Clemson University

Analysis of Tracking Systems

ECE 8540

Lab-4 Report

Submitted by:

Priyanshu Rawat

October 13, 2022

Contents

1 Introduction

Kalman filter is an algorithm that provides estimates of some unknown variables according to the measurements observed over time. The Kalman filter algorithm consists of two stages: prediction and correction (update).

Input : x_{k-1}, P_{k-1}, u_k, z_k
Output : x_k^-, P_k^-

Prediction

$x_{k+1} = \Phi(x)_{k-1} + Bu_k + G\omega_k$;
$P_k = \Phi(P)_{k-1}\Phi^T + Q$;

Correction

$K = PH^T(HPH^T + R)^{-1})$;
$z = (z_k - H_k x_k)$;
$x_k^- = x_k + kz$;
$P_k^- = P - KHP$;

return x_k^-, P_k^-;

Algorithm 1: Kalman Filter

1.1 Problem Statement

Develoving code to operate Kalman filter for each of the problems :

1.1.1 Problem 1: One dimensional position tracking

Estimate position in the given data using the constant velocity 1-D model. Compare results for three different ratios of dynamic noise to measurement noise. Discuss the differences between the outputs.

1.1.2 Problem 2: Two dimensional position tracking

Estimate position in the UWB tracking data using the constant velocity 2-D model. Compare results for three different ratios of dynamic noise to measurement noise. Discuss the differences between the outputs.

2 Results and discussion

The solutions of both the problems are discussed here.

2.1 One dimensional position tracking

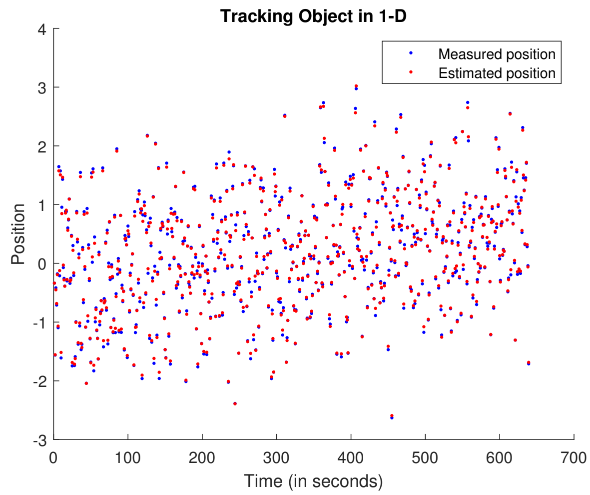

Figure 1:
Position estimation when, $R = 0.01, \quad Q = \begin{bmatrix} 0 & 0.005 \\ 0 & 0.005 \end{bmatrix}$

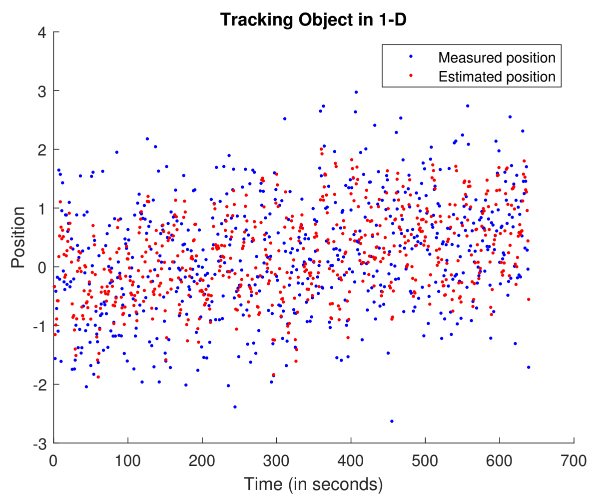

Figure 2:
Position estimation when, $R = 1, \quad Q = \begin{bmatrix} 0 & 0.005 \\ 0 & 0.005 \end{bmatrix}$

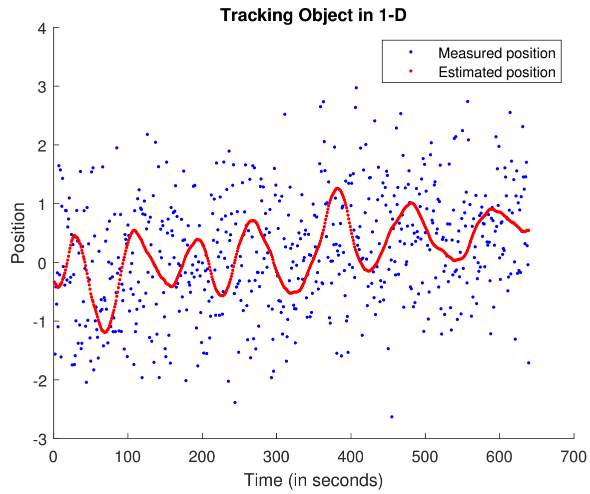

Figure 3:
Position estimation when, $R = 100, \quad Q = \begin{bmatrix} 0 & 0.005 \\ 0 & 0.005 \end{bmatrix}$

Here, the position of an object moving in 1-D with constant velocity is tracked using the implemented Kalman Filter.

