Spring 2019

Stanford
AA 203: Introduction to Optimal Control and Dynamic Optimization
Problem set 2, due on April 18

Problem 1: Consider the shortest path problem in Figure 1, where it is only possible to travel to the right and the numbers represent the travel times for each leg. The control is the decision to go up-right or down-right at each node.

(a) By using Dynamic Programming (DP), find the shortest path from A to B.

(b) Consider a generalized version of the shortest path problem in Figure 1 where the grid has n segments on each side. Find the number of computations required by an exhaustive search algorithm (i.e., the number of routes that such algorithm would need to evaluate) and the number of computations required by a DP algorithm (i.e., the number of DP evaluations). (For example, for the case where $n = 3$, the number of computations for the exhaustive search algorithm is 20 and the number of computations for the DP algorithm is 15.)

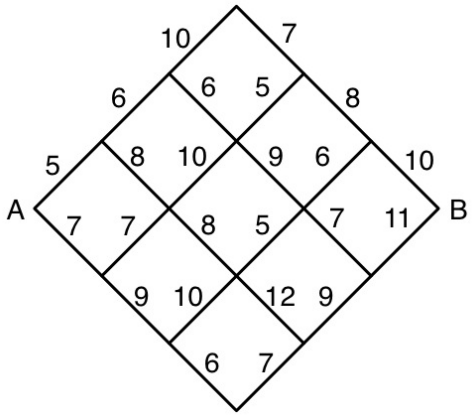

Figure 1: Shortest path problem for Problem 1.

Problem 2: Consider the two-dimensional system

$$\dot{x}_1(t) = u_1(t), \qquad \dot{x}_2(t) = u_2(t),$$

with the control constraint $\|u(t)\| = 1$. We want to find a state trajectory that starts at a given point $x(0)$, ends at another given point $x(T)$, and minimizes

$$\int_0^T r(x(t))\, dt.$$

The function $r(t)$ is nonnegative and continuous, and the final time T is subject to optimization. Suppose we discretize the plane with a mesh of size Δ that passes through $x(0)$ and $x(T)$, and we introduce a shortest path problem of going from $x(0)$ to $x(T)$ using moves of the following type: from each mesh point $\bar{x} = (\bar{x}_1, \bar{x}_2)$ we can go to each of the mesh points $(\bar{x}_1 + \Delta, \bar{x}_2)$, $(\bar{x}_1 - \Delta, \bar{x}_2)$, $(\bar{x}_1, \bar{x}_2 + \Delta)$, and $(\bar{x}_1, \bar{x}_2 - \Delta)$, at a cost $r(\bar{x})\Delta$. Show by example that this is a bad discretization of the original problem in the sense that the shortest distance need not approach the optimal cost of the original problem as $\Delta \to 0$.

Problem 3: Consider the problem of inscribing an N-side polygon in a given circle, so that the polygon has maximal perimeter.

(a) Formulate the problem as a DP problem involving sequential placement of N points in the circle.

(b) Use DP to show that the optimal polygon is regular (all sides are equal).

Problem 4: In this question you will design a stabilizing controller to balance an inverted pendulum on a cart, which is a classic test problem in control. The system has state variables x, corresponding to the horizontal position of the cart, and θ, corresponding to the angle of the pendulum (where $\theta = 0$ corresponds to hanging straight down). The control moves the cart horizontally. We will write the combined state as $\boldsymbol{s} = [x, \theta, \dot{x}, \dot{\theta}]^T$ and the action as u. The continuous time dynamics of the system are given by

$$\dot{\boldsymbol{s}} = f(\boldsymbol{s}, u) = \begin{bmatrix} \dot{x} \\ \dot{\theta} \\ \frac{u + m_p \sin\theta (l\dot{\theta}^2 + g\cos\theta)}{m_c + m_p \sin^2\theta} \\ \frac{-u\cos\theta - m_p l \dot{\theta}^2 \cos\theta \sin\theta - (m_c + m_p) g \sin\theta}{m_c + m_p \sin^2\theta} \end{bmatrix}. \tag{1}$$

We will discretize these dynamics via Euler integration and linearize around the upright position. Note that $f(\boldsymbol{s}^*, u^*) = 0$ for $\boldsymbol{s}^* = [0, 0, \pi, 0]^T$, $u^* = 0$, and thus this is a stationary point for this system. Linearizing around this point, we can write our discrete time dynamics

$$\delta \boldsymbol{s}_{k+1} = \underbrace{(I + dt \frac{\partial f}{\partial \boldsymbol{s}}(\boldsymbol{s}^*, u^*))}_{A} \delta \boldsymbol{s}_k + \underbrace{dt \frac{\partial f}{\partial u}(\boldsymbol{s}^*, u^*)}_{B} u_k \tag{2}$$

