SSY345 - Project: Orientation estimation using smartphone sensors

Nicholas Granlund, nicgra@chalmers.se
Louise Olsson, louolsso@chalmers.se

Introduction

This report concludes the final project in the course *SSY345 Sensor fusion and non-linear filtering*. The aim of this project is to utilize the tools we have obtained throughout the course to implement a non-linear filter for orientation estimation using smartphone sensors. The filter will not track translations and the focus us thus to track the absolute orientation of the smartphone. This project will use a iOS based smartphone along with the Matlab application available from the appstore.

Orientation estimation approach

The approach in this project will be to estimate the orientation of the smartphone using quaternions. Other common methods are to use Euler-angles but these are susceptible to the phenomenon known as gimbal-lock where one degree of freedom is lost. Furthermore, the coordinate system used to describe the phones orientation is defined according to the right-hand rule - which is illustrated in Figure 0.1. The positive x-axis points right, tangential to the screen. The positive y-axis points straight forward, tangential to the screen and perpendicular to the x-axis. The positive-z axis points orthogonal from the screen and perpendicular to all other axis.

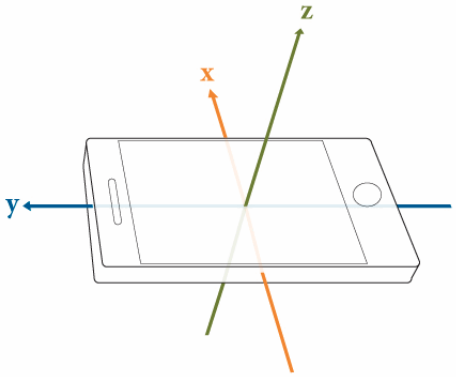


Figure 0.1: Coordinate system used for the phone. Photo reworked from Mathworks.

1 Inputs or outputs (measurements)?

1.1 Using gyroscope measurements as inputs ($u_k = \omega_k$)

An advantage of using the gyroscope measurements as inputs for our system is that it reduces the computational complexity of our system. This means that we implement fewer parameters inside our dynamics matrix A.

However, by doing so we essentially assume that the gyroscope measurements are noise free, which means that we expect the gyroscope to yield deterministic measurements rather

than probabilistic. This assumption may be valid for small amounts of noise, but fails for larger noise-levels.

In that case, it would be preferable to introduce the gyroscope measurements into the dynamics matrix A since it will allow us to model the disturbances and once again treat the measurements as probabilistic rather than deterministic.

2 Data collection and filter design

2.1 Gathering calibration data

By gathering data from the sensors when the body, smartphone, is at rest, we can obtain valuable information of how the sensors may be biased and how the noise is distributed. This will later be helpful when tuning the filter.

The calibration data was obtained by an *iPhone 12 Pro, 2020*. The phone was placed at rest, on a surface that presumably was dead-flat, away from other electrical components and was pointed towards 0° north w.r.t the phones y-direction. The phone then gathered data for around 62 seconds. The sensor mean and covariance from the static measurements are compiled in Table 1-2.

Table 1: Mean value of smartphone sensors

	Accelerometer	Gyroscope	Magnetometer
μ_x	-0.0012	-0.0013	-0.8277
μ_y	0.0579	0.0069	12.1241
μ_z	9.7941	-0.0023	-57.7562

