Project Statement Phase 2

RBE 502 - Robot Control

November 24, 2024

1 Problem Statement

The objective of the project is to design several controllers and a path planning system for a quadrotor that travels in a restricted airspace. Given an specific destination, the quadrotor should reach and hover at that destination without collision with any obstacles. Figure 1 illustrates a sketch of the concept.

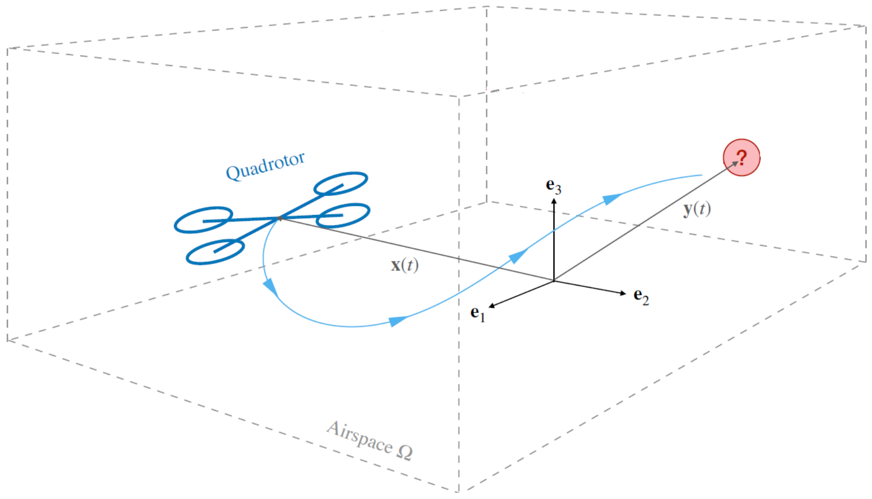


Figure 1: Conceptual illustration of the problem.

2 Problem Specifications

2.1 Coordinate Systems and Reference frames

The coordinate systems and free body diagram for the quadrotor are shown in Figure 2. The inertial frame, $\mathcal{A}$, is defined by the triad $\mathbf{a}_1, \mathbf{a}_2$, and $\mathbf{a}_3$ with $\mathbf{a}_3$ pointing upward. The body frame, $\mathcal{B}$, is attached to the center of mass of the quadrotor with $\mathbf{b}_1$ coinciding with the preferred forward direction and $\mathbf{b}_3$ perpendicular to the plane of the rotors pointing vertically up during perfect hover (see Fig. 1). These vectors are parallel to the principal axes. The center of mass is C. Rotor 1 is a distance L away along $\mathbf{b}, 2$ is L away along $\mathbf{b}_2$, while 3 and 4 are similarly L away along the negative $\mathbf{b}_1$ and $\mathbf{b}_2$ respectively.

2.2 Basic Inertial Properties

A free-body diagram of the quadrotor is depicted in Figure 2 and the corresponding parameters are presented in Table 1. In order to simplify the derivations, we assume

$$\tau_i = \sigma u_i, \quad i \in \{1, 2, 3, 4\}, \tag{1}$$

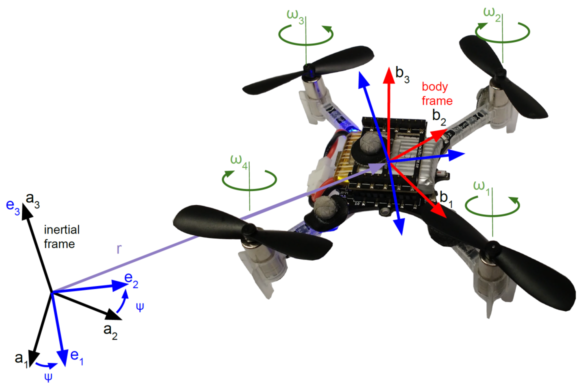
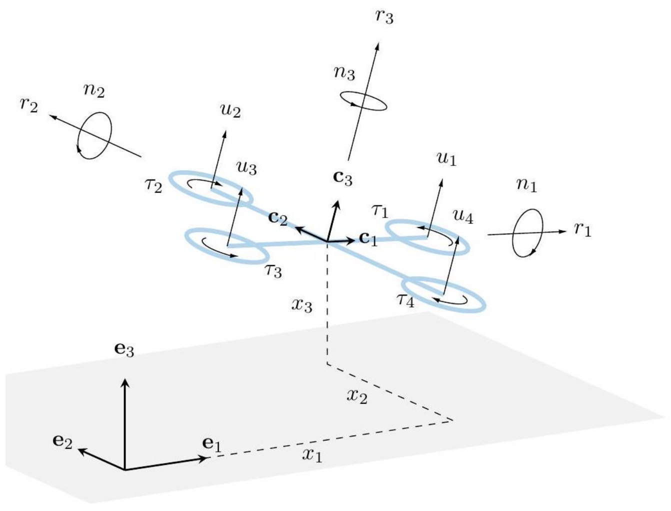


Figure 2: The CrazyFlie 2.0 robot. Note the reflective motion capture markers attached. A pair of motors spins counter clockwise while the other pair spins clockwise, such that when all propellers spin at the same speed, the net torque in the yaw direction is zero. The pitches on the corresponding propellers are reversed so that the thrust is always pointing in the $\mathbf{b}_3$ direction for all propellers. Shown also is the transformation from the inertial frame to the body-fixed frame. First a rotation by ψ around the $\mathbf{a_3}$ axis (leading to coordinate system $\mathbf{e_1}, \mathbf{e_2}, \mathbf{e_3}$) is performed, followed by a translation $\mathbf{r}$ to the center of mass C of the robot. Subsequent rotations by ϕ and θ generate the final body-fixed coordinate system $\mathcal{B}$, where the axes $\mathbf{b}_1$ and $\mathbf{b}_2$ are aligned with the arms, and $\mathbf{b}_3$ is perpendicular to them.

