ECE 8540
Analysis of Tracking Systems
Assignment 4

Vivek Koodli Udupa
C12768888

October - 09, 2018

1 Introduction

In this report we consider the problem of filtering; Kalman filtering to be specific. Filtering is a method of mitigating noise in sensor data. A set of sensor data is used to design a model. The model can be used to smooth out the noise and to make prediction about expected future states. However, in the case of tracking problem, the thing that is to be tracked is not expected to follow a linear predictable behavior all the time. If the object did follow a predictable path, then there would be no point in tracking it. In such a situation, the behavior of the object could just be modeled perfectly and the future behavior is completely known.

For example, Consider the problem of tracking a bullet fired through some kind of gun. Initially the bullet travels straight, but as time progresses, the bullet is slowed down due to gravity, air resistance and multiple other factors and its trajectory changes. Thus the bullet follows certain path for a particular duration and then changes its path for the next set period of time. In order to track such an object, that changes behavior slowly but continuously, it is necessary to change the model fit continuously. In the extreme this leads to the concept of updating the model fit after every new sensor reading is recorded. This is how filtering works.

In the problem of tracking something, we are trying to find the answer for the question " where is the object? ". The answer to this question is rarely if ever certain in real life applications. The answer is mostly a probability distribution of " the object is likely to be in this area ". This is because of the uncertainties in the sensor measurement, object's behavior and other noises that might persist. In this report we will consider data that contains dynamic and measurement noises and we will use Kalman filtering to generate predictions. Kalman filtering is an algorithm that uses a set of measurements collected over time, that contains statistical noises and other inaccuracies and produces estimates of unknown variables that tend to be more accurate than those based on single measurement alone, by estimating a joint probability distribution over the variables for each time frame.

This report deals with the application of kalman filter to track the position of an object in both 1D and 2D space. This report starts with the visual analysis of dataset, building the kalman filter equations and then finally discussing the results of the derived kalman filter.

2 Methods

Kalman filter is a continuous cycle of predict and update. When formulating a problem for the Kalman filter, we consider the following steps:

1. Determine the state variables. Here we are answering the question "what are we tracking?".
2. Write the state transition equations i.e. How things evolve over time.
3. Define the dynamic noise(s). This describes the uncertainties in state transition equation.
4. Determine the observation variables i.e. Sensor readings.

5. Write the observation equations (relating the sensor readings to the state variables).
6. Define the measurement noise(s). These are the uncertainties in observation variables.
7. Characterize the state transition matrix and observation matrix.
8. Check all matrices in the Kalman filter equations to make sure the sizes are appropriate.

2.1 Deriving the equations for 1D Model

We will begin by describing the things that we are tracking. The state X_t can be defined as:

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} \tag{1}$$

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

The second step is to write the state transition equations that describe the expected behavior of the state variables. Here, we are considering a constant 1D velocity model. The state transition equations for the model are as follows:

$$\begin{aligned} X_{t+1} &= x_t + T\dot{x}_t \\ \dot{X}_t &= \dot{x}_t \end{aligned} \tag{2}$$

The third step is to define the dynamic noise(s). Here we will assume that random accelerations can happen between sensor samples.

$$\text{dynamic noise} = \begin{bmatrix} 0 \\ N(0, \sigma_{a1}^2) \end{bmatrix} \tag{3}$$

Notice that the first element of the matrix is zero. The zero indicates that there is no noise in the position of the reading. Noise in position reading would indicate that the object is teleporting.

The forth step is to list the observation variables. Here, we are sensing the x position. Denoting the sensor reading for this as follows:

$$Y_t = \begin{bmatrix} \tilde{x_t} \\ 0 \end{bmatrix} \tag{4}$$

The second element of the matrix is zero because the velocity is not being measured.

