Lab 1 - UAV Attitude
Introduction to Drone Technology

Karina Bay Bloch (kablo17), Rianne Drijver (ridri21) & Nicoline L. Thomsen (nthom18)

September 2021

1 UAV attitude sensors

Gyro (MEMS)
A gyro, also known as an angular rate sensor, measures angular velocity to estimate roll, pitch and yaw. MEMS gyros are small, usually inexpensive and accurate sensors, but they measure relative and are therefore not absolute. They need to be calibrated to have a global reference, and to do this, an accelerometer is usually incorporated.

Accelerometer (MEMS)
Accelerometers measure linear acceleration. These are inaccurate sensors, to estimate position it requires a double integral which introduces double error. They are used to sense the direction of gravity.

Magnetometer (MEMS)
Used to estimate magnetic north, but the orientation of the drone is required. This estimate relies on other sensors, which makes it inaccurate. They are sensitive to disturbances, and are unreliable with metal in the proximity.

2 UAV attitude sensing using accelerometers

2.1 Calculate pitch angle

Equation 1 is used to calculate the pitch [1].

$$\tan\theta_{yxz} = \frac{G_{py}}{\sqrt{G_{px}^2 + G_{pz}^2}} \tag{1}$$

The following code snippet is equation 1 implemented in Python:

```
pitch = np.arctan2(acc_y, sqrt(acc_x**2 + acc_z**2))
```

Figure 1 shows the pitch plotted over time.

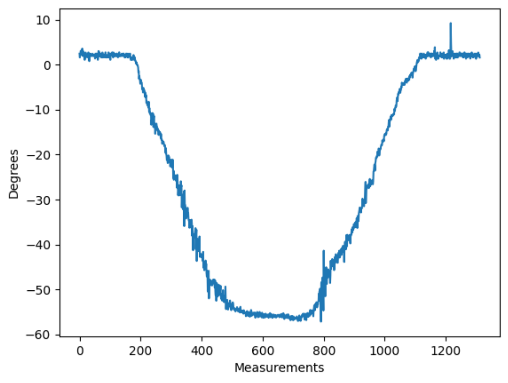


Figure 1: Plot of pitch evolving over time in the file `imu_razor_data_pitch_55deg.txt`.

2.2 Calculate roll angle

Equation 2 is used to calculate the roll [1].

$$\tan \phi_{yxz} = \frac{-G_{px}}{G_{pz}} \tag{2}$$

The following code snippet is equation 2 implemented in Python:

```
roll = np.arctan2(-acc_x, acc_z)
```

Figure 2 shows the roll plotted over time.

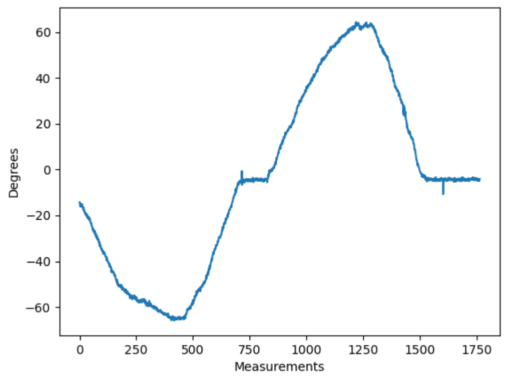
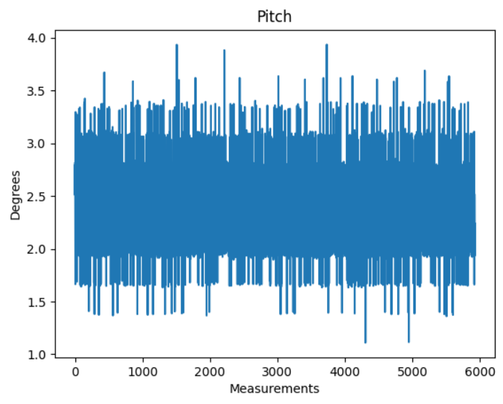


