ROB 501 - Mathematics for Robotics
HW #10

Due Nov. 22, 2021
3PM on Canvas

Remarks: In Problem 2, we start playing with the Kalman filter. Start Problem 2 early enough that you can seek MATLAB help in time!

1. Estimating derivatives of functions numerically. In this problem, we introduce a method to estimate the Jacobian of a function numerically.

   (a) Compute the Jacobian of the following function $f : \mathbb{R}^3 \to \mathbb{R}$ analytically and evaluate it at the point $x^* = [1\ 3\ -1]^\top$

   $$f(x_1, x_2, x_3) = 3x_1 \left[2x_2 - (x_3)^3\right] + \frac{(x_2)^4}{3}. \tag{1}$$

   Recall that the gradient of a scalar valued function is normally written as a row vector

   $$\frac{\partial f(x)}{\partial x} := [\frac{\partial f(x)}{\partial x_1} \quad \frac{\partial f(x)}{\partial x_2} \quad \frac{\partial f(x)}{\partial x_3}].$$

   (b) In calculus, you learned that $\frac{\partial f(x^*)}{\partial x_i} := \lim_{\delta \to 0} \frac{f(x^* + \delta e_i) - f(x^*)}{\delta}$, where e_i, $i = 1:3$ are the natural basis vectors, and thus, for a fixed $\delta > 0$, you might estimate the derivative by

   $$\frac{\partial f(x^*)}{\partial x_i} \approx \frac{f(x^* + \delta e_i) - f(x^*)}{\delta}$$

   It turns out that better numerical accuracy is usually obtained by a *symmetric difference*[1]

   $$\frac{\partial f(x^*)}{\partial x_i} \approx \frac{f(x^* + \delta e_i) - f(x^* - \delta e_i)}{2\delta}. \tag{2}$$

   Compute a numerical approximation to the Jacobian of the function (1) using symmetric differences and report the value(s) you used for δ. You are not obliged to use the same δ for each partial derivative. Use the same x^* as in part (a).

   (c) When you already know the answer, it is easy to play with δ and come up with a good numerical approximation to the derivative. What will you do when you do not know the answer before hand? Let's find out! Download the function `funcPartC.p` from the MATLAB folder. It is a hidden function $f : \mathbb{R}^5 \to \mathbb{R}$. Report its Jacobian at $x^* = [1, 1, 1, 1, 1]$.
   Here is how to call the function

   ```
   x0=[1 2 3 4 5];
   f=funcPartC(x0)
   f = 2.1281e+01
   ```

[1] It can be shown that a symmetric difference is EXACT for a quadratic function. You might want to try that out.

2. Download the file `HW10KFdata.zip` from the MATLAB folder on CANVAS, and unzip it. The file contains a discrete-time planar model of a Segway, some data, and a test file. In MATLAB, run the command

```
>> SegwayTest
```

You should first see a low-budget animation of a Segway, just to convince you that you are working with a physical system. If you want to know more about the model, read the file `Segway560.pdf` (it is contained in the zip file); this is optional. The state vector in the model consists of the angle of the Segway support bar with respect to the vertical, φ, the angle of the wheel with respect to the base, θ, and the corresponding velocities. Hence,

$$x = \begin{bmatrix} \varphi \\ \theta \\ \dot{\varphi} \\ \dot{\theta} \end{bmatrix}.$$

(a) Download the file `KalmanFilterDerivationUsingConditionalRVs` from the Handout folder on CANVAS. Using the data in the file `SegwayData4KF.mat`, implement the one-step Kalman filter on page 9 of the handout, for the model

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + Gw_k \\ y_k &= Cx_k + v_k. \end{aligned}$$

The model data is given to you

```
>> clear all
>> load  SegwayData4KF.mat
>> whos
```

In this example, the model matrices are constant: for all $k \geq 0$, $A_k = A$, $B_k = B$, $G_k = G$, $C_k = C$, and the noise covariance matrices are constant as well $R_k = R$, and $Q_k = Q$. The model comes from a linear approximation about the origin of the nonlinear Segway model. You can learn how to compute such approximations in EECS 562 (Nonlinear Control). A deterministic input sequence u_k is provided to excite the Segway and cause it to roll around. The measurement sequence y_k corresponds to the horizontal position of the base of the Segway. `x0` and `P0` are the mean and covariance of the initial condition x_0. The number of measurements is N.