The fifth step is to describe the observation equation.

$$\tilde{x_t} = x_t \tag{5}$$

The sixth step is to define measurement noise(s). Here, we will assume that the sensor reading is corrupted by noise.

$$\text{measurement noise} = \begin{bmatrix} N(0, \sigma_{n1}^2) \end{bmatrix} \tag{6}$$

The seventh step is to characterize the co-variance matrices. There are three of them.
The co-variance of the state variables is written as:

$$s_t = \begin{bmatrix} \sigma_{x_t}^2 & \sigma_{x_t,\dot{x}_t} \\ \sigma_{x_t,\dot{x}_t} & \sigma_{\dot{x}_t}^2 \end{bmatrix} \tag{7}$$

Co-variance of dynamic noise can be written as:

$$Q = \begin{bmatrix} 0 & 0 \\ 0 & \sigma_{a_1}^2 \end{bmatrix} \tag{8}$$

Co-variance of measurement noise can be written as:

$$R = \begin{bmatrix} \sigma_{n_1}^2 & 0 \\ 0 & 0 \end{bmatrix} \tag{9}$$

The eighth step is to define the state transition and observation matrices. The state transition matrix can be obtained from equation 2.

$$\phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix} \tag{10}$$

The observation matrix can be obtained from equation 4.

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \tag{11}$$

Note that we have just finished formulating the 1D model for filtering problem. This is just a theoretical model of how we expect the system being tracked to behave. Implementation will be explained in section 2.3

2.2 Deriving the equations for 2D Model

Similar to the derivation shown in section 2.1 this section will show the derivation to a constant 2D velocity model.
We will begin by describing the things that we are tracking. The state X_t can be defined as:

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} \tag{12}$$

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

The second step is to write the state transition equations that describe the expected behavior

of the state variables. Here, we are considering a constant 2D velocity model. The state transition equations for the model are as follows:

$$\begin{aligned} X_{t+1} &= x_t + T\dot{x}_t \\ Y_{t+1} &= y_t + T\dot{y}_t \\ \dot{X}_t &= \dot{x}_t \\ \dot{Y}_t &= \dot{y}_t \end{aligned} \tag{13}$$

The third step is to define the dynamic noise(s). Here we will assume that random accelerations can happen between sensor samples.

$$\text{dynamic noise} = \begin{bmatrix} 0 \\ 0 \\ N(0, \sigma_{a1}^2) \\ N(0, \sigma_{a2}^2) \end{bmatrix} \tag{14}$$

The forth step is to list the observation variables. Here, we are sensing the x and y positions. Denoting the sensor reading for this as follows:

$$Y_t = \begin{bmatrix} \tilde{x}_t \\ \tilde{y}_t \end{bmatrix} \tag{15}$$

The fifth step is to describe the observation equation.

$$\begin{aligned} \tilde{x}_t &= x_t \\ \tilde{y}_t &= y_t \end{aligned} \tag{16}$$

The sixth step is to define measurement noise(s). Here, we will assume that the sensor reading is corrupted by noise.

$$\text{measurement noise} = \begin{bmatrix} N(0, \sigma_{n1}^2) \\ N(0, \sigma_{n2}^2) \end{bmatrix} \tag{17}$$

The seventh step is to characterize the co-variance matrices. There are three of them.
The co-variance of the state variables is written as:

$$s_t = \begin{bmatrix} \sigma_{x_t}^2 & \sigma_{x_t,y_t} & \sigma_{x_t,\dot{x}_t} & \sigma_{x_t,\dot{y}_t} \\ \sigma_{x_t,y_t} & \sigma_{y_t}^2 & \sigma_{y_t,\dot{x}_t} & \sigma_{y_t,\dot{y}_t} \\ \sigma_{x_t,\dot{x}_t} & \sigma_{y_t,\dot{x}_t} & \sigma_{\dot{x}_t}^2 & \sigma_{\dot{x}_t,\dot{y}_t} \\ \sigma_{x_t,\dot{y}_t} & \sigma_{y_t,\dot{y}_t} & \sigma_{\dot{x}_t,\dot{y}_t} & \sigma_{\dot{y}_t}^2 \end{bmatrix} \tag{18}$$

Co-variance of dynamic noise can be written as:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{a_1}^2 & \sigma_{a_1,a_2} \\ 0 & 0 & \sigma_{a_1,a_2} & \sigma_{a_2}^2 \end{bmatrix} \tag{19}$$

Co-variance of measurement noise can be written as:

$$R = \begin{bmatrix} \sigma_{n_1}^2 & \sigma_{n_1,n_2} \\ \sigma_{n_1,n_2} & \sigma_{n_2}^2 \end{bmatrix} \tag{20}$$

The eighth step is to define the state transition and observation matrices. The state transition matrix can be obtained from equation 13.

$$\phi = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

The observation matrix can be obtained from equation 15.