Table 2: Covariance of smartphone sensors

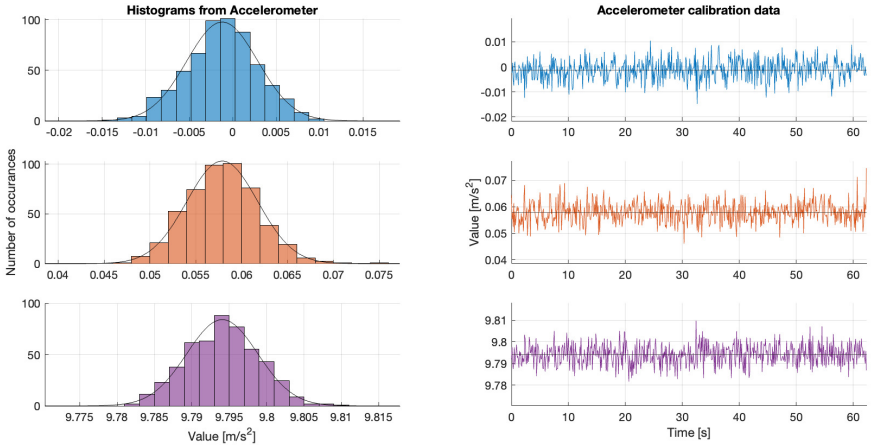
	Accelerometer	Gyroscope	Magnetometer
σ_x^2	$1.6686e-05$	$2.2433e-06$	$0.0395e-00$
σ_y^2	$1.4947e-05$	$2.0894e-06$	$0.0333e-00$
σ_z^2	$2.2537e-05$	$1.9702e-06$	$0.1085e-00$

The histograms describing the noise distribution can be seen in Figure 2.1 for the accelerometer, gyroscope and magnetometer in (a)-(c) respectively. A short summary of the observations:

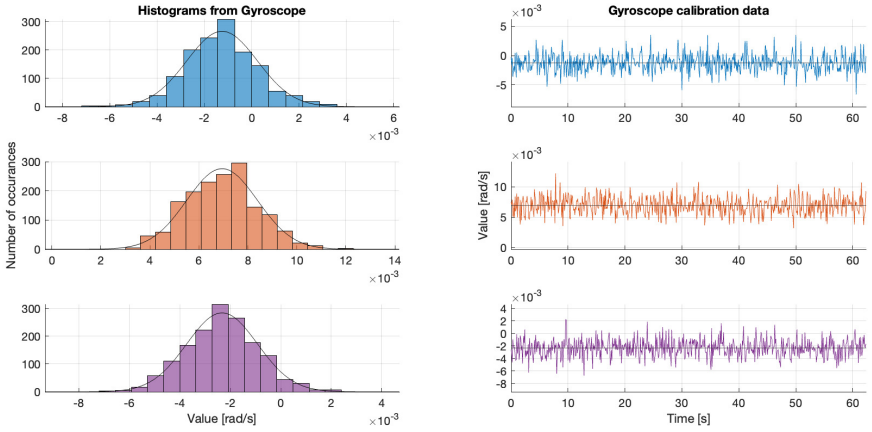
Accelerometer
There is some bias in the measurements that are rather close to zero. The accelerometer measures the gravitational pull in the z-axis as positive, since this acceleration would make the body be at rest. The noise can be well described by a Gaussian distribution.

Gyroscope
There is some bias in the gyroscope, but similar to the accelerometer they are sufficiently close to zero to be considiered unbiased. The distribution of the noise can also be described by a gaussian, however the amount of bins in the histogram do not make this directly apparent.

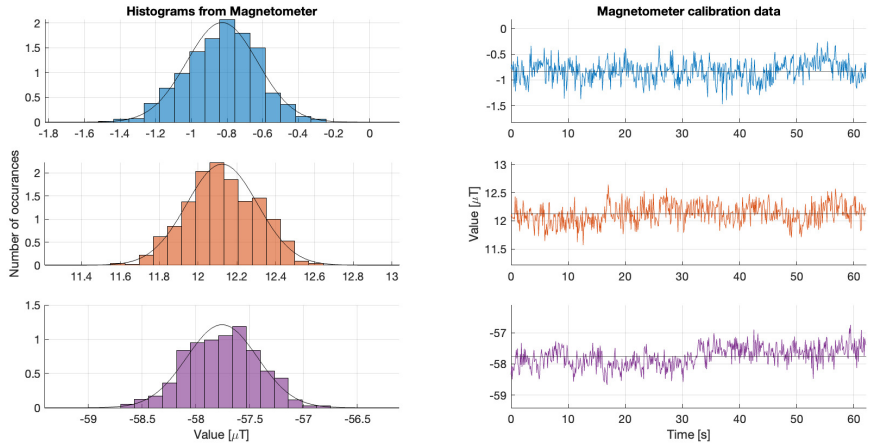
Magnetometer
There is more bias in the magnetometer measurements. It should be emphasized that the phone was pointed 0° north when the data was gathered. The distribution of the measurements is more non-Gaussian than the other sensor which is proven by the histograms. One side of the mean occurs more often than the other side, resulting in a distribution that is better described by a *gamma-distribution*.



