Project: Building an Estimator

Rubric Points

Here I will consider the rubric points individually and describe how I addressed each point in my implementation.

Writeup / README

1. Provide a Writeup / README that includes all the rubric points and how you addressed each one. You can submit your writeup as markdown or pdf.

You're reading it! Below I describe how I addressed each rubric point and where in my code each point is handled.

Implement Estimator

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

1. I extracted the values from config/log/Graph1.txt (GPS X data) and config/log/Graph2.txt (Accelerometer X data)
2. Stored in MS Excel and used stdev formula ,to find the standard devidation of the given samples

2. Implement a better rate gyro attitude integration scheme in the UpdateFromIMU() function.

```
///////////////////////////// BEGIN STUDENT CODE ///////////////////////////
// SMALL ANGLE GYRO INTEGRATION:
// (replace the code below)
// make sure you comment it out when you add your own code -- otherwise e.g. you might integrate yaw twice

/* float predictedPitch = pitchEst + dtIMU * gyro.y;
float predictedRoll = rollEst + dtIMU * gyro.x;
ekfState(6) = ekfState(6) + dtIMU * gyro.z; // yaw */

Quaternion<float> quat = Quaternion<float>::FromEuler123_RPY(rollEst, pitchEst, ekfState(6));
quat.IntegrateBodyRate(gyro,dtIMU);
float predictedRoll =  quat.Roll();
 float predictedPitch  =  quat.Pitch();
ekfState(6) =  quat.Yaw();

if (ekfState(6) > F_PI) ekfState(6) -= 2.f*F_PI;
if (ekfState(6) < -F_PI) ekfState(6) += 2.f*F_PI;

////////////////////////////// END STUDENT CODE ////////////////////////////
```

3. Implement all of the elements of the prediction step for the estimator.

1. The prediction step includes the state update element (PredictState() function), a correct calculation of the Rgb prime matrix, and a proper update of the state covariance.
2. The acceleration is accounted for as a command in the calculation of gPrime.
3. The covariance update follows the classic EKF update equation.

4. Implement the magnetometer update.

```
/////////////////////////////// BEGIN STUDENT CODE ///////////////////////////
float error = magYaw - ekfState(6);

if (error > F_PI)
  error -= 2.f*F_PI;
if (error < -F_PI)
  error += 2.f*F_PI;

zFromX(0) = magYaw - error;
hPrime(0, 6) = 1;
//////////////////////////////// END STUDENT CODE ////////////////////////////
```

5. Implement the GPS update.

```
/////////////////////////////// BEGIN STUDENT CODE ///////////////////////////
for (int i=0;i<6;i++)
{
      hPrime(i,i) = 1.f;
      zFromX(i) = ekfState(i);
}
//////////////////////////////// END STUDENT CODE ////////////////////////////
```

Flight Evaluation

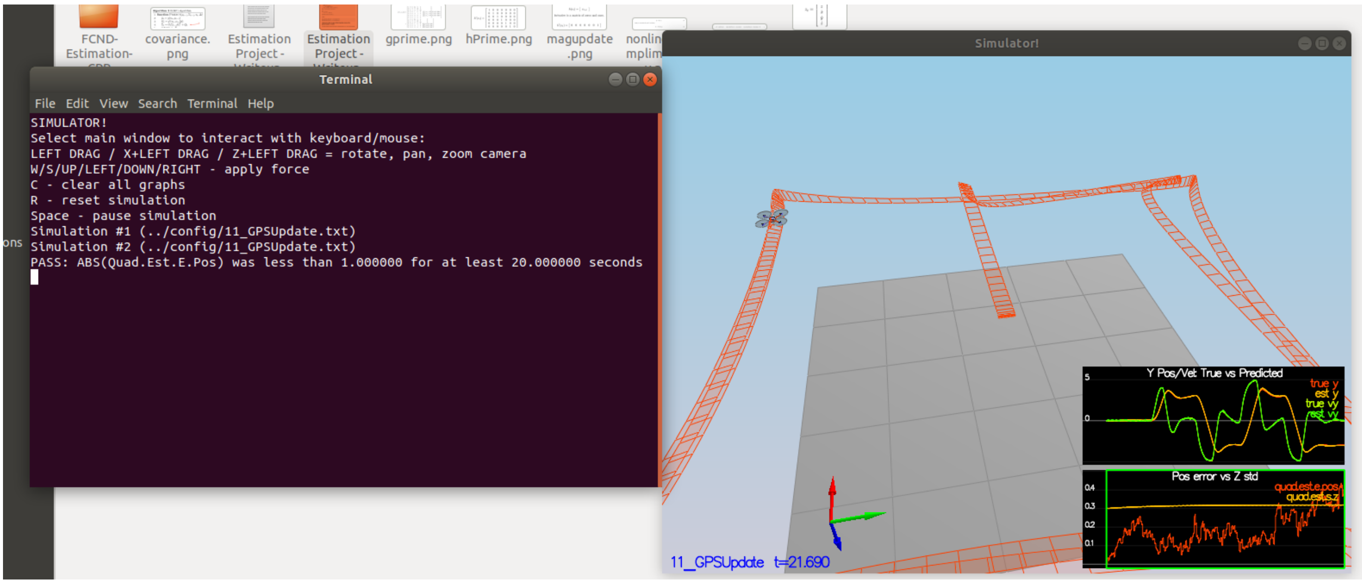
1. Meet the performance criteria of each step.

I have passed the criteria in each step

```
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 71% of the time
PASS: ABS(Quad.IMU.AX-0.000000) was less than MeasuredStdDev_AccelXY for 72% of the time
Simulation #3 (../config/07_AttitudeEstimation.txt)
Simulation #4 (../config/07_AttitudeEstimation.txt)
PASS: ABS(Quad.Est.E.MaxEuler) was less than 0.100000 for at least 3.000000 seconds
Simulation #5 (../config/08_PredictState.txt)
Simulation #6 (../config/08_PredictState.txt)
Simulation #7 (../config/09_PredictCovariance.txt)
Simulation #8 (../config/09_PredictCovariance.txt)
Simulation #9 (../config/09_PredictCovariance.txt)
Simulation #10 (../config/10_MagUpdate.txt)
Simulation #11 (../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 78% of the time
Simulation #12 (../config/11_GPSUpdate.txt)
Simulation #13 (../config/11_GPSUpdate.txt)
PASS: ABS(Quad.Est.E.Pos) was less than 1.000000 for at least 20.000000 seconds
^CPress <RETURN> to close this window...
```

2. De-tune your controller to successfully fly the final desired box trajectory with your estimator and realistic sensors.

The de tuned controller works and passes the Scenario 11



Terminal
File Edit View Search Terminal Help
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
Simulator!
Y Pos/Vel: True vs Predicted
Pos error vs Z std
11_GPSUpdate t=21.690