ECE 8540 Analysis of Tracking Systems - Lab 5 report

Mayuresh Bhosale

October 27, 2022

1 Introduction

This lab report explains how to use the extended Kalman filter on a set of measured data that is sinusoidal. Sinusoidal data is non-linear data that is difficult to estimate in real-time and needs a complex model. For non-linear data, the extended Kalman filter is used to get the estimates. Previously used techniques, such as normal Kalman filtering, can give good estimates for linear data but fails to provide good estimates for non-linear data. In the extended Kalman filtering, the non-linear equation is linearized by using Taylor series expansion. A Taylor series expansion is an equation in terms of derivatives that provides the best approximation of the non-linear equation. For this lab, the sinusoidal extended Kalman filter model is used for filtering the measured data. The measured data and ground truth data are provided to us.

2 Methodology

Similar to Kalman filtering, extended Kalman filtering is performed with a set of equations that continuously predict and update the estimates. Extended Kalman filtering contains slightly modified equations compared to the Kalman filter. The model information is represented in the observation matrix of non-linear equations in g and the state transition matrix of non-linear equations in f.Jacobian matrices related to these equations are used in extended Kalman filtering. Firstly, the next state and the state covariance are predicted. After the prediction, the sensor-recorded measurement is obtained, and Kalman gain is calculated. The Kalman gain consists of weights applied to the prediction and the measured value. Based on this Kalman gain, the state and state covariance are updated. The updated state provides the final output for that particular time, and this cycle is repeated over time. Equations 1 to 5 define the process of extended Kalman filtering.

1. predict next state

$$X_{t,t-1} = f(X_{t-1,t-1}, 0) \tag{1}$$

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

2. predict next state covariance

$$S_{t-1,t-1} \left(\frac{\partial f}{\partial x}\right)^T + \left(\frac{\partial f}{\partial a}\right) Q \left(\frac{\partial f}{\partial a}\right)^T \tag{2}$$

   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)

$$K_t = S_{t,t-1} \left(\frac{\partial g}{\partial x}\right)^T \left[\left(\frac{\partial g}{\partial x}\right) S_{t,t-1} \left(\frac{\partial g}{\partial x}\right)^T + \left(\frac{\partial g}{\partial n}\right) R \left(\frac{\partial g}{\partial n}\right)^T\right]^{-1} \tag{3}$$

   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

$$X_{t,t} = X_{t,t-1} + K_t[Y_t - g(\tilde{x}_t, 0)] \tag{4}$$

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

6. update state covariance

$$S_{t,t} = [I - K_t J_t^{g,x}] S_{t,t-1} \text{EKF}: \qquad S_{t,t} = \left[I - K_t \left(\frac{\partial g}{\partial x} \right) \right] S_{t,t-1} \tag{5}$$

7. loop (now t becomes $t+1$)

2.1 Extended Kalman filtering with Sinusoidal Model

In practice, values in the Jacobian matrices must be calculated at every iteration. For this lab, there were three state variables x_t, $\dot{x}_t$ and h_t. Where h_t represented the height of sinusoidal at time t.

We have three state variables:

$$X_t = \begin{bmatrix} x_t \\ \dot{x}_t \\ h_t \end{bmatrix} \tag{6}$$

For this model, x_t, $\dot{x}_t$ and h_t were set to zero at initial tome 0.

The state transition equations for this model are written as follows:

$$f(x_t, a_t) = \begin{bmatrix} x_{t+1} = x_t + \dot{x}_t T \\ \dot{x}_{t+1} = \dot{x}_t + a_t \\ h_{t+1} = \sin \frac{x_t}{10} \end{bmatrix} \tag{7}$$

These equations were used to predict future states. a_t is a random sample drawn from a Gaussian distribution $N(0, \sigma_a^2)$ representing uncertainty in the propagation of the sinusoid over time.

