UNIVERSITY OF PENNSYLVANIA

ESE 650: LEARNING IN ROBOTICS

SPRING 2023

MIDTERM EXAM

03/22 WED 10.15 AM – 11.45 AM ET

DURATION: 75 MINUTES

MAXIMUM POINTS: 72

Read the following instructions carefully before you begin.

- This is a closed book exam. You are allowed to use one A4/Letter paper cheat sheet (front and back).
- You may not use laptops/phones/tablets or the Internet to look up a textbook/notes/any other material during this exam. You may not discuss with your peers.
- You will have 75 minutes from the time you start the exam. After that you will get 15 minutes to upload your solutions to Gradescope.
- You should use a pen and paper (do not use a pencil because it might not be readable when scanned, there will be plenty of empty space for you to use for scratch work) to write your solutions and click pictures and upload your solutions as a PDF (Dropbox or Apple Notes apps on your phone are great for scanning all pages as a single PDF). Make sure to write in legible handwriting; we cannot grade what we cannot read.
- You should bring a phone/tablet to take pictures of your exam. You will then click pictures of each page and upload these pages to Gradescope from the phone, or after transferring this to your laptop/tablet if you wish. Having all the exams on Gradescope makes it easy for us to grade, attend to regrade requests etc.
- Questions that are not correctly annotated on Gradescope outline will not receive credit.
- Each question mentions how short/long we would like your responses to be. DO NOT write excessively verbose answers.
- None of the questions require long derivations. If you find yourself going through lots of equations reconsider your approach or consider moving on to the next question.
- If you are stuck, explain your answers and what you are trying to do clearly. We will give partial credit if you are on the right track.

- **This exam has 4 problems.** The questions are NOT arranged in order of difficulty. Try to attempt every question. You do not need to write code for any of the questions.

Problem 1 (8 points). Imagine a one-dimensional car ($x, u \in \mathbb{R}$) that starts with zero velocity at the origin and has dynamics

$$\ddot{x} = u; \quad x(0) = \dot{x}(0) = 0;$$

the control input u controls the acceleration of the car and is constrained to be at most 1 in its magnitude. Find the optimal control trajectory if we want to reach the location $x = 10$ as quickly as possible and have zero velocity at the end.

Answer: The maximum acceleration/deceleration of the car is 1. If the acceleration at any time $u(t)$ is less than 1 in magnitude, the time to reach the destination $x = 10$ can be reduced by increasing this magnitude. This proves that the $|u(t)| = 1$ for all times t. The distance traveled in time t with $u(t) = 1$ is

$$\frac{t^2}{2}.$$

So we are simply looking for the solution of

$$t^2 = 10$$

where half the time is spent with $u(t) = 1$ and the other half is $u(t) = -1$ to come to a stop. Thus $t = \sqrt{10}$ and the total time taken to travel to $x = 10$ is $2t = 2\sqrt{10}$. The velocity is zero at the end because we are accelerating and decelerating for equal amounts of time.

Problem 2 (26 points). The following questions require you to write 1-2 sentences each.

(1) **(2 points)** If the dynamics and observations depend linearly on their arguments and are Gaussians with diagonal covariances $\sigma_1^2 I$ and $\sigma_2^2 I$ respectively

$$\mathrm{P}(x' \mid x, u) = N(Ax + Bu,\ \sigma_1^2\ I),$$
$$\mathrm{P}(z \mid x) = N(Cx,\ \sigma_2^2\ I),$$

then the Bayes Filter is equivalent to the Kalman filter. Assume that we can execute the Bayes filter for real-values states using the same equations.

Answer: True. The Kalman filter is a special case of the Bayes filter for systems with linear dynamics and Gaussian noise.

(2) **(2 points)** Let $u^{(k)}$ be the feedback control computed at iteration k of the Policy Iteration algorithm. If $u^{(k)}(x) = u^{(k+1)}(x)$ for all states $x \in X$ in the Markov Decision Process (MDP), we can safely terminate the algorithm and claim that $\mu^{(k)}$ is the optimal feedback controller.

