

SSY345 - Orientation estimation using smartphone sensors

Mohammad Albarham (CID: albarham), Yibo Zhou (CID: yiboz)

May 2025

1 Introduction

In this report, we implement an Extended Kalman Filter (EKF) to estimate the orientation of a smartphone using sensor data from its gyroscope, accelerometer, and magnetometer. Orientation estimation is a fundamental problem in sensor fusion, with critical applications in navigation, robotics, and human-computer interaction. By integrating noisy measurements from multiple sensors, the EKF provides a robust framework for tracking the time-varying orientation of a smartphone in real-time. We represent the orientation using a quaternion, which captures the three degrees of freedom of rotation efficiently. Quaternions were chosen over alternative representations, such as Euler angles or rotation matrices, due to their numerical stability and suitability for estimation tasks. Unlike rotation matrices, which require nine parameters to describe three degrees of freedom, quaternions offer a minimal representation with only four parameters, subject to a unit norm constraint. Compared to Euler angles, which suffer from discontinuities, non-uniqueness, and the gimbal lock problem, quaternions provide a continuous and unambiguous representation, making them ideal for the recursive updates in an EKF.

The project leverages data streamed from an Android smartphone via the Sensor Fusion app. The primary goal is to develop an EKF that fuses inertial measurements from the gyroscope, which provides angular velocity, with accelerometer and magnetometer data, which offer absolute orientation references relative to gravity and the Earth's magnetic field, respectively. Through this process, we aim to understand the complementary nature of these sensor measurements, develop measurement models based on real data analysis, and design outlier rejection mechanisms to enhance filter robustness.

Key learning objectives include gaining insights into the properties of smartphone sensor measurements, deriving appropriate models for the EKF, and evaluating the filter's performance under various conditions, such as sensor biases and external disturbances. By completing this project, we demonstrate the ability to handle asynchronous sensor data, estimate orientation accurately, and compare our filter's performance against the smartphone's built-in orientation estimates provided by the Google API. This report details the theoretical framework, implementation steps, and experimental results, providing a comprehensive analysis of the EKF's effectiveness in orientation estimation.

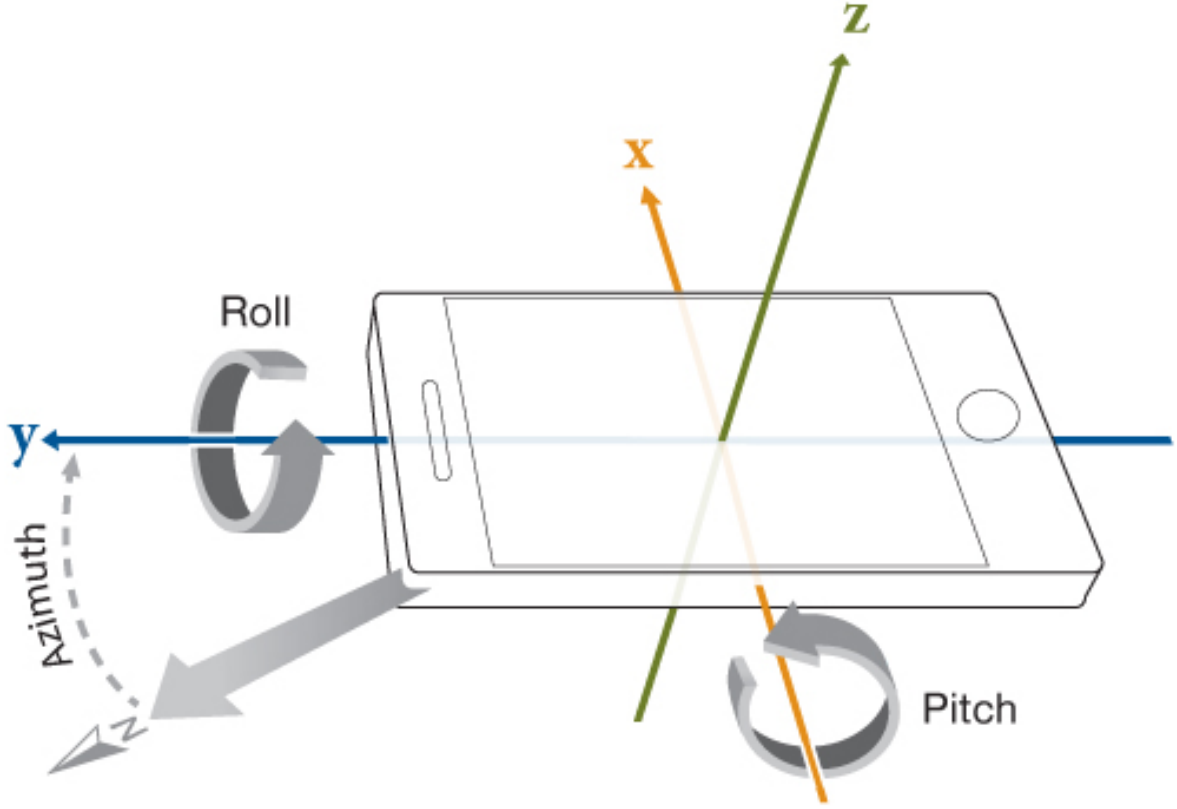


Figure 1: Illustration of a smartphone's sensor frame of reference ($\mathcal{S}$), with axes x , y , and z aligned as per the Android/iOS API convention. The depicted rotations (Roll, Pitch, and Yaw(in z -axis)) correspond to the orientation parameters estimated using an Extended Kalman Filter (EKF) with quaternion representation. The associated proof verifies that a quaternion with $\alpha = 0$ yields the identity rotation matrix, confirming the expected behavior for no rotation.

1.1 Proof that $\alpha = 0$ yields identity rotation matrix

To verify that $\alpha = 0$ yields the expected rotation matrix, consider the quaternion $q = \begin{pmatrix} \cos\left(\frac{\alpha}{2}\right) \\ \sin\left(\frac{\alpha}{2}\right) \hat{v} \end{pmatrix}$, where $\hat{v}$ is a unit vector. For $\alpha = 0$, we have $\cos(0) = 1$, $\sin(0) = 0$, so:

$$q = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

Substitute into the rotation matrix:

$$Q(q) = \begin{pmatrix} 2q_0^2 + 2q_1^2 - 1 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & 2q_0^2 + 2q_2^2 - 1 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & 2q_0^2 + 2q_3^2 - 1 \end{pmatrix}.$$

With $q_0 = 1$, $q_1 = q_2 = q_3 = 0$, we compute:

$$Q(q) = \begin{pmatrix} 2(1)^2 + 2(0)^2 - 1 & 2(0)(0) - 2(1)(0) & 2(0)(0) + 2(1)(0) \\ 2(0)(0) + 2(1)(0) & 2(1)^2 + 2(0)^2 - 1 & 2(0)(0) - 2(1)(0) \\ 2(0)(0) - 2(1)(0) & 2(0)(0) + 2(1)(0) & 2(1)^2 + 2(0)^2 - 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

This is the identity matrix, as expected, since $\alpha = 0$ implies no rotation, aligning the sensor and world frames.

1.2 Task 1

- **Discuss pros and cons regarding this choice of input.**

Pros: Using gyroscope measurements as inputs ($u_k = \omega_k$) reduces model complexity by keeping the state vector small ($x_k = q_k$, 4D) instead of including angular velocities or biases. This simplifies the state transition model ($\dot{q} = \frac{1}{2}S(\omega)q$) and reduces computational load, leveraging the high accuracy of smartphone gyroscopes for reliable prediction steps.

Cons: If gyroscope measurements are noisy or biased, treating them as noise-free inputs introduces errors in the quaternion dynamics, leading to inaccurate orientation estimates. For instance, unmodeled biases or drift can accumulate over time, degrading filter performance, especially without correction from other sensors.

- **Can you imagine a situation where this would not be a good choice?**

This choice is suboptimal for long-term orientation estimation without frequent updates from accelerometer or magnetometer measurements. Gyroscope drift, common in MEMS sensors, causes cumulative errors in the quaternion dynamics ($\dot{q} = \frac{1}{2}S(\omega)q$), leading to unbounded orientation drift over extended periods, especially during continuous motion or when other sensors update infrequently.

- **When would it be better to include angular velocities in the state vector?**

It is better to include angular velocities in the state vector ($x_k = [q_k^T, \omega_k^T]^T$) when gyroscope measurements have significant biases or high noise, as this allows the EKF to estimate ω_k and correct for errors using accelerometer and magnetometer measurements. This is particularly useful in noisy environments, with time-varying biases, or for applications requiring high precision where assuming noise-free inputs is invalid.

2 Data Collection and Filter Design - Task 2

We collected data from the gyroscope, accelerometer, and magnetometer using a Samsung Ultra 22 smartphone. The device was oriented with its y-axis aligned toward the north and placed on a flat surface, away from electrical components that might interfere with the magnetic field, for a duration of one minute. The data was then streamed to a laptop using the recommended streaming mode via the Sensor Fusion app and Matlab FilterTemplate script via a WiFi network.

For each sensor, measurements were recorded across the three axes (x, y, z). We computed the mean and covariance for each axis to assess the statistical properties of the sensor data. Additionally, time-series plots and histograms were generated to visualize the data trends and noise distributions. These visualizations allow us to identify potential biases and characterize the type of noise present in the measurements, which is crucial for tuning the Extended Kalman Filter (EKF) model.