(a) Accelerometer calibration data



(b) Gyroscope calibration data



(c) Magnetometer calibration data

Figure 2.1: Histogram graphs and measured data from accelerometer, gyroscope and magnetometer. Data was gathered for 62 seconds.

3 Design the EKF time update step

3.1 Solve the differential equation

We have the differential equation:

$$\dot{q}(t) = \frac{1}{2} S(\omega^{k-1} + v^{k-1}) q(t) \tag{1}$$

Where v^{k-1} is additive noise. We use the the fact that $S(a+b) = S(a) + S(b)$ and can expand the differential equation to:

$$\dot{q}(t) = \frac{1}{2} S(\omega^{k-1}) q(t) + \frac{1}{2} S(v^{k-1}) q(t) \tag{2}$$

Now if we further expand this expression, we obtain the system of equations:

$$\begin{bmatrix} \dot{q}_0(t) \\ \dot{q}_1(t) \\ \dot{q}_2(t) \\ \dot{q}_3(t) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{bmatrix} \begin{bmatrix} q_0(t) \\ q_1(t) \\ q_2(t) \\ q_3(t) \end{bmatrix} + \frac{1}{2} \begin{bmatrix} 0 & -v_x & -v_y & -v_z \\ v_x & 0 & v_z & -v_y \\ v_y & -v_z & 0 & v_x \\ v_z & v_y & -v_x & 0 \end{bmatrix} \begin{bmatrix} q_0(t) \\ q_1(t) \\ q_2(t) \\ q_3(t) \end{bmatrix} \tag{3}$$

Further we will only consider the first matrix-vector multiplication and reason that the same can be applied for the rest of the expression. If we perform the matrix-vector multiplication we obtain the four equations

$$\begin{aligned} \dot{q}_0(t) &= \tfrac{1}{2}\big(- \omega_x q_1(t) - \omega_y q_2(t) - \omega_z q_3(t)\big) + ... \\ \dot{q}_1(t) &= \tfrac{1}{2}\big(\omega_x q_0(t) + \omega_z q_2(t) - \omega_y q_3(t)\big) + ... \\ \dot{q}_2(t) &= \tfrac{1}{2}\big(\omega_y q_0(t) - \omega_z q_1(t) + \omega_x q_3(t)\big) + ... \\ \dot{q}_3(t) &= \tfrac{1}{2}\big(\omega_z q_0(t) + \omega_y q_1(t) - \omega_x q_2(t)\big) + ... \end{aligned} \tag{4}$$

Now we can use forward different approximation for the time derivative to obtain an approximation of the expression. We will use the superscript $k-1$ and k to show at which discrete time instance the measurement is obtained from. This gives:

$$\begin{aligned} \tfrac{q_0^k - q_0^{k-1}}{\Delta t} &\approx \tfrac{1}{2}\big(- {\omega_x}^{k-1} q_1^{k-1} - {\omega_y}^{k-1} q_2^{k-1)} - {\omega_z}^{k-1} q_3^{k-1}\big) + ... \\ \tfrac{q_1^k - q_1^{k-1}}{\Delta t} &\approx \tfrac{1}{2}\big({\omega_x}^{k-1} q_0^{k-1} + {\omega_z}^{k-1} q_2^{k-1} - {\omega_y}^{k-1} q_3^{k-1}\big) + ... \\ \tfrac{q_2^k - q_2^{k-1}}{\Delta t} &\approx \tfrac{1}{2}\big({\omega_y}^{k-1} q_0^{k-1} - {\omega_z}^{k-1} q_1^{k-1} + {\omega_x}^{k-1} q_3^{k-1}\big) + ... \\ \tfrac{q_3^k - q_3^{k-1}}{\Delta t} &\approx \tfrac{1}{2}\big({\omega_z}^{k-1} q_0^{k-1} + {\omega_y}^{k-1} q_1^{k-1} - {\omega_x}^{k-1} q_2^{k-1}\big) + ... \end{aligned} \tag{5}$$

