FCND Control Project

Son Le

May 11, 2018

1 Mathematical Derivations

We assume the drone used in this assignment is modeled as in Fig. 1. If any of the following assumptions is invalid, the math must be redone:

- The front left rotor R_1 spins and rear right rotor R_4 spin clockwise.
- The front right rotor R_2 and rear left rotor R_3 spins counter-clockwise.
- The origin of the body frame is at the vehicle's mass center.
- The x axis of the body frame points from the origin to the front.
- The y axis of the body frame points from the origin to the right.
- The z axis of the body frame points from the origin to under the vehicle.

Each rotor generates a thrust F_i which and a torque τ_i. The thrust points up and is orthogonal to the body plane. It can be used used to control elevation, roll and pitch. The torque is a reaction caused by the motor motion. So if the motor spins clockwise, the body will be turned counter-clockwise and vice versa.

Control of elevation The elevation is control by the collective force acting on the vehicle, $\vec{F}_{total} = \vec{F}_1 + \vec{F}_2 + \vec{F}_3 + \vec{F}_4$. Because thrust vectors are orthogonal to the xy plane, we will only project this equation onto the z axis, which yields

$$F_{total} = F_1 + F_2 + F_3 + F_4. \tag{1}$$

Note that $\vec{F}_1$, $\vec{F}_2$, $\vec{F}_3$, and $\vec{F}_4$ all point up, there values are *negative* in our chosen body frame.

Control of roll The moments caused by $\vec{F}_1$, $\vec{F}_2$, $\vec{F}_3$, and $\vec{F}_4$ are illustrated in Fig. 2. The total moment is

$$\tau = \vec{\ell}_1 \times \vec{F}_1 + \vec{\ell}_2 \times \vec{F}_2 + \vec{\ell}_3 \times \vec{F}_3 + \vec{\ell}_4 \times \vec{F}_4. \tag{2}$$

Figure 1: Drone's configuration

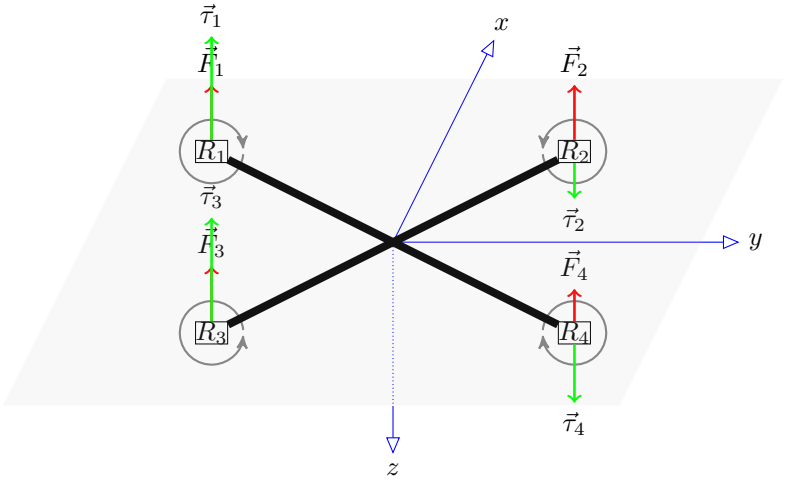
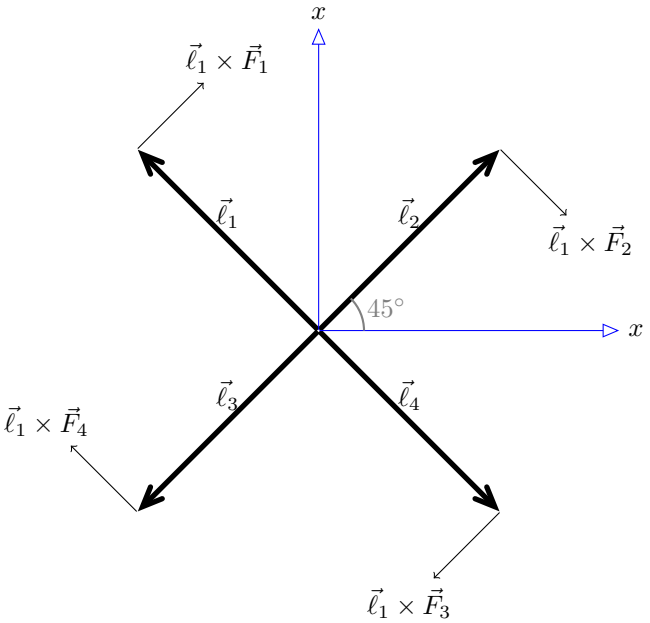


