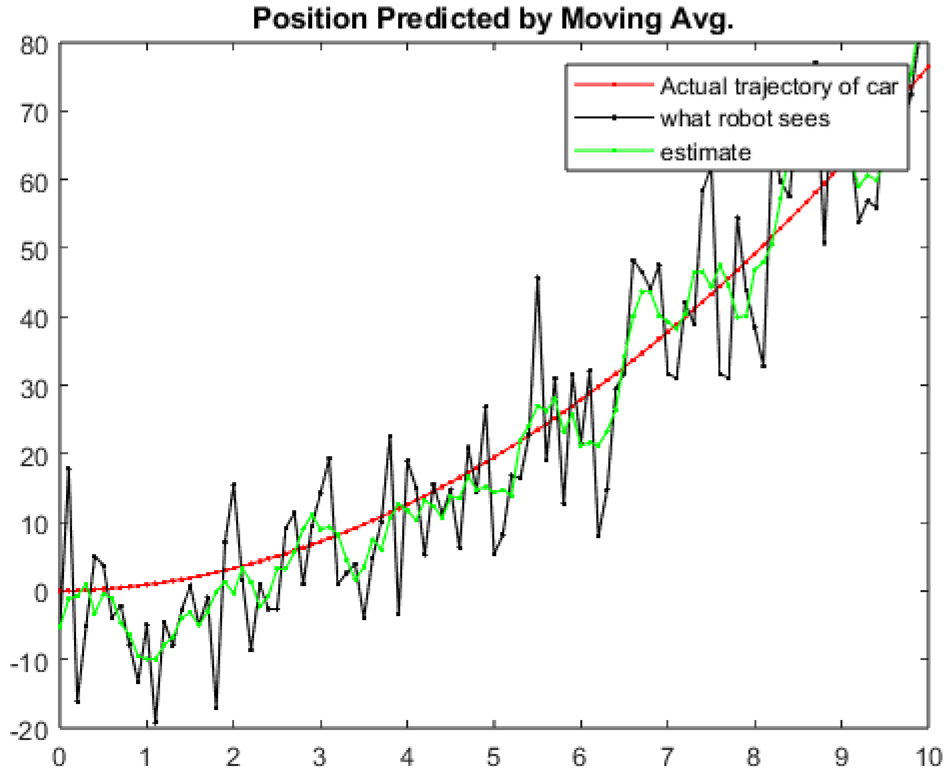
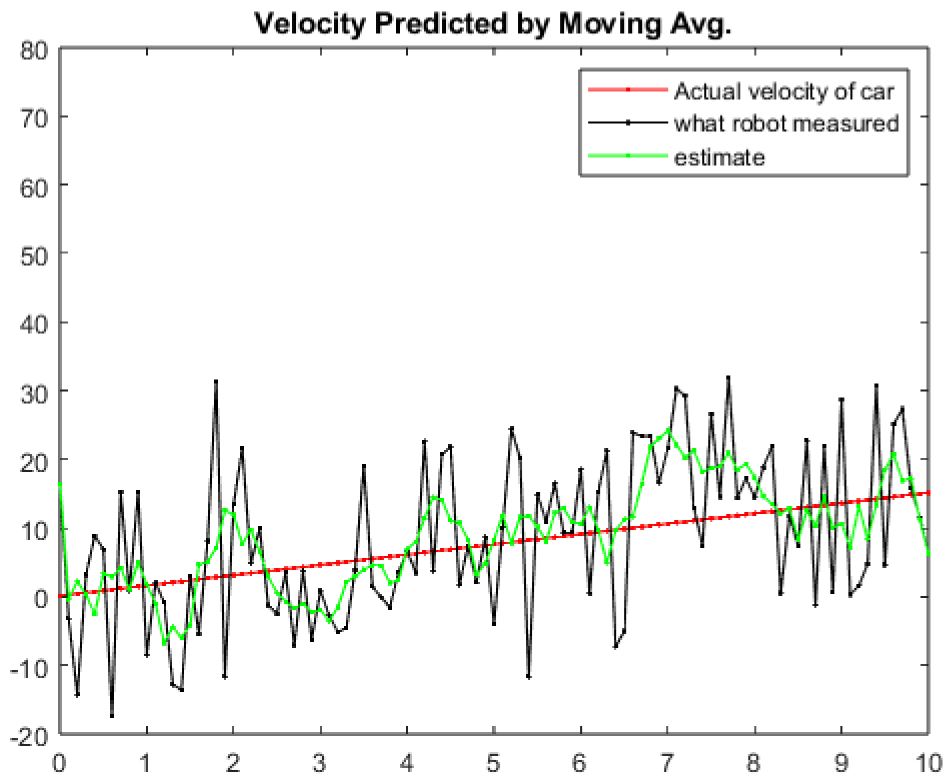
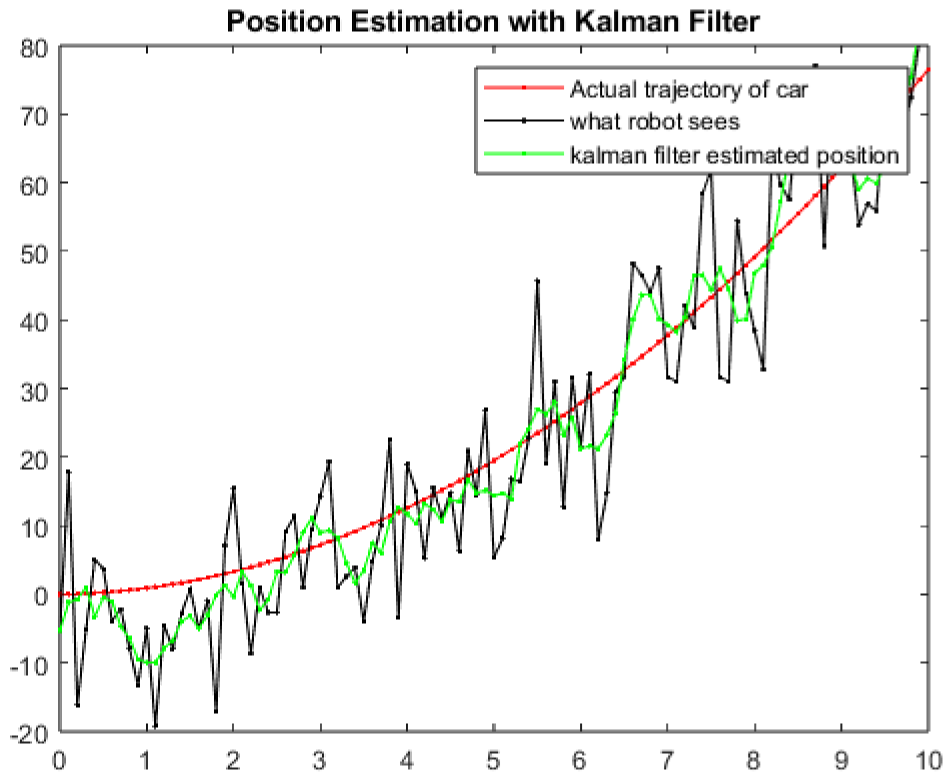
```
clear all;
close all;
clc;

% duration and how often we sample
duration = 10; %car ride duration
dt = 0.1;  % sampling distance

% Define update equations
Fk = [1 dt; 0 1] ; %State Transition Matrix
Bk = [dt^2/2; dt]; %Input Control Matrix
Hk = [1 0; 0 1]; % Measurement matrix for position and velocity

% main variables
u = 1.5; % acceleration mag
x= [0; 0]; %initial state vector, car has two components: [position; velocity]
xhat = x;  %initial state estimation of where the car is (what we are
 updating)
car_accel_noise_mag = 0.05; %process noise -standard deviation of acceleration
robot_noise_mag = .10;  %measurement noise -standard deviation of location
sigmaw = car_accel_noise_mag^2 * [dt^4/4 dt^3/2; dt^3/2 dt^2]; % Process noise
 covariance matrix
Rk = robot_noise_mag^2;% measurement noise covariance matrix
Pk = sigmaw; % initial estimation of car position covariance

% result variables
pos = []; % Actual car ride trajectory
vel = []; % Actual car velocity
Zk = []; % car trajectory that the robot sees (measured) robots perception
%Zk_pos = [];
vel_robot = []; % Velocity measured by the robot
vel_estimate = []; % Velocity estimate using Kalman filter

% simulate what robot sees over time
for t = 0 : dt: duration

    % Generate the car ride
    processNoise = car_accel_noise_mag * [(dt^2/2)*randn; dt*randn];
    x= Fk * x+ Bk * u + processNoise;
    % Generate what the robot sees
    measurementNoise = robot_noise_mag * randn (2,1)*100;
    y = Hk * x+ measurementNoise;
    pos = [pos; x(1)];
    Zk = [Zk; y(1)];
    %Zk_pos = [Zk_pos; y(1)];
    vel = [vel; x(2)];
    vel_robot = [vel_robot; y(2)];
end

% Plot the results
figure(1);
tt1=0:dt:t;
```

