# Attitude determination of a relative position sensor using multi-sensor fusion

TTK4900 - Engineering Cybernetics, Master's Thesis

Martin Tysseland

Department of Engineering Cybernetics
Norwegian University of Science and Technology
January 2022

# *Summary*

This master's thesis contains insight into how to determinant the attitude of a circuit board mounted on a rig. Measurements from 9-axes IMU and GNSS are used, combined with ESKF to estimate the attitude in the Euler angles roll, pitch and yaw. The results shows accuracy from heading correction with relative position from two GNSS antennas, correction with magnetometer measurements and the difference between the solutions. The accelerometer is used to correct the roll and pitch. The GNSS correction works great when getting valid signals, and magnetometer correction needs good calibrating before being usable, and can still be victim of unwanted magnetic interference.

*Denne masteroppgaven gir en innsikt i hvordan man kan bestemme orientering av et kretskort montert på et stativ. Målinger fra et 9-akse IMU og GNSS er brukt. Dette er kombinert med et ESKF for å estimere orienteringen i Euler-vinklene rull, tilt og himmelretning. Resultatene viser nøyaktighet for himmelretning med relativ posisjon fra to GNSS-antenner. De viser også forskjellen i himmelretning når magnetometer og GNSS brukes for korreksjon. Det viser seg at GNSS korreksjon fungerer bra når man har gyldig signal, og korreksjon med hjelp av magnetometer krever veldig god kalibrering for å være brukbar, men er fortsatt utsatt for magnetiske forstyrrelser.*

# *Preface*

I want to thank my supervisor, Torleiv H. Bryne from NTNU, for helping me with this master's thesis. A lot of work have been done for this setup to work, from building the rig, writing working code and find the correct hardware connections. I feel the contents in this master's thesis does not as accurate as wanted reflect the many hours that has been used. Lacking some noticeable depth in different sections, but over all I have learned a lot about getting real life application to work outside in the field. I also want to thank Squarehead Technology for supporting with hardware and office room.

This master's thesis is a continuation of the work done in my specialization project *Attitude determination of a relative position sensor* (Tysseland, 2021). Some section from this projected is reused with none, minor or large changes. The sections affected by this are Section 1.1, 1.2, 2.1, 3.1, 3.2, 3.4, 3.5, 4.2, 5.1, 5.2, 5.3 and 5.4.

**NTNU**
**Norwegian University of**
**Science and Technology**

**Faculty of Information Technology**
**and Electrical Engineering**
**Department of Engineering Cybernetics**

# MSc THESIS DESCRIPTION SHEET

| | |
|---|---|
| **Name:** | Martin Tysseland |
| **Department:** | Engineering Cybernetics |
| **Thesis title (Norwegian):** | Orienteringsbestemmelse av en relativ posisjoneringssensor ved bruk av multi-sensor integrasjon |
| **Thesis title (English):** | Attitude determination of a relative position sensor using multi-sensor fusion |

**Thesis Description:** In object detection applications, e.g., using electro optical sensors (cameras) or microphone arrays, object position or direction (line of bearing) is only known relative to the sensor. To use such information for absolute localization/positioning of an object depends on knowing the sensors location as well as position and orientation versus some reference coordinate system. With the availability of low-cost combined accelerometer / gyroscope / magnetometer chips, so called 9-DOF inertial measurement units (IMUs) a compact and low footprint sensor has the potential to be integrated with many other types of sensor systems. The availability of cost-effective GNSS solutions capable of proving heading measurement are particularly interesting due to such system heigh accuracy.

The following tasks should be considered:

1. Perform a literature study of calibration techniques for MEMS magnetometers, including error modelling of scale and bias error, sensor misalignment and influence of hard and soft iron materials.
2. Determine the practical attitude estimation error introduced by using a combination of accelerometer and magnetometer data in different environments and geographical locations, varying temperatures, and possible drift over time. In particular the heading/yaw attitude error is of interest.
3. Integrate GNSS receivers in a multi-sensor fusion system capable of providing attitude/heading measurements and compare the attitude estimates to those achieved when using magnetometer as a heading sensor.
4. Review effect of magnetic declination and identify means of obtaining a decent estimate of declination for a given a latitude and longitude in a stand-alone offline configuration.
5. If time permits review techniques for magnetometer calibration, including identifying possible “field friendly” techniques minimizing dependency on calibration rigs or specialized user training.
6. Present your results and discuss these.
7. Conclude your results and suggest further work.

| | |
|---|---|
| **Start date:** | 2021-08-09 |
| **Due date:** | 2022-01-10 |
| **Thesis performed at:** | Department of Engineering Cybernetics, NTNU |
| **Supervisor:** | Associate professor Torleiv H. Bryne, Dept. of Eng. Cybernetics, NTNU |

# *Contents*

# 1 Introduction

## 1.1 Project

This master's thesis aims to compute the attitude of a circuit board based on sensor readings from an accelerometer, gyroscope, magnetometer, and GNSS. The gyroscope outputs from the IMU will be integrated through a kinematic model to provide an estimate of the attitude represented as a unit quaternion. The unit quaternion will provide a 4-dimensional attitude representation that will avoid singularities. When describing the attitude in the results, the quaternion will be converted to the triple Euler angles roll, pitch, and yaw for better readability.

Further, an ESKF (error state Kalman filter) will estimate and compensate for bias and filter the noisy measurements. Acceleration, magnetometer and relative position from GNSS measurements will, among other things, be used for this correction. The singularity-free unit quaternion fits well in the state vector in the ESKF. The state vector also contains the gyroscope bias.

The area of usage of a accurate attitude determination could be to accurate display discovered objects in a 2D or 3D map.

## 1.2 Circuit board

Squarehead Technology provides hardware for testing that is the same as what they use in their products. The hardware is a circuit board composed of different audio and video processing features and access to mounted IMUs. The only feature that is of interest by now is the mounted IMUs. The circuit board is shown in Figure 1.1. Because the true look of the complete circuit board is classified, the figure shows only a drawing. There are five mounted IMUs, and their location is highlighted in the figure as a black square with white filling and a black dot in the left lower corner. Four IMUs are located in all corners of the circuit board, and the fifth is located in the middle. All IMUs are operative and gather data, but only the data from the IMU in the middle is used when estimating attitude for the circuit board. For power and data transfer, a 1-meter long USB-C to USB-A cable is used. This cable is connected to a laptop running on Ubuntu 20.04 LTS operative system.

The orientation shown in Figure 1.1 represents the idle orientation of the circuit board. In this orientation roll, pitch and yaw are expected to be zero. The coordinate system for the circuit board is shown in Figure 1.2. Movement about the x-axis will represent changes in roll, movement about the y-axis will represent changes in pitch, and movement about the z-axis will represent changes in yaw. The local coordinate system for the IMUs was not originally aligned with the chosen coordinate system shown in Figure 1.2. This was adjusted by rotating the IMUs measurements vector with a rotation matrix. The method used is further explained in Section 5.2.

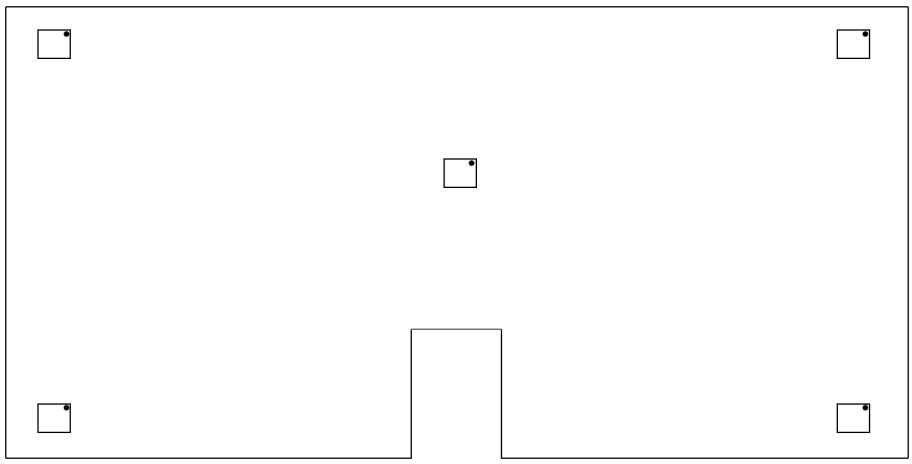


Figure 1.1: A complete drawing of the circuit board provided by Squarehead Technology. Only the IMU located in the middle was used when estimating the results. The dot shown on the IMUs indicates the mounted rotation of the IMU relative to the circuit board.

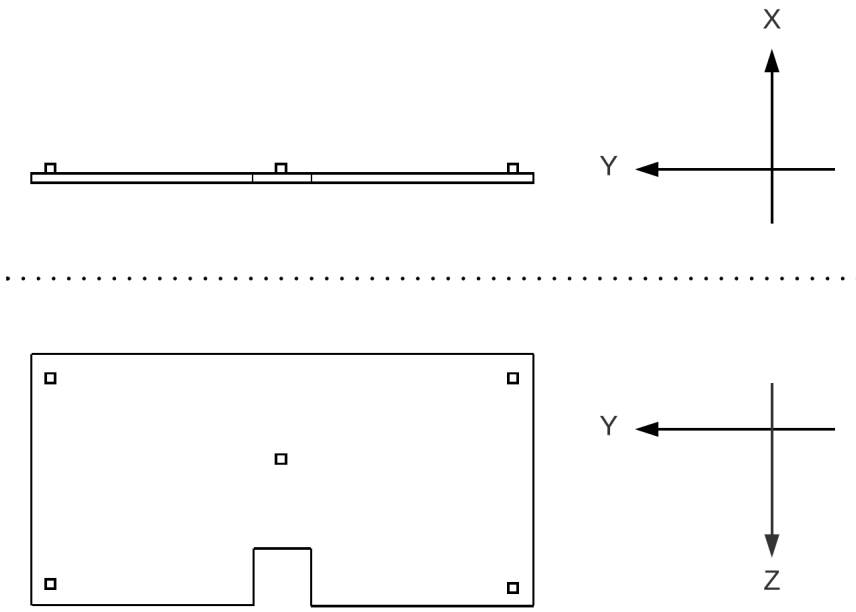



Figure 1.2: The coordinate system that is used for the circuit board. It shows the direction for the x-axis, the y-axis, and the z-axis.

## 1.3 Rig setup

Build a rig in aluminium with parts from RatRig to mount every thing aligned. The base is places to the left of the circuit board and the rover is places to the right. The calculated heading from GNSS will always be from the base to the rover. So to get the correct heading of the circuit board the GNSS heading has to be rotated 90 degrees around the positive z-axis. This could be done with the rotation matrix:

$$\boldsymbol{R} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.1}$$

The upper part of the rig is shown in Figure 1.3 and shows how the circuit board and antennas are mounted. The circuit board appearance is covered by paper because of company confidentiality policy. The circuit board appearance is illustrated in Figure 1.1.

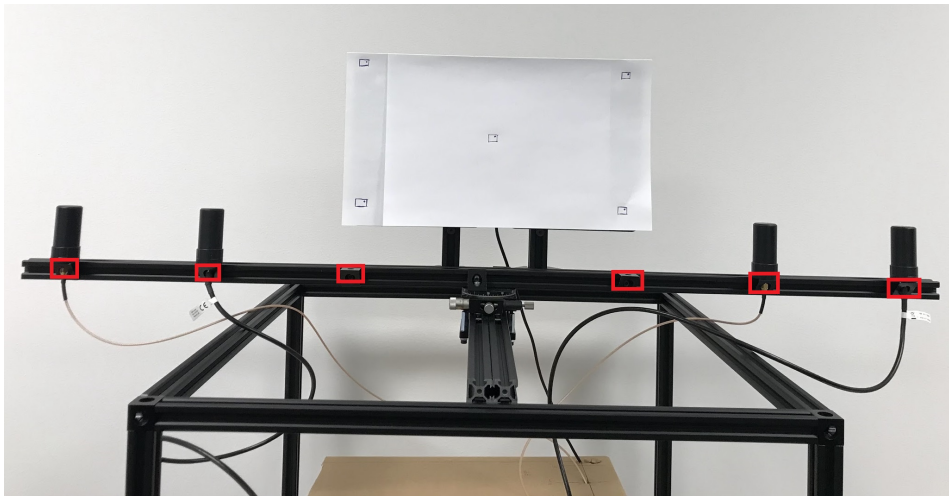
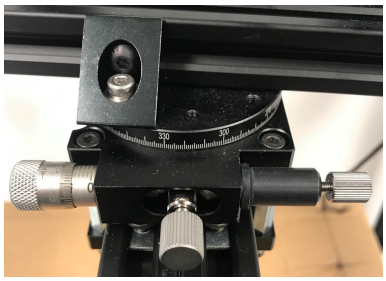


Figure 1.3: Rig setup with mounted circuit board and GNSS antennas. Total of 6 antenna mounting points with a step size of 15 cm from the center.



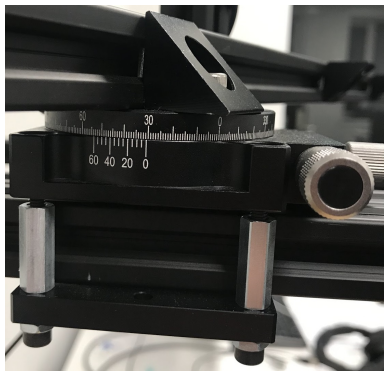



Figure 1.4: Rotating 360 degrees protractor used in the rig. Can take this part of the cubic rig and mount it on a tripod as shown in Figure 5.8.

# 2 Sensors

## 2.1 Inertial Measurement Unit (IMU)

IMU stands for inertial measurement unit and is sensor assembly of inertial sensors. All of the sampled sensor data in this thesis is gathered from a so-called 9 degree of freedom. The chosen IMU is called LSM9DS1 and is featured with a 3D digital linear acceleration sensor, a 3D digital angular rate sensor, and a 3D digital magnetic sensor (STMicroelectronics, 2015). An IMU is never perfect, and the main errors are measurement noise and measurement bias. Even with calibration these errors still has to be taken into account when performing calculation with measured values. In addition, it is not given that the mounted IMU is perfectly aligned with the body frame coordinates. All of these problems give room for possible errors when estimating roll, pitch, and yaw.

An essential aspect of an IMU is the sampling rate that is used. This is most often described in Hz and tells us how many samples are measured in one second. The IMU does typically provide measurements much more frequently than other navigation sensors, like GNSS (Global Navigation Satellite Systems). The sampling frequency of the IMU measurements used when testing is 98 Hz and was present in the hardware delivered by Squarehead Technology. The hardware is further described in Section 1.2 and comes with software to run and extract data from the IMU.

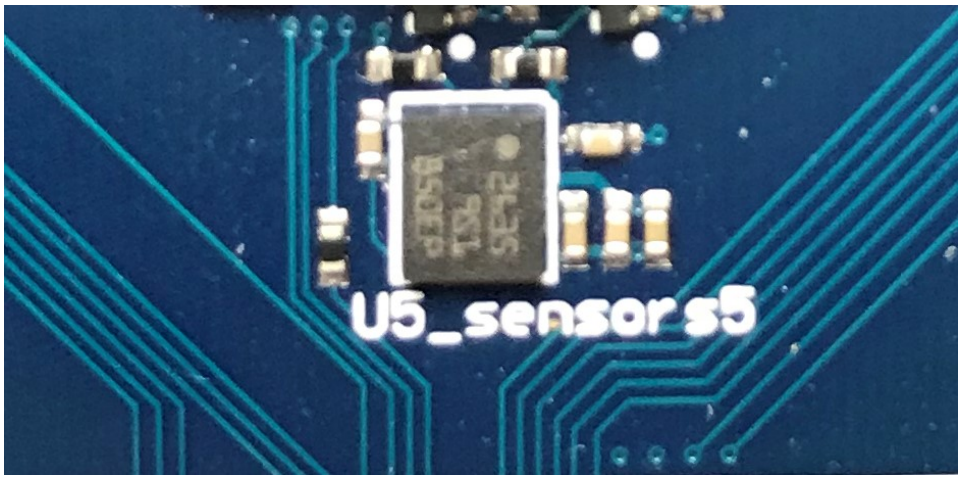



Figure 2.1: A close-up look of the middle IMU that is mounted on the circuit board.

## 2.2 Global Navigation Satellite System (GNSS)

### 2.2.1 u-blox

Using ZED-F9P (ublox, 2021b) as a base and ZED-F9H (ublox, 2021a) as a rover in a moving base application connected with one antenna each to calculate heading. The two antennas are mounted align the y-axis of the circuit board. Since we have a ZED-F9P and ZED-F9H module the relative position will be normalized around one meter. If we had two ZED-F9P, the relative position in mm could be gathered. Since only the heading is of main priority, the cheaper ZED-F9H was chosen as the second module. Both modules are gathering GNSS signals and F9P sends RTCM3 corrections messages over to F9H that can calculate the relative position heading of the system. The sampling frequency used for this is 10 Hz. The modules transmit UBX-messages that can be parsed into human readable values for every sample. The modules are shown in Figure 2.2.

For configuration of F9P and F9H the python library pyserial (pySerial, 2022) and pyubx2 (semuconsulting, 2022) was used for setup. From the integration manual (ublox, 2021c, sec. 3.1.5.6.1) for F9P it is described which RTCM3 message that needs to be enabled. These RTCM3 messages was enabled as out on F9P USB-port and in on the F9H USB-port. Further the UBX message UBX-NAV-RELPOSNED (ublox, 2020a) was enabled on the F9H. This messages contains among other things the calculated heading and the relative position vector from base to rover. On the F9P the UBX message UBX-NAV-PVT (ublox, 2020b) was enabled. This message contains among other things the latitude, longitude and height of the base. Rest of the messages on the F9H and F9P was disables to not clutter the communication. In this setup the sampling rate of the F9P and F9H was set to 10 Hz and saved to both RAM and flash, so it's not changed back to default after power-out. The connection from the computer to the u-blox devices was established over USB with pyserial, and the configuration was set with messages constructed with pyubx2.

### 2.2.2 Septentrio

Using mosaic-H (Septentrio, 2021a) with two antennas connected to calculate heading. Septentrio have accomplished what u-blox needs two modules too in only one module. The antennas setup is the same as for u-blox, align the y-axis of the circuit board. The dev-kit containing the module sent from Septentrio is in more compact plug and play form. The system can be configured to start logging data automatic when is has power. This data is on a SBF (Septentrio Binary Format) and also be parsed into human readable data for every sample. This was done after every test-case. The module have a sampling frequency up to 100 Hz, where mostly 10 Hz was used. This was done to more easily compere it with data from u-blox devices. The module is shown in Figure 2.3.


Figure 2.2: ZED-F9P and ZED-F9H from u-blox.


Figure 2.3: Mosaic-H developer kit from Septentrio.

### 2.2.3 Antenna

The antenna used is a high performance multi brand GNSS active quad helix antenna and har shown in Figure 2.4. Four equal antennas of this sort was in use, two for the u-blox setup and two for the Septentrio setup. They were all connected with a 2 meter long SMA-cable. The different mounting spots for the antennas are shown in Figure 1.3.

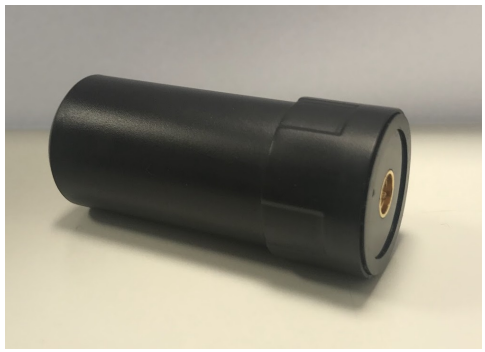


Figure 2.4: GPS L1/L2, Glonass G1/G2, Beidou B1/B2/B3, Galileo E1/E5B IP67 active quad helix antenna.

# 3 Mathematical preliminaries

This chapter will introduce the necessary mathematics to better understand calculation with rotation matrix, quaternion, and converting between quaternion, rotation matrix and Euler angles.

