LAB-4 KALMAN FILTER

Madhumita Krishnan

10/9/2018

1 Introduction

In this lab we apply kalman filter to the data to provide an efficient model to estimate the state of any process. Estimation can be done even when the precise nature of the modeled system is unknown.

The report contains two sections. For the first section, the data provided is used to develop a constant velocity 1D model kalman filter and for the second part of the lab, we choose an UWB tracking data to develop a constant velocity 2D model.

We begin writing the mathematical equations for the continuous cycle of predict-update in matrix notation.

We then implement the Kalman filter in MATLAB and discuss the results.

2 Method

Kalman filter algorithim can be organized under the following stages

1. Prediction: We make a prediction of the state based on some previous values and model.
2. Measurement: We obtain the measurement of the state from the sensors.
3. Update: We update our prediction based on our errors.
4. Repeat.

The equations for the above process are

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

$$S_{t,t-1} = \phi S_{t-1,t-1} \phi^T + Q \tag{2}$$

$$K_t = S_{t,t-1} M^T [M S_{t,t-1} M^T + R]^{-1} \tag{3}$$

$$X_{t,t} = X_{t,t-1} + K_t [Y_t - M X_{t,t-1}] \tag{4}$$

$$S_{t,t} = (I - K_t M) S_{t,t-1} \tag{5}$$

2.1 1D Tracking

Consider an object moving along X axis with constant velocity. let T represent the time interval between sensor readings. The state variables are $\begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix}$ where x_t provides the position and $\dot{x}_t$, the velocity at time t. The state transition equations can be written as

$$x_t = x_{t-1} + T\dot{x_{t-1}} \tag{6}$$

$$\dot{x}_t = \dot{x_{t-1}} \tag{7}$$

The state transition matrix ϕ and the observation matrix are $\phi = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}$ and $M = \begin{bmatrix} 1 & 0 \end{bmatrix}$ .The filter results for different ratios of dynamic noise Q to measurement noise R are plotted.

2.2 2D Tracking

The state space equations that describe the position and velocity of a 2D particle is given as

$$x_t = x_{t-1} + T\dot{x_{t-1}} \tag{8}$$

$$y_t = y_{t-1} + T\dot{y_{t-1}} \tag{9}$$

$$x_t = \dot{x_{t-1}} \tag{10}$$

$$y_t = \dot{y_{t-1}} \tag{11}$$

The state vector is $\begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix}$. The state transformation matrix $\phi = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ and $M = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$ Equations for kalman's filter can be constructed for varying values of Q and R.

3 Results

3.1 1D Tracking

Using the equations in Section 2.1 we can track the postion of the 1D particle which is shown in Figure 1.

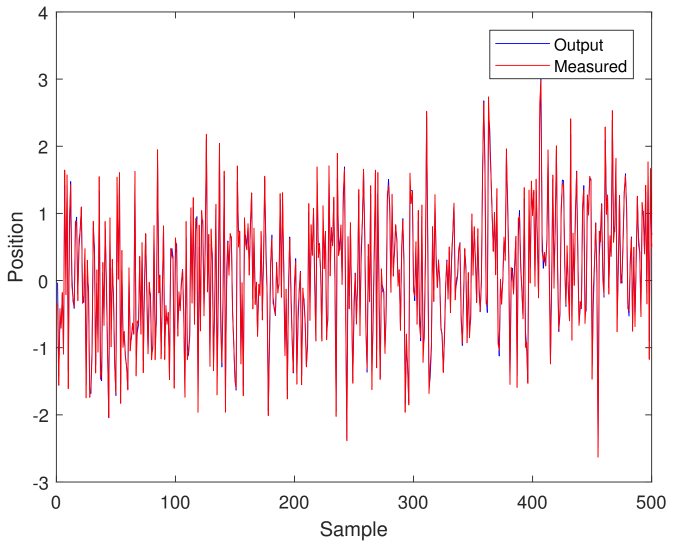

Figure 1: 1D Tracking in Kalman

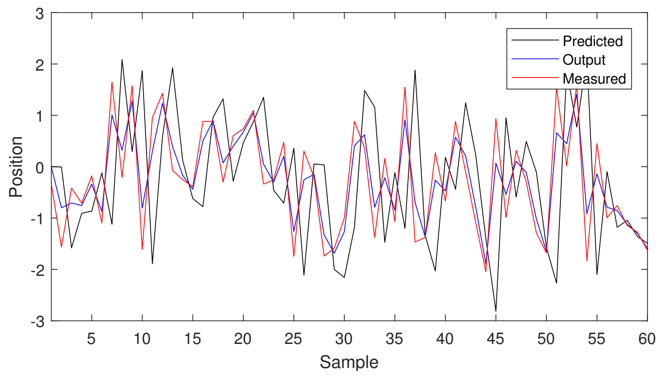

Figure 2: Q=0.00001 and R=0.01

The output of the kalman filter which is almost consistent with the measured value y_t can be seen in Figure 1. The graph is obtained for $Q_{11} = Q_{22} = 0.001, Q_{12} = Q_{21} = 0.0001$ and R=[0.0001]. The filter outputs for other different values of Q and R can be seen in Figures 2 and 3.

