual simulation much justice.

[1] I will encourage any reader who is unfamiliar with video clip to watch it. The video can be found at [25].

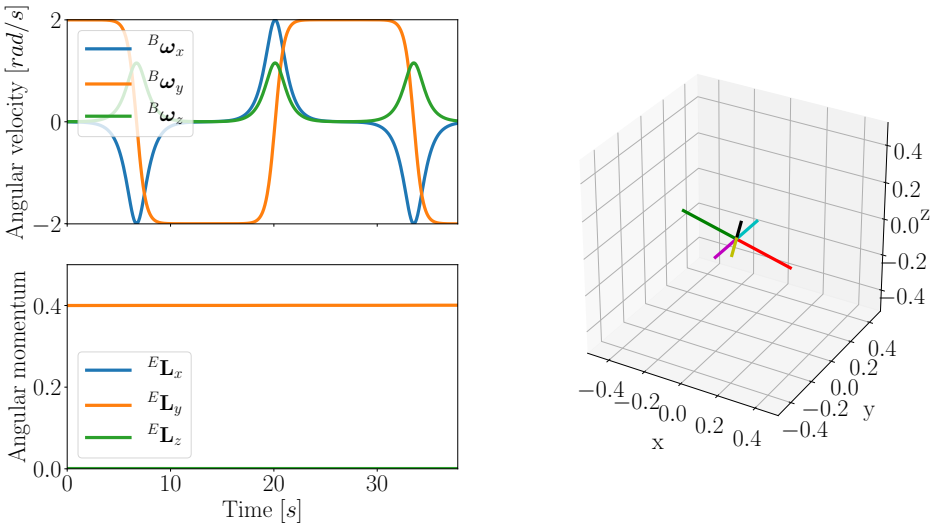

FIGURE 7.1
Visualization of the torque free motion of a spinning asymmetric rigid object. The changes in angular velocity is described by Poinsot's ellipsoid [31]. As there are no external torque on the object its angular momentum is constant even though its angular velocity continuously changes. The plot on the right indicates the ellipsoid axes. The axes segments on each side of the origin are color differently to make it easier to, when the axes flip when the simulation is running. Data from simulations of this behavior is used as a test dataset in other parts of this thesis.

The reason for using this behavior for a dataset is that it contains rotations along different axes and in that way ensures that the filters can be tested with a wide variety of combinations of angular velocities and orientations.

7.2 2D Simulation

A simplification of the full simulation to 2 dimension has also been written. This allows for easier testing of some of the fundamental ideas behind some of the controller that have been tested during the project. A visualization of a functioning 2D flight controller can be seen in figure 7.2. Again the static plot makes it difficult to get an idea of the dynamics of the system, but the UAV simulated with noisy dynamics is able to reach any point without crashing.

7.3 3D Simulation

The full 3D simulation with both rotational and linear dynamics has been used for testing control algorithms, where inputs from an actual joystick connected to the computer was used to control a simulation, with one of the controllers facilitating stabilized flight. In others tests these simulations have been used for investigating automatic trajectory tracking. A visualization of this is given in figure 7.3. At the time of writing tests of the different control methods have no noise introduced which means the control algorithm have perfect information about the orientation of the vehicle.

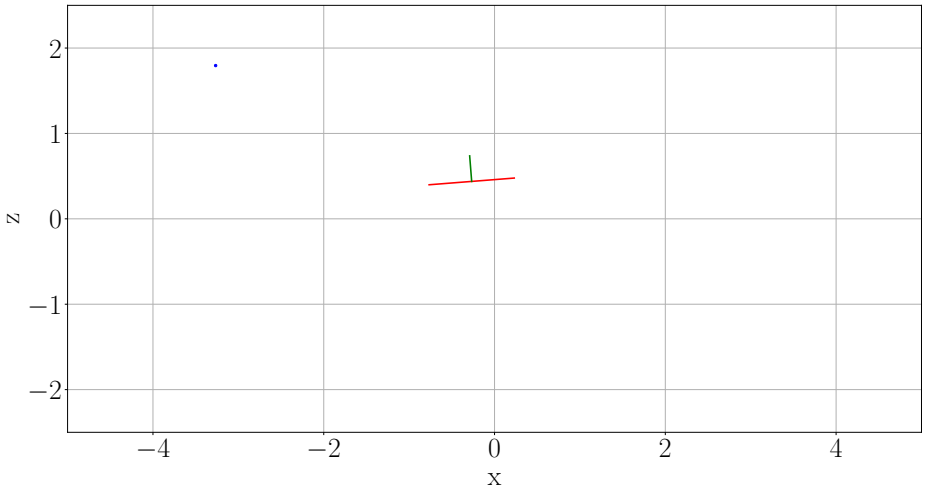

FIGURE 7.2
Visualization of 2D simulation of autonomous multirotor in the middle of a maneuver to reach the blue dot after hovering at the origin. The red bar indicates the plane of the the propellers and the green bar visualizes the body z-axis. The blue destination dots can be placed by left clicking in the plot window of the running simulation.

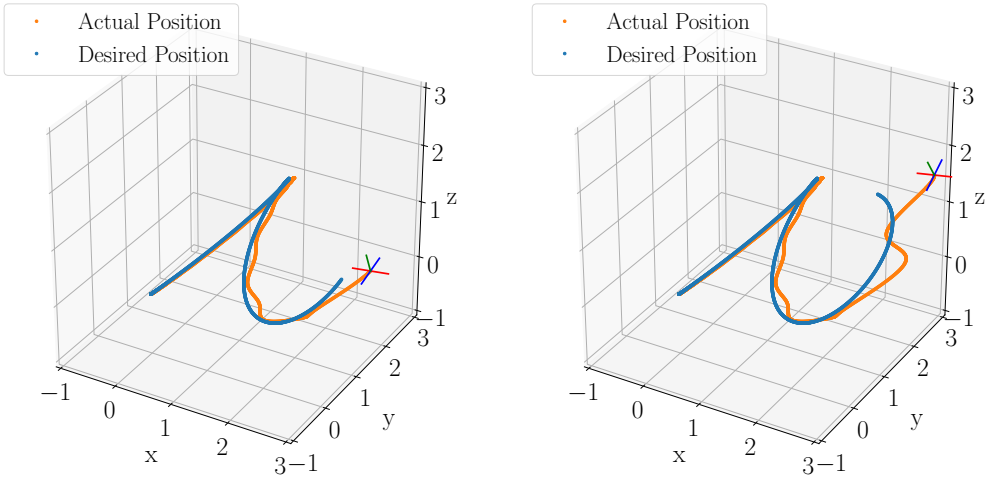

FIGURE 7.3
3D simulation of multirotor doing a poor job of autonomously following a smooth trajectory. Both figures visualize the same flight, with the right being a few seconds further into the flight. In this example the control algorithm is unable to provide consistent error minimization. The UAV exhibits oscillations of increasing amplitude.

