

# Project: Estimation Writeup

Determine the standard deviation of the measurement noise of both GPS X data and Accelerometer

X data: As suggested, I ran scenario 6 NoisySensors to collect data from the two noisy sensors. I then wrote up a c++ script which:

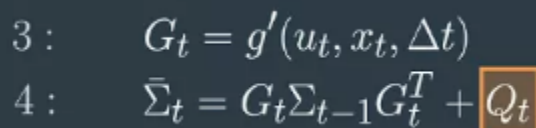
- read the values from the file
- calculated the mean of the collected samples
- calculated the standard deviation of the collected samples (formula included below)

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

Implement a better rate gyro attitude integration scheme in the UpdateFromIMU() function: As suggested, I used quaternions to improve the integration scheme. I used the *FromEuler123\_RPY* function to convert the Euler angles to a quaternion representation, and then integrated the measured gyro body rates using *IntegrateBodyRate*. I also confirmed that scenario 7 passes as required.

Implement all of the elements of the prediction step for the estimator: This step involved three functions.

1. PredictState: To predict the state forward, I implemented the transition model as defined in section 7.2 of the [Estimation for Quadrotors](#) document. I made sure to rotate the accel measurement (which we use as a control) into the body frame as well as accounting for gravity when predicting the z velocity.
2. GetRgbPrime: To get the partial derivate of the Rbg matrix, I once again used the definition provided in the document linked above. Each entry is populated separately, with rows separate by a new line.
3. Predict: To predict the covariance forward, I first populated gPrime as defined in the Estimation for Quadrotors document. I then used the formula shown below to predict the covariance forward, accounting for the transition model covariance Q.



3 :  $G_t = g'(u_t, x_t, \Delta t)$   
4 :  $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + Q_t$

Implement the magnetometer update: The measurement model for the magnetometer was quite straightforward. The only challenge was about how to normalize the difference between the measured yaw from the magnetometer and the predicted yaw from the ekf state. I normalized the difference to the range  $[-\pi, \pi]$ .

Implement the GPS update: For the GPS, the measurement model was once again quite simple. *hPrime* was also filled as required.

Meet the performance criteria of each step: Each of the success criteria in the document are met and pass the test when run in the simulator.

De-tune your controller to successfully fly the final desired box trajectory with your estimator and realistic sensors: Control parameters were detuned such that the scenario 11 GPS update passes with my implementation of the controller as well as my control parameters.