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{22}$$

Note that we have just finished formulating the 2D model for filtering problem. This is just a theoretical model of how we expect the system being tracked to behave. Implementation will be explained in section 2.3

2.3 Implementation

As mentioned previously in section 2.1 and section 2.2 the Kalman filter is a continuous predict and update loop and the equations for Kalman parameters have been derived. This section will describe the implementation of this filter using the equations for the main loop.

The equation for predicting the next state is given by:

$$X_{t,t-1} = \phi \; X_{t-1,t-1} \tag{23}$$

Here $X_{t,t-1}$ represents the next state based on the previous state. ϕ is the state transition matrix and $X_{t-1,t-1}$ is the previous state.
Predicting the next state co-variance as:

$$S_{t,t-1} = \phi \; S_{t-1,t-1} \; \phi^T + Q \tag{24}$$

Then obtain the measurements, i.e Y_t

Next, calculate the Kalman gain(weights) as:

$$K_t = S_{t,t-1}\ M^T\ [\ M\ S_{t,t-1}\ M^T + R\ ]^{-1} \tag{25}$$

Now using the predicted state, Kalman gain and measurements update the state as:

$$X_{t,t} = X_{t,t-1} + K_t\ (\ Y_t\ - M\ X_{t,t-1}\ ) \tag{26}$$

Now update the state co-variance as :

$$S_{t,t} = [\ I\ -\ K_t\ M\ ]\ S_{t,t-1} \tag{27}$$

The implementation of the above described models and equations are done in MATLAB. Please refer the Appendix for the code.

3 Results

In this section we will plot the given dataset and the Kalman fiter's predicted values and see the effect of changing the measurement noise to dynamic noise ratio.

Here we are keeping the measurement noise,R fixed at 1 and we change the dynamic noise for 1D constant velocity model.
For the 2D constant velocity model we keep the dynamic noise,Q fixed at 0.01 and we change the measurement noise.

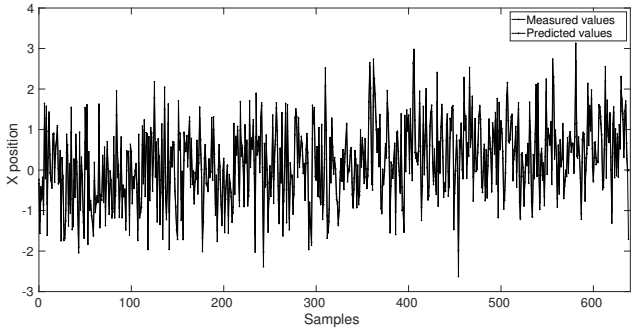

Figure 1: Kalman 1D plot for Ratio 1

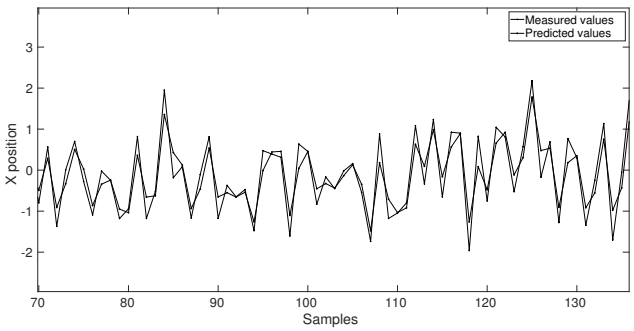

Figure 2: Zoomed Kalman 1D plot for Ratio 1

Figure 1 represents the X-position plot of the given data and the Kalman filter prediction for Ratio 1. Figure 2 represents the zoomed version of the same plot.
For this plot Ratio 1 i.e. **Measurement noise(R) = 1** and **dynamic noise(Q) = 1** is considered.

