Step 1: Sensor noise

The first step was required me to find out the standard deviation of the GPS and acceleration IMU readings. The readings were stored in the log files after each run/iteration. Then I adjusted
MeasuredStdDev_GPSPosXY = .70
MeasuredStdDev_AccelXY = 0.49436808229438284
In the 06_SensorNoise.txt configuration file. After applying the new standard deviation, the two lines in the graph captures 68% of the data.

Step 2: Attitude Estimation

The second step, involved using the 7th scenario: Attitude estimation, which makes the drone to pitch, roll then yaw. In the unedited condition, the drone's sensors readings drift and accumulate throughout time. To remedy this we integrate the body-rate (PQR) using the IntegrateBodyRate function, the result will give us the estimated attitude position. Although the positions are given in quaternion format, we use the helper functions such as Pitch() and Roll() that were provided by udacity to get the body-frame attitude position.

Step 3: Prediction Step

The third step requires to implement the prediction of next states of the drone.
Along with the covariance.

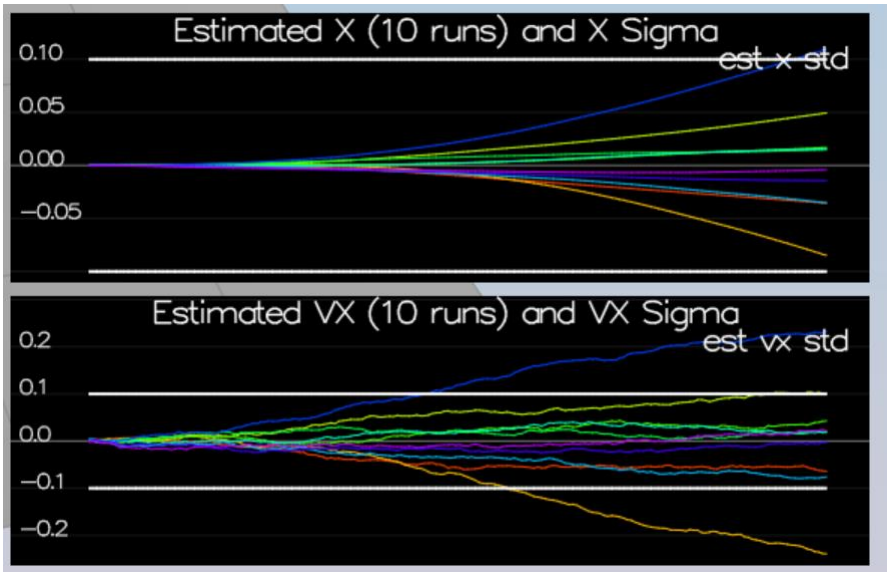

Figure 1: Default Settings

As we can see above without any implementation, the std deviation of the estimation does capture the updated covariance.

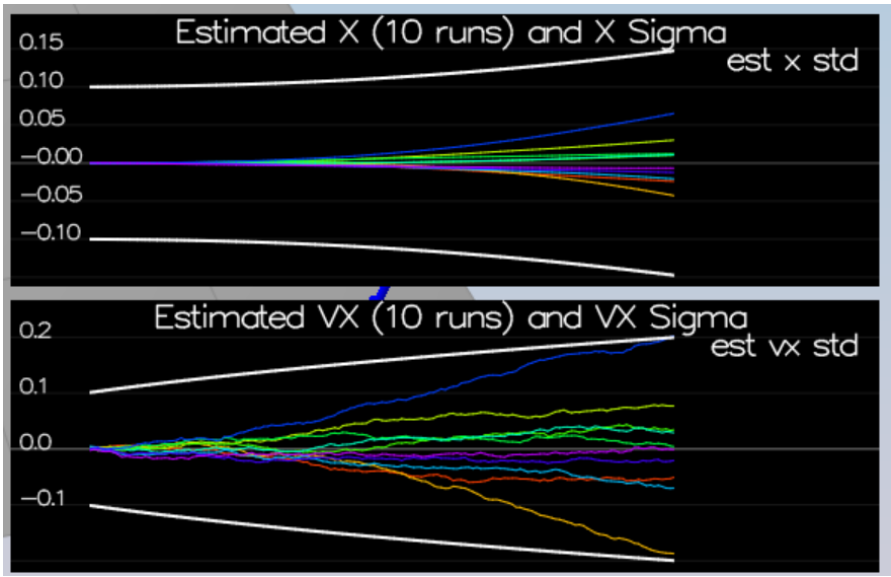

Figure 2: Covariance after Predict() implementation and tuning

After implementing the predict function we can see the standard deviation growing along with the updated covariance.