We can see from Figures 2, 3 that as the value of Q becomes smaller the kalman output stops tracking the measured value and it moves in between the predicted and the measured value. Again larger Q implies less concern about the changes in the states.

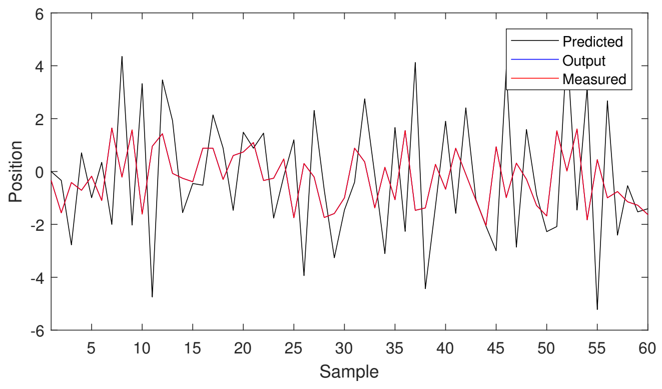

Figure 3: Q=1 and R=0.00001

We can see that in Figure 3 the estimation is relatively straight forward. It is almost consistent with the measured values.

3.2 2D Tracking

Using the equations in Section 2.2 we can track the position of the 2D particle as seen in Figure 4. We can see in figure, the output of kalman filter which is almost consistent with the measured value y_t. These graphs were obtained for $Q_{33} = Q_{44} = 0.0001, Q_{34} = Q_{43} = 0.00001$ and $R_{11} = R_{22} = 0.01, R_{12} = R_{21} = 0.00001$.The filter is tuned for different values of Q and R as seen in Figures 5 and 6.

