Lab 5 - Extended Kalman Filter

Submitted By:
Shashi Shivaraju
C88650674

Clemson University
October 25, 2018

Abstract

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

1 Introduction

This report considers the problem of designing a extended Kalman Filter for mitigating the noises from the sensor measurements which are used to track an object modelled using non-linear equations.

A filter is a mathematical tool that uses an expected dynamic model to help mitigate noise in sensor data.Kalman filtering 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 extended Kalman filter (EKF) is the nonlinear version of the Kalman filter. The EKF works by transforming the nonlinear models at each time step into linearized systems of equations. The extended Kalman filter has numerous applications and is considered to be the standard in the theory of nonlinear state estimation, navigation systems and GPS.

This report describes extended Kalman filtering for given data set "sin-data.txt" by considering the model to be of sinusoidal nature.

2 Methods

Similar to Kalman filtering,extended Kalman filtering is also a continuous cycle of predict-update. It involves three main steps:prediction,measurement and updation. Before we apply the extended 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 extended Kalman filter is implemented. The design and implementation of extended Kalman filter for the given data-set is described in below section.

2.1 Extended Kalman filter design for sinusoidal model

The dataset "sin-data.txt" describes the position of an object moving in sinusoidal pattern.

Let the state transition equations for this model be:

$$f(x_t, a_t) = \begin{bmatrix} x_{t+1} = x_t + \dot{x}_t T \\ \dot{x}_{t+1} = \dot{x}_t + a_t \\ h_{t+1} = \sin \frac{x_t}{10} \end{bmatrix} \tag{1}$$

where x represents time,$\dot{x}$ along with dynamic noise a_t represents the uncertainty in the propagation of the sinusoid over time and the variable h provides the actual value of the sinusoid. The dynamic noise a_t is a random sample drawn from $N(0, \sigma_a^2)$.

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

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \\ h_t \end{bmatrix} \tag{2}$$

Since the given data is the x positions, let us denote the sensor readings representing the current height of the sinusoid d_t:

$$Y_t = \begin{bmatrix} d_t \end{bmatrix} \tag{3}$$

The observation equations for this model is:

$$g(x_t, n_t) = \begin{bmatrix} d_t = h_t + n_t \end{bmatrix} \tag{4}$$

where n_t is a random sample drawn from $N(0, \sigma_n^2)$ representing measurement noise.

Four Jacobians are calculated to use this model in extended Kalman Filter. The derivative of the state transition equations with respect to the state variables is:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial (x, \dot{x}, h)} = \begin{bmatrix} 1 & T & 0 \\ 0 & 1 & 0 \\ \frac{1}{10} \cos \frac{x}{10} & 0 & 0 \end{bmatrix} \tag{5}$$

The derivative of the state transition equations with respect to the dynamic noises is:

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial (0, a_t, 0)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{6}$$

The derivative of the observation equations with respect to the state variables is:

$$\frac{\partial g}{\partial x} = \frac{\partial g}{\partial (x, \dot{x}, h)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{7}$$

The derivative of the observation equations with respect to the measurement noises is:

$$\frac{\partial g}{\partial n} = \frac{\partial g}{\partial (n_t)} = \begin{bmatrix} 1 \end{bmatrix} \tag{8}$$

The covariance of the dynamic noises is:

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma_a^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{9}$$

The covariance of the measurement noises is

$$R = \begin{bmatrix} \sigma_n^2 \end{bmatrix} \tag{10}$$

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

The implementation of extended Kalman filter for this model is explained below.

2.2 Implementation of extended Kalman Filter

The extended Kalman filter is a continuous cycle of predict-update. The following equations form the main loop of the extended kalman filter applied to the model described above:

1. Predict next state

$$\mathrm{X}_{t,t-1} = f(X_{t-1,t-1}, 0) \tag{11}$$

   where $f(X_{t-1,t-1}, 0)$ is the approximated state $\tilde{x}_t$.

2. Predict next state covariance

$$\mathrm{S}_{t,t-1} = \left(\frac{\partial f}{\partial x}\right) S_{t-1,t-1} \left(\frac{\partial f}{\partial x}\right)^T + \left(\frac{\partial f}{\partial a}\right) Q \left(\frac{\partial f}{\partial a}\right)^T \tag{12}$$

   where $\left(\frac{\partial f}{\partial x}\right)$ and $\left(\frac{\partial f}{\partial a}\right)$ are the Jacobians of the state transition equations. The notation $(...)^T$ indicates matrix transpose.

