Implement Estimator

- Determine the standard deviation of the measurement noise of both GPS X data and Accelerometer X data.
    - The data was loaded into Excel and the standard deviation was calculated. The empirical calculations for the standard deviation rounded to the expected values of 0.5 and 0.7.
- Implement a better rate gyro attitude integration scheme in the UpdateFromIMU() function.
    - The only change I made was to convert the body fixed angular rates measured by the gyro into the derivatives of the Euler angles. This 3x3 matrix was given in the 3D control module of the class and can be easily derived from the three compound rotations that make up the Euler transformation. Note: this is NOT a simple rotation since each Euler angle rotation takes place in a different reference frame and is affect by previous rotations. Therefore the 3 Euler rotation are not orthogonal and therefore the resulting transformation is not a simple orthonormal rotation. The important things is that this does account for the coupling and nonlinearities introduced when the Euler angles are not close to 0.
- Implement all the elements of the prediction step for the estimator.
    - Completing the EKF state prediction has three parts that are implemented in the Predict function in QuadEstimatorEKF.cpp.
        - Integrate the current velocity states to update the position states
        - Convert the measured acceleration into NED frame and then removing gravity.
        - Integrate this NED acceleration to update the velocity states

        - Note: predicting the yaw states is actually implemented as part of aforementioned UpdateFromIMU function. This involved calculating the derivative of yaw from the body fixed gyro measurements and then integrating the derivative of yaw to update the yaw state.
    - Next update the EKF covariance
        - The Rbg’ matrix had to implemented in the GetRbgPrime function in QuadEstimatorEKF.cpp. This involves 6 trig evaluations and a lot of book keeping.
        - Calculate g’ using the Rbg’ matrix in the Predict function in QuadEstimatorEKF.cpp
        - Use g’ to update the EKF covariance in the Predict function in QuadEstimatorEKF.cpp
- Implement the magnetometer update.
    - This is a single state update, the only tricky part of this is guaranteeing that error is handled correctly around +-pi since yaw is periodic. This took me way longer than it should have because the fmodf function that I used successfully to set yaw between -pi and +pi in the control project didn’t work properly in the estimation environment. So instead of using one line to ensure that the error around +-pi was handle properly. This had to done in four places:
        - The yaw control update had to be modified compared to my control project implementation since fmodf was not working properly. This change was made to in UpdateFromMag function in QuadControl.cpp.
        - After the yaw prediction update in UpdateFromIMU function in QuadEstimatorEKF.cpp.
        - To the yaw error in the magnetometer update in YawControl function in QuadEstimatorEKF.cpp.

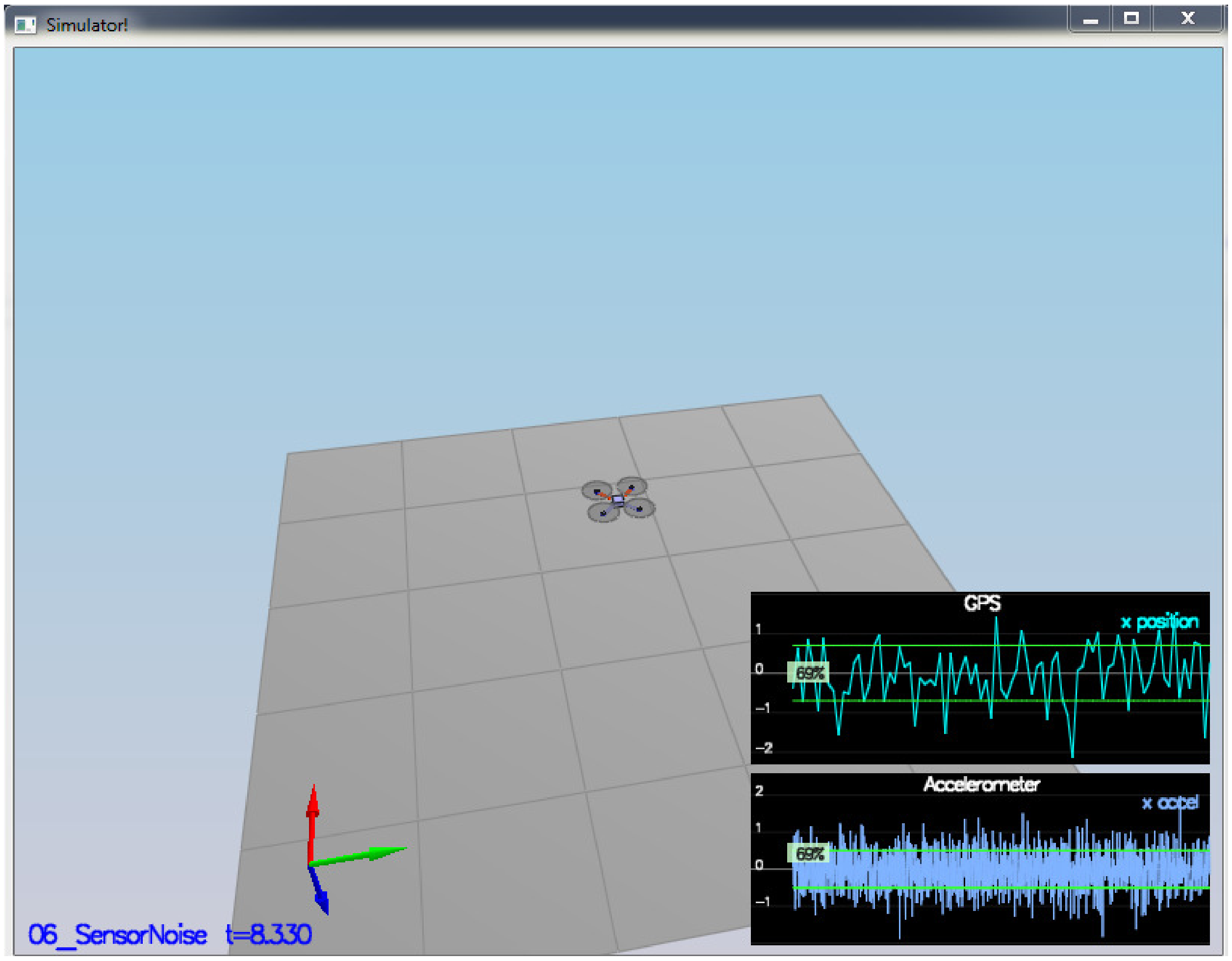
        -
        - After the EFK update from the magnetometer since the standard EKF update does not set yaw between -pi and +pi in UpdateFromMag function in QuadEstimatorEKF.cpp.