By isolating the term for time instance k we get the discrete time approximation:

$$\begin{aligned} q_0^k &\approx q_0^{k-1} + \tfrac{\Delta t}{2}\big(- {\omega_x}^{k-1} q_1^{k-1} - {\omega_y}^{k-1} q_2^{k-1)} - {\omega_z}^{k-1} q_3^{k-1}\big) + ... \\ q_1^k &\approx q_1^{k-1} + \tfrac{\Delta t}{2}\big({\omega_x}^{k-1} q_0^{k-1} + {\omega_z}^{k-1} q_2^{k-1} - {\omega_y}^{k-1} q_3^{k-1}\big) + ... \\ q_2^k &\approx q_2^{k-1} + \tfrac{\Delta t}{2}\big({\omega_y}^{k-1} q_0^{k-1} - {\omega_z}^{k-1} q_1^{k-1} + {\omega_x}^{k-1} q_3^{k-1}\big) + ... \\ q_3^k &\approx q_3^{k-1} + \tfrac{\Delta t}{2}\big({\omega_z}^{k-1} q_0^{k-1} + {\omega_y}^{k-1} q_1^{k-1} - {\omega_x}^{k-1} q_2^{k-1}\big) + ... \end{aligned} \tag{6}$$

Now, the original expression had the form $\dot{q} = \frac{1}{2}S(\omega)q$. And since $S(\omega)$ is not directly a function of time, we can treat this as a constant. The differential equation solution for such a system is hence $q = e^{\frac{1}{2}S\Delta t}$. And by using the approximation $e^A \approx I + A$ we get the solution $q^k \approx \left(I + \frac{\Delta t}{2}S\right)q^{k-1}$ which is the same as we have derived above in expression (6).

Further, if we consider the noise term of expression (2) we have the equality that $S(a)b = \bar{S}(b)a$. This allows us to rewrite the second term in expression (2) to be a function of the quaternions multiplied with the outer disturbances v. To summarize, this gives us the complete expression

$$\mathbf{q}^k = \left(\mathbf{I} + \frac{\Delta t}{2}S(\omega^{k-1})\right)\mathbf{q}^{k-1} + \left(\frac{\Delta t}{2}\bar{S}(\mathbf{q}^{k-1})\right)v^{k-1} \tag{7}$$

Or on the desired form:

$$\mathbf{q}^k = F(\omega^{k-1})\mathbf{q}^{k-1} + G(\mathbf{q}^{k-1})v^{k-1} \tag{8}$$

Now, we do not want G to be a function of the true quaternions states $\mathbf{q}^{k-1}$. We will thus approximate $G(\mathbf{q}^{k-1}) \approx G(\hat{\mathbf{q}}^{k-1})$. This approximation is valid since the EKF work on linearization around the previous time step estimation.

3.2 Time update function

The function was implemented in Matlab and can be seen in the function file tu_qw.m. For convenience the code is appended and can be found in the listing below.

```
function [x, P] = tu_qw(x, P, omega, T, Rw)
    % TU_QW: Time update for quaternion given
    % gyroscope measurements.
    %
    % By: Nicholas Granlund
    %     Louise Olsson
    %
    %Input:
    %   - x             [4x1]    State vector (quaternion)
    %   - P             [4x4]    State covariance matrix
    %   - omega         [3x1]    Angular velocity vector
    %   - T                      Time step
    %   - Rw            [3x3]    Process noise covariance
       matrix
    %
    %Output:
    %   - x             [3x1]    State vector (quaternion)
    %   - P             [4x4]    State covariance matrix

    % Compute x^{k} = F(w^{k-1))q^{k-1} + G(q^{k-1})v^{k-1}
    F = eye(4) + (T/2)*(Somega(omega));
    G = (T/2)*Sq(x);

    % x and P
    x = F*x;
    P = F*P*F' + G*Rw*G';

    % Normalize the quaternion
    [x, P] = mu_normalizeQ(x, P);

    % return

end
```

In the case when there is no new gyroscope measurement available, the same function will be called. Unlike the ordinary case however, the measurement will be set to 0, indicating that the quaternion state has not changed since the last measurement. This will however increase P, the covariance implying that the uncertainty has increased.

