AE353 Homework #10:
Control Design — 2D Quadrotor

(due at the beginning of class on Wednesday, May 6)

Goal

This week, you will design a control system that makes a 2D quadrotor (a "bi-rotor") deliver ten packages as fast as possible. You will implement your control system by modifying the script:

`hw10_ControlLoopTemplate.m`

You will test your control system with the script:

`hw10_Simulation.m`

Instructions on exactly how to do these two things will be posted to piazza.

Model

The dynamics of the robot are:

$$\begin{aligned} m\ddot{q}_1 &= -\left(f_R + f_L\right)\sin q_3 \\ m\ddot{q}_2 &= \left(f_R + f_L\right)\cos q_3 - mg \\ J\ddot{q}_3 &= w\left(f_R - f_L\right) \end{aligned} \tag{1}$$

where

- m is the mass;
- J is the moment of inertia about the center of mass;
- w is the spar length;
- (q_1, q_2) is the position of the center of mass of the robot;
- q_3 is the angle of the robot from horizontal (in radians);
- f_R and f_L are the forces supplied by the right and left rotor, respectively.

To "hover" means to achieve zero motion at $q_3 = 0$, which—by examination of (2)—can only be achieved when $f_R + f_L = mg$. Near hover, the dynamics can be linearly approximated by

$$\begin{aligned} m\ddot{q}_1 &= -mgq_3 \\ m\ddot{q}_2 &= \left(f_R + f_L\right) - mg \\ J\ddot{q}_3 &= w\left(f_R - f_L\right) \end{aligned} \tag{2}$$

You now have a linear set of ODEs, which you can proceed to put in state space form and use as the basis for control design. As you will see in `hw10_ControlLoopTemplate.m`, you have access to a noisy measurement of position (q_1, q_2) and a not-so-noisy measurement of angular velocity $\dot{q}_3$. It is of course possible to define a set of outputs that correspond to these measurements, to put these outputs in state space form, and to use them as the basis for an observer. *Note that rotors cannot produce negative thrust.* So f_R and f_L will be bounded below by 0 (and bounded above by `simdata.params.fmax`). This fact may be important when implementing your observer.

Three Tasks

Your control system has three tasks:

- To pick up packages by hovering near the point `pFrom`.
- To deliver packages by hovering near the point `pTo`.

  Note that `pTo` is only non-empty when the robot has a package to deliver. Note also that packages have mass. When the robot has a package—i.e., when `pTo` is non-empty—both its mass and its moment of inertia increase from m and J to $m + m_{\text{package}}$ and $J + J_{\text{package}}$. Your control design will need to take this fact into account (e.g., by switching between a "no package" design and a "yes package" design, or by using integral action).
- To recharge the robot's battery, when necessary, by hovering near the point `pBattery`.

  Note that the level of charge γ in the battery, when it is not recharging, satisfies

  $$\dot{\gamma} = k_{\text{battery}} \left(f_R^2 + f_L^2 \right)$$

  In other words, the charge decreases faster when the rotors are producing more thrust. You'll want to make sure you don't let your robot run out of charge.

Note that all three of these tasks require you to hover near a point. A good strategy might be to design one control system that can hover near any given point, and then just switch the point.

What to Turn In

You may work, if you like, with one partner. The two of you should submit the following:

- A single MATLAB script to replace `hw10_ControlLoopTemplate.m`, with your final control system. You must call this script `hw10_NAME1_NAME2.m`, where "NAME1" and "NAME2" are replaced with the first five letters of your first name (in capitals) followed by the first letter of your last name (in capitals) and—if you are working in a group—your teammate. Details on how to submit your code will be posted to piazza.
- A brief description of your design process. If you use state space methods, this process should at least include the following:
  - Derivation of a state space model of the system (starting from what is described above).
  - Analysis of controllability and observability.
  - Design of controller and observer, either by eigenvalue placement or by optimality.
  - Simulation results, and their use to refine or validate your control system.

  If you use classical methods, the design process should at least include the following:
  - A block diagram model of the system.
  - Analysis of time response.
  - Analysis of frequency response.
  - Simulation results, and their use to refine or validate your control system.

  Please be very concise. A formal report is not required.

As discussed in class, we will have a contest in class on the due date. Details of this contest—and of opportunities for extra credit—will be posted online.