Answer: True. If $u^{(k)}(x) = u^{(k+1)}(x)$ for all states x, policy iteration has converged and the controller will not change in the successive iterations. We can therefore safely terminate the algorithm. Policy iteration converges under very general assumptions; policy iteration converges if the class of policies over which we iterate upon is such that the system always reaches some terminal state after a finite number of time-steps.

(3) **(2 points)** Explain what will happen if there are two state trajectories that both have the maximal likelihood of generating the observations in Viterbi's algorithm. What will Viterbi's algorithm return and why.

Answer: Viterbi's algorithm computes the most likely trajectory starting from the first time-step, i.e., we initialize $\delta_1(x) = \pi_x M_{xy_1}$ and then update the variables $\delta_{k+1}(x)$ forwards. At the end of the algorithm, we backtrack the parent pointers

$$\hat{x}_t = \operatorname*{argmax}_{x'} \delta_t(x')$$

but the parent pointer is not unique in this problem. Since $\delta_t(x') = \delta_t(x'')$ for some two states x', x'' that correspond to the end states of the two trajectories respectively, the algorithm will therefore return whichever final state we select as a tie-breaker in the argmax.

(4) **(2 points)** If we use an Unscented Kalman filter (UKF) for performing filtering on a linear dynamical system with Gaussian noise

$$x_{k+1} = Ax_k + Bu_k + \epsilon_k$$

with linear observations of the state corrupted by Gaussian noise

$$y_k = Cx_k + \nu_k,$$

then the the mean of the UKF estimate $\mu_{k+1|k+1}$ has the same mean squared error

$$\underset{\epsilon_0,\ldots,\epsilon_{k-1},\nu_0,\ldots,\nu_k}{\mathrm{E}} \left[\left\| x_k - \mu_{k+1|k+1} \right\|^2 \right]$$

as that of the Kalman filter.

Answer: True because the Unscented Transform is exact for a linear transformation. If the original distribution is Gaussian, and it is transformed using a linear function, since the transformed distribution is also Gaussian, the UT will be able to transform these original sigma points exactly to their final locations.

(5) **(3 points)** Answer True or False with a 1-2 sentence explanation. The mean squared error of the Kalman filter can be improved if given access to future observations, i.e.,

$$\mathrm{E}\left[(\mu_k - x_k)^2 \mid y_0, \ldots, y_k \right] \geq \mathrm{E}\left[(\mu_k - x_k)^2 \mid y_0, \ldots, y_{k+1} \right];$$

here x_k is the true state of a linear dynamical system at time k, y_k is an observation linear in the state and μ_k is the mean of the Kalman filter's estimate of the state.

Answer: True. The Kalman filter estimate is the best estimate (smallest mean squared error estimate) given *all* past observations. This estimate can be improved if we know what occurred in the future, e.g., in the case of smoothing.

(6) **(3 points)** Answer in 1-2 sentences why we perform the resampling step in a particle filter.

Answer: Weights of the particles can become very small after the observation step if the innovation is large. This leads do a lot of redundant particles which do not have a large probability mass but are still used in the computation. Resampling step is used to prune these low-weight particles and sample more particles (these are clones of the existing particles) in regions of high probability mass.

(7) **(3 points)** Answer True or False with a 1-2 sentence explanation. The two unit quaternions $q = (u_0, u_1, u_2, u_3)$ and $q' = (-u_0, -u_1, -u_2, -u_3)$ parameterize the same 3D rotation.

Answer: True. Quaterions are a double cover of SO(3). The negative quaternion q' corresponds to the same rotation because it has an axis pointing in the opposite direction as that of q, and an angle that is also the negative of the angle of q.

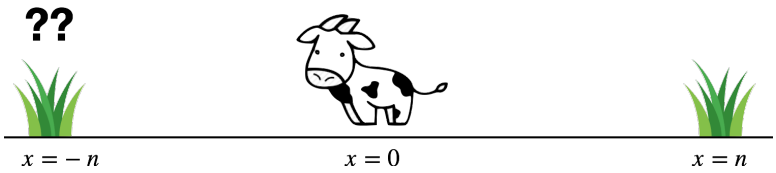
(8) **(3 points)** Suppose you are given a system with linear dynamics and linear observations but the noise in both the dynamics and observations is zero-mean but not Gaussian. Out of the four filtering algorithms that we know, namely the Kalman Filter, the Extended Kalman Filter, the Unscented Kalman Filter and the Particle Filter, which filtering algorithm will you pick? Explain your answer in 1-2 sentences.