3.3 EKF without measurement updates

By utilizing the function described above we are able to get an estimate of the orientation of the smartphone. However, since the filter at this point only uses the gyroscope measurements to estimate the orientation, it cannot obtain an absolute orientation but only relative the starting position. Here follows a summary of the observations:

- Starting the phone pointing directly north the estimate will follow the google/matlab estimate rather well untill the point where the gyroscope drift becomes too aparent.
- Starting the phone pointing in any other direction than true north will yield an estimating that lacks absolute orientation.
- Shaking the phone will make the estimate lose track of the true orientation and the resulting estimate is significantlly worse than the Google/matlab counterpart. This is because the estimate is computed more or less as an integral of the angular velocities -and if the measurments are not valid and/or the samplig time is to small, this will generate a wrong estimate.

4 An EKF update using accelerometer measurements

4.1 Write accelerometer measurement update

The accelerometer measurement update was implemented in the function `mu_g` which can be found in the listing below. The given functions `Qq` and `DQqdq` were used to calculate the measurement model in the function. The algorithm for calculating the EKF update step was then implemented.

```
function [x, P] = mu_g(x, P, yacc, Ra, g0)
    % MU_G Measurement update given accelerometer data
    %
    % By: Nicholas Granlund
    %     Louise Olsson
    %
    %Input:
    %   - x             [4x1]       State vector (quaternions)
    %   - P             [4x4]       Covariance
    %   - yacc          [3x1]       Accelerometer measurement
    %   - Ra            [3x3]       Measurement noise
    %   - g0            [3x1]       Nominal gravity vector
    %
    %Output:
    %   - x             [3x1]     State vector (quaternion)
    %   - P             [4x4]     State covariance matrix

    % Make sure vectors has right dimension
    g0 = reshape(g0, [3,1]);
    fa = zeros(size(g0,1),1);

    % Measurement model
    hx = Qq(x)'*(g0+fa);

    % dH/dq
    [Q0, Q1, Q2, Q3] = dQqdq(x);

    % Measurement model jacobian Hx
    Hx(:,1) = Q0'*(g0+fa);
    Hx(:,2) = Q1'*(g0+fa);
    Hx(:,3) = Q2'*(g0+fa);
    Hx(:,4) = Q3'*(g0+fa);

    % Innovation Covariance S
    S = Hx*P*Hx'+Ra;

    % Kalman Gain K
    K = P*Hx'*inv(S);

    % Update state estimate
    x = x + K*(yacc-hx);

    % Update covariance estimate
    P = P - K*S*K';

    % Normalize the quaternion
    [x, P] = mu_normalizeQ(x, P);

    % return
end
```

4.2 Implementation in EKF

When applying the accelerometer to the filter, the performance is increased. The gyroscope will make the estimation drift over time due to integration. By implementing the accelerometer, the filter will manage to fuse the two sensors together and decrease the drift that occurs. The accelerometer is however not able to estimate the yaw since there is no changes in the accelerations for that rotation, but the gyroscope can. The most apparent improvement is the filters ability to track the tilt relative the nominal gravity vector.

By rotating the phone a lot, the accelerometer and the gyroscope will together give a good estimate of the orientation. The only thing that these two sensors are not the greatest to estimate is the yaw. The gyroscope will estimate it with respect to its own body. The gyroscope will not take into account the world coordinate frame and therefor when compared with the corresponding Google estimate it will deviate.

By translating the phone on a horizontal plane, it will yield faulty estimations for the angles, as displayed in Figure 4.1. The phone starts to slide back and forth quickly on a table. Since the filter is assuming that $f_k^a = 0$, there should not be any acceleration in the world frame. As the figure shows, the filter will therefor yield faulty estimation in pitch when translation is applied, confirming that the filter will not handle this well.