where $u_i \in [0, \mu]$N, for a given $\mu > 0$, is the thrust generated by the propeller i. As depicted, the coordinate system $C = \{\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3\}$ is attached to the body of the quadrotor and the fixed reference frame $E = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ serves as the inertial frame, with its origin coinciding with the nest, see Figure 3. To simplify the model, it is assumed that all the rotors are located on the same plane as the center of mass (that coincides with the origin of C) with the same distance $l > 0$ from the center of mass. The external force and moment vectors $\mathbf{r} = r_1\mathbf{c}_1 + r_2\mathbf{c}_2 + r_3\mathbf{c}_3$ and $\mathbf{n} = n_1\mathbf{c}_1 + n_2\mathbf{c}_2 + n_3\mathbf{c}_3$ are directly applied to the center of mass. Moreover, it is assumed that the mass moment of inertia tensor is

$$I = \begin{bmatrix} I_{11} & 0 & 0 \\ 0 & I_{22} & 0 \\ 0 & 0 & I_{33} \end{bmatrix} \tag{2}$$

where I_{11}, I_{22} and I_{33} denote the mass moment of inertia of the quadrotor about $\mathbf{c}_1, \mathbf{c}_2$ and $\mathbf{c}_3$ axes, respectively. All these information are included in the simulator provided and should not be changed.



Parameter	Value	Units	Description
l	0.046	m	Distance from the center of mass to the center of each rotor
m	0.03	kg	Total mass of the quadrotor
I_{11}	1.43	$\mathrm{kg} \cdot \mathrm{m}^2$	Mass moment of inertia about c_1 axis
I_{22}	1.43	$\mathrm{kg} \cdot \mathrm{m}^2$	Mass moment of inertia about c_2 axis
I_{33}	2.89	$\mathrm{kg} \cdot \mathrm{m}^2$	Mass moment of inertia about c_3 axis
g	9.81	$\mathrm{m/s}^2$	The gravitational acceleration
σ	0.01	m	The proportionality constant relating u_i to τ_i
μ	0.73575	N	Maximum thrust of each rotor

Table 1: The quadrotor parameters

2.3 Motor Model

Each rotor has an angular speed ω_i and produces a vertical force F_i according to

$$F_i = k_F \omega_i^2 \tag{3}$$

Experimentation with a fixed rotor at steady-state shows that $k_F \approx 6.11 \times 10^{-8}$ N/rpm^2. The rotors also produce a moment according to

$$M_i = k_M \omega_i^2. \tag{4}$$

The constant, k_M, is determined to be about 1.5×10^{-9}Nm/rpm^2 by matching the performance of the simulation to the real system.

Data obtain from system identification experiments suggest that the rotor speed is related to the commanded speed by a first-order differential equation

$$\dot{\omega}_i = k_m \left(\omega_i^{\mathrm{des}} - \omega_i \right). \tag{5}$$

2.4 Rigid Body Dynamics

Kinematics We will use $Z - X - Y$ Euler angles to model the rotation of the quadrotor in the world frame. To get from $\mathcal{A}$ to $\mathcal{B}$, we first rotate about $\mathbf{a}_3$ through the the yaw angle, ψ, to get the triad $\mathbf{e}_i$. A rotation about the $\mathbf{e}_1$ through the roll angle, ϕ gets us to the triad $\mathbf{f}_i$ (not shown in the figure). A third pitch rotation about $\mathbf{f}_2$ through θ results in the body-fixed triad $\mathbf{b}_i$. You will need the rotation matrix for transforming components of vectors in $\mathcal{B}$ to components of vectors in $\mathcal{A}$:

$${}^{\mathcal{A}}[R]_{\mathcal{B}} = \begin{bmatrix} c\psi c\theta - s\phi s\psi s\theta & -c\phi s\psi & c\psi s\theta + c\theta s\phi s\psi \\ c\theta s\psi + c\psi s\phi s\theta & c\phi c\psi & s\psi s\theta - c\psi c\theta s\phi \\ -c\phi s\theta & s\phi & c\phi c\theta \end{bmatrix} \tag{6}$$

We will denote the components of angular velocity of the robot in the body frame by p, q, and r :

$${}^{\mathcal{A}}\omega_{\mathcal{B}} = p\mathbf{b}_1 + q\mathbf{b}_2 + r\mathbf{b}_3. \tag{7}$$

These values are related to the derivatives of the roll, pitch, and yaw angles according to

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} c\theta & 0 & -c\phi s\theta \\ 0 & 1 & s\phi \\ s\theta & 0 & c\phi c\theta \end{bmatrix} \begin{bmatrix} \dot{\phi} \\ \dot{\theta} \\ \dot{\psi} \end{bmatrix} \tag{8}$$

Newton's Equations of Motion Let $\mathbf{r}$ denote the position vector of C in $\mathcal{A}$. The forces on the system are gravity, in the $-\mathbf{a}_3$ direction, and the forces from each of the rotors, F_i, in the $\mathbf{b}_3$ direction. The equations governing the acceleration of the center of mass are

