Extended Kalman Filter

Lab-5

Submitted by:

Priyanshu Rawat

---

1 Introduction

The Extended Kalman Filter (EKF) is an extension of the normal Kalman Filter, that works on the similar principle to estimate states using the predict and update cycle. A problem in KF implementation is represented in the form of the observation matrix M and the state transition matrix Θ. However, in the EKF, these matrices are adjusted to use the non-linear equations in the respective g and f matrices, and the related Jacobians.
The following lists the equations for the implementation of EKF:

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

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

   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

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

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

1.1 Problem Statement

To design a sinusoidal EKF model for the given measurements. The filter should operate on the measurements only. The results from the filter should be compared with the true state using three different ratios of dynamic noise to measurement noise. Discuss the differences between the outputs.

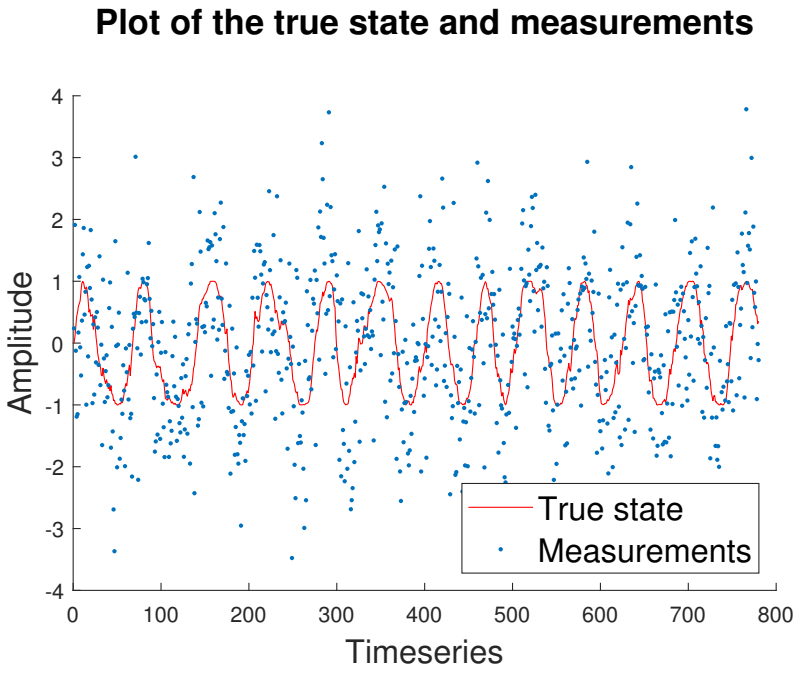

Figure 1:
Plot showing the given measurements and true state

2 Results and discussion

2.1 Case I: $R = \begin{bmatrix} 0.001 \end{bmatrix}$, $Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

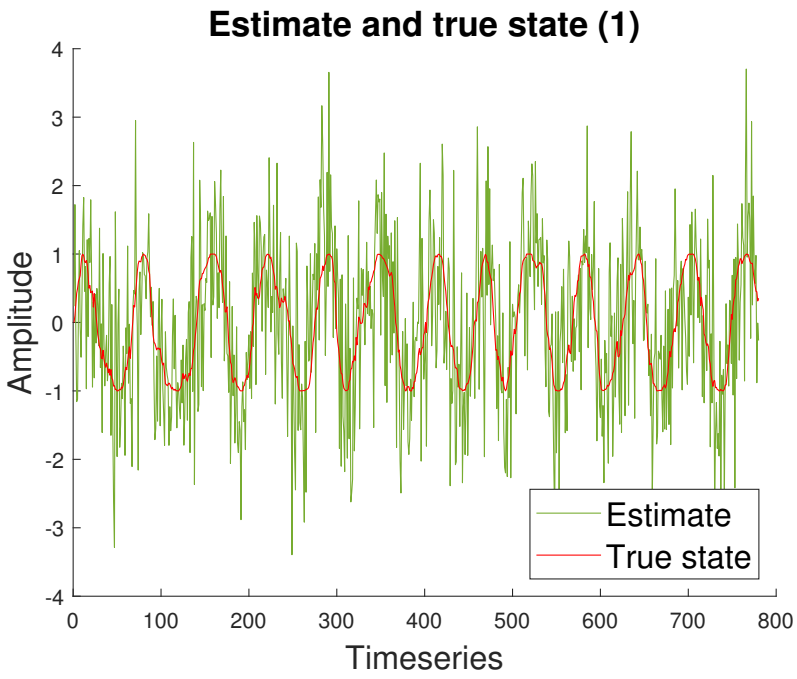

Figure 2:
EKF estimate for case I

2.2 Case II: $R = [1]$, $Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

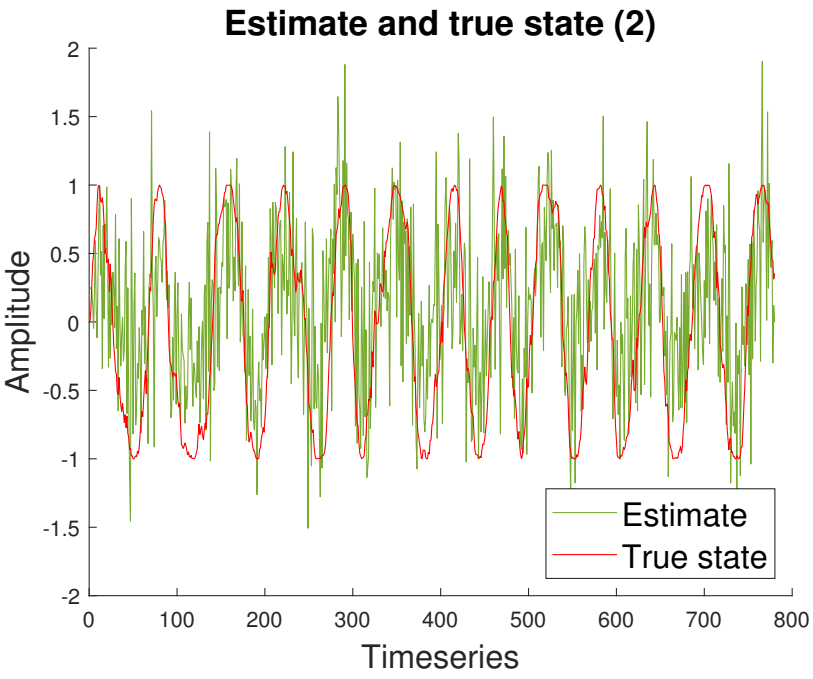

Figure 3:
EKF estimate for case II

2.3 Case III: $R = [1000]$, $Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$

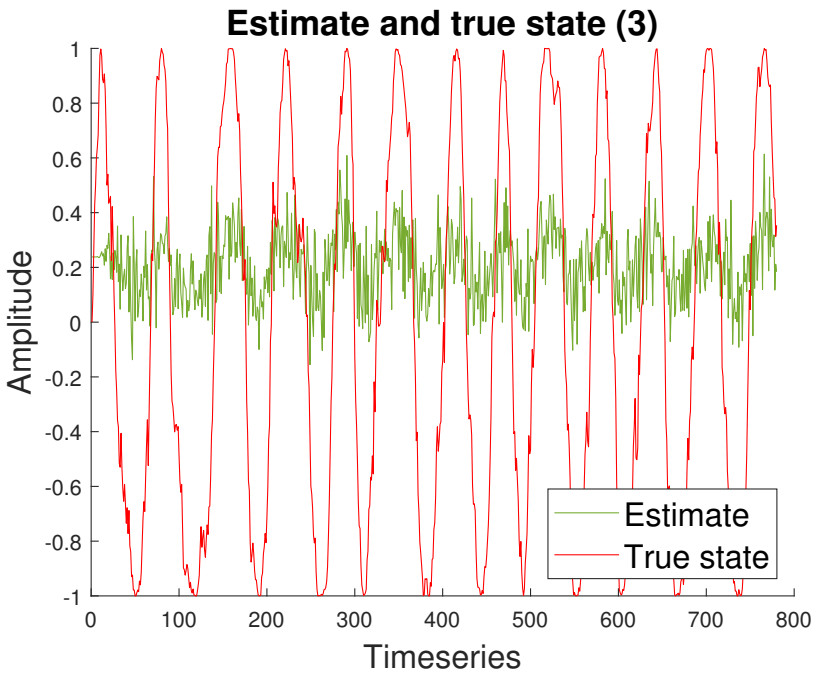

Figure 4:
EKF estimate for case III

In the all the three cases above the dynamic noise co-variance matrix Q was fixed to -

$$Q = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

While the measurement noise co-variance value R was adjusted from *0.001*, to *1*, to *1000*. Figure 2 shows the estimates obtained from the Extended Kalman filter when R was *0.001*. Whereas, figure 3 and 4, show the estimates obtained from the Extended Kalman filter when R was *1* and *1000* respectively.
