Project 3: Infinite-Horizon Stochastic Optimal Control

*Collaboration in the sense of discussion is allowed, however, the assignment is **individual** and the work you turn in should be entirely your own. See the collaboration and academic integrity statement here:* https://natanaso.github.io/ece276b. *Books may be consulted but not copied from. Please acknowledge **in writing** people you discuss the problems with and provide references for any books or papers you used.*

Submission

Please submit the following files on **Gradescope** by the deadline shown at the top right corner.

1. Programming assignment: upload all code you have written for the project and a README file with a clear, concise description of how to run your code.
2. Report: upload your report in pdf format. You are encouraged but not required to use an IEEE conference template[1] for your report.

Problems

In square brackets are the points assigned to each part.

1. In this problem, the task is to balance an inverted pendulum (see Fig. 1). We will model this as an infinite-horizon discounted stochastic optimal control problem. The pendulum dynamic evolve in continuous time and space. We will discretize the time as well as the state and control spaces to formulate a discrete-time finite-state MDP problem and solve it via value iteration and policy iteration. The pendulum state $\mathbf{x} = (x_1, x_2)^\top$ consists of the pendulum angle $x_1 \in [-\pi, \pi)$ with respect to the vertical line and the angular velocity $x_2 \in \mathbb{R}$. The continuous-time pendulum dynamics are:

$$d\mathbf{x} = f(\mathbf{x}, u)\, dt + \boldsymbol{\sigma}\, d\omega \qquad f(\mathbf{x}, u) := \begin{bmatrix} x_2 \\ a \sin x_1 - b\, x_2 + u \end{bmatrix} \qquad \boldsymbol{\sigma} := \begin{bmatrix} \sigma_1 \\ \sigma_2 \end{bmatrix} \in \mathbb{R}^2$$

   where $a > 0$ summarizes the effects of gravity, mass and length of the pendulum, $b > 0$ represents damping and friction, $u \in \mathbb{R}$ is the control input and ω is Brownian motion noise affecting the state through the parameters $\sigma_1, \sigma_2 \in \mathbb{R}$. We will model the stage cost at each state $\mathbf{x}$ and control u as:

$$\ell(\mathbf{x}, u) := 1 - \exp(k \cos x_1 - k) + \frac{r}{2} u^2$$

   where $k > 0$ determines the shape of the cost and $r > 0$ scales the control cost. Note that the cost penalizes x_1 that is away from 0 and is quadratic in the control u. Formulate an infinite-horizon Discounted problem with a discount factor $0 < \gamma < 1$. Experiment with the choice of parameters $(a, b, \boldsymbol{\sigma}, k, r, \gamma)$ discuss how these parameters affect the pendulum dynamics and the performance (convergence speed, pendulum balance, sensitivity to noise, etc.) of your algorithms described below.

   Since time and the control and state spaces are continuous, we need to discretized them. First, choose a time step τ, maximum velocity v_{max} and maximum control u_{max}. You can discretize the state and control spaces into (n_1, n_2, n_u) number of grid points. Be careful that x_1 is an angle and wrap-around should enforced, while x_2 and u may be discretized over the intervals $[-v_{max}, v_{max}]$ and $[-u_{max}, u_{max}]$. Experiment with the constants $(\tau, n_1, n_2, n_u, v_{max}, u_{max})$.

   Euler discretization with step τ leads to a transition model $p_f(\mathbf{x}'|\mathbf{x}, u)$ that is Gaussian with mean $\mathbf{x} + f(\mathbf{x}, u)\tau \in \mathbb{R}^2$ and covariance $\boldsymbol{\sigma}\boldsymbol{\sigma}^\top \tau \in \mathbb{R}^{2\times 2}$. To create transition probabilities in the discretized MDP, for each discrete state $\mathbf{x}$ and each discrete control u, we choose the next states/grid points $\mathbf{x}'$ around the mean $\mathbf{x} + f(\mathbf{x}, u)\tau$, evaluate the the Gaussian at the chosen grid points, and normalize so that the outgoing transition probabilities sum up to 1. See Fig. 2 for an example. The stage cost of the discretized MDP may be chosen as $\ell(\mathbf{x}, u)\, \tau$.

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

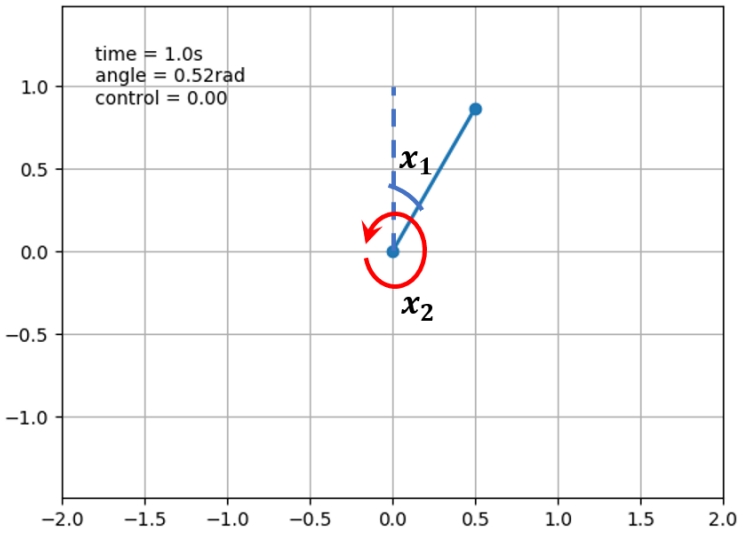

Figure 1: An **inverted pendulum** with angle x_1 and angular velocity x_2 should be controlled by a torque input u to maintain an upright ($x_1 = 0$) position.

