Writeup for Controls project in FCND

Suhasini P R

May 6, 2018

1 The Tasks

1. Scenario 1: (Intro) - Get the mass right.
2. Scenario 2: Body rate and Roll pitch control
3. Scenario 3: Position/velocity and yaw angle control
4. Scenario 4: Non-idealities and robustness
5. Scenario 5: Tracking trajectories

2 Rubric Points

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

2.1 Writeup / README

- 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.

2.2 Implemented Controller

- Implemented body rate control in C++.

  I implemented a simple proportional controller for this. Just took the error and muliplied it by kpPQR and also the Moments of Inertia Ixx, Iyy and Izz.

- Implement roll pitch control in C++.

  For this, i first calculated the target matrix element R13 and R23 using accelration and collective thrust values. Then I restricted these angles to

the max tilt angle. I use the following equation to get p command and q command values

$$\begin{bmatrix} p_{cmd} \\ q_{cmd} \end{bmatrix} = \frac{1}{R33} \begin{bmatrix} R21 & -R11 \\ R22 & -R12 \end{bmatrix} \begin{bmatrix} kpBank * (R13t - R13) \\ kpBank * (R23t - R23) \end{bmatrix} \quad (1)$$

- Implement altitude controller in C++.

  I first restrict the target velocity to maxAscent and maxdescent rate. I then use the following equations to get u_bar.

$$posErr = pos_{cmd} - pos \quad (2)$$

$$integratedError = integratedError + (posErr) * dt \quad (3)$$

$$uBar = kpPosZ*posErr+kpVelZ*(velZ_{Cmd}-velZ)+accelZCmd+KiPosZ*integratedError; \quad (4)$$

$$thrust = (g - uBar) * mass/R33; \quad (5)$$

- Implement lateral position control in C++.

  I use a cascaded P controller for this. I first limit the commanded vel to the max speed and then calculate the acceleration in x and y directions using the following equation

$$horizAccel = kpPosXY*(posCmd-pos)+kpVelXY*(velCmd-vel)+accelCmd; \quad (6)$$

  The horizontal acceleration obtained above is limited t maxAcceleration and then returned.

- Implement yaw control in C++.

  I first limit the yaw error to the range -pi to pi and then use a simple P controller to get the yaw rate. The equations are as follows:

$$yawRateCmd = kpYaw * yaw_error; \quad (7)$$

- Implement calculating the motor commands given commanded thrust and moments in C++.

  To get the motor commanded thrusts, we have F_{tot} and moments τ_x, τ_y, τ_z. Using these values F1, F2, F3 and F4 are obtained. Following are the equations.

$$F1 = F_{tot} + \frac{\tau_x}{L} + \frac{\tau_y}{L} - \frac{\tau_z}{kappa} \quad (8)$$

$$F2 = F_{tot} - \frac{\tau_x}{L} + \frac{\tau_y}{L} + \frac{\tau_z}{kappa} \tag{9}$$

$$F3 = F_{tot} + \frac{\tau_x}{L} - \frac{\tau_y}{L} + \frac{\tau_z}{kappa} \tag{10}$$

$$F4 = F_{tot} - \frac{\tau_x}{L} - \frac{\tau_y}{L} - \frac{\tau_z}{kappa} \tag{11}$$

2.3 Flight Evaluation

- Your C++ controller is successfully able to fly the provided test trajectory and visually passes inspection of the scenarios leading up to the test trajectory.

  Although my drone passes all metrics until scenario 3, it goes flying off in the opposite direction in scenario 4. I'm unable to understand why this is happening.