Human Neuromechanical Control and Learning

Tutorial 5: Sensory Prediction – Linear Quadratic Estimator

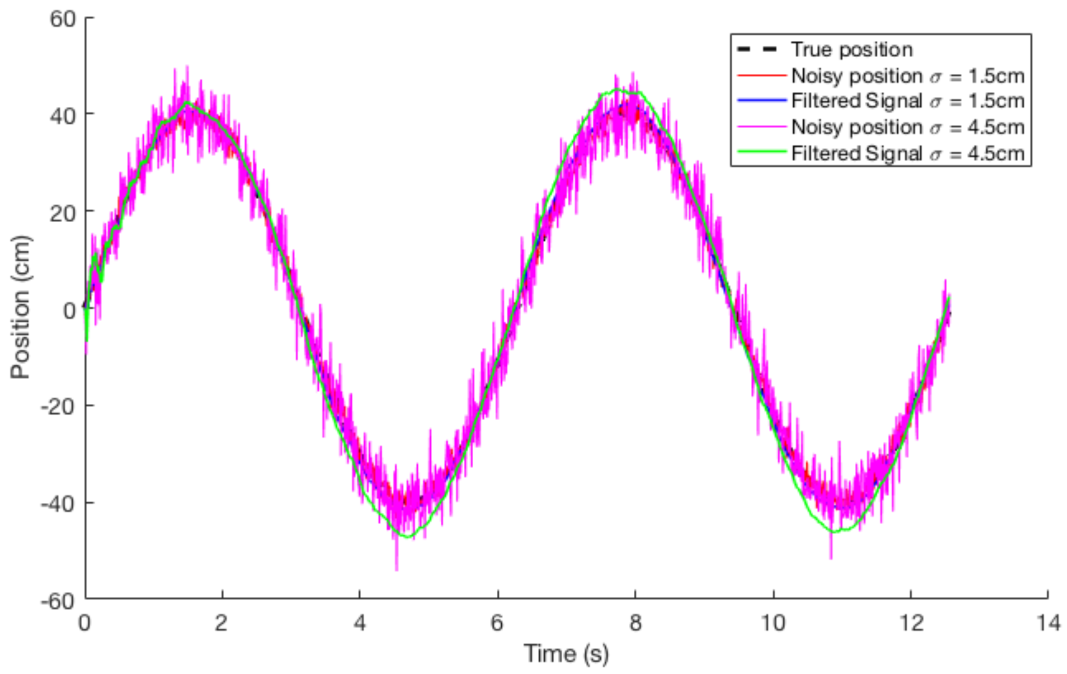
Completed by: Jenna Kelly Luchak
CID: 01429938

March 8, 2018

Question 1

Part A – Track the position of something with one sensor

First we investigate how the central nervous system (CNS) can utilize a noisy signal to detect a smoother movement, such as a butterfly trajectory. The Tutorial 5 Matlab code was ran with a variance, $\sigma_v = 1.5cm$ and $\sigma_v = 4.5cm$ and the resulting plots from each trial can be seen in Figure 1 below.