Step 4: Magnetometer Update

The next step is to update magnetometer. In the default configuration, the yaw error drifts indefinitely, due to the lack of feedback (update).

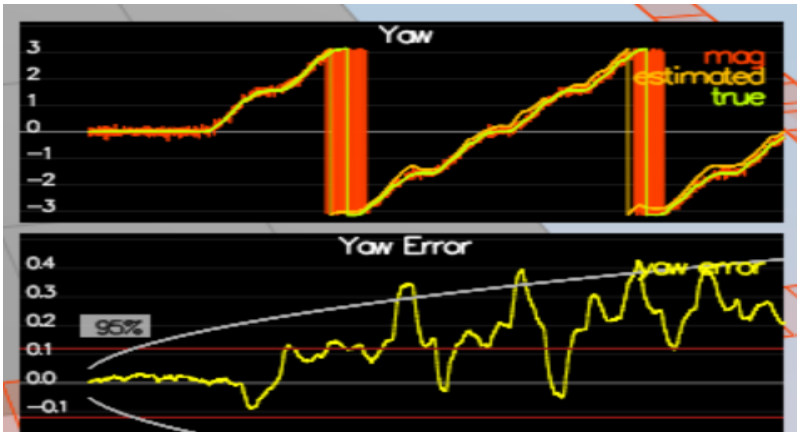

Figure 3: Without magnetometer update

Figure 4: After UpdateFromMagnetometer function implementation

Step 5: Closed Loop + GPS Update

Same with the magnetometer, in the default GPS config, the error grows indefinitely.

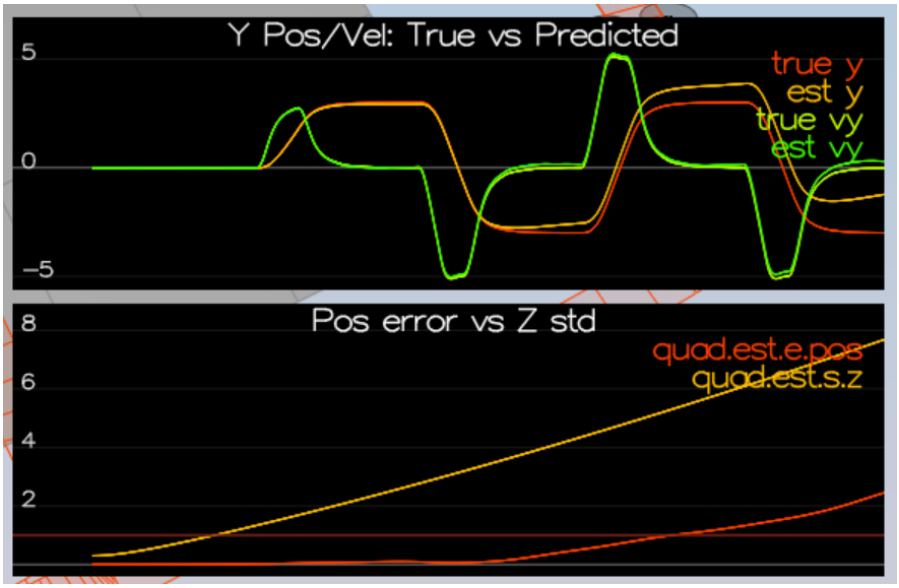

Figure 5: Without updates from the GPS

After the update, the error is lower by a huge margin!