3. Obtain measurement(s) Y_t

4. Calculate the Kalman gain (weights)

$$\mathrm{K}_t = S_{t,t-1} \left(\frac{\partial g}{\partial x}\right)^T \left[\left(\frac{\partial g}{\partial x}\right) S_{t,t-1} \left(\frac{\partial g}{\partial x}\right)^T + \left(\frac{\partial g}{\partial n}\right) R \left(\frac{\partial g}{\partial n}\right)^T\right]^{-1} \tag{13}$$

   where $\left(\frac{\partial g}{\partial x}\right)$ and $\left(\frac{\partial g}{\partial n}\right)$ are the Jacobians of the measurement equations.

5. Update state

$$\mathrm{X}_{t,t} = X_{t,t-1} + K_t[Y_t - g(\tilde{x}_t, 0)] \tag{14}$$

   where $g(\tilde{x}_t, 0)$ is the ideal (noiseless) measurement of the approximated state from above.

6. Update state covariance

$$\mathrm{S}_{t,t} = \left[I - K_t \left(\frac{\partial g}{\partial x}\right)\right] S_{t,t-1} \tag{15}$$

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 extended Kalman filter for model described above.

3 Results

The extended Kalman filter was implemented for sinusoidal models and the values of dynamic noise co-variance matrix Q and measurement noise co-variance matrix R were varied to observe the variations in extended Kalman filter output. For performance analysis of this extended kalman filter,we kept the dynamic noise co-variance matrix Q constant and varied the measurement noise co-variance matrix R.

1. Measurement Noise [MN] set to 0.01 and Dynamic Noise [DN] set to 0.001

   This setting of noises produces an extended Kalman filter output which is smoother than the sensor measurements but is jumpy in nature. The filter output looks to follow the sensor measurements more. The corresponding sensor measurements,extended kalman filter output and actual position is shown in figure(1).

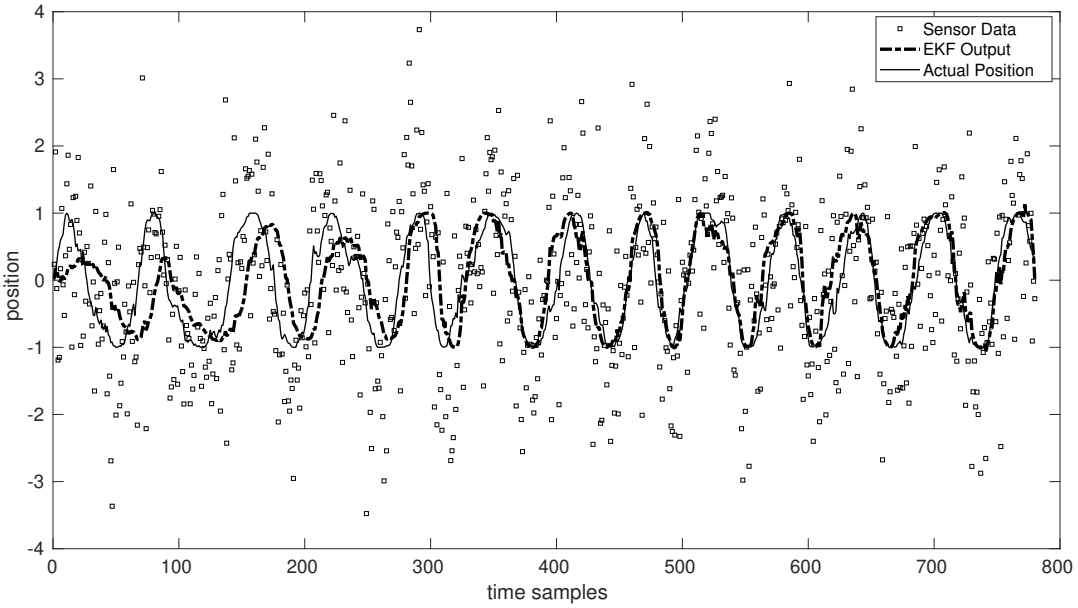

Figure 1: MN=0.01 & DN=0.001

2. Measurement Noise [MN] set to 0.1 and Dynamic Noise [DN] set to 0.001

   This setting of noises produces an extended Kalman filter output which is smoother than the filter output is shown in figure(1). Since the Measurement Noise is increased the filter output tries to follow the predictions than the measurements. This seems to be the best combination of Measurement Noise and Dynamic Noise for this given dataset. The corresponding sensor measurements,kalman filter output and actual position is shown in figure(2).