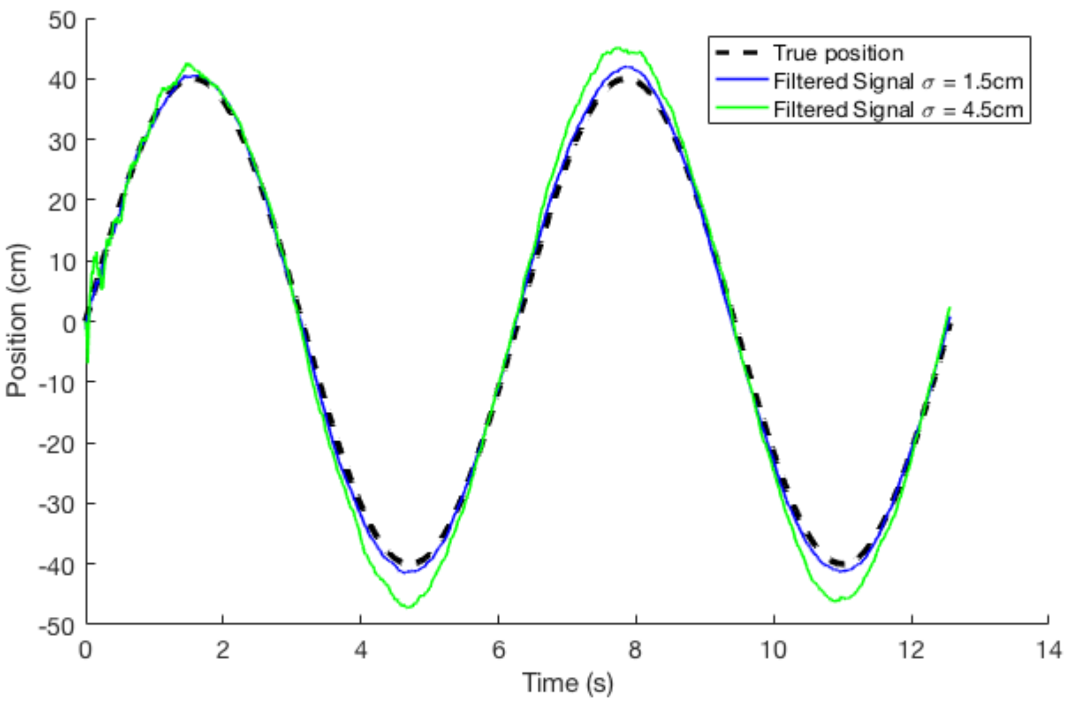


Figure 1: Results from a Linear Quadratic Estimator (LQE) with a variance of 1.5cm and 4.5cm. [Top] Plots shown are the butterfly's true trajectory, the noisy sensory information received and the filtered signal as a function of time. [Bottom] For clarity, the same plot is shown, but with the noisy sensory information removed.

As can bee seen from Figures 1, as the covariance of the model increases from 1.5cm to 4.5 cm, the filtered position agrees less with the true position; the error between the true position and the filtered signal increases. The filtered signal reaches a higher maximum and minimum when the covariance was 4.5cm, rather than 1.5cm. In Figure 1 the noisy position oscillates about the true position; however when $\sigma_v = 4.5$ cm, the noisy position oscillates with a larger amplitude relative to the true position curve than when $\sigma_v = 1.5cm$. Thus these results show that the signal is tracked using only visual feedback.

Additionally, the convergence of the filtered signals was investigated. It is assumed that the filtered signals depend on the limited covariance matrix. Two different covariance matrices were simulated, (i) $P_o = diag(10^5, 10^5, 10^5)$ and (ii) $P_o = diag(1,1,1)$. When $P_o = diag(1,1,1)$ is used, the filtered signal is delayed relative to the true position. Meaning, the maximum values reached by the filtered signals do not occur at the same time as the true position. This observation is most clear when $\sigma_v = 4.5cm$ as opposed to 1.5cm. However, when the covariance matrix is $P_o = diag(10^5, 10^5, 10^5)$ the filtered signals agree more closely to the true position, therefore the convergence of the filtered signals depends on the initial covariance matrix.

Part B – Track position with two sensors

Next we investigate how filtering is affected when two sensors are used compared to one sensor. This investigation will be evaluated using the square estimation error of the filtered signal relative to the true position, $E = \left\|x_{filtered} - x_{true}\right\|^2$. To use two sensors in our model, the covariance matrix and projection matrix were changed to the following equations.

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

$$Covariance\ \ Matrix \rightarrow R = \begin{bmatrix} \sigma_v^2 & 0 \\ 0 & \sigma_p^2 \end{bmatrix}$$

Where $\sigma_v^2 = \sigma_p^2 = 1.5cm$.

When tested with only vision (one sensor) and when tested with vision and proprioception (two sensors), the results of the Kalman filtering can be seen in Figure 2 below.

