The Cooper Union Department of Electrical Engineering
Prof. Fred L. Fontaine
ECE416 Adaptive Algorithms
Project: Kalman Filters
November 2, 2023

Problem I: Basic Kalman Filter

The dynamical system is:

$$\begin{aligned} x_{n+1} &= Ax_n + v_n^{(x)} \\ y_n &= Cx_n + v_n^{(y)} \end{aligned}$$

where A, C can be time-varying. You are going to implement the Kalman filter in covariance form, in the Joseph form. Specifically, you should write a function that does one iteration, taking $K(n, n-1), x(n|n-1)$ and producing $K(n, n), K(n+1, n), x(n|n), x(n+1|n)$ as output.

There are two A matrices:

$$A_1 = \begin{bmatrix} -\frac{3}{2} & -\frac{5}{2} \\ \frac{1}{2} & \frac{1}{2} \end{bmatrix} \qquad A_2 = \begin{bmatrix} -3 & -5 \\ 1 & 1 \end{bmatrix}$$

and:

$$C = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

Also:

$$Q_x = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \qquad Q_y = 0.05$$

1. **Preliminary analysis:**

   Compute the eigenvalues of A_1, A_2. You will note in one case the system is stable, in the other it is unstable- which is which?

   Also compute the observability matrix $\mathcal{O}$ and find its rank to confirm each system is observable.

   Let K denote the steady-state prediction error covariance matrix, i.e., the limit of $K(n, n-1)$. The matrix K is found as the solution to the discrete time algebraic Riccati equation (DARE). The MATLAB function *idare* solves this. However, the notation MATLAB uses in the function differs from our notation here. The following table provides the "mapping" between the notations; note that the number of states is N, number of inputs is m (here 0) and number of outputs is p.

Kalman	*idare*
K	X
A'	A
$I_{N\times N}$	E
C'	B
Q_y	R
Q_x	Q
$0_{N\times p}$	S

   Call *idare* to compute K for each case. Later, you will be comparing your results to K.

2. **Kalman Filter**

We want to run the filter over 100 iterations. During the first 50, $1 \leq n \leq 50$, run it with the stable form A_1, then switch to A_2 for iterations $51 \leq n \leq 100$. Use initial condition $x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ to generate the state trajectory and output x_n, y_n for $1 \leq n \leq 100$. As you are recursively computing the trajectory, use initial conditions for the Kalman filter $\hat{x}(1|0) = x_0$ and $K(1,0) = 0.1I$ to run the Kalman filter over this time. As you run through the iterations, there are three quantities you should store:

- The actual state trajectory $\mathbf{x}$ and output y.
- The predicted state $\hat{x}(n|n-1)$ and estimated state $\hat{x}(n|n)$ for all n.
- You don't need to "keep" the full covariance matrices $K(n,n)$ or $K(n,n-1)$ across time. However, I would like you to compute and store:

$$\|K(n,n-1) - K_{\text{ideal}}(n)\|$$

  where $K_{\text{ideal}}(n)$ is the steady-state covariance matrix as determined by the algebraic Riccati equation. The reason I am labelling it with time n is that it takes on one value for the first 50 iterations (when $A = A_1$), and a different value for the remaining iterations (when $A = A_2$).

Then perform the following analysis of results. **Remark:** Be careful with indexing in MATLAB, which starts at 1.

(a) Let us write the state as $(x_1(n), x_2(n))$, and similarly for the predicted and estimated states: $(\hat{x}_1(n|n-1), \hat{x}_2(n|n-1))$, $(\hat{x}_1(n|n), \hat{x}_2(n|n))$. Superimpose plots of the prediction error and the estimation error, for each coordinate. That is, on one set of axes $x_1(n) - \hat{x}_1(n|n-1)$ and $x_1(n) - \hat{x}_1(n|n)$, and on separate axes $x_2(n) - \hat{x}_2(n|n-1)$, $x_2(n) - \hat{x}_2(n|n)$.