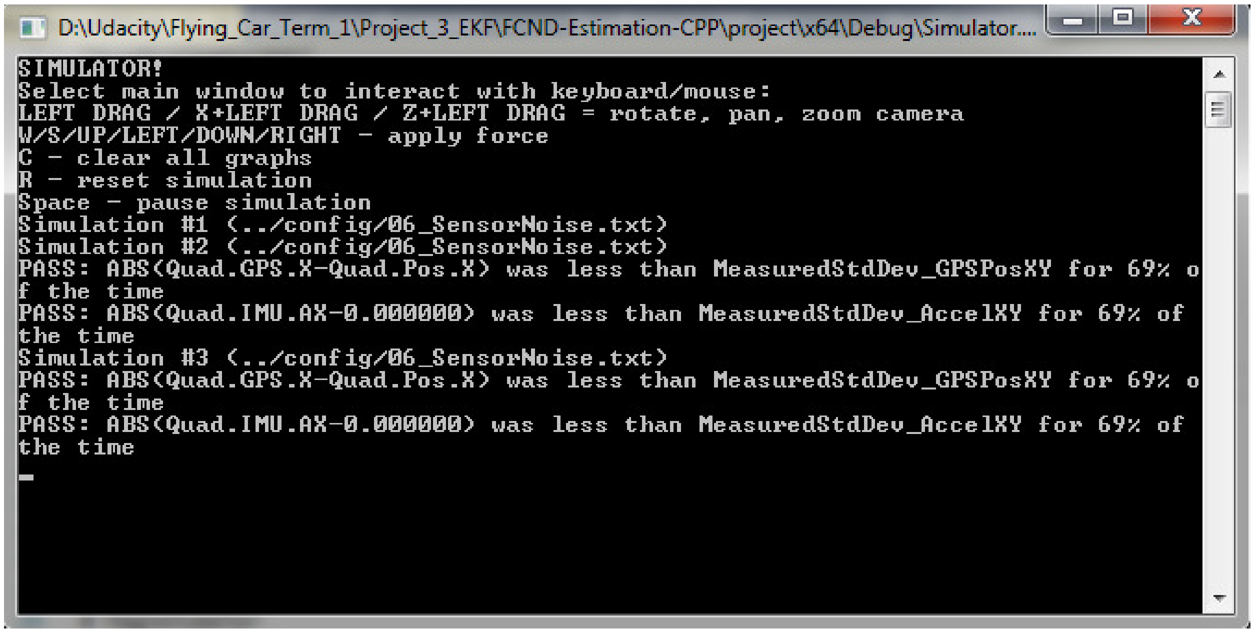
- Implement the GPS update.
    - This was implemented entirely in UpdateFromGPS function in QuadEstimatorEKF.cpp. This is a very straight forward update since the GPS measurements correspond directly to the first 6 states in the EKF

- ## Flight Evaluation

- Meet the performance criteria of each step.
    - See screen shots for scenarios 6-11 on following pages
- De-tune your controller to successfully fly the final desired box trajectory with your estimator and realistic sensors.
    - Detuning the controller was not required, setting and logic from control projects satisfied scenario 11.