Answer: We know that both the KF and EKF can only handle Gaussian noise. The UKF also handles only Gaussian noise because it its update equations are based on the same equations as those of the Kalman filter (in particular, the expression of the Kalman gain in 3.35 is optimal only if the matrix noise in observations is Gaussian). The PF on the other hand can handle any kind of noise, we can simply add the non-Gaussian dynamics noise when we propagate the particles forward in the dynamics step and use the appropriate non-Gaussian likelihood to compute $\mathrm{P}(y_{k+1} \mid x^{(i)}_{k+1|k})$.

(9) **(3 points)** Answer True or False with a 1-2 sentence explanation. In the forward Dijkstra's algorithm, just like we do for Value Iteration, we can initialize the dist variable for every node in the graph to zero; recall that the dist variable maintains the "best cost-to-come" to every node from the source node. Assume that the cost of every edge in the graph is non-negative and the graph is acyclic.

Answer: False. We should initialize the dist variable to infinity for Dijkstra's algorithm to work correctly. For instance, if dist is zero for some nodes, it means that there is a zero-cost path-to-come to that node from the source node.

(10) **(3 points)**



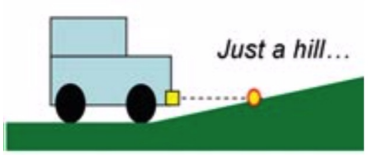
Answer in 1-2 sentences. Let us consider a classical problem in game theory known as the Cow-Path problem. A cow starts at the origin on the infinite real line. There is a patch of grass which the cow would like to get to, either at $x = n$ or at $x = -n$ with equal probability for some integer n that is not known to the cow. What is the optimal policy that the cow should use to find the patch of grass such that it has to travel the shortest distance in its search? *Hint: Think of what the optimal policy should be if the cow knew the value of n and then generalize.*

Answer: This is a trick question. As the hint suggests, if the value of n were known, the optimal policy is to first go to $x = n$ (or $x = -n$) and then go to $x = -n$ (or $x = n$) if there is no grass there. This suggests a good (not optimal!) strategy for the Cow-Path problem. The cow begins by moving 1 step either to the left or right. At each successive iteration, if it has not found the grass yet, it switches its direction of movement and doubles (or in general, increase by any factor $\alpha > 1$) the number of steps.

Notes You can prove that if $\alpha = 2$, the worst case number of steps walked by the cow is 9 n and this is the optimal deterministic policy. If the Cow flips a coin at each iteration and goes left or right depending upon its outcome, and also randomizes the amount it travels, the minimal number of steps is much smaller, it is about $4.9n$.

Problem 3 (24 points). Answer the following questions in 5-6 sentences each.

(1) **(8 points)** Imagine a robot that maintains a 2-dimensional occupancy grid to assimilate the observations from a planar LIDAR; see the picture below.



An upwards slope looks like an obstacle in this case and will prevent the robot from making any progress. Describe how you will modify the occupancy grid to tackle this situation.
Hint: Think about the angle at which you will mount the LIDAR. Can it be horizontal as shown in the picture?

Answer: The LIDAR should be mounted at a downward angle, this eliminates the blind spot at fixed height. Separate out the height from each return of the LIDAR and record this height in the cell in the occupancy grid. If the occupancy grid is denoted by $x \in \mathbb{R}^2$, let the height recorded for each cell be denoted by $h(x)$. We now have to find an algorithm that separates out distinct obstacles from slopes. This involves checking if the magnitude of the gradient $g(x) := \|\nabla_x h(x)\| \in \mathbb{R}$ is continuous at x. You can use the definition of continuity to ignore the height in the occupancy grid of all locations x such that

$$g(x + \epsilon e) \approx g(x)$$

for some small ϵ and chosen cardinal directions $e = [\pm 1, 0]$ and $e = [0, \pm 1]$.