The dynamic noise co-variance matrix Q was fixed to

$$Q = \begin{bmatrix} 0 & 0.005 \\ 0 & 0.005 \end{bmatrix}$$

while the measurement noise co-variance value R was adjusted from *0.01*, to *1*, to *100*.

Figure 1 shows the position estimates obtained from the Kalman filter when R was *0.01*. Whereas, figure 2 and 3, show the position estimates obtained from the Kalman filter when R was *1* and *100* respectively.

From the figures 1,2 and 3; it can be seen that as the value of R increases suggesting more noise in measurements, the estimated position values lie closer to the predictions then the measurements. On the other hand, when the value of R was less say, 0.01. As seen in figure 1, the estimated position values lie closer to the measurements than predictions resulting in a much better tracking.

This is because higher value of R suggests more uncertainty in the measurements which makes the Kalman estimates rely more towards the predictions rather than the measured values.

2.2 Two dimensional position tracking

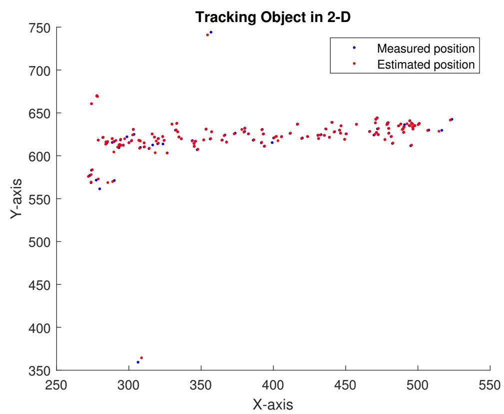

Figure 4:

Position estimation when, $R = \begin{bmatrix} 0.01 & 0.005 \\ 0.005 & 0.01 \end{bmatrix}$, $Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.005 \\ 0 & 0 & 0.005 & 0 \end{bmatrix}$

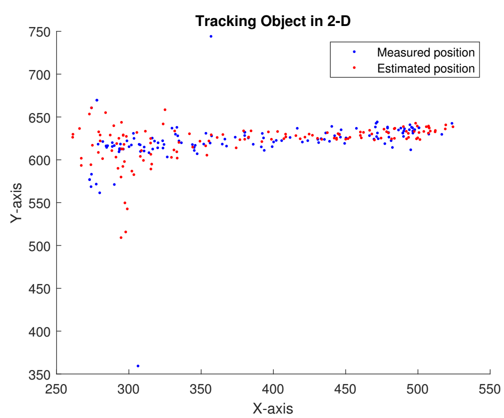

Figure 5:

Position estimation when, $R = \begin{bmatrix} 1 & 0.005 \\ 0.005 & 1 \end{bmatrix}$, $Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.005 \\ 0 & 0 & 0.005 & 0 \end{bmatrix}$

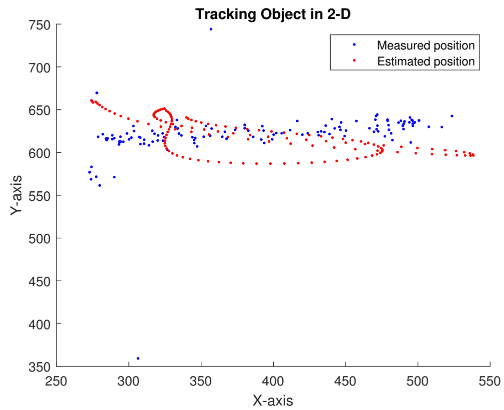

Figure 6:

Position estimation when, $R = \begin{bmatrix} 100 & 0.005 \\ 0.005 & 100 \end{bmatrix}$, $Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.005 \\ 0 & 0 & 0.005 & 0 \end{bmatrix}$

Here, the position of an object moving in 2-D with constant velocity is tracked using the implemented Kalman Filter. The UWB tracking data from the course website was used for this problem.

