Georgia Institute of Technology

ECE 6555 - Assignment 4

Thursday, November 10 2022 - v1.0

- There is 1 problem over 8 pages (including the cover page).
- Each question is graded as follows: no credit without meaningful work, half credit for partial work, full credit if essentially correct.
- Unless otherwise specified, you should concisely indicate your reasoning and show all relevant work.
- The grade on each question is based on our judgment of your level of understanding as reflected by what you have written. If we cannot read it, we cannot grade it.
- Please use a pen and not a pencil if you handwrite your solution.
- **You must submit your assignment on Gradescope**.

Problem 1: Kalman filter in action

The objective of this problem is to perform a more hands-on investigation of the Kalman filter to balance a bit the heavy theoretical content of the course and to study the *smoothing* Kalman filter. You are *allowed* and *encouraged* to use built-in python functions provided by `numpy` and `scipy` in the programming part, **but not other libraries**. The style of this problem is a bit unusual, but my goal is to make you appreciate what you have learned and what you can now do this semester, rather than drill you through yet another linear algebra problem.

State space model of a vehicle subject to unknown forces Consider a vehicle moving in a plane, which we model as a single point with known mass m for simplicity. The position of the vehicle is modeled as a time-varying vector

$$\mathbf{x}(t) \triangleq \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix} \in \mathbb{R}^2.$$

The vehicle is subject to an *unknown* force modeled as a time-varying vector

$$\mathbf{g}(t) \triangleq \begin{bmatrix} g_1(t) \\ g_2(t) \end{bmatrix} \in \mathbb{R}^2.$$

To obtain a state-space model, we start with Newtonian physics, which states that $\mathbf{g}(t) = m\dfrac{d^2\mathbf{x}(t)}{dt^2}$. We choose to model $\mathbf{g}(t)/m$ as a 2D Gaussian white noise process with autocorrelation function $\delta(t)$ (the Dirac delta-function), so that

$$\frac{d^2\mathbf{x}(t)}{dt^2} \triangleq \mathbf{w}(t) \triangleq \begin{bmatrix} w_1(t) \\ w_2(t) \end{bmatrix}.$$

We set $x_3(t) \triangleq \dfrac{dx_1(t)}{dt}$ and $x_4(t) \triangleq \dfrac{dx_2(t)}{dt}$ to consider the augmented state of the car

$$\mathbf{z}(t) \triangleq \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t). \end{bmatrix}$$

[Q1] Show that the evolution of the state $\mathbf{z}(t)$ s governed by the matrix differential equation

$$\frac{d\mathbf{z}(t)}{dt} = \mathbf{F}\mathbf{z}(t) + \mathbf{G}\mathbf{w}(t). \tag{1}$$

Specify clearly the matrices $\mathbf{F}$ and $\mathbf{G}$ (*Hint:* there are plenty of zeros).

The position of the vehicle is measured through a noisy sensor that outputs

$$\mathbf{y}(t) \triangleq \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} \in \mathbb{R}^2,$$

a corrupted version of the true vehicle position $\begin{bmatrix} x_1(t) & x_2(t) \end{bmatrix}^\intercal$ by a 2D Gaussian white noise process $\mathbf{v}(t)$ with autocorrelation function $\sigma^2\delta(t)$.

[Q2] Show that the measurement equation is given by

$$\mathbf{y}(t) = \mathbf{H}\mathbf{z}(t) + \mathbf{v}(t). \tag{2}$$

Clearly specify the matrix $\mathbf{H}$.

Since our simulations and our filters run in discrete time, we need sample the state of the vehicle with a period Δ. We therefore only consider the state of the vehicle a times $t_k \in \{k\Delta : k \in \mathbb{N}\}$. The discretization of the continuous-time state-space equation requires a bit of care to obtain a meaningful discrete time model. One can show that the solution of (1) satisfies for $k \geq 0$

$$\mathbf{z}(t_{k+1}) = \exp(\mathbf{F}\Delta)\mathbf{z}(t_k) + \underbrace{\int_0^\Delta \exp(\mathbf{F}(\Delta - \tau))\mathbf{G}\mathbf{w}(\tau + t_k)d\tau}_{\triangleq \mathbf{u}(t_k)}. \tag{3}$$