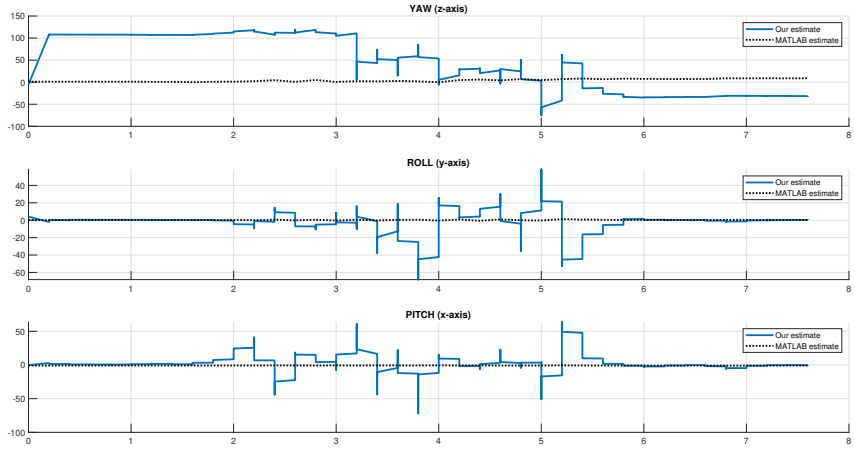


Figure 4.1: Result sliding the phone back and forth on stable surface

4.3 Outlier rejection for big accelerations

The outlier rejection algorithm was implemented in the function `filterTemplateiOS`. To exclude to the measurement outliers, the accelerometer measurements are only used when they are in the interval $[9.81 \cdot 0.8 \quad 9.81 \cdot 1.2]$. When trying the same test as in the last task, i.e. sliding the phone back and forth quickly, it will perform better. The result of the test is visible in Figure 4.2. In the figure it is possible to see that the estimated angles will be less corrupted by the translation, thus resulting in reduced values for the distorted angles.

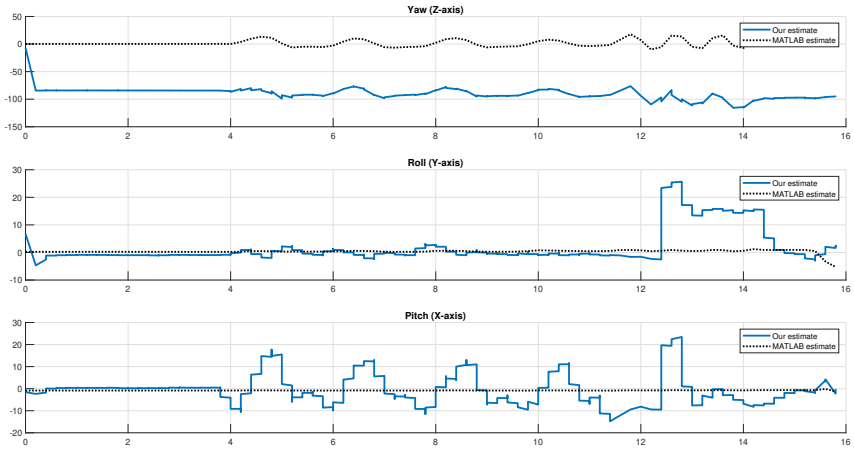


Figure 4.2: Result sliding the phone back and forth on stable surface with outlier rejection

