Generator Commands

The commands are generated in function ***GenerateMotorCommands()*** solving the equations:

$$F_1 + F_4 - F_2 - F_3 = \tau_x / l \quad = t_1$$

$$F_1 + F_2 - F_3 - F_4 = \tau_y / l \quad = t_2$$

$$F_1 - F_2 + F_3 - F_4 = -\tau_z / \kappa \quad = t_3$$

$$F_1 + F_2 + F_3 + F_4 = F_t \quad = t_4$$

Where,

all the F_1 to F_4 are the motor's thrust,

$\tau_{(x,y,z)}$ are the moments on each direction,

F_t is the total thrust,

k is the drag/thrust ratio and

l is the drone arm length over square root of two.

Body Rate Control

The body rate control is implemented applying a P controller and the moments of inertia. The parameter ***KpPQR*** has to be tuned to stop the drone from flipping. However, since ***GenerateMotorCommands()*** doesn't command thrust, we have to command some thrust in the altitude control. A good value would be ***thrust = mass * CONST_GRAVITY.***

Roll/Pitch Control

We need to apply a P controller to the elements R_{13} and R_{23} of the rotation matrix from body frame acceleration and world frame acceleration.

$$\dot{b_c^x} = k_p(b_c^x - b_a^x)$$

$$\dot{b_c^y} = k_p(b_c^y - b_a^y)$$

$$b_a^x = R_{13}$$

$$b_a^y = R_{23}$$

In order to calculate roll and pitch rates, we need to apply the following equation:

$$\begin{pmatrix} p_c \\ q_c \end{pmatrix} = \frac{1}{R_{33}} \begin{pmatrix} R_{21} & -R_{11} \\ R_{22} & -R_{12} \end{pmatrix} \times \begin{pmatrix} \dot{b}_c^x \\ \dot{b}_c^y \end{pmatrix}$$

It is important to notice you received thrust and it needs to be inverted and converted to acceleration before applying the equations. After the implementation is done, start tuning ***kpBank*** and ***kpPQR*** until the drone flies more or less stable upward.

Position/Velocity and Yaw Angle Control

There are three methods to implement here:

1. **AltitudeControl**: This is a PD controller to control the acceleration, meaning the thrust needed to control the altitude.

$$\begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ g \end{pmatrix} + R \begin{pmatrix} 0 \\ 0 \\ c \end{pmatrix}$$

$$b^x = R_{13}$$

$$b^y = R_{23}$$

$$b^z = R_{33}$$

$$\bar{u}_1 = \ddot{z} = cb^z + g$$

$$c = (\bar{u}_1 - g)/b^z$$

$$\bar{u}_1 = k_{p-z}(z_t - z_a) + k_{d-z}(\dot{z}_t - \dot{z}_a) + \ddot{z}_t$$

2. **LateralPositionControl**: This is another PID controller to control acceleration on *x* & ***y***.
3. **YawControl**: This is a simpler case because it is P controller. It is better to optimize the yaw to be between [-pi, pi].

Once all the code is implemented, put ***kpPosXY,kpVelXY, kpPosZ and kpVelZ*** to zero and start tuning from the altitude controller to the yaw controller.