EECS 270 – Robot Algorithms
Lab N. 2 – Fall 2017

Important Information

1. This lab is due by the deadline specified in CatCourses.
2. You solution must be submitted electronically through CatCourses.
3. Labs MUST be solved individually.
4. Your solution must be coded in Matlab and strictly follow the given requirements. Failure to comply with the desired structure for the input/output files leads to an immediate 0.

Using an MDP controller for the inverted pendulum

Consider the inverted pendulum (pole balancing) setup described in Example 3.4 in the book and let us consider the discounted formulation described therein with $\gamma = 0.95$. Assume that the state is $[\theta \quad \dot{\theta}]^T$ and the same dynamics used in the first lab. Failure is defined when $|\theta| > \pi/4$. Assume that at each step the state of the pendulum is subject to Gaussian noise with mean $\mu = [0\ 0]$ and covariance matrix

$$\Sigma = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.001 \end{bmatrix}$$

Assume all constants in the model (m, l, J) are equal to 1, except $g = 9.8m/s^2$.

1. determine the optimal policy using the value iteration algorithm. That is to say you need to determine the optimal value function, and then the optimal policy. Once the algorithm terminates, print to the screen, for each state, the optimal value function, and the optimal action.
    - as a preliminary step, you have to discretize the state space. It is up to you to experiment and figure out a good discretization step. There is an obvious tradeoff in resolution vs. speed.
    - it is probably simpler if you model the MDP using $r(s, a)$ and $P(s'|s, a)$ as opposed to $P(s', r|s, a)$. To this end, you should exploit the fact that the two components of the noise vector are independent, as evidenced by the diagonal matrix Σ.

- you can initially assume that all states have the following set of actions: $\{-2, -1, 0, 1, 2\}$. However, depending on the discretization you pick, you can also experiment with adding more actions. Keep the set always symmetric with respect to 0, e.g., add ± 3, etc.

2. write a function `simulatePendulum` that simulates one episode of the evolution using the optimal policy you computed in the previous question. Once the episode terminates, produce a plot where on the x axis you have the temporal index and on the y axis you display θ. If the episode does not end within 500 steps, stop it and produce the chart for the 500 steps.