The statistical summary of the sensor data, including means and covariances, is presented in the table below. The accompanying figures illustrate the time-series data alongside their respective noise distributions for each axis, providing a comprehensive view of the sensor behavior under static conditions.

Sensors	Mean	Covariance
Gyroscope (rad s ⁻¹)	0	0.1231 0.1014 -0.0020
	0	0.1014 0.3475 -0.0089
	0	-0.0020 -0.0089 0.0170
Accelerometer (m s ⁻²)	0.0473	0.0575 -0.0075 0.0324
	0.0809	-0.0075 0.1298 -0.0651
	9.8743	0.0324 -0.0651 0.6122
Magnetometer (μT)	1.0710	0.9197 0.1279 -0.3767
	18.8779	0.1279 0.5496 0.0331
	-45.0483	-0.3767 0.0331 1.1012

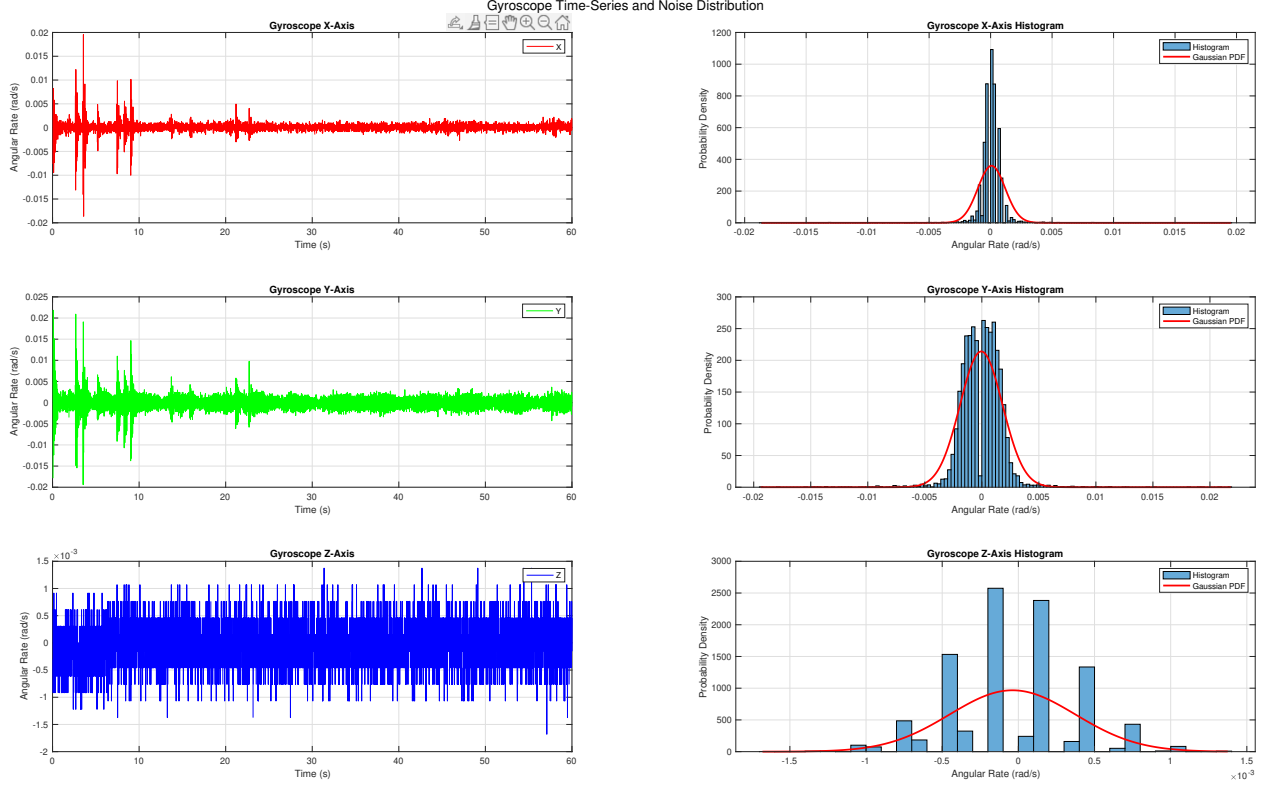


Figure 2: Gyroscope Time-Series (left) and Noise Distribution Histograms (right) Across X, Y, and Z Axes.

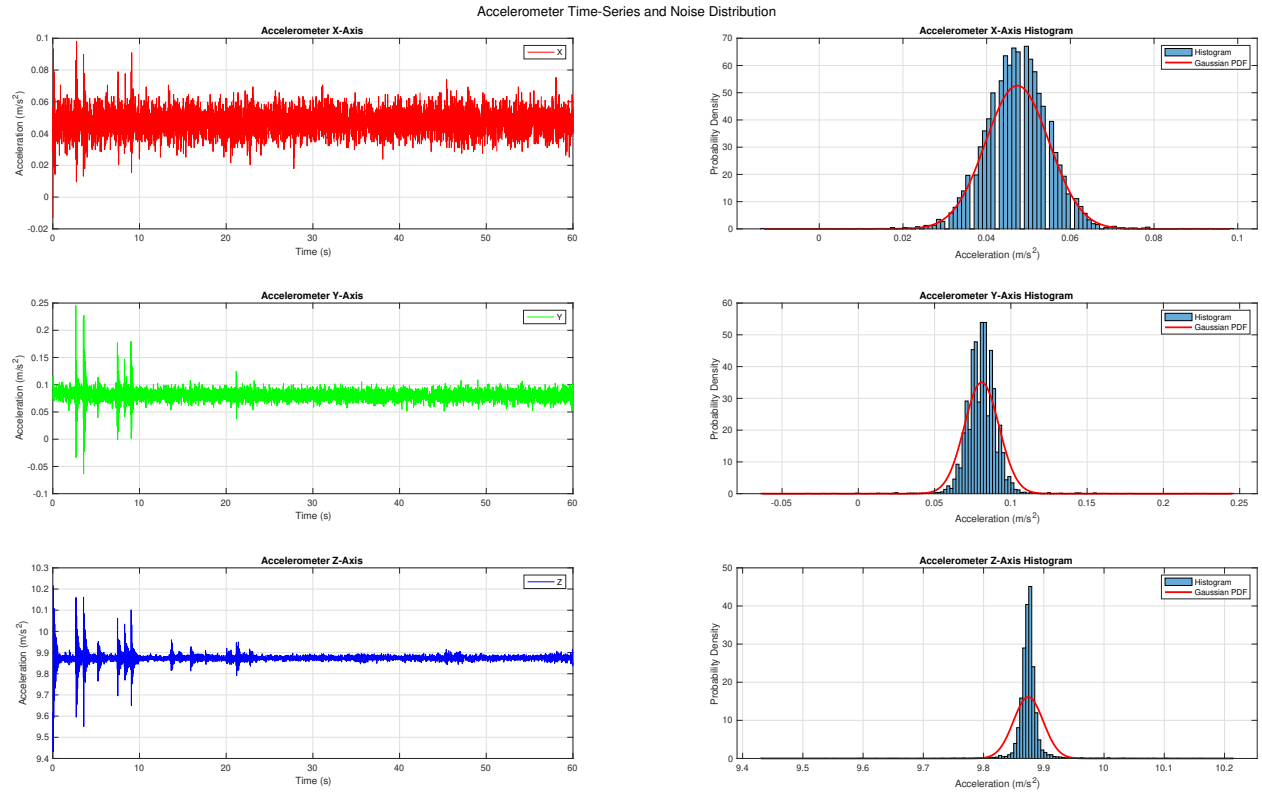


Figure 3: Accelerometer Time-Series (left) and Noise Distribution Histograms (right) Across X, Y, and Z Axes.