(b) Run your Kalman filter using the data in `SegwayData4KF.mat`. Turn in the following plots:

- On one plot, $\widehat{\varphi}$ and $\widehat{\theta}$ versus time, t, or versus the time index k (either is fine).
- On a second plot, $\widehat{\dot{\varphi}}$ and $\widehat{\dot{\theta}}$ versus time, t or versus the time index k (either is fine).
- On a third plot, the four components of your Kalman gain K versus time, t, or versus the time index k (either is fine).
  Remark: $t(k) = kT_s$, where T_s is the sampling period.

(c) You should see the components of your Kalman gain K_k converging to constant values. Report these steady-state values. Then, execute the command below and compare Kss to your steady-state value of K.

```
[Kss,Pss] = dlqe(A,G,C,R,Q)
```

Using K_{ss} in place of K_k is called the steady-state Kalman filter. When the model matrices are time invariant, it is quite common to use the steady-state Kalman filter.

3. **One step update of a robot's state using a Kalman filter**. You are given a simple robot that moves in one dimension, say along the X-axis.

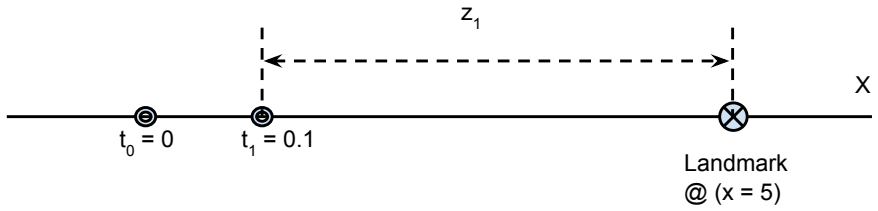

Figure 1: Robot's motion is along the X-axis. The landmark is a fixed point at $x = 5$m.

Your robot starts at a position[2] x_0, which is normally distributed $x_0 \sim \mathcal{N}(1, 0.25)$, i.e. $\mu_0 = 1$ and $\Sigma_0 = \sigma^2 = 0.25$. The state of your robot at the next time step is given by the following equation

$$x_1 = x_0 + u_1 \cdot \delta_t \quad \text{(state propagation model)} \tag{3}$$

where u_1, the action taken, is also normally distributed[3], $u_1 \sim \mathcal{N}(10, 16)$, i.e. $\hat{u}_1 = 10$, $R = 16$ and $\delta_t = 0.1$ is the time step.

Fortunately, your robot has a reasonably accurate time of flight sensor[4] which can measure the robot's distance from a fixed landmark. Based on the physics of the situation, you expect the output of your robot's sensor (i.e., the measurement) to be related to the robot's position by

$$\hat{z}_t = \frac{2}{c}(5 - x_t), \tag{4}$$

where x_t is the robot's position at time t, c is the speed of light[5] and $\hat{z}_t$ is the corresponding expected output[6](in seconds) from the sensor.

Because your sensor is a real device, it has error in its measurement; the actual sensor output is modeled to be, $z_1 \sim \mathcal{N}(\hat{z}_t, Q)$, where Q is the uncertainty in the measurement. In this case, the manufacturer tells you that your LiDAR has uncertainty, $Q = 10^{-18}$.

Problem: Suppose at time $t = 0.1s$ your robot takes action u_1 and moves according to (3). Your sensor outputs $z_1 = 2.2 \times 10^{-8}$s. Estimate the position x_1 of your robot at time $t = 0.1$ and the uncertainty in its position. Give your answer in the form of a normal distribution $x_1 \sim \mathcal{N}(\mu_1, \Sigma_1)$.

[2]We have used the notation, $x \sim \mathcal{N}(\mu, \Sigma)$, in previous HW sets.

[3]Why would the control signal be random? Well, what if the feet or the wheels of your robot are slipping in sand or wet grass? Then the control signal you command to the motor is not the action that will be applied to the body of the robot!

[4]Understanding how LiDAR works https://www.youtube.com/watch?v=EYbhNSUnIdU. More information about LiDARs specifically in mobile robots can be found here, https://www.clearpathrobotics.com/2015/04/robots-101-lasers/

[5]Use $c = 3 \times 10^8$ m/s

[6]The expected output, $\mathbb{E}\{z\} = \mu_z$, is the mean value

4. The following problem arises in camera calibration:

$$\widehat{x} = \arg \min_{x^\top x = 1} x^\top A^\top A x.$$

Show that if A is real[7], then $\widehat{x}$ is given by the last column of V where $A = U\Sigma V^\top$ is the SVD of A.

