FCND Estimation Project Writeup

Grace Livingston

July 29, 2018

Contents

1 Implementation and Tests

1.1 Measurement Standard Deviation

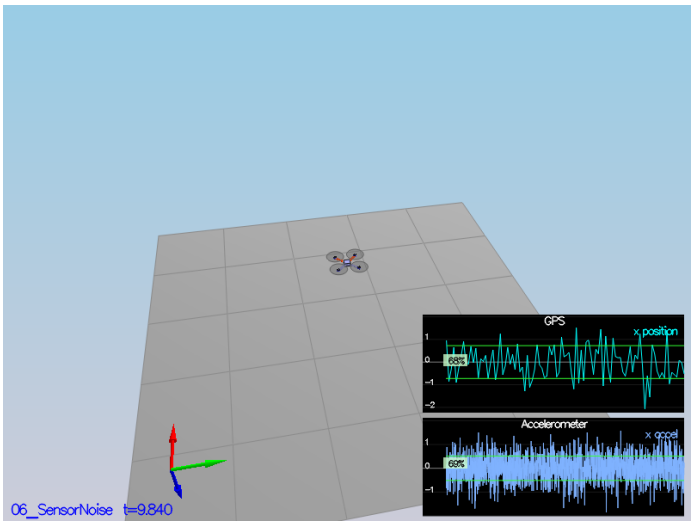
I calculated the standard deviation of `GPSPosXY` and `AccelXY` using Pandas on the simulator logs generated by the `06_SensorNoise` scenario:

```
import pandas as pd

a = pd.read_csv("config/log/Graph1.txt")
b = pd.read_csv("config/log/Graph2.txt")

round(a.iloc(1)[1].std(), 6)
round(b.iloc(1)[1].std(), 6)
```

The calculated standard deviations were:

$GPSPosXY$	0.725163
$AccelXY$	0.512859

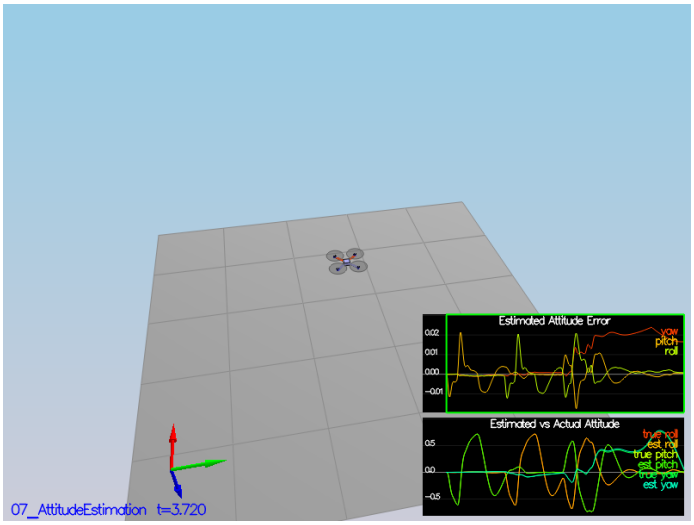
After this, `06_SensorNoise` passed:




```
Simulation #12 (../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
Simulation #13 (../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
Simulation #14 (../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
Simulation #15 (../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
```

1.2 Gyro Attitude Integration

In `UpdateFromIMU()`, I improved the gyro integration using the rotation matrix. I also precompute the sin, cos, and tan values used for the matrix. After this, the drone could maintain attitude:



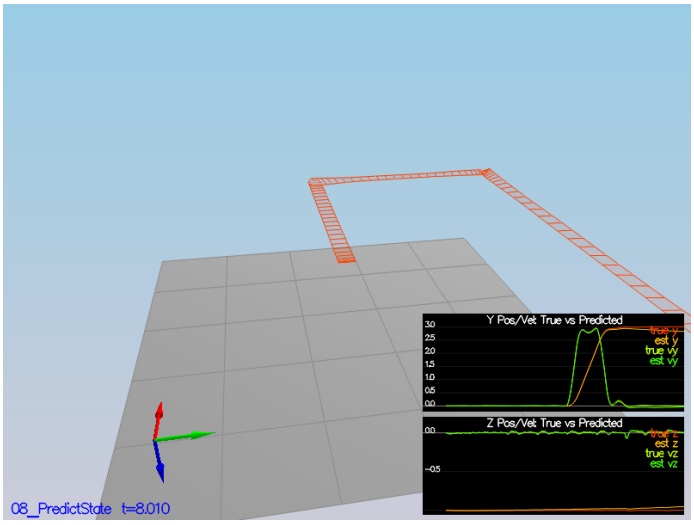

```
Simulation #22 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
Simulation #23 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
Simulation #24 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
Simulation #25 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
Simulation #26 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
```

