ELEC-E8107 - Stochastic models, estimation and control

Arto VISALA, and Issouf OUATTARA

December 2, 2022

Exercises Session 5: Solution

Exercise 1

A nonlinear system dynamic model of a robot moving on the plane is given by the following equation.

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t cos(\theta_k) \\ 0 & 1 & 0 & \Delta t sin(\theta_k) \\ 0 & 0 & 1 & \frac{\Delta t}{L} tan(\phi) \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ v_k \end{bmatrix} + q_k \tag{1}$$

Where v is the speed of the vehicle, θ is the heading and q_k is the process noise vector with covariance matrix Q_k. This covariance matrix can be assumed to be a diagonal matrix. The parameter ϕ is the steering angle and is considered a known input to the system. The constant parameter L is the distance between the front and back wheel of the robot. Here we assume $L = 15cm$.

Only the positions x and y of the robot are measured. The measurement noise is assumed Gaussian with zero mean and has 0.5 meters standard deviation. The measurement noise of x-axis and y-axis are assumed independent.

1. Write the measurement equation for the system.
2. Implement the bootstrap particle filter to estimate the state of the system.

Solution Exercise 1

See the second part of the solution of exercise session 4.

Exercise 2

The process dynamic of a system is given as;

$$\begin{aligned} x(k+1) &= 0.8x(k) + 0.3u(k) + v(k) \\ y_1(k) &= x(k) + w_1(k) \\ y_2(k) &= x(k-1) + w_2(k) \end{aligned}$$

Where,

$$\begin{aligned} E[v(k)v(k)^T] &= 0.01 \\ E[w_1(k)w_1(k)^T] &= 0.1 \\ E[w_2(k)w_2(k)^T] &= 0.01 \end{aligned}$$

Following measurements are made from the system;

k	1	2	3
y	10	-10	0
y_1	0.0	3.2	-0.8
y_2	-	0.0	3.0

The task is to devise a Matlab routine that calculates estimates for $x(k)$ using the data available at time k.

Solution Exercise 2

In order to compensate the delay in the measurement, an additional state is introduced to the system such that the new system is written as; The system is given as;

$$\begin{aligned} x_1(k+1) &= 0.8x_1(k) + 0.3u(k) + v_1(k) \\ x_2(k+1) &= x_1(k) + v_2(k) \\ y_1(k) &= x_1(k) + w_1(k) \\ y_2(k) &= x_2(k) + w_2(k) \end{aligned}$$

Where,

$$E[v_1(k)v_1(k)^T] = 0.01$$
$$E[v_2(k)v_2(k)^T] = 0$$
$$E[w_1(k)w_1(k)^T] = 0.1$$
$$E[w_2(k)w_2(k)^T] = 0.01$$

