ECE 8540 Analysis of Tracking Systems
Lab 4 - Kalman Filter

Submitted By:
Shashi Shivaraju
C88650674

Clemson University
October 9, 2018

Abstract

This report explains the process involved in designing a Kalman filter to mitigate noise in sensor measurements and to estimate the parameters of the model used to track the object.

Contents

List of Figures

Chapter 1

Introduction

This report considers the problem of designing a Kalman Filter for mitigating the noises from the sensor measurements which are used to track an object.

A filter is a mathematical tool that uses an expected dynamic model to help mitigate noise in sensor data.Kalman filtering, also known as linear quadratic estimation (LQE), is an algorithm that uses a series of measurements observed over time, containing statistical noise and other inaccuracies, and produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone, by estimating a joint probability distribution over the variables for each time frame. The filter is named after Rudolf E. Klmn, one of the primary developers of its theory.

The Kalman filter has numerous applications in technology. In tracking applications,The Kalman filter is one of the popular filtering algorithms. A common application is for guidance, navigation, and control of vehicles, particularly aircraft and spacecraft. Furthermore, the Kalman filter is a widely applied concept in time series analysis used in fields such as signal processing and econometrics.

This report describes Kalman filtering for given data sets "1D-data.txt" and "2D UWB tracking data" considering the model to be constant velocity 1D model and constant velocity 2D model respectively.

Chapter 2

Methods

Kalman filtering is a continuous cycle of predict-update. It involves three main steps:prediction,measuremen
and updation. Before we apply the Kalman filtering to the data,one has to determine the a mathematical model which describes the behavior of the object being tracked. This follows the below steps:

1. Determine the state variables,which describes the property of the object being tracked.
2. Determine the state transition equations, which provides description of nominal expected behavior of the state variables.
3. Define the dynamic noises,which determine the possible deviations during a state transition.
4. Determine the observation variables,which are sensor measurement used to track the object.
5. Define the observation equations,which relate the sensor readings to the state variables.
6. Define the measurement noises,which describes the possible corruptions during a sensor reading.

Using these mathematical models,the Kalman filter is implemented. The design and implementation of Kalman filter for the given data-sets are described in below sections.

2.1 Kalman filter design for 1D constant velocity model

The dataset "1D-data.txt" describes the position of an object moving along an x axis.

Let the the state X_t of the system be defined as:

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} \tag{2.1}$$

where x_t is the position and $\dot{x}_t$ is the velocity.

Let the state transition equations for this model be:

$$\begin{aligned} x_{t+1} &= x_t + T\dot{x}_t \\ \dot{x}_{t+1} &= \dot{x}_t \end{aligned} \tag{2.2}$$

Let us assume that random acceleration a, can happen between the sensor samples, and define the dynamic noises as:

$$\text{dyn noise} = \begin{bmatrix} 0 \\ N(0, \sigma_a^2) \end{bmatrix} \tag{2.3}$$

Since the given data is the x positions, let us denote the sensor readings as:

$$Y_t = \begin{bmatrix} \tilde{x}_t \\ 0 \end{bmatrix} \tag{2.4}$$

The observation equation for this system can be written as:

$$\tilde{x}_t = x_t \tag{2.5}$$

Let us assume that the sensor readings are corrupted by noise n,that can be modeled as:

$$\text{measurement noise} = \begin{bmatrix} N(0, \sigma_n^2) \end{bmatrix} \tag{2.6}$$

The co-variance of the state variables can be written as:

$$\text{cov(state)} = \text{cov(X)} = S_t = \begin{bmatrix} \sigma_{x_t}^2 & \sigma_{x_t,\dot{x}_t} \\ \sigma_{x_t,\dot{x}_t} & \sigma_{\dot{x}_t}^2 \end{bmatrix} \tag{2.7}$$

The co-variance of the dynamic noises can be written as:

$$\text{cov(dyn noise)} = Q = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_a^2 \end{bmatrix} \tag{2.8}$$

The co-variance of the measurement noise can be written as:

$$\text{cov(meas noise)} = R = \begin{bmatrix} \sigma_n^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.9}$$

The state transition matrix Φ can be obtained by looking at equation (2.2).It is defined as:

$$\Phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{2.10}$$

The observation matrix M can be obtained by looking at equation (2.5). For the example, it is defined as:

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{2.11}$$

The above equations formulate the model for the filtering problem of 1D constant velocity model . The equations provide a theoretical model of the system being tracked and describes its expected behavior.