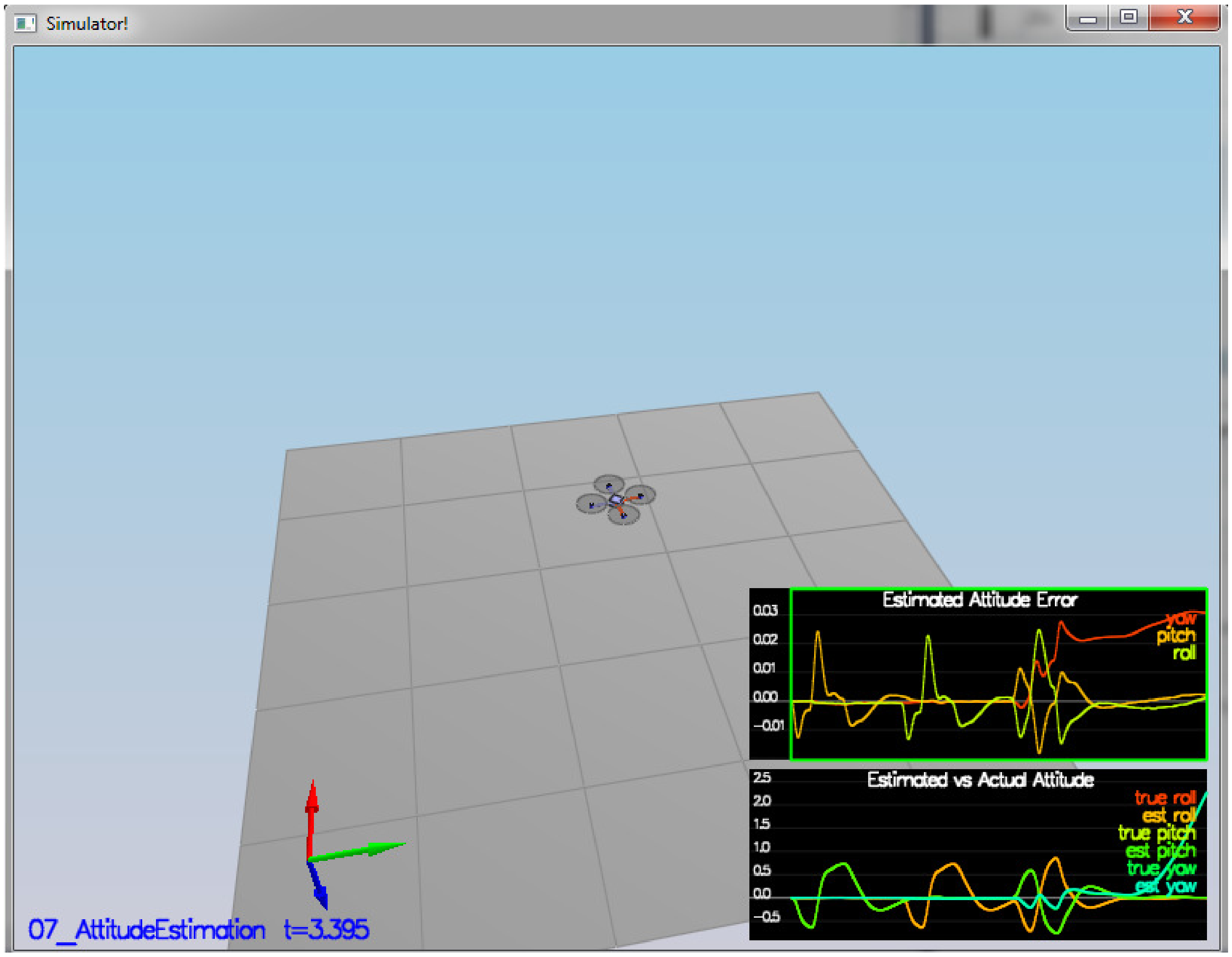
Sensor Noise – Scenario 6

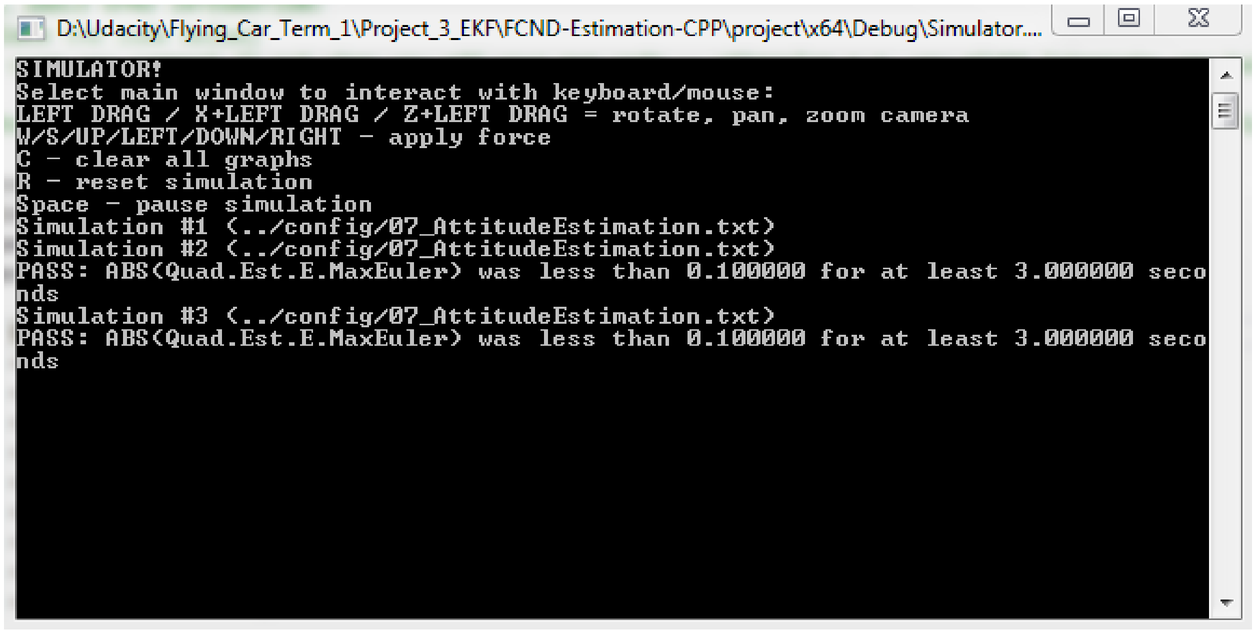



D:\Udacity\Flying_Car_Term_1\Project_3_EKF\FCND-Estimation-CPP\project\x64\Debug\Simulator....

```
SIMULATOR!
Select main window to interact with keyboard/mouse:
LEFT DRAG / X+LEFT DRAG / Z+LEFT DRAG = rotate, pan, zoom camera
W/S/UP/LEFT/DOWN/RIGHT - apply force
C - clear all graphs
R - reset simulation
Space - pause simulation
Simulation #1 (../config/06_SensorNoise.txt)
Simulation #2 (../config/06_SensorNoise.txt)
PASS: ABS(Quad.GPS.X-Quad.Pos.X) was less than MeasuredStdDev_GPSPosXY for 69% o
f the time
PASS: ABS(Quad.IMU.AX-0.000000) was less than MeasuredStdDev_AccelXY for 69% of
the time
Simulation #3 (../config/06_SensorNoise.txt)
PASS: ABS(Quad.GPS.X-Quad.Pos.X) was less than MeasuredStdDev_GPSPosXY for 69% o
f the time
PASS: ABS(Quad.IMU.AX-0.000000) was less than MeasuredStdDev_AccelXY for 69% of
the time
```