Figure 2: Moments caused by thrusts



In this project, we also assume that the drone is symmetric, which means $\left\|\vec{\ell}_1\right\| = \left\|\vec{\ell}_2\right\| = \left\|\vec{\ell}_3\right\| = \left\|\vec{\ell}_4\right\| =: \ell$. By projecting Eq. 2 onto the x axis, we have

$$\begin{aligned}\tau_x &= -\ell F_1 \cos 45^\circ + \ell F_2 \cos 45^\circ - \ell F_3 \cos 45^\circ + \ell F_4 \cos 45^\circ \\ &= \frac{\ell}{\sqrt{2}}(-F_1 + F_2 - F_3 + F_4) \end{aligned} \tag{3}$$

Control of pitch By projecting Eq. 2 onto the x axis, we have

$$\begin{aligned}\tau_x &= -\ell F_1 \sin 45^\circ - \ell F_2 \sin 45^\circ + \ell F_3 \sin 45^\circ + \ell F_4 \sin 45^\circ \\ &= \frac{\ell}{\sqrt{2}}(-F_1 - F_2 + F_3 + F_4) \end{aligned} \tag{4}$$

Control of yaw Yawing is caused by motor torques $\vec{\tau}_1$, $\vec{\tau}_2$, $\vec{\tau}_3$, and $\vec{\tau}_4$. Because these torque vectors are orthogonal to the xy plane (Fig. 1), we will only project this equation onto the z axis, which yields

$$\tau_z = \kappa(F_1 - F_2 - F_3 + F_4). \tag{5}$$

Combining Eqs. 1, 3, 4 and 5, we have

$$\begin{bmatrix} F_{total} \\ \sqrt{2}\tau_x/\ell \\ \sqrt{2}\tau_y/\ell \\ \tau_z/\kappa \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} \tag{6}$$

$$\Leftrightarrow \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 \end{bmatrix}^{-1} \begin{bmatrix} F_{total} \\ \sqrt{2}\tau_x/\ell \\ \sqrt{2}\tau_y/\ell \\ \tau_z/\kappa \end{bmatrix} \tag{7}$$

$$\Leftrightarrow \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} = \frac{1}{4}\begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} F_{total} \\ \sqrt{2}\tau_x/\ell \\ \sqrt{2}\tau_y/\ell \\ \tau_z/\kappa \end{bmatrix} \tag{8}$$

2 Implementation

In this section I only highlight tricky parts of the code.

2.1 Computing Motor Thrusts

The inputs of `GenerateMotorCommands` are the desired thrust and moments. There is no complication with moments because their values are compatible with the chosen body frame, and can there by be used directly in Eq. 8. However, the thrust input is always positive, and it points upward, meaning in our body

frame, it has to be negated before fed into Eq. 8. Eq. 8 gives us values for F_1, F_2, F_3, and F_4 in the chosen body frame with the z axis pointing downward, i.e. their values are always negative. We therefore will need to negate those values before sending them as rotor commands:

$$\begin{bmatrix} \text{thrust}_1 \\ \text{thrust}_2 \\ \text{thrust}_3 \\ \text{thrust}_4 \end{bmatrix} = -\frac{1}{4} \begin{bmatrix} 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} -\text{thrust} \\ \sqrt{2}\tau_x/\ell \\ \sqrt{2}\tau_y/\ell \\ \tau_z/\kappa \end{bmatrix}$$

2.2 Altitude Control

Altitude control is tricky because the output is the collective thrust which is a positive value, but is negative in the chosen body frame. Let the output thrust be $c > 0$. In the body frame we have the following equation:

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ -c/m \end{bmatrix} \tag{9}$$

$$\Rightarrow \ddot{z} = g - \frac{r_{3,3}c}{m} \tag{10}$$

$$\Leftrightarrow c = r_{3,3}\frac{g - \ddot{z}}{m} \tag{11}$$

2.3 Parameter Tuning

- If we want the control output to converge faster, we need to use a large value for k_p. However a large k_p can result two issues: overshooting and faster oscillation.
- If a large value for k_p overshoots, we need to use a large value for k_d to compensate. The issue with a large k_d is that the path might not be smooth. The symptom is that the drone appears to slow down while it is traveling and then accelerate.
- When adjusting k_p and k_d of a particular controller does not dampen the output, try tuning other controller. For example, I tried many values for `kPosXY` and `kVelXY`, but they did not dampen the lateral controller output. It appears to struggle to control when the drone is in the horizontal position. I tried tilting the drone more by increasing `kpBank` which worked nicely.