For observations, we will consider a sensor that detects the current height of the sinusoid d_t:

$$Y_t = \begin{bmatrix} d_t \end{bmatrix} \tag{8}$$

The observation equations for this model are:

$$g(x_t, n_t) = \begin{bmatrix} d_t = h_t + n_t \end{bmatrix} \tag{9}$$

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

We must calculate the four Jacobians to use this model in the EKF. The derivative of the state transition equations with respect to the state variables is:

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

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

$$\frac{\partial f}{\partial a} = \frac{\partial f}{\partial (0, a_t, 0)} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{11}$$

The values in the matrix $\frac{\partial f}{\partial x}$ are calculated every iteration. They are calculated using values from the current filtered estimate of the state variables.

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

$$\frac{\partial g}{\partial x} = \frac{\partial g}{\partial (x, \dot{x}, h)} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \tag{12}$$

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

$$\frac{\partial g}{\partial n} = \frac{\partial g}{\partial (n_t)} = \begin{bmatrix} 1 \end{bmatrix} \tag{13}$$

The covariance of the dynamic noises is:

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & \sigma_a^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{14}$$

The covariance of the measurement noises is

$$R = \begin{bmatrix} \sigma_n^2 \end{bmatrix} \tag{15}$$

The state covariance matrix was initialized as an identity matrix:

$$S_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16}$$

The final step was to check whether the matrix sizes were correct by fitting them into the extended Kalman filtering equations. Once the sizes were checked and corrected, the equations and respective Jacobian matrices were written in MATLAB software. R2021a version of MATLAB was used on a Windows 11 Operating System. Final estimated values of sinusoidal height h_t were obtained as output from equation 4. For the filtering, a time step of 1 second was considered ($T = 1$).

3 Results

Results for three different values of dynamic noise Q and measurement noise R, as written in Table 1, were plotted. Each plot depicts the measured data, the ground truth data, and the filtered estimates. In figures 1, 2, and 3, the x-axis represents the time ranging from 0 to 800 seconds, and the y-axis represents the height of the respective data.

Table 1: Represents value of two noises R (σ_n^2) and Q (σ_a^2) for three different iterations.

Trial No	R (σ_n^2)	Q (σ_a^2)
1	1.2	0.001
2	10^{-10}	10^{10}
3	1.2	100

Figure 1 shows the results for trial 1 with measurement noise R(σ_n^2) = 1.2 and dynamic noise Q(σ_a^2) = 0.001. As seen in the figure, the estimated values closely follow the ground truth values with less error. This suggests that dynamic and measurement noise values are nearly accurate for estimating given data. Figure 2 shows the results for trial 2 with very low measurement noise R(σ_n^2)=10^{-10} and very high dynamic noise Q(σ_a^2)=10^{10}. As seen in the figure, the estimated values do not follow the ground truth values and have large errors. The straight-line estimate output suggests that the measurement data does not correspond to the estimated value. Figure 3 shows the results for trial 3 with the measurement noise R(σ_n^2)=1.2 and dynamic noise Q(σ_a^2)=100. As seen in the figure, the estimated values do not follow the ground truth values and have large errors. The estimated output closely matches the measured data and deviates from the ground truth data. This suggests that the estimated output has large errors, and Q and R values should be balanced to get estimates close to the ground truth value, similar to the trial 1 results.

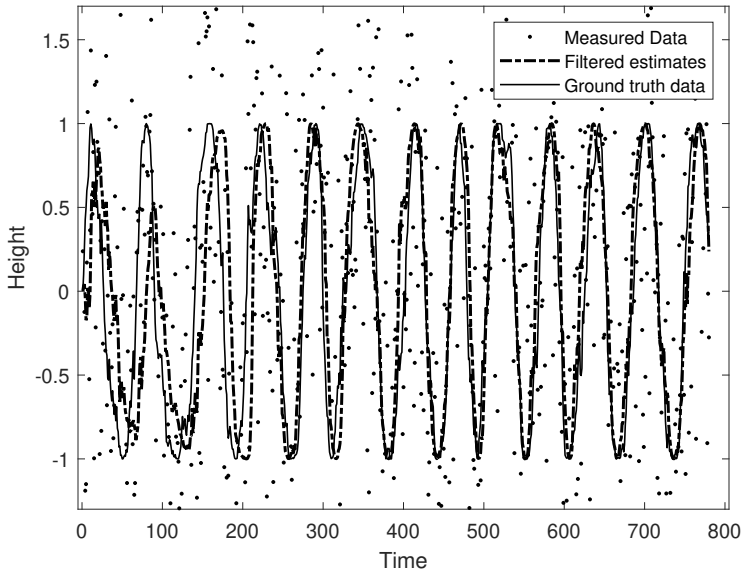

Figure 1: Plot of measured data, ground truth data, and filtered estimates for Trial 1 of extended Kalman filtering where $\mathrm{R}(\sigma_n^2) = 1.2$ and $\mathrm{Q}(\sigma_a^2) = 0.001$.