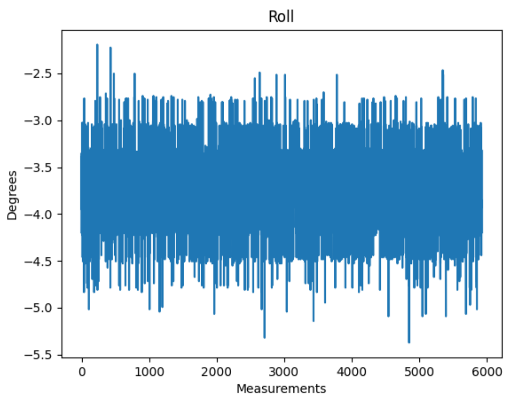
Figure 2: Plot of roll evolving over time in the file `imu_razor_data_roll_65deg.txt`.

2.3 Accelerometer noise

Equation 1 and 2 is used on the dataset `imu_razor_data_static.txt` to generate the plots of pitch and roll in figure 3.



(a) Pitch



(b) Roll

Figure 3: Plot of pitch and roll evolving over time in the file `imu_razor_data_static.txt`.

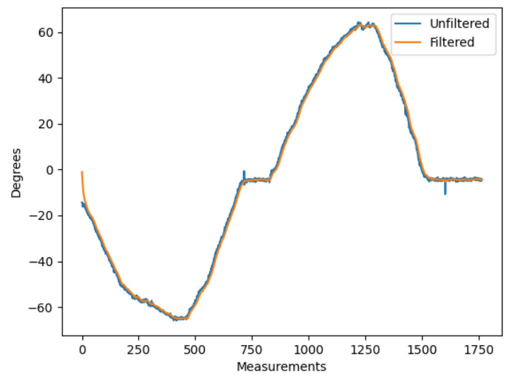
The amount of noise itself is not critical, yet the bias is misplaced in both circumstances, which could result in drifting. A low pass filter can be one solution, but using a kalman filter can be a better choice for such a situation, because it's ideal for systems, that are continuously changing, because it's light on memory and very fast.

2.4 Low-pass filtering

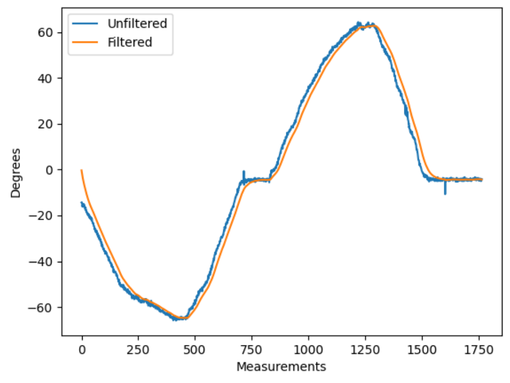
A low-pass filter can be applied to reduce higher frequency noise from the signal. The sampling time determined from the data set was estimated to be 101.6 Hz (`f_sample`). A cutoff frequency of 10Hz and 3Hz was chosen (`f_cutoff`). The following code shows how to obtain the filtered signal, where `acc_x_array` is the signal to filter:

```
from scipy import signal
wn = f_cutoff / (f_sample * 2)
sos = signal.butter(1, wn, btype='low', analog=False, output='sos', fs=None)
filtered = signal.sosfilt(sos, acc_x_array)
```

Figure 4 compares the filtered and non-filtered signal at different cut-off frequencies. It is clear that using a low-pass filter most dominant noise can be mitigated. However, this type of filtering introduces a delay, which is unfavourable in a UAS control system, that typically operates at frequencies higher than 100Hz.



(a) Cut-off = 10Hz



(b) Cut-off = 3Hz

Figure 4: Plot of roll evolving over time in the file `imu_razor_data_roll_65deg.txt`, without and with filter, at different cut-off frequencies.

(╯°□°) ╯︵ ┻━┻

2.5 Limitations of Euler angles

Using Euler angles, and considering an object with a “front”; if it were to point straight upwards, the pitch and yaw would become aligned, leading to rotation in a 2-dimensional space. This is because all rotations would occur around yaw.

This problem can be mitigated by the use of Quarternions, they avoid the issue of Gimbal locks because it is a 4D representation, instead of a 3D representation like Euler angles. This results in the ability to describe all possible states of orientation.