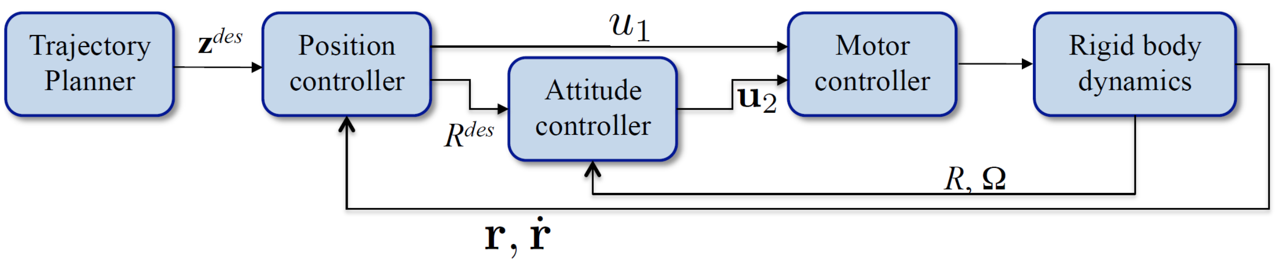


Figure 3: Block Diagram for a Quadrotor Control.

$$m\ddot{\mathbf{r}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ F_1 + F_2 + F_3 + F_4 \end{bmatrix} \tag{9}$$

We will define our first input u_1 to be

$$u_1 = \Sigma_{i=1}^{4} F_i \tag{10}$$

Euler's Equations of Motion In addition to forces, each rotor produces a moment perpendicular to the plane of rotation of the blade, M_i. Rotors 1 and 3 rotate in the $-\mathbf{b}_3$ direction while 2 and 4 rotate in the $+\mathbf{b}_3$ direction. Since the moment produced on the quadrotor is opposite to the direction of rotation of the blades, M_1 and M_3 act in the $\mathbf{b}_3$ direction while M_2 and M_4 act in the $-\mathbf{b}_3$ direction. L is the distance from the axis of rotation of the rotors to the center of mass of the quadrotor.

The angular acceleration determined by the Euler equations is

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} L(F_2 - F_4) \\ L(F_3 - F_1) \\ M_1 - M_2 + M_3 - M_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{11}$$

We can rewrite this as:

$$I \begin{bmatrix} \dot{p} \\ \dot{q} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix} - \begin{bmatrix} p \\ q \\ r \end{bmatrix} \times I \begin{bmatrix} p \\ q \\ r \end{bmatrix} \tag{12}$$

where $\gamma = \frac{k_M}{k_F}$ is the relationship between lift and drag given by Equations (1-2). Accordingly, we will define our second set of inputs to be the vector of moments $\mathbf{u}_2$ given by:

$$\mathbf{u}_2 = \begin{bmatrix} 0 & L & 0 & -L \\ -L & 0 & L & 0 \\ \gamma & -\gamma & \gamma & -\gamma \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{bmatrix}$$

3 Robot Controllers

3.1 The Nominal State

Our controllers are derived by linearizing the equations of motion and motor models at an operating point that corresponds to the nominal hover state, $\mathbf{r} = \mathbf{r}_0, \theta = \phi = 0, \psi = \psi_0, \dot{\mathbf{r}} = 0$, and $\dot{\phi} = \dot{\theta} = \dot{\psi} = 0$, where the roll and pitch angles are small ($c\phi \approx 1, c\theta \approx 1, s\phi \approx \phi$, and $s\theta \approx \theta$). At this state the lift from the propellers is given by:

$$F_{i,0} = \frac{mg}{4}$$

The nominal values for the inputs at hover are $u_{1,0} = mg, \mathbf{u}_{2,0} = \mathbf{0}$. Linearizing eq. (9), we get:

$$\begin{aligned}\ddot{r}_1 &= g\left(\Delta\theta\cos\psi_0 + \Delta\phi\sin\psi_0\right)\\ \ddot{r}_2 &= g\left(\Delta\theta\sin\psi_0 - \Delta\phi\cos\psi_0\right)\\ \ddot{r}_3 &= \frac{1}{m}u_1 - g\end{aligned} \tag{13}$$

Linearizing eq. (12), we get:

$$\begin{bmatrix}\dot{p}\\ \dot{q}\\ \dot{r}\end{bmatrix} = I^{-1}\begin{bmatrix}0 & L & 0 & -L\\ -L & 0 & L & 0\\ \gamma & -\gamma & \gamma & -\gamma\end{bmatrix}\begin{bmatrix}F_1\\ F_2\\ F_3\\ F_4\end{bmatrix}$$

If we assume the rotor craft is symmetric so $I_{xx} = I_{yy}$, we get:

$$\begin{aligned}\dot{p} &= \frac{u_{2,x}}{I_{xx}} = \frac{L}{I_{xx}}\left(F_2 - F_4\right)\\ \dot{q} &= \frac{u_{2,y}}{I_{yy}} = \frac{L}{I_{yy}}\left(F_3 - F_1\right)\\ \dot{r} &= \frac{u_{2,z}}{I_{zz}} = \frac{\gamma}{I_{zz}}\left(F_1 - F_2 + F_3 - F_4\right)\end{aligned} \tag{14}$$

In other words, the equations of motion are decoupled in terms of angular accelerations. Each component of angular acceleration depends only on the appropriate component of $\mathbf{u}_2$.

3.2 Position and Attitude Control

The control problem is to determine the four inputs, $\{u_1, \mathbf{u}_2\}$ required to hover or to follow a desired trajectory, $\mathbf{z}^{\text{des}}$. As shown in Figure 2, we will use errors in the robot's position to drive a position controller from eq. (13) which directly determines u_1. The model in eq. (13) also allows us to derive a desired orientation. The attitude controller for this orientation is derived from the model in eq. (14).

The transformation of the inputs into $\{u_1, \mathbf{u}_2\}$ is straightforward and is described in [1]. The inner attitude control loop uses onboard accelerometers and gyros to control the roll, pitch, and yaw and runs at approximately 500 Hz , while the outer position control loop uses estimates of position and velocity of the center of mass to control the trajectory in three dimensions at $100 - 200$ Hz.