5. Compute a rank 2 approximation of

$$A = \begin{bmatrix} 4.041 & 7.046 & 3.014 \\ 10.045 & 17.032 & 7.027 \\ 16.006 & 27.005 & 11.048 \end{bmatrix}.$$

In particular, find ΔA of smallest norm such that $\widehat{A} := A + \Delta A$ has rank 2. The matrix norm being used is

$$||M||_2 = \sqrt{\lambda_{max}(M^\top M)}.$$

It is enough to give $\widehat{A}$, the rank 2 approximation, in your solution.

[7]When we write down `arg min`, we should also have conditions so that the answer is unique. For this problem, we need the smallest e-value of $A^\top A$ to be unique, and even then, it is only unique up to a sign (i.e., if $\widehat{x}$ is an answer, then so is $-\widehat{x}$). Unfortunately, in the literature, you typically see the problem given as stated.

Hints

Hints: Prob. 1 Oh, the age-old problem, how to tune parameters in algorithms? In this case, a *rule of thumb* is, if decreasing δ by a factor of 2 does not significantly change the estimate, then you can stop.

`Remark on exactness for a quadratic function:` Let $f(x) = ax^2 + bx + c$ be a quadratic. Performing the symmetric difference quotient about a point x^*, we have

$$\begin{aligned}
\frac{f(x^*+h) - f(x^*-h)}{2h} &= \frac{a(x^*+h)^2 + b(x^*+h) + c - (a(x^*-h)^2 + b(x^*-h) + c)}{2h} \\
&= \frac{ax^{*2} + 2ax^*h + ah^2 + bx^* + bh + c - ax^{*2} + 2ax^*h - ah^2 - bx^* + bh - c}{2h} \\
&= \frac{4ax^*h + 2bh}{2h} \\
&= \frac{2h(2ax^* + b)}{2h} \\
&= 2ax^* + b,
\end{aligned}$$

where we recognize the last line as the derivative at x^*.

Hints: Prob. 2

(a) The file `testSegway` illustrates how to simulate a deterministic discrete-time model using a `for loop`. While the physical model is assumed to be subjected to random perturbations, the noise terms themselves are <u>not</u> part of the Kalman filter: it uses the noise statistics, such as the covariance matrices. Hence, your implementation of the Kalman filter is a deterministic system and can be done in a manner similar to the `for loop` in the file `testSegway`.

(b) If you get stuck, it is OK to post MATLAB questions on Piazza.

Hints: Prob. 3 You have all the values for the Kalman filter available to you. Recall that you have seen in the handout on Gaussian Random Variables that if x_1 and x_2 are uncorrelated and $Y = Ax_1 + Bx_2$, then

$$\begin{aligned}
\mu_y &= A\mu_{x_1} + B\mu_{x_2} \\
\Sigma_y &= A\Sigma_{x_1}A^\top + B\Sigma_{x_2}B^\top
\end{aligned}$$

In this problem, you can see that $A = 1$ and $B = 0.1$, which will allow you to compute $\hat{x}_{1|0}$ and $P_{1|0}$. The rest of the data is available to you. Plug the values into the Kalman filter equations and note that you want to compute $\hat{x}_{1|1}$ and $P_{1|1}$, i.e. $x_1 \sim \mathcal{N}(\hat{x}_{1|1}, P_{1|1})$

Kalman Filter Equations with action model:
Prediction Step:

$$\begin{aligned}
\hat{x}_{k+1|k} &= A\hat{x}_{k|k} + B\hat{u}_1 \\
P_{k+1|k} &= AP_{k|k}A^\top + BRB^\top
\end{aligned}$$

Measurement Update Step:

$$\begin{aligned}
K_{k+1} &= P_{k+1|k}C^\top(CP_{k+1|k}C^\top + Q)^{-1} \\
\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + K_{k+1}(z_{k+1} - \hat{z}_{k+1|k}) \\
P_{k+1|k+1} &= P_{k+1|k} - K_{k+1}CP_{k+1|k}
\end{aligned}$$

$\hat{z}_{k+1|k}$ is the expected output if the position and measurement were not stochastic, i.e. using (4),

$$\hat{z}_{k+1|k} = \frac{2}{c}(5 - \hat{x}_{k+1|k})$$

Hints: Prob. 4 We have seen this problem before. Recall HW #2, Prob. 7-(b). How does the e-vector of $A^\top A$ corresponding to the minimum e-value relate to the columns of V? (See statement of SVD Theorem given in lecture).

Hints: Prob. 5 See the handout SVD_Rob501 in the resource folder...the one everyone slept through! If this were not very useful and practical stuff, I would not pester you about it. I know, about this point in the term, everyone is pretty tired.