(2) **(8 points)** How you would build a system with 10 drones that together count the number of apples in an orchard of area 10 square km. Focus on what you think are the most important parts of this system (the ones that you absolutely need to build); do not worry about details.

Answer: The most important features we need from the system are as follows.

(a) The 10 drones should be able to localize themselves in the orchard. 10 sq. km is a large area, they need to be able to transmit data back to the base station or to each other.

(b) Each drone should have a (relatively high-resolution) camera that can detect the fruit, count the number of fruits on a given tree. This can be done using a deep learning model to detect objects. But the more important point here is to be able to count *all* the apples on the tree. So we must be able to say that the apple detected in the previous frame is the same or different from the ones detected in the current frame from the camera. This can be done heuristically by comparing some local features of the apple, or by tracking the motion of the camera (using our applied control commands and rigid-body dynamics) to guess-estimate the location of the apple in the next frame.

(c) We need some algorithm to compute a good trajectory for each drone, e.g., raster scan the orchard to fly over the rows of the trees, and an algorithm to track this trajectory, e.g., LQR.

(d) We need an algorithm to split the orchard into 10 regions and have one drone operate in each region, this can be a Voronoi diagram or something much simpler like creating strips of land for each drone.

(3) **(8 points)** Consider a Hidden Markov Model (HMM) with a finite number of states and a finite number of observations. The transition probabilities $T_{x,x'} = \mathrm{P}(X_{k+1} = x' \mid X_k = x)$ and the observation model $M_{x,y} = \mathrm{P}(Y_k = y \mid X_k = x)$ are known to us along with the initial distribution of the states $\mathrm{P}(X_0 = x) = \pi(x)$. We are given a sequence of observations $(y_1, y_2, \ldots, y_T)$ from this system and would like to compute the most likely sequence of states $(\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_T)$ given these observations. Discuss how you will use dynamic programming to do so.

Answer: See Section 2.4.7 in the lecture notes.

Problem 4 (8 points). Consider a linear dynamical system given by

$$x_{k+1} = Ax_k + Bu_k.$$

We have observations of the states given by

$$y_k = Cx_k + \nu_k$$

where the noise ν_k is not white, it is correlated in time with correlations modeled as

$$\nu_k = D\nu_{k-1} + \xi_{k-1}.$$

where $\xi_k \sim N(0, \sigma^2 I)$ is zero-mean Gaussian noise with covariance $\sigma^2 I$ for all k. We would like to compute the estimate of the state at time k, denoted by $\hat{x}_k$, such that this estimate is unbiased, i.e.,

$$\mathop{\mathrm{E}}_{\nu_0,\ldots,\nu_k}\left[\hat{x}_k - x_k \mid y_0, y_1, \ldots, y_k\right] = 0$$

and it minimizes the variance

$$\mathop{\mathrm{E}}_{\nu_0,\ldots,\nu_k}\left[(x_k - \hat{x}_k)^2 \mid y_0, y_1, \ldots, y_k\right].$$

You do not need to derive the answer exactly, give a sketch of the solution procedure.

Answer: The observation noise ν_k is not white so we cannot use the Kalman filtering equations directly. However, we know how this noise changes in time so we can incorporate ν_k as a state of the system and define a new state

$$z_k = [x_k, \nu_k].$$

The new dynamics for z_k is given by

$$z_{k+1} = \underbrace{\begin{bmatrix} A & 0 \\ 0 & D \end{bmatrix}}_{A'} z_k + \underbrace{\begin{bmatrix} B \\ 0 \end{bmatrix}}_{B'} u_k + \begin{bmatrix} 0 \\ I \end{bmatrix} \xi_k.$$

The observations for this system are given by

$$y_k = \underbrace{\begin{bmatrix} C & I \end{bmatrix}}_{C'} z_k;$$

there is no noise in the observations, there is only noise in the dynamics. This is a linear dynamical system with additive Gaussian noise; the question asks for an unbiased estimator of the state x_k given past observations with the least variance which we know to be the result of the Kalman filter. We can now plug in the update equations of the Kalman filter in this setup.