3.2.1 Attitude Control

We now present a proportional plus derivative (PD) attitude controller to track a trajectory in $SO(3)$ specified in terms of a desired roll, pitch and yaw angle. Since our development of the controller will be based on linearized equations of motion, the attitude must be close to the nominal hover state where the roll and pitch angles are small.

Near the nominal hover state the proportional plus derivative control laws take the form:

$$\mathbf{u}_2 = I\begin{bmatrix}k_{p,\phi}\left(\phi^{\text{des}} - \phi\right) + k_{d,\phi}\left(p^{\text{des}} - p\right)\\ k_{p,\theta}\left(\theta^{\text{des}} - \theta\right) + k_{d,\theta}\left(q^{\text{des}} - q\right)\\ k_{p,\psi}\left(\psi^{\text{des}} - \psi\right) + k_{d,\psi}\left(r^{\text{des}} - r\right)\end{bmatrix} \tag{15}$$

3.2.2 Position Control

In this subsection, we will present two position control methods that use the roll and pitch angles as inputs to drive the position of the quadrotor. In both methods, the position control algorithm will determine the desired roll and pitch angles, θ^{des} and ϕ^{des} , which can be used to compute the desired speeds from eq. (14).

The first, a hover controller, is used for station-keeping or maintaining the position at a desired position vector, $\mathbf{r}_0$. The second tracks a specified trajectory, $\mathbf{r}_T(t)$, in three dimensions. In both cases, the desired yaw angle, is specified independently. It can either be a constant, ψ_0 or a time varying quantity, $\psi_T(t)$. We will assume that the desired trajectory:

$$\mathbf{z}^{des} = \begin{bmatrix}\mathbf{r}_T(t)\\ \psi_T(t)\end{bmatrix}$$

will be provided by a trajectory planner as an input to specify the trajectory of the position vector and the yaw angle we are trying to track.

Hover Controller For hovering, $\mathbf{r}_T(t) = \mathbf{r}_0$ and $\psi_T(t) = \psi_0$. The command accelerations, $\ddot{r}_i^{\text{des}}$, are calculated from a proportional plus derivative (PD) controller. Define the position error in terms of components using the standard reference triad $\mathbf{a}_i$ by:

$$e_i = (r_{i,T} - r_i)$$

In order to guarantee that this error goes exponentially to zero, we require

$$\left(\ddot{r}_{i,T} - \ddot{r}_i^{\text{des}}\right) + k_{d,i}\left(\dot{r}_{i,T} - \dot{r}_i\right) + k_{p,i}\left(r_{i,T} - r_i\right) = 0$$

where $\dot{r}_{i,T} = \ddot{r}_{i,T} = 0$ for hover.

From eq. (13) we can obtain the relationship between the desired accelerations and roll and pitch angles. Given that $\Delta\theta = \theta - \theta_0 = \theta$ and $\Delta\phi = \phi - \phi_0 = \phi$, we can write

$$\begin{aligned} \ddot{r}_1^{\text{des}} &= g\left(\theta^{\text{des}}\cos\psi_T + \phi^{\text{des}}\sin\psi_T\right) \\ \ddot{r}_2^{\text{des}} &= g\left(\theta^{\text{des}}\sin\psi_T - \phi^{\text{des}}\cos\psi_T\right) \\ \ddot{r}_3^{\text{des}} &= \frac{1}{m}u_1 - g \end{aligned} \tag{16}$$

For hover, the last equation yields:

$$u_1 = mg + m\ddot{r}_3^{\text{des}} = mg - m\left(k_{d,3}\dot{r}_3 + k_{p,3}\left(r_3 - r_{3,0}\right)\right)$$

The other two equations can be used to compute the desired roll and pitch angles for the attitude controller:

$$\begin{aligned} \phi^{\text{des}} &= \frac{1}{g}\left(\ddot{r}_1^{\text{des}}\sin\psi_T - \ddot{r}_2^{\text{des}}\cos\psi_T\right) \\ \theta^{\text{des}} &= \frac{1}{g}\left(\ddot{r}_1^{\text{des}}\cos\psi_T + \ddot{r}_2^{\text{des}}\sin\psi_T\right) \end{aligned} \tag{17}$$

The desired roll and pitch velocities are taken to be zero.

$$\begin{aligned} p^{\text{des}} &= 0 \\ q^{\text{des}} &= 0 \end{aligned} \tag{18}$$

Since the yaw, $\psi_T(t)$ is prescribed independently by the trajectory generator, we get:

$$\begin{aligned} \psi^{\text{des}} &= \psi_T(t) \\ r^{\text{des}} &= \dot{\psi}_T(t) \end{aligned} \tag{19}$$

These equations provide the setpoints for the attitude controller in eq.(15). Note that the attitude controller must run an order of magnitude faster than the position control loop in order for this to work. In practice as discussed in , the position controller runs at 100 Hz , while the inner attitude control loop runs at 500 Hz .

3D Trajectory Control The 3-D Trajectory Controller is used to follow three-dimensional trajectories with modest accelerations so the nearhover assumptions hold. To derive this controller we follow the same steps as in eq.(16) except that $\dot{r}_{i,T}$ and $\ddot{r}_{i,T}$ are no longer zero but are obtained from the specification of the trajectory. If the near-hover assumption holds and the dynamics are linear with no saturation on the inputs, a controller that generates the desired acceleration $\ddot{r}_i^{\text{des}}$ according to eq.(16) is guaranteed to drive the error exponentially to zero. However, it is possible that the commanded trajectory has twists and turns that are too hard to follow perfectly because of errors in the model or limitations on the input thrusts. We suggest a modification that allows the robot to follow the path even if it may not be able to follow the time-parameterized trajectory.