(b) Plot $\|K(n,n-1) - K_{\text{ideal}}\|$. How well do the results match, in general, and long does it take the Kalman filter to switch to the new solution after the transition point occurs?

(c) In your plot of how close $K(n,n-1)$ matches the value found from the Riccati equation, you should see some strong spikes (for the first few iterations, and again after the switch from A_1 to A_2 occurs). Excluding these spikes, look at how close they match. We can consider the algorithm to have "converged" after these spikes pass– for example, obtained zoomed in plots for the times where these seem to match well.

(d) Make some overall comments regarding the following issues: does the Kalman filter still work reasonably for the unstable system? does the Kalman filter handle the "switch" well?

Problem II: A Tracking Problem

We want to construct a "toy problem" for tracking applications of the Kalman filter. We start with a state trajectory described as a two-dimensional curve $(x_1(t), x_2(t))$ parametrized by continuous time. Our goal is to estimate the velocity along the trajectory from measurements of the position. We do this with the following strategy:

- Formulate a continuous-time dynamical system model where the state is comprised of both the position and velocity components.
- Obtain a suitable discretization of the model.
- Apply the Kalman filter to recover the velocity.

One could ask why a direct computation of the velocity could not be used, for example simply:

$$v_1(t) \approx \frac{1}{dt}\left(x_1(t+dt) - x_1(t)\right)$$

Here we are assuming the trajectory is the consequence of a dynamical system model that describes the physics of the moving object. The Kalman filter, in essence, uses this information to try to "fit" estimated velocity to the system dynamics, and in that sense does better than attempting a "blind" estimation of velocity from (noisy) measurements of position.
Let us start with the trajectory:

$$(x_1(t), x_2(t)) = \left(e^{\alpha t}\cos\left(\beta t^2\right), e^{\alpha t} sin\left(\beta t^2\right)\right)$$

The velocity components $(v_1(t), v_2(t))$ are such that:

$$\begin{aligned} \dot{x}_1 &= v_1 \\ \dot{x}_2 &= v_2 \end{aligned}$$

We define our state vector as:

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \\ v_1(t) \\ v_2(t) \end{bmatrix}$$

Our goal is to write:

$$\dot{x}(t) = A(t)\,\mathbf{x}(t)$$

for some matrix A. So far we have:

$$A(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \times & \times & \times & \times \\ \times & \times & \times & \times \end{bmatrix}$$

To complete the bottom two rows, we need to derive expressions for $\dot{v}_1, \dot{v}_2$ in terms of x_1, x_2, v_1, v_2. It is easier to express the trajectory in complex form:

$$x(t) = x_1(t) + jx_2(t) = e^{\alpha t + j\beta t^2}$$

Then:

$$v(t) = v_1(t) + jv_2(t) = (\alpha + j2\beta t)\, e^{\alpha t + j\beta t^2}$$

Then:

$$\begin{aligned} \dot{v}(t) &= \dot{v}_1(t) + j\dot{v}_2(t) \\ &= j2\beta e^{\alpha t + j\beta t^2} + (\alpha + j2\beta t)^2 e^{\alpha t + j\beta t^2} \\ &= j2\beta x(t) + (\alpha + j2\beta t)\, v(t) \end{aligned}$$

Matching real and imaginary parts gives us the final form:

$$A(t) = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\beta & \alpha & -2\beta t \\ \beta & 0 & 2\beta t & \alpha \end{bmatrix}$$

Also, by computing the position and velocity at $t = 0$ we get the initial state is:

$$\mathbf{x}(0) = \begin{bmatrix} 1 \\ 0 \\ \alpha \\ 0 \end{bmatrix}$$

The discrete-time model has the general form:

$$\begin{aligned} \mathbf{x}[n+1] &= A_d[n]\,\mathbf{x}[n] + e^{(x)}[n] \\ \mathbf{y}[n] &= C\mathbf{x}[n] + e^{(y)}[n] \end{aligned}$$

where $e^{(x)}$, $e^{(y)}$ are statistically white and Gaussian, independent of each other and the initial state $\mathbf{x}[0]$, with respective covariance matrices $\sigma_x^2 I$, $\sigma_y^2 I$. Also:

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

which means the output is a (noisy) measurement of the position. The Kalman filter estimates of the state variables can be viewed as serving a dual role– a reduced noise estimate of the position, and an estimate of the velocity (which is not readily available from our measurements).
In the experiment, set:

$$dt = 0.01$$

and we look at the trajectory for $0 \le t \le T$ where $T = 10$. That is:

$$t = 0 : dt : T$$

This gives $N = 1001$ points $0 \le n \le 1000$. Use the formula for A_d obtained via the midpoint method:

$$A_d[n] = I + A(t)\,dt + \frac{1}{2}A(t)\,(dt)^2 \qquad [t = n \cdot dt]$$

Other parameters:

- $\alpha = 0.1$
- $\beta = 0.02$
- $Q_x = \sigma_x^2 I$ where $\sigma_x^2 = 10^{-6}$
- $Q_y = \sigma_y^2 I$ where $\sigma_y^2 = 10^{-5}$

Initialize the Kalman filter algorithm with:

$$\begin{aligned} \hat{x}(1|0) &= \mathbf{x}[0] \\ K(1,0) &= 10Q_x \end{aligned}$$

In other words, we start with an exact match for the initial condition, and with a prediction covariance matrix that is somewhat large compared to the underlying process noise.
Run the Kalman filter over this time.

1. Superimpose graphs of the ideal position trajectory $(x_1(t), x_2(t))$ and the one computed via our discretized model with *zero disturbance.* This will give you a sense of how well the discretized model matches the continuous-time one. Compute the maximum error in each component that occurs at any point in the trajectory.
2. Repeat the above for the case of the velocity trajectory $(v_1(t), v_2(t))$.
3. Now compute and draw the trajectory $(x_1[n], x_2[n])$ and, on a separate graph, the velocity $(v_1[n], v_2[n])$, as computed by your discrete-time model **with** the random process disturbance in place.
4. On separate graphs, superimpose each of the state variables with their estimated values. That is superimpose a graph of $x_k[n]$ versus $\hat{x}_k(n|n)$ for $k = 1, 2$, and $v_k[n]$ versus $\hat{v}_k(n|n)$ for $k = 1, 2$. These will give you a visual sense as to how well the Kalman filter is performing the tracking task.
5. Examine the values of $K(n,n)$ at the 100^{th} iteration, and at the final iteration,. Display the matrices, and compute $\|K\|$ in each case.
6. The observability condition in the time-varying case is a bit more complicated, but assuming the system can be considered to be slowly varying relative to our time scale dt, which it is, it is reasonable to at least look at the observability matrix at time t, treating $A(t)$ as a constant. To that end:
   (a) Compute the matrix:
   $$\begin{bmatrix} C \\ CA \end{bmatrix}$$
   [You can do this very easily by hand!] This is not the full observability matrix (that would include CA^2, CA^3 blocks), but you should get something very "pretty" that confirms observability of the underlying continuous-time model.

(b) Now consider the discrete-time model. Using the symbolic toolbox in MATLAB. Define α, β, t, dt as symbolic variables, and with the A_d as defined above construct a symbolic representation for:

$$\begin{bmatrix} C \\ CA_d \end{bmatrix}$$

Again, this is only the upper 4×4 block of the observability matrix. Compute the (symbolic) determinant of this matrix and confirm if $\alpha > 0$ this cannot be 0, hence observability at least based on a quasi-time-invariant analysis is confirmed. [Another way to put this is, if not for the disturbances, the velocity could be recovered from the output within two time steps in the discrete-time model.