One can also show that the discrete-time process $\{\mathbf{u}(t_k)\}_{k\geq 0}$ is Gaussian and white with a covariance matrix $\mathbf{Q}$ that can be computed from $\mathbf{F}$, Δ, $\mathbf{G}$, and the autocorrelation function of $\mathbf{w}(t)$. For $\Delta \ll 1$, one can show that

$$\mathbf{Q} = \Delta \begin{bmatrix} \frac{\Delta^2}{3} & 0 & \frac{\Delta}{2} & 0 \\ 0 & \frac{\Delta^2}{3} & 0 & \frac{\Delta}{2} \\ \frac{\Delta}{2} & 0 & 1 & 0 \\ 0 & \frac{\Delta}{2} & 0 & 1 \end{bmatrix} \tag{4}$$

Showing (3) and (4) is not that difficult but this requires the careful analysis of differential equations with continuous time stochastic processes that we did no cover in class.

[Q3] Writing $\mathbf{x}_k \triangleq \mathbf{z}(t_k)$ $\mathbf{u}_k \triangleq \mathbf{u}(t_k)$ and using (3), show that the discretized state-space model when $\Delta \ll 1$ is

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & 0 & \Delta & 0 \\ 0 & 1 & 0 & \Delta \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + \mathbf{u}_k \tag{5}$$

where $\{\mathbf{u}_k\}$ is white with covariance $\mathbf{Q}$.

Converting the continuous time measurement equation is much easier since we just need to sample, so that the sensors measuring the vehicle position provide observations at times t_k given by

$$\mathbf{y}_k \triangleq \mathbf{y}(t_k) \triangleq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_k + \mathbf{v}_k \tag{6}$$

where $\{\mathbf{v}_k\}$ is white Gaussian with covariance $\mathbf{R} \triangleq \sigma^2 \mathbf{I}_2$, the identify matrix in $\mathbb{R}^{2\times 2}$. Equations (5) and (6) define the discrete-time state-space model that we use in the remaining of the exam. To simulate this experimentally, we will use Python together with the `numpy` and `scipy` libraries.

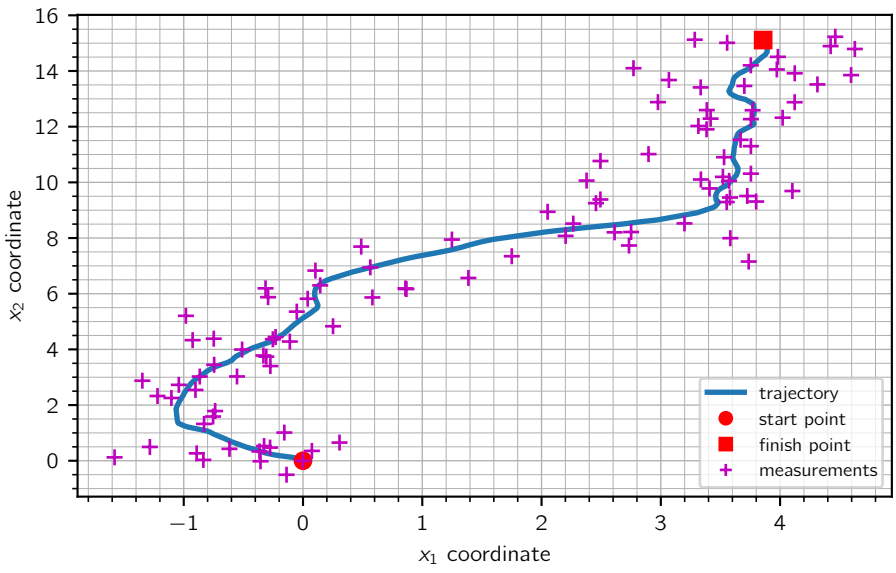
[Q4] Create a jupyter notebook to hold your code and initialize it as follows (do **not** change the seed).

```
import numpy as np
from scipy import signal

np.random.seed(1992)
NumSteps = 201
TimeScale = np.linspace(0,10,NumSteps)
```

```
DeltaSim = np.diff(TimeScale)[0]
SigmaInput = 1
SigmaNoise = 0.5

F = np.array([[1,0,DeltaSim,0],[0,1,0,DeltaSim],[0,0,1,0],[0,0,0,1]])
Q = SigmaInput**2 * np.array([[DeltaSim**3/3,0,DeltaSim**2/2,0],
                              [0,DeltaSim**3/3,0,DeltaSim**2/2],
                              [DeltaSim**2/2,0,DeltaSim,0],
                              [0,DeltaSim**2/2,0,DeltaSim]])
H = np.array([[1,0,0,0],[0,1,0,0]])
R = SigmaNoise**2 * np.identity(2)

State = np.zeros((4,NumSteps))
NoisyMeasurements = np.zeros((2,NumSteps))

for t in np.arange(1,NumSteps):
    ProcessNoise = np.squeeze(np.matmul(np.linalg.cholesky(Q),np.random.randn(4,1)))
    State[:,t] = np.matmul(F,State[:,t-1]) + ProcessNoise
    MeasurementNoise = SigmaNoise * np.squeeze(np.random.randn(2))
    NoisyMeasurements[:,t] = np.matmul(H,State[:,t]) + MeasurementNoise

StateX1 = State[0,:]
StateX2 = State[1,:]

DownSampling=2
NoisyMeasurements = NoisyMeasurements[:,::DownSampling]
MeasurementY1 = NoisyMeasurements[0,:]
MeasurementY2 = NoisyMeasurements[1,:]
```