## 3.1 Norm

In further calculation the term Euclidean norm will be used on vectors when need of scaling. Euclidean norm (Farrell, 2008, B.2) of a $n$-dimensional vector $\boldsymbol{v} = \{v_1, ..., v_n\}$ is defined as the square root of the scalar product of a vector with itself:

$$||\boldsymbol{v}||_2 = \sqrt{\boldsymbol{v}^T \cdot \boldsymbol{v}} \tag{3.1}$$

$$||\boldsymbol{v}||_2 = \sqrt{v_1^2 + ... + v_n^2} \tag{3.2}$$

The Euclidean norm is also the length of the vector. This property can further be used to normalize a vector with

$$\bar{\boldsymbol{v}} = \frac{\boldsymbol{v}}{||\boldsymbol{v}||_2} \tag{3.3}$$

that can be used when only the relation between the values in the vector is of interest. This will be done with the measurements from the accelerometer and magnetometer. For these measurements, only the direction of the vector is necessary. Normalizing a vector is a way to scale the vector, so the vector's length is equal to 1.

## 3.2 Skew-symmetric matrix

For some calculation in the ESKF it is needed to take the cross product between two vectors. To simplify this the Skew-symmetric matrix operator will be used. The definition for this is such that $\boldsymbol{v} \times \boldsymbol{a} = \boldsymbol{S}(\boldsymbol{v})\boldsymbol{a}$, or equivalently (Brekke, 2020, eq. 10.5):

$$\boldsymbol{S}(\boldsymbol{v}) = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{3.4}$$

An important property of the Skwe-symmetric matrix is the relation between normal and the transposed matrix (Fossen, 2011, def. 2.1) and is shown in:

$$\boldsymbol{S}(\boldsymbol{v}) = -\boldsymbol{S}(\boldsymbol{v})^T \tag{3.5}$$

Another property that is also worth mentioned is:

$$\boldsymbol{S}(\boldsymbol{v})\boldsymbol{a} = -\boldsymbol{S}(\boldsymbol{a})\boldsymbol{v} \tag{3.6}$$

## 3.3 Rotation matrix

The rotation matrix is used to transform a coordinate vector in frame $b$ to a coordinate vector in frame $a$ (Egeland and Gravdahl, 2003, ch. 6.4). This can be written with the equation:

$$\boldsymbol{v}^a = \boldsymbol{R}_b^a \boldsymbol{v}^b \tag{3.7}$$

where $a$ and $b$ is two decided frames and $\boldsymbol{R}_b^a$ is the rotation matrix transforming the vector $\boldsymbol{v}^b$ into the vector $\boldsymbol{v}^a$. The notation used is illustrated as (Fossen, 2011, eq. 2.8):

$$\boldsymbol{v}^{to} = \boldsymbol{R}_{from}^{to} \boldsymbol{v}^{from} \tag{3.8}$$

An important property of the rotation matrix is the relation between normal, transposed and inverse of the matrix as shown in:

$$\boldsymbol{R}_b^a = \left(\boldsymbol{R}_a^b\right)^T = \left(\boldsymbol{R}_a^b\right)^{-1} \tag{3.9}$$

## 3.4 Quaternion

The quaternion will be used as the main way to represent the attitude state. This is a 4-dimensional attitude notation and will not suffer from singularities, like any 3-dimensional representation, where an infinite number of possible Euler angles are possible (Brekke, 2020, ch. 10). The mathematical notation for a quaternion used in this thesis is (Brekke, 2020, eq. 10.19):

$$\boldsymbol{q} = \begin{bmatrix} \eta \\ \boldsymbol{\epsilon} \end{bmatrix} \tag{3.10}$$

where

$$\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]^T \tag{3.11}$$

Taking the norm of a quaternion is done the same way as in (3.2) and yields (Brekke, 2020, eq. 10.25):

$$||\boldsymbol{q}|| = \sqrt{\eta^2 + \epsilon_1^2 + \epsilon_2^2 + \epsilon_3^2} \tag{3.12}$$

Now it is very straightforward to normalize the quaternion with

$$\bar{\boldsymbol{q}} = \frac{\boldsymbol{q}}{||\boldsymbol{q}||} \tag{3.13}$$

and is similar as done for a normal vector in (3.3).

Another good property for the quaternion is when you want to rotate the quaternion attitude representation. Suppose you have one quaternion representing the attitude and another quaternion representing the change. You can take the product between these two for obtaining the new attitude. The calculation of the quaternion product (Brekke, 2020, eq. 10.34) is:

$$\boldsymbol{q}_a \otimes \boldsymbol{q}_b = \left(\eta_a \boldsymbol{I} + \begin{bmatrix} 0 & -\boldsymbol{\epsilon}_a^T \\ \boldsymbol{\epsilon}_a & -\boldsymbol{S}(\boldsymbol{\epsilon}_a) \end{bmatrix}\right) \begin{bmatrix} \eta_b \\ \boldsymbol{\epsilon}_b \end{bmatrix} \tag{3.14}$$

## 3.5 Conversion between quaternion, rotation matrix and Euler angles

It is helpful to convert between different attitude representations. While the quaternion is used as the attitude state, a mix of quaternion and rotation matrix will be used in the ESKF calculations. When plotting the estimated attitude for the results, the Euler angles roll, pitch, and yaw will be used. This makes a more familiar and readable representation of the attitude.

Firstly we have the equation for converting from quaternion to rotation matrix (Brekke, 2020, eq. 10.37):

$$\boldsymbol{R} = \boldsymbol{I} + 2\eta \boldsymbol{S}(\boldsymbol{\epsilon}) + 2\boldsymbol{S}(\boldsymbol{\epsilon})\boldsymbol{S}(\boldsymbol{\epsilon}) \tag{3.15}$$

$$= \begin{bmatrix} \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2 & 2\left(\epsilon_1\epsilon_2 - \eta\epsilon_3\right) & 2\left(\epsilon_1\epsilon_3 + \eta\epsilon_2\right) \\ 2\left(\epsilon_1\epsilon_2 + \eta\epsilon_3\right) & \eta^2 - \epsilon_1^2 + \epsilon_2^2 - \epsilon_3^2 & 2\left(\epsilon_2\epsilon_3 - \eta\epsilon_1\right) \\ 2\left(\epsilon_1\epsilon_3 - \eta\epsilon_2\right) & 2\left(\epsilon_2\epsilon_3 + \eta\epsilon_1\right) & \eta^2 - \epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2 \end{bmatrix} \tag{3.16}$$

The rotation matrix is an attitude representation of 9 dimensions and will be used for intermediate calculations in the ESKF method.

Secondly, we have the equation for converting from quaternion to Euler angles roll, pitch, yaw, respectively:

$$\phi = \arctan2\left(2(\epsilon_3\epsilon_2 + \eta\epsilon_1), \eta^2 - \epsilon_1^2 - \epsilon_2^2 + \epsilon_3^2\right) \tag{3.17}$$

$$\theta = \arcsin\left(2(\eta\epsilon_2 - \epsilon_1\epsilon_3)\right) \tag{3.18}$$

$$\psi = \arctan2\left(2(\epsilon_1\epsilon_2 + \eta\epsilon_3), \eta^2 + \epsilon_1^2 - \epsilon_2^2 - \epsilon_3^2\right) \tag{3.19}$$

(Brekke, 2020, eq. 10.38). This will be used when plotting the attitude state in the results. Then we have to go from quaternion to a more readable representation of the attitude.

The last conversion is the equation for converting from Euler angle to quaternion (Brekke, 2020, eq. 10.39):

$$\boldsymbol{q} = \begin{bmatrix} \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{bmatrix} \tag{3.20}$$

It is more intuitive to represent the initial attitude of the circuit board as Euler angles in the Python code. This is then immediately converted to quaternion for the initial attitude state.

# 4 *Attitude estimation of a relative position sensor*

This chapter will introduce theory to better understand a step by step implementation of a error state Kalman filter (ESKF) with explanatory equations. It will dive into attitude estimates with correction based on principle with Earth's gravity, Earth's magnetic field and GNSS signals, and explain the coordinate system that are used.

## 4.1 Coordinate systems

For coordinate systems we switch between the world frame and the body frame. The world frame are set in NED, North ($x$) - East ($y$) - Down ($z$) (Brekke, 2020, fig. 10.1), and the body frame is the frame relative to the circuit board, what we try to estimate the attitude of. The notation in this master thesis will be:

- NED: $n$
- Body: $b$

Since we have some measurements in the body frame and some in the world frame we have switch between frames. An example of this is illustrated with a rotation matrix:

$$\boldsymbol{v}^{NED} = \boldsymbol{R}_{Body}^{NED}\boldsymbol{v}^{Body} \tag{4.1}$$

$$\boldsymbol{v}^{n} = \boldsymbol{R}_{b}^{n}\boldsymbol{v}^{b} \tag{4.2}$$

In Figure 4.1 the circuit board is placed in the body frame. The $xyz$-axes will be locked to this for every rotations. Then we transform over to the world frame to find the real attitude.

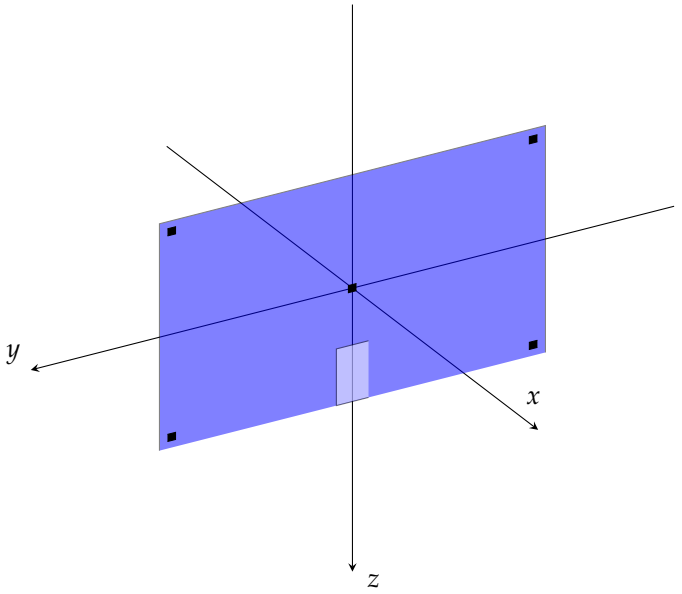



Figure 4.1: Body frame of the relative position sensor.

## 4.2 The error state Kalman filter (ESKF)

### 4.2.1 States

Will step by step show the mathematics and method behind the ESKF. We have three different state kinematics for the system, the nominal state, error state, and true state. The nominal and true state will represent the attitude and gyro bias, and the error state will represent the error between these state vectors. The notation for these will be the same as in (Brekke, 2020, eq. 10.52), but with fewer states. Since it is only of interest to estimate the attitude of the circuit board, the states containing the position and velocity are removed. The accelerometer is only used for correction and compared with Earth's gravity, so acceleration bias is removed to simplifying the state. A bias in acceleration will only yield a slight offset in the attitude representation and is durable. If the position and velocity were also predicted, this would lead to a constant drift and make the position more inaccurate over time.

Since the error state will represent the error in attitude and gyro bias, it is no need to use a 4-dimensional representation of the attitude. Here a 3-dimensional Euler angles representation is more suited. The ESKF will always try to make all of the entities in this state go towards zero. The representation is:

$$\delta \boldsymbol{x} = \begin{bmatrix} \delta \boldsymbol{\theta} & \delta \boldsymbol{\omega}_b \end{bmatrix}^T \tag{4.3}$$

The nominal state will be the predicted representation of the attitude and gyro bias. It is from this state that we extract the results shown in the plots in Section 6. The representation is:

$$\boldsymbol{x} = \begin{bmatrix} \boldsymbol{q} & \boldsymbol{\omega}_b \end{bmatrix}^T \tag{4.4}$$

The true state is completely unknown, given that this is a real experiment. If all of the data was generated through simulations, we could have used the true state of compassion when evaluating the performance of the implemented ESKF. This is not done in this thesis, but the true state is mentioned for possible later usage in further work. The representation is:

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{q}_t & \boldsymbol{\omega}_{bt} \end{bmatrix}^T \tag{4.5}$$

So when trying to estimate the error state, we use the measurements from the accelerometer and magnetometer to assist. Extracting the quaternion from the nominal state, converting it to a rotation matrix, and multiply it with the earth's gravity vector yields an acceleration vector close to what the accelerometer measurement from the IMU should be. Looking at the difference here gives us information on possible errors in the attitude and gyro bias. This is explained in more detail when updating the Kalman filter with measurement correction. The same principle is used for the magnetometer measurements and the magnetic field around the circuit board, and the relative position from GNSS of the two mounted antennas. Combining these measurements, we can validate the roll and pitch with an accelerometer and yaw with a magnetometer or relative position from GNSS.

Further, we have the error state covariance matrix $\boldsymbol{P}$. When estimating the attitude and gyroscope bias for the circuit board, we follow these steps:

1. Set $\boldsymbol{x}_0^-$, $\boldsymbol{P}_0^-$ and $k = 0$
2. Predict $\boldsymbol{x}_{k+1}^-$ and $\boldsymbol{P}_{k+1}^-$ with gyroscope measurement.
3. With new specific force and magnetometer measurements or GNSS measurements we calculate the error state $\delta \boldsymbol{x}_k$, correct the nominal state $\boldsymbol{x}_k^+$ and update the error state matrix $\boldsymbol{P}_k^+$. Otherwise we set $\boldsymbol{x}_k^+ = \boldsymbol{x}_k^-$ and $\boldsymbol{P}_k^+ = \boldsymbol{P}_k^-$.
4. Set $k = k + 1$ and repeat from 2.

### 4.2.2 Predict the next nominal state

Now the next nominal state has to be predicted. Starting with obtaining the IMU measurements assuming $\omega$ is constant over the sampling time period:

$$\boldsymbol{\omega} \approx \boldsymbol{z}_{gyro} - \boldsymbol{\omega}_b \tag{4.6}$$

Further extracting $\boldsymbol{q}$ from the nominal states and predict the next $\boldsymbol{q}$ with respect to $\boldsymbol{\omega}$ (Solà, 2017, eq. 214):

$$\boldsymbol{\kappa} = T_s \cdot \boldsymbol{\omega} \tag{4.7}$$

$$\Delta\boldsymbol{q} = e^{\frac{\boldsymbol{\kappa}}{2}} = \begin{bmatrix} \cos(\frac{||\boldsymbol{\kappa}||_2}{2}) \\ \sin(\frac{||\boldsymbol{\kappa}||_2}{2}) \frac{\boldsymbol{\kappa}^T}{||\boldsymbol{\kappa}||_2} \end{bmatrix} \tag{4.8}$$

$$\boldsymbol{q} = \boldsymbol{q} \otimes \Delta\boldsymbol{q} \tag{4.9}$$

Same as for after updating the nominal state, the quaternion also has to be normalized in this case.

To predict the complete nominal state, also the gyroscope bias is predicted with (Brekke, 2020, eq. 10.58):

$$\boldsymbol{\omega}_b \approx \boldsymbol{\omega}_b + Ts * \dot{\boldsymbol{\omega}}_b \tag{4.10}$$

$$\boldsymbol{\omega}_b = \boldsymbol{\omega}_b - p_{\omega b} \cdot \boldsymbol{I}_3 \cdot \boldsymbol{\omega}_b \tag{4.11}$$

were the bias is modeled as a Gauss-Markov process (Brekke, 2020, eq. 10.50).

### 4.2.3 Predict the error state covariance matrix

After predicting the next nominal state, our last error state covariance matrix is no longer up to date. To overcome this, a new error state covariance matrix is predicted. Starting with extracting the $\boldsymbol{q}_b^n$ from the nominal state and convert this to the rotation matrix $\boldsymbol{R}_b^n$. Further, it will be shown a way to calculate the needed $\boldsymbol{F}$ and $\boldsymbol{G}$ matrices. In this project, the given method is used because the correction of the quaternion is done with the local angular error (Solà, 2017, table 4). This decision is seen in (4.45) when updating the nominal states with the corrections from the error states and is repeated here:

$$\boldsymbol{q} = \boldsymbol{q} \otimes \delta\boldsymbol{q} \tag{4.12}$$

and yields (Brekke, 2020, eq. 10.68):

$$\boldsymbol{F}_{6x6} = \begin{bmatrix} -\boldsymbol{S}(\boldsymbol{\omega}) & -\boldsymbol{I} \\ \boldsymbol{0} & -p_{\omega b}\boldsymbol{I} \end{bmatrix} \qquad \boldsymbol{G}_{6x6} = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{4.13}$$

Now, the next error state covariance prediction can be obtained with Van Loan's formula (Brekke, 2020, eq. 4.63). First, construct the Van Loan matrix:x:

$$\boldsymbol{Q}_{6x6} = \begin{bmatrix} \sigma_{\omega}^2\boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{0} & \sigma_{\omega b}^2\boldsymbol{I} \end{bmatrix} \tag{4.14}$$

$$\boldsymbol{V}_{12x12} = \begin{bmatrix} -\boldsymbol{F} & \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T \\ \boldsymbol{0} & \boldsymbol{F}^T \end{bmatrix} \tag{4.15}$$

With this, we use Van Loan's formula:

$$\exp\left(\boldsymbol{V}T_s\right) = \exp\left(\begin{bmatrix} -\boldsymbol{F} & \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T \\ \boldsymbol{0} & \boldsymbol{F}^T \end{bmatrix} T_s\right) = \begin{bmatrix} \times & \boldsymbol{V}_2 \\ \boldsymbol{0} & \boldsymbol{V}_1 \end{bmatrix} \tag{4.16}$$

and further calculate:

$$\boldsymbol{F}_d = \boldsymbol{V}_1^T \tag{4.17}$$

$$\boldsymbol{Q}_d = \boldsymbol{V}_1^T\boldsymbol{V}_2 \tag{4.18}$$

Taking the matrix exponential of the Van Loan matrix can be time-consuming. A faster approach for this is to use a 2. order Taylor approximation:

$$\exp\left(\boldsymbol{V}T_s\right) \approx \boldsymbol{I} + \boldsymbol{V}T_s + \frac{(\boldsymbol{V}T_s)^2}{2} \tag{4.19}$$

Finally, the prediction of the next error states covariance matrix can be done with:

$$\boldsymbol{P}_k^- = \boldsymbol{F}_d\boldsymbol{P}_{k-1}^+\boldsymbol{F}_d^T + \boldsymbol{Q}_d \tag{4.20}$$

### 4.2.4 Kalman filter measurement correction

The Kalman filter will be updated with correction from the measurements, that are acceleration and magnetic strength from the IMU and relative position from GNSS. This will be divided into two different cases. The first case will use correction from the accelerate and magnetometer measurements. The second case will use correction from accelerometer and GNSS measurements. These two cases will yield different measurement matrices. This will first be described with a generic

measurement vector $\boldsymbol{y}$ and a generic reference vector $\boldsymbol{v}$ for both cases. Later in Section 4.3-4.5 the actual measurement and reference that are used will be described. As mentioned in Section 4.2.1 we further extract $\boldsymbol{q}_b^n$ from the nominal state and convert this to the rotation matrix $\hat{\boldsymbol{R}}_b^n$.