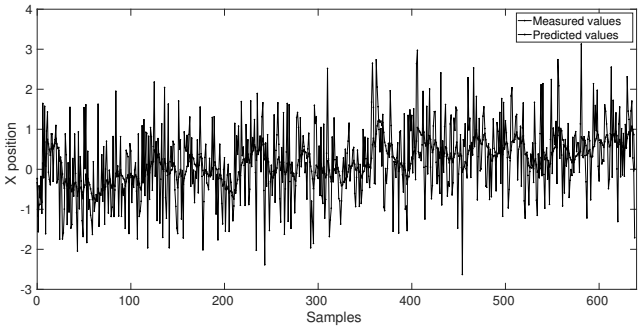

Figure 3: Kalman 1D plot for Ratio 2

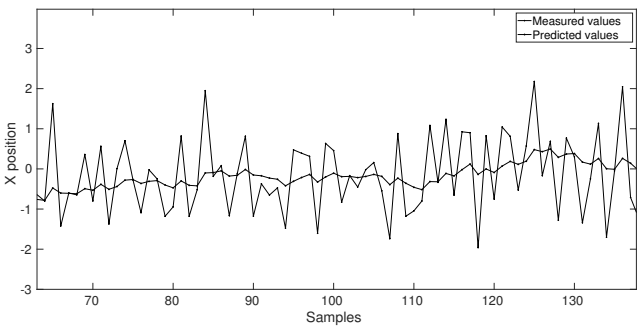

Figure 4: Zoomed Kalman 1D plot for Ratio 2

Figure 3 represents the X-position plot of the given data and the Kalman filter prediction for Ratio 2. Figure 4 represents the zoomed version of the same plot.
For this plot Ratio 2 i.e. **Measurement noise(R) = 1** and **dynamic noise(Q) = 0.0001** is considered.

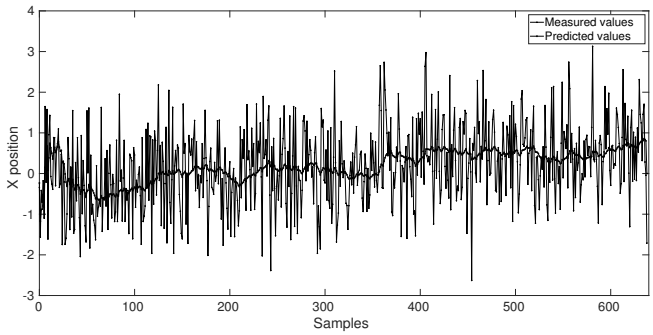

Figure 5: Kalman 1D plot for Ratio 3

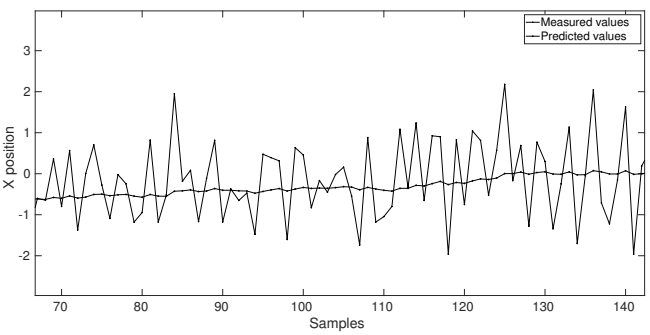

Figure 6: Zoomed Kalman 1D plot for Ratio 3

Figure 5 represents the X-position plot of the given data and the Kalman filter prediction for Ratio 2. Figure 6 represents the zoomed version of the same plot.
For this plot Ratio 3 i.e. **Measurement noise(R) = 1** and **dynamic noise(Q) = 0.000001** is considered.