Let $\mathbf{r}_T$ denote the closest point on the desired trajectory to the the current position $\mathbf{r}$, and let the desired velocity and acceleration obtained by differentiating the specified trajectory be given by $\dot{\mathbf{r}}_T$ and $\ddot{\mathbf{r}}_T$ respectively. Let the unit tangent vector of the trajectory (unit vector along $\dot{\mathbf{r}}_T$) be $\hat{\mathbf{t}}$. The unit normal to the trajectory, $\hat{\mathbf{n}}$, is derived by differentiating the tangent vector with respect to time or arc length, and finally the unit binormal vector is the cross product $\hat{\mathbf{b}} = \hat{\mathbf{t}} \times \hat{\mathbf{n}}$. We define the position and velocity errors according to the following equations:

$$\mathbf{e}_p = ((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{n}})\, \hat{\mathbf{n}} + \left((\mathbf{r}_T - \mathbf{r}) \cdot \hat{\mathbf{b}}\right) \hat{\mathbf{b}}$$

and

$$\mathbf{e}_v = \dot{\mathbf{r}}_T - \dot{\mathbf{r}}$$

Note that here we ignore position error in the tangential direction and only considering position error in the plane that is normal to the curve at the closest point. Once again we calculate the commanded acceleration, $\ddot{r}_i^{\text{des}}$, from the PD feedback as shown in eq.(16). In vector notation, this is:

$$\left(\ddot{\mathbf{r}}_T - \ddot{\mathbf{r}}^{\text{des}}\right) + \mathbf{k}_d \mathbf{e}_v + \mathbf{k}_p \mathbf{e}_p = 0$$

Finally we use (17 - 19) to compute the desired roll and pitch angles. Again we refer the readers to [1] for more details including experimental results obtained from these controllers.

4 Optimal Control

4.1 Linear Quadratic Regulator (LQR)

LQR is a design technique from optimal control theory to design controllers for linear systems to minimize a quadratic cost function subject to the dynamic constraint $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$, resulting in a linear state-feedback controller.

Theorem 1 *If the linear system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ is controllable and if $\mathbf{Q}$ is a positive semi-definite matrix and $\mathbf{R}$ is positive definite matrix, then there exists a solution to the optimization problem*

$$\begin{gathered} \min \sum_{i=0}^{N-1} \left(\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i + \mathbf{u}_i^T \mathbf{R} \mathbf{u}_i\right) + \mathbf{x}_N^T \mathbf{Q}_N \mathbf{x}_N \\ \textit{Subject to } \mathbf{x}_{i+1} = \mathbf{A}_i \mathbf{x}_i + \mathbf{B}_i \mathbf{u}_i \end{gathered} \tag{20}$$

and the solution is given by

- $\mathbf{u}^*(i) = -\mathbf{K}_i \mathbf{x}_i$, *where* $\mathbf{K}_i = \left(\mathbf{B}_i^T \mathbf{P}_{i+1} \mathbf{B}_i + \mathbf{R}_i\right)^{-1} \mathbf{B}_i^T \mathbf{P}_{i+1} \mathbf{A}_i$,
- *and* $\mathbf{P}_i = \mathbf{Q}_i + \mathbf{A}_i^T \mathbf{P}_{i+1} \mathbf{A}_i - \mathbf{A}_i^T \mathbf{P}_{i+1} \mathbf{B}_i \mathbf{K}_i$, *for* $\mathrm{N}-1$ *to* 0 *do the backward recursion*
- *and* $\mathbf{P}_N = \mathbf{Q}_N$

The solution of the infinite-horizon ($N \to \infty$) optimization problem,

$$\begin{gathered} \min \sum_{i=0}^{\infty} \left(\mathbf{x}_i^T \mathbf{Q} \mathbf{x}_i + \mathbf{u}_i^T \mathbf{R} \mathbf{u}_i\right) \\ \textit{Subject to } \mathbf{x}_{i+1} = \mathbf{A} \mathbf{x}_i + \mathbf{B} \mathbf{u}_i \end{gathered} \tag{21}$$

also exists and is given by

- $\mathbf{u}^* = -\mathbf{K}\mathbf{x}$, *where* $\mathbf{K} = (\mathbf{B}^T \mathbf{P} \mathbf{B} + \mathbf{R})^{-1} \mathbf{B}^T \mathbf{P} \mathbf{A}$,
- *and* $\mathbf{P}$ *is the solution of the Algebraic Ricatti Equation (ARE):* $\mathbf{Q} + \mathbf{A}^T \mathbf{P} \mathbf{A} - \mathbf{A}^T \mathbf{P} \mathbf{B} (\mathbf{B^T} \mathbf{P} \mathbf{B} + \mathbf{R})^{-1} \mathbf{B^T} \mathbf{P} \mathbf{A} = \mathbf{P}$.

- Matlab Command: $[KS] = \text{lqr}(A, B, Q, R)$.

