Homework 4: Linear Quadratic Control & Reinforcement Learning

Collaboration in the sense of discussion is allowed, however, the work you turn in should be your own - you should not split parts of the assignments with other students and you should certainly not copy other students' code or papers. See the collaboration and academic integrity statement here: https://natanaso.github.io/ece276b. *Books may be consulted but not copied from.*

Submission

You should submit the following four files on **Gradescope** by the deadline shown in the top right corner.

1. **FirstnameLastname_HW4_P1.pdf**: upload your solution to Problem 1. You may use latex, scanned handwritten notes (write legibly!), or any other method to prepare a pdf file. Do not just write the final result. Present your work in detail, explaining your approach at every step.
2. **FirstnameLastname_HW4_P2.zip**: upload the code you have written for Problem 2 in a zip file
3. **FirstnameLastname_HW4_P3.zip**: upload the code you have written for Problem 3 in a zip file
4. **FirstnameLastname_HW4_P4.pdf**: upload the report for Problem 4. You are encouraged but not required to use an IEEE conference template[1] for your report.

Problems

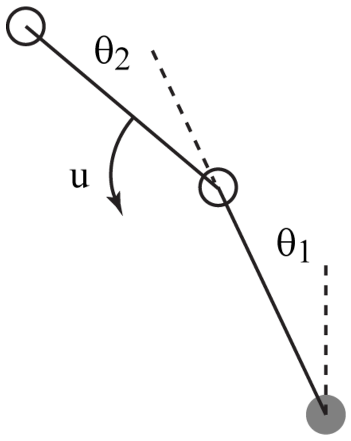
1. [25 pts] Consider the continuous-time scalar nonlinear system $\dot{x}(t) = -x(t)u(t)$ with initial condition $x(0) = 1$. We are interested in minimizing the cost:

$$\int_0^1 [x(t)u(t)]^2 dt + x^2(1)$$

   (a) Write the Hamilton-Jacobi-Bellman partial differential equation (HJB PDE) for this problem
   (b) Simplify the HJB PDE by computing the minimum analytically to obtain a quadratic PDE
   (c) Guess a solution of the form $V(t,x) = \frac{1}{2}\alpha(t)x^2 + \beta(t)x + \gamma(t)$ and use the terminal conditions to find the functions $\alpha(t)$, $\beta(t)$, $\gamma(t)$ that make $V(t,x)$ a solution to the HJB PDE (*Hint*: The solution to an ODE of the form $\dot{y}(t) = y^2(t)$ can be obtained by dividing both sides by $y^2(t)$ and integrating).
   (d) We have shown that the specific choice of $\alpha(t)$, $\beta(t)$, $\gamma(t)$ you found in part c) makes $V(t,x)$ a solution to the HJB PDE and hence $V(t,x)$ is the optimal cost-to-go. Determine the corresponding optimal policy.

2. [25 pts] **Programming Assignment** In this problem, we will consider a deterministic double pendulum system known as the acrobot (Fig. 1). The acrobot state $x = \left(\theta_1, \theta_2, \dot{\theta}_1, \dot{\theta}_2\right)^T$ is defined by the joint angles θ_1, θ_2 and their velocities. The scalar control input u is the torque acting on the second joint, leading to

[1] https://www.ieee.org/conferences_events/conferences/publishing/templates.html



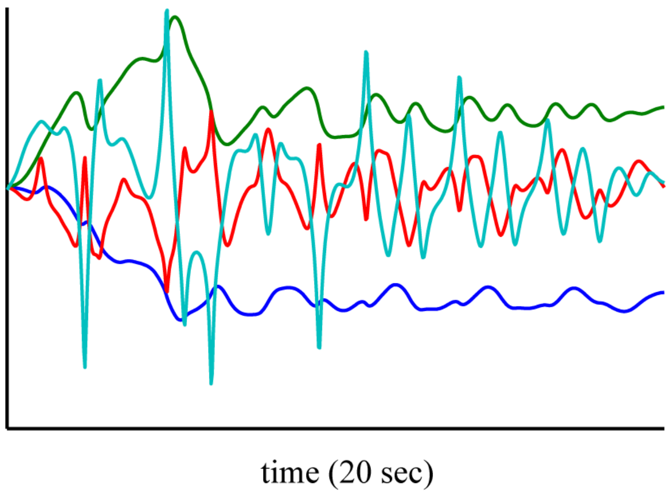


Figure 1: Acrobot system (left) and joint poisitions and velocities over time for an uncontrolled evolution. Only the second joint (elbow) is actuated.

the following system dynamics:

moment of Inertia : $M(x) = \begin{bmatrix} 3+2\cos(x_2) & 1+\cos(x_2) \\ 1+\cos(x_2) & 1 \end{bmatrix}$

coriolis, centripetal, and gravitational forces : $c_1(x) = x_4(2x_3+x_4)\sin(x_2) + 2g\sin(x_1) + g\sin(x_1+x_2)$

$c_2(x) = -x_3^2\sin(x_2) + g\sin(x_1+x_2)$

passive dynamics : $a(x) = \begin{bmatrix} I_{2\times2} & 0 \\ 0 & M^{-1}(x) \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \\ c_1(x) - \zeta x_3 \\ c_2(x) - \zeta x_4 \end{bmatrix}$

control gain : $B(x) = \begin{bmatrix} I_{2\times2} & 0 \\ 0 & M^{-1}(x) \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

complete system dynamics : $\dot{x} = a(x) + B(x)u,$

where ζ is the friction and g is the gravity acceleration. The acrobot dynamics can be simulated and visualized using the provided script `acrobot.py`. Our goal is to design a control policy which makes the acrobot stay in upright position ($\theta_1 = \theta_2 = 0$) with minimal control effort. Thus, we will consider the stage cost:

$$g(x,u) = 1 - e^{k\cos x_1 + k\cos x_2 - 2k} + \frac{r}{2}u^2$$

where r determines the relative importance of conserving control energy versus staying upright and k is the width of the position cost. You should adjust r and k so as to generate interesting behaviors. Note that the angles should be kept inside the interval $[-\pi, \pi]$.