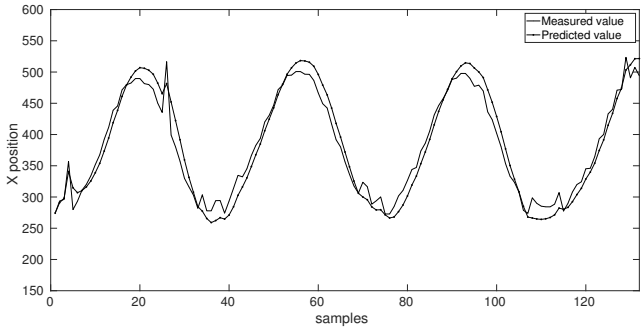

Figure 7: Kalman 2D X-position plot for Ratio 1

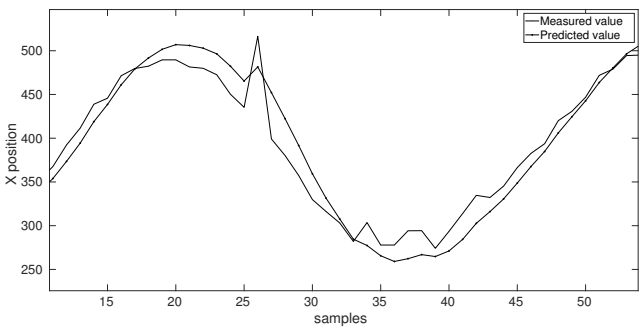

Figure 8: Zoomed X-position plot for Ratio 1

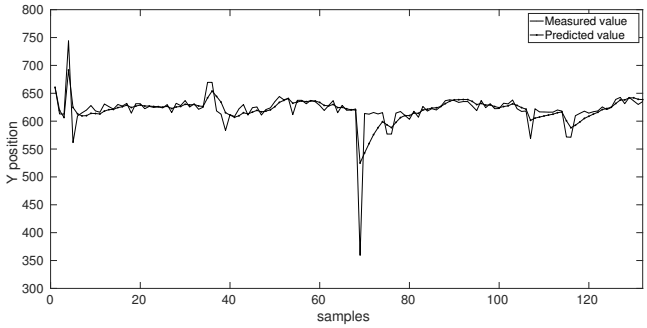

Figure 9: Kalman 2D Y-position plot for Ratio 1

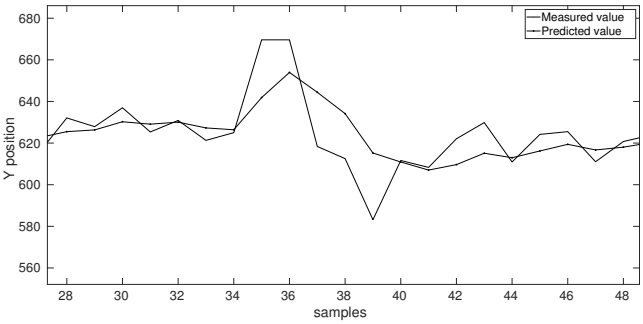

Figure 10: Zoomed Y-position plot for Ratio 1

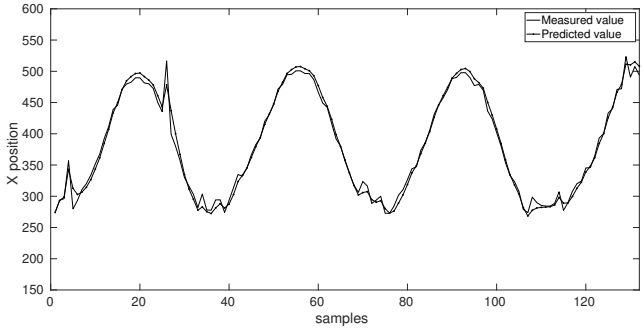

Figure 11: Kalman 2D X-position Ratio 2 plot

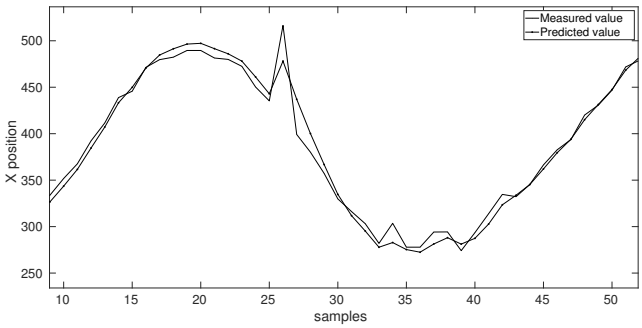

Figure 12: Zoomed X-position plot for Ratio 2

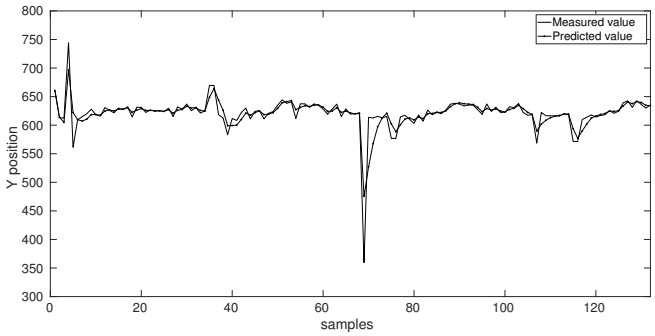

Figure 13: Kalman 2D Y-position Ratio 2 plot

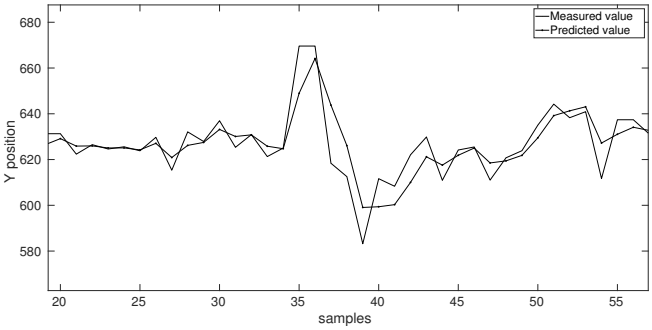