Remark 1 *There is a trade-off between speed of response and the magnitude of control signals. Penalizing the controls more heavily than the states (R large in comparison to Q), tends to result in slower response, while penalizing the states more heavily than the controls (Q large in comparison to R) tends to result in faster response but at the expense of larger control signals.*

Remark 2 *A few choices of Q and R for an example system* $\dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u$ *are below:*

- $\mathbf{Q} = 0.1\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{R} = 1$. *(This would result in slow convergence but with small control signals.)*
- $\mathbf{Q} = 100\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \mathbf{R} = 1$. *(This results in fast convergence but at the expense large control signals)*
- $\mathbf{Q} = \begin{bmatrix} \frac{1}{1^2} & 0 \\ 0 & \frac{1}{0.2^2} \end{bmatrix}, \mathbf{R} = \frac{1}{1^2}$. *(This requests* $\max|x_1| \leq 1, \max|x_2| \leq 0.2$ *and* $\max|u| \leq 1$*, which is respected by the lq controller)*

Theorem 2 ***Trajectory Tracking:*** *If the linear system $\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u}$ is controllable, and is required to track a desired trajectory $\mathbf{x}_{des}$, and if $\mathbf{Q}$ is a positive semi-definite matrix and $\mathbf{R}$ is positive definite matrix, then there exists a solution to the optimization problem*

$$\begin{aligned} \min \frac{1}{2}\mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N + \mathbf{q}_N^T\mathbf{x}_N + \sum_{i=0}^{N-1} \frac{1}{2}\mathbf{x}_i^T\mathbf{Q}_i\mathbf{x}_i + \mathbf{q}_i^T\mathbf{x}_i + \frac{1}{2}\mathbf{u}_i\mathbf{R}_i\mathbf{u}_i \\ \textit{Subject to } \mathbf{x}_{i+1} = \mathbf{A}_i\mathbf{x}_i + \mathbf{B}_i\mathbf{u}_i \end{aligned} \tag{22}$$

and the solution is given by:

- $\mathbf{u}^* = -\mathbf{K}_i\mathbf{x} - \mathbf{k}_i$*, where* $\mathbf{K}_i = \left(\mathbf{R}_i + \mathbf{B}_i^T\mathbf{P}_{i+1}\mathbf{B}_i\right)^{-1}\mathbf{B}_i^T\mathbf{P}_{i+1}\mathbf{A}_i$
- $\mathbf{q}_i = -\mathbf{Q}_i\mathbf{x}_{des,i}$
- $\mathbf{P}_i = \mathbf{Q}_i + \mathbf{A}_i^T\mathbf{P}_{i+1}\mathbf{A}_i - \mathbf{A}_i^T\mathbf{P}_{i+1}\mathbf{B}_i\mathbf{K}_i$
- $\mathbf{k}_i = \left(\mathbf{R}_i + \mathbf{B}_i^T\mathbf{P}_{i+1}\mathbf{B}_i\right)^{-1}\mathbf{B}_i^T\mathbf{p}_{i+1}$
- $\mathbf{p}_i = \mathbf{q}_i + \mathbf{A}_i^T\mathbf{p}_{i+1} - \mathbf{A}_i^T\mathbf{P}_{i+1}\mathbf{B}_i\mathbf{k}_i$ *for* $\mathrm{N}-1$ *to 0 do the backward recursion.*
- *and* $\mathbf{P}_N = \mathbf{Q}_N, \quad \mathbf{p}_N = \mathbf{q}_N$

4.2 MPC Design

Model Predictive Control (MPC) is a robust and flexible control strategy that can optimize system performance while respecting system constraints. Unlike traditional controllers, MPC explicitly accounts for constraints on inputs and states by solving an optimization problem at each time step.

Theorem 3 *If the nonlinear system $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$ is linearized around an operating point, the dynamics can be represented as:*

$$\dot{\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u},$$

where $\mathbf{x}$ is the state vector and $\mathbf{u}$ is the control input. For such a system, MPC can be formulated as an optimization problem:

$$\begin{aligned} \min_{\mathbf{u}_0,\mathbf{u}_1,\ldots,\mathbf{u}_{N-1}} \sum_{i=0}^{N-1} \left(\mathbf{x}_i^T\mathbf{Q}\mathbf{x}_i + \mathbf{u}_i^T\mathbf{R}\mathbf{u}_i\right) + \mathbf{x}_N^T\mathbf{Q}_N\mathbf{x}_N, \\ \textit{Subject to:} \quad & \mathbf{x}_{i+1} = \mathbf{A}\mathbf{x}_i + \mathbf{B}\mathbf{u}_i, i = 0, \cdots, N-1 \\ & \mathbf{x}_0 = \bar{\mathbf{x}}_0 \\ & \mathbf{x}_{\min} \leq \mathbf{x}_i \leq \mathbf{x}_{\max}, \\ & \mathbf{u}_{\min} \leq \mathbf{u}_i \leq \mathbf{u}_{\max}. \end{aligned} \tag{23}$$

Here, $\mathbf{Q}$ and $\mathbf{R}$ are positive semi-definite and positive definite weighting matrices, respectively, and $\mathbf{Q}_N$ is the terminal cost matrix. Constraints on the states and inputs are enforced using $\mathbf{x}_{\min}, \mathbf{x}_{\max}, \mathbf{u}_{\min}$, and $\mathbf{u}_{\max}$.

We can modify the MPC formulation into a quadratic programming problem. First we let

$$\tilde{\mathbf{x}} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}, \quad \tilde{\mathbf{u}} = \begin{bmatrix} \mathbf{u}_0 \\ \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}, \quad \bar{\mathbf{x}} = \begin{bmatrix} \tilde{\mathbf{x}} \\ \tilde{\mathbf{u}} \end{bmatrix} \tag{24}$$