8 Path Planning And Control

One desirable feature for an autonomous UAV is for it to be able to follow a path given by a collection of coordinates or waypoints in 3D space. In some cases it might be sufficient to simply have the path specified as a continuous collection of straight line segments, whereas if fast traversal of the path is needed then the possibly sharp corners between each segment will pose a problem, as accelerations exceeding the capabilities of the vehicle might be needed in order to stay sufficiently close to the path. One possible way of handling this is to construct a smooth path going through all the waypoints. This method is described in the following section on derivative minimization.

For a fixed-wing aircraft the planned trajectory needs to respect the limits of the maneuverability of the specific aircraft, such as not exceeding some maximum curvature. Constructing trajectories for fixed-wing aircrafts is out of the scope of this thesis, but a good staring point would the work of L. E. Dubins [28]. He proves that the shortest path in 2D between two points with predefined initial and final tangential directions, which respects a maximum curvature limit, is a path consisting of arcs of maximum curvature and straight lines. Having the additional feature of being a shortest path makes this type of trajectory a good candidate fixed-wing trajectories.

8.1 Minimized Derivative Paths

Our goal is to construct a trajectory $\mathbf{x}(t)$ that passes through an ordered collection of points in space $P = \{\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n\}$ at times $\{t_0, t_1, \ldots, t_n\}$. Infinitely many of such trajectories exist, so we need a sufficient number of constraints if we want to find a unique solution. The method for constructing such paths described here is based on material from [48]. We impose a constraint, that also ensures that the path is smooth, by demanding its m^{th}, $m \geq 2$ derivative to be minimized. Functions fulfilling this, as can be found using calculus of variation, are piecewise-defined polynomials of order $2m-1$. For trajectories passing through $n+1$ waypoints in 3 dimensions this function is described by $3n$ polynomials, one in each dimension for each piece. For multirotors we want to minimize the fourth order derivative, as this produces a path that the vehicle is able to follow using the linear trust and torques from its motors [48].

The problem can be formulated as

$$\mathbf{x}_{opt}(t) = \underset{\mathbf{x}(t)}{\operatorname{argmin}} \left(\int_{t_0}^{t_1} \frac{d^m}{dt^m}\mathbf{x}(t)^2 dt + \ldots + \int_{t_{n-1}}^{t_n} \frac{d^m}{dt^m}\mathbf{x}(t)^2 dt \right), \quad m = 4 \tag{8.1}$$

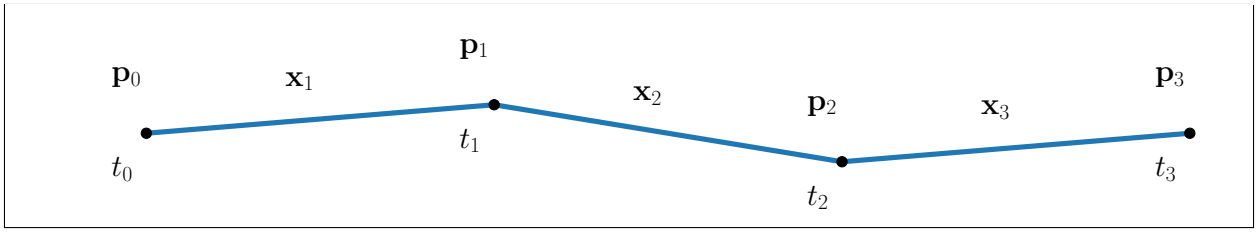

FIGURE 8.1
Piecewise-defined trajectory of first order polynomials. The trajectory consists of 3 segments passing through 4 points.

which has the piecewise solution

$$\mathbf{x}_{opt,d}(t) = \begin{cases} \sum_{i=0}^{2m-1} c_{0,i} \left(\frac{t-t_0}{t_1-t_0}\right)^i & t_0 \leq t < t_1 \\ \vdots & \\ \sum_{i=0}^{2m-1} c_{n-1,i} \left(\frac{t-t_{n-1}}{t_n-t_{n-1}}\right)^i & t_{n-1} \leq t < t_n \end{cases} \qquad \begin{array}{l} m = 4 \\ d \in \{x, y, z\} \end{array} \tag{8.2}$$

leading to $6nm$ coefficients to be determined. If we look now only at a single dimension we need $2nm$ constraint. The first set of constraints comes from the path having to pass through the n points. Each segment $\mathbf{x}_i$ needs to both start and end at a point giving $2n$ constraints. In addition to these constraints we demand that the UAV starts and ends smoothly by setting the first $m-1$ derivatives at the endpoints to zero. The remaining constraints we can get by demanding that the trajectory is sufficiently smooth across waypoints. These set of constraints are listed in tables 8.1 and 8.2.

TABLE 8.1
Start and end at waypoint

Constraint	For	Number of constraints
$\mathbf{x}_i(t_i) = p_i$	$i \in [1, n]$	n
$\mathbf{x}_i(t_{i-1}) = p_{i-1}$	$i \in [1, n]$	n

TABLE 8.2
Smooth start, stop and transitions

Constraint	For	Number of constraints
$\mathbf{x}_i(t_0)^{(j)} = 0$	$j \in [1, m-1]$	$m-1$
$\mathbf{x}_i(t_i)^{(j)} = \mathbf{x}_{i+1}(t_{i+1})^{(j)}$	$j \in [1, 2(m-1)]$, $i \in [1, n-1]$	$2(n-1)(m-1)$
$\mathbf{x}_i(t_n)^{(j)} = 0$	$j \in [1, m-1]$	$m-1$

As mentioned, we want to minimize the fourth derivative when constructing paths for multirotors, however it is still interesting to see the trajectories, and their second derivatives, resulting from minimizing lower order derivatives. This illustrates the dynamics the UAV would need to exhibit to follow them.