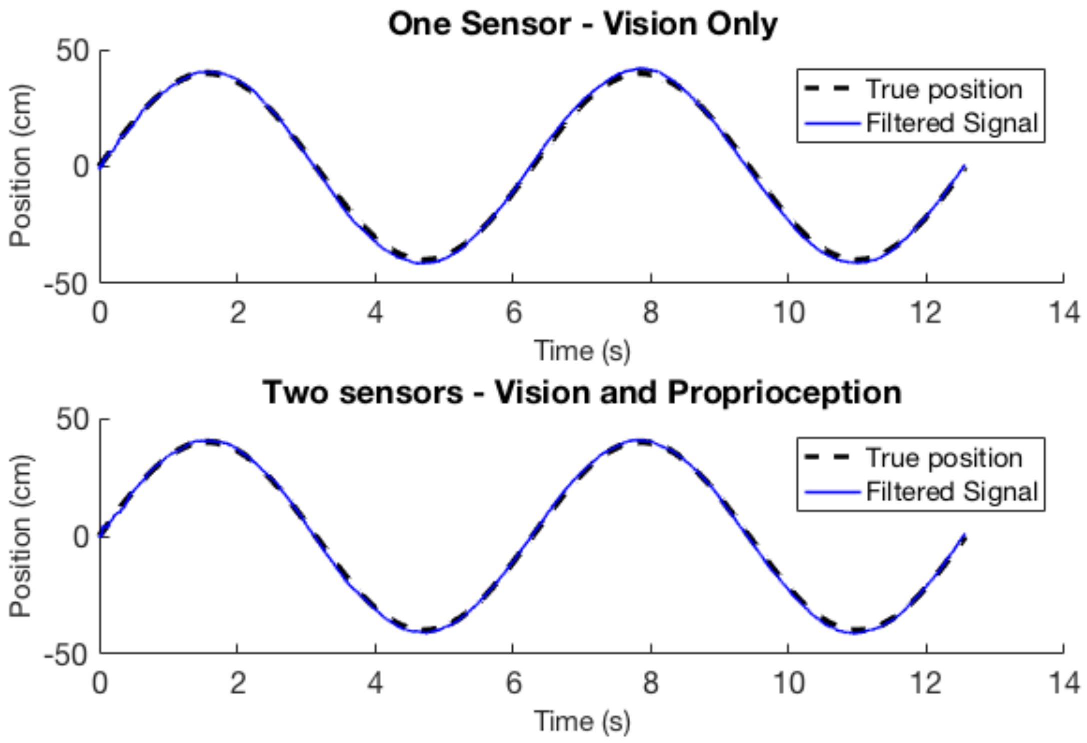


Figure 2: Results from a Linear Quadratic Estimator (LQE) modelled with $\sigma_v^2 = \sigma_p^2 = 1.5cm$. [Top] One sensor with only vision and [Bottom] two sensors with vision and proprioception.

From the plots shown in Figure 2 above, the filtered signals agree almost identically well with the true position, whether there is one sensor or two sensors present. However when you compare their errors using the square estimation error equation, it can be seen that the filter using vision and proprioception feedback agrees more closely to the true position than the filter with only visual feedback. The errors obtained using Kalman filtering with only vision (one sensor) as opposed to vision and proprioception (two sensors) was 1.314 and 0.628 respectively. The addition of a second sensor reduced the error of the model. Therefore, it can be inferred that the more sensors present in a model, the lower the error.

The Kalman filtering works better when it uses two sensors as opposed to one sensor. The filtering performance improved when two sensors were used, as the error reduced from 1.314 and 0.628.

Part C

Next we investigate how filtering is affected when proprioception noise increases, but visual feedback stays the same. Therefore, $\sigma_v^2 = 1.5cm\ and\ \sigma_p^2 = 4.5cm$. Two cases were tested, (i) When one sensor is used with just visual feedback, and (ii) when two sensors are used with visual and proprioception feedback. A plot with all of the results of this investigation can be seen in Figure 3 below.