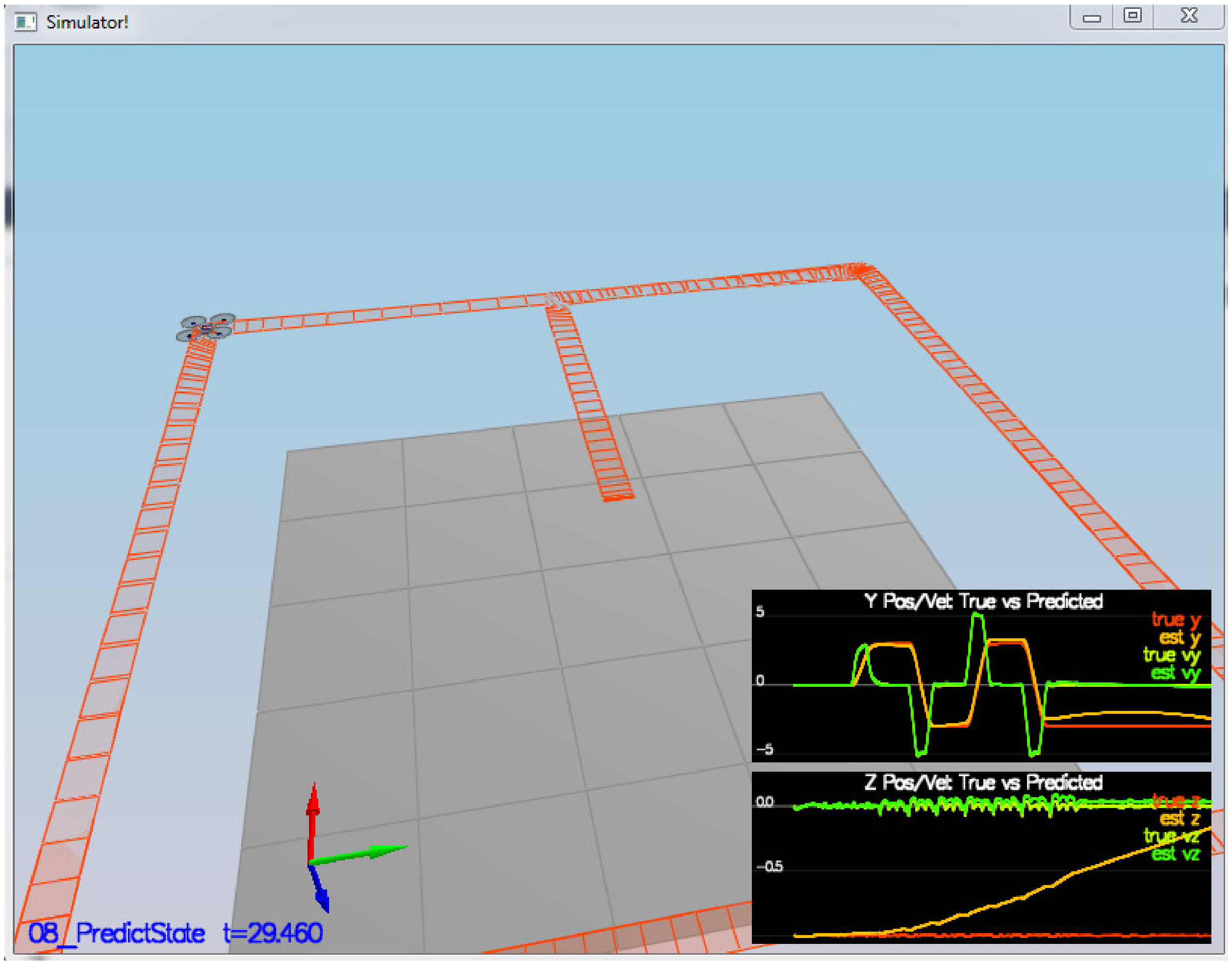
Attitude Estimation – Scenario 7

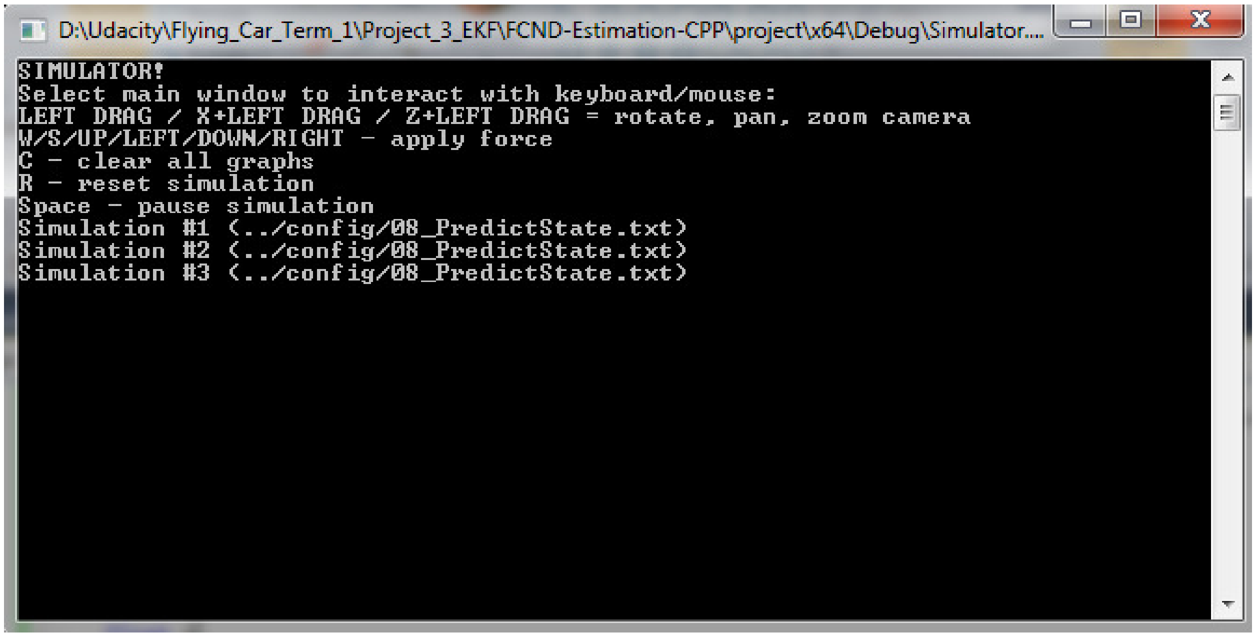



D:\Udacity\Flying_Car_Term_1\Project_3_EKF\FCND-Estimation-CPP\project\x64\Debug\Simulator....

```
SIMULATOR!
Select main window to interact with keyboard/mouse:
LEFT DRAG / X+LEFT DRAG / Z+LEFT DRAG = rotate, pan, zoom camera
W/S/UP/LEFT/DOWN/RIGHT - apply force
C - clear all graphs
R - reset simulation
Space - pause simulation
Simulation #1 (../config/07_AttitudeEstimation.txt)
Simulation #2 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seco
nds
Simulation #3 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seco
nds
```