The implementation of Kalman filter for this model is explained in section 2.3.

2.2 Kalman filter design for 2D constant velocity model

The dataset "2D UWB tracking data" describes the position of an object moving in a 2D plane.

Let the the state X_t of the system be defined as:

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} \tag{2.12}$$

where x_t and y_t are the position and $\dot{x}_t$ and $\dot{y}_t$ are the velocity.

Let the state transition equations for this model be:

$$\begin{aligned} x_{t+1} &= x_t + T\dot{x}_t \\ y_{t+1} &= y_t + T\dot{y}_t \\ \dot{x}_{t+1} &= \dot{x}_t \\ \dot{y}_{t+1} &= \dot{y}_t \end{aligned} \tag{2.13}$$

Let us assume that random accelerations a_1,a_2 can happen between the sensor samples and define the dynamic noises as:

$$\text{dyn noise} = \begin{bmatrix} 0 \\ 0 \\ N(0, \sigma_{a_1}^2) \\ N(0, \sigma_{a_2}^2) \end{bmatrix} \tag{2.14}$$

Since the given dataset contains (x, y) positions, let us denote the sensor readings as:

$$Y_t = \begin{bmatrix} \tilde{x}_t \\ \tilde{y}_t \end{bmatrix} \tag{2.15}$$

The observation equation for this system can be written as:

$$\begin{aligned} \tilde{x}_t &= x_t \\ \tilde{y}_t &= y_t \end{aligned} \tag{2.16}$$

Let us assume that the sensor readings are corrupted by noises n_1, n_2,that can be modeled as:

$$\text{meas noise} = \begin{bmatrix} N(0, \sigma_{n_1}^2) \\ N(0, \sigma_{n_2}^2) \end{bmatrix} \tag{2.17}$$

The co-variance of the state variables can be written as:

$$\text{cov(state)} = \text{cov(X)} = S_t = \begin{bmatrix} \sigma_{x_t}^2 & \sigma_{x_t,y_t} & \sigma_{x_t,\dot{x}_t} & \sigma_{x_t,\dot{y}_t} \\ \sigma_{x_t,y_t} & \sigma_{y_t}^2 & \sigma_{y_t,\dot{x}_t} & \sigma_{y_t,\dot{y}_t} \\ \sigma_{x_t,\dot{x}_t} & \sigma_{y_t,\dot{x}_t} & \sigma_{\dot{x}_t}^2 & \sigma_{\dot{x}_t,\dot{y}_t} \\ \sigma_{x_t,\dot{y}_t} & \sigma_{y_t,\dot{y}_t} & \sigma_{\dot{x}_t,\dot{y}_t} & \sigma_{\dot{y}_t}^2 \end{bmatrix} \tag{2.18}$$

The co-variance of the dynamic noises can be written as:

$$\text{cov(dyn noise)} = Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{a_1}^2 & \sigma_{a_1,a_2} \\ 0 & 0 & \sigma_{a_1,a_2} & \sigma_{a_2}^2 \end{bmatrix} \tag{2.19}$$

The co-variance of the measurement noise can be written as:

$$\text{cov(meas noise)} = R = \begin{bmatrix} \sigma_{n_1}^2 & \sigma_{n_1,n_2} \\ \sigma_{n_1,n_2} & \sigma_{n_2}^2 \end{bmatrix} \tag{2.20}$$

The state transition matrix Φ can be obtained by looking at equation (2.13).It is defined as:

$$\Phi = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.21}$$

The observation matrix M can be obtained by looking at equation(2.16). For the example, it is defined as:

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{2.22}$$

The above equations formulate the model for the filtering problem of 2D constant velocity model . The equations provide a theoretical model of the system being tracked and describes its expected behavior.

The implementation of Kalman filter for this model is explained in section 2.3.