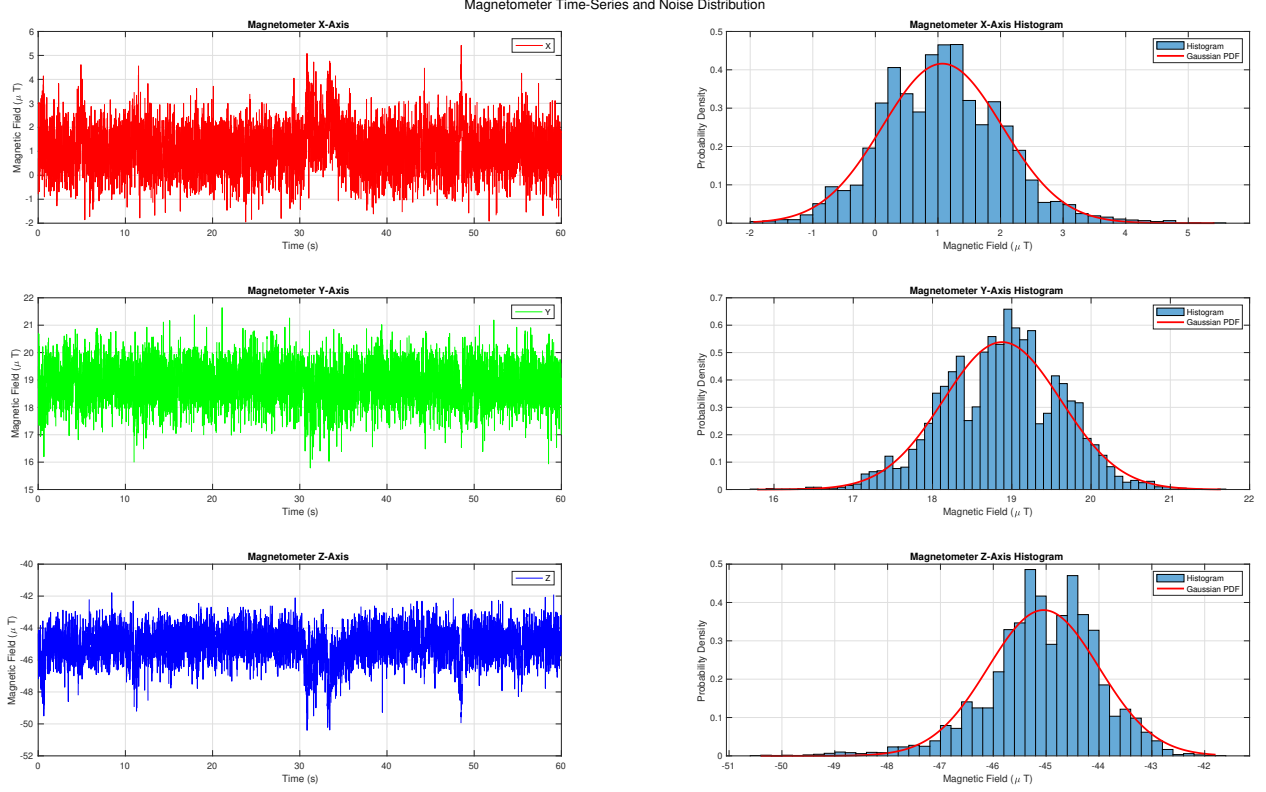


Figure 4: Magnetometer Time-Series (left) and Noise Distribution Histograms (right) Across X, Y, and Z Axes.

2.1 Gyroscope

The gyroscope data, presented in Table 2 and Figure 2, reveals minimal bias across all axes, indicating a stable sensor response under static conditions. The time-series plots show consistent behavior with no significant drift, while the histograms suggest that the noise distribution is approximately Gaussian. This characteristic supports the application of standard noise models in the Extended Kalman Filter (EKF) for orientation estimation.

2.2 Accelerometer

As detailed in Table 2 and Figure 3, the accelerometer exhibits small biases on the X and Y axes, with a slight offset on the Z-axis compared to the expected gravitational acceleration. The time-series plots indicate stable measurements over time, and the histograms reveal a noise distribution that is approximately Gaussian. This noise behavior aligns well with the assumptions of the EKF, facilitating effective sensor fusion for orientation tracking.

2.3 Magnetometer

The magnetometer data, shown in Table 2 and Figure 4, reflects the Earth’s magnetic field components along the X, Y, and Z axes, with a magnitude consistent with typical geomagnetic field strengths. The time-series plots demonstrate steady measurements, and the histograms indicate an approximately Gaussian noise distribution. This supports the use of standard noise modeling in the EKF, enabling the magnetometer to serve as a reliable reference for orientation estimation.

2.4 Final Thoughts on the data from the sensors

The time-series plots (Figures 2, 3, 4) for the gyroscope, accelerometer, and magnetometer reveal no significant trends, confirming the stability of the measurements under static conditions. The gyroscope exhibits negligible bias, while the accelerometer shows minor biases on the X and Y axes, which are sufficiently small for practical purposes. The magnetometer measurements align with the Earth's magnetic field, making them suitable as a reference for orientation estimation rather than indicating bias. Although the gyroscope noise appears slightly non-Gaussian, requiring careful handling in the EKF, the accelerometer and magnetometer noise distributions are approximately Gaussian, supporting the standard assumptions of the EKF framework.

3 Task 3

Firstly, we need to solve the differential equation:

$$\dot{q} = \frac{1}{2}S(w_{k-1} + v_{k-1})q(t), t \in [t_{k-1}, t_k] \quad (1)$$

It can be observed that the differential equation are in the form of

$$\dot{q}(t) = Aq(t) \quad (2)$$

whose solution is in the form of

$$q(t) = e^{A(t-t_{k-1})}q(t_{k-1}) \quad (3)$$

Thus, the solution of the given differential equation is

$$q(t_k) = e^{\frac{1}{2}S(w_{k-1}+v_{k-1})\Delta t}q(t_{k-1}) \quad (4)$$

By applying the approximation $e^A = I + A$, the solution becomes to

$$q(t_k) = (I + \frac{1}{2}S(w_{k-1} + v_{k-1})\Delta t)q(t_{k-1}) \quad (5)$$

$$= [I + \frac{1}{2}S(w_{k-1})\Delta t]q(t_{k-1}) + \frac{1}{2}S(v_{k-1})\Delta t q(t_{k-1}) \quad (6)$$

$$= [I + \frac{1}{2}S(w_{k-1})\Delta t]q(t_{k-1}) + \frac{1}{2}\Delta t \begin{bmatrix} 0 & -v_x & -v_y & -v_z \\ v_x & 0 & v_z & -v_y \\ v_y & -v_z & 0 & v_x \\ v_z & v_y & -v_x & 0 \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (7)$$

$$= [I + \frac{1}{2}S(w_{k-1})\Delta t]q(t_{k-1}) + \frac{1}{2}\Delta t \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad (8)$$

Thus, for the given discretized model of the form

$$q_k = F(w_{k-1})q_{k-1} + G(\hat{q}_{k-1})v_{k-1} \quad (9)$$

The parameters are

$$F(w_{k-1}) = I + \frac{1}{2}S(w_{k-1})\Delta t \quad (10)$$

$$G(\hat{q}_{k-1}) = \frac{1}{2}\Delta t \begin{bmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{bmatrix}, \hat{q}_{k-1} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} \quad (11)$$

In this problem, we approximate $G(q_{k-1})$ by $G(\hat{q}_{k-1})$ because the true state q_{k-1} is not directly observable. Instead, we rely on the filter's estimate $\hat{q}_{k-1}$ to linearize the noise influence. This approach is consistent with the Extended Kalman Filter (EKF) methodology, where all linearizations are performed around the current state estimate. It enables correct propagation of uncertainty through the system model while maintaining computational tractability.

4 Task 4: Derivation of the EKF Time Update for Quaternion State

In this task, we consider the quaternion $[q_0, q_1, q_2, q_3]^\top$ as the state vector $\mathbf{x}$ representing orientation. According to EKF formulae, the state vector is updated as:

$$\mathbf{x}_{k+1|k} = F(w_{k-1})\mathbf{x}_{k|k} \quad (12)$$

And the state covariance matrix $\mathbf{P}$ is updated as:

$$\mathbf{P}_{k+1|k} = F(w_k)\mathbf{P}_{k|k}F(w_k)^\top + G(\hat{q}_k)\mathbf{R}_wG(\hat{q}_k)^\top$$

where $\mathbf{R}_w$ is the process noise covariance matrix associated with the gyroscope noise. The predicted quaternion $\mathbf{q}$ is then normalized:

$$\mathbf{q}_{k+1} \leftarrow \frac{\mathbf{q}_{k+1}}{\|\mathbf{q}_{k+1}\|}$$

Also, the code snippet of the function is shown below:

```

1 function [x, P] = tu_qw(x, P, omega, T, Rw)
2 % Inputs:
3 % x - 4x1 state vector (quaternion [q0; q1; q2; q3])
4 % P - 4x4 covariance matrix
5 % omega - 3x1 angular rate vector [wx; wy; wz]
6 % T - scalar time interval
7 % Rw - 4x4 process noise covariance
8
9 % Compute skew-symmetric matrix S(omega)
10 % State transition matrix F
11 Fk = eye(4) + 0.5 * Somega(omega) * T;
12
13 % Noise gain matrix
14 G = 0.5 * T * Sq(x);
15
16 % Predicted x and P:
17 x = Fk * x;
18
19 % Normalize quaternion
20 [x,P] = mu_normalizeQ(x,P);
21
22 % Predict covariance
23 P = Fk * P * Fk' + G * Rw * G'; % Rw;
24 end

```

Listing 1: MATLAB function for EKF time update step

Case without Angular Rate Measurements

In the case where the angular rate ω is not available (e.g., due to sensor failure or dropout), we assume that the orientation has not changed significantly since the last update. Therefore, the quaternion remains unchanged:

$$\mathbf{x}_{k+1|k} = \mathbf{x}_{k|k}$$

However, uncertainty still increases over time due to unmodeled motion or system noise. We reflect this increase by updating the covariance as:

$$\mathbf{P}_{k+1|k} = \mathbf{P}_{k|k} + G(\hat{q}_k) \mathbf{R}_w G(\hat{q}_k)^\top$$

This ensures the filter remains consistent even in the absence of measurements.

```

1 function [x, P] = tu_qw_no_omega(x, P, T, Rw)
2     % x - 4x1 state vector [q0; q1; q2; q3]
3     % P - 4x4 covariance matrix
4     % T - scalar time interval
5     % Rw - 4x4 process noise covariance
6
7     % No angular rate, state remains nominal
8     x = x;
9
10    % Noise gain matrix
11    G = 0.5 * T * Sq(x);    % 0.5 * T * eye(4);
12
13    % Covariance update
14    P = P + G * Rw * G';
15
16    % Normalize quaternion
17    [x,P] = mu_normalizeQ(x,P);
18 end

```

Listing 2: MATLAB function for EKF time update step without angular rate measurements

5 Task 5

In Task 5, we implemented an EKF time update using the function `tu_qw` to estimate the phone's orientation using only gyroscope data. Below, we describe the implementation, performance, and behavior under different conditions.