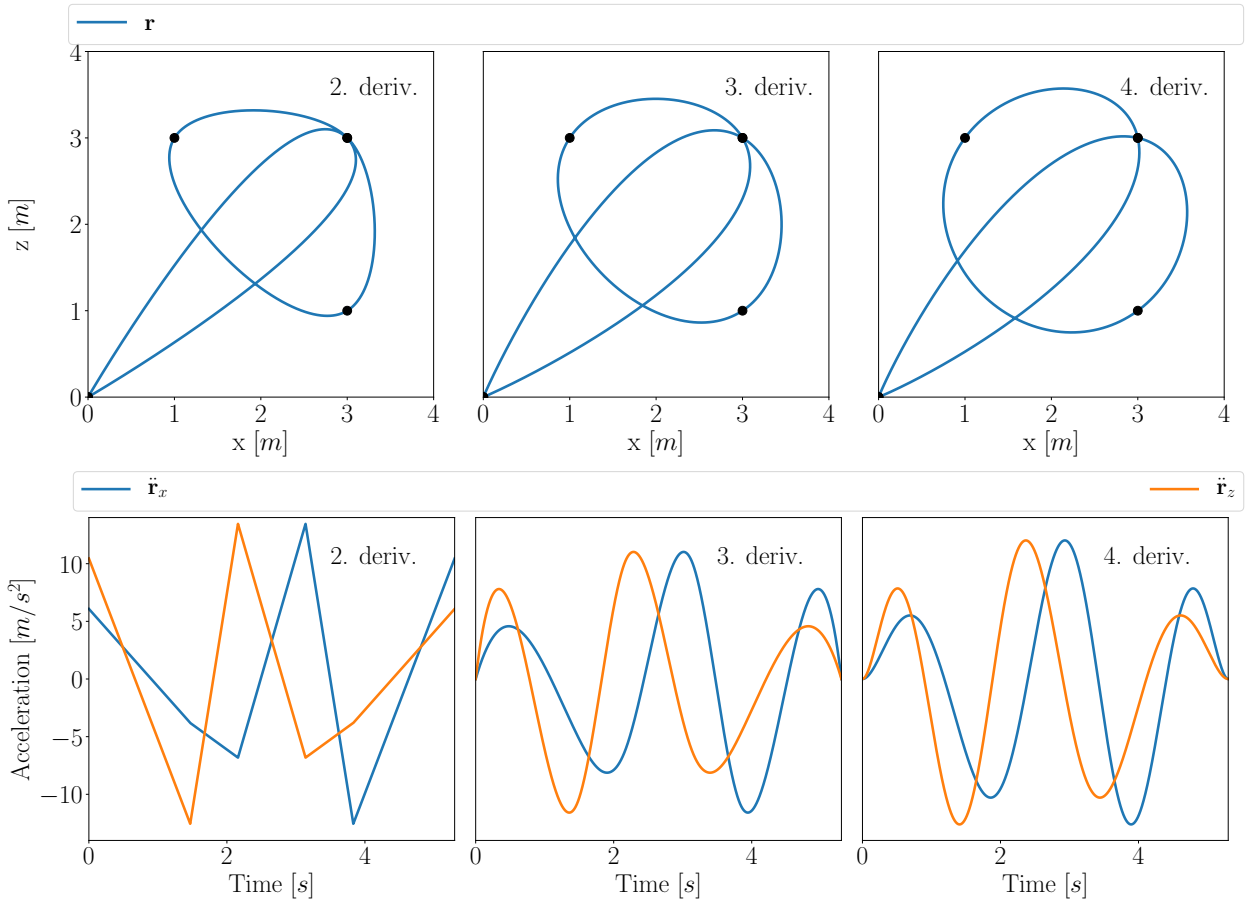

FIGURE 8.2
Top plots, left to right: trajectories resulting from minimizing the second, third and fourth order derivative, respectively. The trajectories start and end in the lower left corner. Bottom plots: second order derivatives of the generated trajectories.

When we have a path $\mathbf{x}(t)$ we still need to take gravity into account, so the desired force from the propellers must equal ${}^E\mathbf{f}_{thrust} = m({}^E\ddot{\mathbf{x}} + {}^E\mathbf{g})$. The direction of this vector is then the desired direction of the body z-axis.

A Python implementation of trajectory generation from a set of waypoints resulting from minimizing the n^{th} order derivative, where n is a parameter that can be passed to the path construction function, can be found at [23].

8.1.1 Orientation Discontinuities

The trajectory obtained by minimizing the fourth order derivative[1] as show in figure 8.2 does not guarantee that the UAV will be able to smoothly follow this path as intended. For some specific

[1]It is likely that this is the case for trajectories generated by minimizing any derivative of order $m > 2$ but no proof for this will be provided.

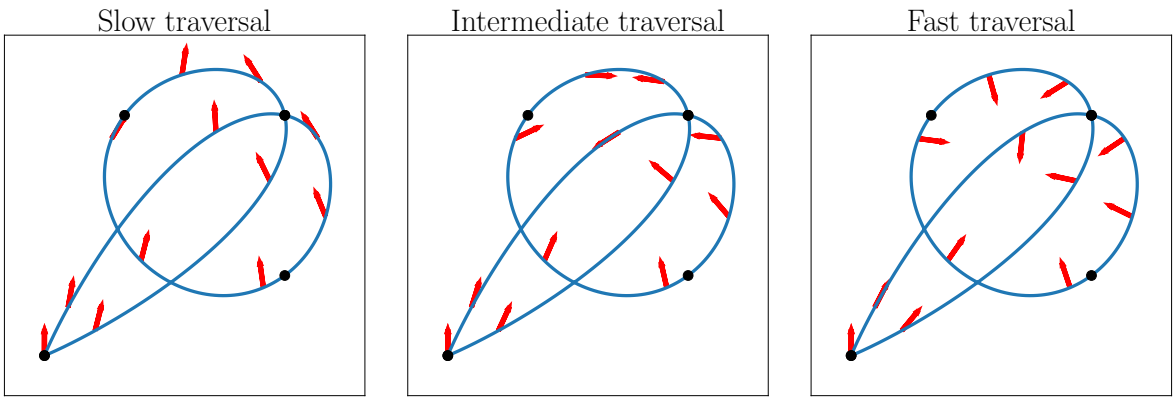

FIGURE 8.3
Illustration of a path in the xz-plane generated by minimizing the fourth order derivative. The red arrows indicate the direction of the desired acceleration at discrete moments along the path. For the slow traversal, gravity is dominating the direction of the desired acceleration, whereas it is the acceleration from the path-curvature that dominating in the fast traversal. For the intermediate traversal time, the direction of the desired acceleration flips as the path reaches the highest point. This poses a problem, as the UAV has to instantaneously flip as well to produce thrust in the right direction.

traversal times $T_{crit} = t_n - t_0$ the desired thrust vector passes through zero ${}^{E}\mathbf{f}_{thrust}(t) = (0, 0, 0)$. The problem with this is that the direction of acceleration flips as it passes through zero and thus the desired direction of the vehicle. This is illustrated in figure 8.3.

To overcome this problem some constraint other than just minimization of the m^{th} order derivative of position is needed. Investigating methods for obtaining trajectories that does not suffer from this problem is outside the scope of this thesis, but solutions might involve penalizing accelerations close to zero, or minimizing some derivative of orientation simultaneously.