Then we can rewrite the cost function in eq.(23) as following:

$$\sum_{i=0}^{N} \mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i = \tilde{\mathbf{x}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{x}} + \mathbf{x}_0^T \mathbf{Q}_0 \mathbf{x}_0 = \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix}^T \begin{bmatrix} \mathbf{Q} & & & \\ & \ddots & & \\ & & \mathbf{Q} & \\ & & & \mathbf{Q}_N \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} + \bar{\mathbf{x}}_0^T \mathbf{Q} \bar{\mathbf{x}}_0 \tag{25}$$

$$\sum_{i=0}^{N-1} \mathbf{u}_i^T \mathbf{R} \mathbf{u}_i = \tilde{\mathbf{u}}^T \tilde{\mathbf{R}} \tilde{\mathbf{u}} = \begin{bmatrix} \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix}^T \begin{bmatrix} \mathbf{R} & & & \\ & \mathbf{R} & & \\ & & \ddots & \\ & & & \mathbf{R} \end{bmatrix} \begin{bmatrix} \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix} \tag{26}$$

$$\sum_{i=0}^{N} \mathbf{x}_i^T \mathbf{Q}_i \mathbf{x}_i + \sum_{i=0}^{N-1} \mathbf{u}_i^T \mathbf{R} \mathbf{u}_i = \tilde{\mathbf{x}}^T \tilde{\mathbf{Q}} \tilde{\mathbf{x}} + \bar{\mathbf{x}}_0^T \mathbf{Q} \bar{\mathbf{x}}_0 + \tilde{\mathbf{u}}^T \tilde{\mathbf{R}} \tilde{\mathbf{u}} = \bar{\mathbf{x}}^T \underbrace{\begin{bmatrix} \tilde{\mathbf{Q}} & \\ & \tilde{\mathbf{R}} \end{bmatrix}}_{\bar{\mathbf{Q}}} \bar{\mathbf{x}} + \bar{\mathbf{x}}_0^T \mathbf{Q} \bar{\mathbf{x}}_0 \tag{27}$$

Since the $\bar{\mathbf{x}}_0$ is the measured initial states, we can ignore the cost $\bar{\mathbf{x}}_0^T \mathbf{Q} \bar{\mathbf{x}}_0$ as it's constant. Therefore, in the quadratic programming, we only consider the cost function $\bar{\mathbf{x}}^T \bar{\mathbf{Q}} \bar{\mathbf{x}}$. Further, we can rewrite the constraints as following:

$$\underbrace{\begin{bmatrix} \mathbf{I} & 0 & 0 & \cdots & 0 & -\mathbf{B}_0 & 0 & 0 & \cdots & 0 \\ -\mathbf{A}_1 & \mathbf{I} & 0 & \cdots & 0 & 0 & -\mathbf{B}_1 & 0 & \cdots & 0 \\ 0 & -\mathbf{A}_2 & \mathbf{I} & \cdots & 0 & 0 & 0 & -\mathbf{B}_2 & \cdots & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & -\mathbf{A}_{N-1} & \mathbf{I} & 0 & 0 & 0 & 0 & -\mathbf{B}_{N-1} \end{bmatrix}}_{\bar{\mathbf{A}}} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \\ \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} \mathbf{A}\mathbf{x}_0 \\ 0 \\ 0 \\ \vdots \end{bmatrix}}_{\bar{\mathbf{b}}} \tag{28}$$

$$\mathbf{x}_{\min} \le \mathbf{x}_i \le \mathbf{x}_{\max}, \quad \mathbf{u}_{\min} \le \mathbf{u}_i \le \mathbf{u}_{\max} \Rightarrow \underbrace{\begin{bmatrix} -\mathbf{I} & 0 & 0 & \cdots & 0 \\ \mathbf{I} & 0 & 0 & \cdots & 0 \\ 0 & -\mathbf{I} & 0 & \cdots & 0 \\ 0 & \mathbf{I} & 0 & \cdots & 0 \\ 0 & 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \cdots & -\mathbf{I} \\ 0 & 0 & 0 & \cdots & \mathbf{I} \end{bmatrix}}_{\bar{\mathbf{H}}} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \\ \mathbf{u}_0 \\ \vdots \\ \mathbf{u}_{N-1} \end{bmatrix} \le \underbrace{\begin{bmatrix} \mathbf{x}_{\min} \\ \mathbf{x}_{\max} \\ \vdots \\ \mathbf{x}_{\min} \\ \mathbf{x}_{\max} \\ \mathbf{u}_{\min} \\ \mathbf{u}_{\max} \\ \vdots \\ \mathbf{u}_{\min} \\ \mathbf{u}_{\max} \end{bmatrix}}_{\bar{\mathbf{h}}_{limit}} \tag{29}$$

Therefor the final formulation of the MPC in quadratic programming can be expressed as

$$\begin{aligned} \min_{\bar{\mathbf{x}}} \quad & \bar{\mathbf{x}}^T \bar{\mathbf{Q}} \bar{\mathbf{x}}, \\ \text{Subject to:} \quad & \bar{\mathbf{A}} \bar{\mathbf{x}} = \bar{\mathbf{b}}, \\ & \bar{\mathbf{H}} \bar{\mathbf{x}} \le \bar{\mathbf{h}}_{limit}, \end{aligned} \tag{30}$$

- Matlab Command: $x = quadprog(H, f, A, b, Aeq, beq)$ for

$$\min_{x} \frac{1}{2} x^T H x + f^T x \text{ s.t.} \begin{cases} A \cdot x \le b, \\ Aeq \cdot x = beq. \end{cases} \tag{31}$$

link: https://www.mathworks.com/help/optim/ug/quadprog.html

Theorem 4 *To track a trajectory using MPC with nonlinear dynamics model $\mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i)$, we can formulate the MPC problem as following:*

$$\begin{aligned} &\min_{u_0,\cdots,u_{N-1}} \sum_{i=0}^{N-1} (\mathbf{x}_i - \mathbf{x}^*)^T \mathbf{Q} (\mathbf{x}_i - \mathbf{x}^*) + (\mathbf{u}_i - \mathbf{u}^*)^T \mathbf{R} (\mathbf{u}_i - \mathbf{u}^*) + (\mathbf{x}_N - \mathbf{x}^*)^T \mathbf{Q}_N (\mathbf{x}_N - \mathbf{x}^*) \\ &\textit{subject to } \mathbf{x}_{i+1} = f(\mathbf{x}_i, \mathbf{u}_i), i = 0, \cdots, N-1 \\ &\qquad\qquad \mathbf{x}_0 = \bar{\mathbf{x}}_0 \\ &\qquad\qquad \mathbf{x}_{\min} \le \mathbf{x}_i \le \mathbf{x}_{\max}, \quad \mathbf{u}_{\min} \le \mathbf{u}_i \le \mathbf{u}_{\max}. \end{aligned} \tag{32}$$