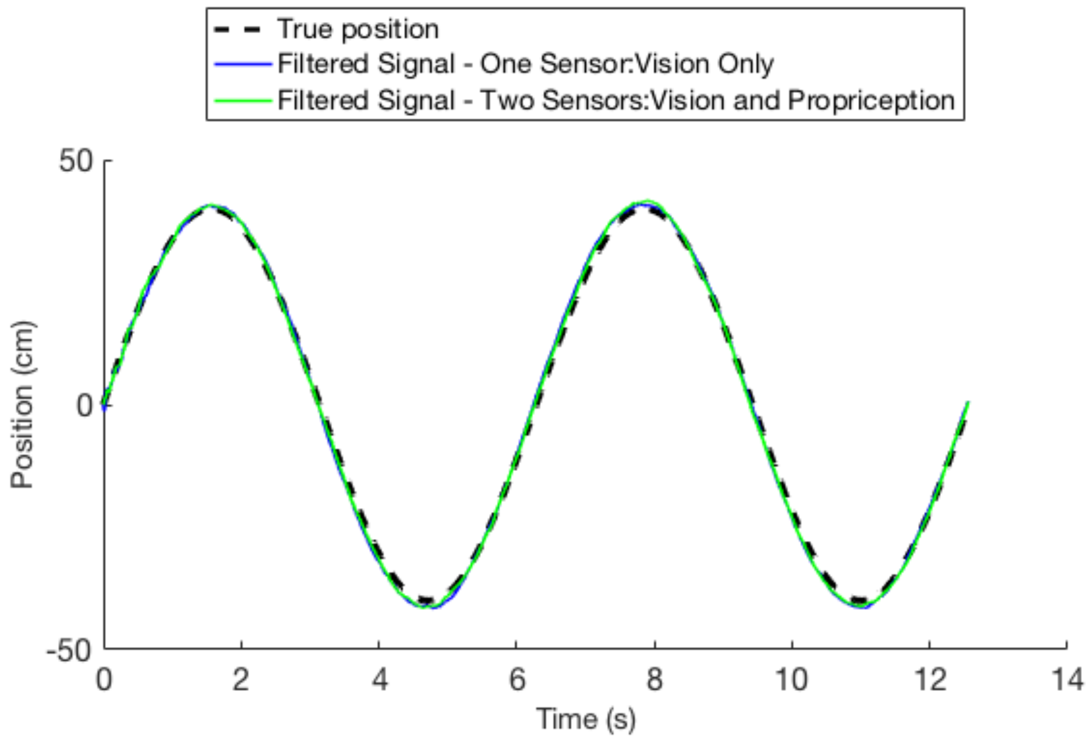


Figure 3: Results from a Linear Quadratic Estimator (LQE) modelled with $\boldsymbol{\sigma_v^2 = 1.5cm\ and\ \sigma_p^2 = 4.5cm}$ (two sensors) and an LQE modelled with $\boldsymbol{\sigma_v^2 = 1.5cm\ (one\ sensor)}$.

Based on the information shown in Figure 3, the filtered signals agree almost identically well with the true position, whether there is one sensor or two sensors present with increased proprioception. However when you compare their errors using the square estimation error equation, it can be seen that the filter using vision and proprioception feedback agrees more closely to the true position than the filter with only visual feedback. The errors obtained using Kalman filtering with only vision (one sensor) as opposed to vision and increased proprioception (two sensors) was 1.314 and 0.942 respectively. The addition of a second sensor reduced the error of the model. Therefore, it can be inferred that the more sensors present in a model, the lower the error.

The observed error of using two sensors with $\sigma_p = 4.5cm$ (0.942) was higher than the observed error in the filter when $\sigma_p = 1.5cm$ (0.628) in part B above. This result is expected, as the increase in proprioception noise will reduce the accuracy of the filter; however, even with increased proprioception noise into the filter, using two sensors as opposed to one improves the estimation of the true signal. Therefore, using a filtered signal with vision and proprioception tracks the true position better.