8.2 Hover Controller

Control algorithms are a cornerstones in an autonomous robotics system as these facilitate the move from manipulating digital data to manipulating physical objects.
Most of the material in this section is based on material from [48, 45]. Control of the UAV is possible through the control of its motors and the speed they drive the propellers at. The thrust produced by the motors can be modeled as $f_i = k_t \Omega_i^2$ where k_t is some constant related to the size and shape of the propeller and Ω_i is the angular velocity of the motor. For a multirotor as the one depicted in figure 2.1 and sketched in figure 8.4 we can write the thrust and torque from the motors

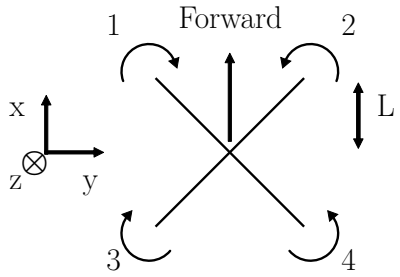

FIGURE 8.4
Visualization of direction of angular velocity as well as labeling of each motor. Note that two of the motors has an angular velocity along body z-axis and the two others opposite it. This allows the multirotor to spin around its z-axis by speeding up or slowing down pairs of propellers with angular velocity in the same direction.

as

$$ {}^{B}\mathbf{F} = \begin{bmatrix} 0 \\ 0 \\ F \end{bmatrix} \qquad F = \sum_{i=1}^{4} f_i \tag{8.3} $$

$$ {}^{B}\mathbf{M} = \begin{bmatrix} (\ \ f_1 - f_2 + f_3 - f_4)L \\ (\ \ f_1 + f_2 - f_3 - f_4)L \\ (-f_1 + f_2 - f_3 + f_4)\tilde{L}, \end{bmatrix} \tag{8.4} $$

where L is distance from the propellers to the axis of the UAV and $\tilde{L}$ is an effective arm length related to the drag of the propellers that cause a torque along the body z-axis. We can now write the linear and angular acceleration of the UAV as functions of the thrusts from each motor

$$ m^{E}\ddot{\mathbf{r}} = m^{E}\mathbf{g} + {}^{E}\mathbf{R}^{B}(\theta, \phi, \psi)^{B}\mathbf{F} \tag{8.5} $$

$$ {}^{B}\mathbf{I} \cdot {}^{B}\dot{\boldsymbol{\omega}}_1 = {}^{B}\mathbf{M} - {}^{B}\boldsymbol{\omega} \times ({}^{B}\mathbf{I} \cdot {}^{B}\boldsymbol{\omega}) \tag{8.6} $$

where the rotation matrix is parameterized by the Euler angles introduced in chapter 5, and the last term on the right hand side of the z-component of Euler rotation equation drops out, as $\mathbf{I}_{xx} = \mathbf{I}_{yy}$ due to the symmetry of the multirotor.
If we design the controller for a situation where the multirotor is close to horizontal at constant height and not rapidly spinning about its z-axis then we can make the following approximations

1. ${}^{E}\mathbf{F} \approx m^{E}\mathbf{g}$
2. $\Omega_i \approx \Omega_0 + \Delta\Omega_i$
3. $s\phi \approx \phi, s\theta \approx \theta, c\phi \approx c\theta \approx 1$
4. ${}^{B}\boldsymbol{\omega}_z \approx 0$

In this situation the above equations 8.5 and 8.6 reduce to

$$
\begin{aligned}
{}^{E}\ddot{\mathbf{r}}_x &\approx -g(\phi\, c\psi + \theta\, s\psi) && (8.7)\\
{}^{E}\ddot{\mathbf{r}}_y &\approx -g(\phi\, s\psi - \theta\, c\psi) && (8.8)\\
{}^{E}\ddot{\mathbf{r}}_z &\approx g - \frac{F}{m} && (8.9)\\
{}^{B}\dot{\boldsymbol{\omega}}_1 &\approx \frac{2k_tL(\quad \Delta\Omega_1 - \Delta\Omega_2 + \Delta\Omega_3 - \Delta\Omega_4)}{{}^{B}\mathbf{I}_{xx}} = \frac{8k_tL}{{}^{B}\mathbf{I}_{xx}}\Delta\Omega_x && (8.10)\\
{}^{B}\dot{\boldsymbol{\omega}}_2 &\approx \frac{2k_tL(\quad \Delta\Omega_1 + \Delta\Omega_2 - \Delta\Omega_3 - \Delta\Omega_4)}{{}^{B}\mathbf{I}_{yy}} = \frac{8k_tL}{{}^{B}\mathbf{I}_{yy}}\Delta\Omega_y && (8.11)\\
{}^{B}\dot{\boldsymbol{\omega}}_3 &\approx \frac{2k_t\tilde{L}(-\Delta\Omega_1 + \Delta\Omega_2 - \Delta\Omega_3 + \Delta\Omega_4)}{{}^{B}\mathbf{I}_{zz}} = \frac{8k_t\tilde{L}}{{}^{B}\mathbf{I}_{zz}}\Delta\Omega_z && (8.12)
\end{aligned}
$$

[45] use proportional and derivative (PD) terms to define the change in motor speeds from desired angles and angular velocities

$$
\begin{aligned}
\Delta\Omega_x &= k_{px}(\theta^{des} - \theta) + k_{dx}({}^{B}\boldsymbol{\omega}_1^{des} - {}^{B}\boldsymbol{\omega}_1) && (8.13)\\
\Delta\Omega_y &= k_{py}(\phi^{des} - \phi) + k_{dy}({}^{B}\boldsymbol{\omega}_2^{des} - {}^{B}\boldsymbol{\omega}_2) && (8.14)\\
\Delta\Omega_z &= k_{pz}(\psi^{des} - \psi) + k_{dz}({}^{B}\boldsymbol{\omega}_3^{des} - {}^{B}\boldsymbol{\omega}_3) && (8.15)
\end{aligned}
$$

where we have used that equation 5.14 reduces to the identity in the hover state. These equations can, together with equation 8.9, be used to determine the resulting motor speeds. The coefficients $k_{type,axis}$ decide the controller's behavior, i.e. how it reacts to deviations from the desired state. The controller will generally only be stable for a certain subset of the possible values of these coefficients. A position controller also described in [45], which will not be detailed here, is implemented in the simulations described in chapter 7. This controller is able to follow the trajectories generated using the derivative minimization method above.

9 Future Work

9.1 Improving IMU Based Estimates

Parts of this project, especially the orientation estimation methods, lack a better handling of uncertainties. Ideally the measurements that are used in the estimates should be weighted according to their uncertainty so as to not corrupt the estimates by blindly relying on very noisy and distorted data. Ideally, all estimates, produced by the filters, should by associated with an uncertainty at all times.

The best way of estimating the uncertainty of the orientation estimates using the current filters is not clear at the time of writing. However, as some of the methods listed as solutions to Wahba's problem do include uncertainty estimates, the remaining task consists only of determining how this uncertainty propagates through the estimation filters when combined with the angular rate measurements.

The Slerp filter's computationally and conceptually simple way of combining the angular rate based information with the field based information makes it a good candidate for further development towards obtaining these uncertainty estimates.

A different version of the direct quaternion methods exists, which weighs the two field observations differently, leading to improved orientation estimates by relying more on the measurement with the lower uncertainty [42]. Although this, for some applications, might be a sufficient solution for using measurements with different amounts of noise, it still does not solve the problem of the lacking uncertainty estimates on the final orientation estimate.

As shown in [37] and as has been confirmed during this work, the sensors used in the estimation tasks need to be well calibrated, for bias and other distortions, before accurate estimates can be obtained. As described, this can to some degree be done before the sensors are to be used or instead while they are being used. The estimation filters introduced in chapter 6 each used a method for continuously estimating the bias of the angular rate sensor. However, if estimated incorrectly, these bias estimates[1] will severely worsen the data. This trade off between possibly improving the orientation estimate by accurately determining the measurement model parameters versus risking completely inaccurate estimates due to erroneous parameter estimates is a problem. Some time should be devoted to designing methods allowing for continuous verification of the estimated calibration parameters. This task is not straightforward, as the available information is already being used to compute the calibration parameters, and at the same time is to be used for verification.