For the time update function `[x, P] = tu_qw(x, P, omega, T, Rw)`, the input parameters are as follows:

- **x**: the current quaternion prior.
- **P**: the current state covariance prior.
- **omega**: the angular velocity measurement from the gyroscope (`meas.gyr`, rad/s).
- **T**: the time interval (in seconds) since the last update, which is computed as the difference between consecutive timestamps
- **Rw**: the process noise covariance matrix of the gyroscope from data collected when the phone was on rest. The **Rw** is below:

$$\mathbf{R}_w = 1.0 \times 10^{-5} \begin{bmatrix} 0.3759 & 0 & 0 \\ 0 & 0.6197 & 0 \\ 0 & 0 & 0.0171 \end{bmatrix} \quad (13)$$

The filter loops over the stored `meas` data, applying the time update at each step to predict the next orientation.

Below, we show the results and interpretation for the 3 cases:

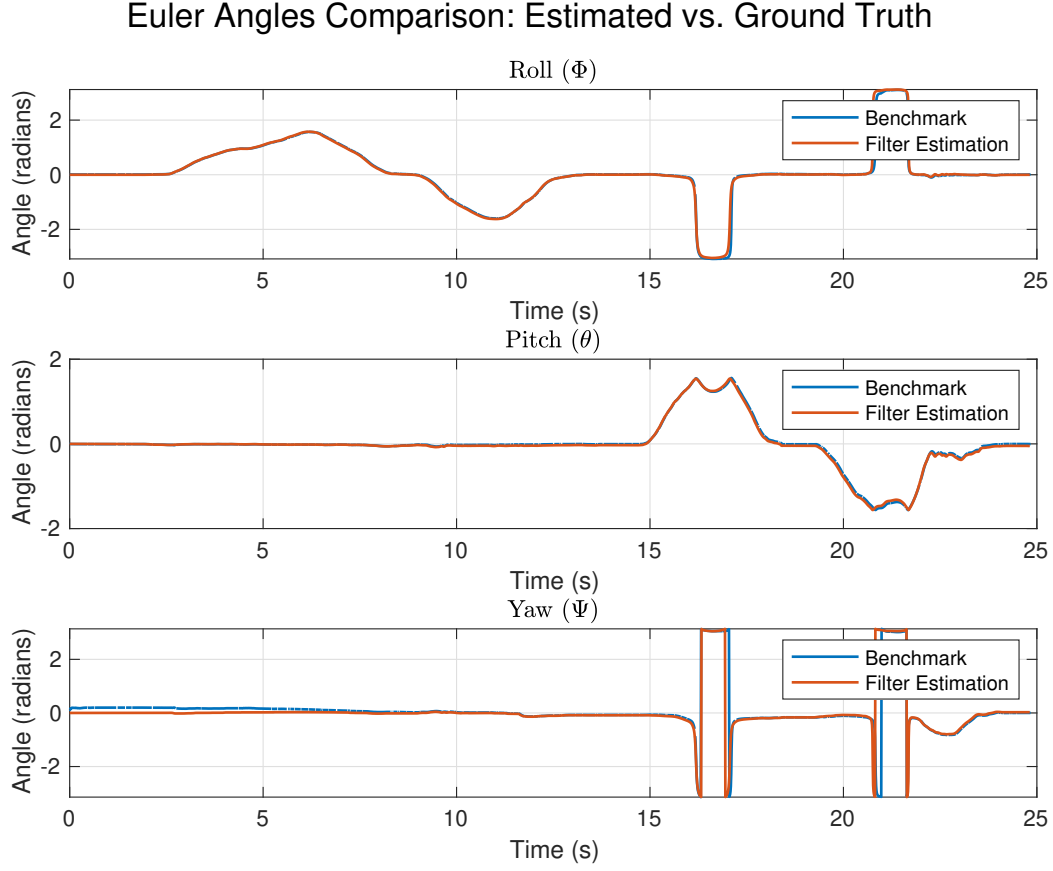


Figure 5: Euler Angles comparison between the ground truth and the EKF estimations by using the gyroscope only. The filter is started with the phone on the side laying face up on the desk.

As shown in Figure 5, the filter is very reactive to rotations because it directly integrates ω to update the quaternion. However, without accelerometer or magnetometer updates, it cannot correct for drift. The gyroscope noise and small biases accumulate over time, causing the estimated orientation to drift. This means the reference frame slowly rotates even when the phone is still.

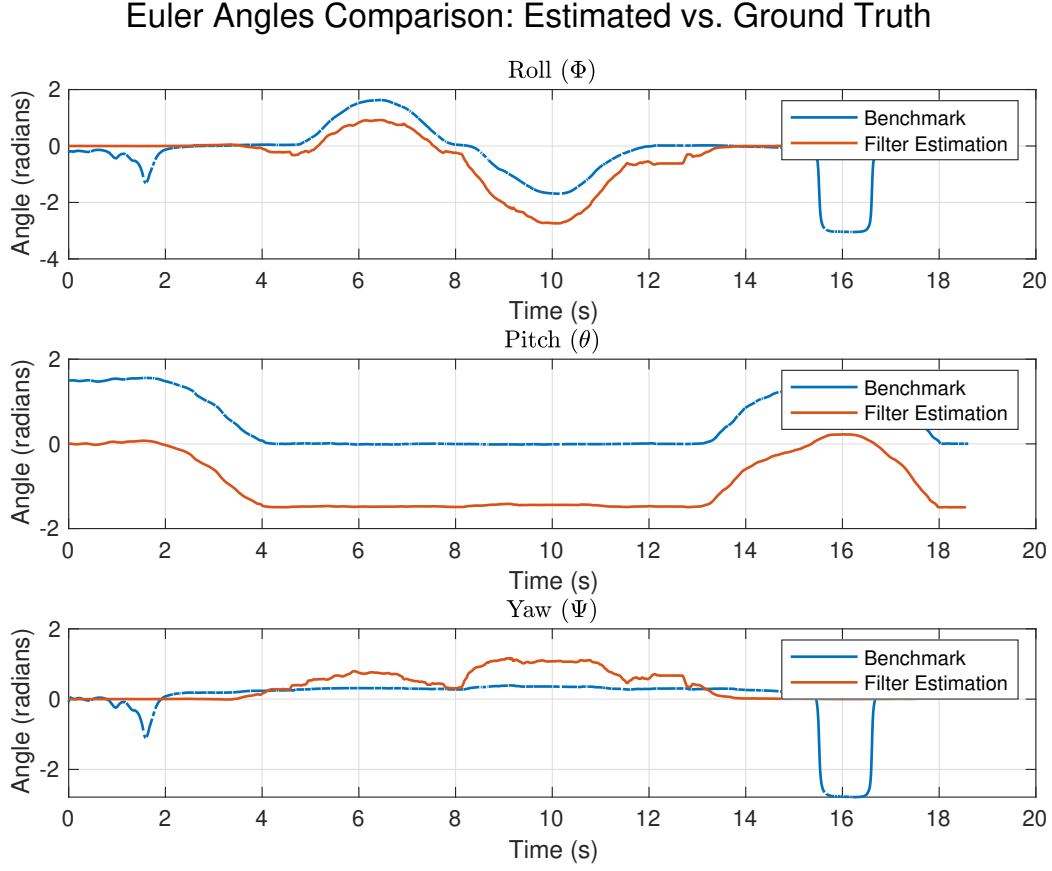


Figure 6: Euler Angles estimations by using the gyroscope only. The filter is started with the phone on the side instead of laying face up on the desk.

As shown in Figure 6, when starting the filter with the phone on its side instead of face up, the initial state $x = [1; 0; 0; 0]$ does not match the phone's actual orientation (e.g., rotated 90°). Since the filter relies only on the gyroscope, it interprets this mismatch as a rotation, leading to an immediate deviation from the true orientation (`meas.orient`). Without measurement updates to provide an absolute reference (e.g., gravity), the filter cannot correct this error.

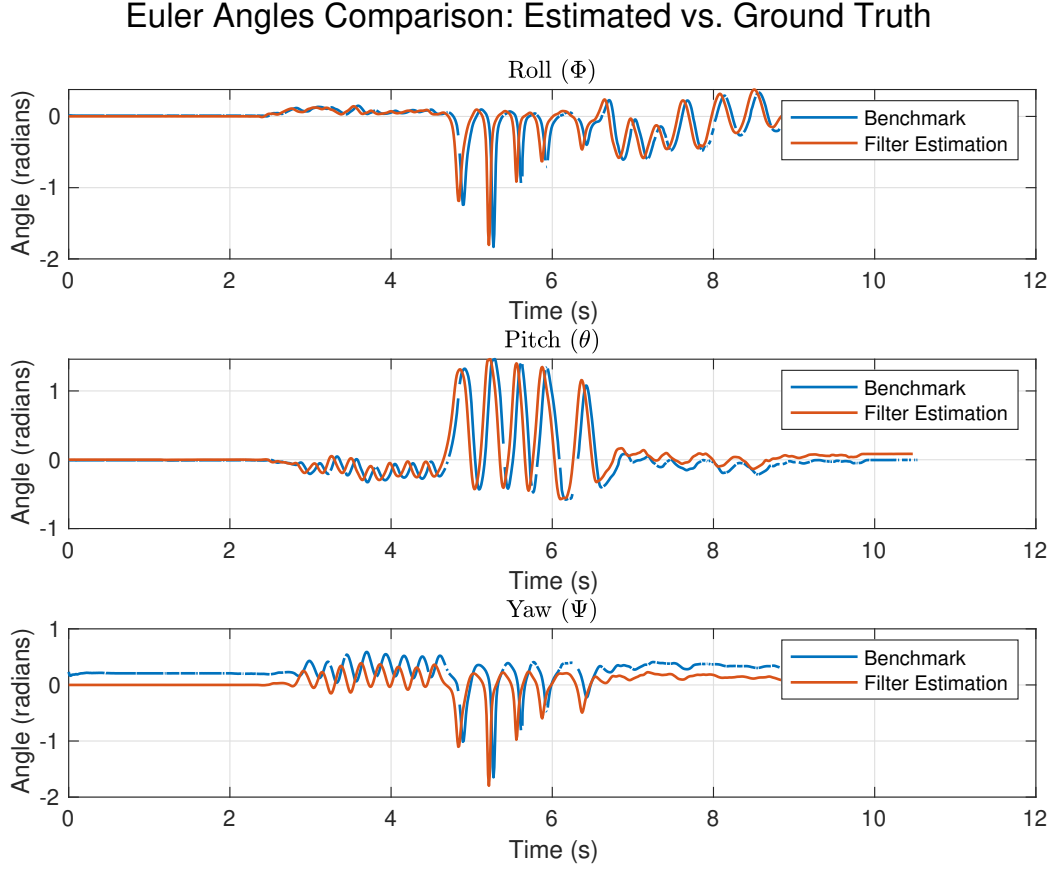


Figure 7: Euler Angles estimations by using the gyroscope only. The phone is shaken in the test.