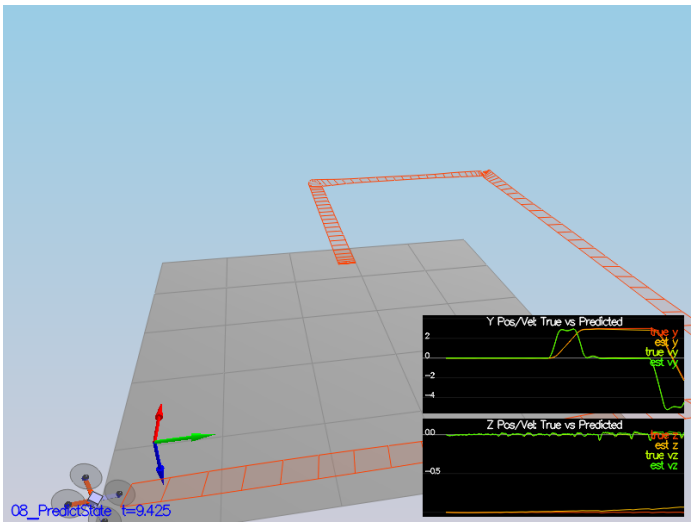
1.3 Prediction Step For The Estimator

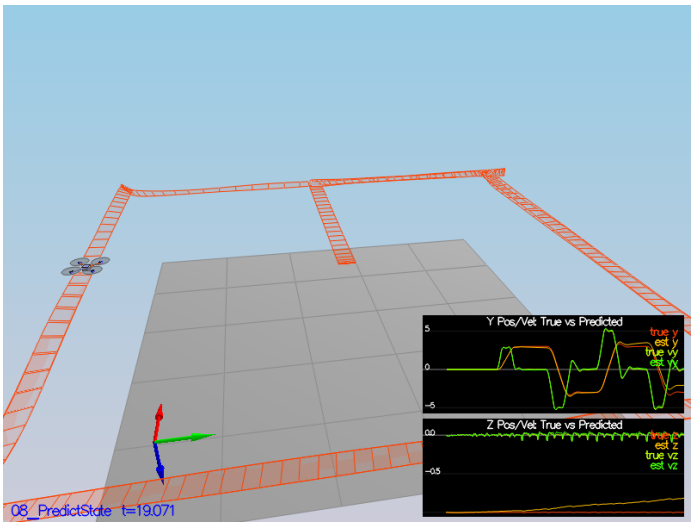
In `PredictState`, I integrate the current states for position and velocity over the time step, transform the prediction to the inertial frame, and then compensate for gravity:

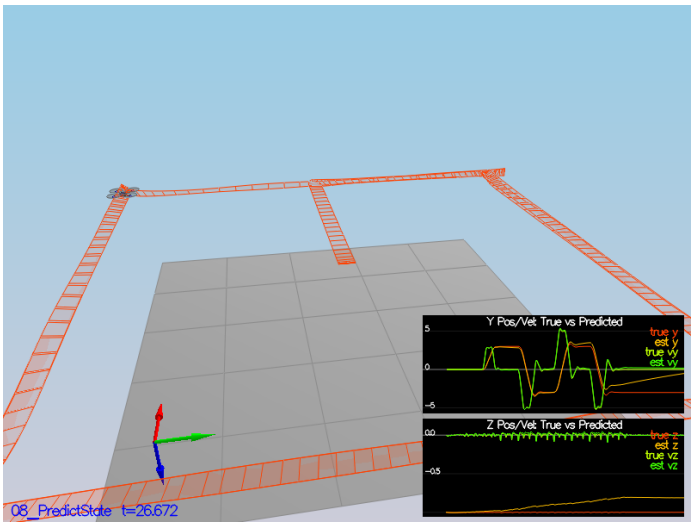
```
predictedState(0) = curState(0) + dt * curState(3);
predictedState(1) = curState(1) + dt * curState(4);
predictedState(2) = curState(2) + dt * curState(5);

// Transform the acceleration to the inertial frame.
V3F a_I = attitude.Rotate_BtoI(accel);

// Update the predicted state with the current acceleration and
// compensate for gravity.
predictedState(3) = curState(3) + dt * a_I.x;
predictedState(4) = curState(4) + dt * a_I.y;
predictedState(5) = curState(5) + dt * a_I.z -
    dt * (float) CONST_GRAVITY;
```

