Building a Controller

Writeup

Body Rate Controller

This method is a first-order proportional controller that outputs moments along 3-axis of the vehicle using target and current body rates PQR. The method is implemented in file **QuadControl.cpp** at line 107. The method name is **BodyRateControl**.

It calculates the error between target and current body rates and applies a proportional term kpPQR to it to calculate angular accelerations. Next using the relationship

$M = I \times \alpha$ where M is moment, I is moment of inertia and α is angular acceleration.

Moments along all 3 axis is calculate and returned.

Pictch Roll Controller

This method is a first-order proportional controller that outputs pitch and roll angle rates based on target accelerations along XY axis, collective thrust from altitude controller and current attitude from the vehicle. The method is implemented in file **QuadControl.cpp** at line 144. The method name is **RollPitchControl**.

First acceleration based on collective thrust is calculated, then target elements bx and by for the rotation matrix are calculated. These represent the direction of target thrust along XY axis. After constraining these values to maximum tilt angle, a p-controller is applied on the error between target and current bx by terms to get bx_dot and by_dot. These are then mapped to roll and pitch rates using 2x2 matrix from the 3x3 rotation matrix. The yaw angle rate is set to zero as it is calculated separately.

Altitude Controller

This method is a second-order PID controller that outputs an upward thrust based on drone's target and current positions and velocities, feed-forward acceleration and estimated attitude. The method is implemented in file **QuadControl.cpp** at line 197. The method name is **AltitudeControl**.

It accounts for the roll pitch angles and calculates the thrust based on the z-diection of the thrust vector from the Rotation matrix R. It also applies an integration control that accumulates the error in the model (different weights of different drones).

Lateral Position Controller

This method outputs an acceleration vector for XY axis based on drone's XY position and velocity and a feed-forward acceleration term.It is a PD controller. The method is implemented in file **QuadControl.cpp** at line 236. The method name is **LateralPositionControl**.

Target velocity of the drone is constrained before applying the control plus the resulting target acceleration is also constrained w.r.t. the provided limits.

Yaw Controller

This method is a linear p-controller that outputs a yaw rate based on the current and target heading of the vehicle. The unwrapping of the radian measures was handled using fmodf.

The method is implemented in file **QuadControl.cpp** at line 280. The method name is **YawControl**.

Motor Commands

This method outputs four individual thrusts, one for each propeller based on the collective thrust input by the altitude controller and 3-axis moments from BodyRateController. The problem was formulated as a system of 4 equations which was then converted to its equivalent Matrix form. The derivation was done on paper and resulting equations for four thrusts was implemented in this method. The method is implemented in **QuadControl.cpp** at line 56 and is called **GenerateMotorCommands**.

The implementation files QuadControl.cpp and QuadControlParams.txt are shared along with this write-up for review.