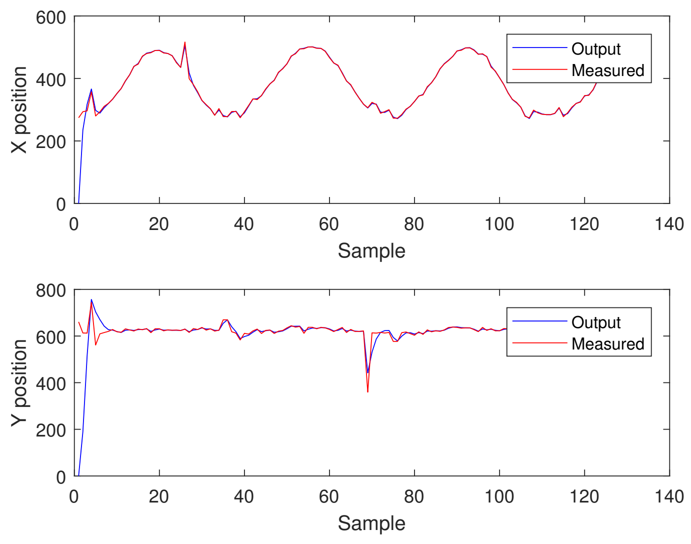

Figure 4: 2D Tracking in Kalman

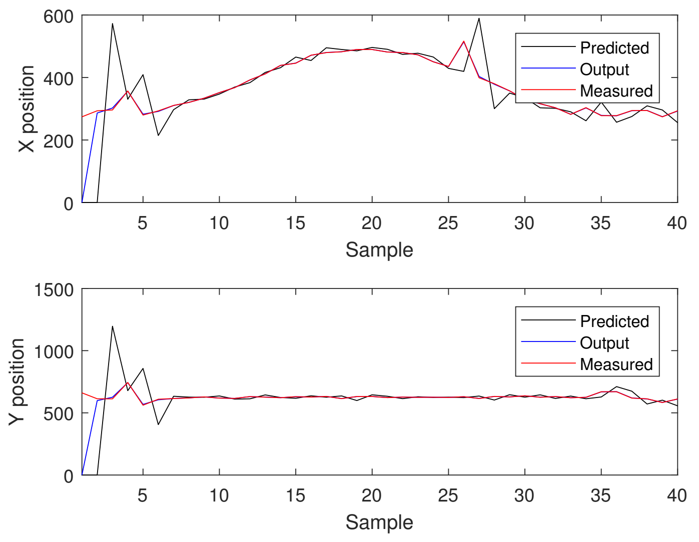

Figure 5: Q=0.001 and R=0.0001

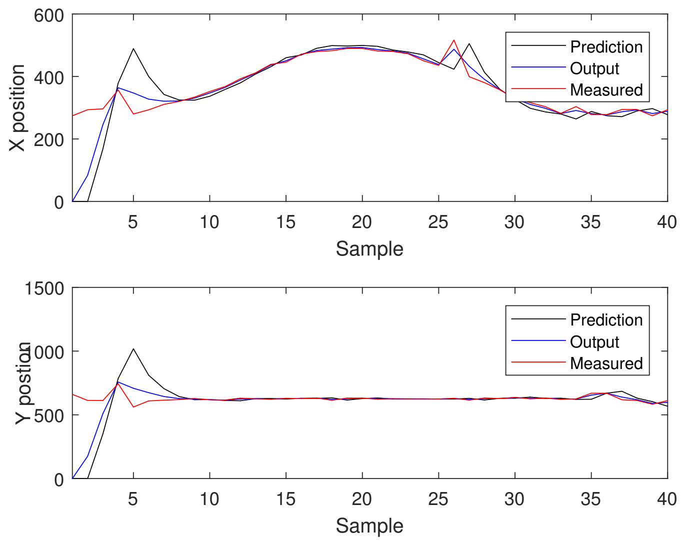

Figure 6: Q=0.0001 and R=0.01

We can see the kalman filter's output and prediction when compared to the measured values in Figures 5 and 6. We observe from Figures 4, 5, 6 that as Q becomes smaller the kalman filter slowly stops tracking the measured value.

4 Conclusion

The main advantage of the kalman filter is its ability to provide an optimal estimate of the state of any process with relatively low complexity. However the limitation in the use of this filter is that it provides accurate results only for Gaussian and Linear models. For Gaussian models with limited non-linearity, Extended Kalman filter will be appropriate. For non-Gaussian non-linear models other methods like particle filtering is the most appropriate approach.

5 Appendix

5.1 Code for 1D Tracking

```
x=zeros(2,1);
xt=zeros(2,1);
s=zeros(2,2);
st=zeros(2,2);
```

```
kt=zeros(2,1);
i=zeros(2,2);
i=eye(2);
T=10;
phi=[1 T;0 1];
Q=[.0001 0.00001;0.00001 .0001];
R=0.01;
M=[1 0] ;
t=1;

while t<=639
    xt=phi*x;           %Predict-update cycle
    st=(phi*s*transpose(phi))+Q;
    yt=Ddata{t,1};
    kt=st*transpose(M)*inv(M*st*(transpose(M))+R)
    x=xt+kt*(yt-(M*xt));
    s=(i-(kt*M))*st;
    y(t,1)=yt(1,1);     %values are stored in an array
    xt1(t,1)=xt(1,1);
    g(t,1)=x(1,1);
    t1(t,1)=t;

      t=t+1;
end
plot(t1,xt1,'k',t1,g,'b',t1,y,'r');
    xlabel('Sample')
    ylabel('Position')
    legend('Predicted','Output','Measured')
```

5.2 Code for 2D Tracking

```
x=zeros(4,1);
xt=zeros(4,1);
s=zeros(4,4);
st=zeros(4,4);
R=zeros(1,1);
kt=zeros(2,1);
i=eye(4);
T=10;
```

```
phi=[1 0 T 0;0 1 0 T;0 0 1 0;0 0 0 1];
Q=[0 0 0 0;0 0 0 0;0 0 .0001 .00001  ;0 0 .00001 .0001 ];
R=[.0001 .00001;.00001 .001];
M=[1 0 0 0;0 1 0 0] ;
t=1;
figure(1)
while t<=134
    xt=phi*x;           %Predict-Update cycle
    st=(phi*s*transpose(phi))+Q;
    yt=[DUWBdata{t,1};DUWBdata{t,2}];
    kt=st*transpose(M)*inv(M*st*(transpose(M))+R);
    x=xt+kt*(yt-(M*xt));
    s=(i-(kt*M))*st;
    xt1(t,1)=xt(1,1);   %values are stored in an array
    xt2(t,1)=xt(2,1);
    y(t,1)=yt(1,1);
    y1(t,1)=yt(2,1);
    g(t,1)=x(1,1);
    h(t,1)=x(2,1);
    t1(t,1)=t;
    t=t+1;
end
subplot(2,1,1)
    plot(t1,g,'b',t1,y,'r');
    xlabel('Sample')
    ylabel('X position')
    legend('Output','Measured')
    hold on
    subplot(2,1,2)
    plot(t1,h,'b',t1,y1,'r');
    xlabel('Sample')
    ylabel('Y position')
    legend('Output','Measured')
hold off
```