Figure 14: Zoomed Y-position plot for Ratio 2

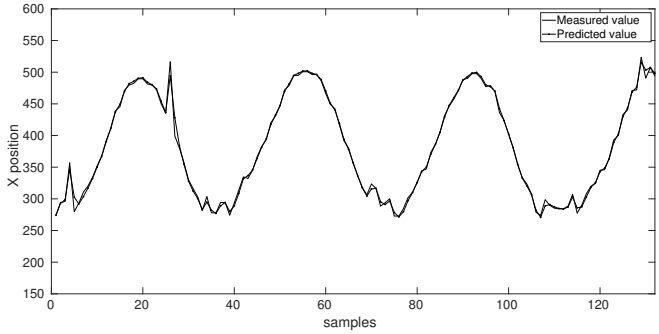

Figure 15: Kalman 2D X-position Ratio 3 plot

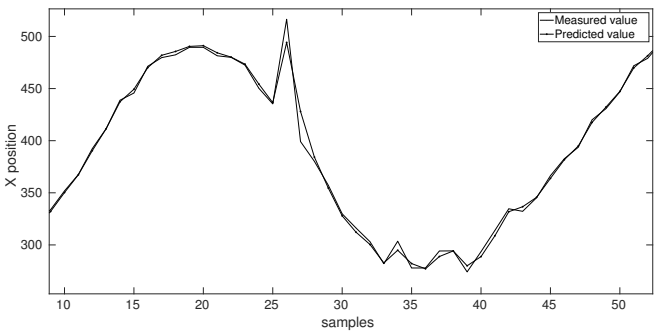

Figure 16: Zoomed X-position plot for Ratio 3

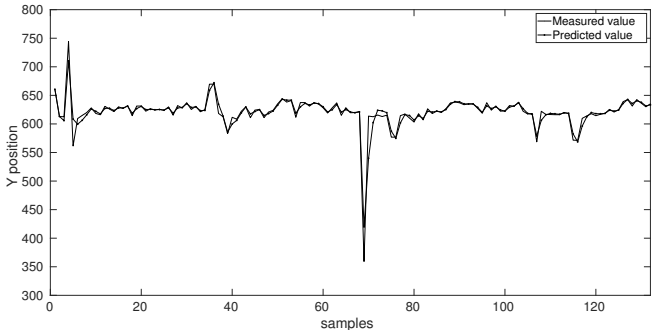

Figure 17: Kalman 2D Y-position Ratio 3 plot

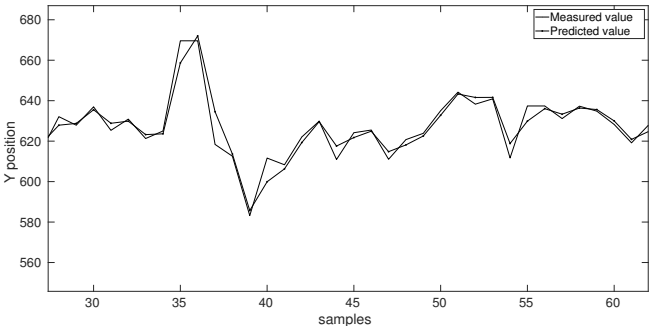

Figure 18: Zoomed Y-position plot for Ratio 3

Figure 7 and Figure 9 represents the X-position and Y-position plot of the 2D dataset and the Kalman filter prediction. For this plot Ratio 1 i.e.**Measurement noise(R) = 0.1** and **dynamic noise(Q) = 0.001** is considered.

