Estimator Writeup
Ashutosh Zade
July 1st, 2018

The purpose of the project is to build an EKF estimator for the QuadControl and test it using a simulator.

The tasks are as follows:

1. Sensor noise -- the objective is to find the correct mean of the sensor measurements to estimate the noise using most basic technique of finding arithmetic mean.

Following sensor measurements are used.

a. GPU x signal
b. Accelerometer x signal

2. Attitude or Pose estimation

This is done using complementary filter type attitude filter

I reviewed few internet resources on complementary filter and found below to be effective in explaining it well.

The purpose of the complementary filter is to use gyro and acc reading complementary to each other. To combine noisy acc reading (accurate over longter time) with low noisy Gyro reading (accurate over shorter interval)

The Balance Filter -- Shane Colton <scolton@mit.edu)
http://d1.amobbs.com/bbs_upload782111/files_44/ourdev_665531S2JZG6.pdf

2-Axis Accelerometer:
Measures ,acceleration, but really force per unit mass. (F = ma, so a = F/m)
Can be used to measure the force of gravity. Above, X-axis reads 0g, Y-axis reads -1g.

Gyroscope:
Measures angular rate (speed of rotation), Reads ,zero, when stationary. ,Reads positive or negative when rotating:

Some of the pros of complementary filter:

Pros:
Can help fix noise, drift, and horizontal acceleration dependency.
Fast estimates of angle, much less lag than low-pass filter alone.
Not very processor-intensive.

The complementary filter is essentially, summing Accelerometer reading from low pass filter with integrated gyro reading with high pass filter.

The details of these filters are as follows:

Integration: This is easy. Think of a car traveling with a known speed and your program is a clock that ticks once every few milliseconds. To get the new position at each tick, you take the old position and add the change in position. The change in position is just the speed of the car multiplied by the time since the last tick, which you can get from the timers on the microcontroller or some other known timer. In code: position += speed*dt;, or for a balancing platform, angle += gyro*dt;.

Low-Pass Filter: The goal of the low-pass filter is to only let through long-term changes, filtering out short-term fluctuations. One way to do this is to force the changes to build up little by little in subsequent times through the program loop. In code:
angle = (0.98)*angle + (0.02)*x_acc;
If, for example, the angle starts at zero and the accelerometer reading suddenly jumps to 10o, the angle estimate changes like this in subsequent iterations:

If the sensor stays at 10o, the angle estimate will rise until it levels out at that value. The time it takes to reach the full value depends on both the filter constants (0.98 and 0.02 in the example) and the sample rate of the loop (dt).

High-Pass Filter: The theory on this is a bit harder to explain than the low-pass filter, but conceptually it does the exact opposite: It allows short-duration signals to pass through while filtering out signals that are steady over time. This can be used to cancel out drift.

3. Prediction

Essentially, this is an implementation of state transition function based upon EKF. (re: section 7 3D Quad). It starts with definition of a rotation matrix and then transitioning from body frame to local frame and then taking a jacobian.

4. Magnetometer update

This step adds magnetometer estimation to gyro and accelerometer for better estimation of the vehicles heading vector. The step requires vehicles yaw tuning to reduce the drift.

5. GPS update

This step adds update from GPS which provides position and velocity measurements to the vehicle. Re: 7.3.1 suggests the partial derivative as the identity matrix.

```
 // GPS UPDATE
// Hints:
//  - The GPS measurement covariance is available in member variable
R_GPS
//  - this is a very simple update
```