As shown in Figure 7, when shaking the phone, the filter tracks the rapid rotations well because ω changes quickly, and `tu_qw` updates x reactively. However, the drift persists due to accumulated noise, causing the estimated orientation to gradually rotate away from the true orientation over time.

6 Task 6

6.1 Estimation of g_0 from Accelerometer Data

To estimate g_0 (gravitational acceleration), We used the accelerometer data from `meas.acc` when the phone was stationary (first 60 seconds). Here's how I did it:

- **Filter Data:** We selected the first 60 seconds (`meas.t <= 60`) and removed NaNs, giving `valid.acc` ($3 \times M$ matrix, m/s^2).
- **Compute Magnitude:** For each timestamp, I calculated the magnitude: $\sqrt{a_x^2 + a_y^2 + a_z^2}$, where $[a_x; a_y; a_z]$ are the accelerometer readings.
- **Estimate g_0 :** We took the mean of the magnitudes to get g_0 .

The code is as follows:

```

1 time_idx = meas.t <= 60;
2 acc = meas.acc(:, time_idx);
3 valid_acc = acc(:, ~any(isnan(acc), 1));
4 magnitudes = sqrt(sum(valid_acc.^2, 1));
5 g0 = mean(magnitudes);

```

Listing 3: MATLAB code for computing acceleration magnitudes

The result was $g_0 \approx 9.8747 \text{ m/s}^2$, which matches the expected value of gravity. The accelerometer means (X: 0.0473, Y: 0.0809, Z: 9.8743 m/s^2) confirm the phone was face up, with the Z-axis aligned with gravity. This method works because, when stationary, the accelerometer measures only gravity, and the magnitude is independent of orientation.

6.2 Accelerometer EKF update

The measurements from the accelerometer can be modeled using the relation

$$\mathbf{x}_k^a = Q^\top(\mathbf{q}_k)g_0 + e_k^a \quad (14)$$

where e_k^a is the measurement noise. This model is based on the assumption that the phone is not accelerating in the world frame.

Thus, the EKF update function of the accelerometer measurement is

$$H_k = \frac{d[Q^\top(\mathbf{x}_{k|k-1})g_0]}{dq} \quad (15)$$

$$S_k = H_k P_{k|k-1} H_k^\top + \mathbf{R}_a \quad (16)$$

$$K_k = P_{k|k-1} H_k^\top S_k^{-1} \quad (17)$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + K_k [y_k^a - Q^\top(\mathbf{x}_{k|k-1})g_0] \quad (18)$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - K_k S_k K_k^\top \quad (19)$$

Below is the $\mathbf{R}_a$ matrix:

$$\mathbf{R}_a = 1.0 \times 10^{-3} \begin{bmatrix} 0.0777 & 0 & 0 \\ 0 & 0.1514 & 0 \\ 0 & 0 & 0.9775 \end{bmatrix} \quad (20)$$

```

1 function [x, P] = mu_g(x, P, yacc, Ra, g0)
2 % mu_g Extended Kalman Filter measurement update for acceleration-based quaternion
   estimation
3
4 % --- Extended Kalman Filter Update ---
5 % 1. Compute the expected measurement and its Jacobian at the predicted state
6
7 % Here fk is zero, no acceleration
8 fa = zeros(3,1);
9 y_x = Qq(x)' * (g0 + fa);
10
11 % Calculate the Hx matrix
12 [Q0, Q1, Q2, Q3] = dQdq(x); % 3x4 matrices
13 Hx = [Q0' * g0, Q1' * g0, Q2' * g0, Q3' * g0]; % 3x4
14
15 % 2. Calculate the innovation (difference between actual and expected measurement)
16 innov = yacc - y_x;
17

```

```

18 % 3. Compute the innovation covariance
19 % Formula:  $S = H * P * H' + R$ 
20 S = Hx * P * Hx' + Ra;
21
22 if rcond(S) < 1e-10, S = S + 1e-6 * eye(3); warning('Regularized S'); end
23
24 % 4. Compute the cross-covariance between state and measurement
25 % Formula:  $C = P * H'$ 
26 crossCov = P * Hx';
27
28 % 5. Calculate the Kalman gain
29 % Formula:  $K = C * inv(S)$ 
30 K = crossCov / S;
31
32 % 6. Update the state mean
33 % Formula:  $x = x_{pred} + K * innov$ 
34 x = x + K * innov;
35
36 % 7. Update the state covariance
37 % Formula:  $P = P - K * S * K'$ 
38 P = P - K * S * K';
39
40 % Important: normalize the quaternion
41 [x, P] = mu_normalizeQ(x, P);
42 end

```

Listing 4: MATLAB function for EKF measurement update with acceleration-based quaternion estimation

7 Task 7

7.1 Estimation after adding accelerometer measurements

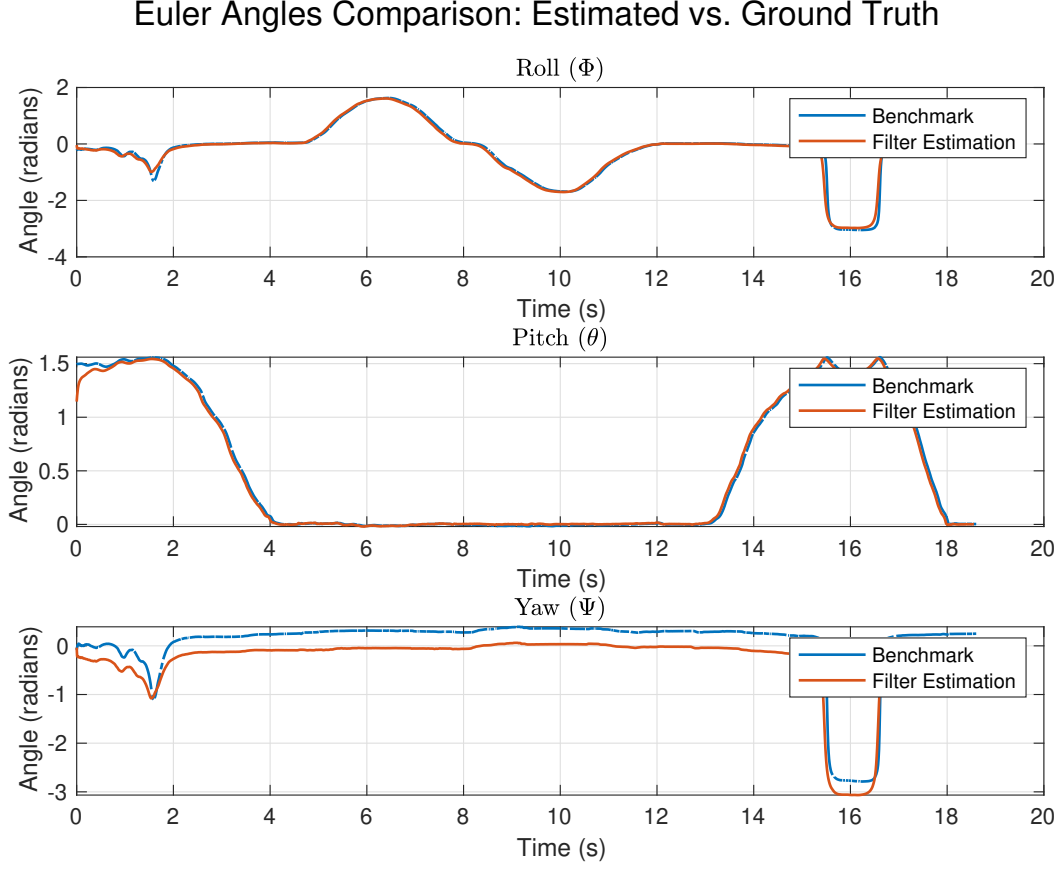


Figure 8: Euler Angles estimations by using the gyroscope and accelerometer measurements.

Figure 8 shows the result after the accelerometer measurement update is added. According to the pitch benchmark, it can be seen that the filter starts with the phone on the side instead of facing up on the desk. Although the initial prior of the filter is facing up, e.g. $q_0 = [1 \ 0 \ 0 \ 0]$, it converges to the true rotation in the horizontal plane (roll and pitch) after few time steps. This is contributed by the accelerometer, as it knows the gravity direction. However, the absolute yaw estimation is still incorrect, as there is no knowledge on the horizontal direction.