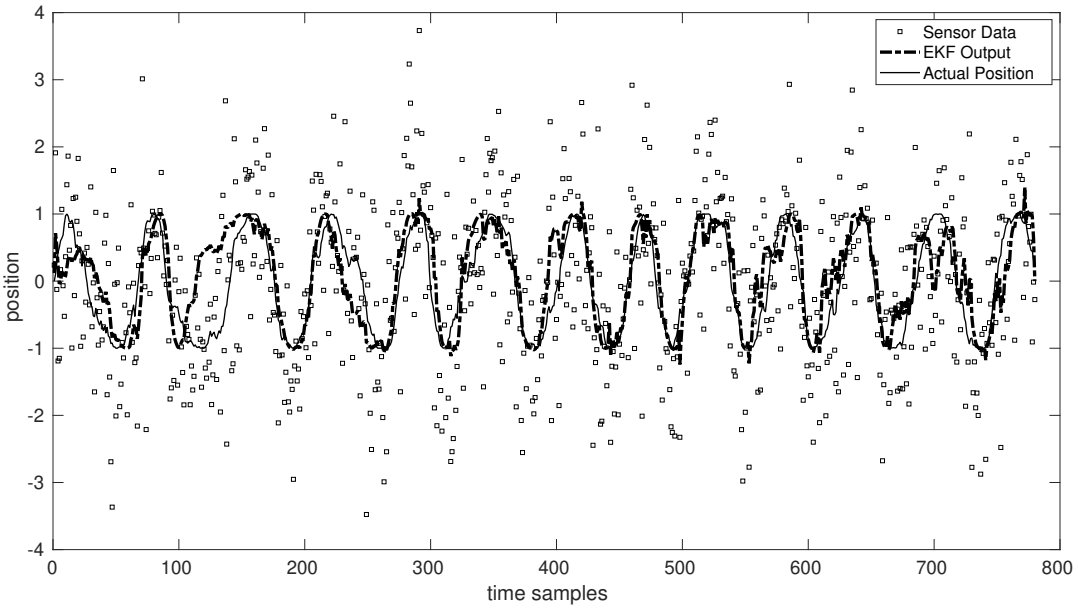

Figure 2: MN=0.1 & DN=0.001

3. Measurement Noise[MN] set to 0.5 and Dynamic Noise [DN] set to 0.001

   This setting of noises produces a extended Kalman filter output which is smoother than the filter output is shown in figure(2). Since the Measurement Noise is increased the filter output tries to follow the predictions than the measurements. The corresponding sensor measurements,extended kalman filter output and actual position is shown in figure(3).