Figure 11 and Figure 13 represents the X-position and Y-position plot of the 2D dataset and the Kalman filter prediction. For this plot Ratio 2 i.e.**Measurement noise(R) = 0.01** and **dynamic noise(Q) = 0.001** is considered.

Figure 15 and Figure 17 represents the X-position and Y-position plot of the 2D dataset and the Kalman filter prediction. For this plot Ratio 3 i.e.**Measurement noise(R) = 0.001** and **dynamic noise(Q) = 0.001** is considered.

Please note, dark and thick lines in all plots above represent the Predicted values and the thin lines represent the measurement values

4 Conclusion

The goal of this lab was to design a Kalman filter to track or predict the measurements values in the given dataset. This report begins with the description and necessity of Kalman Filter. Then the kalman filter parameters were derived for 1D and 2D constant velocity models. Using the models, implementation equations were defined. The model was coded using MATLAB and the results were plotted. The data-set used for 1D and 2D modeling are "1D-data.txt" and "2D-UWB-data.txt" respectively.

For the constant 1D velocity model, filter result of three different ratios of measurement noise to dynamic noise were plotted. Theoretically, with the reduction in dynamic noise, the prediction must move away from the measurement values. Here we fixed the measurement noise(R) to 1 and tuned the dynamic noise(Q) value. It can be observed in Figure 1 where $\mathbf{Q = 1}$, the prediction is very close to the measurement values. Figure 2 shows a zoomed version of the plot, where it is easier to see this result. As we reduce the Q value to **0.0001** and **0.000001** the prediction sways away from the measurement values. This can be observed in Figure 4 and Figure 6.

For the constant 2D model, the dynamic noise,**Q was fixed at 0.01** and the measurement noise,R was tuned. Theoretically, with the reduction in measurement noise, the prediction must move towards the measured value. Three R values, $\mathbf{R = 0.1}$, $\mathbf{R = 0.01}$ and $\mathbf{R = 0.001}$ were considered. It can be observed that the prediction does move towards measured values as R decreases. Figure 7 through Figure 18 graphically shows this observation.

Thus, we can conclude that the R : Q ratio affects the Kalman gain, which in turn makes the prediction go towards or farther away from the measured sensor reading. As the measurement noise reduces, sensor reading is more reliable and due to this the prediction comes closer to sensor reading and vice versa. Similarly the reduction of dynamic noise means the actual position of the object that we are tracking moves in a very predictable manner. Thus the sensor reading gets lower priority while predicting the next state.

5 Appendix

5.1 MATLAB Code

Part 1 1D Constant Velocity model.

```
clc
clear
close all

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

%Initialization
t = 1; %Time
```

```
10 noise_measure = 1; %Measurement noise
11 noise_dynamic = 0.000001; %Dynamic nosie
12
13 ST = [1 t ; 0 1]; %State transition matrix
14 M = [1 0; 0 0]; %Observation matrix
15 X_tPrev = [0 ; 0]; %State matrix
16 R = [noise_measure 0.1; 0.1 0.1 ]; %Co-Variance of Measurement noise
17 Q = [0 0 ; 0 noise_dynamic]; %Co-variance of Dynamic noise
18 k_t = [0 0; 0 0]; %Kalman gain
19 I = [1 0; 0 1]; %Identity matrix
20 S_tPrev = I; %State Co-variance
21 predicted_Data = zeros(1, length(Data)); %Output
22 yt = [Data' ; zeros(1, length(Data)) ]; %Observation variables
23
24
25
26 %Kalman Filter Loop
27  while(t < length(Data) )
28
29       X_tNext = ST * X_tPrev;
30
31       S_tNext = (ST * S_tPrev * ST') + Q ;
32
33       k_t = (S_tNext * M') / ( M * S_tNext * M' + R );
34
35       X_pred = X_tNext + (k_t * (yt(:,t) - (M * X_tNext) ));
36
37       S_pred = (I - (k_t * M) ) * S_tNext ;
38
39       predicted_Data(t) = X_pred(1,1); %Store for plotting
40
41       X_tPrev = X_pred;
42       S_tPrev = S_pred;
43
44       t = t + 1 ;
45
46  end %end of while
47
48  %Plotting
49
50 x = 0:length(Data)-1;
51
52 figure(1)
53 plot(x,Data,"kx-","markersize",3) ;
```

```
hold on
plot(x, predicted_Data, "k.-","markersize",12,"Linewidth",2);
hold off
legend("Measured values","Predicted values");
xlabel("Samples");
ylabel("X position");
axis([0 640 -3 4])
set(gca,"FontSize",28)



%Ratio 1
%Measure:Dynamic = 1: 1
%Ratio 2
%Measure:Dynamic = 1: 0.0001
%Ratio 3
%Measure:Dynamic = 1: 0.000001
```