The additional angular rate sensor is likely to be a useful source of additional information that might ease the difficulties in this task.

[1] Generally this is the case for all the calibration parameters no matter how they are estimated.

9.2 Improved Test Framework

The convergence time test and the 2D parameter scan of the orientation estimation filters in chapter 6 have proven to be valuable tools for both debugging and performance testing. Extending and improving the testing framework will further improve productivity when developing orientation algorithms for the autopilot.

Improving the resemblance of the test framework to real world scenarios will improve the usefulness the results from tests. Such improvements will involve better modeling of the sensor characteristics, delays, disturbances from the environment, etc.

Additional tests of the Kalman filter is likely to also result in improvements and valuable debugging. Further development towards such a tests framework should be prioritizes.

9.3 Control Algorithms

In the few flight tests that has been carried out during this project, the stability and control of the platform did not match the intended behavior. The reason for this has still not been determined. The orientation estimation algorithm used at the time of the test was found to not be correctly implemented, so the problem might be fixed by using a proper implementation. Future test should indicate whether other problems played a role as well.

9.4 Vehicle Mounted Camera

During this project the idea arose to include information from a video stream provided by a camera mounted on the platform. Though some work has been made on this during the project, this did not reach a point where it was combined with the rest of the software. The reason that using a camera is of interest is that the video stream holds information provided by all other sensors. This means it can used to verify and improve the information used in both orientation and position estimation. From a downwards facing camera, information about translation and rotation along all axes of the UAV can be extracted from the stream, as these motions will lead to translation, divergence and rotation between frames of the stream. Determining the transformation from frame to frame can be used to extract information about what motions of the camera that led to this transformation. The task of determining the motion of a camera from specific changes between frames is called egomotion [36]. If included, this addition will extend the capabilities of the current implementation.

Another approach, also using a camera, is to build a full 3D map of the surroundings using the information in the video stream, while at the same time using this map to navigate through the environment. This approach termed **s**imultaneous **l**ocalization **a**nd **m**apping (SLAM) has not been a focus of this project but is being considered a possible future addition to the project.

10 Conclusion

The goal of this project was to acquire accurate and continuous estimates of orientation and position for use in autonomous flight of a UAV. In the process of achieving this goal, this thesis has demonstrated the dynamic noise characteristics of the accelerometer, angular rate sensor and magnetometer used on the autopilot board and the implications hereof have been discussed. Three methods for continuously estimating the angular rate sensor's bias while it is being used have been compared. It has been shown that all three methods are able to provide estimates of the bias leading to improved orientation estimates. Methods for estimating calibration parameters using specialized datasets collected before the use of the sensors have been suggested for all three sensor types.

The effectiveness of fusing measured or estimated quantities with their measured rate of change has been demonstrated. Specifically, the inaccuracy in orientation estimates obtained from integrating angular rate measurements has been shown to decrease, while the precision in orientation obtained from field measurement has been shown to increase when the two are fused.

The dependence on orientation in both position estimation and control algorithms has been shown, and the importance of accurate estimates of orientation has been discussion.

Important characteristics of various modules of the autopilot were revealed through thorough testing, allowing separate subproblems to be addressed in isolated scenarios before combining them to a complete solution.

In this thesis accurate state estimation methods have been developed, effective sensor calibration procedures tested, and a technique for generating trajectories as well as a control algorithm has been demonstrated. With these parts the foundation for autonomous flight of UAVs has been established.

A Euler Angle Sequences

All the following rotation matrices describe the body frame relative to some reference following the passive interpretation. For convenience we might call the reference frame E. The first matrix with the notation used throughout this document will thus be ${}^{B}R^{E}_{xyx}$ and thus ${}^{B}r = {}^{B}R^{E}_{xyx}\,{}^{E}r$

$$R_{xyx} = R_x(\theta_3)R_y(\theta_2)R_x(\theta_1) = \begin{bmatrix} c\theta_2 & s\theta_1\, s\theta_2 & -c\theta_1\, s\theta_2 \\ s\theta_2\, s\theta_3 & c\theta_1\, c\theta_3 - c\theta_2\, s\theta_1\, s\theta_3 & c\theta_1\, c\theta_2\, s\theta_3 + c\theta_3\, s\theta_1 \\ c\theta_3\, s\theta_2 & -c\theta_1\, s\theta_3 - c\theta_2\, c\theta_3\, s\theta_1 & c\theta_1\, c\theta_2\, c\theta_3 - s\theta_1\, s\theta_3 \end{bmatrix} \tag{A.1}$$

$$R_{xyz} = R_x(\theta_3)R_y(\theta_2)R_z(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_2 & c\theta_2\, s\theta_1 & -s\theta_2 \\ c\theta_1\, s\theta_2\, s\theta_3 - c\theta_3\, s\theta_1 & c\theta_1\, c\theta_3 + s\theta_1\, s\theta_2\, s\theta_3 & c\theta_2\, s\theta_3 \\ c\theta_1\, c\theta_3\, s\theta_2 + s\theta_1\, s\theta_3 & -c\theta_1\, s\theta_3 + c\theta_3\, s\theta_1\, s\theta_2 & c\theta_2\, c\theta_3 \end{bmatrix} \tag{A.2}$$

$$R_{xzx} = R_x(\theta_3)R_z(\theta_2)R_x(\theta_1) = \begin{bmatrix} c\theta_2 & c\theta_1\, s\theta_2 & s\theta_1\, s\theta_2 \\ -c\theta_3\, s\theta_2 & c\theta_1\, c\theta_2\, c\theta_3 - s\theta_1\, s\theta_3 & c\theta_1\, s\theta_3 + c\theta_2\, c\theta_3\, s\theta_1 \\ s\theta_2\, s\theta_3 & -c\theta_1\, c\theta_2\, s\theta_3 - c\theta_3\, s\theta_1 & c\theta_1\, c\theta_3 - c\theta_2\, s\theta_1\, s\theta_3 \end{bmatrix} \tag{A.3}$$

$$R_{xzy} = R_x(\theta_3)R_z(\theta_2)R_y(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_2 & s\theta_2 & -c\theta_2\, s\theta_1 \\ -c\theta_1\, c\theta_3\, s\theta_2 + s\theta_1\, s\theta_3 & c\theta_2\, c\theta_3 & c\theta_1\, s\theta_3 + c\theta_3\, s\theta_1\, s\theta_2 \\ c\theta_1\, s\theta_2\, s\theta_3 + c\theta_3\, s\theta_1 & -c\theta_2\, s\theta_3 & c\theta_1\, c\theta_3 - s\theta_1\, s\theta_2\, s\theta_3 \end{bmatrix} \tag{A.4}$$