5 An EKF update using magnetometer measurements

5.1 Write magnetometer measurement update function

The magnetometer measurement update was implemented in the function `mu_m`. The implementation is to the greatest extent, identical to the function for the accelerometer update `mu_g`. The difference is that this function utilizes the measurements from the magnetometer, covariance from the magnetometer and the nominal magnetic field m^0 instead of g^0.

```
function [x, P] = mu_m(x, P, mag, m0, Rm)
% MU_M Measurement update given magnetometer data
%
% By: Nicholas Granlund
%     Louise Olsson
%
%Input:
%   - x             [4x1]       State vector (quaternions)
%   - P             [4x4]       Covariance
%   - mag           [3x1]       Magnetometer measurement
%   - m0            [3x1]       Nominal magnetic vector
%   - Rm            [3x3]       Measurement noise
%
%Output:
%   - x             [3x1]    State vector (quaternion)
%   - P             [4x4]    State covariance matrix

% Make sure vectors has right dimension
m0 = reshape(m0, [3,1]);
fm = zeros(size(m0,1),1);

% Measurement model
hx = Qq(x)'*(m0+fm);

% dH/dq
[Q0, Q1, Q2, Q3] = dQqdq(x);

% Measurement model jacobian Hx
Hx(:,1) = Q0'*(m0+fm);
Hx(:,2) = Q1'*(m0+fm);
Hx(:,3) = Q2'*(m0+fm);
Hx(:,4) = Q3'*(m0+fm);

% Inovation covariance matrix
S = Hx*P*Hx' + Rm;

% Kalman gain
K = P*Hx'*inv(S);

% Update state estimate
x = x + K*(mag-hx);

% Update covariance estimate
P = P - K*S*K';

% Normalize the quaternion
[x, P] = mu_normalizeQ(x, P);

% return
end
```

5.2 Adding magnetometer update to EKF

By implementing the magnetometer update described above we get a filter with similar behaviour to the one implemented inside the phone. However, by introducing a disturbance in the magnetic field it may still disrupt the azimuth estimation to be attracted by the disturbance source. For this we will have to implement an outlier rejection to disregard the the disturbances. We will also have to introduce a lowpass filter so the estimate is affected slower by other disturbances.

5.3 Outlier rejection for magnetic disturbances

The outlier rejection was implemented as suggested by the assignment description. the low-pass filter was implemented with a low $\alpha = 0.01$ which means that outer disturbances will influence with small amounts over time. When the magnitude of the magnetic field exceeds the expected amount of earths gravitational field (25-65 μT), the measurement will be considered an outlier and not taking into account when estimation the orientation of the phone at that time step.

When a disturbance is introduced to this updated implementation, the `setMagDist` will be turned on and disregard the magnetometer measurements. However, if the disturbance is small enough to not activate the outlier rejection, the azimuth estimation will slowly be attracted by the disturbance source, just as we expected.

5.4 Evaluation of filter

The complete EKF for the orientation estimation works remarkably well and follows the Matlab/Google estimate to great extent. By using all of the available sensor data it was possible to obtain an estimate significantly close to the true orientation. This is illustrated in Figure 5.1.

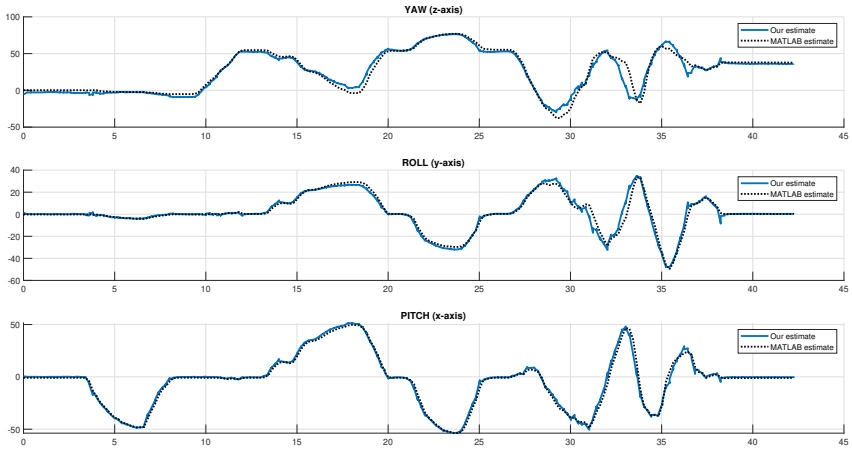


Figure 5.1: Final filter compared to Matlab/Google filter.

This concludes the final project in the course SSY345 - Sensor fusion and non-linear filtering.