Create and report a plot showing the trajectory `State` and the noisy measurements `NoisyMeasurements` in the 2D plane.

As an indication, your plot should look like this.



Kalman filtering The objective of this section is to implement the Kalman filter seen in class. No derivation is required here, but you have to make sure that you use and program the correct equations.

[Q5] Write the Kalman filter equations for the state-space model in (5)-(6). Use the prediction/update form seen in class.

[Q6] Implement the Kalman filter on the measurements `NoisyMeasurements`. You can reuse some of the code provided above to define the state-space model, but note that you will need to use the parameters $\Delta = 0.1$ in the covariance $\mathbf{Q}$, $\sigma = 0.5$ in the covariance $\mathbf{R}$. To ensure consistency, please initialize your Kalman filter as follows:

$$\hat{\mathbf{x}}_{0|-1} = 0 \qquad P_{0|-1} = \mathbf{I}_4$$

Be careful when implementing the filter, the most common mistake is to confuse indices and to not use the correct predictors in the recursive filter. Include your code below.

For our estimation, we use $\Delta = 0.1$, vs. the generation of data used 0.05, although we only sampled at 0.1 (why we will estimate at 0.1s to for our missed samples)

[Q7] Report the resulting trajectory by drawing it on top of the previous plot

[Q8] Numerically compute the rms error between the measurements and the true trajectory, and the filtered estimates and the true trajectory. In other words, compute

$$\sqrt{\sum_{i=1}^{n} \|\mathbf{y}_i - \mathbf{x}_i\|_2^2} \quad \text{and} \quad \sqrt{\sum_{i=1}^{n} \left\|\hat{\mathbf{x}}_{i|i-1} - \mathbf{x}_i\right\|_2^2}.$$

Which one is the largest? Is that expected?

Kalman smoother The Kalman filter is useful when one wants to predict a state trajectory $\{\hat{\mathbf{x}}_{i|i-1}\}$ in a online fashion, when measurements $\mathbf{y}_i$ arrive sequentially. However, there are situations in which the entire sequence of measurements $\{\mathbf{y}_i\}_{i=1}^n$ is available to predict the state at time i. In other words, we could try to form the estimate $\hat{x}_{0|1:n}$ of $\mathbf{x}_i$ based on *all* measurements. In this case, we talk about *smoothing* instead of filtering.

The objective of this section is to guide you through the derivation of the *Kalman smoother*, a recursive algorithm to compute the smoothed estimators, as well as to implement it. The proof will follow steps similar to those seen when deriving the Kalman filter from probabilistic state space models. In what follows, we consider a discrete-time probabilistic state-space model

$$\mathbf{x}_{i+1} \sim p(\mathbf{x}_{i+1}|\mathbf{x}_{0:i}, \mathbf{y}_{0:i}) \qquad \mathbf{y}_i \sim p(\mathbf{y}_i|\mathbf{x}_{0:i}\mathbf{y}_{0:i-1})$$

that is Markovian and satisfies the conditional independent property of measurements, i.e.,

$$p(\mathbf{x}_{i+1}|\mathbf{x}_{0:i}\mathbf{y}_{0:i}) = p(\mathbf{x}_{i+1}|\mathbf{x}_i) \qquad p(\mathbf{y}_i|\mathbf{x}_{0:i}\mathbf{y}_{0:i-1}) = p(\mathbf{y}_i|\mathbf{x}_i)$$

[Q9] Show that the state $\mathbf{x}_i$ is independent of $\mathbf{y}_{i+1:n}$ given $\mathbf{x}_{i+1}$, i.e.,

$$p(\mathbf{x}_i|\mathbf{x}_{i+1}, \mathbf{y}_{0:n}) = p(\mathbf{x}_i|\mathbf{x}_{i+1}, \mathbf{y}_{0:i})$$

You are encouraged to use a functional dependence graph as a proof, but you can also derive the result by manipulating the equations.