where dt is the timestep of our integration. In this expression,

$$\frac{\partial f}{\partial \boldsymbol{s}}(\boldsymbol{s}^*, u^*) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & a_1 & 0 & 0 \\ 0 & a_2 & 0 & 0 \end{bmatrix}, \quad \frac{\partial f}{\partial u}(\boldsymbol{s}^*, u^*) = \begin{bmatrix} 0 \\ 0 \\ \frac{1}{m_c} \\ \frac{1}{l m_c} \end{bmatrix}, \tag{3}$$

where

$$a_1 = \frac{m_p g}{m_c}, \quad a_2 = \frac{(m_c + m_p)g}{l m_c} \tag{4}$$

We will use A, B to design a stabilizing LQR controller. We will choose $Q = I, R = I$. In class, we saw that finite horizon LQR controllers could be designed which result in time-varying feedback gains. Here, we will design an infinite horizon LQR controller to minimize the cost

$$\sum_{k=0}^{\infty} \frac{1}{2}(\boldsymbol{s}_k - \boldsymbol{s}_k^*)^T Q(\boldsymbol{s}_k - \boldsymbol{s}_k^*) + \frac{1}{2} u_k^T R u_k. \tag{5}$$

After n iterations of the Riccati equation, the value matrices $\{P_k\}_{k=0}^n$ and the feedback gains $\{L_k\}_{k=0}^n$ gives the time-varying LQR controller which minimizes the quadratic cost over horizon n. We will write P_∞ and L_∞ for the value and feedback gain of the infinite horizon LQR problem, respectively. If a linear time-invariant system (A, B) is controllable, the feedback gain matrix L_∞ will be constant for all time, and the value matrix P_∞ will be finite. Moreover, if the Riccati recursion is carried out repeatedly, P_n will asymptotically converge to P_∞ and L_n to L_∞. Therefore, a way to approximate P_∞ is to run the Riccati recursion a number of times large enough such that it has approximately converged. This is the approach we will take in this exercise.

(a) Fill in `lqr_infinite_horizon_solution.m`, first writing the A, B matrices, followed by writing the Riccati recursion. The parameter values are provided in that function. Use `simulate_p4.m` to simulate the system without disturbances.

(b) We will now simulate the system with noise to investigate disturbance rejection of the stabilizing controller. Turn on noise (by setting line 13 to true) and simulate the system again. Submit the plots for both simulations.

Problem 5: Next, we will implement a controller to solve the more challenging "swing up" problem, in which the pendulum starts facing downwards and is brought to an upright position. For this problem, it is not sufficient to linearize around a single stationary point, and so we will turn to iterative LQR. Our dynamics will take the form

$$\delta \boldsymbol{s}_{k+1} = \underbrace{(I + dt \frac{\partial f}{\partial \boldsymbol{s}}(\bar{\boldsymbol{s}}_k, \bar{u}_k))}_{A_k} \delta \boldsymbol{s}_k + \underbrace{dt \frac{\partial f}{\partial u}(\bar{\boldsymbol{s}}_k, \bar{u}_k)}_{B_k} \delta u_k. \tag{6}$$

Evaluating the derivatives of the dynamics can be unwieldy, so we have included a function to perform this linearization and return A_k, B_k. To implement the controller, you will fill in `ilqr_solution.m`. We will use the cost function

$$\frac{1}{2}(\boldsymbol{s}_N - \boldsymbol{s}^*)^T Q_N (\boldsymbol{s}_N - \boldsymbol{s}^*) + \sum_{k=0}^{N} \left(\frac{1}{2}(\boldsymbol{s}_k - \boldsymbol{s}^*)^T Q (\boldsymbol{s}_k - \boldsymbol{s}^*) + \frac{1}{2} u_k^T R u_k \right) \tag{7}$$

with Q_N large to enforce a soft terminal constraint.

(a) Rewrite the cost function in terms of deviations $\delta \boldsymbol{s}_k, \delta \boldsymbol{s}_N, \delta u_k$. This will result in a cost function with additional linear terms. Fill in these linear terms in lines 40, 51, 52 of `ilqr_solution.m`.

(b) The backward pass of iLQR consists of evaluating the Riccati recursion. Write the function `backward_riccati_recursion` to return the controller term l, L and the value terms p, P.

(c) Finally, we will implement the forward pass. Implement the nominal control update on line 63.

Run `simulate_p5.m` to run the simulations after you complete the controller. This script first performs the swing up maneuver generated by the iLQR controller, and then switches to the stabilizing infinite horizon LQR controller from the previous question. It will generate plots of the state and control trajectories. First, run the simulations without noise (the default), and observe the response. Then, turn on noise (set line 13 of `simulate_p5.m` to true) and run the simulation with noise. Submit the plots for simulations both with and without noise.

Learning goals for this problem set:

Problem 1: To familiarize with the DP algorithm and to appreciate the computational savings of DP versus an exhaustive search algorithm.

Problem 2: To gain insights into the delicate issue of discretization.

Problem 3: To familiarize with the process of casting an optimal control problem into a form amenable to a DP solution.

Problems 4 and 5: To gain experience with implementing LQR and iterative LQR controllers.