(a) Linearize the acrobot dynamics around the state $x_0 = \left(0,0,0,0\right)^T$ to obtain the following linear approximation (*Hint*: consider using sympy):

$$\dot{x} = A_0 x + B_0 u$$

(b) Compute a quadratic approximation to the stage cost in the form:

$$g_0(x,u) = \frac{1}{2}x^T Q x + \frac{r}{2}u^2$$

(c) Now that you have a linear system and a quadratic cost, solve the continuoust-time linear quadratic regulation (LQR) problem via the continous algebraic Riccati equation[2]. This will give you a linear policy of the form $\pi(x) = Mx$ that should make the linearized system go to the goal state from any initial state (test this in simulation).

(d) What about the actual nonlinear system? Plug in your linear policy in the nonlinear dynamics:

$$\dot{x} = a(x) + B(x)Mx$$

and integrate (using scipy.integrate.odeint or scipy.integrate.ode) the closed-loop system forward in time for a variety of initial states. Find a way to characterize (numerically) the region of initial state around x_0 for which the policy works well.

3. [25 pts] **Programming Assignment** In this problem, we will consider the hill automobile problem. Our automobile is stuck between two hills and its underpowered engine, even at full throttle, is unable to climb up the steep hill on the right (Fig. 2). Our task is to find a control policy that takes advantage of

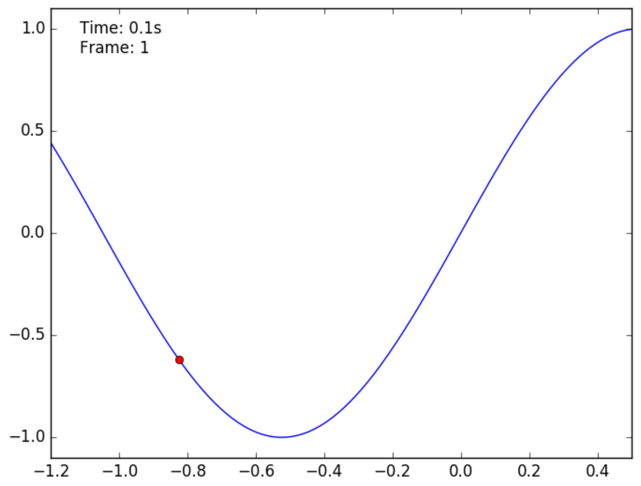


Figure 2: Our automobile (red dot) is stuck between two hills and needs to find a way to overcome the steep hill on the right despite its underpowered engine.

the potential energy obtained by swinging between the two hills before driving straight towards the goal on the right. The state space is defined by the position x of the automobile in the interval $[-1.2, 0.5]$ (m) and the velocity v in $[-1.5, 1.5]$ (m/s). The height of the hills is determined by $\sin(3x)$. There are three possible controls at each state $u \in [-2.0, 0.0, 2.0]$ representing the force applied to the automobile. The mass of the car is $m = 0.2\,kg$, the gravity is $g = 9.8\,m/s^2$, the friction is $k = 0.3$, and the time step is $\tau = 0.1s$. The terminal reward for reaching the goal is $+1$ and there is a stage cost of -0.01 at each time step (we want to get there fast!). The discretized automobile dynamics are:

$$\begin{aligned} v_{t+1} &= v_t + \left(g\,m\cos(3x_t) + \frac{u_t}{m} - k\,v_t\right)\tau \\ x_{t+1} &= x_t + v_{t+1}\tau. \end{aligned}$$

Suppose that the system dynamics above are unknown or we just don't want to deal with them. We will try to learn a good control policy (and associated value function) by using reinforcement learning. Discretize the position space $[-1.2, 0.5]$ amd velocity space $[-1.5, 1.5]$ into N bins each (*Hint*: numpy.linspace() and numpy.digitize() might be useful here).

[2] https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.linalg.solve_continuous_are.html

(a) Apply policy iteration, using a Monte-Carlo (MC) method to estimate the cost-to-go J^π of a given policy π and an ϵ-greedy strategy to improve the policy.

(b) Apply policy iteration, using a Temporal-Difference (TD) method to estimate the cost-to-go J^π of a given policy π and an ϵ-greedy strategy to improve the policy.

(c) **(optional)** Apply policy iteration, using the known system dynamics to compute the cost-to-go J^π of a given policy π and the usual policy improvement step.

For the TD and MC methods above, make sure to present in your report:

- a plot of the mean and variance of the sample costs per episode as the number of simulation episodes grows
- a plot of the mean and variance of the number of steps to goal per episode as the number of simulation episodes grows
- a visualization of your best policy (N by N matrix) and the minimum time it takes it to reach the goal to the right

Compare and discuss the performace of all methods in your report (Problem 4).

4. [25 pts] Write a report describing your approach to the two programming assignments above. Your report should include the following sections:

   - **Introduction**: discuss why the problem is important and present a brief overview of your approach
   - **Problem Formulation**: state the problem you are trying to solve in mathematical terms. This section should be short and clear and should define the quantities you are interested in using mathematical terms.
   - **Technical Approach**: describe your approach to the acrobot and hill-automobile problems. In the former case, explain the linearization of the system dynamics and cost function and the form and solution of the Riccati equation. In the latter case, describe the Monte-Carlo and Temporal-Difference algorithms you used.
   - **Results**: present and visualize the results for both problems and discuss your choice of parameters and algorithm performance – what worked as expected and what did not and why.