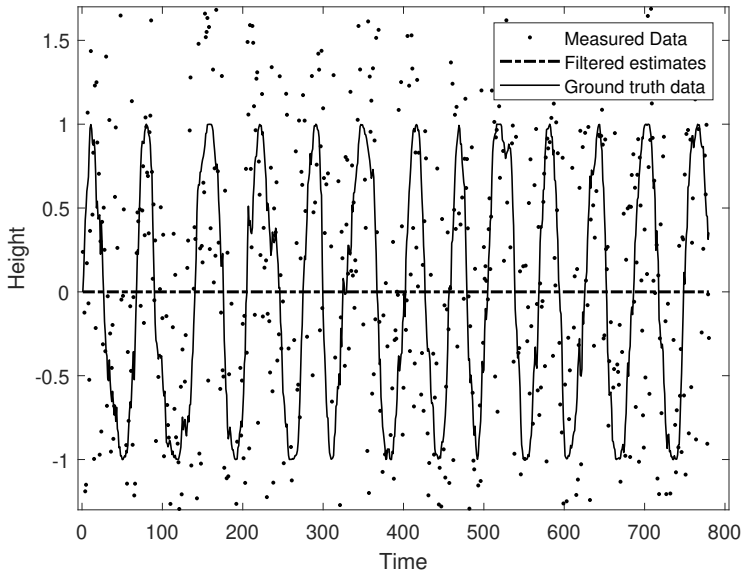

Figure 2: Plot of measured data, ground truth data, and filtered estimates for Trial 2 of extended Kalman filtering where $\mathrm{R}(\sigma_n^2)$=10^{-10} and $\mathrm{Q}(\sigma_a^2)$=10^{10}.

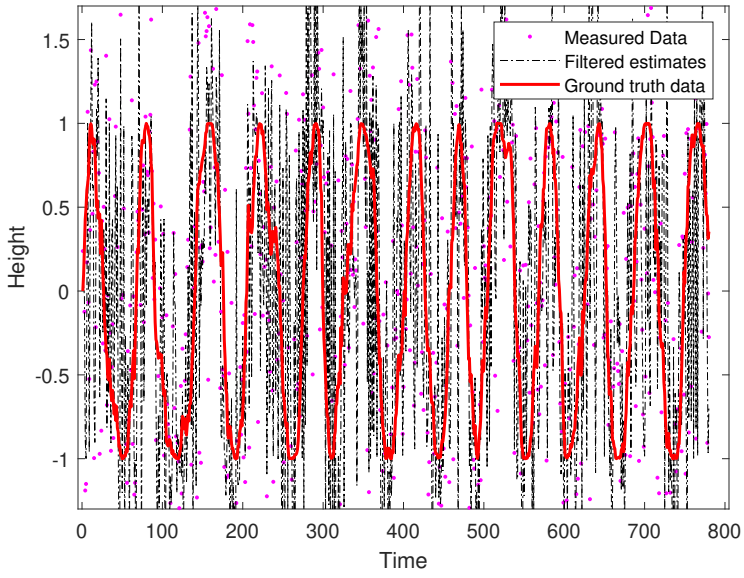

Figure 3: Plot of measured data, ground truth data, and filtered estimates for Trial 3 of extended Kalman filtering where R(σ_n^2)=1.2 and Q(σ_a^2)=100.

4 Conclusion

For non-linear data, the extended Kalman filter can generate accurate estimates with appropriate model equations and noise covariance values. Similar to the Kalman filtering, the measurement noise and dynamic noise values are to be weighed appropriately in order to generate accurate estimated output. For this lab, accurate estimated output was obtained with measurement noise R(σ_n^2) = 1.2 and dynamic noise Q(σ_a^2) = 0.001. Computation and programming software such as MATLAB used in this lab allows us to calculate complex model equations and matrices easily and within seconds.