Part D – Implement a time delay into the Kalman filter

Our last investigation is to see how a time delay affects the Kalman filtering process. We assume that the central nervous system (CNS) compensates for the sensory delay by applying the following equations to the Kalman filter.

$$Current\ state \rightarrow z_j^* = A^{\delta}\hat{z}_{j-\delta}$$

$$Future\ state \rightarrow \hat{z}_{j+1-\delta} = \hat{z}_{j-\delta} + K_{j+1}\left(y_{j-\delta} - C\hat{z}_{j-\delta}\right)$$

Where $\delta = 0.1$, however, with a time step of dt=0.01, the delay is equivalent to 10 time steps.
The plot of the results of this time delay investigation can be seen in Figure 4 below.

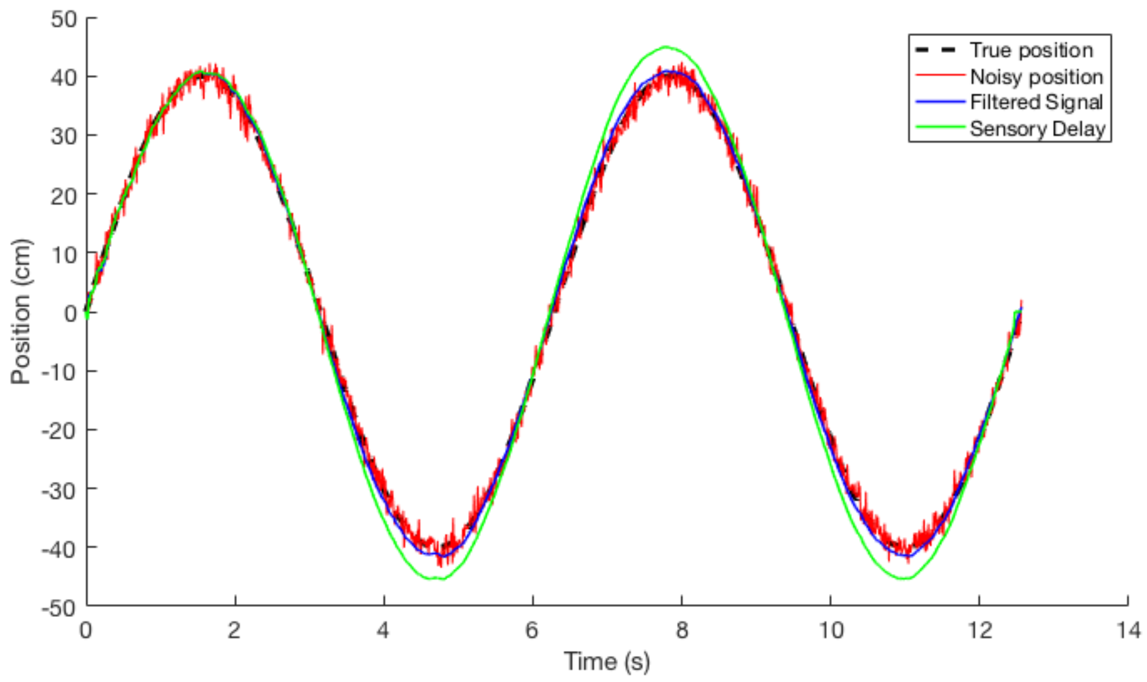


Figure 4: Results from a Linear Quadratic Estimator (LQE) modelled with $\sigma_v^2 = 1.5cm$ and visual feedback only. The filtered signal plot was modelled without a sensory delay to the Kalman filter, where as the sensory delay plot was modelled with a sensory delay of $\delta = 0.1$.

Based on the results from Figure 4, it can be seen that the sensory delay affects the amplitude of the signal. Compared against the filtered signal, true position and noisy signal, which were plotted without a time delay, the sensory delay's signal causes larger amplitude. As a result of this increased amplitude, the error of the model increased relative to the true position. The errors obtained using Kalman filtering without a sensory delay and with a sensory delay was 1.314 and 12.15 respectively. The addition of a sensory delay significantly increased the error of the model. Therefore, it can be inferred that the CNS must compensate for the increased visual feedback of a time delay by adjusting the amplitude.