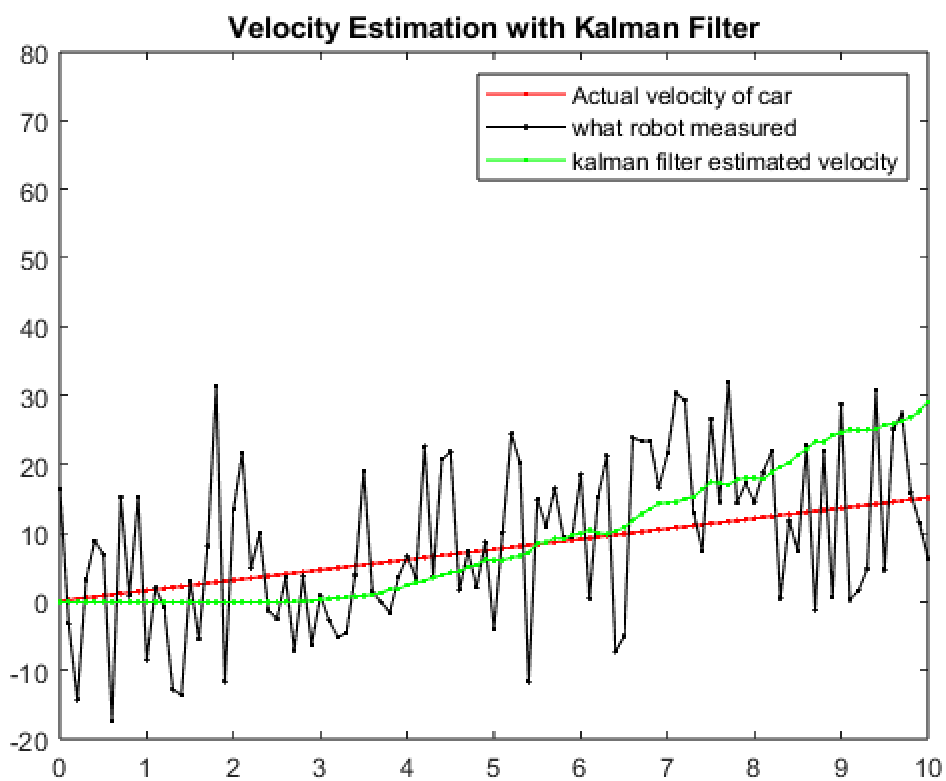
```
% Actual ride of car % what robot sees contineously %theoretical trajectory of
 robot that doesn't use kalman,but using moving average summing in window
plot(tt1, pos, '-r.',tt1, Zk, '-k.',tt1, smooth(Zk), '-g.'),title ('Position
 Predicted by Moving Avg.'),
axis([0 10 -20 80]),legend('Actual trajectory of car','what robot
 sees','estimate' );

% Plot the results
figure(2);
tt1=0:dt:t;
% Actual velocity of car % what robot measured contineously %theoretical
 trajectory of robot that doesn't use kalman,but using moving average summing
 in window
plot(tt1, vel, '-r.', tt1, vel_robot, '-k.',tt1, smooth(vel_robot), '-
g.'),title ('Velocity Predicted by Moving Avg.'),
axis([0 10 -20 80]),legend('Actual velocity of car','what robot
 measured','estimate' );


% using kalman filtering
% estimation variables
pos_estimate = []; % car position estimate
vel_estimate = []; % car velocity estimate
x= [0; 0]; % reinitialize the state

P_mag_estimate = [];
predict_state = [];
predict_var = [];
for t = 1:length(pos)
    % Predict next state of the car with the last state and predicted motion.
    xhat = Fk * xhat + Bk * u;
    predict_state = [predict_state; xhat(1)] ;

    %predict next covariance
    Pk = Fk * Pk * Fk' + sigmaw;
    predict_var = [predict_var; Pk] ;

    % predicted robot measurement covariance
    % Kalman Gain
    K = Pk*Hk'*inv(Hk*Pk*Hk'+Rk);
    % Update the state estimate.
    xhat = xhat + K * (Zk(t) - Hk * xhat);
    % update covariance estimation.
    Pk =  (eye(2)-K*Hk)*Pk;

    %Store result for plotting
    pos_estimate = [pos_estimate; xhat(1)];
    vel_estimate = [vel_estimate; xhat(2)];
    P_mag_estimate = [P_mag_estimate; Pk(1)];
end