Generic measurement matrix for the first case when the measurement used for correction is in the body frame and the reference is in the NED frame. In this illustration the reference vector will be $\boldsymbol{v}^n$, the measured measurement will be $\boldsymbol{y}^b$ and the estimated measurement will be $\hat{\boldsymbol{y}}^b$. Then the estimated measurement vector can be described as

$$\hat{\boldsymbol{y}}^b = (\hat{\boldsymbol{R}}_b^n)^T \boldsymbol{v}^n \tag{4.21}$$

and the measured measurement vector described as

$$\begin{aligned}
\boldsymbol{y}^b &= (\boldsymbol{R}_b^n)^T \boldsymbol{v}^n && (4.22)\\
&= \left(\hat{\boldsymbol{R}}_b^n (\boldsymbol{I} + \boldsymbol{S}(\delta\boldsymbol{\theta}))\right)^T \boldsymbol{v}^n && (4.23)\\
&= (\boldsymbol{I} + \boldsymbol{S}(\delta\boldsymbol{\theta}))^T (\hat{\boldsymbol{R}}_b^n)^T \boldsymbol{v}^n && (4.24)\\
&= (\hat{\boldsymbol{R}}_b^n)^T \boldsymbol{v}^n + \boldsymbol{S}(\delta\boldsymbol{\theta})^T (\hat{\boldsymbol{R}}_b^n)^T \boldsymbol{v}^n && (4.25)\\
&= \hat{\boldsymbol{y}}^b - \boldsymbol{S}(\delta\boldsymbol{\theta})\hat{\boldsymbol{y}}^b && (4.26)\\
&= \hat{\boldsymbol{y}}^b + \boldsymbol{S}(\hat{\boldsymbol{y}}^b)\delta\boldsymbol{\theta} && (4.27)
\end{aligned}$$

Now we can extract $\boldsymbol{S}(\hat{\boldsymbol{y}}^b)$ to make the measurement matrix as

$$\boldsymbol{H}_1 = \begin{bmatrix} \boldsymbol{S}(\hat{\boldsymbol{y}}^b) & \boldsymbol{0} \end{bmatrix} \tag{4.28}$$

Then we have the generic measurement matrix for the second case when the measurement used for correction is in the NED frame and the reference is in the body frame. For this we have the reference vector will be $\boldsymbol{v}^b$, the measured measurement will be $\boldsymbol{y}^n$ and the estimated measurement will be $\hat{\boldsymbol{y}}^n$. Then the estimated measurement vector can be described as

$$\hat{\boldsymbol{y}}^n = \hat{\boldsymbol{R}}_b^n \boldsymbol{v}^b \tag{4.29}$$

and the measured measurement vector described as

$$\begin{aligned}
\boldsymbol{y}^n &= \boldsymbol{R}_b^n \boldsymbol{v}^b && (4.30)\\
&= \hat{\boldsymbol{R}}_b^n (\boldsymbol{I} + \boldsymbol{S}(\delta\boldsymbol{\theta}))\boldsymbol{v}^b && (4.31)\\
&= \hat{\boldsymbol{R}}_b^n \boldsymbol{v}^b + \hat{\boldsymbol{R}}_b^n \boldsymbol{S}(\delta\boldsymbol{\theta})\boldsymbol{v}^b && (4.32)\\
&= \hat{\boldsymbol{y}}^n - \hat{\boldsymbol{R}}_b^n \boldsymbol{S}(\boldsymbol{v}^b)\delta\boldsymbol{\theta} && (4.33)
\end{aligned}$$

Now we can extract $-\hat{\boldsymbol{R}}_b^n \boldsymbol{S}(\boldsymbol{v}^b)$ for the measurements matrix as

$$\boldsymbol{H}_2 = \begin{bmatrix} -\hat{\boldsymbol{R}}_b^n \boldsymbol{S}(\boldsymbol{v}^b) & \boldsymbol{0} \end{bmatrix} \tag{4.34}$$

For correction of the nominal state, we can combine multiple correction sources in on measurement matrix. For the first case the measurement matrix will be:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_{acc} \\ \boldsymbol{H}_{mag} \end{bmatrix} \quad \text{, case 1} \tag{4.35}$$

Here $\boldsymbol{H}_{acc}$ and $\boldsymbol{H}_{mag}$ are described in Section 4.3 and 4.4. For the second case the measurement matrix will be:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{H}_{acc} \\ \boldsymbol{H}_{pos} \end{bmatrix} \quad \text{, case 2} \tag{4.36}$$

Here $\boldsymbol{H}_{acc}$ and $\boldsymbol{H}_{pos}$ are described in Section 4.3 and 4.5.

Further, we calculate the Kalman gain matrix from (Solà, 2017, eq. 274) and (Brekke, 2020, eq. 10.75):

$$\boldsymbol{K} = \boldsymbol{P}_k^- \boldsymbol{H}^T \left( \boldsymbol{H} \boldsymbol{P}_k^- \boldsymbol{H}^T + \boldsymbol{R}_y \right)^{-1} \tag{4.37}$$

For the first case the accuracy matrix of the measurements is:

$$\boldsymbol{R}_y = \begin{bmatrix} \boldsymbol{R}_{acc} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{mag} \end{bmatrix} \quad \text{, case 1} \tag{4.38}$$

For the second case the accuracy matrix of the measurements is:

$$\boldsymbol{R}_y = \begin{bmatrix} \boldsymbol{R}_{acc} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{R}_{pos} \end{bmatrix} \quad \text{, case 2} \tag{4.39}$$

The values of the accuracy matrices can be found in Table 5.1.

The Kalman gain matrix is then used to calculate the estimated error state (Solà, 2017, eq. 275). For the first case this is:

$$\delta \boldsymbol{x}_k = \boldsymbol{K} \begin{bmatrix} \tilde{\boldsymbol{y}}_{acc}^b - \hat{\boldsymbol{y}}_{acc}^b \\ \tilde{\boldsymbol{y}}_{mag}^b - \hat{\boldsymbol{y}}_{mag}^b \end{bmatrix} \quad \text{, case 1} \tag{4.40}$$

For the second case the calculation is:

$$\delta \boldsymbol{x}_k = \boldsymbol{K} \begin{bmatrix} \bar{\boldsymbol{y}}_{acc}^b - \hat{\boldsymbol{y}}_{acc}^b \\ \bar{\boldsymbol{y}}_{pos}^n - \hat{\boldsymbol{y}}_{pos}^n \end{bmatrix} \quad \text{, case 2} \tag{4.41}$$

The different measurement vectors of $\boldsymbol{y}$ are described in Section 4.3-4.5.

Now we have to update the error state covariance matrix $\boldsymbol{P}_k^+$ (Solà, 2017, eq. 276) and is done with the symmetric and positive Joseph form:

$$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}) \boldsymbol{P}_k^- (\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H})^T + \boldsymbol{K}\boldsymbol{R}_y\boldsymbol{K}^T \tag{4.42}$$

$$\boldsymbol{P}_k^+ = \frac{\boldsymbol{P}_k^+ + (\boldsymbol{P}_k^+)^T}{2} \tag{4.43}$$

After estimating the error state, it is time to update the nominal state $\boldsymbol{x}_k^+ = \boldsymbol{x}_k^- \otimes \delta\boldsymbol{x}_k$ with corrections from the error state (Brekke, 2020, eq. 10.72) and (Solà, 2017, eq. 283c, 283e):

$$\delta\boldsymbol{q} = \begin{bmatrix} 1 \\ \delta\boldsymbol{\theta}/2 \end{bmatrix} \tag{4.44}$$

$$\boldsymbol{q} = \boldsymbol{q}_b^n \otimes \delta\boldsymbol{q} \tag{4.45}$$

$$\boldsymbol{\omega}_b = \boldsymbol{\omega}_b + \delta\boldsymbol{\omega}_b \tag{4.46}$$

We want to keep the quaternion as a unit quaternion. This means that the length of the quaternion is equal to 1 and will be achieved by normalizing it, as done in (3.13).

Finally, we also correct the error states covariance matrix $\boldsymbol{P}_k^+$ after correction of the nominal states (Brekke, 2020, eq. 10.86):

$$\boldsymbol{P}_k^+ = \boldsymbol{G}\boldsymbol{P}_k^+\boldsymbol{G}^T \qquad \boldsymbol{G}_{6x6} = \begin{bmatrix} \boldsymbol{I} - S(\delta\boldsymbol{\theta}/2) & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I} \end{bmatrix} \tag{4.47}$$

## 4.3 Acceleration measurements for pitch and roll correction

For a stationary system, the accelerometer measurements can be used to correct the roll and pitch (Farrell, 2008, eq. 11.144). This is illustrated in Figure 4.2 and 4.3. It's shows clearly that pitch goes to zero when $acc_x$ goes to zero and roll goes to zero when $acc_y$ goes to zero. The measurement vector from the accelerometer can be described as:

$$\boldsymbol{y}_{acc}^n = \begin{bmatrix} acc_x & acc_y & acc_z \end{bmatrix}^T \tag{4.48}$$

All of the measurements received from the IMU have the true magnitude of the measured values. In the case of the acceleration and magnetometer, only the direction of the measurements is of interest. With this in mind, we can scale the measurement vectors and the corresponding reference vectors with normalizing (3.3):

$$\bar{\boldsymbol{v}}_{acc}^n = \frac{\boldsymbol{v}_{acc}^n}{||\boldsymbol{v}_{acc}^n||_2} \tag{4.49}$$

$$\bar{\boldsymbol{y}}_{acc}^b = \frac{\boldsymbol{y}_{acc}^b}{||\boldsymbol{y}_{acc}^b||_2} \tag{4.50}$$

Than we estimate the acceleration measurement in this orientation given Earth's gravity as the only force:

$$\hat{\boldsymbol{y}}_{acc}^{b} = (\hat{\boldsymbol{R}}_{b}^{n})^{T} \cdot \bar{\boldsymbol{v}}_{acc}^{n} \tag{4.51}$$

This will be used for correction in roll and pitch. Here the referance vector $\boldsymbol{v}_{acc}^{n} = \boldsymbol{g}^{n} = [0, 0, -9.81]^{T}$, the Earth's gravity vector when in idle position. All the force goes down in the $z$-direction in the body frame. The measurement matrix for this part will be:

$$\boldsymbol{H}_{acc} = \begin{bmatrix} \boldsymbol{S}(\hat{\boldsymbol{y}}_{acc}^{b}) & \boldsymbol{0} \end{bmatrix} \tag{4.52}$$

We then use $\bar{\boldsymbol{y}}_{acc}^{b} - \hat{\boldsymbol{y}}_{acc}^{b}$ to calculate $\delta \boldsymbol{x}_{k}$ as described in Section 4.2.4.

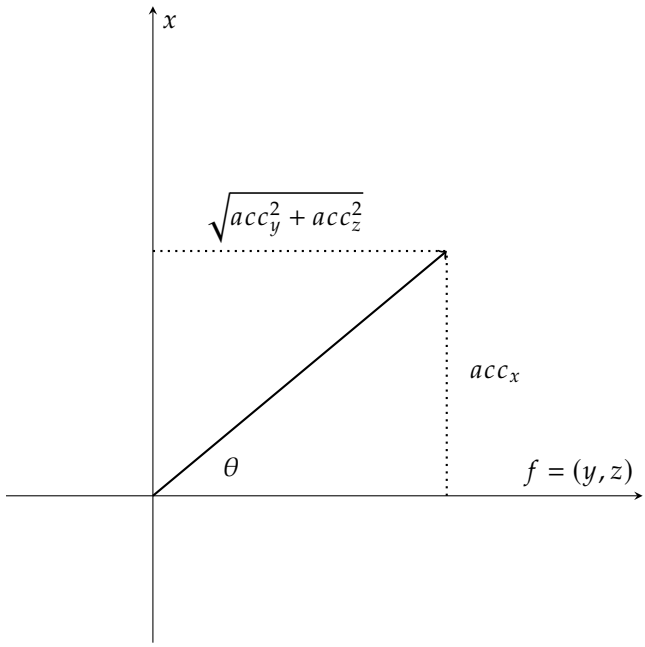



Figure 4.2: Using the accelerometer measurements vector to calculate pitch in the world frame for a stationary device.

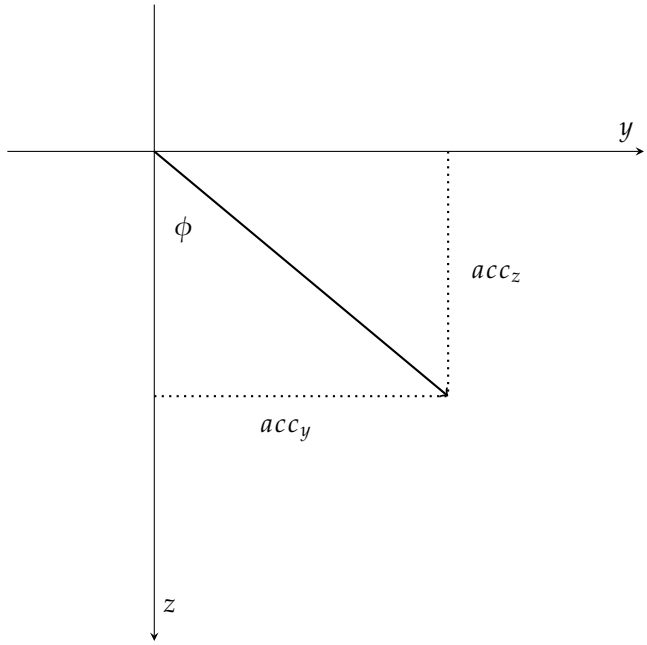



Figure 4.3: Using the accelerometer measurements vector to calculate roll in the world frame for a stationary device.

## 4.4 Magnetometer measurements for yaw correction

We can use the measurement from the magnetometer to find a correction of the yaw angle (Farrell, 2008, eq. 10.18). To use this for correction in our ESKF we have the magnetic reference vector $\boldsymbol{v}_{mag}^n$ and the measured magnetic force vector $\boldsymbol{y}_{mag}^b$ obtained by the magnetometer. The measurement vector obteind by the magnetometer can be described as:

$$\boldsymbol{y}_{mag}^b = \begin{bmatrix} mag_x & mag_y & mag_z \end{bmatrix}^T \tag{4.53}$$

Those are then normalised since we only need the direction of the magnetic force:

$$\bar{\boldsymbol{v}}_{mag}^n = \frac{\boldsymbol{v}_{mag}^n}{||\boldsymbol{v}_{mag}^n||_2} \tag{4.54}$$

$$\bar{\boldsymbol{y}}_{mag}^b = \frac{\boldsymbol{y}_{mag}^b}{||\boldsymbol{y}_{mag}^b||_2} \tag{4.55}$$

Here magnetic reference vector $\boldsymbol{v}_{mag}^n$ is set equal to the Earth's magnetic field at the IMU's location on Earth. How this magnetic reference is obtained is explained in 5.5.

For correction in yaw, we calculate the cross-product between the accelerometer and magnetic field measurement for better stability. We do the same for Earth's gravity and magnetic reference.

$$\tilde{\boldsymbol{y}}^b_{mag} = \bar{\boldsymbol{y}}^b_{acc} \times \bar{\boldsymbol{y}}^b_{mag} \tag{4.56}$$

$$\hat{\boldsymbol{y}}^b_{mag} = (\hat{\boldsymbol{R}}^n_b)^T \cdot (\bar{\boldsymbol{v}}^n_{acc} \times \bar{\boldsymbol{v}}^n_{mag}) \tag{4.57}$$

The measurement matrix for this part will be:

$$\boldsymbol{H}_{mag} = \begin{bmatrix} \boldsymbol{S}(\hat{\boldsymbol{y}}^b_{mag}) & \boldsymbol{0} \end{bmatrix} \tag{4.58}$$

We then use $\tilde{\boldsymbol{y}}^b_{mag} - \hat{\boldsymbol{y}}^b_{mag}$ to calculate $\delta \boldsymbol{x}_k$ as described in Section 4.2.4.

## 4.5 GNSS measurements for yaw and roll correction

With two GNSS antennas, one as a base and the other as a rover, the relative position from base to rover in the NED frame can be calculated. This is done by the ZED-F9P combined with ZED-F9H modules from u-blox or the mosiac-H module from Septentrio. This can further be used to calculate the heading. We can describe the relative position from the base to the rover in the NED frame with:

- $N$: Relative position north
- $E$: Relative position east
- $D$: Relative position down

The same principle used for correction with accelerometer measurements can be applied for relative position measurements. The estimate of yaw is shown in Figure 4.4, and illustrate clearly that if $E$ goes to zero the antenna alignment is heading straight north. Same as if $N$ goes to zero the heading is east. This works since the two antennas is mounted on a fixed location on the system align the y-axis. Since the y-axis is selected roll is the second attitude angle that can be estimated as shown in Figure 4.5. If we placed the antennas align the x-axis or had three antennas, the pitch could also be estimated.

With the $N$, $E$, and $D$ components we can calculate the heading and roll or pitch:

$$\psi = \text{arctan2}\,(E, N) \cdot \frac{180}{\pi} \tag{4.59}$$

If the GNSS antennas are placed on the Y-axis:

$$\phi = \text{arctan2}\left(D, \sqrt{N^2 + E^2}\right) \cdot \frac{180}{\pi} \tag{4.60}$$

If the GNSS antennas are placed on the X-axis:

$$\theta = \text{arctan2}\left(D, \sqrt{N^2 + E^2}\right) \cdot \frac{180}{\pi} \tag{4.61}$$

With this in mind we can use the relative position measurement to update the Kalman filter for a more accurate heading estimate. The measurement will be:

$$\boldsymbol{y}_{pos}^{n} = \begin{bmatrix} N & E & D \end{bmatrix}^T \tag{4.62}$$

Since only the direction is important we normalize the measurement with:

$$\bar{\boldsymbol{y}}_{pos}^{n} = \frac{\boldsymbol{y}_{pos}^{n}}{||\boldsymbol{y}_{pos}^{n}||_2} \tag{4.63}$$

Further we need to know the relative position of the antennas placed on the rig relative to the circuit board in the body frame. This will be the relative position reference in meters when the baseline between the antennas is 0.75 meters:

$$\boldsymbol{v}_{base}^{b} = \begin{bmatrix} 0.03 & 0.35 & 0.15 \end{bmatrix} \tag{4.64}$$

$$\boldsymbol{v}_{rover}^{b} = \begin{bmatrix} 0.03 & -0.45 & 0.15 \end{bmatrix} \tag{4.65}$$

$$\boldsymbol{v}_{pos}^{b} = \boldsymbol{v}_{rover}^{b} - \boldsymbol{v}_{base}^{b} = \begin{bmatrix} 0 & -0.75 & 0 \end{bmatrix} \tag{4.66}$$

This is also normalized with:

$$\bar{\boldsymbol{v}}_{pos}^{b} = \frac{\boldsymbol{v}_{pos}^{b}}{||\boldsymbol{v}_{pos}^{b}||_2} \tag{4.67}$$

Since the relative position refernace vector is normalized a change in the baseline does not yield a change in $\bar{\boldsymbol{v}}_{pos}^{b}$.

Our estimated relative position vector in body frame is:

$$\hat{\boldsymbol{y}}_{pos}^{n} = \hat{\boldsymbol{R}}_{b}^{n} \cdot \bar{\boldsymbol{v}}_{pos}^{b} \tag{4.68}$$

The measurement matrix is then:

$$\boldsymbol{H}_{pos} = \begin{bmatrix} -\hat{\boldsymbol{R}}_{b}^{n}\boldsymbol{S}(\bar{\boldsymbol{v}}_{pos}^{b}) & \boldsymbol{0} \end{bmatrix} \tag{4.69}$$

We then use $\bar{\boldsymbol{y}}_{pos}^{n} - \hat{\boldsymbol{y}}_{pos}^{n}$ to calculate $\delta\boldsymbol{x}_k$ as described in Section 4.2.4.