2.3 Implementation of Kalman Filter

The Kalman filter is a continuous cycle of predict-update. The following equations form the main loop of the kalman filter applied to the models described in section 2.1 and section 2.2:

1. Predict the next state
$$X_{t,t-1} = \Phi X_{t-1,t-1} \tag{2.23}$$
2. Predict the next state co-variance
$$S_{t,t-1} = \Phi S_{t-1,t-1} \Phi^T + Q \tag{2.24}$$
3. Obtain sensor measurement(s) Y_t
4. Calculate the Kalman gain (weights)
$$K_t = S_{t,t-1} M^T [M S_{t,t-1} M^T + R]^{-1} \tag{2.25}$$
5. Update the state
$$X_{t,t} = X_{t,t-1} + K_t (Y_t - M X_{t,t-1}) \tag{2.26}$$
6. Update the state co-variance
$$S_{t,t} = [I - K_t M] S_{t,t-1} \tag{2.27}$$
7. loop (now t becomes $t+1$)

The output from the filter is the result of the state update and state co-variance update equations. These provide the combined estimate from the model (prediction equations) and latest observation (measurements).

Please refer the Appendix for the Matlab implementation of Kalman filter for models described in section 2.1 and section 2.2.

Chapter 3

Results

The Kalman filter was implemented for 1D and 2D constant velocity models and the values of dynamic noise co-variance matrix Q and measurement noise co-variance matrix R is varied to observe the variations in Kalman filter output.

3.1 Kalman filter output for 1D constant velocity model

For 1D constant velocity model,we kept the measurement noise co-variance matrix R constant and varied the dynamic noise co-variance matrix Q.

1. Measurement Noise [MN] set to 1.0 and Dynamic Noise [DN] set to 0.1

   This setting of noises produces a Kalman filter output which is smoother than the sensor measurements but is jumpy in nature. The Kalman filter output looks to follow the sensor measurements more. The corresponding sensor measurements and the kalman filter output is shown in figure(3.1).

2. Measurement Noise [MN] set to 1.0 and Dynamic Noise [DN] set to 0.0001

   This setting of noises produces a Kalman filter output which is smoother than the kalman filter output is shown in figure(3.1). The corresponding sensor measurements and the kalman filter output is shown in figure(3.2).

3. Measurement Noise[MN] set to 1.0 and Dynamic Noise [DN] set to 0.000001

   This setting of noises produces a Kalman filter output which is smoother than the kalman filter output is shown in figure(3.2). The corresponding sensor measurements and the kalman filter output is shown in figure(3.3).

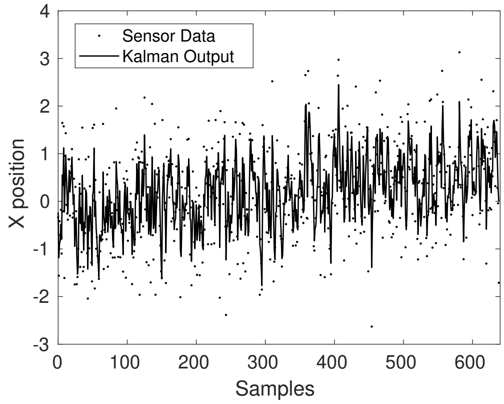

Figure 3.1: MN=1.0 & DN=0.1

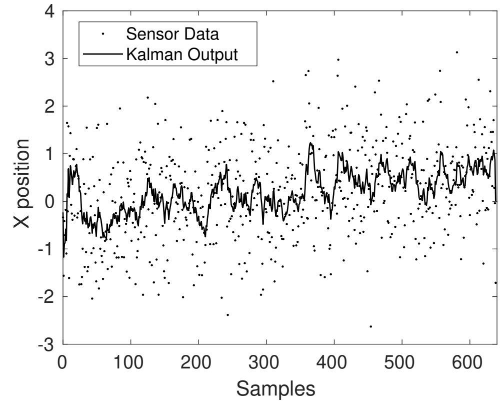

Figure 3.2: MN=1.0 & DN=0.0001

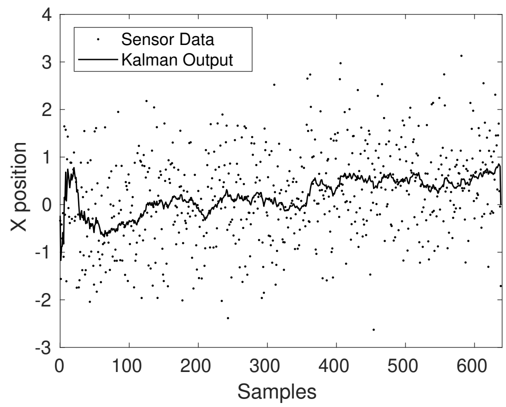

Figure 3.3: MN=1.0 & DN=0.000001

3.2 Kalman filter output for 2D constant velocity model

For 2D constant velocity model,we kept the dynamic noise co-variance matrix Q constant and varied the measurement noise co-variance matrix R.