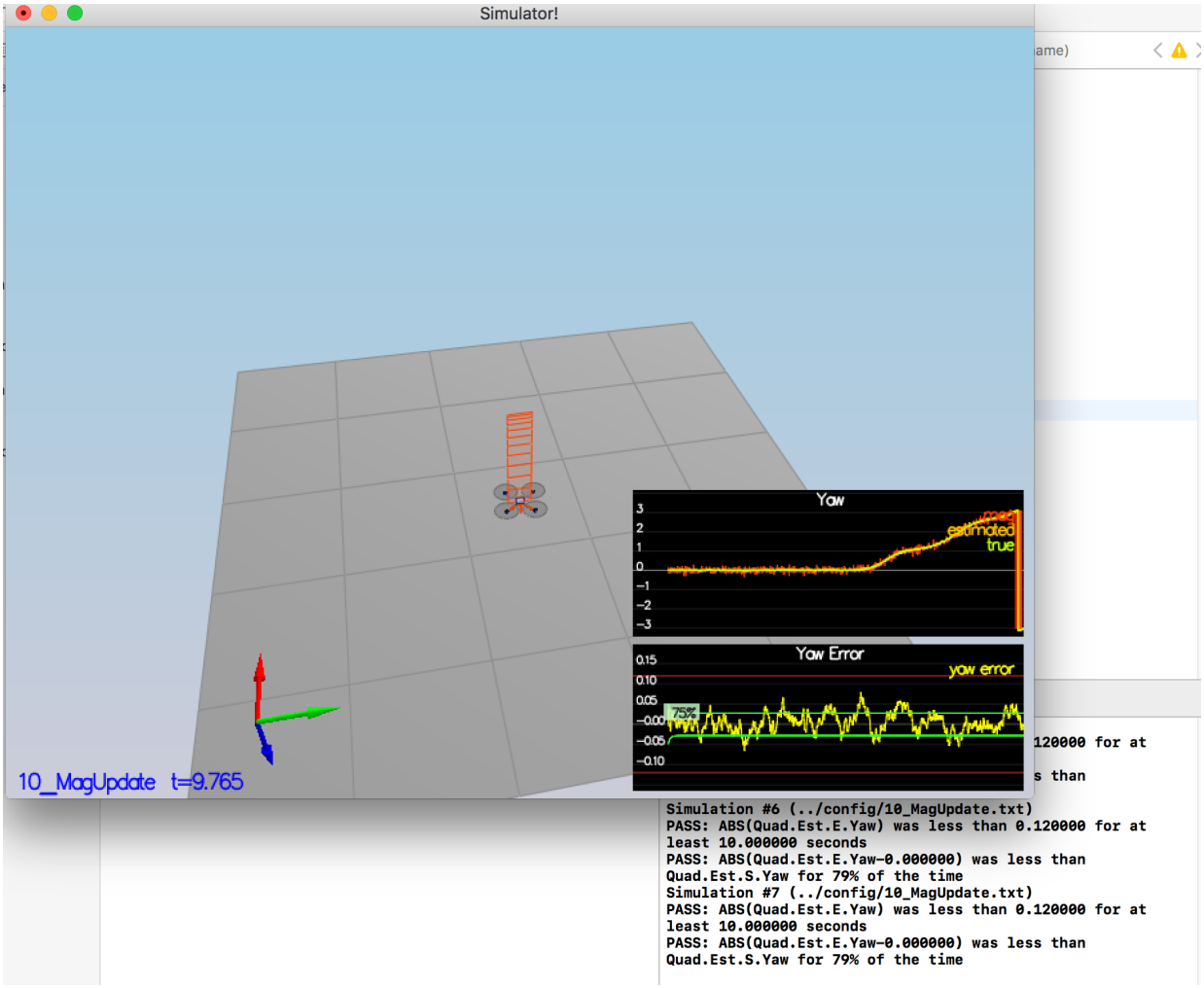
6. Adding the controller

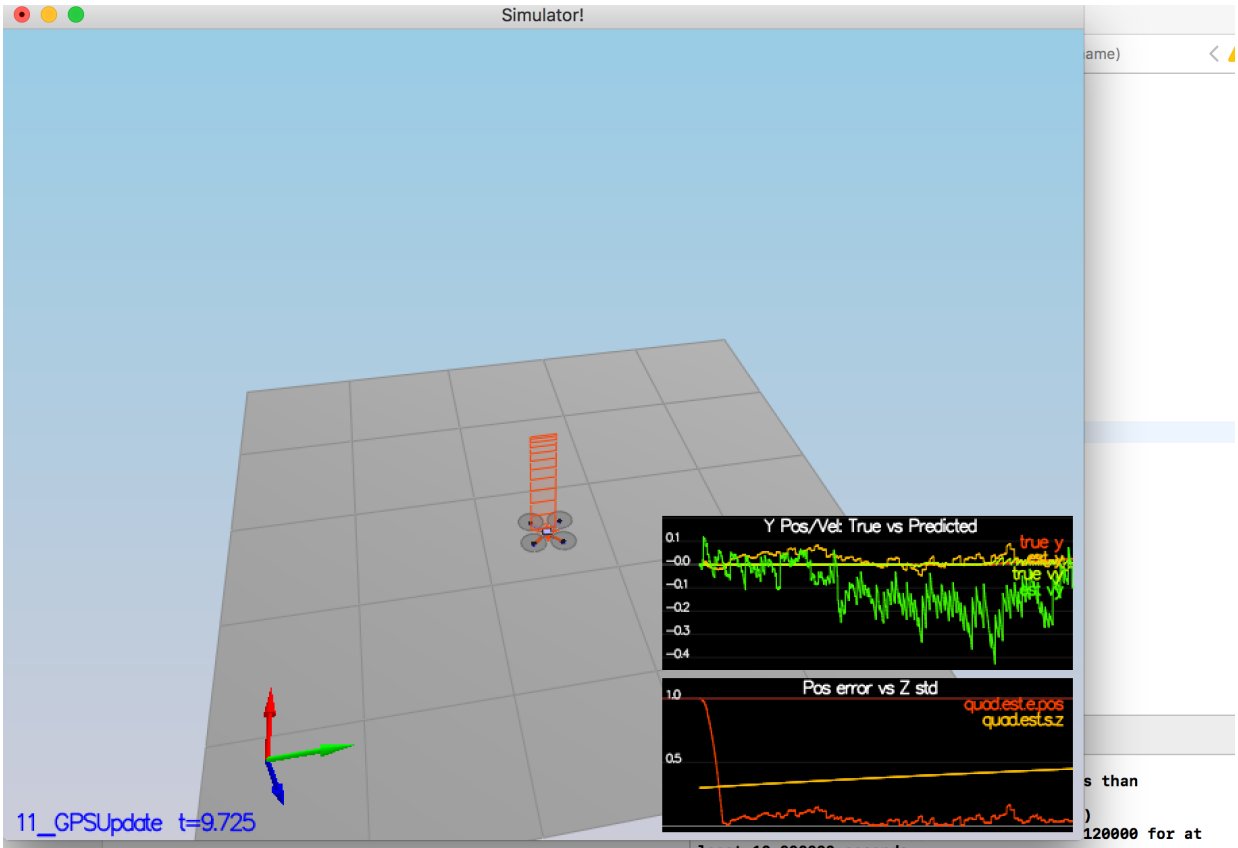
This step adds the controller used in previous project along with the parameters used earlier. The de-tuning is necessary since this project has estimated accuracy on sensors instead of assuming ideal sensors in earlier project. Decreasing position and velocity gains stabilizes the vehicle.

Output after updating with real controller from previous project:

```
PASS: ABS(Quad.GPS.X-Quad.Pos.X) was less than MeasuredStdDev_GPSPosXY
for 68% of the time
PASS: ABS(Quad.IMU.AX-0.000000) was less than MeasuredStdDev_AccelXY for
69% of the time
Simulation #4 (../config/06_SensorNoise.txt)

PASS: ABS(Quad.GPS.X-Quad.Pos.X) was less than MeasuredStdDev_GPSPosXY
for 68% of the time
PASS: ABS(Quad.IMU.AX-0.000000) was less than MeasuredStdDev_AccelXY for
69% of the time

PASS: ABS(Quad.Est.E.Yaw-0.000000) was less than Quad.Est.S.Yaw for 59%
of the time

Simulation #4 (../config/07_AttitudeEstimation.txt)
Simulation #5 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least
3.000000 seconds

PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least
3.000000 seconds
Simulation #10 (../config/08_PredictState.txt)

Simulation #5 (../config/10_MagUpdate.txt)
PASS: ABS(Quad.Est.E.Yaw) was less than 0.120000 for at least 10.000000
seconds
PASS: ABS(Quad.Est.E.Yaw-0.000000) was less than Quad.Est.S.Yaw for 79%
of the time

Simulation #6 (../config/11_GPSUpdate.txt)
Simulation #7 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000
seconds
Simulation #8 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000
seconds
```

Results after updates on 7/9 tuning post review:






```
least 10.000000 seconds
PASS: ABS(Quad.Est.E.Yaw-0.000000) was less than
Quad.Est.S.Yaw for 79% of the time
Simulation #11 (../config/11_GPSUpdate.txt)
Simulation #12 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at
least 20.000000 seconds
Simulation #13 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at
least 20.000000 seconds
```