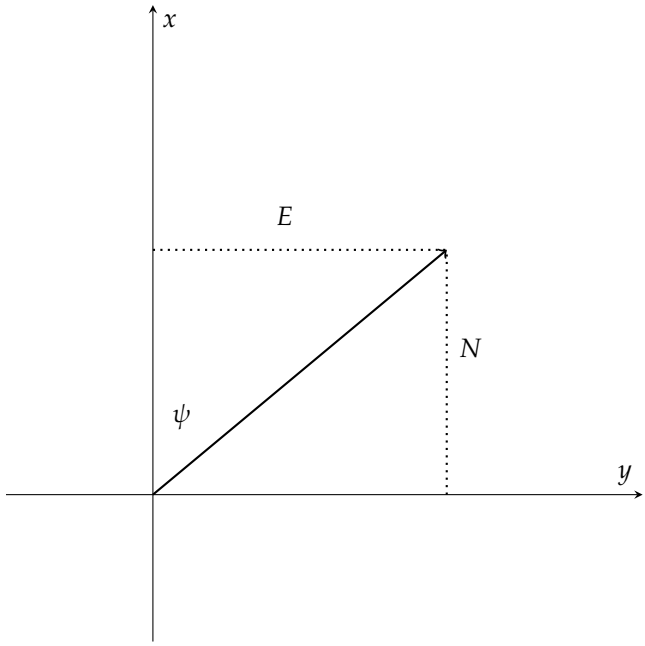



Figure 4.4: Using relative position in north and east direction to calculate yaw in the world frame.

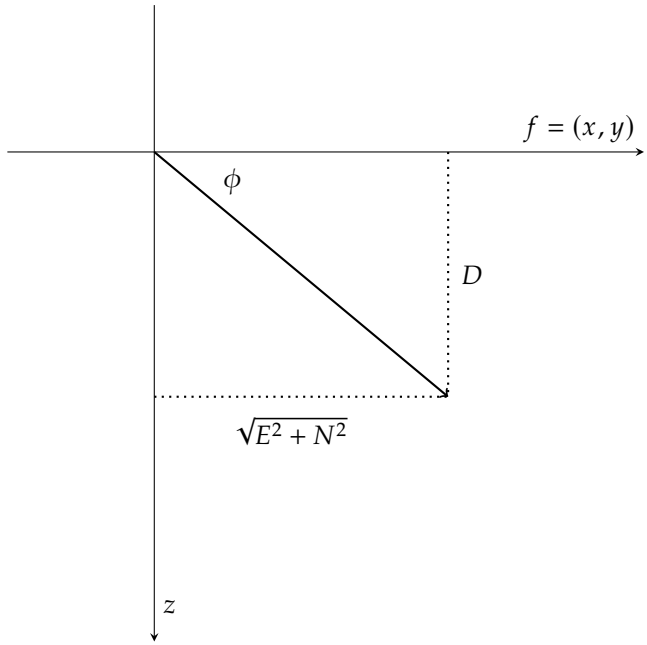



Figure 4.5: Using relative position in north, east and down direction to calculate roll in the world frame.

## 4.6 Real-Time Kinematic (RTK)

The RTK is based on the principle of differential GNSS. This uses the knowledge that the absolute position error for two close by antennas are the same and with this the relative position between the antennas can be calculated with centimetre's precision (Ardusimple, 2022).

When the "Base" and "Rover" are placed on the same unit we can use this relative position between the two GNSS antennas to calculate the heading of the unit. We can call this a "Moving Base" application. The principle for this is shown in Figure 4.4.

For this to work the "Base" has to send the "Rover" correction messages with the RTCM3 protocol. All of this is already implemented in the GNSS modules delivered from u-blox and Septentrio used in this project.

# 5 Implementation

## 5.1 Software setup

### 5.1.1 Gathering of sensor data

All of the sensor data extracted from the IMU in Figure 2.1 and the GNSS modules from u-blox in Figure 2.2 and Septentrio in Figure 2.3 is stored in CSV files for post estimation of attitude. Data from the IMU is gathered by a program in Python running inside a Docker container. Docker is needed here since the Python library communicating with the circuit board requires CentOS 7 as the operating system, and my computer is running Ubuntu 20.04. With a real time stream of IMU data when connected to the circuit board the measurements from accelerometer, gyroscope and magnetometer are consecutively saved two a CSV file together with a timestamp. Here one line in the CSV file equal one IMU measurement.

At the same time the GNSS modules are running. Connection to the u-blox modules is done with command line program str2str from RTKLIB (RTKLIB, 2022). With str2str the data stream can be split into multiple streams. For the F9P, one stream is sent to a Python program over a TCP connection, one stream is redirected to the F9H for RTK correction with RTCM3 meassages, and one stream is saved as a raw binary file if needed for later validation of the data. For the F9H the data stream is split into two, one for TCP connection to a Python program, and another for saving the raw data to a binary file. The flow of communication when gathering data from IMU and u-blox devices is also illustrated in Figure 5.1.

When collecting data from the Septentrio device you have less control over the software since you are forced to use the provided software from Septentrio. The mosaic-H development kit comes with a pre-install web interface. From this web interface you can log a data stream of the necessary data and save it to a SBF (Septentrio binary format) file. Later this file can be converted to a CSV file with the program SBF Converter in RxTool (Septentrio, 2021b). The flow of communication when gathering data from IMU and Septentrio device is also illustrated in Figure 5.2.

Every data sample from the GNSS devises contains a GNSS time that will be used to synchronize it with the corresponding IMU sample. This if further explained in Section 5.7.

When every thing is connected it's all run by one python program controlling multiple threads working simultaneous. For the u-blox setup the python program first starts the data streams for the IMU and for both the GNSS modules with two suppressor call to str2str. Then another thread starts a TCP server that str2str can connect to for sending data. With all of the streams up and running the data is recived and saved as different CSV-files. The data from u-blox is recived as UBX messages and are converted to human readable data with the Python library

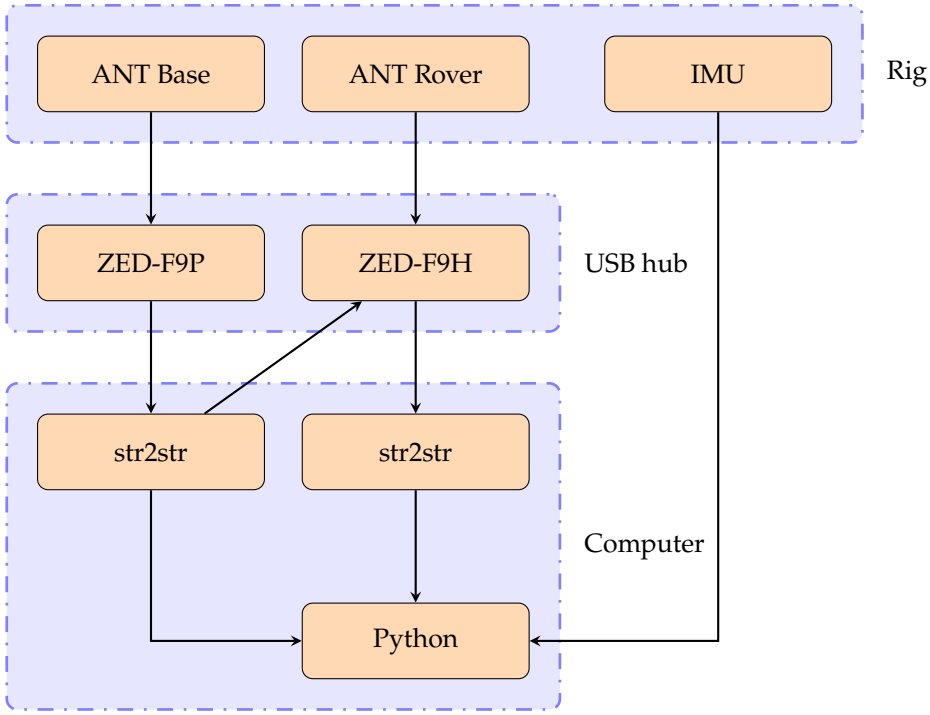



Figure 5.1: Flow chart of interaction between devices and programs when collecting data. The u-blox GNSS modules are used here.

pyubx2 (semuconsulting, 2022) before saving.

The Septentrio data stream is manually started in the web interface and is run isolated from the python program controlling the data streams from IMU and u-blox devices.

Different data set is labeled based on the given test case (type of movement in pitch, roll and yaw direction for a set time duration) to keep control over the increasing test cases conducted.

### 5.1.2 Estimation of attitude and gyroscope bias

After all of the sensor data is gathered the attitude of the circuit board and the gyroscope bias can be estimated. The calculation for this is done in Python. Before running the estimation the version of measurements correction needs to be set, magnetometer measurements or relative position from GNSS measurements. When only using the measurements from the IMU all of the samples is perfectly synced with each other with sample rate of 98 Hz. When combining mesuremtns from the IMU and GNSS there is a different in sample rate, where GNSS use a sample rate of 10 Hz. This means that around 90% of the iterations in the ESKF loop

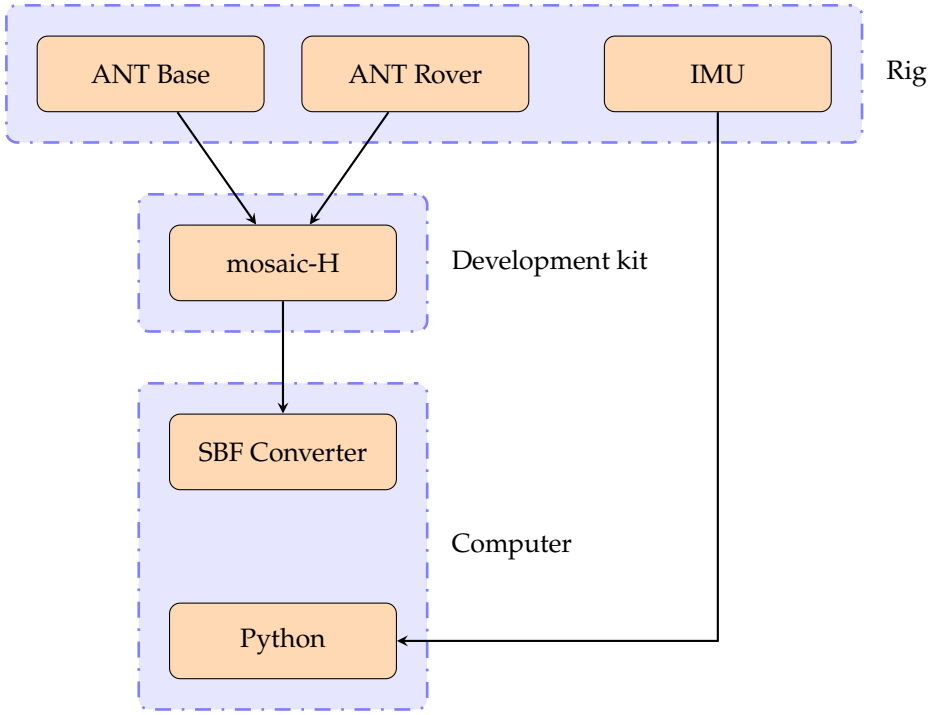



Figure 5.2: Flow chart of interaction between devices and programs when collecting data. The Septentrio GNSS module is used here.

only have the acceleration measurement for the Kalman filter correction. When this is the case, the measurement matrix $\boldsymbol{H}_{pos}$ and the measurement vector $\boldsymbol{y}^n_{pos}$ are removed from the update. Further the central part of the Python program can be described with the numbered stages:

1. Load sensor data from labeled data set.
2. Rotate IMU data as shown in Section 5.2.
3. Allocate memory for all the results.
4. Set parameters values as shown in Table 5.1.
5. Set initialization values for the nominal state and the error state covariance matrix as shown in (5.3).
6. Run a loop for all the iterations:
    a) Predict the next nominal state and error state covariance matrix as shown in Sections 4.2.2 – 4.2.3.
    b) Update the Kalman filter as shown in Section 4.2.4.

7. Plot the estimated nominal state for each iteration. The quaternion is converted to Euler angles with (3.17 – 3.19), and the gyro bias is taken directly from the state.

## 5.2 Rotation of sensors axes

From Figure 5.3 we see the original rotation for the IMU in Figure 2.1. The white dot shown on the IMU in Figure 5.3 corresponds to the dots on the mounted IMUs shown on the circuit board in Figure 1.1. For the axes displayed in Figure 1.2 to be correct, all of the measurements received were rotated before further use in the implemented system. The accelerometer and the gyroscope were rotated with the rotation matrix:

$$\boldsymbol{R} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{5.1}$$

The magnetometer was rotated with the rotation matrix:

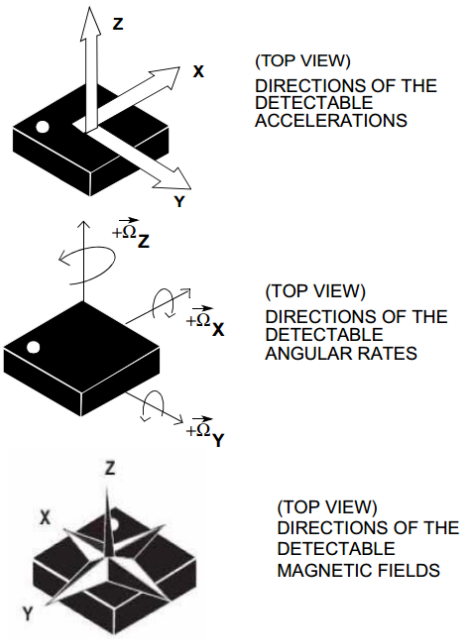



Figure 5.3: Sensor axes from the datasheet STMicroelectronics (2015, Figure 1.)

$$\boldsymbol{R} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{5.2}$$

Multiplying all of the measurements with its corresponding rotation matrix yields new orientations set in the same frame. With this, we get the wanted x-, y-, and z-direction for accelerometer, gyroscope, and magnetometer as shown in Figure 1.2. Now roll, pitch, and yaw are zero when the circuit board is in idle position. The new IMU rotation is used for all the measurement data that is gathered to estimate the attitude for the circuit board.

## 5.3 Tuning

The parameters were set by testing the system and adjusting for a better result. From the last table shown in Zinn (2018), focusing on the values for the LSM9DS1 model. This model is the same IMU that is mounted on the circuit board used in this project. Here we gather the standard deviation for the gyroscope measurement as 1.38 deg/$\sqrt{\text{h}}$ and the gyro bias stability as 61.2 deg/h. For usage in the model implemented in python, these values are converted to rad/s as shown in Table 5.1.

## 5.4 Parameters and initial states

Accelerator and magnetometer measurements are used in the Kalman filter to correct the estimated attitude based on gyroscope measurements. The constant noise variance matrix values for acceleration, magnetometer and relative position from GNSS used in the Kalman filter are shown in Table 5.1 as $\boldsymbol{R}_{acc}$, $\boldsymbol{R}_{mag}$ and $\boldsymbol{R}_{pos}$.

| Parameter | Value | Unit |
|---|---|---|
| $\boldsymbol{R}_{acc}$ | $0.5^2 \cdot \boldsymbol{I}_3$ | |
| $\boldsymbol{R}_{mag}$ | $0.05^2 \cdot \boldsymbol{I}_3$ | |
| $\boldsymbol{R}_{pos}$ | $0.05^2 \cdot \boldsymbol{I}_3$ | |
| $\sigma_\omega$ | $\frac{61.2}{3600} \cdot \frac{\pi}{180}$ | rad/s |
| $\sigma_{\omega b}$ | $\frac{1.38}{60} \cdot \frac{\pi}{180}$ | rad/s |
| $p_{\omega b}$ | $\frac{1}{3600}$ | 1/s |
| $fs$ | 100 | 1/s |
| $g$ | 9.81 | m/s$^2$ |

Table 5.1: Parameters used for IMU data set.

$\boldsymbol{x}_0$ is the initial nominal state used when the sensor is in an assumed idle starting position. Here the identity quaternion is used and corresponds to roll, pitch, and yaw equal to zero. The gyro bias is also set to zero. This is fitting to the initial idle state that is wanted. $\boldsymbol{P}_0$ is used for the initial error state covariance matrix. These initial states are shown in (5.3).

$$\boldsymbol{x}_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{P}_0 = \begin{bmatrix} \boldsymbol{I}_3(6 \cdot \frac{\pi}{180})^2 & \boldsymbol{0}_3 \\ \boldsymbol{0}_3 & \boldsymbol{I}_3(1 \cdot \frac{\pi}{180})^2 \end{bmatrix} \tag{5.3}$$

## 5.5 Earth's magnetic filed as magnetometer reference

Used data from World Magnetic Model (WMM, 2021) combined with latitude, longitude and altitude to calculate the Earth's magnetic field on a given location. The source code for the single point calculator was downloaded. This calculator yields magnetic field strength for a given location on a give data. With some modification to the source code it was complied to a run-able pogrom that could be automatic called from a sub-process in python. With location information gathered from GNSS the magnetic reference is automatic set when running a data set. Used With a perfectly calibrated magnetometer this reference will give a accurate estimate of the heading. The location used for testing is:

- Latitude: 59.949887547155775 [deg]
- Longitude: 10.763307182011644 [deg]
- Height: 100 [m]

and are also shown in Figure 5.9. For the date 20.12.2021 this yields the magnetic reference:

$$\boldsymbol{v}^n_{mag} = \begin{bmatrix} 0.15087342 \\ 0.01107929 \\ 0.49120977 \end{bmatrix} \tag{5.4}$$

The WMM gives the result in nano tesla, so $\boldsymbol{v}^n_{mag}$ was convertet to guass with the relation between gauss and tesla shown here:

$$1\text{gauss} = 10^{-4}\text{tesla} \tag{5.5}$$

## 5.6 Calibration of magnetometer

For the magnetometer measurements to be usable the magnetometer has to be calibrated. There is a lot of disturbance that can interfere with the magnetometer measurements. Many of these disturbance can be stronger then the magnetic force from the Earth's magnetic field, the reference vector we compere our measurements with. Therefore it is absolutely necessary to calibrate the magnetometer for any hope of a accurate heading correction. To calibrate the magnetometer the hard and soft iron offset has to be found. The hard iron offset is often generated based on permanent magnetic field from materials on or close to the device. Where the soft iron offset is often generated based on items inside the device that generate a time varying magnetic field. For this project only the hard iron offset is calculated and applied. Reason for this is that the main focus was used on the GNSS solution and from history the product that will use the IMU has a really strong and unpredictable soft iron offset. This make the magnetometer unreliable source of heading correction out in the field.

Finding the hard iron offset in the magnetometer is done by rotating the circuit board around every axes and are based in the principle described in the apllication note *Using LSM303DLH for a tilt compensated electronic compass* (STMicroelectronics, 2010, sec. 3.2). With the data set generated of all the rotation we can calculate the minimum and maximum value of magnetometer measurements in all of the three axes. Then we find the offset that makes the minimum and maximum value of each axis as equal as possible the same distance from zero, but in opposite directions.

This calibration was done on the test location. A result of the raw magnetometer measurements before calibrating is shown in Figure 5.4, and clearly display a strong hard iron offset. This is fixed with the simple hard iron calibration and are shown in Figure 5.5. After calibration the magnetometer data set yields a sphere as shown in Figure 5.6. The magnetometer offset found by calibration is described with numbers as

- Offset x: -0.14527
- Offset y: 0.16070
- Offset z: -0.30865

and are subtracted from all the magnetometer measurements used in the ESKF.

