University of Pennsylvania

ESE 650: Learning in Robotics

Spring 2020

[02/25] Homework 3

Due: 03/06 11.59pm

Submission: You will submit LaTeX typeset solutions (strongly encouraged) on Gradescope. You can use template.tex on Canvas in the "Homeworks" folder to do so.

- Please start a new problem on a fresh page and mark all the pages corresponding to each problem. Failure to do so may result in your work not being graded completely.
- **Mention how much time (in hours) you worked on each problem. It helps us keep track of the perceived difficulty of the homework problems.**
- You are encouraged to collaborate on your problem set. However, you should write your solutions independently. Clearly indicate the name and PennKey of all your collaborators on your submitted solutions.
- For each problem in the problem set, mention the total amount of time you spent on it.
- Code for each problem should be clearly marked. Make sure you follow the provided template exactly, the autograder assumes function and file names.
- You can be informal while typesetting problem sets, e.g., if you want to draw a picture feel free to draw it on paper clearly, click a picture and include it in your solution. Do not spend undue time on typesetting solutions.

Credit: The points for the problems add up to 135. You only need to solve for 100 points to get full credit, i.e., your final score will be $\min(\text{your total points}, 100)$.

Problem 1 (25 points). Consider a damped harmonic oscillator which is governed by the ordinary differential equation (ODE)

$$m\,\ddot{x} + kx + b\dot{x} = u; x(0) = 1, \dot{x}(0) = 0.$$

The state $x \in \mathbb{R}$ and the control $u \in \mathbb{R}$. We would like to control this system using LQR.

1. (3 points) Convert the ODE into the state-space representation $\dot{z} = Az + Bu$.
2. (5 points) If $m = 1$ kg, $b = 0.5$ and the spring constant $k = 3$ N/m, plot the autonomous response of the system from the given initial condition. Autonomous response is the response when the control input $u(t) = 0$ for all times t. What is the equilibrium state of the system with zero control input?
3. (2 points) Design an LQR controller to force the oscillator's state to the equilirbium more quickly than the autonomous version. Write down the cost function $J(x(t), u(t))$ that you will optimize for LQR.
4. (15 points) Calculate the LQR gain for the following values of matrices Q and R

   (a) $Q_{11} = 1, Q_{22} = 1, R = 1$,

   (b) $Q_{11} = 100, Q_{22} = 1, R = 1$,

   (c) $Q_{11} = 1, Q_{22} = 1, R = 100$.

   All other entries of Q are set to zero. In the same plot, show the response of the system $x(t)$ as a function of t for all these different costs. In a different plot, show the control input $u(t)$ as a function of t for all these different costs. Explain the trends you see in these plots.

You can to use scipy.linalg.solve_continuous_are to solve the Riccati equation. You don't need to submit the code for this problem.

Problem 2 (50 points). The Acrobot is a classical under-actuated dynamical system. It is a model for a gymnast swinging up on a bar.

The gymnast only applies control input at his waist. He begins from his initial position of hanging down from the bar and wants to end up at a terminal position where he is upright. Note that the vertical position is an equilibrium position, he stay there indefinitely without any control input (https://www.youtube.com/watch?v=O2b03YtMeRU). In this problem you will write a dynamic programming solution to take the Acrobot from the initial condition to the terminal condition.

1. The dynamical equations for the Acrobot are involved, you do not have to derive them, you can directly use equations 1-3 in http://www.cds.caltech.edu/ murray/preprints/erl-M91-46.pdf. The numerical values of the parameters are given in Table 1. Use the "balanced" values for the parameters in Table 1. You are encouraged to read Sections 1-2 of this paper to become more familiar with this system.

2. (15 points) We want to use discrete-state, discrete-control dynamic programming to solve this problem. The problem has a 4-dimensional state-space $\left[\theta_1, \theta_2, \dot{\theta_1}, \dot{\theta_2}\right]$ and one control input $\tau \in \mathbb{R}$. Let us think of how to discretize the state-space. We know that $\theta_1, \theta_2 \in [0, 2\pi]$, if we imagine the number of intervals to be about 100, we use a discretization of $\Delta\theta_1 = \Delta\theta_2 = 0.05$. Discretizing the angular velocities is harder. Swingup involves very quick motion of the limbs and therefore we want to be able to handle large $\dot{\theta_1}, \dot{\theta_2}$. Let us set $\dot{\theta_1}, \dot{\theta_2} \in [-5, 5]$ radians/sec and $\Delta\dot{\theta_1} = \Delta\dot{\theta_2} = 0.1$ to obtain about 100 discrete intervals. Note that the total number of states in the problem is exponential in the dimensionality, even with our careful choices we have about 100^4 states. Assume that the control $\tau \in [-1, 1]$ and pick $\Delta\tau = 0.1$ to get 20 discrete control inputs. You should not use transition matrices to store the discrete dynamical system, you could use a sparse matrix or (in view of sub-part 5 below) simply store the transitions as a list before giving it to NetworkX.

3. (5 points) Let's assume that the run-time cost does not depend on time (the gymnast does not get tired during swingup). Argue why the run-time cost should not depend on $x(t)$. Pick a cost function $q(x(t), u(t))$. Pick a terminal cost function $q_f(x(T))$.

4. (5 points) You will solve this as a stochastic shortest path problem because we do not know what time-horizon T we should pick for the gymnast to swing up. What is the terminal "sink" state?

5. (25 points) Write the code to solve the stochastic shortest path problem. You may use the shortest path algorithms inside https://networkx.github.io. If you cannot solve sub-part 4 above, you can also pick a reasonable value for the time horizon T and solve Acrobot as a standard shortest-path problem. Plot the (four) state trajectories as a function of time on the same plot; also plot the control as a function of time. If you find debugging the code difficult, you could think of coding up a simplistic animation using Python's Matplotlib to see the Acrobot in action.

Problem 3 (60 points). In this project, you will implement the structure of mapping and localization in an indoor environment using information from an IMU and range sensors. You will integrate the IMU orientation and odometry information from a walking humanoid with a 2D laser range scanner (LIDAR) in order to build a 2D occupancy grid map of the walls and obstacles in the environment. Training sets of odometry, inertial, and range measurements from a THOR-OP humanoid robot will be provided for this project.

1. Data

Once you download the code base, all training data are provided in `data/train`. There are 4 training datasets differentiated by ID numbers from 0 to 3, corresponding to 4 different trajectories generated by the THOR robot in Town building at Penn. For example, dataset 0 consists of `data/train/train_lidar0.mat` and `data/train/train_joint0.mat`. You are also provided a rich set of documents to guide you on this problem. To understand more about the data, go to `docs/guidance/config_slam.pdf`. Each dataset contains timestamped sensor values, corresponding to the raw sensor readings. To understand more about the data, go to `docs/guidance/config_slam.pdf`.

2. General Guidance

A walk-through guide for this project is provided in detail at `docs/guidance/guidance_slam.pdf`. You are also provided a code structure with two unit tests to debug your code. `README.md` details how to run the code. Your task is to fulfill all `TODO` appearing on the code. The `main()` function will return an array of size $3 \times N$, representing the state estimate over time, with N is the number of time steps starting when both IMU and LiDAR are launched.

There will be a long list of transformation functions need for this project. To reduce your stress, most of these functions are already provided in `transformation.py`.

3. Deliverables

Upload your code to `Homework 3 Project - Code` on Gradescope. The autograder score is worth 20 points. In addition, please provide the maps from the 4 datasets in your writeup. These will constitute the remaining 40 points (10 each).