1. Measurement Noises [MN] are set to 0.1 and Dynamic Noises [DN] are set to 0.001

   This setting of noises produces a Kalman filter output which varies from the the sensor measurements since the Measurement Noises [MN] are greater than the Dynamic Noises [DN]. The corresponding sensor measurements and the kalman filter output are shown in figure(3.4) and figure(3.5).

2. Measurement Noises [MN] are set to 0.01 and Dynamic Noises [DN] are set to 0.001

   This setting of noises produces a Kalman filter output which is closer to the sensor measurements when compared to the kalman filter output as shown in figure(3.4) and

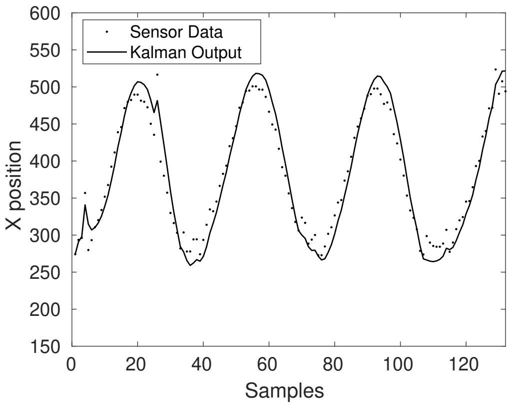

Figure 3.4: MN=0.1 & DN=0.001

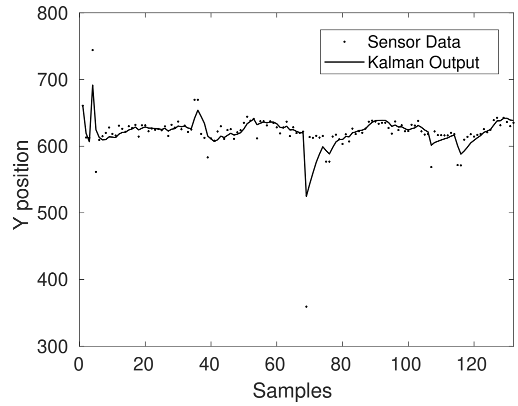

Figure 3.5: MN=0.1 & DN=0.001

figure(3.5).This is because of the decrease in the Measurement Noises [MN]. The corresponding sensor measurements and the kalman filter output are shown in figure(3.6) and figure(3.7).

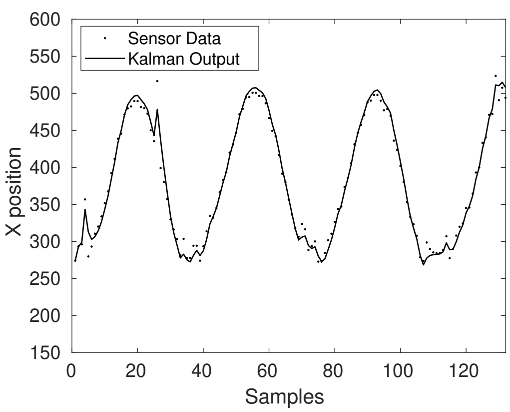

Figure 3.6: MN=0.01 & DN=0.001

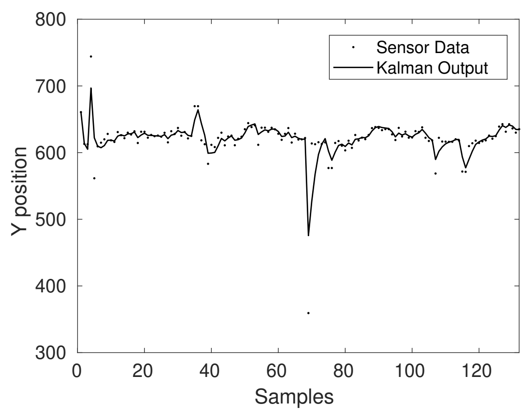

Figure 3.7: MN=0.01 & DN=0.001

3. Measurement Noises [MN] are set to 0.001 and Dynamic Noise [DN] are set to 0.001

   This setting of noises produces a Kalman filter output which is closer to the sensor measurements when compared to the kalman filter output as shown in figure(3.6) and figure(3.7).This is because of the further reduction in the Measurement Noises [MN]. The Kalman filter output looks to follow the sensor measurements more. The corresponding sensor measurements and the kalman filter output are shown in figure(3.8) and figure(3.9).

