AE483 Homework #3: Quad-Rotor Controllers and Observers

(due at the beginning of class on Monday, October 24)

1. *Minimizing quadratic functions.* A symmetric matrix $A \in \mathbb{R}^{n \times n}$ is called *positive definite* if

$$x^T A x > 0$$

for all non-zero $x \in \mathbb{R}^n$. We indicate that A is positive definite by writing $A > 0$. It is a fact that a symmetric matrix is positive definite if and only if all of its eigenvalues are positive (note that a symmetric matrix has only real eigenvalues).

(a) Consider the quadratic function

$$\psi(x) = x^T A x$$

where $x \in \mathbb{R}^n$ and $A > 0$. Does $\psi(x)$ have a minimum value? If so, what is that value, what is a minimizer x^* that achieves the minimum value $\psi(x^*)$, and is this minimizer unique? Justify your answer.

(b) Consider the quadratic function

$$\phi(x) = \frac{1}{2}(x - r)^T L (x - r) + s,$$

where $x \in \mathbb{R}^n$ and $L > 0$. Does $\phi(x)$ have a minimum value? If so, what is that value, what is a minimizer x^* that achieves the minimum value $\phi(x^*)$, and is this minimizer unique? Justify your answer.

(c) Consider the quadratic function

$$\gamma(x) = \frac{1}{2} x^T F x + g^T x + h$$

where $x \in \mathbb{R}^n$ and $F > 0$. Does $\gamma(x)$ have a minimum value? If so, what is that value, what is a minimizer x^* that achieves the minimum value $\gamma(x^*)$, and is this minimizer unique? Justify your answer by finding the matrices L, r, and s that would put $\gamma(x)$ in the same form as $\phi(x)$.

(d) Consider the same function $\gamma(x)$ as in the previous problem. If h changes, does the minimum value change? Does the minimizer change?

(e) Consider again the same function $\gamma(x)$. Suppose

$$g = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \qquad \text{and} \qquad h = 5$$

and consider the following four choices of F:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \qquad \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix} \qquad \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}$$

When F is positive definite, find the minimizer and the minimum value. When F is *not* positive definite, try to find a minimizer and a minimum value anyway, and—should they exist—say if the minimizer is unique.

2. *The finite-horizon, discrete-time, linear quadratic regulator.* Solve the following problem:

$$\begin{aligned} \underset{u_d(0),\dots,u_d(n-1)}{\text{minimize}} \quad & \frac{1}{2}x_d(n)^T Q_{\text{final}} x_d(n) + \frac{1}{2}\sum_{i=0}^{n-1}\left(x_d(i)^T Q_d x_d(i) + u_d(i)^T R_d u_d(i)\right) \\ \text{subject to} \quad & x_d(i+1) = A_d x_d(i) + B_d u_d(i) \text{ for all } i \in \{0,\dots,n-1\} \\ & x_d(0) = x_{d,0}. \end{aligned}$$

In this problem, Q_{final}, Q_d, R_d, A_d, B_d, and $x_{d,0}$ are parameters. You may assume that $Q_{\text{final}} > 0$, $Q_d > 0$, and $R_d > 0$. Countless people have derived the solution to this problem before you, and many versions of the solution appear in books, papers, course notes, and other online resources. So the point here is not "the answer," but rather to understand and clearly express—in your own way—the process of getting there. I suggest the following steps:

(a) Define a value function $v(i, x(i))$ with the form

$$v(i, x_d(i)) = \underset{\text{a list of variables}}{\text{minimum}} \left\{\text{a cost : a constraint}\right\}.$$

Say in words what the value function means.

(b) Verify that the value function satisfies an equation of the form

$$v(i, x_d(i)) = \underset{\text{a list of variables}}{\text{minimum}} \left\{\text{a cost that involves } v(i+1, \text{ something})\right\}.$$

Say in words what this equation means.

(c) Verify that

$$v(n, x_d(n)) = \frac{1}{2}x_d(n)^T P(n) x_d(n)$$

for some $P(n) > 0$. In other words, express $P(n)$ in terms of other parameters and verify that it is positive definite.

(d) Assume that

$$v(i+1, x_d(i+1)) = \frac{1}{2}x_d(i+1)^T P(i+1) x_d(i+1)$$

for some $P(i+1) > 0$. By evaluating the equation you wrote in part (b), show that

$$v(i, x_d(i)) = \frac{1}{2}x_d(i)^T P(i) x_d(i)$$

for some $P(i)$. In other words, express $P(i)$ in terms of $P(i+1)$ and other parameters. You may simply assume for now that $P(i) > 0$. We will return to this assumption later. Please also show that the choice of input $u_d(i)$ that minimizes the cost—the minimizer in the equation from part (b)—is unique and has the form

$$u_d(i) = -K_d(i) x_d(i)$$

for some $K_d(i)$. In other words, express $K_d(i)$ in terms of other parameters.

Do you realize that you have now solved the problem? The proof is by induction. Part (c) provides the "base case." Part (d) provides the "inductive step."

