Welcome to the estimation project. In this project, you will be developing the estimation portion of the controller used in the CPP simulator. By the end of the project, your simulated quad will be flying with your estimator and your custom controller (from the previous project)!

Setup

This project will continue to use the C++ development environment you set up in the Controls C++ project.

1. Clone the repository

```
git clone https://github.com/udacity/FCND-Estimation-CPP.git
```

2. Import the code into your IDE like done in the Controls C++ project
3. You should now be able to compile and run the estimation simulator just as you did in the controls project

Project Structure

For this project, you will be interacting with a few more files than before.

- The EKF is already partially implemented for you in `QuadEstimatorEKF.cpp`
- Parameters for tuning the EKF are in the parameter file `QuadEstimatorEKF.txt`
- When you turn on various sensors (the scenarios configure them, e.g. `Quad.Sensors += SimIMU, SimMag, SimGPS`), additional sensor plots will become available to see what the simulated sensors measure.
- The EKF implementation exposes both the estimated state and a number of additional variables. In particular:
    - `Quad.Est.E.X` is the error in estimated X position from true value. More generally, the variables in `<vehicle>.Est.E.*` are relative errors, though some are combined errors (e.g. MaxEuler).
    - `Quad.Est.S.X` is the estimated standard deviation of the X state (that is, the square root of the appropriate diagonal variable in the covariance matrix). More generally, the variables in `<vehicle>.Est.S.*` are standard deviations calculated from the estimator state covariance matrix.
    - `Quad.Est.D` contains miscellaneous additional debug variables useful in diagnosing the filter. You may or might not find these useful but they were helpful to us in verifying the filter and may give you some ideas if you hit a block.

`config` Directory

In the `config` directory, in addition to finding the configuration files for your controller and your estimator, you will also see configuration files for each of the simulations. For this project, you will be working with simulations 06 through 11 and you may find it insightful to take a look at the configuration for the simulation.

As an example, if we look through the configuration file for scenario 07, we see the following parameters controlling the sensor:

```
# Sensors
Quad.Sensors = SimIMU
# use a perfect IMU
SimIMU.AccelStd = 0,0,0
SimIMU.GyroStd = 0,0,0
```

This configuration tells us that the simulator is only using an IMU and the sensor data will have no noise. You will notice that for each simulator these parameters will change slightly as additional sensors are being used and the noise behavior of the sensors change.

Now that you have everything set up, you'll be able to dive right into the project!

The math that you'll need to implement is explained in the Estimation for Quadrotors document that you worked with in the lessons on Kalman Filters. The **Three Dimensional Quad** section should be most helpful.

Acknowledgment

The C++ simulator and project scenarios were largely designed and built by Fotokite. Big thanks to the Fotokite team (and Sergei Lupashin in particular) for their great work!