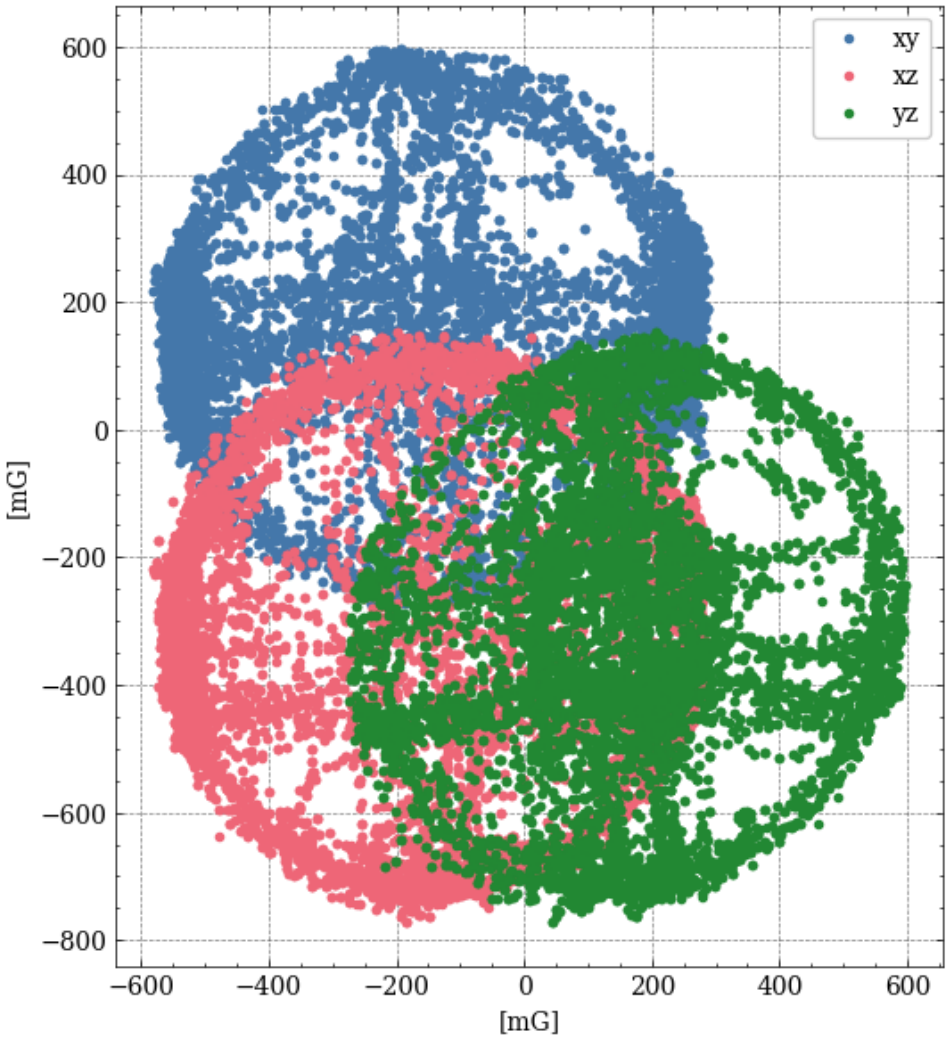



Figure 5.4: Raw magnetometer data (not calibrated) in 2d plane for every axes.

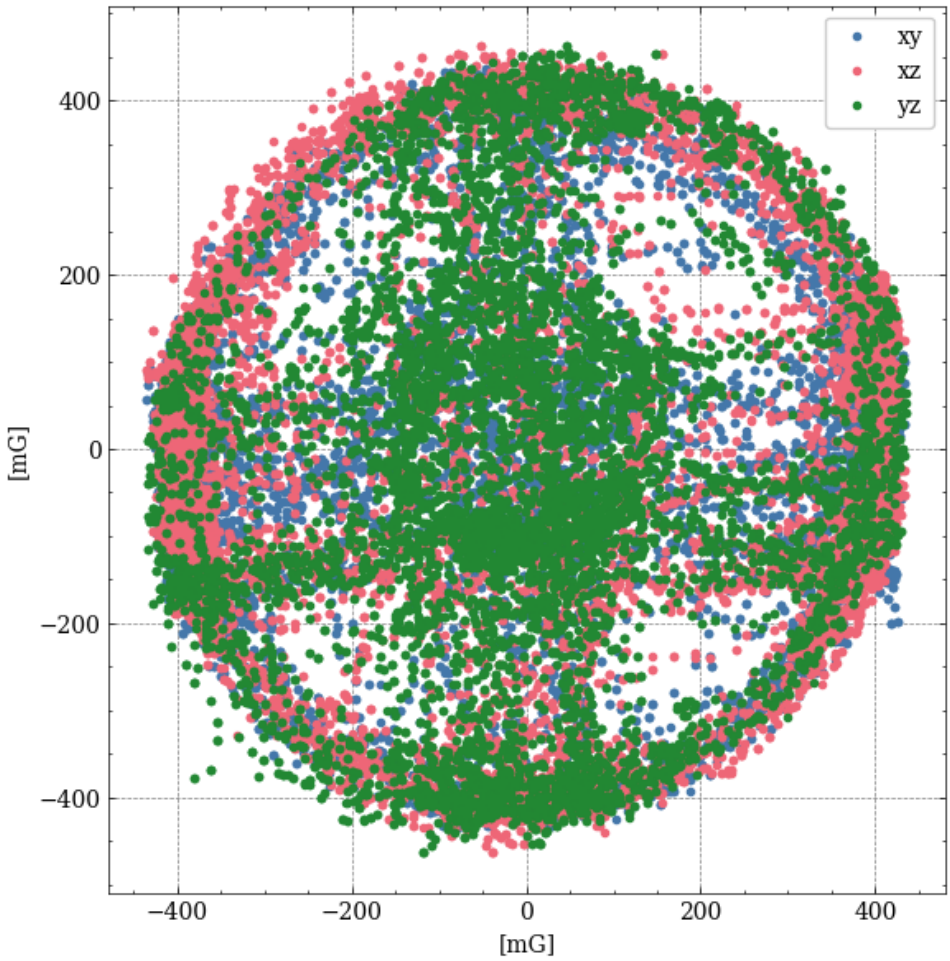



Figure 5.5: Magnetometer data in 2D plane for every axes after calibration of hard iron offset.

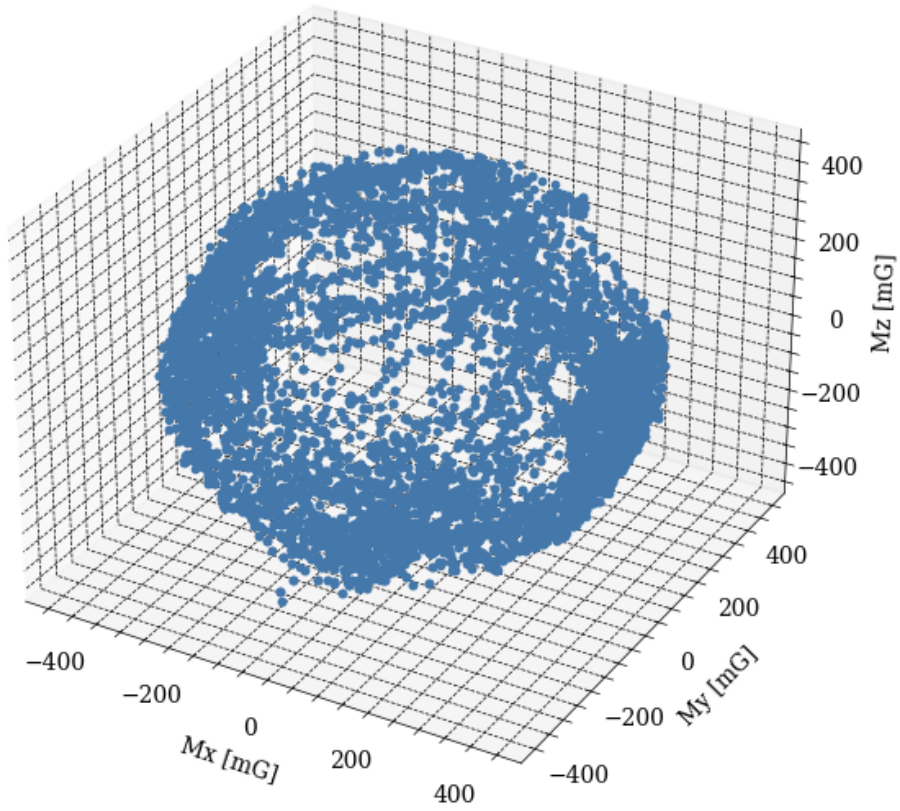



Figure 5.6: Magnetometer data in 3D after calibration of hard iron offset.

## 5.7 Synchronization between GNSS data and IMU data

The GPS time from Septentrio and u-blox is given by TOW (time of week) and the timestamp for the IMU samples are given by the clock on the computer in UTC time. In 2021 the number of leap seconds between GPS time and UTC time was 18 seconds (EndRunTechnologies, 2022), where GPS time is ahead. Based on the week the data was gathered, you can convert from TOW to UTC since TOW is given in seconds since the start of that week. After that the 18 seconds is subtracted and a time that correspond with the correct IMU sample timestamp is obtained. This synchronization is not perfect, but since the main focus of the system is in idle position some minor delay between GNSS and IMU measurements are durable.

## 5.8 Conducting testing

Testing was done on a outside terrace shown in Figure 5.7 and 5.8. The cube setup was used for more stable usage when roll and pitch was wanted to be zero. The more mobile tripod setup was used movement around all the axes was wanted. The location of the testing is also shown as a red dot in the map in Figure 5.9. Gathering of data has been on multiple days in the duration from November 2021 to December 2021, with different kind of weather and time of the day. It is worth to mention that in the north-east direction the building keeps going up around 20-30 meters and could be blocking or reflecting some of the GNSS signals. It is also several fans blowing air from the building to the outside connected to the terrace. This could be a source of magnetic interference. It is observable from the data set that the same calibration of the magnetometer does not work for every day that data is sampled, still when it is the exact same location and rig position.

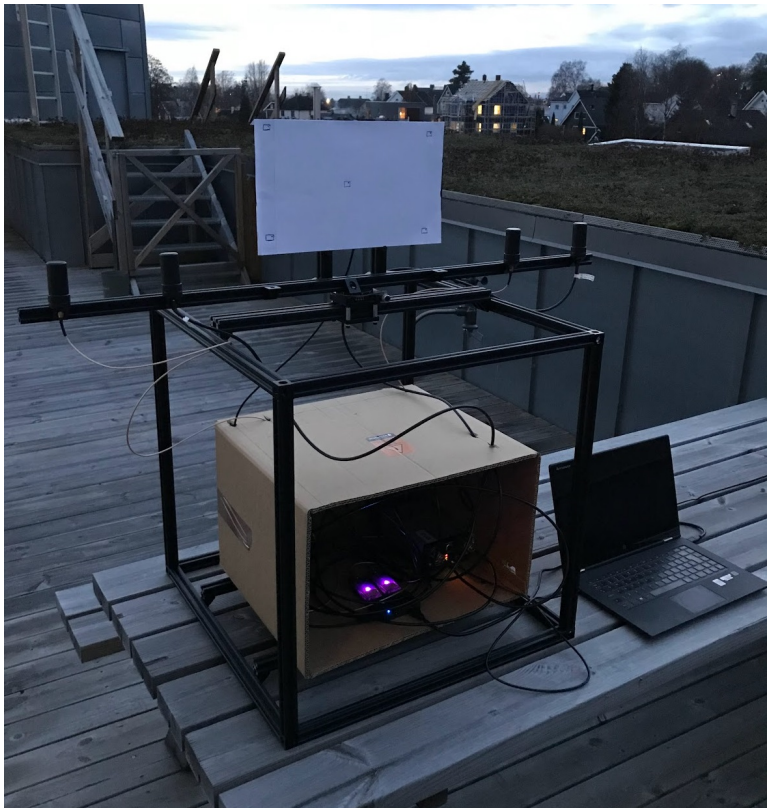


Figure 5.7: Rig setup with mounted circuit board and GNSS antennas.

Figure 5.8: Rig setup with mounted circuit board and GNSS antennas for tripod.


Figure 5.9: Test location at Nydalen with GNSS position.

## 5.9 Simulation of IMU data

To test the theoretical accurate of a system with perfect IMU measurements a simple simulator was build. This was done with Sensor Models from the Navigation Toolbox in Matlab (MathWorks, 2021). With the simulator data for accelerometer, gyroscope and magnetometer for different movement can be obtained. The function `kinematicTrajectory` was used to make a angular rotation trajectory. This trajectory was later added to a `imuSensor` class object containing the detail of the IMU, as magnetic field reference and gyroscope bias. With this object the accelerometer, gyroscope and magnetometer readings for the given trajectory would be extracted and converted to a CSF file on the same format for the real IMU used in this project.

# 6 Results

The results are divided into 9 different cases with different data set and showcase different discoveries found in this project. For the cases with estimation of of attitude and gyro bias the same figure are shown and are state estimates, difference in heading and the raw IMU data used. Only GNSS data from u-blox is used in ESKF correction. This choice was based on time and not getting a lot of repetitive results. Since both u-blox and Septentrio yields almost the same heading when used with the same baseline between antennas, as shown in the first case. For the the other and more unique cases describable figures are used. All of the plotting is done in Python.

## 6.1 Case 1 - Heading comparison between u-blox and Septentrio

In this case the testing is done to show the difference between the heading from mosaic-H from Septentrio and F9P and F9P from u-blox on the same rig at the same time. In Figure 6.1 we have the same baseline and this yields a minor difference. When the baseline for the u-blox setup is decreased the difference in heading increases as shown in Figure 6.2 and 6.3.

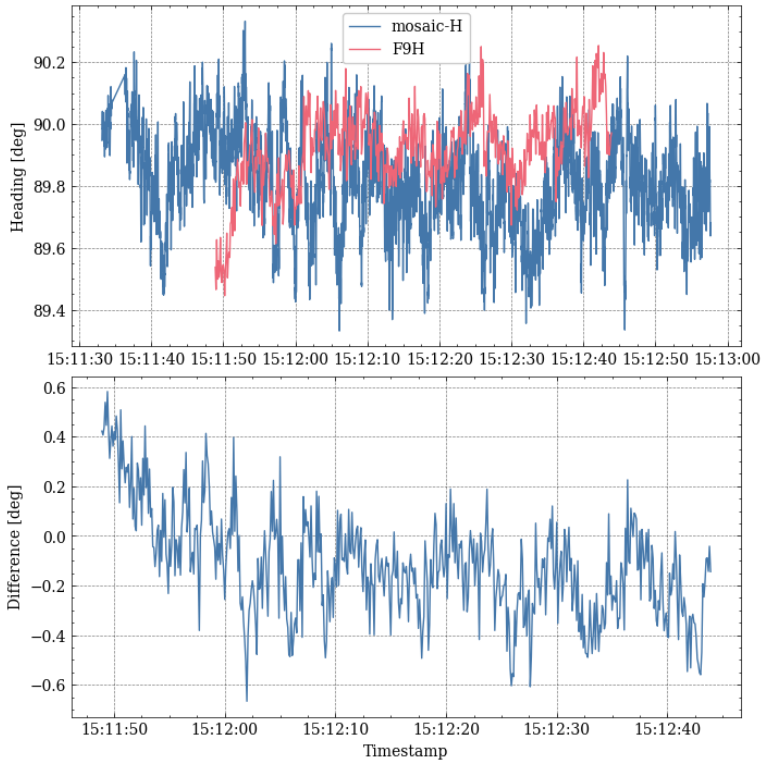



Figure 6.1: Difference between u-blox and Septentrio modules when the antenna alignment is straight east. Both antenna pairs are 75 cm apart of each other. Average difference was 0.2 degrees.

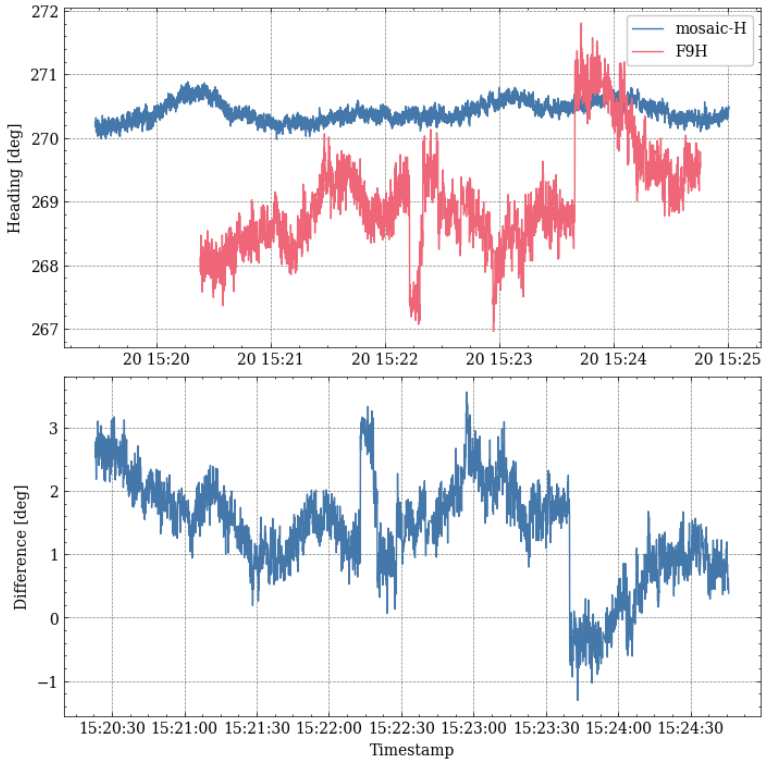



Figure 6.2: Difference between u-blox and Septentrio modules when the antenna alignment is straight west. The antennas for F9H and F9P is 30 cm apart and the antennas for mosaic-H are 90 cm apart from each other. Average difference was 1.456 degrees.

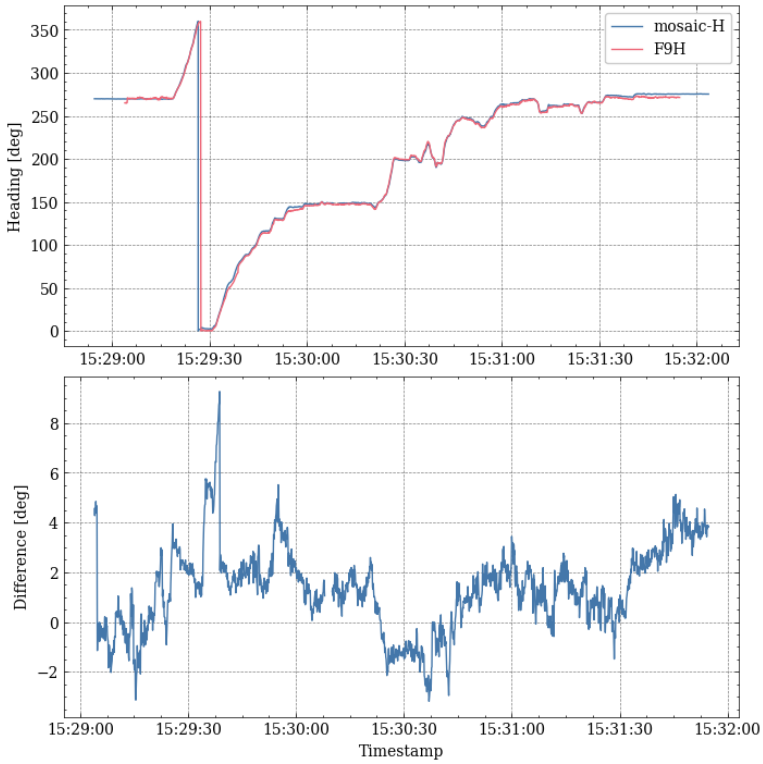



Figure 6.3: Difference between u-blox and Septentrio modules when roting 360 degree yaw. The antennas for F9H and F9P is 30 cm apart and the antennas for mosaic-H are 90 cm apart from each other.

## 6.2 Case 2 - Heading accuracy for u-blox modules with different antenna baselines

In this case we compere the heading and accuracy of different baselines length for the u-blox setup. In Figure 6.4 we see that the accuracy is increasing (lower value) when the baseline length is increasing. The biggest jump of improvements comes when the baseline increases from 30 cm to 45 cm. All of these measurements is gathered when the rig was in the same locked position. The result is similar to test done by u-blox in the data sheet for F9H (ublox, 2021a, fig. 1). My results does not get as good of a accuracy, but shows the same trend.

