HW 2: Kalman Filters

Matthew Hong

Instructions

- Complete all the questions, *including* the "Resources Consulted" question. We expect that most students will use about a paragraph, or a few bullet points, to answer that question, but you can go longer or shorter.
- Submit the PDF *and* the code separately on Gradescope (look for "HW2: PDF" and "HW2: Code").
- For the PDF: To make things simpler for us to grade / inspect, please answer the questions in order (resources consulted first, then question one, then two, etc.). There is no page limit, but (as a high-level piece of advice) avoid writing excessively long-winded solutions. Use LaTeX for this; you can build upon this file. If you do that, you can remove the text that describes the actual questions if you want, please just make it *really easy* for us to see which question you are answering.
- For the code: some questions have code, which you will need to write. **Please use Python 3** for your code. Just submit everything as a single `.zip` file. Please include a brief README or brief comments in the code that make it *very easy* for us to run.

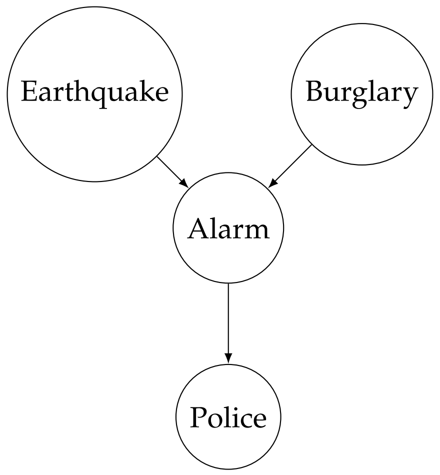
Resources Consulted

Question: Please describe which resources you used while working on the assignment. You do not need to cite anything directly part of the class (e.g., a lecture, the CSCI 545 course staff, or the readings from a particular lecture). Some examples of things that could be applicable to cite here are: (1) did you collaborate with a classmate; (2) did you use resources like Wikipedia, StackExchange, or Google Bard in any capacity; (3) did you use someone's code? When you write your answers, explain not only the resources you used but HOW you used them. If you believe you did not use anything worth citing, *you must still state that below in your answer* to get full credit.

Answer:
- Utilized ChatGPT to format plots and lookup matrix functions in numpy
- Utilized numpy documentation (https://numpy.org/doc/) to lookup matrix functions in numpy

1. Answer each of the following questions.

   (a) If an earthquake occurs, or there is a burglary, the alarm is likely to go off. If the alarm goes off, a police may arrive. Design a Bayesian network illustrating the causal relationships.



   (b) What happens if at any point in Bayesian filtering the probability of a state assignment becomes 1? What are ways to avoid that? Explain your answer in 2-3 sentences.

   If the probability of assigning a state reaches 1, then further beliefs will be guaranteed to be in that state, even if in reality that is not the case. One approach to prevent this issue is to prevent the assignment of a fully certain probability (0 or 1), always leaving a small margin for uncertainty, thereby allowing for the possibility that a state may not be entirely true.

   (c) Why do Extended Kalman Filters (EKFs) fail in handling multiple hypotheses? Explain your answer in 2-3 sentences.

   EKFs rely on Gaussians possessing unimodal distributions since linearization approximates the non-linear functions using a linear function tangent to the mean of the Gaussian. Simply using the arithmetic mean in a multi-modal Gaussian is not possible thus, EKFs cannot be used. Other methods, such as multi-hypothesis Kalman filters, can be used in place of the EKF.

2. We want to track the position of an object of unknown dynamics. We assume that the object moves in one dimension. A common model for unkown dynamics is the *constant jerk* model, that is assume that the acceleration is linear. The constant jerk

model can be represented as follows: $\mathbf{x}(t) = [p_t, v_t, a_t, j_t]^T$, with the elements being position, velocity, acceleration and jerk. Assuming discrete time-steps of $\Delta_t = 0.1$, the dynamics based on that model are:

$$\mathbf{x}(t+1) = A\mathbf{x}(t)$$
$$A = \begin{bmatrix} 1 & 0.1 & 0 & 0 \\ 0 & 1 & 0.1 & 0 \\ 0 & 0 & 1 & 0.1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

We additionally assume that we can measure the object's position with some noise:

$$\mathbf{z}(t) = C\mathbf{x}(t) + \mathbf{v}(t) \tag{1}$$
$$C = [1\ 0\ 0\ 0] \tag{2}$$

$\mathbf{v}(t)$ is a zero-mean Gaussian sensor noise with variance Q = 1.0.