At this point the drone can fly the square in `08_PredictState`:




Y Pos/Vel True vs Predicted
true y
est y
true vy
est vy
Z Pos/Vel True vs Predicted
est z
true vz
est vz
08_PredictState t=9.425


Y Pos/Vel True vs Predicted
true y
est y
true vy
est vy
Z Pos/Vel True vs Predicted
est z
true vz
est vz
08_PredictState t=19.071

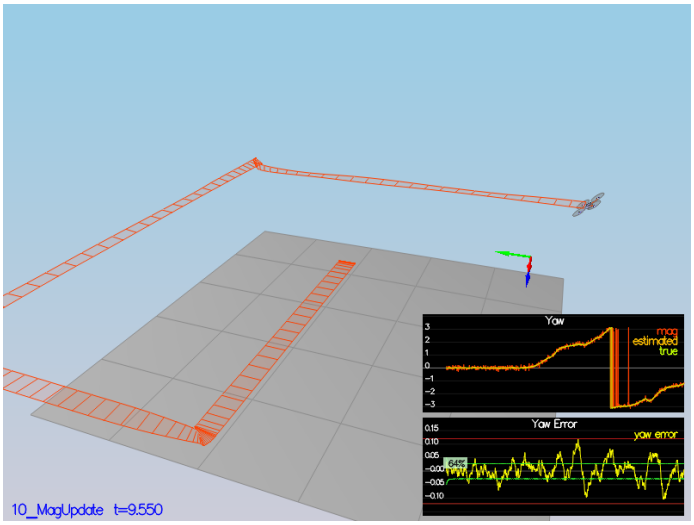

Y Pos/Vel True vs Predicted
true y
est y
true vy
est vy
5
0
−5
Z Pos/Vel True vs Predicted
true z
est z
true vz
est vz
0.0
−0.5
08_PredictState t=26.672

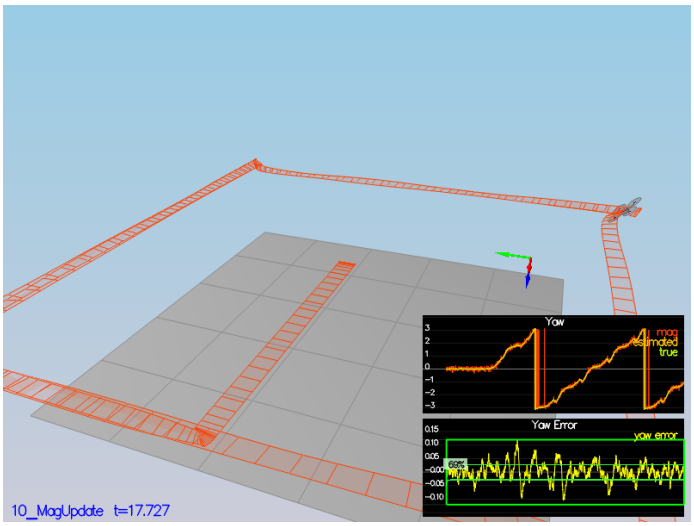
1.4 Magnetometer Update

The magnetometer update is done by integrating the partial derivatives of the magnetometer data over the prior timestep:

```
gPrime(0,3) = dt;
gPrime(1,4) = dt;
gPrime(2,5) = dt;

gPrime(3, 6) = (RbgPrime(0) * accel).sum() * dt;
gPrime(4, 6) = (RbgPrime(1) * accel).sum() * dt;
gPrime(5, 6) = (RbgPrime(2) * accel).sum() * dt;

ekfCov = gPrime * ekfCov * gPrime.transpose() + Q;
```

At this point the drone maintains attitude even with magnetometer noise and passes the `10_MagUpdate` test:



