CS 4610/5335: Robotic Science and Systems (Spring 2021) Robert Platt
Northeastern University

HW 9: Optimal Control

Please remember the following policies:

- Submissions should be made electronically via the Canvas. Please ensure that your solutions for both the written and/or programming parts are present and zipped into a single file.
- Solutions may be handwritten or typeset. For the former, please ensure handwriting is legible.
- You are welcome to discuss the programming questions (but *not* the written questions) with other students in the class. However, you must understand and write all code yourself. Also, you must list all students (if any) with whom you discussed your solutions to the programming questions.

C1. Finite horizon LQR (**3 points**).

hw9(1): In this question, you must implement a finite horizon discrete time LQR for the damped mass system described in class and illustrated in the course slides. The time horizon, T, and the A and B matrices that encode the system dynamics are already encoded in hw9.m and you don't need to change them. Also, the QT, Q, and R cost matrices as well as the initial state, $x0$, are already encoded in hw9.m. What you need to do is to implement two functions: FH_DT_Riccati and getControl as described in Q1.m. FH_DT_Riccati should do the Riccati equation recursion. It should return a cell array, $Pseq$, where each cell is a 4×4 P matrix and the cell index is the same as the time index ($Pseq\{i\}$ denotes the 4×4 P matrix at time i). getControl should calculate the control action when the system is in state x at time i.

C2. Receeding Horizon LQR (**1 point**).

hw9(2): Exactly the same as above except that you should now implement a receeding horizon controller. Whereas above you were to find the optimal control for a fixed time horizon T, now you must execute a receeding horizon controller where you calculate a control action at each time step by optimizing over the *next* T time steps.

C3. LQR with different inputs (**3 points**).

hw8(3): In this question, we are going to use the LQR framework in a new way. As we have studied it so far, LQR can generate large control inputs that can change quickly. For example, the control output calculated in Q1 and Q2 is very large on the first few time steps. It is sometimes desirable to find a trajectory that minimizes the *change* in control input rather than the magnitude of the control input itself. It turns out that we can use LQR to calculate these sorts of control policies as well. Suppose that we want to minimize a cost function of the form:

$$J(X,U) = x_T^T Q_T x_T + \sum_{t=1}^{T-1} x_t^T Q x_t + u_t^T R u_t + \Delta u_t^T \hat{R} \Delta u_t,$$

where the last term in the summation imposes a cost on change in control input. We can achieve this behavior by defining a new system:

$$\begin{pmatrix} x_{t+1} \\ u_{t+1} \end{pmatrix} = \begin{pmatrix} A & B \\ 0 & I \end{pmatrix} \begin{pmatrix} x_t \\ u_t \end{pmatrix} + \begin{pmatrix} B \\ I \end{pmatrix} \Delta u_t$$

with cost function

$$\begin{aligned} J(X,U) &= \begin{pmatrix} x_T \\ u_T \end{pmatrix}^T \begin{pmatrix} Q_T & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} x_T \\ u_T \end{pmatrix} \\ &+ \sum_{t=1}^{T-1} \begin{pmatrix} x_t \\ u_t \end{pmatrix}^T \begin{pmatrix} Q & 0 \\ 0 & R \end{pmatrix} \begin{pmatrix} x_t \\ u_t \end{pmatrix} + \Delta u_t^T \hat{R} \Delta u_t. \end{aligned}$$

You should ask yourself the following questions: what is the new state vector representation? What are the new "A" and "B" matrices? Use this new representation to calculate the optimal trajectory in this scenario. You need to create three functions in Q3.m: FH_ DT_Riccati, getControl, and getSyntheticDynamics.

However, FH_DT_Riccati and getControl should be exactly the same functions as you created in Q1.m. The only new function is getSyntheticDynamics. This takes the underlying parameters of the system as input and produces as output the new modified parameters.

C4. Affine LQR (**3 points**).

This question is completely written – no programming required! The standard LQR formulation finds an optimal solution to the linear system,

$$x_{t+1} = Ax_t + Bu_t,$$

with a quadratic cost function,

$$J(u_{1:T-1}) = x_T^T Q_f x^T + \sum_{t=1}^{T-1} x_t^T Q x_t + u_t^T R u_t.$$

But sometimes we want to find an optimal solution to this problem (same cost function):

$$x_{t+1} = Ax_t + Bu_t + c,$$

where c is an additional constant term that makes the system dynamics *affine* instead of linear. It turns out that we can solve the affine version of the LQR problem by defining the following matrices:

$$\bar{A} = \begin{pmatrix} A & c \\ 0 & 1 \end{pmatrix}, \quad \bar{B} = \begin{pmatrix} B \\ 0 \end{pmatrix},$$

$$\bar{Q}_f = \begin{pmatrix} Q_f & q_f \\ q_f^T & \eta \end{pmatrix}, \quad \bar{Q} = \begin{pmatrix} Q & q \\ q^T & \eta \end{pmatrix},$$

where η is an arbitrary constant, and vectors:

$$\bar{x}_t = \begin{pmatrix} x_t \\ 1 \end{pmatrix}, \quad \bar{u}_t = u_t.$$

(a) Substitute the matrices above into $\bar{A}\bar{x}_t + \bar{B}\bar{u}_t$ in order to show that this parameterization solves the affine problem.

(b) Substitute into:

$$\bar{x}_T^T \bar{Q}_f \bar{x}_T + \sum_{t=1}^{T-1} \bar{x}_t^T \bar{Q} \bar{x}_t + \bar{u}_t^T R \bar{u}_t$$

to find the cost function that is optimized for.

(c) Give an example of an affine system that could be controlled using this method.