Appendix

The code used for this assignment can be found below as follows. Please note that part A, B and C of this code used the Kalman function provided in this assignment and was not altered. For part, D the Kalman filter was changed and can be seen below the main codes script.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This script was prepared by:
% Jenna Luchak
% CID: 01429938
% For Human Neuromechanical Control: Tutorial #5
% March 8, 2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

clear all; clc; close all;

% Discretization of continuous system
dt = 0.01;

% Generate time array
t = 0:dt:4*pi;

%%%%%%% Kalman filter matrices %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% State transition matrix
A = [1 dt dt^2/2;
     0  1     dt;
     0  0     1];

% System noise and covariance matrix
SigmaQ = 0.3;
Q=[ SigmaQ^6/36    SigmaQ^5/12   SigmaQ^4/6
    SigmaQ^5/12    SigmaQ^4/4    SigmaQ^3/2
    SigmaQ^4/6     SigmaQ^3/2    SigmaQ^2];

% Observation matrix
C_1 = [1 0 0]; % Part A
C_2 = [1 0 0;1 0 0]; % PArt B and C

for i=1:2
% Observation noise
Sigma_A = [1.5 4.5]; % Question A
Sigma_pB = 1.5; Sigma_vB = 1.5; % Question B
Sigma_pC = 4.5; Sigma_vC = 1.5; % Question C

% Covariance matrix
R_A=(Sigma_A(i)^2); % Question A
R_B = [Sigma_vB^2 0;0 Sigma_pB^2]; % Question B
R_C = [Sigma_vC^2 0;0 Sigma_pC^2]; % Question C

% Initialize state and error covariance matrices
xInit = zeros(3,1);
PInit = diag([1 1 1]).*10^5; % Change to diag([1 1 1]); for part A

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generate signal
Signal_A(i,:) = 40*sin(t); % Question A
Signal_BC =[40*sin(t);40*sin(t)]; % Question B and C

% Generate noise corrupted signal
```

```
SignalNoisy_A(i,:) = Signal_A(i,:) +
sqrt(R_A)*randn(size(R_A,1),length(Signal_A(i,:))); % Part A or Part B or C
with 1 sensor
SignalNoisy_B = Signal_BC + sqrt(R_B)*randn(size(R_B,1),length(Signal_BC)); %
Part B with 2 sensors
SignalNoisy_C = Signal_BC + sqrt(R_C)*randn(size(R_C,1),length(Signal_BC)); %
Part C with 2 sensors

% LQE function
S_A(i).a = KalmanFilter(A,C_1,Q,R_A,xInit,PInit,SignalNoisy_A(i,:)); % Part A
or Part B or C with 1 sensor
S_B = KalmanFilter(A,C_2,Q,R_B,xInit,PInit,SignalNoisy_B(i,:)); % Part B with
2 sensors
S_C = KalmanFilter(A,C_2,Q,R_C,xInit,PInit,SignalNoisy_C); % Part C with 2
sensors
S_D = KalmanFilter_delay(A,C_1,Q,R_A,xInit,PInit,SignalNoisy_A(1,:)); % Part
D with sigma = 1.5

end
% Re define Kalman filtering outputs from struct to variable
x_15 = S_A(1).a; % Part A
x_45 = S_A(2).a; % Part A

% Square Estimation Error for Part B and C only
error_1 = norm(x_15(1,:)-Signal_A(1,:))^2/length(t); % One sensor
error_2B = norm(S_B(1,:)-Signal_BC(1,:))^2/length(t); % Two sensors
error_2C = norm(S_C(1,:)-Signal_BC(1,:))^2/length(t); % Two sensors
error_D = norm(S_D(1,:)-Signal_A(1,:))^2/length(t); % Two sensors