7.2 Accelerometer Estimation with additional body acceleration

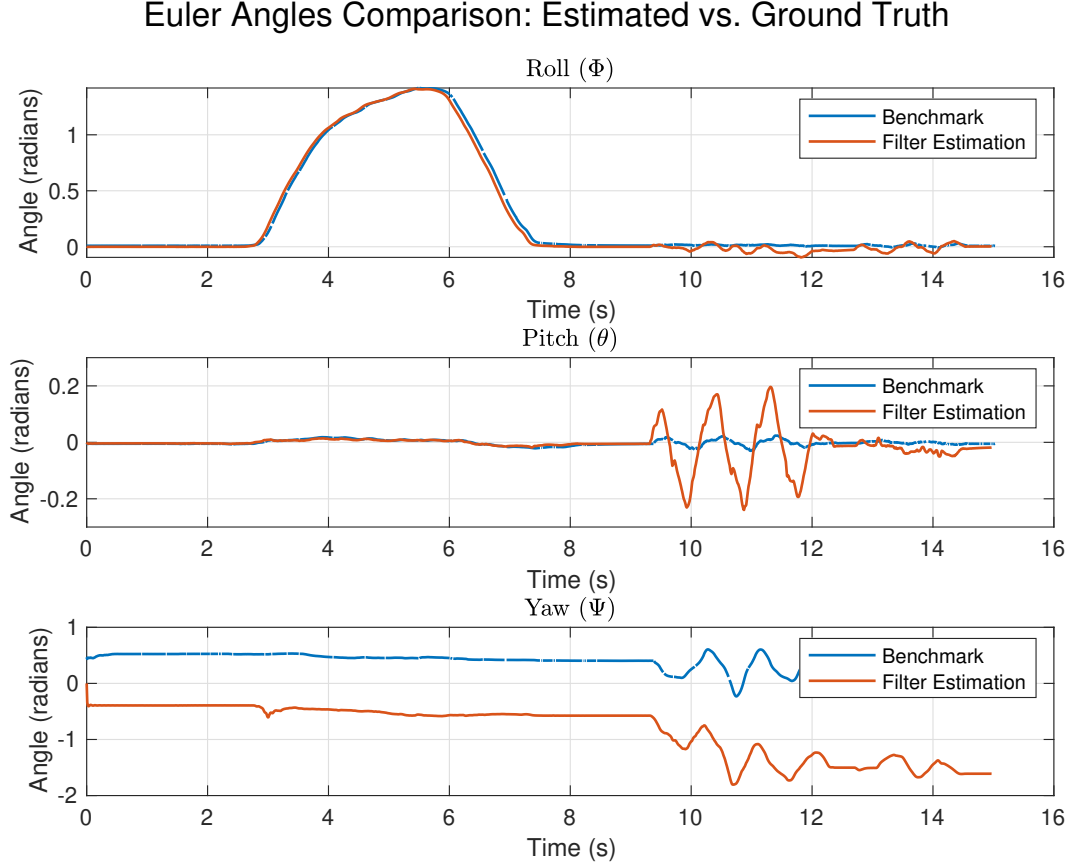


Figure 9: Euler Angles estimations by using the gyroscope and accelerometer measurements when the device is moving quickly.

Figure 9 shows the estimation result when the device is moving quickly. It can be seen that horizontal angel pitch cannot align well with the benchmark anymore when extra body acceleration is added.

This is because the accurate accelerometer measurement model is:

$$\mathbf{y}_k^a = \mathbf{Q}^\top(\mathbf{q}_k)(g_0 + f_k^a) + e_k^a \quad (21)$$

where f_k^a is the non-zero force if the sensor accelerates in the world frame. As f_k^a is ignored by our simplified model, the accelerometer update performs poorly when additional body acceleration is added.

8 Task 8

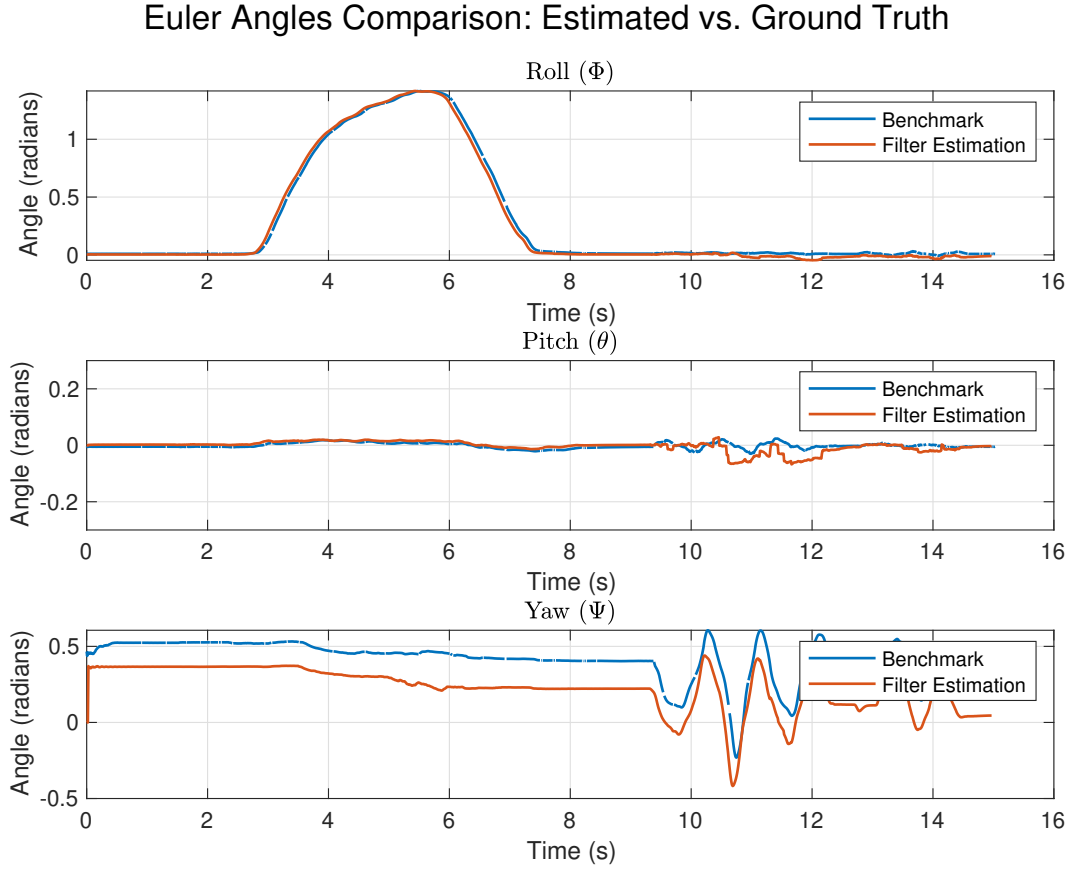


Figure 10: Euler Angles estimations by using the gyroscope and accelerometer measurements when the device is moving quickly. Outlier rejection is implemented to skip the accelerometer updates when the measurement is regarded to be an outlier.

To solve the problem that the simplified accelerometer measurement model performs poorly when additional body acceleration is added, the outlier rejection algorithm for a_k is applied. When the error between $\|a_k\|$ and $\|g_0\|$ is larger than the threshold, the measurement is regarded an outlier. Then the accelerometer update would be skipped in this timestep.

The results after applying the outlier rejection algorithm is shown in Figure 10. The device is moving quickly while the EKF is tested as in Task 7. Compared to the result shown in Figure 9, it can be seen that the horizontal angel pitch can now align well with the benchmark, even the model is simplified and f_k^a is assumed to be zero. The code used is listed below:

```

1 % Accelerometer
2 acc = data(1, 2:4)';
3
4 if ~any(isnan(acc)) % Acc measurements are available.
5     % Outlier rejection within 0.9 and 1.1 of gravity
6     if norm(acc) < 9.81 * 0.90 || norm(acc) > 9.81 * 1.1
7         % Outlier detected
8         ownView.setAccDist(1);
9     else

```

```

10 % Update EKF using the acceleration data, f_k_a = 0
11 [x, P] = mu_g(x, P, acc, Ra, g0);
12 ownView.setAccDist(0);
13 end
14 end

```

Listing 5: MATLAB code for accelerometer data processing and EKF update

9 Task 9

9.1 Estimation of m_0 from magnetometer Data

To estimate m_0 (earth magnetic field), we used the magnetometer data from `meas.mag` when the phone lies horizontally without knowledge of the direction of north. The mean value of the data

$$m = [m_x \ m_y \ m_z]^\top = [1.0710 \ 18.8779 \ -45.0483]^\top \quad (22)$$

Then, m_0 is calculated using the relation

$$m_0 = \begin{bmatrix} 0 & \sqrt{m_x^2 + m_y^2} & m_z \end{bmatrix}^\top = [0.0000 \ 18.9083 \ -45.0483]^\top \quad (23)$$

9.2 Magnetometer EKF update

The measurements from the magnetometer can be modeled using the relation

$$\mathbf{y}_k^m = Q^\top(\mathbf{q}_k)m_0 + e_k^m \quad (24)$$

where e_k^m is the measurement noise. This model is based on the assumption that the extra magnetic field f_k^m is zero.