$$R_{yxy} = R_y(\theta_3)R_x(\theta_2)R_y(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_3 - c\theta_2\, s\theta_1\, s\theta_3 & s\theta_2\, s\theta_3 & -c\theta_1\, c\theta_2\, s\theta_3 - c\theta_3\, s\theta_1 \\ s\theta_1\, s\theta_2 & c\theta_2 & c\theta_1\, s\theta_2 \\ c\theta_1\, s\theta_3 + c\theta_2\, c\theta_3\, s\theta_1 & -c\theta_3\, s\theta_2 & c\theta_1\, c\theta_2\, c\theta_3 - s\theta_1\, s\theta_3 \end{bmatrix} \tag{A.5}$$

$$R_{yxz} = R_y(\theta_3)R_x(\theta_2)R_z(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_3 - s\theta_1\, s\theta_2\, s\theta_3 & c\theta_1\, s\theta_2\, s\theta_3 + c\theta_3\, s\theta_1 & -c\theta_2\, s\theta_3 \\ -c\theta_2\, s\theta_1 & c\theta_1\, c\theta_2 & s\theta_2 \\ c\theta_1\, s\theta_3 + c\theta_3\, s\theta_1\, s\theta_2 & -c\theta_1\, c\theta_3\, s\theta_2 + s\theta_1\, s\theta_3 & c\theta_2\, c\theta_3 \end{bmatrix} \tag{A.6}$$

$$R_{yzx} = R_y(\theta_3)R_z(\theta_2)R_x(\theta_1) = \begin{bmatrix} c\theta_2\, c\theta_3 & c\theta_1\, c\theta_3\, s\theta_2 + s\theta_1\, s\theta_3 & -c\theta_1\, s\theta_3 + c\theta_3\, s\theta_1\, s\theta_2 \\ -s\theta_2 & c\theta_1\, c\theta_2 & c\theta_2\, s\theta_1 \\ c\theta_2\, s\theta_3 & c\theta_1\, s\theta_2\, s\theta_3 - c\theta_3\, s\theta_1 & c\theta_1\, c\theta_3 + s\theta_1\, s\theta_2\, s\theta_3 \end{bmatrix} \tag{A.7}$$

$$R_{yzy} = R_y(\theta_3)R_z(\theta_2)R_y(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_2\, c\theta_3 - s\theta_1\, s\theta_3 & c\theta_3\, s\theta_2 & -c\theta_1\, s\theta_3 - c\theta_2\, c\theta_3\, s\theta_1 \\ -c\theta_1\, s\theta_2 & c\theta_2 & s\theta_1\, s\theta_2 \\ c\theta_1\, c\theta_2\, s\theta_3 + c\theta_3\, s\theta_1 & s\theta_2\, s\theta_3 & c\theta_1\, c\theta_3 - c\theta_2\, s\theta_1\, s\theta_3 \end{bmatrix} \tag{A.8}$$

$$R_{zxy} = R_z(\theta_3)R_x(\theta_2)R_y(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_3 + s\theta_1\, s\theta_2\, s\theta_3 & c\theta_2\, s\theta_3 & c\theta_1\, s\theta_2\, s\theta_3 - c\theta_3\, s\theta_1 \\ -c\theta_1\, s\theta_3 + c\theta_3\, s\theta_1\, s\theta_2 & c\theta_2\, c\theta_3 & c\theta_1\, c\theta_3\, s\theta_2 + s\theta_1\, s\theta_3 \\ c\theta_2\, s\theta_1 & -s\theta_2 & c\theta_1\, c\theta_2 \end{bmatrix} \tag{A.9}$$

$$R_{zxz} = R_z(\theta_3)R_x(\theta_2)R_z(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_3 - c\theta_2\, s\theta_1\, s\theta_3 & c\theta_1\, c\theta_2\, s\theta_3 + c\theta_3\, s\theta_1 & s\theta_2\, s\theta_3 \\ -c\theta_1\, s\theta_3 - c\theta_2\, c\theta_3\, s\theta_1 & c\theta_1\, c\theta_2\, c\theta_3 - s\theta_1\, s\theta_3 & c\theta_3\, s\theta_2 \\ s\theta_1\, s\theta_2 & -c\theta_1\, s\theta_2 & c\theta_2 \end{bmatrix} \tag{A.10}$$

$$R_{zyx} = R_z(\theta_3)R_y(\theta_2)R_x(\theta_1) = \begin{bmatrix} c\theta_2\, c\theta_3 & c\theta_1\, s\theta_3 + c\theta_3\, s\theta_1\, s\theta_2 & -c\theta_1\, c\theta_3\, s\theta_2 + s\theta_1\, s\theta_3 \\ -c\theta_2\, s\theta_3 & c\theta_1\, c\theta_3 - s\theta_1\, s\theta_2\, s\theta_3 & c\theta_1\, s\theta_2\, s\theta_3 + c\theta_3\, s\theta_1 \\ s\theta_2 & -c\theta_2\, s\theta_1 & c\theta_1\, c\theta_2 \end{bmatrix} \tag{A.11}$$

$$R_{zyz} = R_z(\theta_3)R_y(\theta_2)R_z(\theta_1) = \begin{bmatrix} c\theta_1\, c\theta_2\, c\theta_3 - s\theta_1\, s\theta_3 & c\theta_1\, s\theta_3 + c\theta_2\, c\theta_3\, s\theta_1 & -c\theta_3\, s\theta_2 \\ -c\theta_1\, c\theta_2\, s\theta_3 - c\theta_3\, s\theta_1 & c\theta_1\, c\theta_3 - c\theta_2\, s\theta_1\, s\theta_3 & s\theta_2\, s\theta_3 \\ c\theta_1\, s\theta_2 & s\theta_1\, s\theta_2 & c\theta_2 \end{bmatrix} \tag{A.12}$$

B Kalman Filter

Kalman Filter Matrices

$$\mathbf{A} = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_{00} & -R_{01} & -R_{02} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_{10} & -R_{11} & -R_{12} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -R_{20} & -R_{21} & -R_{22} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{B.1}$$

$$\mathbf{B} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ R_{00} & R_{01} & R_{02} & 0 \\ R_{10} & R_{11} & R_{12} & 0 \\ R_{20} & R_{21} & R_{22} & -1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{B.2}$$

$$\mathbf{C} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \tag{B.3}$$

The following approximations has been used in calculating $\mathbf{F}$ and $\mathbf{G}$

$$\mathbf{F} = e^{\mathbf{A}T} \approx \mathbf{I} + \mathbf{A}T \tag{B.4}$$

$$\mathbf{G} = \int_0^{T_s} e^{\mathbf{A}t}\mathbf{B}dt \approx \int_0^{T_s} (\mathbf{I} + \mathbf{A}t)\mathbf{B}dt \tag{B.5}$$

$$= \mathbf{B}T_s + \frac{1}{2}\mathbf{A}\mathbf{B}T_s^2 \tag{B.6}$$

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & T & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & T & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -R_{00}T & -R_{01}T & -R_{02}T \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -R_{10}T & -R_{11}T & -R_{12}T \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -R_{20}T & -R_{21}T & -R_{22}T \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{B.7}$$

$$\mathbf{G} = \begin{pmatrix} \frac{1}{2}T_s^2R_{00} & \frac{1}{2}T_s^2R_{01} & \frac{1}{2}T_s^2R_{02} & 0 \\ \frac{1}{2}T_s^2R_{10} & \frac{1}{2}T_s^2R_{11} & \frac{1}{2}T_s^2R_{12} & 0 \\ \frac{1}{2}T_s^2R_{20} & \frac{1}{2}T_s^2R_{21} & \frac{1}{2}T_s^2R_{22} & -\frac{1}{2}T_s^2 \\ T_sR_{00} & T_sR_{01} & T_sR_{02} & 0 \\ T_sR_{10} & T_sR_{11} & T_sR_{12} & 0 \\ T_sR_{20} & T_sR_{21} & T_sR_{22} & -T_s \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \tag{B.8}$$