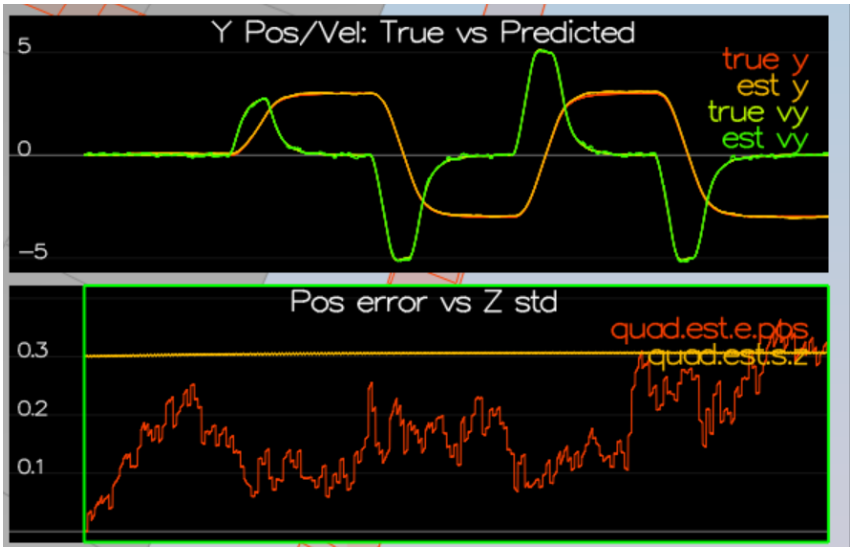

Figure 6: After UpdateFromGPS implementation

Step 6: Adding Your Controller

Final step; implementing the controller I have built in the last project. The drone was highly unstable.

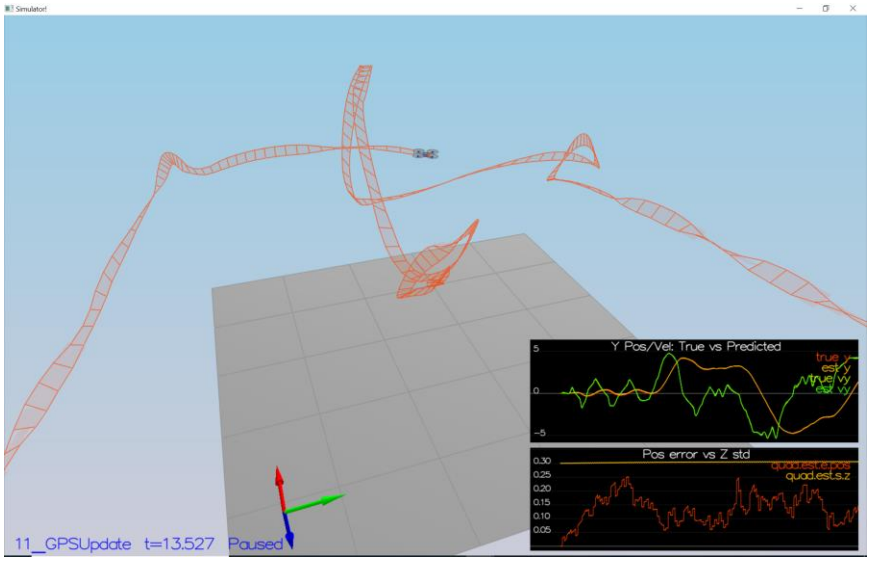

Figure 7: default controller from previous project

After relaxing the drone's control paramaters, although not perfect, was stable and able to pass the required criteria.

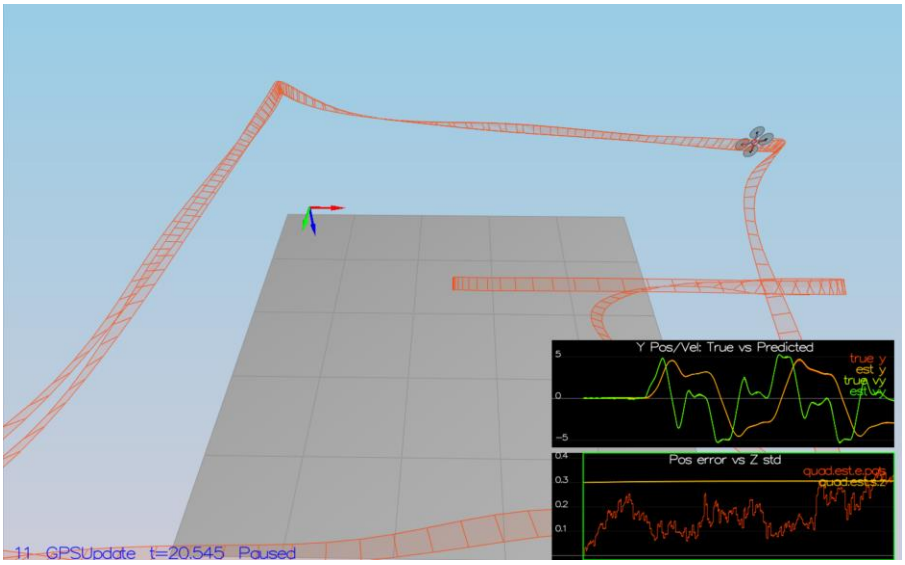

Figure 8: after detuning the control parameters