3 Gyro measurements

3.1 Calculating relative angle

$$\sum_{0}^{T} \omega \cdot \Delta t \tag{3}$$

The formula for discrete integration is written in equation 3. This formula is implemented in the following code and a plot of the estimated angles are show in figure 5.

```
gyro += gyro_z * (ts_now - ts_prev)
```

The shape and values of this plot aligns with the description of the data set.

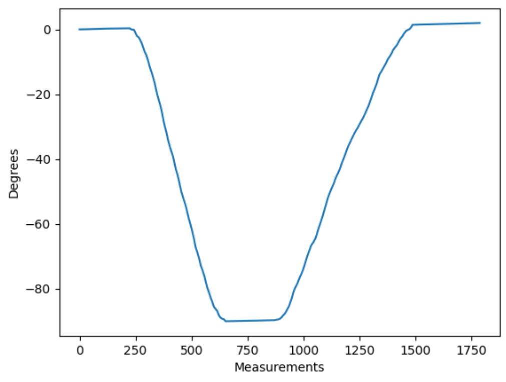


Figure 5: Plot of yaw evolving over time in the file `imu_razor_data_yaw_90deg.txt`.

3.2 Static data

The same method of integration from the previous exercise is applied here. Figure 6 shows the plot. The graph shows the relative angle drifting, illustrating the bias of the gyro sensor.

(╯°□°）╯︵ ┻━┻

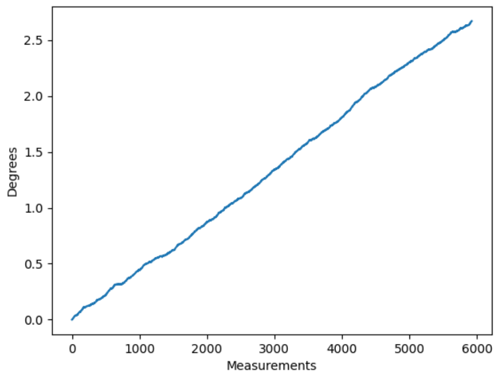


Figure 6: Plot of yaw evolving over time in the file `imu_razor_data_static.txt`.

3.3 Observing bias

From figure 6 the drifting angle error is demonstrated. It is reasonable to assume this bias to be the linear function $b(x)$ in equation 4. For every integration step the bias $b(x)$ is subtracted, yielding the plot in figure 7.

$$b(x) = \frac{2.67}{5924}x \tag{4}$$

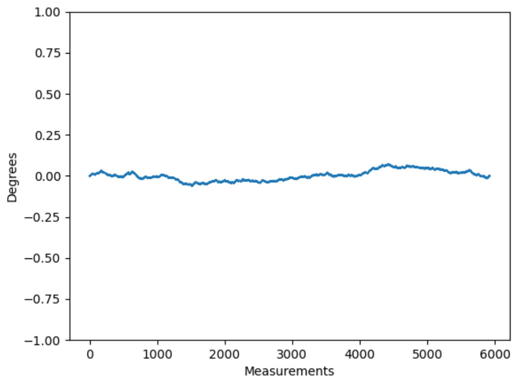


Figure 7: Plot of yaw evolving over time in the file `imu_razor_data_static.txt` where the bias has been subtracted.