% Plot the results
figure(3);
tt2 = 0 : dt : duration;
```

```
plot(tt2,pos,'-r.',tt2,Zk,'-k.', tt2,smooth(Zk),'-g.'),title ('Position
 Estimation with Kalman Filter'),
axis([0 10 -20 80]),legend('Actual trajectory of car','what robot
 sees','kalman filter estimated position' );

figure(4);
tt2 = 0 : dt : duration;

plot(tt2,vel,'-r.',tt2,vel_robot,'-k.', tt2,vel_estimate,'-g.'),title
 ('Velocity Estimation with Kalman Filter'),
axis([0 10 -20 80]),legend('Actual velocity of car','what robot
 measured','kalman filter estimated velocity' );


% %plot the evolution of the distributions
% figure(3);
% for T = 1:length(pos_estimate)
% clf
%     x = pos_estimate(T)-5:.01:pos_estimate(T)+5; %  x axis range
%
%     %predicted next position of the car
%     hold on
%     mu = predict_state(T); % mean
%     sigma = predict_var(T); % standard deviation
%     y = normpdf(x,mu,sigma); % pdf
%     y = y/(max(y));
%     hl = line(x,y,'Color','m');
%
%     %data measured by the robot
%     mu = Zk(T); % mean
%     sigma = robot_noise_mag; % standard deviation
%     y = normpdf(x,mu,sigma); % pdf
%     y = y/(max(y));
%     hl = line(x,y,'Color','k'); % or use hold on and normal plot
%
%     %combined position estimate
%     mu = pos_estimate(T); % mean
%     sigma = P_mag_estimate(T); % standard deviation
%     y = normpdf(x,mu,sigma); % pdf
%     y = y/(max(y));
%     hl = line(x,y, 'Color','g');
%     axis([pos_estimate(T)-5 pos_estimate(T)+5 0 1]);
%
%
%     %actual position of the car
%     plot(pos(T));
%     ylim=get(gca,'ylim');
%     line([pos(T);pos(T)],ylim.','linewidth',2,'color','b');
%     legend('state predicted','measurement','state estimate','actual car
 position')
%     % pause
% end
```