[Q10] Using Bayes' rule and the properties of the state space model show that

$$p(\mathbf{x}_i|\mathbf{x}_{i+1}, \mathbf{y}_{0:n}) = \frac{p(\mathbf{x}_{i+1}|\mathbf{x}_i)p(\mathbf{x}_i|\mathbf{y}_{0:i})}{p(\mathbf{x}_{i+1}|\mathbf{y}_{0:i})}$$

[Q11] Show that the joint distribution of $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ given $\mathbf{y}_{0:n}$ is given by

$$p(\mathbf{x}_i, \mathbf{x}_{i+1}|\mathbf{y}_{0:n}) = \frac{p(\mathbf{x}_{i+1}|\mathbf{x}_i)p(\mathbf{x}_i|\mathbf{y}_{0:i})p(\mathbf{x}_{i+1}|\mathbf{y}_{0:n})}{p(\mathbf{x}_{i+1}|\mathbf{y}_{0:i})}$$

These results suggest the following algorithm to compute the distribution of the smoothed estimate recursively. From the result above, taking the marginal yields that

$$p(\mathbf{x}_i|\mathbf{y}_{0:n}) = p(\mathbf{x}_i|\mathbf{y}_{0:i}) \int \left(\frac{p(\mathbf{x}_{i+1}|\mathbf{x}_i)p(\mathbf{x}_{i+1}|\mathbf{y}_{0:n})}{p(\mathbf{x}_{i+1}|\mathbf{y}_{0:i})} \right) d\mathbf{x}_{i+1}.$$

Note that the distributions $p(\mathbf{x}_i|\mathbf{y}_{0:i})$ and $p(\mathbf{x}_{i+1}|\mathbf{y}_{0:i})$ are those derived in class for the Bayesian optimal filtering. The equation tells us that we can compute $p(\mathbf{x}_i|\mathbf{y}_{0:n})$ from those distributions and $p(\mathbf{x}_{i+1}|\mathbf{y}_{0:n})$. In other words, we need to use the recursion of Bayesian optimal filtering and use a *backward* recursion to develop the smoothed estimates. Of course, these equations are not very useful unless we simplify them. *In the following, we consider a Gauss-Markov model for which all distributions are Gaussian.*

$$\mathbf{x}_{i+1} = \mathbf{F}\mathbf{x}_i + \mathbf{u}_i \qquad \mathbf{y}_i = \mathbf{H}\mathbf{x}_i + \mathbf{v}_i$$

where $\{\mathbf{u}_i\}$ and $\{\mathbf{v}_i\}$ are independent, Gaussian white noises with covariances $\mathbf{Q}$ and $\mathbf{R}$, respectively.

[Q12] Assume that $\mathbf{x}_{i-1}|\mathbf{y}_{0:i-1}$ is distributed according to a Gaussian distribution

$$\mathcal{N}\left(\hat{\mathbf{x}}_{i-1|0:i-1}, \mathbf{P}_{i-1|0:i-1}\right)$$

Reproduce the steps seen in class to show that the joint distribution of $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$ given $\mathbf{y}_{0:i-1}$ is

$$\mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{i-1|0:i-1} \\ \hat{\mathbf{x}}_{i|0:i-1} \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{i-1|0:i-1} & \mathbf{P}_{i-1|0:i-1}\mathbf{F}^\intercal \\ \mathbf{F}\mathbf{P}_{i-1|0:i-1} & \mathbf{P}_{i|0:i-1} \end{bmatrix} \right)$$

with $\hat{\mathbf{x}}_{i|0:i-1} = \mathbf{F}\hat{\mathbf{x}}_{i-1|0:i-1}$ and $\mathbf{P}_{i|0:i-1} = \mathbf{F}\mathbf{P}_{i-1|0:i-1}\mathbf{F}^\intercal + \mathbf{Q}$. (*Hint:* remember that this is an application of a lemma we proved in Lecture 28 and note that I have slightly changed the notation to use $\hat{\mathbf{x}}_{i-1|0:i-1}$ instead of $\hat{\mathbf{x}}_{i-1|i-1}$)