Part 2 2D Constant Velocity model.

```

clc
clear all

%Import Data
Data = dlmread("2D_UWB_data.txt");

%Initialization
t = 1; %Time
noise_measure1 = 0.001; %Measurement noise1
noise_measure2 = 0.001; %Measurement noise2

noise_dynamic1 = 0.001; %Dynamic nosie1
noise_dynamic2 = 0.001; %Dynamic nosie2

ST = [1 0 t 0 ; 0 1 0 t; 0 0 1 0; 0 0 0 1]; %State transition matrix
M = [1 0 0 0 ; 0 1 0 0]; %Observation matrix
X_tPrev = [Data(1,1) ; Data(1,2) ; 0 ; 0]; %State matrix
R = [noise_measure1 0.00001; 0.00001 noise_measure2 ]; %Co-Variance of
    Measurement noise
Q = [0 0 0 0; 0 0 0 0; 0 0 noise_dynamic1 0.00001; 0 0 0.00001
   noise_dynamic2]; %Co-variance of Dynamic noise
k_t = zeros(4,2); %Kalman gain
I = [1 0 0 0; 0 1 0 0; 0 0 1 0; 0 0 0 1]; %Identity matrix
```

```
S_tPrev = [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 Co-variance
predicted_Data = zeros(2, length(Data(:,1))); %Output
yt = Data'; %Observation variables

count = 0

%Kalman Filter Loop
 while(t < length(Data(:,1) ))

      X_tNext = ST * X_tPrev;

      S_tNext = (ST * S_tPrev * ST') + Q;

      k_t = (S_tNext * M') / ( M * S_tNext * M' + R );

      X_pred = X_tNext + (k_t * (yt(:,t) - (M * X_tNext) ));

      S_pred = (I - (k_t * M) ) * S_tNext ;

      predicted_Data(1,t) = X_pred(1,1); %Store for plotting
      predicted_Data(2,t) = X_pred(2,1); %Store for plotting

      X_tPrev = X_pred;
      S_tPrev = S_pred;

      t = t + 1 ;

 end %end of while

 %Plotting
x = 1:length(Data(:,1));

figure(1)
plot(x,Data(:,1),"k.-","markersize",3) ;
hold on
plot(x,predicted_Data(1,:), "k.-","markersize",12,"linewidth",2);
hold off
legend("Measured value","Predicted value")
xlabel("samples");
ylabel("X position");
axis([0 132 150 600])
set(gca,"FontSize",28)

```

```
66
67 figure(2)
68 plot(x,Data(:,2),"k.-","markersize",3) ;
69 hold on
70 plot(x,predicted_Data(2,:), "k.-","markersize",12,"linewidth",2);
71 hold off
72 xlabel("samples");
73 ylabel("Y position");
74 legend("Measured value","Predicted value")
75 axis([0 132 300 800])
76 set(gca,"FontSize",28)
77
78 % Ratio 1
79 % Measured:Dynamic 0.1:0.001
80 % Ratio 2
81 % Measured:Dynamic 0.01:0.001
82 % Ratio 3
83 % Measured:Dynamic 0.001:0.001
```