4.2.1 Linearization Around Nominal State

$$\begin{aligned} \mathbf{x}_{i+1} &\approx \mathbf{x}_i + f(\mathbf{x}_i, \mathbf{u}_i)\Delta t = \bar{f}(\mathbf{x}_i, \mathbf{u}_i) \\ &= \bar{f}(\mathbf{x}^*, \mathbf{u}^*) + \left.\frac{\partial \bar{f}}{\partial \mathbf{x}_i}\right|_{\mathbf{x}_i=\mathbf{x}^*,\mathbf{u}_i=\mathbf{u}^*} (\mathbf{x}_i - \mathbf{x}^*) + \left.\frac{\partial \bar{f}}{\partial \mathbf{u}_i}\right|_{\mathbf{x}_i=\mathbf{x}^*,\mathbf{u}_i=\mathbf{u}^*} (\mathbf{u}_i - \mathbf{u}^*) \end{aligned} \tag{33}$$

By letting

$$\bar{\mathbf{x}}_{i+1} = \mathbf{x}_{i+1} - \bar{f}(\mathbf{x}^*, \mathbf{u}^*), \quad \bar{\mathbf{x}}_i = \mathbf{x} - \mathbf{x}^*, \quad \bar{\mathbf{u}}_i = \mathbf{u} - \mathbf{u}^*, \quad \mathbf{A}_i = \left.\frac{\partial \bar{f}}{\partial \mathbf{x}_i}\right|_{\mathbf{x}_i=\mathbf{x}^*,\mathbf{u}_i=\mathbf{u}^*}, \quad \mathbf{B}_i = \left.\frac{\partial \bar{f}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*,\mathbf{u}=\mathbf{u}^*} \tag{34}$$

Hence eq. (33) turns into

$$\bar{\mathbf{x}}_{i+1} = \mathbf{A}_i \bar{\mathbf{x}}_i + \mathbf{B}_i \bar{\mathbf{u}}_i. \tag{35}$$

And the MPC formulation in Theorem 4 can be expressed as

$$\begin{aligned} &\min_{u_0,\cdots,u_{N-1}} \sum_{i=0}^{N-1} \bar{\mathbf{x}}_i^T \mathbf{Q} \bar{\mathbf{x}}_i + \bar{\mathbf{u}}_i^T \mathbf{R} \bar{\mathbf{u}}_i + \bar{\mathbf{x}}_N^T \mathbf{Q}_N \bar{\mathbf{x}}_N \\ &\text{subject to } \bar{\mathbf{x}}_{i+1} = \mathbf{A}_i \bar{\mathbf{x}}_i + \mathbf{B}_i \bar{\mathbf{u}}_i, i = 0, \cdots, N-1 \\ &\qquad\qquad \mathbf{x}_0 = \bar{\mathbf{x}}_0 \\ &\qquad\qquad \mathbf{x}_{\min} \le \mathbf{x}_i \le \mathbf{x}_{\max}, \quad \mathbf{u}_{\min} \le \mathbf{u}_i \le \mathbf{u}_{\max}. \end{aligned} \tag{36}$$

5 Project Work

5.1 Tasks

You will simulate the quadrotor dynamics and control using the matlab simulator posted on the meam620 wiki. The simulator relies on the numerical solver ode 45, details about this solver can be easily found online. You don't need to be an expert in numerical methods, but it would be beneficial if you know the basics of how ode solvers work. Your tasks include:

5.1.1 Phase 2: Trajectory Following with PD/PID Controller. And Trajectory Following with LQR Controller or MPC (Due December 13th, 2024, 11:59 PM)

In this phase, you will integrate the trajectory generator and controller to simulate the quadrotor flying in the space tracking a trajectory. Combine with PD controller you designed in previous step. In addition you are expected to implement a LQR or MPC controller for the quadrotor, combine with trajectory generation system designed at previous step, see if the results are better compared to using PID/PD controller.

You are given 2 trajectory generators:

1. circle.m
   A helix in the xy plane of radius 1 m centered about the point $(0.5, 0, 0)$ starting at the point $(1, 0, 0)$. The z coordinate should start at 0 and end at 0.5 . The helix should go counter-clockwise.

2. diamond.m
   A"diamond helix" with corners at $(0, 0, 0), (0, \sqrt{2}, \sqrt{2}), (0, 0, 2\sqrt{2})$, and $(0, -\sqrt{2}, \sqrt{2})$ when projected into the yz plane, and an x coordinate starting at 0 and ending at 1 . The quadrotor should start at $(0, 0, 0)$ and end at $(1, 0, 0)$.

You can switch between this 2 different trajectory generators by modifying the trajhandle in the test_trajectory.m file

6 Submission

The students are expected to submit the following files:

1. lqr_controller.m or mpc_controller.m;
2. final report in pdf form, reporting the tracking results of the quadrotor.

References

[1] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The grasp multiple micro-uav testbed," *IEEE Robotics and Automation Magazine*, vol. 17, no. 3, pp. 56–65, 2010.