Position Predicted by Moving Avg.
Actual trajectory of car
what robot sees
estimate
80
70
60
50
40
30
20
10
0
-10
-20
0
1
2
3
4
5
6
7
8
9
10


Velocity Predicted by Moving Avg.
Actual velocity of car
what robot measured
estimate
80
70
60
50
40
30
20
10
0
-10
-20
0
1
2
3
4
5
6
7
8
9
10


Position Estimation with Kalman Filter
80
70
60
50
40
30
20
10
0
-10
-20
0
1
2
3
4
5
6
7
8
9
10
Actual trajectory of car
what robot sees
kalman filter estimated position


Velocity Estimation with Kalman Filter
80
70
60
50
40
30
20
10
0
-10
-20
0
1
2
3
4
5
6
7
8
9
10
Actual velocity of car
what robot measured
kalman filter estimated velocity

Published with MATLAB® R2023a

Ex-6

Task 6.2

a) $m_Z^{(1)} = 0, \quad \sigma_Z = 1$

$$S_{ZZ}(\tau) = a e^{-\alpha|\tau|} + b$$

$$m^{(1)}(t) = \sqrt{\lim_{\tau \to \infty} a e^{-\alpha|\tau|} + b} = 0$$

$$0 + b = 0$$

$$\boxed{b = 0}$$

$$S_{ZZ}(0) = Var(Z)$$

$$\lim_{\tau \to 0} S_{xx}(\tau) = 1$$

$$\lim_{\tau \to 0} a e^{-\alpha|\tau|} + b = 1$$

$$a + b = 1$$

$$\because b = 0 \quad \boxed{a = 1}$$

b) $S_{Zy}(t_1, t_2) = E\{Z(\zeta, t_1)\, y(\zeta, t_2)\}$

$$= E\left\{Z(\zeta, t_1)\int_{t_0}^{t_2} \cancel{z}(\zeta, \lambda)\, d\lambda\right\}$$

$$= \int_{t_0}^{t_2} E\{Z(\zeta, t_1)\, Z(\zeta, \lambda)\, d\lambda\}$$

For a stationary process

$$= \int_{t_0}^{t_2} S_{ZZ}(\lambda - t_1)\, d\lambda$$

$$= \int_{t_0}^{t_2} \left(a e^{-\alpha|\lambda - t_1|} + b\right) d\lambda$$

$\boxed{a = 1 \text{ and } b = 0}$

$$= \int_{t_0}^{t_2} e^{-\alpha|\lambda - t_1|}\, d\lambda$$

Case 1: $t_0 \le t_1 \le t_2$

$$S_{xy}(t_1,t_2) = \int_{t_0}^{t_1} e^{-\alpha|\lambda - t_1|}\,d\lambda + \int_{t_1}^{t_2} e^{-\alpha|\lambda - t_1|}\,d\lambda$$

For $t_0 < \lambda < t_1 \Rightarrow |\lambda - t_1| = -(\lambda - t_1)$

& $t_1 < \lambda < t_2 \Rightarrow |\lambda - t_1| = (\lambda - t_1)$

$$\therefore S_{xy}(t_1,t_2) = \int_{t_0}^{t_1} e^{\alpha(\lambda - t_1)}\,d\lambda + \int_{t_1}^{t_2} e^{-\alpha(\lambda - t_1)}\,d\lambda$$

$$S_{xy}(t_1,t_2) = \left[\frac{e^{\alpha(\lambda - t_1)}}{\alpha}\right]_{t_0}^{t_1} + \left[\frac{e^{-\alpha(\lambda - t_1)}}{-\alpha}\right]_{t_1}^{t_2}$$

$$= \frac{1}{\alpha} - \frac{e^{\alpha(t_0 - t_1)}}{\alpha} + \frac{e^{-\alpha(t_2 - t_1)}}{(-\alpha)} + \frac{1}{\alpha}$$

$$= \frac{2}{\alpha} - \frac{e^{-\alpha(t_1 - t_0)}}{\alpha} - \frac{e^{-\alpha(t_2 - t_1)}}{\alpha}$$

$$\boxed{S_{xy}(t_1,t_2) = \frac{1}{\alpha}\left(2 - e^{-\alpha(t_1 - t_0)} - e^{-\alpha(t_2 - t_1)}\right)}$$

Case 2: $t_0 \le t_2 \le t_1$

$$S_{xy}(t_1,t_2) = \int_{t_0}^{t_2} e^{-\alpha|\lambda - t_1|}\,d\lambda \cancel{+ \int_{t_2}^{t_1} e^{-\alpha|\lambda - t_1|}\,d\lambda}$$

$$= \int_{t_0}^{t_2} e^{-\alpha(t_1 - \lambda)}\,d\lambda \cancel{+ \int_{t_2}^{t_1} e^{-\alpha(t_1-\lambda)}\,d\lambda}$$

$$= \left[\frac{e^{-\alpha(t_1 - \lambda)}}{\alpha}\right]_{t_0}^{t_2} \cancel{+ \left[\frac{e^{-\alpha(t_1-\lambda)}}{\alpha}\right]_{t_2}^{t_1}}$$

$$= \cancel{\cdots} - \frac{e^{-\alpha(t_1-t_0)}}{\alpha} + \frac{e^{-\alpha(t_1-t_2)}}{\alpha}$$

$$\boxed{S_{xy}(t_1,t_2) = \frac{1}{\alpha}\left(e^{-\alpha(t_1-t_2)} - e^{-\alpha(t_1-t_0)}\right)}$$