Thus, the EKF update function of the accelerometer measurement is

$$H_k = \frac{d[Q^\top(\mathbf{x}_{k|k-1})m_0]}{dq} \quad (25)$$

$$S_k = H_k P_{k|k-1} H_k^\top + \mathbf{R}_m \quad (26)$$

$$K_k = P_{k|k-1} H_k^\top S_k^{-1} \quad (27)$$

$$\mathbf{x}_{k|k} = \mathbf{x}_{k|k-1} + K_k [y_k^m - Q^\top(\mathbf{x}_{k|k-1})m_0] \quad (28)$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - K_k S_k K_k^\top \quad (29)$$

where the magnetometer covariance $\mathbf{R}_m$ is estimated from training data

$$\mathbf{R}_m = \begin{bmatrix} 0.9197 & 0.1279 & -0.3767 \\ 0.1279 & 0.5496 & 0.0331 \\ -0.3767 & 0.0331 & 1.1012 \end{bmatrix} \quad (30)$$

```

1 function [x, P] = mu_m(x, P, mag, m0, Rm)
2 % mu_m Extended Kalman Filter measurement update for magnetic field-based
3   quaternion estimation
4 % --- Extended Kalman Filter Update ---
5 % 1. Compute the expected measurement and its Jacobian at the predicted state
6
7 % Here fm is zero, no acceleration
8 fm = zeros(3,1);
9 y_x = Qq(x)' * (m0 + fm);

```

```

10
11 % Calculate the Hx matrix
12 [Q0, Q1, Q2, Q3] = dQqdq(x); % 3x4 matrices
13 Hx = [Q0' * m0, Q1' * m0, Q2' * m0, Q3' * m0]; % 3x4
14
15 % 2. Calculate the innovation (difference between actual and expected measurement)
16 innov = mag - y_x;
17
18 % 3. Compute the innovation covariance
19 % Formula: S = H * P * H' + R
20 S = Hx * P * Hx' + Rm;
21
22 if rcond(S) < 1e-10, S = S + 1e-6 * eye(3); warning('Regularized S'); end
23
24 % 4. Compute the cross-covariance between state and measurement
25 % Formula: C = P * H'
26 crossCov = P * Hx';
27
28 % 5. Calculate the Kalman gain
29 % Formula: K = C * inv(S)
30 K = crossCov / S;
31
32 % 6. Update the state mean
33 % Formula: x = x_pred + K * innov
34 x = x + K * innov;
35
36 % 7. Update the state covariance
37 % Formula: P = P - K * S * K'
38 P = P - K * S * K';
39
40 % Important: normalize the quaternion
41 [x, P] = mu_normalizeQ(x, P);
42 end

```

Listing 6: MATLAB function for EKF measurement update with magnetic field-based quaternion estimation

10 Task 10

10.1 Estimation after adding magnetometer measurements

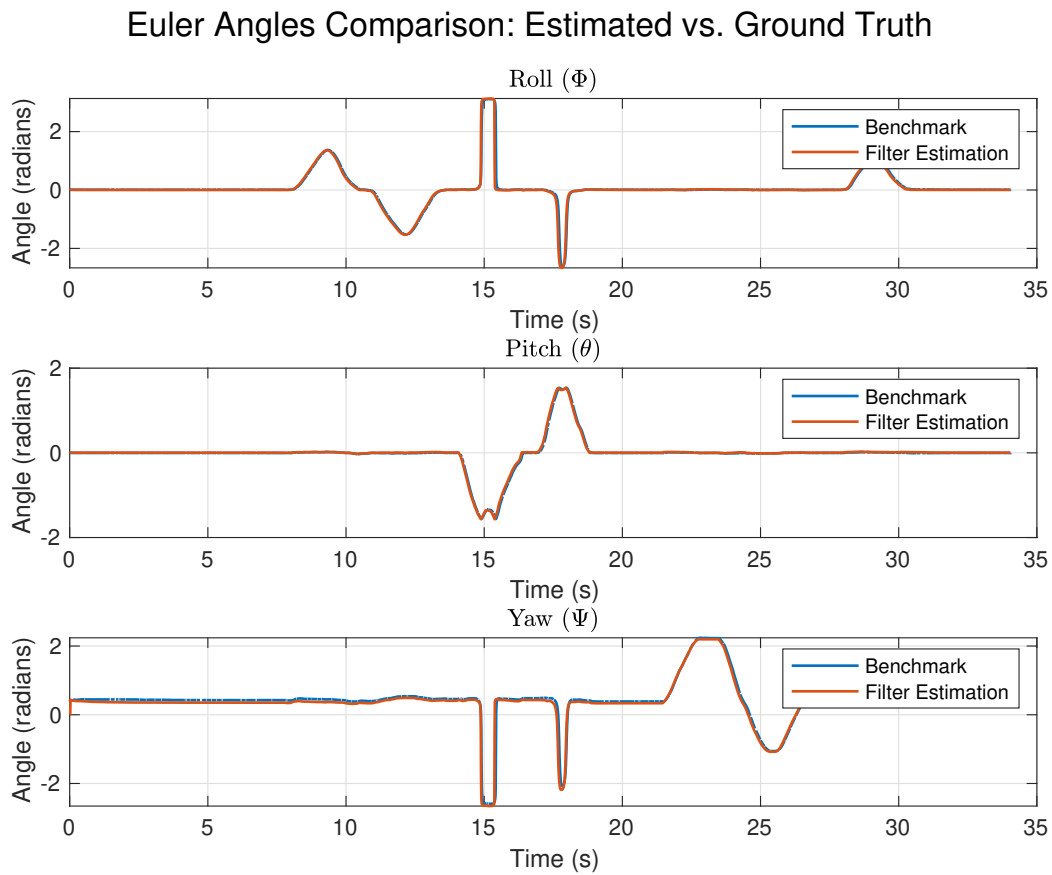


Figure 11: Euler Angles estimations by using the gyroscope, accelerometer and magnetometer measurements.

Figure 11 shows the result after the magnetometer measurement update is added. According to the yaw benchmark, it can be seen that even though the prior yaw is unknown, it aligns perfectly with the benchmark. This is contributed by the magnetometer, as it knows the earth magnetic field direction.

10.2 Magnetometer Estimation with magnetic disturbance

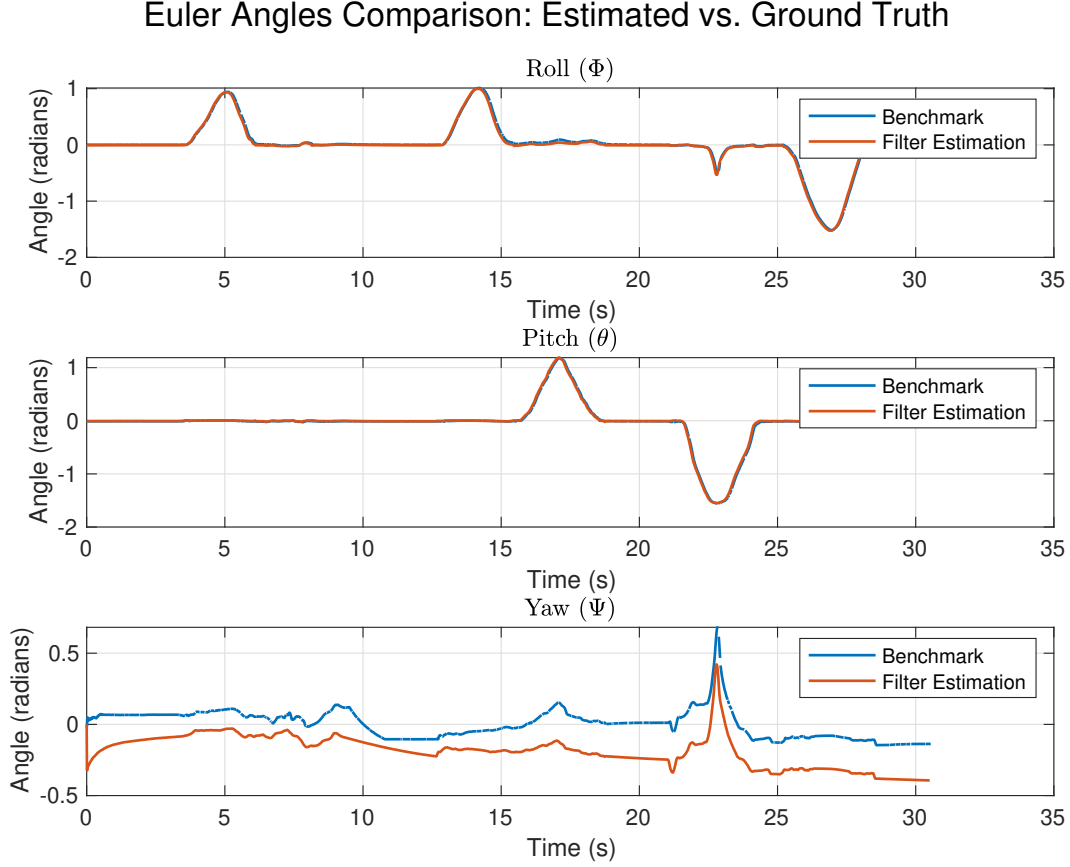


Figure 12: Euler Angles estimations by using the gyroscope and accelerometer measurements when the magnetic disturbance is introduced.

Figure 12 shows the estimation result when the magnetic disturbance is introduced. It can be seen that the yaw estimation cannot align well with the benchmark anymore.