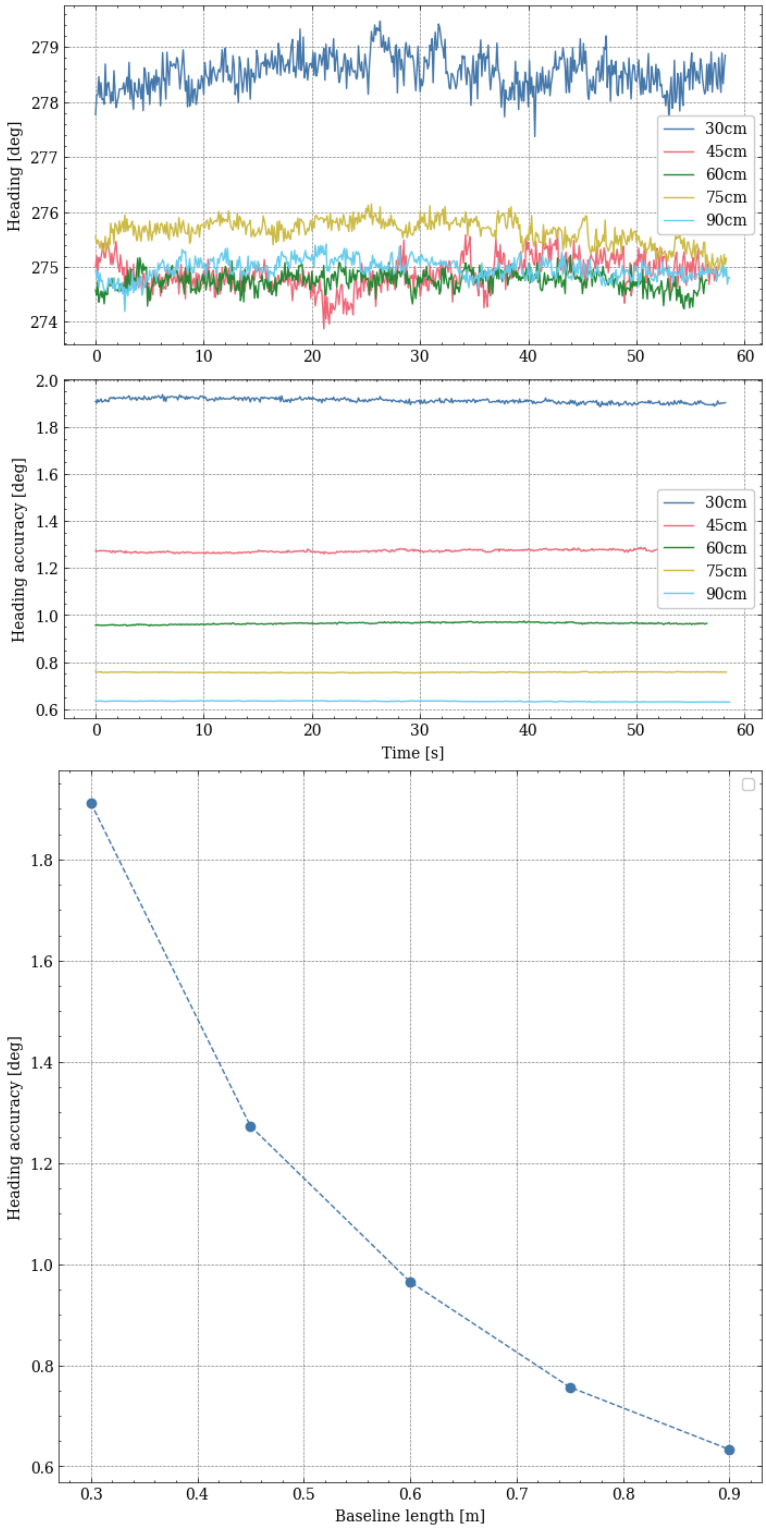



Figure 6.4: Heading accuracy for different baseline lengths between antennas for the F9P and F9H u-blox modules.

## 6.3 Case 3 - Map view of test location with position and heading

This case shows a map view combined of the heading and position on the test location as shown in Figure 6.5. It is an illustration of what and working system can yield of information, also in real time if implemented.

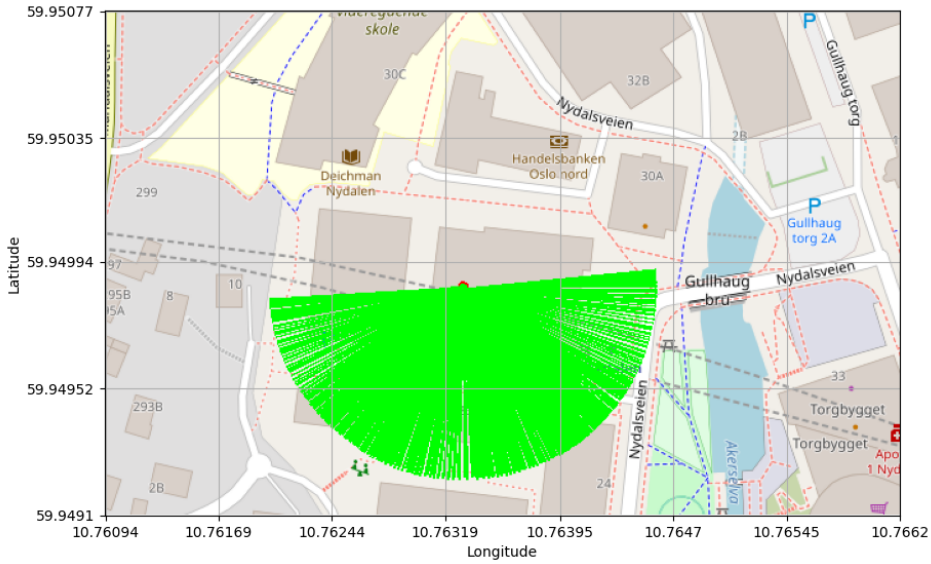



Figure 6.5: Test location at Nydalen with GNSS position and heading covrage from east to west direction (positive rotation in yaw).

## 6.4 Case 4 - Attitude with simulated data

This case illustrate the scenario with perfect accelerometer and gyroscope measurements, and a gyroscope with only a constant bias given by the simulator in Section 5.9. As shown in Figure 6.6, the states stabilize around a value that match the trajectory given by the IMU measurements in Figure 6.7.

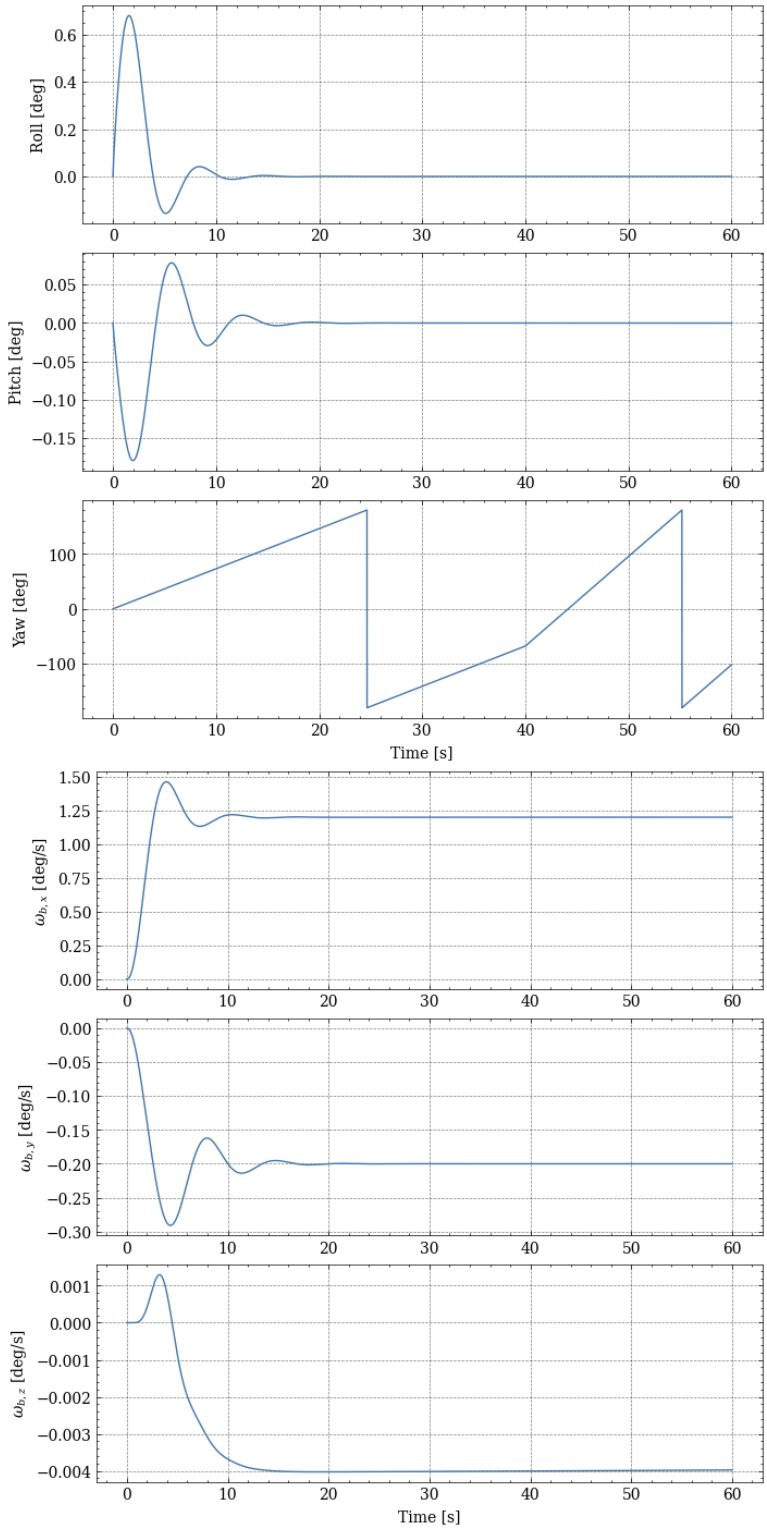



Figure 6.6: State estimates from the ESKF with correction from accelerometer and magnetometer.

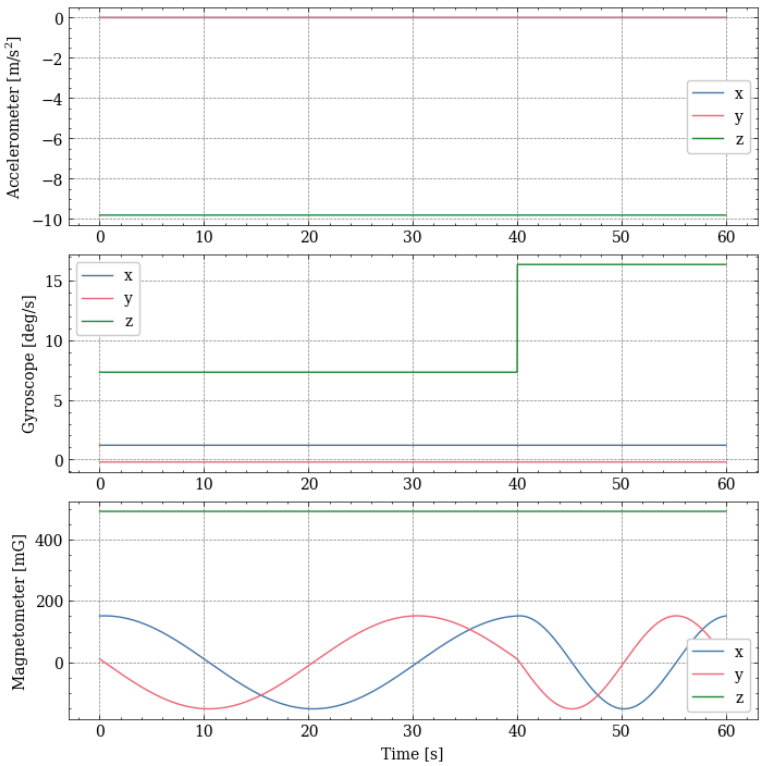



Figure 6.7: Raw measurements data from the IMU simulation.

## 6.5 Case 5 - Attitude with magnetometer heading correction

This case show estimation of attitude and gyroscope bias only using measurements from the IMU. This is done on the same day as the magnetometer was calibrated. In Figure 6.8 we see the estimated state for the movement that is 360 degree rotation round the z-axis. The difference from the heading given by the GNSS solution (u-blox) is noticeable as shown in figure 6.9. The raw IMU data is shown in Figure 6.10.

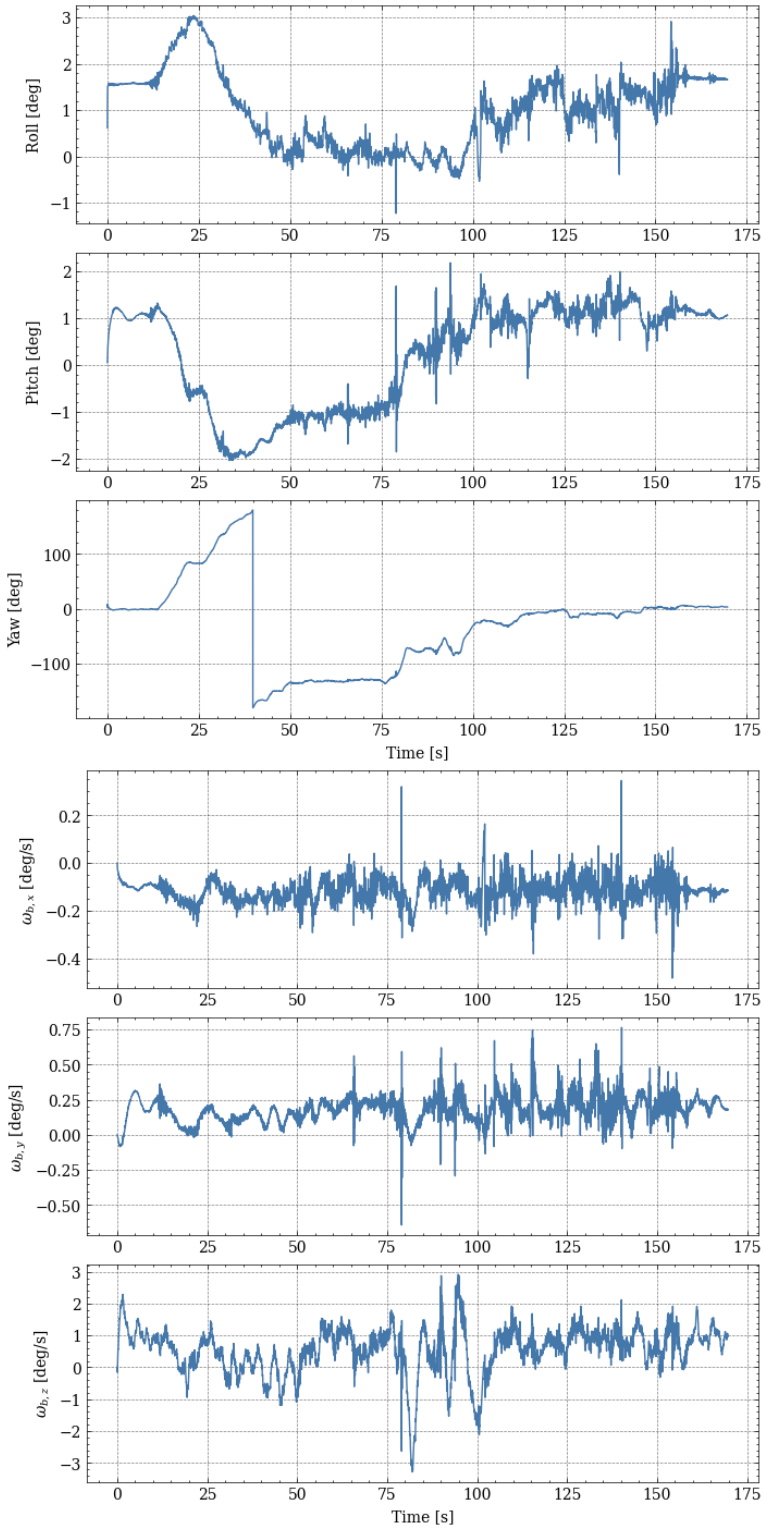



Figure 6.8: State estimates from the ESKF with correction from accelerometer and magnetometer.

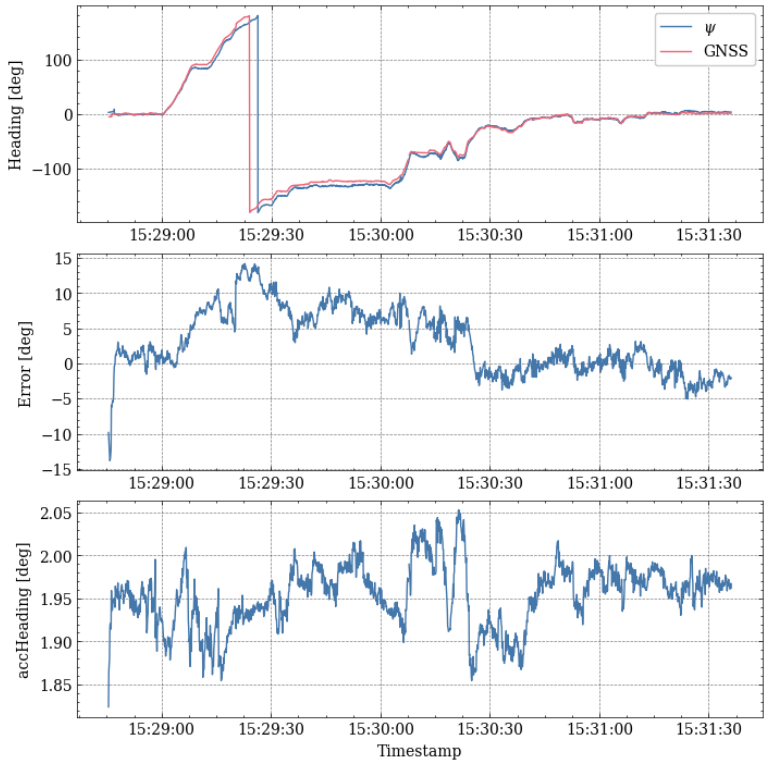



Figure 6.9: Compering estimated yaw from ESKF and GNSS heading.

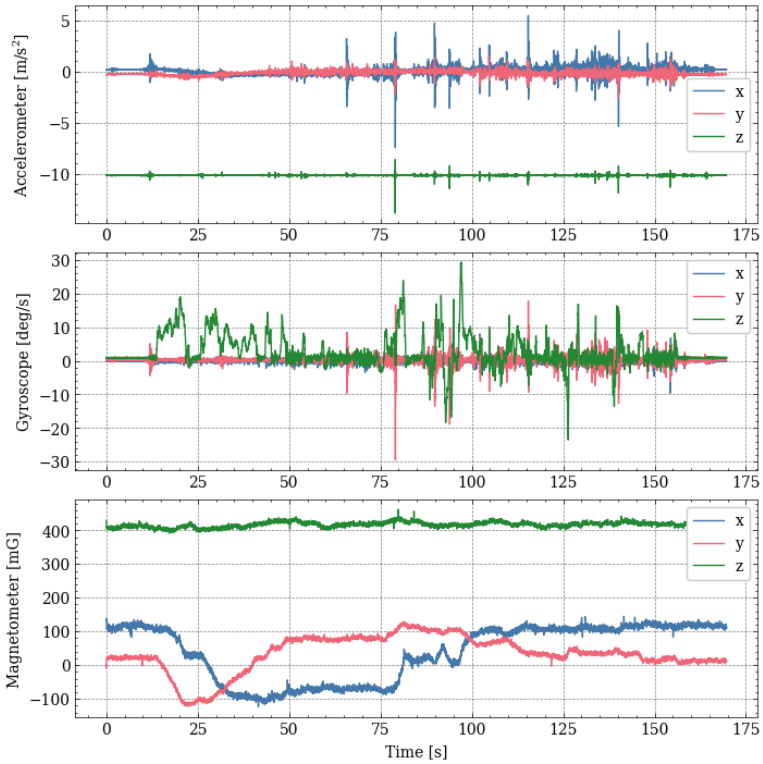



Figure 6.10: Raw measurements data from the IMU.