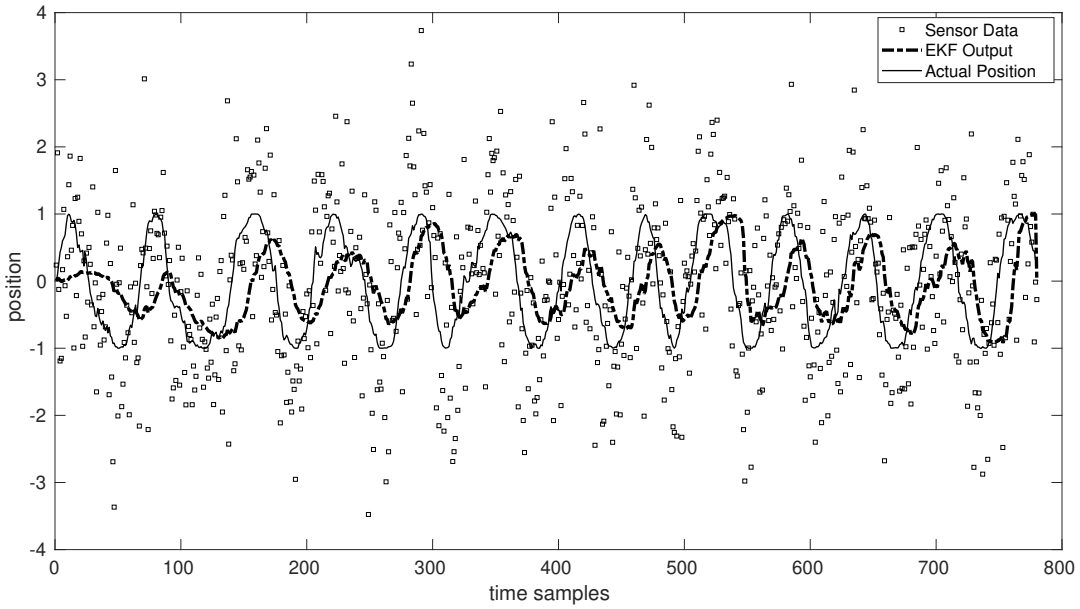

Figure 3: MN=0.5 & DN=0.001

4 Conclusion

The report describes in detail the analytical process involved in designing and implementing a extended Kalman Filter to track a nonlinear model. It was observed that the output of the extended 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 extended Kalman filter for sinusoidal model

```
% FILE NAME      : EKF.m
%
% DESCRIPTION    : Code to implement extended Kalman filter for
    sinusoidal Model.
%
% PLATFORM               : Matlab
%
% DATE                   NAME
% 22nd-Oct-2018      Shashi Shivaraju

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

%Read data from file
Data = dlmread("sin-data.txt");
ActualPos = Data(:,1); %Actual position of data for reference
SensorData = Data(:,2); %Sensor measurement data
n = length(SensorData); %Length of data

%Initialize
t = 1; %time
measure_noise = 0.5;%Measurement noise
dynamic_noise = 0.001;%Dynamic nosie
%Jacobians
Dfda = [0 0 0; 0 1 0; 0 0 0];
Dgdx = [0 0 1];
Dgdn = [1];

Q = [0 0 0; 0 dynamic_noise 0; 0 0 0]; %CoVariance of Dynamic
    noise
R = [measure_noise]; %CoVariance of Measurement noise
deltaTime = 0.001; %variation in time interval

X_t1_t1 = [deltaTime;0.01; SensorData(1,1)]; %State matrix
I = eye(3) ; %Identity matrix (3x3)
S_t1_t1 = I; %State Co-variance

Filter_Output = zeros(1, n); %EKF output
```

```
37
38 %loop through the data set
39 while(t <  length(Data))
40
41     %Predict the next state
42     X_t_t1 = [ X_t1_t1(1,1) + (deltaTime*(t-1) * X_t1_t1(2,1));
43                X_t1_t1(2,1);
44                sin(0.1*X_t1_t1(1,1))];
45
46     %Calculate jacobain for each loop
47     Dfx = [1 deltaTime*t 0; 0 1 0; 0.1*cos(0.1 * X_t_t1(1,1)) 0 0];
48
49     %Predicting next state co-variance
50     S_t_t1 = (Dfx * S_t1_t1 * Dfx') + (Dfda * Q * Dfda');
51
52     %Measurement data
53     Y_t = SensorData(t);
54
55     %Calculating kalman gain
56     K_t = S_t_t1 * Dgdx'/((Dgdx * S_t_t1 * Dgdx') + (Dgdn * R *
           Dgdn'));
57
58     %Update state
59     X_t_t = X_t_t1 + K_t * (Y_t - X_t_t1(3,1));
60
61     %Store filter output for plotting
62     Filter_Output(1,t) = X_t_t(3,1);
63
64     %Update state co-variance
65     S_t_t = (I - (K_t * Dgdx)) * S_t_t1;
66
67     %Incriment loop counter
68     t = t  + 1;
69
70     %Update prev variables
71     S_t1_t1 = S_t_t;
72     X_t1_t1 = X_t_t;
73
74 end
75
76 %plot the actual,measured and filtered data
77 x = 1:length(Data);
78 figure (1)
79 plot(x,SensorData, "ks",'MarkerSize',4);
80 hold on;
```

```
81 plot(x, Filter_Output,'k-.','LineWidth',2.5);
82 plot(x, ActualPos,'k','LineWidth',1);
83 hold off;
84 xlabel("time samples");
85 ylabel("position");
86 set(gca,"FontSize",14);
87 legend('Sensor Data', 'EKF Output', 'Actual Position');
```

References

1.Lecture notes of Dr.Adam Hoover

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

2.TeX Live - TeX Users Group

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