$$\hat{z}_k = \left(I - L_k C'\right)\left(A' \hat{z}_{k-1} + B'\, u_k\right) + L_k y_k$$

$$\Sigma'_k = \left(A' \Sigma_{k-1}\, A'^{\top} + \begin{bmatrix} 0 \\ I \end{bmatrix} \begin{bmatrix} 0 & I \end{bmatrix}\right) \quad \text{(note the covariance of dynamics noise)}$$

$$\Sigma_k = \left(I - L_k C'\right)\, \Sigma'_k$$

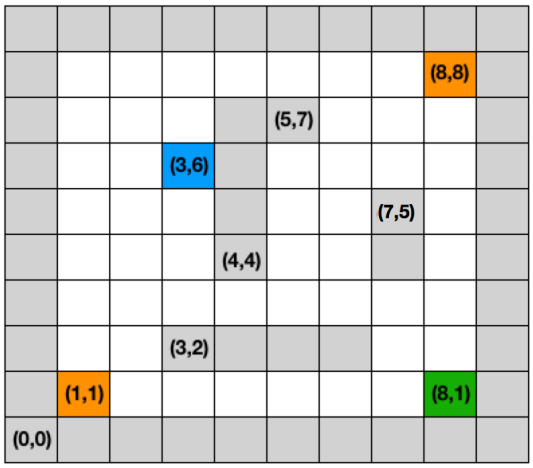
$$L_k = \Sigma'_k C'^{\top} \left(C' \Sigma'_k C'^{\top} + 0\right)^{-1} \quad (Q \text{ is zero}).$$

The estimate of $\hat{x}_k$ is finally given by

$$\hat{x}_k = \begin{bmatrix} I & 0 \end{bmatrix} \hat{z}_k.$$

Note: You can also define an alternative state $z_k := [x_k, \nu_{k-1}]$ in which case we will have noise in both the dynamics and observations. The problem with using this state is that the noise in the dynamics equation and observation equation will not be independent (it is the same ξ_{k-1}). The Kalman filter equations that we saw in the lectures require the dynamics and observation noise to be independent. Although, you can also derive slightly different KF equations for the case of correlated noise.

Problem 5 (6 points). Consider the following Markov Decision Process. The state-space is a 10×10 grid. The cells marked in gray are obstacles. The initial state is the cell marked in blue or orange and our desired terminal state is the one marked in green. A reward of 1 is collected when you reach the green cell and the discount factor is 0.9; there is no other reward or cost. At each non-obstacle cell, the agent can attempt to move to any of the neighboring cells. The move succeeds with probability 0.75. Otherwise, the agent moves to any of the other non-diagonal neighboring non-obstacle cells with equal probability. The agent can always stay put at a cell, this move succeeds with probability 1.



The numbers in some of the cells indicate their coordinates on a Cartersian grid, you may find them useful.

(1) What is the maximum expected sum of rewards of a trajectory obtained by the agent if it starts from the blue cell?

(2) Is the maximum expected reward of the orange cells the same? If yes, argue why it is. If not, argue why it is not.

You *do not* need to write any code to answer this question.

Answer:

1. The shortest path to reach the green cell from the blue cell has 10 steps. The reward is obtained after you reach the green cell, so in this case the maximum expected sum of rewards obtained is $\gamma^9 = 0.39$. Similarly the maximum possible expected sum of reward from the orange cells is $\gamma^6 = 0.53$.
2. If you solved for the expected reward using value iteration or policy iteration, you would have to write the code for either of them and obtain the solutions to be 0.1793, 0.2784, 0.2852 for the blue cell, bottom left orange cell, top right orange cell, respectively.
3. The optimal policy is quite intuitive in this case, it is simply the shortest path to the green cell from each of the cells. You can exploit this observation to first write down the policy and perform policy evaluation instead of coding up value/policy iteration.
4. If you solved for the maximum reward, the maximum reward of the orange cells are the same because they are equidistant from each cell. If you solved for the expected reward, the expected reward of the orange cells are not the same because different obstacles/asymmetry of the map results in different values. We graded this problem assuming that your taken approach is consistent.

END OF EXAM