$\boldsymbol{\mu}_0$ and Σ_0 represent the initial belief state of the Kalman Filter. We have that:

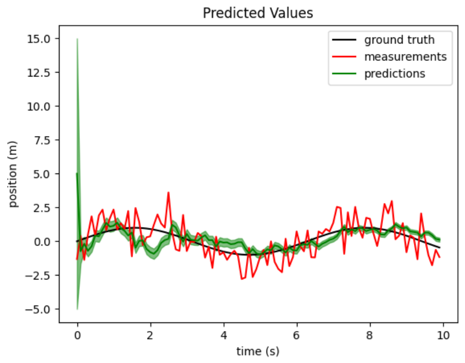
$$\boldsymbol{\mu}_0 = (5, 1, 0, 0), \qquad \Sigma_0 = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix}$$

The *true* position of the object changes as follows:

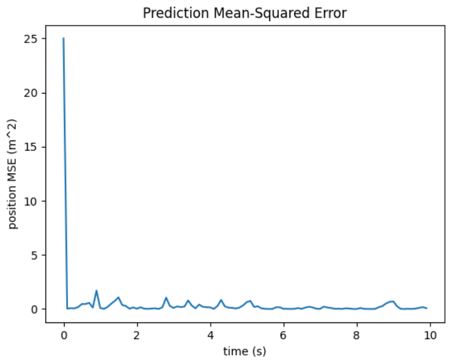
$$\mathbf{p}(t) = \sin(0.1 * t)$$

with $\mathbf{p}(0) = \mathbf{0}$. You must generate your own noisy sensor data $\tilde{\mathbf{p}}(t)$ by adding zero mean Gaussian noise, $\mathbf{v_m}(t)$, with variance $Q = 1.0$.

(a) Implement a Kalman Filter for $T = 100$ timesteps and plot how the error evolves over time. Compute the Mean Squared Error (MSE) of the position of the object, averaged over $N = 10000$ trials. *Include both of these figures in your PDF.*



(a) Prediction for noiseless system dynamics



(b) Mean Squared Error for noiseless system dynamics

Figure 1: Noiseless system dynamics

(b) To deal with model uncertainty, a technique frequently used is adding "fictitious" process noise. That is, assume that there is noise in the state dynamics. Rewrite your system dynamics as:

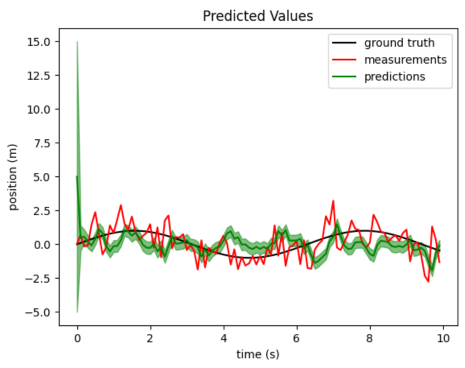
$$\mathbf{x}(t+1) = A\mathbf{x}(t) + \mathbf{w}(t)$$

Add fictitious noise in the form of $\mathbf{w}(t)$ in the dynamics, with zero mean and covariance[1]:

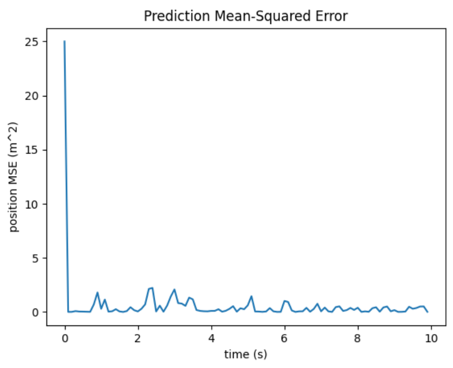
$$R = \begin{bmatrix} 0.1 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 \\ 0 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 0.1 \end{bmatrix}$$

Run the KF again and compare the MSE error with the absence of fictitious noise. *Save your figure and add it to the PDF.*

[1]Note that the discretized covariance matrix of the process noise has typically also non-diagonal elements, but we assume a diagonal matrix for simplicity



(a) Prediction with noise



(b) Mean Squared Error with noise

Figure 2: With noise, **w**(t)

3. Consider the following scalar system:

$$x(t+1) = \alpha x(t) + w(t)$$
$$z(t) = \sqrt{x(t)^2 + 1} + v(t)$$

$w(t)$ is zero-mean Gaussian process noise with variance R. $v(t)$ is zero-mean Gaussian sensor noise with variance Q.