## 6.6 Case 6 - Attitude with GNSS heading correction

For this case the results show one positive rotation around the z-axis starting with the heading in west direction as the state for this is shown in 6.11 and the IMU data is shown in Figure 6.13. Here the u-blox GNSS modules is used for correction of heading and the magnetometer is discard. When using the GNSS as correction in ESKF and the raw heading yielded from GNSS the error (difference) explicable gets really small as shown in Figure 6.12.

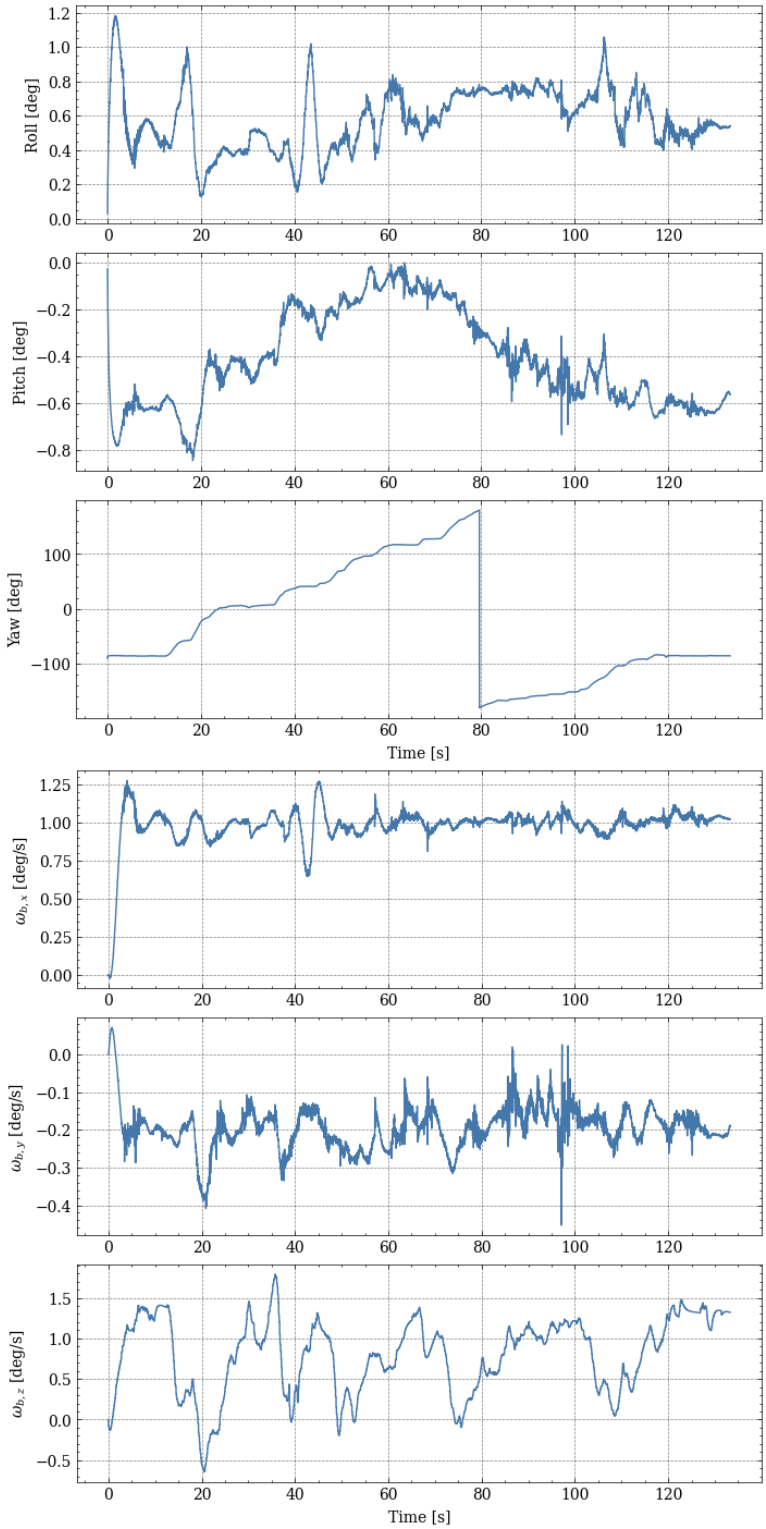



Figure 6.11: State estimates from the ESKF.

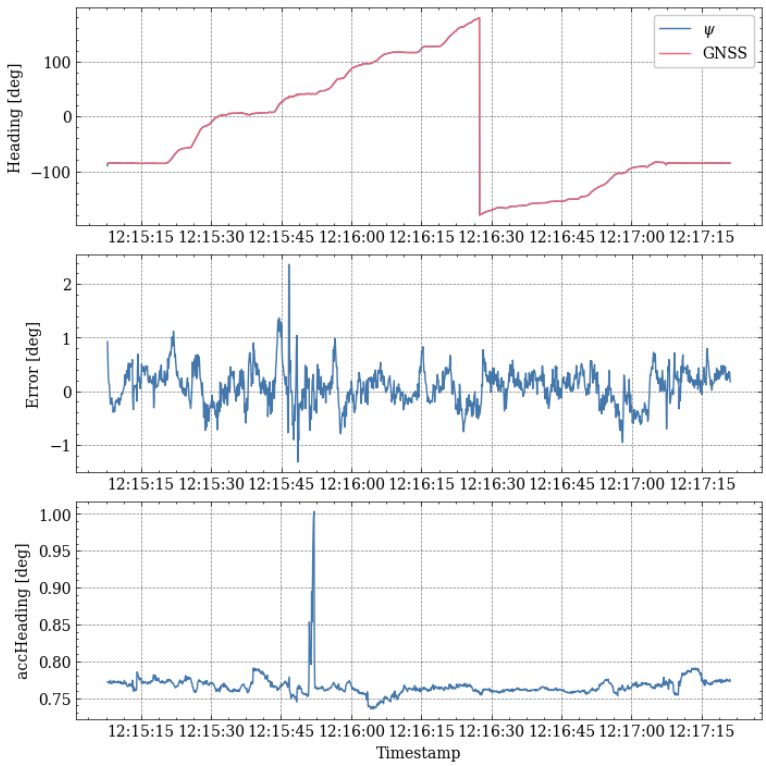



Figure 6.12: Compering estimated yaw from ESKF and GNSS heading.

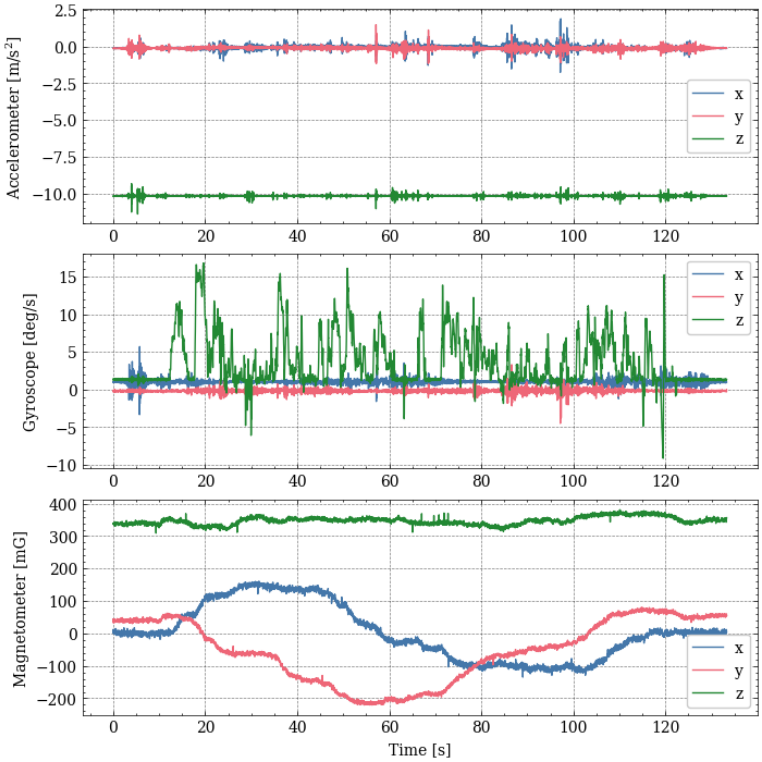



Figure 6.13: Raw measurements data from the IMU.

## 6.7 Case 7 - Attitude with GNSS heading correction

For this case the same correction as in case 6 applies, but with a different movement. Here we first rotate around the x-axis, then z-axis and in the end the y-axis back to idle as shown in Figure 6.14 and 6.16. With movement in roll and pith, yaw for the most time keeps up with the raw heading composed from GNSS as shown in Figure 6.15. The movement in roll and pitch looks like the same movement that was made in the test. This is hard to verify with the current rig setup, and is based on personal observation of the test.

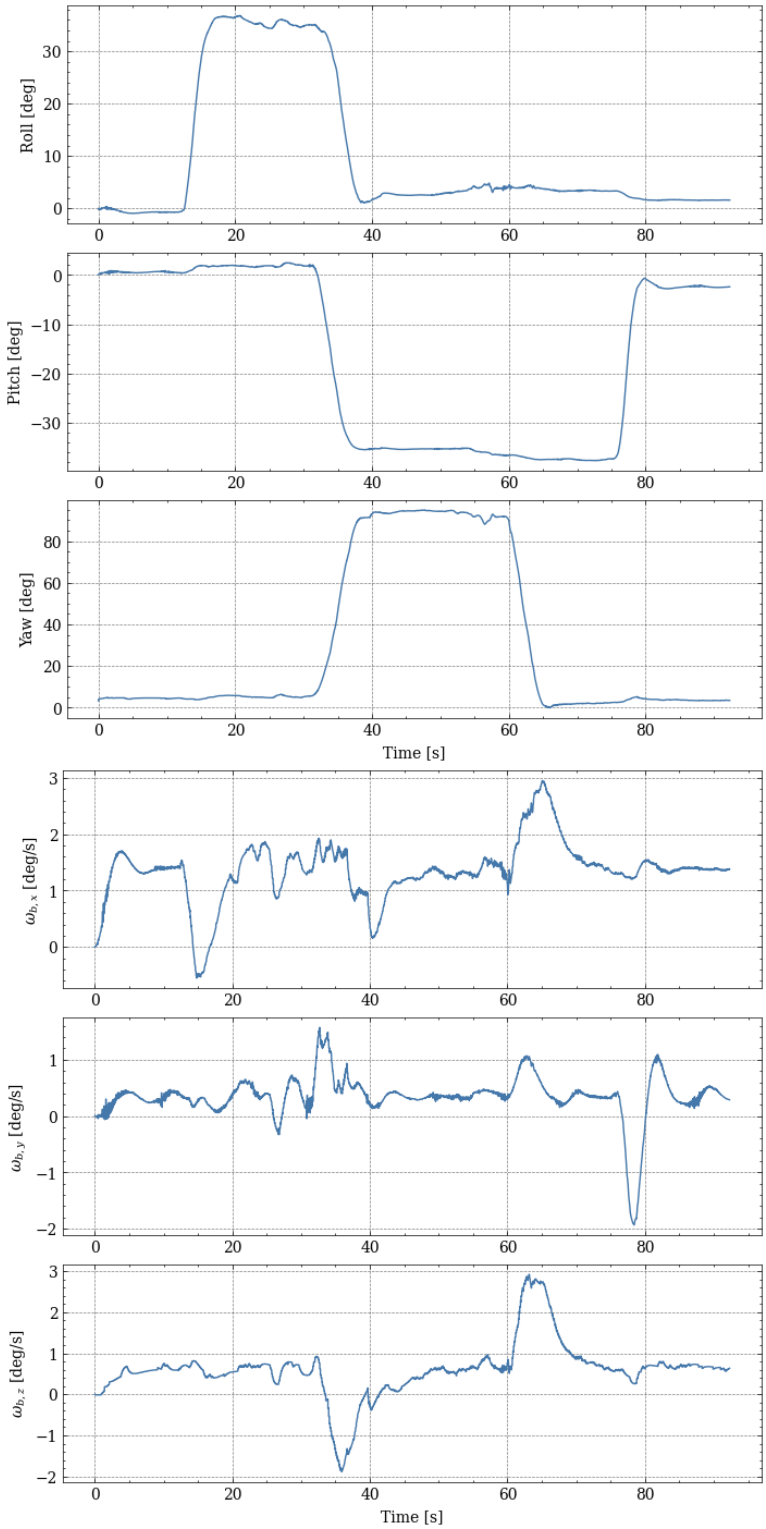



Figure 6.14: State estimates from the ESKF.

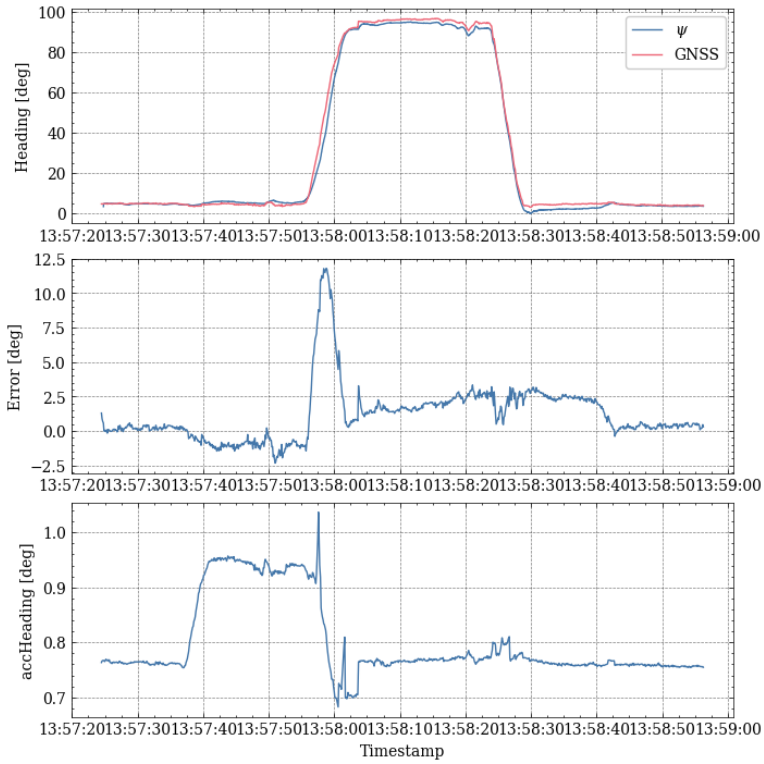



Figure 6.15: Compering estimated yaw from ESKF and GNSS heading.

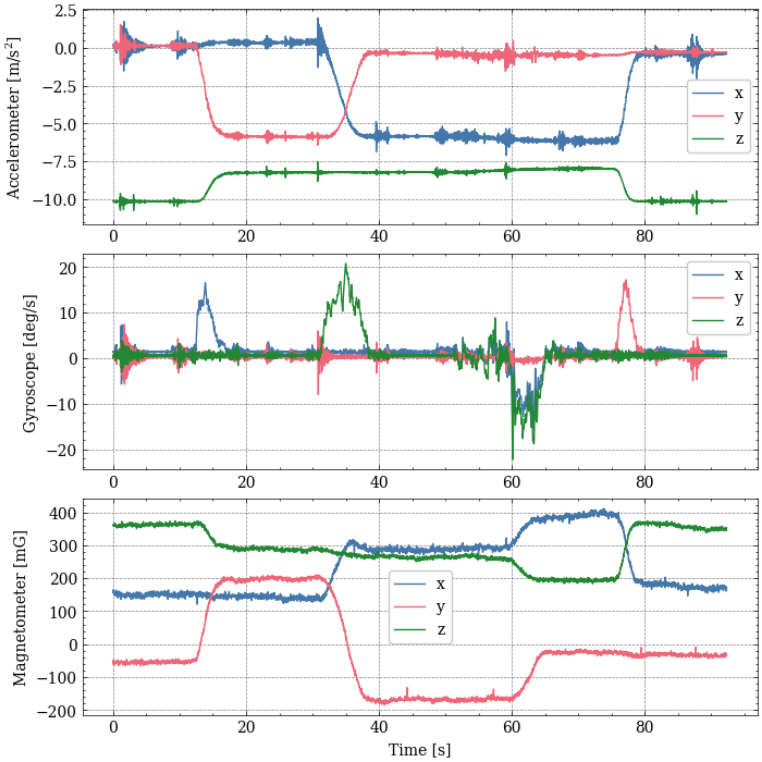



Figure 6.16: Raw measurements data from the IMU.

## 6.8 Case 8 - Attitude with GNSS heading correction

This case is based on the same ESKF correction method as case 6 and 7. Here the 360 degree protractor from Figure 1.4 is used to make 3 step by step of 10 degrees each step around the z-axis. As shown in Figure 6.17 the step of 10 degrees fit well the result. This was done with a antenna baseline of 30 cm as seen with the high value accuracy heading as shown in Figure 6.15 and fits with the baseline in 6.4.

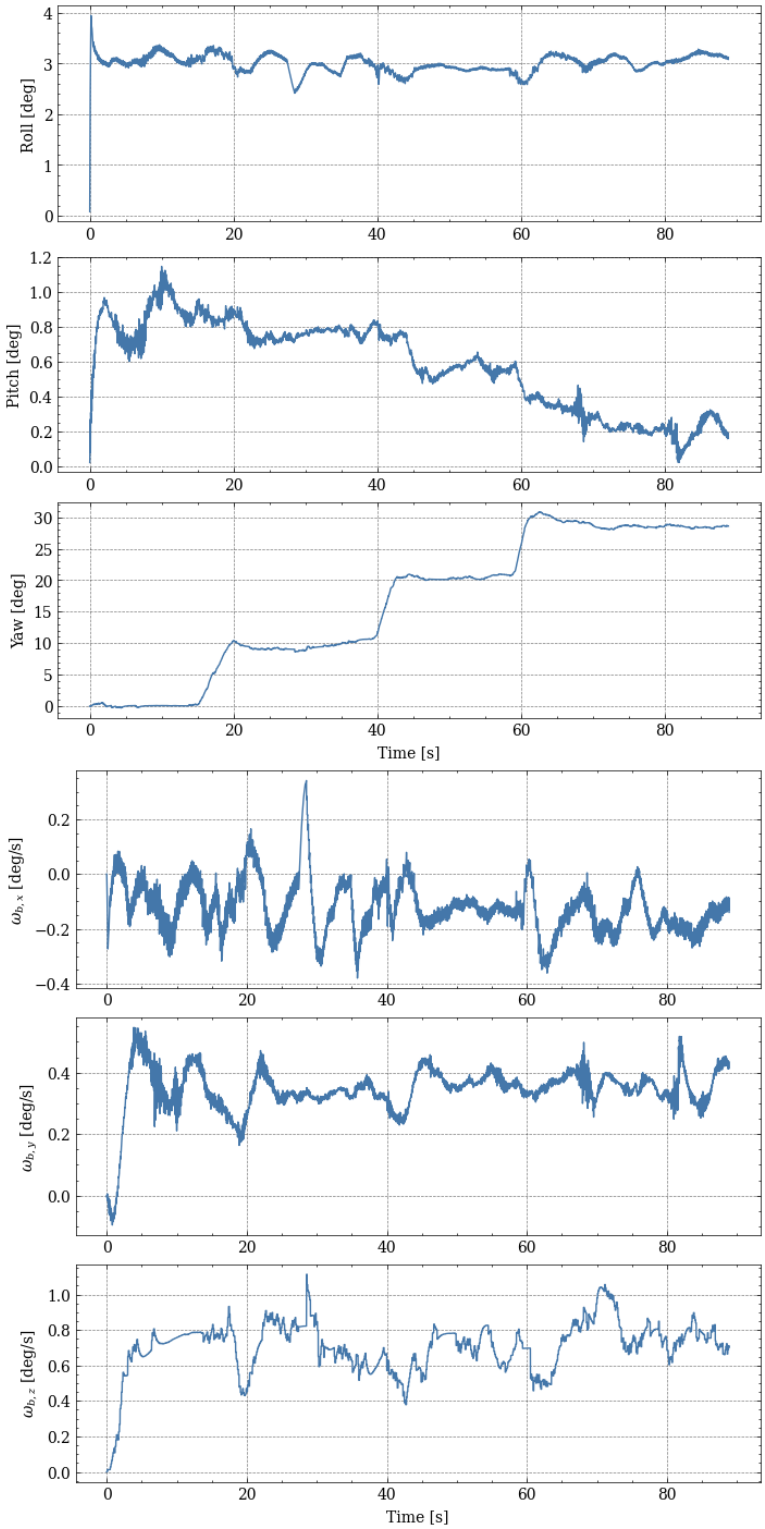



Figure 6.17: State estimates from the ESKF.