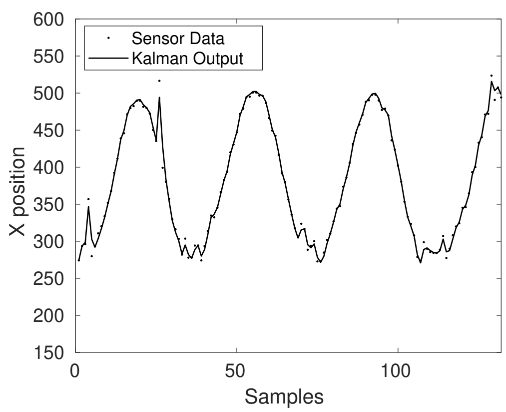

Figure 3.8: MN=0.001 & DN=0.001

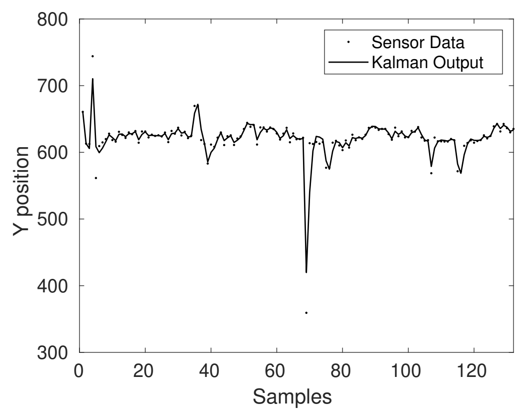

Figure 3.9: MN=0.001 & DN=0.001

Chapter 4

Conclusion

The report describes in detail the analytical process involved in designing and implementing a Kalman Filter. It was observed that the output of the Kalman filter(due to Kalman gain) depends on the relative ratio of the measurement noise to the dynamic noise.Thus we fix one of the above mentioned noise and tune the other noise to achieve the desired level of smoothing.

Appendix

Matlab Code for 1D constant velocity model

```
% FILE NAME       : OneD_Constant_Velocity.m
%
% DESCRIPTION     : Code to implement Kalman filter for 1D Constant
    Velocity Model.
%
% PLATFORM                    : Matlab
%
% DATE                        NAME
% 9th-Oct-2018        Shashi Shivaraju

clear; %clear all the varaibles
clc; %clear the screeen

%Read data from file
Data = dlmread("1D-data.txt");

%Initialize the matrices
time = 1; %current time instant
measure_noise = 1; %Measurement noise
dynamic_noise = 0.000001; %Dynamic nosie
Yt = [Data' ; zeros(1, length(Data)) ]; %Sensor measurements
Filter_Output = zeros(1, length(Data)); %Filter Output
I = [1 0; 0 1]; %Identity matrix
STrans_Mat = [1 time ; 0 1]; %State Transition Matrix
M = [1 0; 0 0]; %Observation Matrix
X_t1_t1 = [0 ; 0]; %State matrix
R = [measure_noise 0.1; 0.1 0.1 ]; %CoVariance of Measurement
    noise
Q = [0 0 ; 0 dynamic_noise]; %CoVariance of Dynamic noise
S_t1_t1 = I; %State Covariance
K_G = [0 0; 0 0]; %Kalman Gain


%loop through the data set
while(time < length(Data))

    %Predict the next state
    X_t_t1 = STrans_Mat * X_t1_t1;
```

```
    %Predict the next state covariance
    S_t_t1 = (STrans_Mat * S_t1_t1 * STrans_Mat') + Q ;
    %Calculate the Kalman Gain
    K_G = (S_t_t1 * M') / ( M * S_t_t1 * M' + R); % (S_t_t1 * M')
        * inv( M * S_t_t1 * M' + R)

    %Update the state
    X_t_t = X_t_t1 + (K_G * (Yt(:,time) - (M * X_t_t1) ));
    %Update state covariance
    S_t_t = (I - (K_G * M) ) * S_t_t1;

    %Store the filter output for plotting
    Filter_Output(time) = X_t_t(1,1);

    %loop (now t becomes t+1)
    time = time + 1 ;
    X_t1_t1 = X_t_t;
    S_t1_t1 = S_t_t1;

 end %end of while


%Plot the sensor data and the filter output
x = 0:length(Data)-1;
figure(1)
plot(x,Data,"k.","markersize",3) ;
hold on
plot(x, Filter_Output, "k-","Linewidth",1);
hold off
set(gca,"FontSize",14);
xlabel('Samples');
ylabel('x-position');
legend('Sensor Data','Kalman Output');
axis([0 640 -3 4]);
```