(a) Write the equations for the Kalman filter to estimate the unknown constant α given $z(t)$. *Hint:* The state should be augmented with the unknown parameter α.

Rewriting the system dynamics to have α be apart of the state:

$$\begin{bmatrix} x(t+1) \\ \alpha \end{bmatrix} = \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x(t) \\ \alpha \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} w(t)$$

We take the Jacobian,

$$J = \frac{df}{dx(t)}$$

where,

$$f(x(t), \alpha = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} \alpha x(t) + w(t) \\ \alpha \end{bmatrix}$$

Jacobian, of our observation model, G, becomes,

$$G_t = \begin{bmatrix} \alpha & x(t) \\ 0 & 1 \end{bmatrix}$$

and $g(x_t)$ becomes,

$$g(x_t) \approx \begin{bmatrix} \alpha & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \mu_t \\ \alpha \end{bmatrix} + \begin{bmatrix} \alpha & x(t) \\ 0 & 1 \end{bmatrix} (x_t - \mu_t)$$

Doing the same for our sensor measurements, we take the Jacobian of our observation model, $z(t)$, resulting in,

$$H_t = \frac{\bar{\mu}_t}{\sqrt{\bar{\mu}_t^2 + 1}}$$

and finally our observation model, h, becomes,

$$h(x_t) \approx \sqrt{\bar{\mu}_t^2 + 1} + \frac{\bar{\mu}_t}{\sqrt{\bar{\mu}_t^2 + 1}} (x_t - \bar{\mu}_t)$$

(b) Using $Q = 1, R = 0.5$, write a script to test the algorithm in Python. Let *true* $x(0) = 2, \alpha = 0.1$. Assume initial estimates as follows, where $\hat{\alpha}$ is the initial estimate of α:

$$\mu_0 = 1 \text{ and } E[(x(0) - \mu_0)(x(0) - \mu_0)] = 2$$
$$\hat{\alpha} = 2 \text{ and } E[(x(0) - \hat{\alpha})(x(0) - \hat{\alpha})] = 2$$

How well does it work? Visualize the results for $T = 20$, then *save your figure and add it to the PDF.*

The Extended Kalman Filter, estimates the shape of x well, however, the α estimation does not converge to the true α.

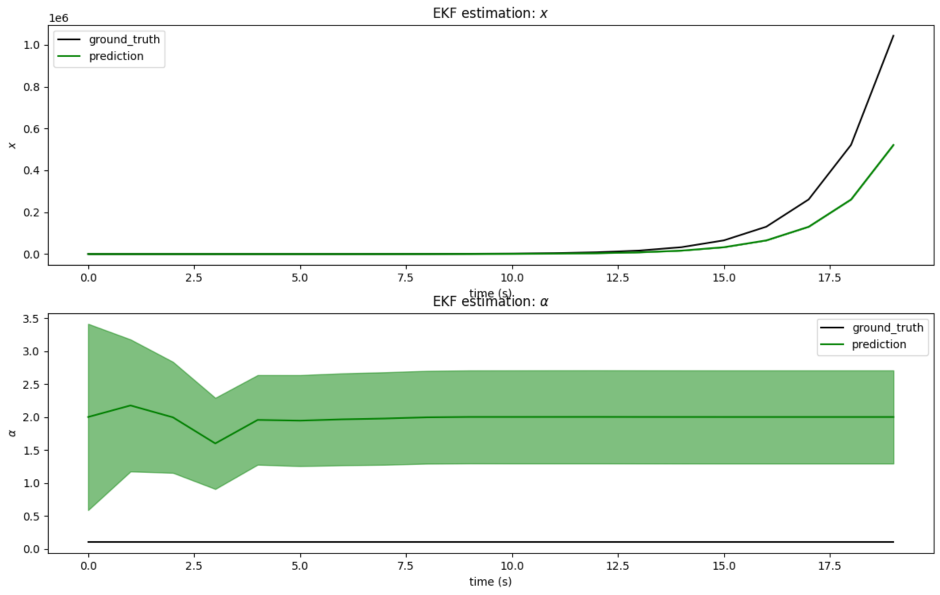


Figure 3: EKF Estimation of x and α