[Q13] Use the result of **[Q12]** to show that the distribution of $\mathbf{x}_{i-1}$ given $\mathbf{x}_i$ and $\mathbf{y}_{0:n}$, which is equal to the distribution of $\mathbf{x}_{i-1}$ given $\mathbf{x}_i$ and $\mathbf{y}_{0:i-1}$ by **[Q9]**, is

$$\mathcal{N}\left(\tilde{\mathbf{x}}_{i-1|i-1}, \tilde{\mathbf{P}}_{i-1|i-1}\right)$$

with $\tilde{\mathbf{x}}_{i-1|i-1} = \hat{\mathbf{x}}_{i-1|0:i-1} + \mathbf{K}_i(\mathbf{x}_i - \mathbf{F}\hat{\mathbf{x}}_{i-1|0:i-1})$ and $\tilde{\mathbf{P}}_{i-1|i-1} = \mathbf{P}_{i-1|0:i-1} - \mathbf{K}_i(\mathbf{F}\mathbf{P}_{i-1|0:i-1}\mathbf{F}^\intercal + \mathbf{Q})\mathbf{K}_i^\intercal$ and $\mathbf{K}_i = \mathbf{P}_{i-1|0:i-1}\mathbf{F}^\intercal(\mathbf{F}\mathbf{P}_{i-1|0:i-1}\mathbf{F}^\intercal + \mathbf{Q})^{-1}$. (*Hint:* apply a lemma we proved in Lecture 28)

If we assume that the distribution of $\mathbf{x}_i$ given $\mathbf{y}_{0:n}$ is $\mathcal{N}(\hat{\mathbf{x}}_{i|0:n}, \mathbf{P}_{i|0:n})$, one can show (painfully) that the joint distribution of $\mathbf{x}_i$ and $\mathbf{x}_{i-1}$ and $\mathbf{y}_{0:n}$ takes the form

$$\mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{i|0:n} \\ \hat{\mathbf{x}}_{i-1|0:i-1} + \mathbf{K}_i(\hat{\mathbf{x}}_{i|0:n} - \mathbf{F}\hat{\mathbf{x}}_{i-1|0:i-1}) \end{bmatrix}, \begin{bmatrix} \mathbf{P}_{i|0:n} & \mathbf{P}_{i|0:n}\mathbf{K}_i^\intercal \\ \mathbf{K}_i\mathbf{P}_{i|0:n} & \mathbf{K}_i\mathbf{P}_{i|0:n}\mathbf{K}_i^\intercal + \tilde{\mathbf{P}}_{i-1|i-1} \end{bmatrix} \right) \tag{7}$$

This shows that the conditional distribution $\hat{x}_{i-1|0:n}$ given $\mathbf{y}_{0:n}$ must therefore be $\mathcal{N}(\hat{\mathbf{x}}_{i-1|0:n}, \mathbf{P}_{i-1|0:n})$ with

$$\hat{\mathbf{x}}_{i-1|0:n} = \hat{\mathbf{x}}_{i-1|0:i-1} + \mathbf{K}_i(\hat{\mathbf{x}}_{i|0:n} - \mathbf{F}\hat{\mathbf{x}}_{i-1|0:i-1}) \qquad \mathbf{P}_{i-1|0:n} = \mathbf{K}_i\mathbf{P}_{i|0:n}\mathbf{K}_i^\intercal + \tilde{\mathbf{P}}_{i-1|i-1}$$

Putting everything together, this tells us that we can obtain the smoothing by running the Kalman filter to estimate $\hat{\mathbf{x}}_{i-1|0:i-1}$ and $\mathbf{P}_{i-1|0:i-1}$, and then proceed backwards from the last estimate to apply the correction $\hat{\mathbf{x}}_{i-1|0:n} = \hat{\mathbf{x}}_{i-1|0:i-1} + \mathbf{K}_i(\hat{\mathbf{x}}_{i|0:n} - \mathbf{F}\hat{\mathbf{x}}_{i-1|0:i-1})$.

[Q14] Implement the Kalman smoother for the state space model of the vehicle from Section **??**. You can ignore computing $P_{i-1:0:n}$. Show your code below

[Q15] Report the resulting trajectory by drawing it on top of the previous plot

[Q16] Numerically compute the rms error between the smoothed estimates and the true trajectory. In other words, compute

$$\sqrt{\sum_{i=1}^{n} \left\| \hat{\mathbf{x}}_{i|0:n} - \mathbf{x}_i \right\|_2^2}.$$

How does this compare to you previous rms for the filtered estimator?