Matlab Code for 2D constant velocity model

```

% FILE NAME      : TwoD_Constant_Velocity.m
%
% DESCRIPTION    : Code to implement Kalman filter for 2D Constant
    Velocity Model.
%
% PLATFORM              : Matlab
```

```
%
% DATE                    NAME
% 9th-Oct-2018        Shashi Shivaraju

clear; %clear all the varaibles
clc; %clear the screeen

%Read data from file
Data = dlmread("2D_UWB_data.txt");

%Initialize the matrices
time = 1; %current time instant
measure_noise1 = 0.0001; %Measurement noise
measure_noise2 = 0.001; %Measurement noise
dynamic_noise1 = 0.0001; %Dynamic nosie
dynamic_noise2 = 0.0001; %Dynamic nosie
Yt = Data';  %Sensor measurements
Filter_Output = zeros(2, length(Data(:,1))); %Filter Output
I = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1]; %Identity matrix
STrans_Mat = [1 0 time 0 ; 0 1 0 time; 0 0 1 0; 0 0 0 1]; %State transition matrix
M = [1 0 0 0 ; 0 1 0 0]; %Observation matrix
X_t1_t1 = [Data(1,1) ; Data(1,2) ; 0 ; 0]; %State matrix
R = [measure_noise1 0.00001; 0.00001 measure_noise2 ]; %CoVariance of Measurement noise
Q = [0 0 0 0; 0 0 0 0; 0 0 dynamic_noise1 0.00001; 0 0 0.00001 dynamic_noise2]; %Covariance of Dynamic noise
S_t1_t1 = [1 0.1 0.1 0.1; 0.1 1 0.1 0.1; 0.1 0.1 1 0.1; 0.1 0.1 0.1 1]; %State Covariance
K_G = zeros(4,2); %Kalman gain

%loop through the data set
while(time < length(Data(:,1) ))

     %Predict the next state
     X_t_t1 = STrans_Mat * X_t1_t1;
     %Predict the next state covariance
     S_t_t1 = (STrans_Mat * S_t1_t1 * STrans_Mat') + Q;
     %Calculate the Kalman Gain
     K_G = (S_t_t1 * M') / ( M * S_t_t1 * M' + R ); % (S_t_t1 * M') * inv( M * S_t_t1 * M' + R)

     %Update the state
     X_t_t = X_t_t1 + (K_G * (Yt(:,time) - (M * X_t_t1) ));
     %Update state covariance
```

```
        S_t_t = (I - (K_G * M) ) * S_t_t1 ;

       %Store the filter output for plotting
        Filter_Output(1,time) = X_t_t(1,1);
        Filter_Output(2,time) = X_t_t(2,1);

            %loop (now t becomes t+1)
            time = time + 1 ;
        X_t1_t1 = X_t_t;
        S_t1_t1 = S_t_t;

   end %end of while

%Plot the sensor data and the filter output
x = 1:length(Data(:,1));
figure(1)
plot(x,Data(:,1),"k.","markersize",3) ;
hold on
plot(x,Filter_Output(1,:), "k-","linewidth",1);
hold off
legend('Sensor Data','Kalman Output');
xlabel('Samples');
ylabel('x-position');
axis([0 132 150 600])


figure(2)
plot(x,Data(:,2),"k.","markersize",3) ;
hold on
plot(x,Filter_Output(2,:), "k-","linewidth",2);
hold off
legend('Sensor Data','Kalman Output');
xlabel('Samples');
ylabel('x-position');
axis([0 132 300 800])
```

References

1.Lecture notes of Dr.Adam Hoover

http://cecas.clemson.edu/ ahoover/ece854/lecture-notes/lecture-kf.pdf

2.TeX Live - TeX Users Group

https://www.tug.org/texlive/