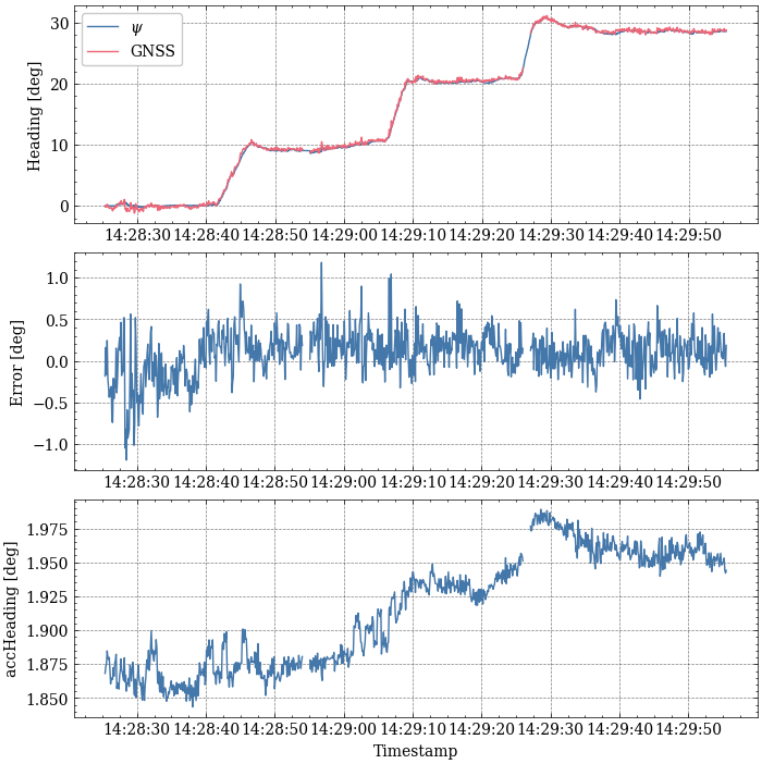



Figure 6.18: Compering estimated yaw from ESKF and GNSS heading.

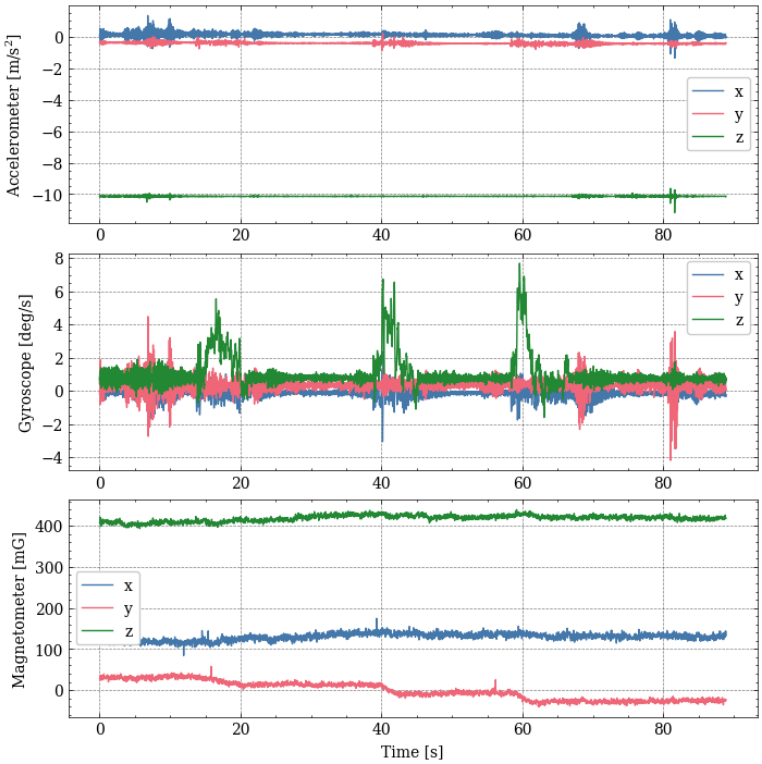



Figure 6.19: Raw measurements data from the IMU.

## 6.9 Case 9 - Attitude comparison between correction with magnetometer and GNSS

This case shows the difference between the ESKF with correction from magnetometer and correction from relative position given by GNSS measurements. It's showcase multiple rotation around all the axes as shown in figure 6.20 and 6.21. Difference in roll and pitch is really low and make sense since both setup use the same accelerometer measurement for correction. For yaw the difference is bigger, special in some instances.

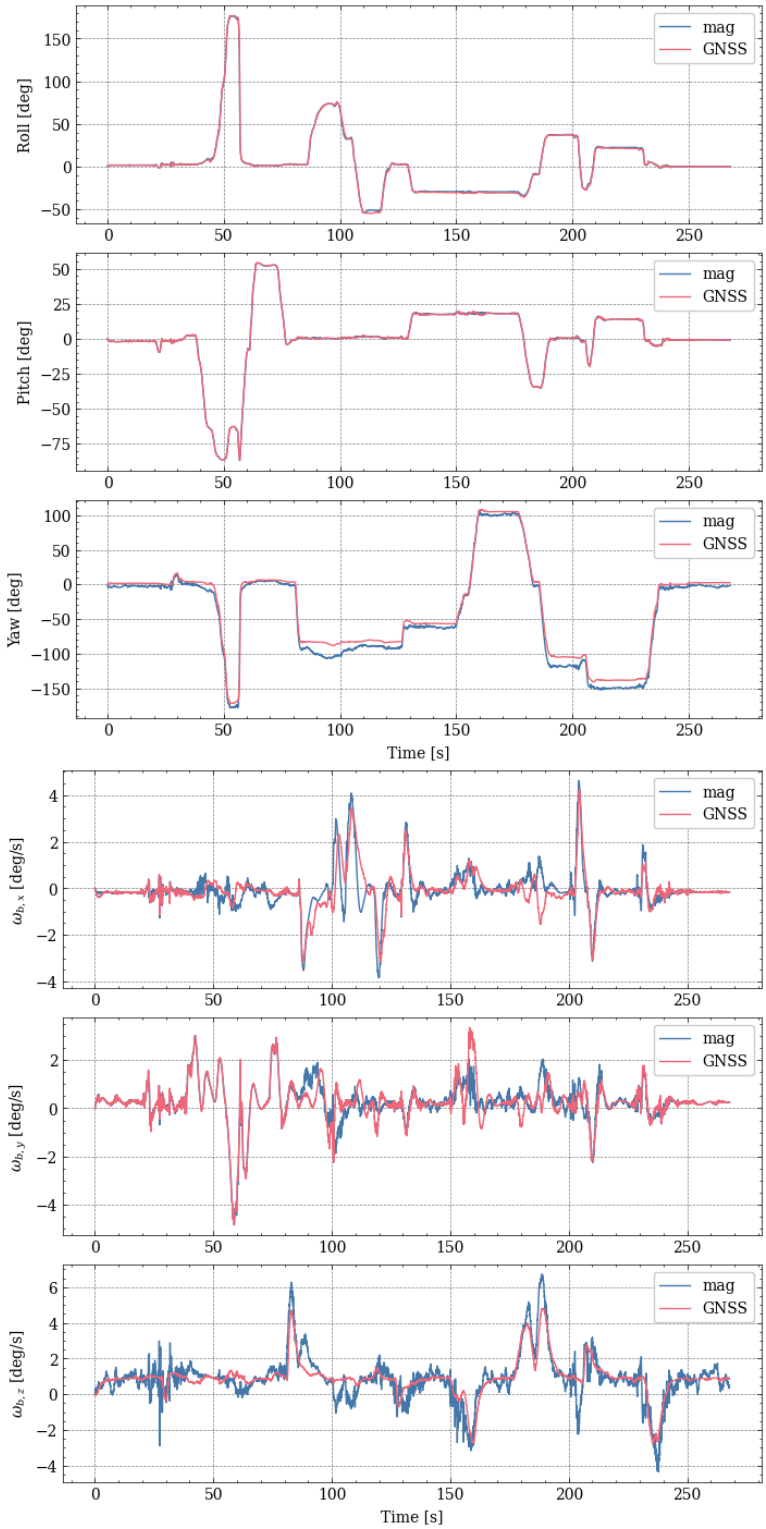



Figure 6.20: State estimates from the ESKF with correction from accelerometer and GNSS RTK, and accelerometer and magnetometer.

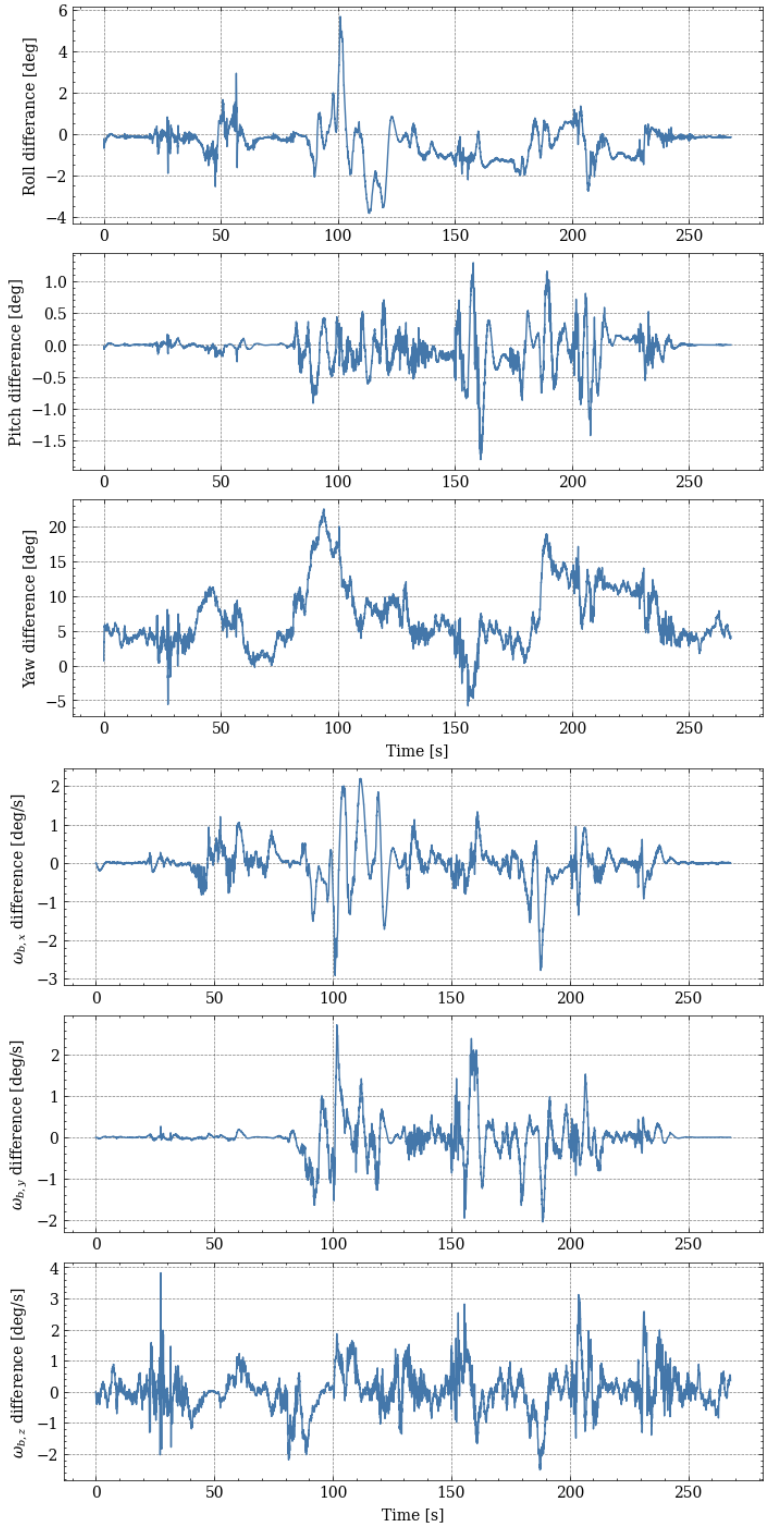



Figure 6.21: Difference in state estimates from the ESKF with correction from accelerometer and GNSS RTK, and accelerometer and magnetometer.

# 7 Discussion

In this project two different GNSS module solutions have been used, one from u-blox and one from Septentrio. Throughout the work pros and cons with both of them have been discovered. The u-blox devices have better documentation and is cheaper then then Septentrio device. But on the other hand the u-blox setup needs two modules, when Septentrio only need one. Septentrio also have the options to yield a sampling rate of 100 Hz if necessary, that is significant higher then when u-blox (yields around 10 Hz). From the first case in the results it's shown then with the same baseline, there is really small difference between the two applications. This is only relative to each other, and the true attitude is not know. If the absolute error in heading should be conducted we need a accurate known reference direction to compere with. This is not obtained in this project. So it's possible that the GNSS solution is wrong based on errors in the antennas that is not discovered. But over all it's safe to say that the GNSS solution yields trustworthy result.

You may argue that it's worth to update to two F9P u-blox devices, since then it is possible to get the actual relative position vector. Then you can calculate the length of this vector and check if it is correct with the length of the baseline between the antennas used. This feature all ready exist in the mosaic-H module. When extracting the relative position vector from base to rover from the Septentrio generated data, the length of this vector was calculating and verified that the length corresponded with the length of the given baseline.

The results shows that an increasing baseline yields better accuracy, as shown from Figure 6.4. It's also shows a problem with the 30 cm baseline that the modules think the relative position is better then what it actually is, shown in the larger distance in the absolute heading compered to the longer baselines. The validation of this result could have been improved if the relative position vector was not normalized because of the F9H module.

When testing outside the magnetometer measurements was not consistent on the same location, where the GNSS was. A lot of magnetic disturbance that you can not have control over could be the case for this, together with only simple magnetometer calibration. It was observed different magnetometer measurements for the same position over different days. This raise the case that if even with perfect calibration, it could be necessary with a new calibration over time, even if the device is standing on the same location and is not moved.

From the result the roll and pitch is much the same for both correction method. This make sense since correction from accelerometer in both cases are used. It should be said that the correction from magnetometer and GNSS also have impact on the other states. This could be the reason the gyroscope bias never fully stabilize around a constant value as seen in the result cases. Tuning of the parameters used also impact this and could have been look deeper into.

# 8 *Conclusion and further work*

From the results it's shown that GNSS heading works well with good satellite coverage. For RTK to work this has to be really good and a sufficient long distance between the antennas is important for good precision. The RTK setup works best when the baseline is at least 45 cm, where 30 cm yields more room for unwanted results. The magnetometer is too unpredictable with the case of sudden change in the local magnetic field, but seems to give a good direction of heading the same day as calibrating on the test site, but not with the same accuracy as the GNSS solution. On the other hand the magnetometer could be used for a back up (only check for changes) if the RTK looses signals, that was observed if moving the rig really close to the building on the test location.

**Further work**

More testing in the field for different scenarios and testing on different locations and with different hardware mounted on the rig. Get a better idea of which environments the RTK can not yield a valid relative position vector. Also make a real time application with a GUI displaying the estimated attitude in real time. More investigation around tuning of every parameter is also needed.

# *References*


Ardusimple (2022). Centimeter precision gps/gnss - rtk explained. (Accessed: 10.01.2022).
**URL:** *https://www.ardusimple.com/rtk-explained/*

Brekke, E. (2020). *Fundamentals of Sensor Fusion: Target tracking, navigation and SLAM*. Norwegian University of Science and Technology, third edition.

Egeland, O. and Gravdahl, J. T. (2003). *Modeling and Simulation for Automatic Control*. Marine Cybernetics AS, second edition. ISBN: 82-92356-01-0.

EndRunTechnologies (2022). Leap seconds. (Accessed: 04.01.2022).
**URL:** *https://endruntechnologies.com/support/leap-seconds*

Farrell, J. A. (2008). *Aided Navigation: GPS with High Rate Sensors*. The McGraw-Hill Companies, first edition. doi: 10.1036/0071493298.

Fossen, T. I. (2011). *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons Ltd., first edition. ISBN: 978-1-119-99149-6.

MathWorks (2021). Navigation toolbox. (Accessed: 13.12.2021).
**URL:** *https://se.mathworks.com/help/nav/index.html?s_tid=CRUX_lftnav*

pySerial (2022). pyserial: Python serial port access library. Version 3.5. (Accessed: 10.01.2022).
**URL:** *https://github.com/pyserial/pyserial*

RTKLIB (2022). Rtklib: An open source program package for gnss positioning. Version 2.4.2. (Accessed: 10.01.2022).
**URL:** *http://www.rtklib.com/*

semuconsulting (2022). pyubx2: Python library for parsing and generating ubx gps/gnss protocol messages. Version 1.1.5. (Accessed: 10.01.2022).
**URL:** *https://github.com/semuconsulting/pyubx2*

Septentrio (2021a). mosaic-h – gnss module with heading capability. (Accessed: 21.06.2021).
**URL:** *https://www.septentrio.com/en/products/gnss-receivers/rover-base-receivers/receivers-modules/mosaic-h*

Septentrio (2021b). Rxtools: Gnss receiver control and analysis software. (Accessed: 16.11.2021).
**URL:** *https://www.septentrio.com/en/products/software/rxtools*

Solà, J. (2017). *Quaternion kinematics for the error-state Kalman filter*. Cornell University, v1 edition. (Accessed: 25.02.2021).
**URL:** *https://arxiv.org/pdf/1711.02508.pdf*

STMicroelectronics (2010). Using lsm303dlh for a tilt compensated electronic compass. Rev 1. (Accessed: 3.01.2022).
**URL:** *https://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf*

STMicroelectronics (2015). Inemo inertial module: 3d accelerometer, 3d gyroscope, 3d magnetometer. LSM9DS1 datasheet revision 3. (Accessed: 01.03.2021).
**URL:** *https://www.st.com/resource/en/datasheet/lsm9ds1.pdf*

Tysseland, M. (2021). *Attitude determination of a relative position sensor*. Norwegian University of Science and Technology, first edition.

ublox (2020a). Zed-f9h interface description. (Accessed: 13.12.2021).
**URL:** *https://www.u-blox.com/en/product/zed-f9h-module#tab-documentation-resources*

ublox (2020b). Zed-f9p interface description. (Accessed: 13.12.2021).
**URL:** *https://www.u-blox.com/en/product/zed-f9p-module#tab-documentation-resources*

ublox (2021a). Zed-f9h-01b data sheet. (Accessed: 13.12.2021).
**URL:** *https://www.u-blox.com/en/product/zed-f9h-module#tab-documentation-resources*

ublox (2021b). Zed-f9p-04b data sheet. (Accessed: 13.12.2021).
**URL:** *https://www.u-blox.com/en/product/zed-f9p-module#tab-documentation-resources*

ublox (2021c). Zed-f9p integration manual. (Accessed: 13.12.2021).
**URL:** *https://www.u-blox.com/en/product/zed-f9p-module#tab-documentation-resources*

WMM (2021). Online calculators for the world magnetic model. (Accessed: 13.12.2021).
**URL:** *https://www.ngdc.noaa.gov/geomag/WMM/calculators.shtml*

Zinn, N. (2018). Characterizing an imu for a raspberry pi. (Accessed: 18.03.2021).
**URL:** *https://www.linkedin.com/pulse/characterizing-imu-raspberry-pi-noel-zinn/*