Here, $E[v_2(k)v_2(k)^T] = 0$ beacuse it is just an auxiliary state. Following Matlab code implements the Kalman Filter for the given settings;

```
%% Fixed-Lag Smoother
% The original system is:
% x(k+1) = 0.8*x(k) + 0.3*u(k)+ v(k)
% y1(k) = x(k) + w1(k)
% y2(k) = x(k-1) + w2(k)

% New system is
% x1(k+1) = 0.8*x1(k)+0.3*u(k)+v1(k)
% x2(k+1) = x1(k)+v2(k)
% y1(k) = x1(k)+w1(k)
% y2(k) = x2(k)+w2(k)

% Modified system matrices.
F = [0.8 0; 1 0];
H = [1 0; 0 1];
G = [0.3; 0];
Q = [0.01 0; 0 0];
R = [0.1 0; 0 0.01]

%% Kalman Filter
% Input and Measurements
u = [10 -10 0]
y = [0 3.2 -0.8 ; NaN 0 3]
% The initial conditions are:
x_p(:,1) = [0;0];
P(:,:,1) = [1 0; 0 1];

k = 1
% Calculate Innovation Covariance S(k+1).
S = H*P(:,:,k)*H'+R;
% Feed S(k+1) into Filter Gain W(k+1) Matrix.
W = P(:,:,k)*H'*(inv(S'));
% Updated state covariance
P_hat =P(:,:,k)-W*S*W';

% Second, update estiamtes of the states.
```

```
% Measurement predictions are made.
y_hat(:,k) = H*x_p(:,k);
v = zeros(size(y_hat))
% Measurement residual is computed for one measurement only.
v(1,k) = y(1,k)-y_hat(1,k);
% Update the state estimate to compensate measurements
    residuals.
x_hat = x_p(:,k)+W*v(:,k);

% Finally, prediction of the covariance matirx and state
    estimates
% when we have some measurements.

% State prediction step.
x_p(:,k+1) = F*x_hat+G*u(:,k);
% State prediction covariance is updated.
P(:,:,k+1) = F*P_hat*F'+Q;

for k=2:3
    % Copy the Kalman filter structure from P2!
    % First deal with the state covariance computation part.

    % Calculate Innovation Covariance S(k+1).
    S(:,:,k) = H*P(:,:,k)*H'+R;
    % Feed S(k+1) into Filter Gain W(k+1) Matrix.
    W(:,:,k) = P(:,:,k)*H'*(inv(S(:,:,k)'));
    % Updated state covariance
    P_hat(:,:,k) =P(:,:,k)-W(:,:,k)*S(:,:,k)*W(:,:,k)';

    % Second, update estiamtes of the states.

    % Measurement predictions are made.
    y_hat(:,k) = H*x_p(:,k);
    % Measurement residual is computed.
    v(:,k) = y(:,k)-y_hat(:,k);
    % Update the state estimate to compensate measurements
        residuals.
    x_hat(:,k) = x_p(:,k)+W(:,:,k)*v(:,k);

    % Finally, prediction of the covariance matirx and state
        estimates
    % when we have some measurements.

    % State prediction step.
    x_p(:,k+1) = F*x_hat(:,k)+G*u(:,k);
    % State prediction covariance is updated.
    P(:,:,k+1) = F*P_hat(:,:,k)*F'+Q;
end
% This Filter is actually so called "fixed-lag smoother"
```

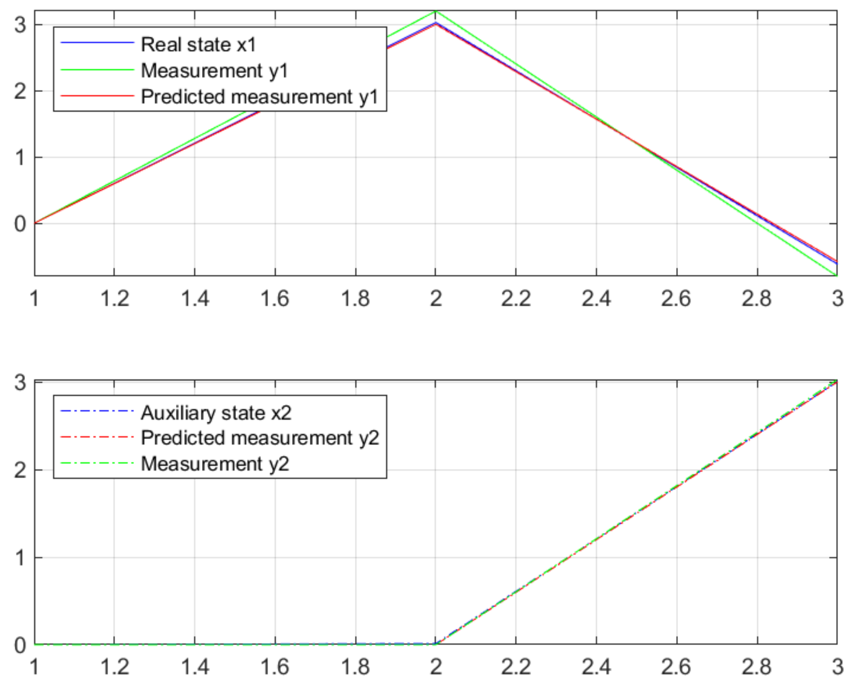

Figure 1: Evolution of the states with given measurements.

Note that due to the term $x(k-1)$ in the original system only one measurement y_1 is available for $k = 1$. So, the first cycle of KF has to be done using only one measurement. Such a filtering process is also know as **Fixed-Lag Smoother**.