Generally, denser grids produce more accurate solutions in the original continuous-time, continuous-space problem. It is also useful to think about "compatibility" of the grid in the following sense. Suppose that the time step τ and the angular velocity discretization step are small but the angle discretization step is large. Then, it will be very difficult for the MDP to make any state transitions, i.e., the transition probabilities will be close to delta functions centered at the current state. This is to be avoided. Instead, we should aim for

$$\frac{2v_{max}}{n_2}\tau \approx \frac{2\pi}{n_1}$$

and apply similar reasoning to the velocity and control discretization.

Now, that we have formulated a discrete-space discounted problem, we can use value iteration (VI) and policy iteration (PI) to obtain an optimal control strategy. For your choice of parameters, compare the convergence of VI and PI. After you solve the MDP, you have a control policy defined on the grid. Use interpolation to extend the control policy to continuous space and simulate several trajectories for the continuous system, starting at different initial states. Use `main.py` to help visualize your controller and if your controller and problem and discretization parameters are sensible. If things are working well, you should see the pendulum going to the vertical state and balancing there. Gradually increase the noise and discuss its effect. At what noise levels are you not able to achieve stable equilibrium in the vertical state? Feel free to modify `main.py` as needed. The file is just provided as a starting point.

Write a project report describing your approach with the following sections:

- [5 pts] **Introduction:** present a brief overview of the infinite-horizon stochastic optimal control problem and the techniques used in this project.
- [10 pts] **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. In particular, you should include the basic elements of the optimal control problem you are trying to solve. Make sure to include:
  - the MDP formulation,
  - the optimization problem with proper objective function & constraints,
  - the interpolation problem.

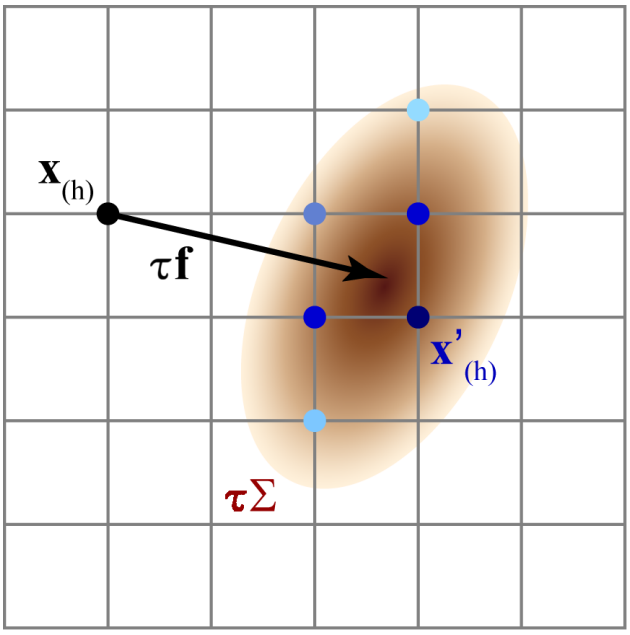

Figure 2: To create transition probabilities in the discretized MDP, for each discrete state $\mathbf{x}$ and each discrete control u, we choose the next states/grid points $\mathbf{x}'$ around the mean $\mathbf{x} + f(\mathbf{x}, u)\tau$, evaluate the the Gaussian at the chosen grid points, and normalize so that the outgoing transition probabilities sum up to 1.

- [30 pts] **Technical Approach**: describe your approach to value & policy iteration and the key differences between these algorithms. Make sure to include:
  - the Value Iteration algorithm,
  - the Policy Iteration algorithm,
  - your policy interpolation method that allows you to apply the discrete actions to the continuous-time/space system.
- [30 pts] **Results**: present any interesting details from your implementation and discuss the performance statistics of the two algorithms in detail. Make sure to include:
  - discussion of how the problem parameters $(a, b, \boldsymbol{\sigma}, k, r, \gamma)$ and the discretization parameters $(\tau, n_1, n_2, n_u, v_{max}, u_{max})$ affect the pendulum dynamics and the algorithm performance (convergence speed, pendulum balance, sensitivity to noise, etc.),
  - comparison of $V(x)$ over episodes between VI and PI for a set of states,
  - plots of the optimized policy $\pi(x)$ over the discretized state space for VI and PI.

  What worked as expected and what did not and why? Consider including images or videos of the performance of your algorithms.
- [25 pts] **Code:** upload your code to Gradescope. It will be evaluated based on correctness and efficiency. The results presented in the report should be consistent with the output of your code. The code should have a clear structure and comments. **Copying, rephrasing, or even looking at anyone else's code is strictly forbidden. Writing your own code is the best way to avoid plagiarism discussions or consequences.**