If using that $R_{ij} \leq 1$ and assuming that $n_v = n_{vx} = n_{vy} = n_{vz}$ we get:

$$\mathbf{Q} \approx B_v V_1 B_v^T Ts = Ts \begin{pmatrix} n_{px} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & n_{py} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & n_{pz} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3n_v & 3n_v & 3n_v & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3n_v & 3n_v & 3n_v & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3n_v & 3n_v & 3n_v & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & n_{bar} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{acc} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{acc} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & n_{acc} \end{pmatrix} \tag{B.9}$$

where n is for process noise.

$$\mathbf{R} \approx V_2 T^{-1} = \begin{pmatrix} \frac{\sigma_{GpsP}}{T_{Gps}} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\sigma_{GpsP}}{T_{Gps}} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\sigma_{GpsP}}{T_{Gps}} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{\sigma_{GpsV}}{T_{Gps}} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\sigma_{GpsV}}{T_{Gps}} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\sigma_{GpsV}}{T_{Gps}} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\sigma_{bar}}{T_{bar}} \end{pmatrix} \tag{B.10}$$

C Supplementary calculations

Madgwick Filter

Assuming that what Madgwick means by minimizing the vector difference between the observations and the result of rotating the reference vectors be the estimated quaternion is to minimize the norm, or squared norm of this difference. In that case we get the following.
Using $q = {}^{B}q^{E}$ in all the following

$$\mathbf{f_r}(q, {}^{E}\mathbf{r}, {}^{B}\mathbf{r}) = \mathrm{Im}(q \otimes (0, {}^{E}\mathbf{r}) \otimes q^*) - {}^{B}\mathbf{r} \tag{C.1}$$

$$e(q, {}^{E}\mathbf{r}, {}^{B}\mathbf{r}) = \mathbf{f_r} \cdot \mathbf{f_r} \tag{C.2}$$

$$\frac{\partial e}{\partial q_j} = \sum_i \left(\frac{\partial \mathbf{f}_{\mathbf{r},i}}{\partial q_j} \mathbf{f}_{\mathbf{r},i} + \mathbf{f}_{\mathbf{r},i} \frac{\partial \mathbf{f}_{\mathbf{r},i}}{\partial q_j} \right) \tag{C.3}$$

$$\frac{\partial e}{\partial q_j} = 2 \sum_i \frac{\partial \mathbf{f}_{\mathbf{r},i}}{\partial q_j} \mathbf{f}_{\mathbf{r},i} \tag{C.4}$$

$$= 2(\mathbf{J}_{\mathbf{r}}^T \mathbf{f_r})_j \tag{C.5}$$

$$\mathbf{J}_{\mathbf{r}}^T \mathbf{f_r} = \nabla \mathbf{f_r} \tag{C.6}$$

$$\mathbf{J}_{\mathbf{g}}^T \mathbf{f_g} + \mathbf{J}_{\mathbf{b}}^T \mathbf{f_b} = \nabla \mathbf{f} \tag{C.7}$$

Acceleration from constant thrust

Accelration in the body frame at constant thrust as the 2d UAV is tilted by some angle theta.

$${}^{B}\ddot{\mathbf{r}} = {}^{B}\mathbf{R}^{E}({}^{E}\ddot{\mathbf{r}} + {}^{E}\mathbf{g}) \tag{C.8}$$

$$= \begin{bmatrix} c(\theta) & s(\theta) \\ -s(\theta) & c(\theta) \end{bmatrix} \left(\begin{bmatrix} -s(\theta)|\mathbf{g}| \\ (c(\theta) - 1)|\mathbf{g}| \end{bmatrix} + \begin{bmatrix} 0 \\ |\mathbf{g}| \end{bmatrix} \right) \tag{C.9}$$

$$= \begin{bmatrix} (c(\theta)s(\theta) - c(\theta)s(\theta) - s(\theta))|\mathbf{g}| \\ (s(\theta)s(\theta) + c(\theta)c(\theta) - c(\theta))|\mathbf{g}| \end{bmatrix} + \begin{bmatrix} s(\theta)|\mathbf{g}| \\ c(\theta)|\mathbf{g}| \end{bmatrix} \tag{C.10}$$

$$= \begin{bmatrix} 0 \\ |\mathbf{g}| \end{bmatrix} \tag{C.11}$$

And for increased thrust to maintain height thrust $= \tilde{\mathbf{g}} = |\mathbf{g}|/\cos(\theta)$

$$
\begin{aligned}
{}^{B}\ddot{\mathbf{r}} &= {}^{B}\mathbf{R}^{E}({}^{E}\ddot{\mathbf{r}} + {}^{E}\mathbf{g}) && \text{(C.12)}\\
&= \begin{bmatrix} c(\theta) & s(\theta) \\ -s(\theta) & c(\theta) \end{bmatrix} \left(\begin{bmatrix} -s(\theta)|\tilde{\mathbf{g}}| \\ c(\theta)\tilde{\mathbf{g}} - \mathbf{g} \end{bmatrix} + \begin{bmatrix} 0 \\ |\mathbf{g}| \end{bmatrix} \right) && \text{(C.13)}\\
&= \begin{bmatrix} c(\theta) & s(\theta) \\ -s(\theta) & c(\theta) \end{bmatrix} \left(\begin{bmatrix} -s(\theta)|\tilde{\mathbf{g}}| \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ |\mathbf{g}| \end{bmatrix} \right) && \text{(C.14)}\\
&= \begin{bmatrix} -c(\theta)s(\theta)|\tilde{\mathbf{g}}| \\ +s(\theta)s(\theta)|\tilde{\mathbf{g}}| \end{bmatrix} + \begin{bmatrix} s(\theta)|\mathbf{g}| \\ c(\theta)|\mathbf{g}| \end{bmatrix} && \text{(C.15)}\\
&= \begin{bmatrix} 0 \\ (s(\theta)\tan(\theta) + c(\theta))|\mathbf{g}| \end{bmatrix} && \text{(C.16)}\\
&= \begin{bmatrix} 0 \\ |\mathbf{g}|/c(\theta) \end{bmatrix} && \text{(C.17)}\\
&= \begin{bmatrix} 0 \\ |\tilde{\mathbf{g}}| \end{bmatrix} && \text{(C.18)}
\end{aligned}
$$

Bibliography

[1] URL: https://github.com/erikbrntsn/.