This is because the accurate magnetometer measurement model is:

$$\mathbf{y}_k^m = \mathbf{Q}^\top(\mathbf{q}_k)(m_0 + \mathbf{f}_k^m) + \mathbf{e}_k^m \quad (31)$$

where $\mathbf{f}_k^m$ is the additional magnetic field brought by the magnetic disturbance. As $\mathbf{f}_k^m$ is ignored by our simplified model, the magnetometer update performs poorly when magnetic disturbance is added.

11 Task 11

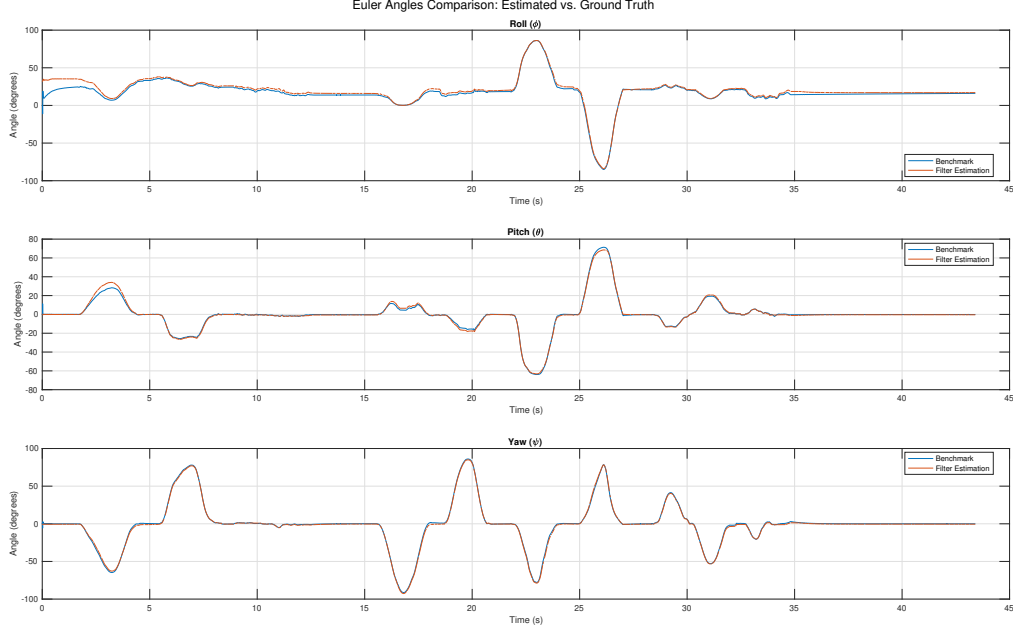


Figure 13: Euler Angles estimations by using the gyroscope and accelerometer measurements when the magnetic disturbance is introduced. Outlier rejection is implemented to skip the magnetometer updates when the measurement is regarded to be an outlier.

To solve the problem that the simplified magnetometer measurement model performs poorly when the magnetic disturbance is introduced, the outlier rejection algorithm is applied for f_k^m . When the error between $\|m_k\|$ and L_k is larger than the threshold, the measurement is regarded an outlier. Then the magnetometer update would be skipped in this timestep.

L_k is the expected magnitude, which updates according to the formula:

$$L_k = (1 - \alpha)L_{k-1} + \alpha\|m_k\| \quad (32)$$

to adapt the drifting sensor. $\alpha = 0.01$ as we have tuned it to give good performance. $L_k = \text{norm}(m_0)$ as initial value for L_k .

The results after applying the outlier rejection algorithm is shown in Figure 13. The EKF is tested when the magnetic disturbance is introduced as in Task 10. Compared to the result shown in Figure 12, it can be seen that the yaw angel can now align well with the benchmark, even the model is simplified and f_k^m is assumed to be zero.

Furthermore, `setMagDist` function is implemented in the orientation view to indicate when the magnetic field measurements are considered disturbed/outliers. The results are shown in Figure 14 and Figure 15.

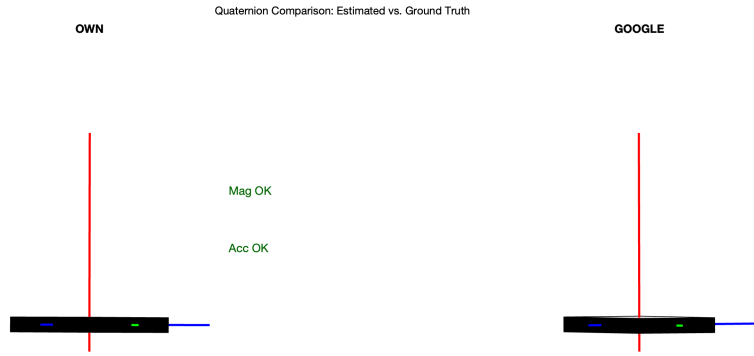


Figure 14: Our phone orientation estimation in comparison with Google orientation estimation without magnetic disturbances as shown from the the flag, Mag OK in green color.

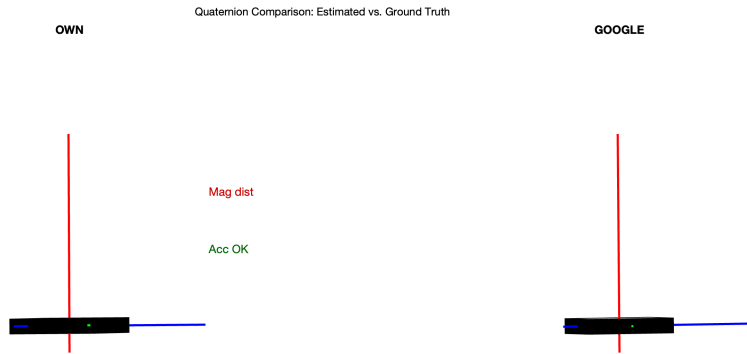


Figure 15: Our phone orientation estimation in comparison with Google orientation estimation with magnetic disturbance as shown from the the flag, Mag dist in red color.

The code used is listed below:

```

1 % Magnetometer
2 mag = data(1, 8:10)';
3 if ~any(isnan(mag)) % Mag measurements are available.
4     % AR(1) filter: estimate Lk
5     Lk = (1-alpha) * Lk + alpha * norm(mag);
6
7     % Reject the outliers
8     if Lk < 40 || Lk > 55 % outlier out of [40, 55] based on our location
9         ownView.setMagDist(1);
10    else
11        [x, P] = mu_m(x, P, mag, m0, Rm);
12        ownView.setMagDist(0);
13    end
14 end

```

Listing 7: MATLAB code for magnetometer data processing and EKF update

12 Task 12

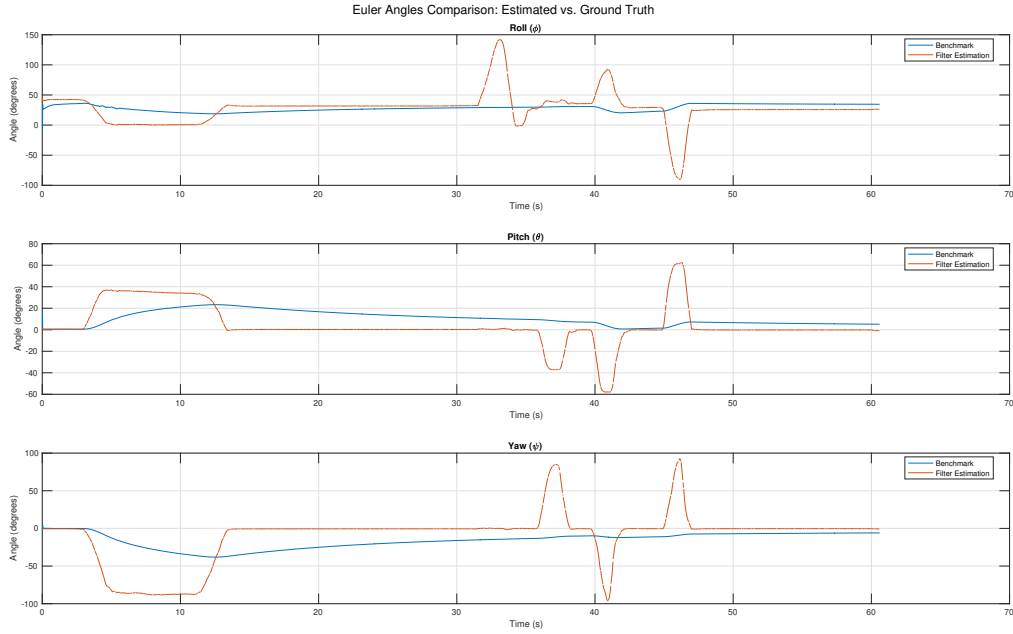


Figure 16: Euler Angles comparison between the ground truth and the EKF estimations by using only accelerometer and magnetometer measurements.

Our EKF is evaluated for different combinations of the sensors.

As shown in Figure 16, only accelerometer and magnetometer updates are applied, and the gyroscope prediction with gyro data is skipped but we kept a motion model (We just pass the states and adding noise). Also, we tried to skip the motion model but we got the same results. It can be seen that without the gyroscope, EKF works poorly. This is because the gyroscope provides direct measurements of angular velocity, which are crucial for accurately predicting orientation changes over time. Without it, the filter relies solely on the accelerometer and magnetometer, which are more susceptible to noise and disturbances. The

accelerometer is mainly useful for detecting gravity direction, and the magnetometer measures the earth's magnetic field. Both sensors are insufficient for capturing fast or small orientation changes, especially during dynamic motion.