Prediction State – Scenario 8

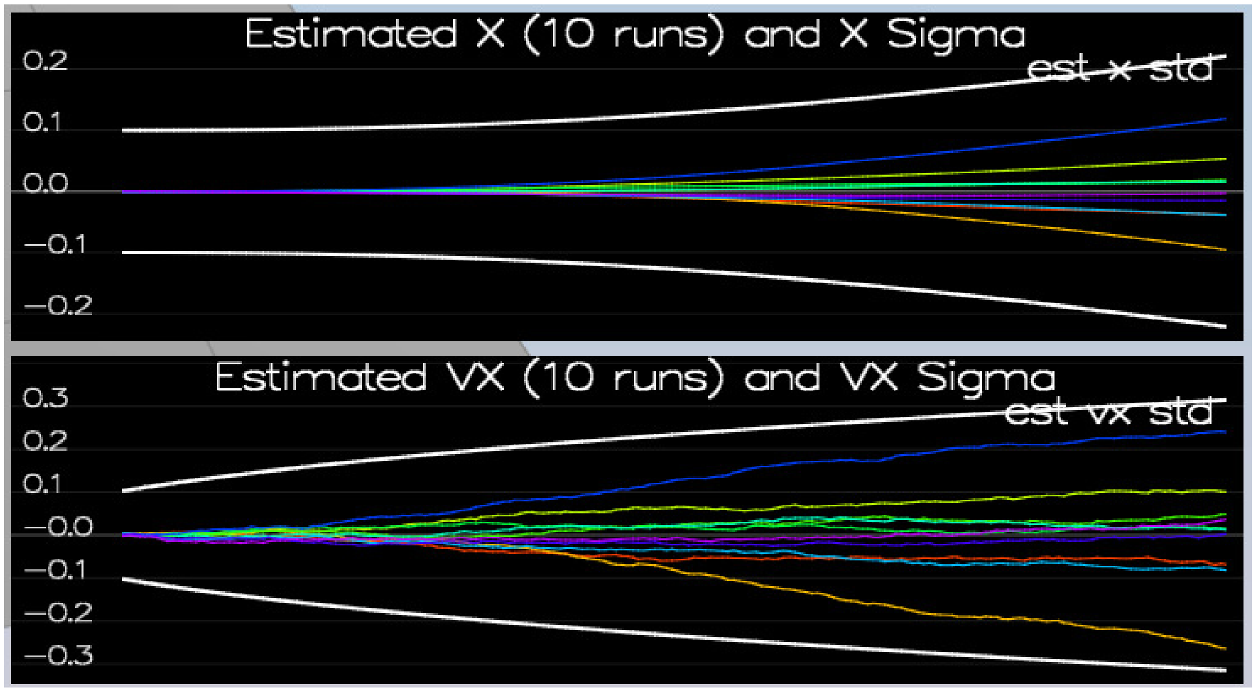



D:\Udacity\Flying_Car_Term_1\Project_3_EKF\FCND-Estimation-CPP\project\x64\Debug\Simulator....

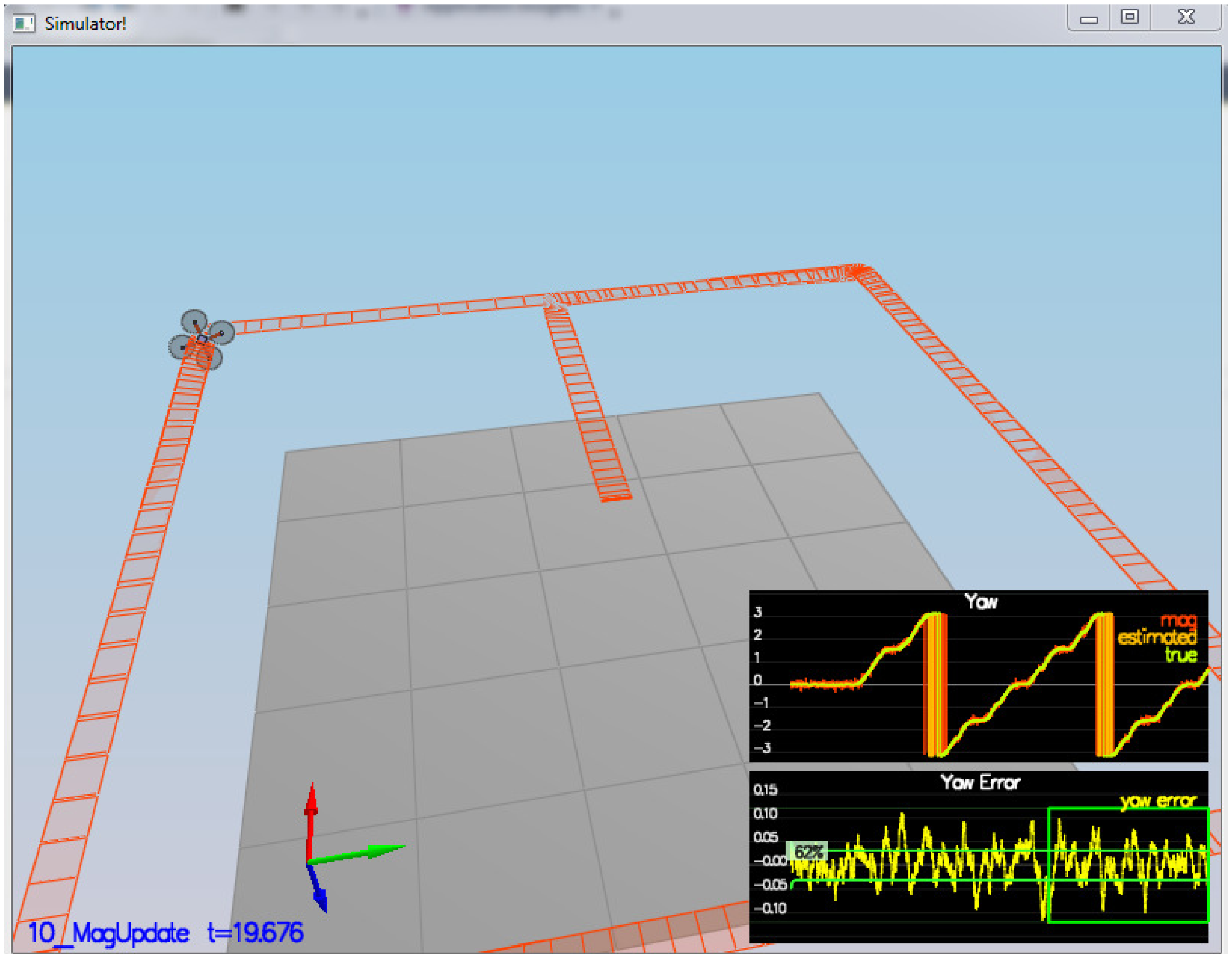
```
SIMULATOR!
Select main window to interact with keyboard/mouse:
LEFT DRAG / X+LEFT DRAG / Z+LEFT DRAG = rotate, pan, zoom camera
W/S/UP/LEFT/DOWN/RIGHT - apply force
C - clear all graphs
R - reset simulation
Space - pause simulation
Simulation #1 (../config/08_PredictState.txt)
Simulation #2 (../config/08_PredictState.txt)
Simulation #3 (../config/08_PredictState.txt)
```