[2] URL: https://www.invensense.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf.

[3] URL: http://www.digikey.com/product-detail/en/stmicroelectronics/LSM6DS3TR/497-15383-1-ND/5180534.

[4] URL: http://www.st.com/en/mems-and-sensors/lis3mdl.html.

[5] URL: https://www.bosch-sensortec.com/bst/products/all_products/bmp280.

[6] URL: http://www.locosystech.com/product.php?zln=en&id=20.

[7] URL: http://www.robotshop.com/en/lidar-lite-2-laser-rangefinder-pulsedlight.html..

[8] URL: https://www.ngdc.noaa.gov/geomag/WMM/data/WMM2015/WMM2015_F_MERC.pdf.

[9] URL: https://www.ngdc.noaa.gov/geomag/WMM/data/WMM2015/WMM2015_I_MERC.pdf.

[10] URL: https://www.ngdc.noaa.gov/geomag/WMM/data/WMM2015/WMM2015_D_MERC.pdf.

[11] URL: http://www.st.com/en/microcontrollers/stm32f745vg.html.

[12] URL: http://www.chibios.org/dokuwiki/doku.php.

[13] URL: https://github.com/erikbrntsn/Simple-Matrix-Lib.

[14] URL: https://www.raspberrypi.org/products/.

[15] URL: https://github.com/erikbrntsn/python-live-plotting-from-data-stream.

[16] URL: https://en.wikipedia.org/wiki/File:Geoida.svg.

[17] URL: vedder.se.

[18] URL: https://github.com/vedderb/QuadcopterSystem.

[19] URL: https://commons.wikimedia.org/wiki/File:Yaw_Axis_Corrected.svg.

[20] URL: https://github.com/erikbrntsn/Orientation-Estimation.

[21] URL: https://github.com/erikbrntsn/Orientation-Estimation-cpp.

[22] URL: http://matplotlib.org/.

[23] URL: https://github.com/erikbrntsn/UAVsimulation.

[24] URL: https://github.com/erikbrntsn/uav-sim-cpp.

[25] URL: https://www.youtube.com/watch?v=1n-HMSCDYtM.

[26] Guowei Cai, Ben M. Chen, and Tong Heng Lee. *Unmanned Rotorcraft Systems*. Springer, 2011.

[27] Erik B. Dam, Martin Koch, and Martin Lillholm. *Quaternions, Interpolation and Animation.* Technical Report. DIKU, 1998.

[28] Lester Eli Dubins. "On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents". In: *On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents* (1957).

[29] David Eberly. *Distance from a Point to an Ellipse, an Ellipsoid, or a Hyperellipsoid.* Tech. rep. Geometric Tools, LLC, 2013.

[30] David Eberly. *Least Squares Fitting of Data.* Tech. rep. Geometric Tools, LLC, 2015.

[31] Herbert Goldstein, Charles Poole, and John Safko. *Classical Mechanics.* Third Edition. Pearson Education Limited, 2002.

[32] Basile Graf. "Quaternions And Dynamics". In: (2007).

[33] *"History of the Prime Meridian -Past and Present".* URL: `http://gpsinformation.net/main/greenwich.htm`.

[34] ISO. *Standard representation of geographic point location by coordinates.* URL: `https://www.iso.org/obp/ui/#iso:std:iso:6709:ed-2:v1:en`.

[35] Christopher Konvalin. *Motion/Velocity/Displacement Compensating for Tilt, Hard-Iron, and Soft-Iron Effects.* URL: `http://www.sensorsmag.com/sensors/motion-velocity-displacement/compensating-tilt-hard-iron-and-soft-iron-effects-6475`.

[36] Yi Ma, Stefano Soatto, Jana Kosecka, and S. Shankar Sastry. *An Invitation to 3-D Vision.* Hardcover First Edition 2004. Springer-verlag New York Inc., 2004.

[37] Sebastian O.H. Madgwick. "AHRS algorithms and calibration solutions to facilitate new applications using low-cost MEMS". PhD thesis. University of Bristol, 2014.

[38] Sebastian O.H. Madgwick. "An effcient orientation filter for inertial and inertial/magnetic sensor arrays". In: (2010).

[39] *magnetometer.* URL: `http://www.vectornav.com/support/library/magnetometer`.

[40] R. Mahony, T. Hamel, and J. Pflimlin. "Complementary filter design on the special orthogonal group SO(3)". In: *Proceedings of the 44th IEEE Conference on Decision and Control* (2005).

[41] F. L. Markley and D Mortari. "How To Estimate Attitude From Vector Observations". In: *Journal of the Astronautical Sciences* (1999).

[42] F. Landis Markley. "ATTITUDE DETERMINATION USING TWO VECTOR MEASUREMENTS". In: (1998).

[43] F. Landis Markley. "Attitude Determination using Vector Observations and the Singular Value Decomposition". In: *The Journal of the Astronautical Sciences, Vol 36, No. 3, July-September 1988, pp. 245-258* (1988).

[44] F. Landis Markley and Daniele Mortari. "QUATERNION ATTITUDE ESTIMATION USING VECTOR OBSERVATIONS". In: *Journal of the Astronautical Sciences* (2000).

[45] Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. "The GRASP Multiple Micro UAV Testbed". In: *IEEE Robotics and Automation Magazine* (2010).

[46] *National Geospatial-Intelligence Agency*. URL: http://earth-info.nga.mil/GandG/images/ww15mgh2.gif.

[47] *National Ocean Service*. URL: http://oceanservice.noaa.gov/facts/geoid.html.

[48] University of Pennsylvania. *Robotics: Aerial Robotics*. 2016.

[49] William Premerlani and Paul Bizard. "Direction Cosine Matrix IMU: Theory". 2009.

[50] D. Rose. *Converting between Earth-Centered, Earth Fixed and Geodetic Coordinates*. 2014. URL: http://danceswithcode.net/engineeringnotes/geodetic_to_ecef/geodetic_to_ecef.html.

[51] Ken Shoemake. "Animating Rotation with Quaternion Curves". In: (1985).

[52] Dan Simon. *Optimal State Estimation*. John Wiley and Sons Inc, 2006.

[53] D. A. Turner, I. J. Anderson, J. C. Mason, and M. G. Cox. *An algorithm for tting an ellipsoid to data*. Tech. rep. The University of Huddersfield and National Physical Laboratory, Teddington, 1999.

[54] G. Wahba. "A least squares estimate of satellite attitude". In: *Society for Industrial and Applied Mathematics* (1965).

[55] Eric W. Weisstein. *Alibi Transformation*. URL: http://mathworld.wolfram.com/AlibiTransformation.html.

[56] Bong Wie. 2nd Edition. American Institute of Aeronautics and Astronautics, 2008.