Georgia Institute of Technology

ECE 6555 - Final take-home exam

Thursday, December 15, 2022 - v1.0

- The exam contains 12 pages (including the cover page)
- Each question is graded out of two points as follows: no credit without meaningful work, half credit for partial work, full credit if essentially correct.
- The exception to this rubric is the programming for the EKF in Q6 or Problem 2, which is worth 10 points
- Unless otherwise specified, you should concisely indicate your reasoning and show all relevant work.
- The grade on each problem is based on our judgment of your level of understanding as reflected by what you have written. If we can't read it, we can't grade it.
- Please use a pen and not a pencil.
- Because this is a take-home exam, there will be no tolerance for cheating. Communication with others is forbidden ("others" is including and not limited to other students, websites providing solutions, codes, solution manuals, etc.). You are allowed to use the Internet as long as this doesn't violate the previous clause.

Problem 1

Assume that $U \sim \mathcal{N}(0,1)$, i.e., U follows a normal distribution with zero mean and variance 1. We set $V = |U|$. Our goal is to find various estimators of U.

[Q1] Find the optimal estimator of V from U of the form αU, i.e, a linear estimator, for minimizing the mean-square error, and provide the corresponding mean square error.

[Q2] Find the optimal estimator of V from U of the form $\alpha + \beta U$, i.e, an affine estimator, for minimizing the mean-square error, and provide the corresponding mean square error.

[Q3] Find the optimal estimator of V from U of the form $\alpha + \beta U + \gamma U^2$, i.e, a quadratic estimator, for minimizing the mean-square error, and provide the corresponding mean square error.

(*Hint:* use the orthogonality principle to derive the values of the coefficients α, β, γ).
Important: You may not find nice closed form expressions. Provide numerical values instead.

Problem 2

The objective of this problem is to investigate a simple model of a non linear system, as well as implement various tracking techniques to compare their performance. Please pay attention to the following.

- You are expected to implement your *own* tracking algorithms (do *not* rely on readily available libraries).
- In addition to uploading your answer in gradescope, please send an email with a copy of your python code to matthieu.bloch@ece.gatech.edu in the form of a single `.py` file. If you like to code using a jupyter notebooks, make sure you copy your code into a single `.py` file and that the script executes properly.

We consider a pendulum model without friction as illustrated in the figure below. The state of the pendulum is described by the angle $\alpha(t)$ that the pendulum makes with the vertical axis at all times, as well as the angular velocity $\frac{d\alpha}{dt}$. The pendulum has a mass of $m = 3.5$ kg and the length of the arm is $\ell = 1$ m. It is subject to gravity as well as an unknown force, which we will model as noise.

The differential equation governing the pendulum motion is then

$$\frac{d^2\alpha}{dt^2} = -\frac{g}{\ell}\sin\alpha + w(t), \tag{1}$$

where $w(t)$ is the unknown force, modeled as a random process. Define the state of the system as

$$\mathbf{x} \triangleq \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \triangleq \begin{bmatrix} \alpha \\ \frac{d\alpha}{dt} \end{bmatrix} \tag{2}$$

[Q1] Show that the state space model is of the form

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f_1(x_2) \\ f_2(x_1) \end{bmatrix} + \mathbf{G}w(t) \tag{3}$$

where $f_1(\cdot)$ is a function of x_2 alone, f_2 is a function of x_1 alone, and $\mathbf{G} \in \mathbb{R}^{2\times 1}$. Provide *explicit* expressions for f_1, f_2, and $\mathbf{G}$.

We cannot track the angle α directly. Instead, we use a sensor that only tracks the horizontal position y of the pendulum subject to some additive noise $v(t)$

[Q2] Show that that measurement model is then given by

$$y = \ell\sin x_1 + v(t). \tag{4}$$

We need to discretize the continuous system in order to deploy our tracking algorithms. We employ a simple discretization of step Δt by which we identify

$$\frac{dx}{dt} = \frac{x(t+\Delta t) - x(t)}{\Delta t}. \tag{5}$$

The discretized model then follows by setting $\mathbf{x}_k \triangleq \mathbf{x}(k\Delta t)$ so that $\frac{dx}{dt}|_{t=k\Delta t} \approx \frac{x_{k+1}-x_k}{\Delta t}$

[Q3] Show that the discretized state space model takes the form

$$\begin{bmatrix} x_{1,k+1} \\ x_{2,k+1} \end{bmatrix} = \begin{bmatrix} x_{1,k} + x_{2,k}\Delta t \\ x_{2,k} - g\Delta t\sin(x_{1,k}) \end{bmatrix} + \mathbf{v}_{k+1} \tag{6}$$

$$y_{k+1} = \sin x_{1,k+1} + u_{k+1}. \tag{7}$$

If the noise $w(t)$ is white Gaussian, it can also be shown that the noise $\mathbf{q}_{k+1}$ is white Gaussian with covariance matrix

$$\mathbf{Q} \triangleq \sigma_p^2 \begin{bmatrix} \frac{\Delta t^3}{3} & \frac{\Delta t^2}{2} \\ \frac{\Delta t^2}{2} & \Delta t \end{bmatrix} \tag{8}$$

The noise u_{k+1} is assumed white Gaussian with variance σ_m^2.

We are faced with a non linear model of the form $\mathbf{x}_{k+1} = f(\mathbf{x}_x) + \mathbf{q}_k$ and $y_{k+1} = h(\mathbf{x}_{+1}k) + u_{k+1}$ for which we will have to use an EKF.

[Q4] Show that the linearization of the functions f and g around a state $x^* \triangleq [x_1^*, x_2^*]$ takes the form

$$f(\mathbf{x}) \approx f(\mathbf{x}^*) + \begin{bmatrix} 1 & \Delta t \\ -g\cos(x_1^*)\Delta t & 1 \end{bmatrix} (\mathbf{x} - \mathbf{x}^*) \tag{9}$$

$$g(\mathbf{x}) \approx g(\mathbf{x}^*) + \begin{bmatrix} \cos(x_1^*) & 0 \end{bmatrix} (\mathbf{x} - \mathbf{x}^*) \tag{10}$$

(*Hint:* recall that $f(\mathbf{x}) \approx f(\mathbf{x}^*) + \mathbf{F}(\mathbf{x} - \mathbf{x}^*)$ where $\mathbf{F}$ is a matrix whose entries are $F_{ij} \triangleq \frac{\partial f_i}{\partial x_j}$)

With all this in place, we are now in a position to implement an EKF. The parameters we will use are the following:

$$\sigma_p = 0.1 \quad \sigma_m = 0.3 \quad \Delta t = 20 \text{ ms} \tag{11}$$

You are provided with two files for your simulations.

- `groundtruth.npy` contains the true trajectory sampled at a high rate; the sampling period is 1 ms. Load both the sampling times and the trajectory as follows.

```
import numpy as np
tscale, x = np.load("groundtruth.npy")
```

- `measurements.npy` contains the noisy measurements on which you have to apply your filter; the sampling period is here 20 ms. Load both the sampling times and the measurements as follows.

```
tscale_measurement, y = np.load("measurements.npy")
```

[Q5] Write the equations of the EKF that you will implement. There is nothing to invent, we gave the equations in class, but I want you to tell me what you will be coding in our algorithm.

[Q6] Implement an EKF and report the root mean square error that you observed for the estimates after the measurement update. **Important:** You have full freedom to report your results. The quality of your plots *will* be taken into account when assigning points.

Bonus Implement a UKF and report the root mean square error. You may want to try using 2 or 4 sigma points.

Bonus Implement a particle filter and report the root mean square error.