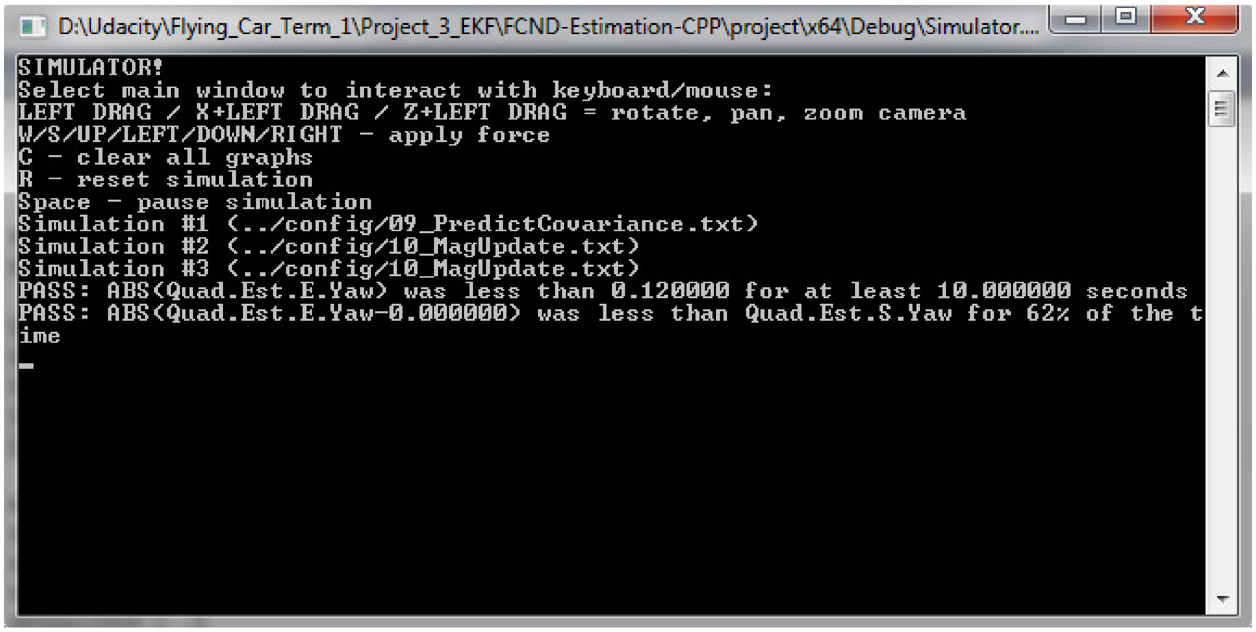
Predict Covariance – Scenario 9

The error and uncertainty in the position seems to grow at almost exactly the same rate. The error and uncertainty don't actually grow at the same rate for velocity, but they are in the ball park.