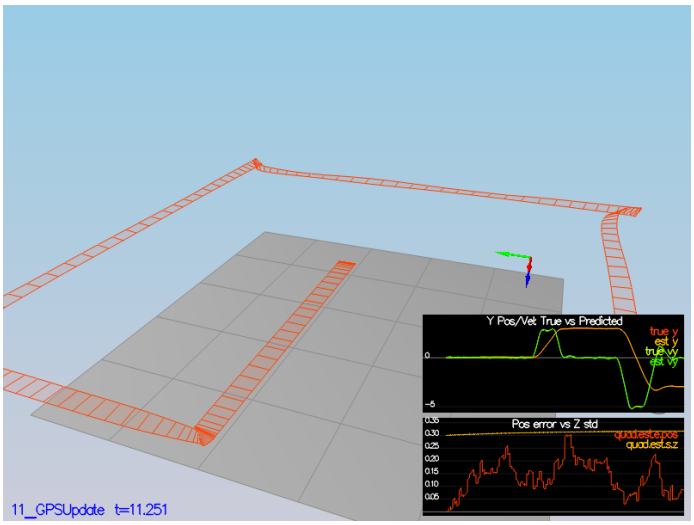

Yaw
mag
estimated
true
Yaw Error
yaw_error
10_MagUpdate t=17.727

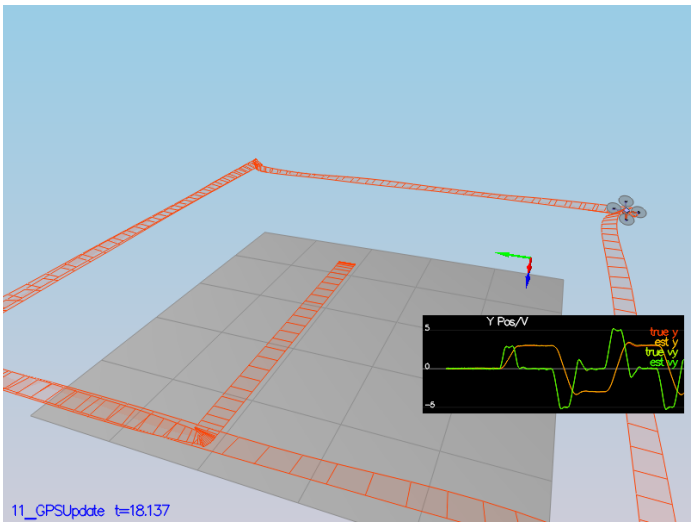
1.5 GPS Update

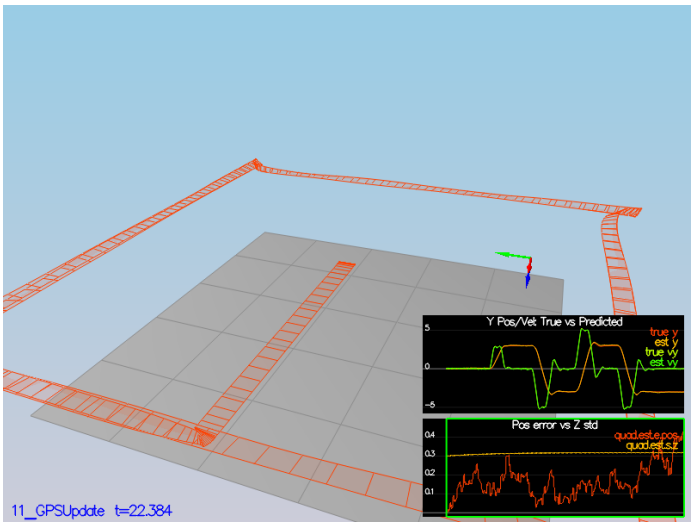
Finally the GPS covariance is updated:

```
zFromX(0) = ekfState(0);
zFromX(1) = ekfState(1);
zFromX(2) = ekfState(2);
zFromX(3) = ekfState(3);
zFromX(4) = ekfState(4);
zFromX(5) = ekfState(5);

for ( int i = 0; i < 6; i++) {
    hPrime(i,i) = 1;
}
```

At this point the drone manages GPS noise in `11_GPSUpdate`:







2 Flight Evaluation

The `QuadPhysicalParams` had too much gain for the sensor noise this exercise introduced. Initially I increased all of the gains to try and increase the precision of the drone, but that ended with several crashes, considerable instability in flight, and lots of consternation.

After switching back to the original parameters and seeing a noticeable improvement over the increases, I instead decreased the gains, and as I did this the flight characteristics improved considerably. I didn't reduce them too much past the point the drone could remain stable.