3. *The infinite-horizon, discrete-time, linear quadratic regulator.* In the "control design example" video (https://piazza.com/class/is654xmwlgxd9?cid=83), you saw that a discrete-time model of a propellor—linearized about a nominal spin rate—had the form

$$x_d(i+1) = A_d x_d(i) + B_d u_d(i)$$

where

$$A_d = 1 - 2c\bar{\sigma}\Delta t \qquad\qquad B_d = \Delta t.$$

Suppose $\Delta t = 0.001$, $\bar{\sigma} = 1000$, and $c = 0.005$.

(a) Suppose you want to apply a finite-horizon, discrete-time, linear quadratic regulator to this system, with parameters $Q_{\text{final}} = Q_d = R_d = 1$ and with time horizon $n = 1000$ (i.e., 1 second for $\Delta t = 0.001$). Use your results from Problem 2 to compute $K_d(i)$, and plot it versus $i\Delta t$ in solid black, for all $i \in \{0, 1, \ldots, n\}$.

(b) You should have found that $K_d(i)$ converged to something as i got small. What value did it converge to?

(c) Apply the function `dlqr` in MATLAB to compute a controller for this system:

```
Kd = dlqr(Ad,Bd,Qd,Rd)
```

Compare your result to part (b). What do you notice?

(d) For the system considered in this problem, it is possible to compute the value to which $K_d(i)$ converges by hand. In particular, consider the expression you found for $P(i)$ in Problem 2, part (d). What it means for $P(i)$ to have converged is that it is no longer changing. In particular, suppose that $P(i) = P(i+1) = P_{\text{ss}}$ (where "ss" is meant to imply "steady state"). Solve the equation in 2(d) for P_{ss}. You should find that there are two solutions, but that only one of these is positive definite. Use the positive value of P_{ss} to find a corresponding value of K_d. Plot this value on the same figure as in part (a), as a horizontal dashed red line. Also compare it to what you found in parts (b) and (c). What do you notice?

(e) What you have just found in three different ways is the solution to the "infinite-horizon" version of what you saw in Problem 2, i.e., the solution as $n \to \infty$. In particular, you have found that when n gets large, $K_d(i)$ stops changing. Let's call the constant value simply K_d. In other words, drop the dependence in our notation on the time step i. Simulate the response of the system to the (infinite-horizon) optimal control $u_d(i) = -K_d x_d(i)$ starting from the initial condition $x_d(0) = 50$. Plot $x_d(i)$ for $i = \{0, 1, \ldots, n\}$ and $u_d(i)$ for $i = \{0, 1, \ldots, n-1\}$. Are good things happening?

(f) What happens when you increase Q_d while keeping R_d constant? What happens when you increase R_d while keeping Q_d constant? What happens when you increase or decrease both Q_d and R_d, keeping their ratio the same? Include enough plots to justify yourself.

4. *Implement your own version of dlqr.* Write a MATLAB function that solves the infinite-horizon, discrete-time, linear quadratic regulator problem. The method of solution should be to iterate until $K_d(i)$ converges (as in Problems 2 and 3). As a criterion for "convergence," I suggest you check that the absolute difference between all elements of $K_d(i)$ and the corresponding elements of $K_d(i+1)$ fall below a threshold `tol`, which can be passed as an optional argument to the function. Here is one way to implement this optional argument:

```
function Kd = myDLQR(Ad,Bd,Qd,Rd,tol)
if (nargin<5)
    tol = 1e-3;
end
% Add stuff here...
end
```

I've chosen `tol=1e-3` in this example, but you are free to choose whatever default tolerance you think works best. Your function should verify that $P(i)$ remains positive definite for all iterations until convergence and should throw an error otherwise. For example:

```
if ( blah ) % replace 'blah' with your check
    error('P(%d) is not positive definite',i);
end
```

Test your function by comparing its output to the output of MATLAB's built-in `dlqr` function, at least on the example from Problem 3 and on the following example:

$$A_d = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \qquad B_d = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \qquad Q_d = \begin{bmatrix} 10 & 0 \\ 0 & 1 \end{bmatrix} \qquad R_d = \begin{bmatrix} 1 \end{bmatrix}$$

Also test your function with at least one choice of Q_d that is not positive definite. You are, of course, free to test your function on other examples as well. From now on, please use your function `myDLQR` instead of `dlqr` in your code.

5. *Design and implement both an observer and a controller for trajectory tracking.* In this problem, you will continue building a quad-rotor simulation in MATLAB. Here, you will add an observer to compute state estimates and reimplement your controller so it uses these state estimates (and not the true states) and so it tracks a trajectory. There are four basic steps:

   - Design and implement an observer.
   - Change your controller so it operates on state estimates and not on states.
   - Implement trajectory tracking.
   - Tune your controller and observer so the quadrotor moves really fast (a race!).

   You will do so by modifying your code from HW2. Guidance will be posted to piazza in the coming days (as well as submission details) and will be discussed in class.