%%  Plots Part A
set(0,'DefaultFigureWindowStyle','docked')
figure(1);set(gcf,'color','white');
set(gca,'FontSize',14)
hold on;
% When sigma = 1.5
plot(t,Signal_A(1,:),'--k','linewidth',2.5); % True signal
plot(t,SignalNoisy_A(1,:),'r'); % noisy signal 1.5
plot(t,x_15(1,:),'b','linewidth',1.5); %filter 1.5
% When sigma = 4.5
plot(t,SignalNoisy_A(2,:),'m'); % noisy 4.5
plot(t,x_45(1,:),'g','linewidth',1.5); % filter 4.5
hold off;
legend('True position','Noisy position \sigma = 1.5cm','Filtered Signal
\sigma = 1.5cm',...
   'Noisy position \sigma = 4.5cm','Filtered Signal \sigma = 4.5cm');
xlabel('Time (s)','fontsize',15);
ylabel('Position (cm)','fontsize',15);

%% Part B
%One Sensor
figure(2);set(gcf,'color','white');
subplot(2,1,1)
set(gca,'FontSize',16)
hold on;
plot(t,Signal_BC(1,:),'--k','linewidth',2.5); % True signal
plot(t,x_15(1,:),'b','linewidth',1.5); % filter
hold off;
legend('True position','Filtered Signal');
xlabel('Time (s)','fontsize',15);
ylabel('Position (cm)','fontsize',15);
title('One Sensor - Vision Only')
hold off
```

```
% Two Sensors
subplot(2,1,2)
set(gca,'FontSize',16)
hold on;
plot(t,Signal_BC(1,:),'--k','linewidth',2.5); % True signal
plot(t,S_B(1,:),'b','linewidth',1.5); % filter
hold off;
legend('True position','Filtered Signal');
xlabel('Time (s)','fontsize',15);
ylabel('Position (cm)','fontsize',15);
title('Two sensors - Vision and Proprioception');

%% Part C -
figure(3);set(gcf,'color','white');
set(gca,'FontSize',16)
hold on;
plot(t,Signal_BC(1,:),'--k','linewidth',2.5); % True signal
% One Sensor
plot(t,x_15(1,:),'b','linewidth',1.5); % filter
% Two Sensors
plot(t,S_C(1,:),'g','linewidth',1.5); % filter
hold off;
legend('True position','Filtered Signal - One Sensor:Vision Only',...
    'Filtered Signal - Two Sensors:Vision and
Propriception','location','northoutside');
xlabel('Time (s)','fontsize',15);
ylabel('Position (cm)','fontsize',15);

%% Part D
figure(4);set(gcf,'color','white');
set(gca,'FontSize',14)
hold on;
plot(t,Signal_A(1,:),'--k','linewidth',2.5); % True signal
plot(t,SignalNoisy_A(1,:),'r'); % noisy signal
plot(t,x_15(1,:),'b','linewidth',1.5); %filtered
plot(t,S_D(1,:),'g','linewidth',1.5); % Delayed
hold off;
legend('True position','Noisy position','Filtered Signal',...
   'Sensory Delay');
xlabel('Time (s)','fontsize',15);
ylabel('Position (cm)','fontsize',15);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Script
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This function was altered by:
% Jenna Luchak
% CID: 01429938
% For Human Neuromechanical Control: Tutorial #5
% March 8, 2018
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% z - State variable over time
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [z] = KalmanFilter_delay(A,C,Q,R,z0,P0,y)

    % Allocate memory for speed
    z = zeros(size(z0,1),length(y));
    % Initialize Kalman filter
    z(:,1) = z0;
    P = P0;

    % Kalman filter loop
    for i=11:1:length(y)-1
        % Prediction
        z_Prior = A*z(:,i-10);
        P = A*P*A'+Q;

        % Correction
        K = P*C'/(C*P*C'+R);
        z(:,i+1-10) = z_Prior + K*(y(:,i-10)-C*z_Prior);
        P = (eye(3)-K*C)*P;
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End of Function
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```