Case 3: $t_1 \le t_0 \le t_2$

$$S_{xy}(t_1,t_2) = \int_{t_0}^{t_2} e^{-\alpha|\lambda - t_1|}\, d\lambda$$

$$= \int_{t_0}^{t_2} e^{-\alpha(\lambda - t_1)}\, d\lambda$$

$$= \left[\frac{e^{-\alpha(\lambda-t_1)}}{-\alpha}\right]_{t_0}^{t_2}$$

$$= \left[\frac{e^{-\alpha(t_2-t_1)}}{-\alpha} + \frac{e^{-\alpha(t_0-t_1)}}{\alpha}\right]$$

$$S_{xy}(t_1,t_2) = \frac{1}{\alpha}\left[e^{-\alpha(t_0-t_1)} - e^{-\alpha(t_2-t_1)}\right]$$

$$\therefore S_{xy}(t_1,t_2) = \begin{cases} \frac{1}{\alpha}\left[2 - e^{-\alpha(t_1-t_0)} - e^{-\alpha(t_2-t_1)}\right] & \text{for } t_0 \le t_1 \le t_2 \\ \frac{1}{\alpha}\left[e^{-\alpha(t_1-t_2)} - e^{-\alpha(t_1-t_0)}\right] & \text{for } t_0 \le t_2 \le t_1 \\ \frac{1}{\alpha}\left[e^{-\alpha(t_0-t_1)} - e^{-\alpha(t_2-t_1)}\right] & \text{for } t_1 \le t_0 \le t_2 \\ 0 & \text{elsewhere} \end{cases}$$

Task 6.3

a)

Random Process

$x(e,t)$

Pattern function

$x(e_i,t)$

Random variable

$x(e,t_i)$

Process Variable

$x(e_i,t_i)$

Relation in a Random process where

$e_i, t_i \rightarrow$ fixed

$e, t \rightarrow$ variable

Random process $\rightarrow$ A random process refers to a collection of random variables indexed by time or another parameter. It represents a set of outcomes or values that can occur over time or in a sequence.

Random variable $\rightarrow$ Random variable depends on a random elementary event e and whose value can be determined by the outcome of a random experiment

Pattern f^n $\rightarrow$ Refers to a deterministic function or deterministic process, which represents a predictable or deterministic relationship between variables or quantities.

Process variable → is a quantity that characteristic the state or behaviour of a physical process.

b) The fundamental prerequisite for the construction of matched filter is the availability or prior knowledge of the expected signal waveform in order to construct a filter that can effectively discriminate the desired signal from noise and interference.

c) If the shape of the transmitted signal is a rectangular impulse and a perfectly working matched filter is used, the output of the filter will be a scaled and time shifted version of the rectangular impulse.

d) As the signal-to-noise ratio increases, the bit error rate decreases in a matched filter receiver

e) The weiner - Hopf equation relates the autocorrelation f^n of the input and output signals of a LTI system to the system's transfer f^n.

$$\boxed{R_y(h) = R_x(h) * H(h)}$$

$R_y(h)$ → Auto correlation f^n of the output signal
$R_x(h)$ ⇒ Auto correlation f^n of the input signal
$H(h)$ → filter Transfer f^n

In frequency domain, the eq^n can be represented as

$$\boxed{S_{xy}(f) = S_y(f) * H(f)}$$

$S_{xy}(f)$: cross power spectral density between two signal $x(t)$ and $y(t)$ at frequency f

$S_y(f)$: Power spectral density of the signal $y(t)$.

$H(f)$: represents the frequency response of the system or filter.

f) A casual wiener kolmogorov filter should be used if the input signal is casual and the output signal is required to be casual. An acausal Wiener-kolmogorov filter should be used if the input signal is not causal or if the output signal does not need to be casual.