The dynamic noise co-variance matrix Q was fixed to

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.005 \\ 0 & 0 & 0.005 & 0 \end{bmatrix}$$

while the measurement noise co-variance value R was adjusted from

$$R = \begin{bmatrix} 0.01 & 0.005 \\ 0.005 & 0.01 \end{bmatrix}, \text{ to } R = \begin{bmatrix} 1 & 0.005 \\ 0.005 & 1 \end{bmatrix}, \text{ to } R = \begin{bmatrix} 100 & 0.005 \\ 0.005 & 100 \end{bmatrix}$$

Figure 4 shows the position estimates obtained from the Kalman filter when variance in readings along position x and y was *0.01*. Whereas, figure 5 and 6, show the position estimates obtained from the Kalman filter when variance in readings along position x and y, was *1* and *100* respectively.

Again, from the figures 4,5 and 6; it can be seen that as the value of R increases suggesting more noise in measurements, the estimated position values lie closer to the predictions then the measurements. On the other hand, when the value of R was less say, 0.01. As seen in figure 4, the estimated position values lie closer to the measurements than predictions resulting in a much better tracking.

This is because higher value of R suggests more measurement noise which makes the Kalman estimates rely more towards the prediction rather than the measured values.

3 Conclusion

To conclude, as the ratio of dynamic noise to measurement noise decreases the position estimates obtained from Kalman filter rely more on the predictions than on the measurements.

4 Appendix

4.1 Code: Kalman filter (1-D) problem

```
clc;
clear;

% Loading the data in matlab
data = load("1D-data.txt");

% Assuming the data is recorded after every 1 seconds
timeseries = [1:1:numel(data)];

% Plotting the raw data
figure(1)
scatter(timeseries, data,'.','blue')
title('Measured Position (1D)')
xlabel('Time (in seconds)')
ylabel('Position')

dt = 1; % 1 second interval
n_states = 2; % number of state variables (position and velocity)
P = [1,dt;0,1]; % state transition matrix
M = [1,0]; % observation matrix
Q = [0 0.005
     0 0.005]; % dynamic uncertainity matrix
R = 1000; % measurement noise covariance
S = eye(2); % initial variance
X = [data(1); 0]; % initial state
Y =  data; % measurements
X_est(:,1) = [data(1); 0]; % position estimate matrix

for i = 2:numel(data)
    %% Prediction
    X_pred = P*X; % predict next stage
    S_pred = P*S*P' + Q; % predicting state variance

    %% Correction
    K = S_pred*M'/((M*S_pred*M'+R));
    X = X_pred+K*(Y(i)-M*X_pred);
    S = (eye(n_states)-K*M)*S;
    %% Storing estimation
    X_est(:,i) = X;
end
```

```
% Plotting the estimates
hold on
% scatter(samples,X, '.')
scatter(timeseries,X_est(1,:),'.','red');
title('Tracking Object in 1-D')
xlabel('Time');
ylabel('Position');
legend('Measured position', 'Estimated position')
hold off
```

4.2 Code: Kalman filter (2-D) problem

```
clc;
clear;

% Loading the data in matlab
data = load("2D-UWB-data.txt");
x = data(:,1); % saving x data
y = data(:,2); % saving y data

% Plotting the raw data
figure(2)
scatter(x,y,'.','blue')
title('Measured Position (2D)')
xlabel('X-axis')
ylabel('Y-axis')

dt = 1; % 1 second interval
n_states = 4; % state variables (position and velocity in x&y)
P = [1 0 dt 0; % state transition matrix
     0 1 0 dt;
     0 0 1 0;
     0 0 0 1];
M = [1 0 0 0;
     0 1 0 0]; % observation matrix
Q = [0 0 0 0; % dynamic covariance matrix
     0 0 0 0;
     0 0 0 0.005;
     0 0 0.005 0];
R = [1 0.005; % measurement noise covariance matrix
     0.005 1];
S = eye(4); % initial variance]
X = [data(1,1); % initial state
     data(1,2);
```

```
     0;
     0];
Y =  data';% measurement
X_est(:,1) = X; % position estimate matrix

for i = 2:length(data)
    %% Prediction
    X_pred = P*X; % predict next stage
    S_pred = P*S*P' + Q; % predicting state variance

    %% Correction
    K = S_pred*M'/((M*S_pred*M'+R));
    X = X_pred+K*(Y(:,i)-M*X_pred);
    S = (eye(n_states)-K*M)*S;
    %% Storing estimation
    X_est(:,i) = X;
end

% Plotting the estimates
hold on
scatter(X_est(1,:),X_est(2,:),'.','red')
title('Tracking Object in 2-D')
xlabel('Time');
ylabel('Position');
legend('Measured position', 'Estimated position')
hold off
```