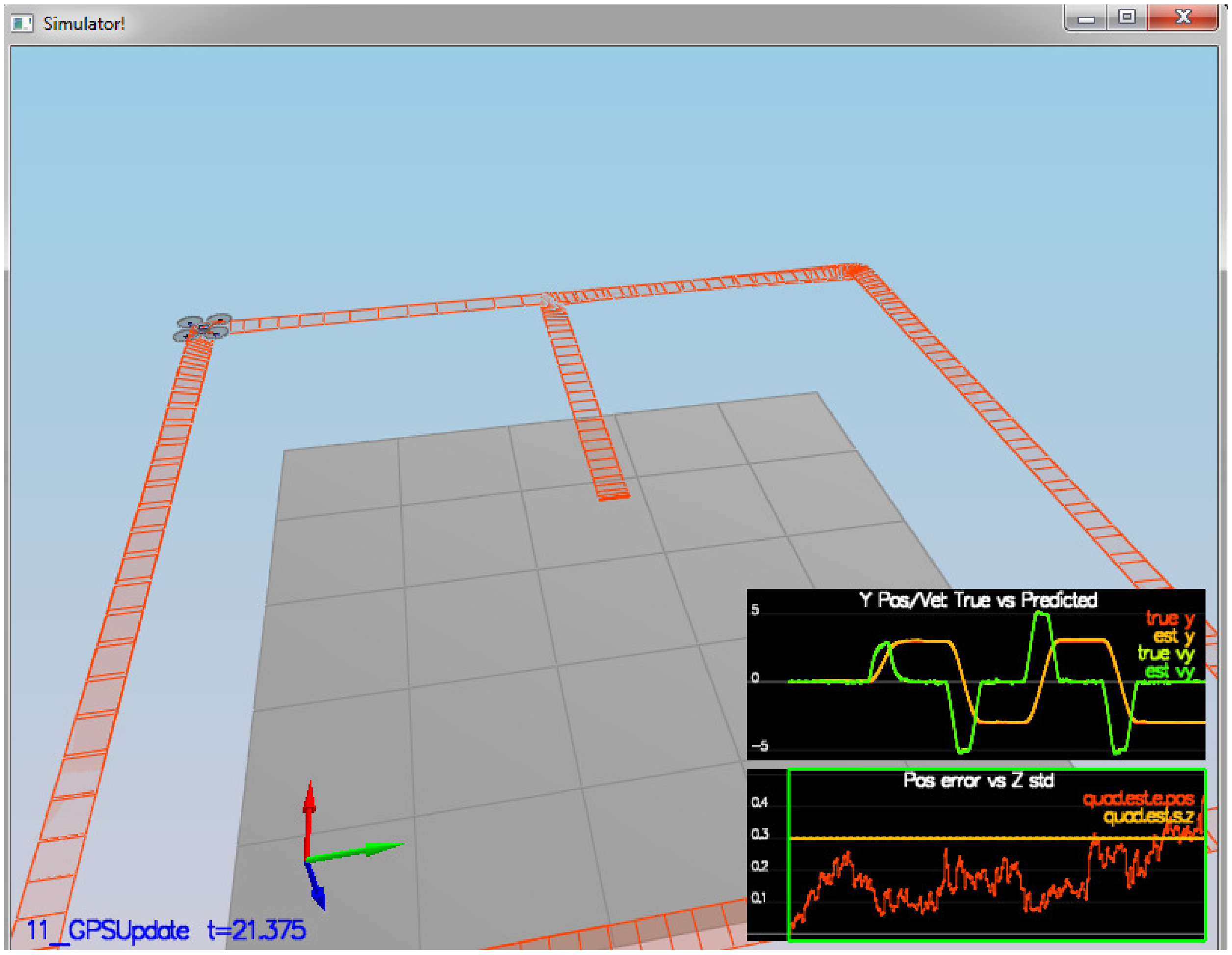
Magnetometer Update – Scenario 10

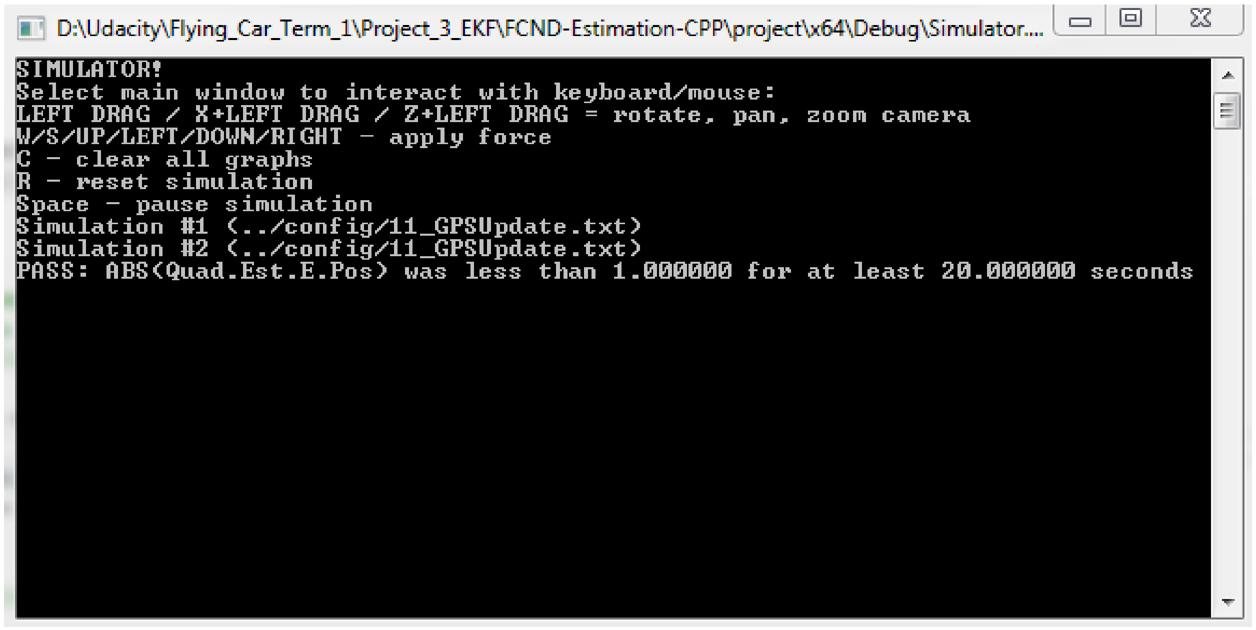



D:\Udacity\Flying_Car_Term_1\Project_3_EKF\FCND-Estimation-CPP\project\x64\Debug\Simulator....

```
SIMULATOR!
Select main window to interact with keyboard/mouse:
LEFT DRAG / X+LEFT DRAG / Z+LEFT DRAG = rotate, pan, zoom camera
W/S/UP/LEFT/DOWN/RIGHT - apply force
C - clear all graphs
R - reset simulation
Space - pause simulation
Simulation #1 (../config/09_PredictCovariance.txt)
Simulation #2 (../config/10_MagUpdate.txt)
Simulation #3 (../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 62% of the t
ime
```

Close-Loop + GPS Update – Scenario 11




D:\Udacity\Flying_Car_Term_1\Project_3_EKF\FCND-Estimation-CPP\project\x64\Debug\Simulator....

```
SIMULATOR!
Select main window to interact with keyboard/mouse:
LEFT DRAG / X+LEFT DRAG / Z+LEFT DRAG = rotate, pan, zoom camera
W/S/UP/LEFT/DOWN/RIGHT - apply force
C - clear all graphs
R - reset simulation
Space - pause simulation
Simulation #1 (../config/11_GPSUpdate.txt)
Simulation #2 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000 seconds
```