3.4 Bias sources

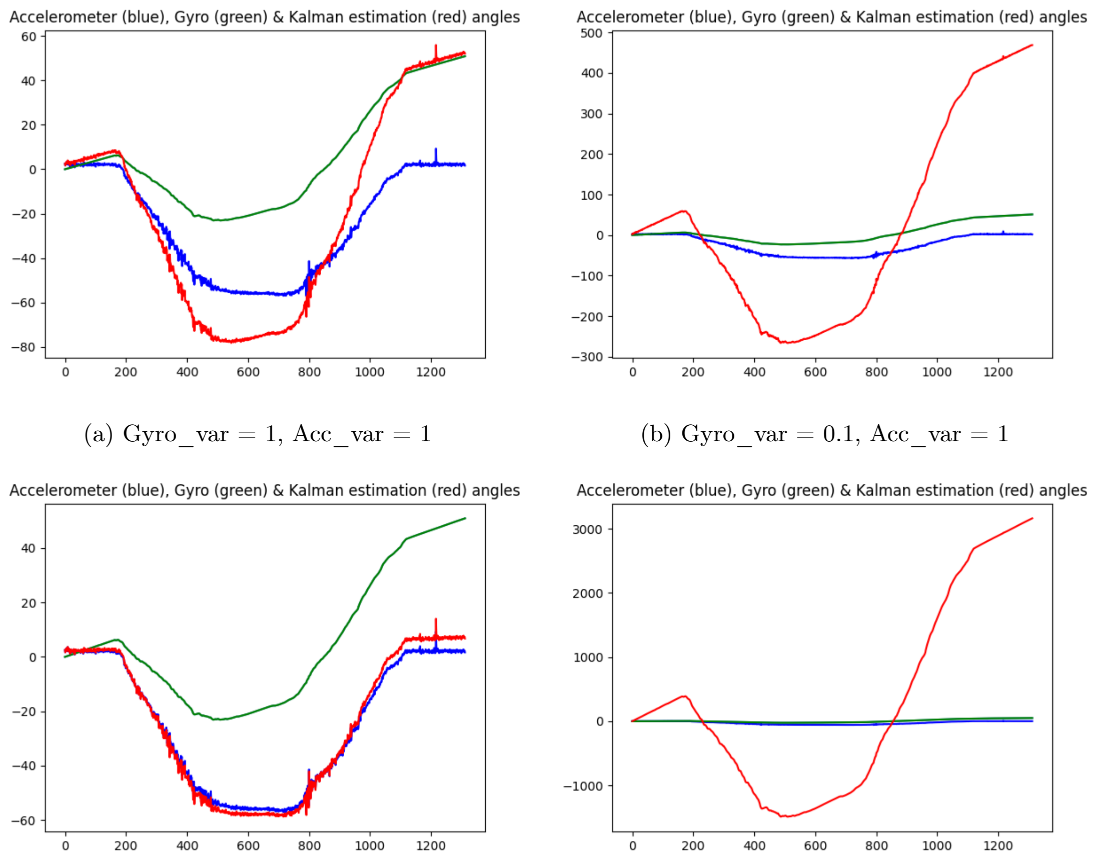
A gyro is a non-absolute sensor, that produces relative measurements. It needs to be calibrated to have a reference zero-initial reading, however this initial value can cause drift due the the integration of noise within the device [2].

4 Kalman filter

The following code is the implementation of the 1D Kalman filter, followed the example from [3]. As seen in figure 8 the Kalman estimation is not as expected. The shape seems to follow the expected behaviour; that the shape is dominated by the sensor with the lowest variance. However it seems to be scaled by the ratio of the variances. We were not able to identify the issue.

```
# Prediction Step
predAngle = estAngle + gyro_x_rel * T
predVar = estVar + gyroVar * T
estAngle = predAngle
estVar = predVar

# Update Step K = predVar / (predVar + accVar)
corrAngle = predAngle + K * (pitch - predAngle)
corrVar = predVar * (1 - K)
estAngle = corrAngle
estVar = corrVar
kalman_estimate = estAngle
```



(a) Gyro_var = 1, Acc_var = 1

(b) Gyro_var = 0.1, Acc_var = 1

(c) Gyro_var = 1, Acc_var = 0.1

(d) Gyro_var = 0.01, Acc_var = 1

Figure 8: Plots of the Kalman estimate at different sensor variances.

5 Bibliography

[1] Mark Pedley. Tilt sensing using a three-axis accelerometer. Provided document from ItsLearning.

[2] Ian Beavers. The case of the misguided gyro. `https://www.analog.com/en/analog-dialogue/raqs/raq-issue-139.html`. Online; accessed 18 September 2021.

[3] Kjeld Jensen. Kalman filter notes. Provided document from ItsLearning.

[4] MuRata. Gyro sensors. `https://www.murata.com/en-global/products/sensor/library/mems-basic/gyro`. Online; accessed 14 September 2021.