5 Appendix

The code for this lab can be found below:

```
% This Program is written by Mayuresh Bhosale that depicts Extended Kalman
% Filter applied on a sinusoidal data

clc
clear all
close all
%1-D Data & Plot
Adata = importdata('sin-data.txt');
xaxis = 1:height(Adata);



% State variable matrix
X_t = [0; 0; 0];
% State transition equations/matrix
S_t = [1 0 0; 0 1 0; 0 0 1];
%Measurement variable
Y_t = Adata(1,2);

```

```
%Jacobian matrices
% derrivative of State transition equations wrt State variables
df_dx = [1 1 0;
         0 1 0;
         0.1*(cos(0.1*X_t(1))) 0 0];
% derrivative of State transition equations wrt Dyn noise
df_da = [0 0 0;
         0 1 0;
         0 0 0];
% derrivative of observation equations wrt State variables
dg_dx = [0 0 1];
%derrivative of observation equations wrt measurement noise
dg_dn = [1];

%Dynamic noise covariance
Q = [0 0 0;
     0 100 0;
     0 0 0];
%Measurement Noise Covariance
R = [1.2];
plotx = [];
Error = [];
for i = 1:length(Adata)
    % Predict next state
    X_t_t1 = [X_t(1)+X_t(2); X_t(2); sin(0.1*(X_t(1)))];
    %Predict next state covariance
    df_dx(3,1) = 0.1*cos(0.1*(X_t(1)));
    S_t_t1 = df_dx*S_t*(df_dx') + df_da*Q*(df_da');
    %Obtain measurement
    Y_t = [Adata(i,2)];
    %Caculate Kalman Gain
    K = S_t_t1*(dg_dx')*((dg_dx*S_t_t1*(dg_dx'))+(dg_dn*R*(dg_dn')))^(-1);
    %Update State
    X_t = X_t_t1 + K*(Y_t - (sin(0.1*(X_t(1)))));
    %Update state covariance
    S_t = (eye(3) - (K*dg_dx))*S_t_t1;
    plotx(i) = X_t(3);
    Error = sum(abs(X_t(3)-Adata(i,1)));
end

% t1 = tiledlayout(1,1);
% nexttile
% % plot(xaxis, Adata(:,2));
% % hold on
% plot(xaxis, Adata(:,1),"LineWidth",1.5,"Color",'k',"LineStyle","-");
% hold on
% plot(xaxis, Adata(:,2),"LineWidth",0.5,"Color",'k',"LineStyle","--");
% hold off
% ylabel('Height')
% xlabel('Time')
% legend('Ground truth data','Measured data')
% exportgraphics(t1,'Raw_data.eps')

% t2 = tiledlayout(1,1);
```

```
% nexttile
%% plot(xaxis, Adata(:,2));
%% hold on
% plot(xaxis, Adata(:,2),'k.');
% hold on
% plot(xaxis,plotx,"LineWidth",1.5,"Color",'k',"LineStyle","-.");
% plot(xaxis, Adata(:,1),"LineWidth",0.8,"Color",'k',"LineStyle","-");
% ylabel('Height')
% xlabel('Time')
% legend('Measured Data','Filtered estimates','Ground truth data')
% ylim([-1.3 1.7]);
% xlim([-5 805]);
% exportgraphics(t2,'Filter1.eps')


t2 = tiledlayout(1,1);
nexttile
% plot(xaxis, Adata(:,2));
% hold on
plot(xaxis, Adata(:,2),'m.');
hold on
plot(xaxis,plotx,"LineWidth",0.5,"Color",'k',"LineStyle","-.");
plot(xaxis, Adata(:,1),"LineWidth",1.5,"Color",'r',"LineStyle","-");
ylabel('Height')
xlabel('Time')
legend('Measured Data','Filtered estimates','Ground truth data')
ylim([-1.3 1.7]);
xlim([-5 805]);
exportgraphics(t2,'Filter3.eps')
```