From the figures 2, 3, and 4; it can be seen that as the value of R increases suggesting more noise in measured values, the estimated values from EKF lie closer to the predictions than the measurements. On the other hand, when the

value of R was less, say, 0.001, as seen in figure 2; the estimated values lie closer to the measurements than predictions. This is because a higher value of R suggests more uncertainty in the measurements, which makes the EKF estimates rely more upon the predictions rather than the measured values. A much more accurate sinusoidal estimate can be observed in the figure 2, when the value of R is 1.

3 Conclusion

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

4 Appendix (MATLAB script)

```
clc;
clear;

% Loading the data in matlab
data = load("sin-data.txt");
true_state = data(:,1);
measurements = data(:,2);

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

% Plotting the raw data
figure
hold on
plot(timeseries, true_state,'Color',[1 0 0])
scatter(timeseries, measurements,'.','MarkerEdgeColor',[0 0.4470 0.7410])
legend('True state', 'Measurements')
hold off
title('Plot of the true state and measurements')
xlabel('Timeseries')
ylabel('Amplitude')

syms T x xdot h d h n a

% State space equations
f = [x+xdot*T;
     xdot+a;
     sin(x/10)];

% State variables
X = [x;
     xdot;
     h];

% Observation
Y = [d];

% State observation equations
g = [h + n];

% Jacobians
Fx = jacobian(f,X);
Gx = jacobian(g,X);
Fa = [0 0 0
```

```
      0 1 0
      0 0 0];
Gn = jacobian(g,n);

% Dynamic uncertainity matrix
Q = [0 0 0
     0 5 0
     0 0 0];

% Measurement noise covariance
R = [1];

% Initial variance
S = [1 0 0
     0 1 0.001
     0 0.001 1];

% Number of state variables and randoms
n_states = 3;
a = 1;
n = 0.01;
T = 1;

% Measurements
Y =  measurements;

% Inital states
h = measurements(1,1);
x = asin(h)*10;
xdot=x/T;
fx = double(subs(f));
Fa = double(subs(Fa));
Gn = double(subs(Gn));

% Estimates matrix
X_est(:,1) = double(subs(X));

%% EKF loop
for i = 2:numel(measurements)
    %% State prediction
    X_pred = double(fx);

    %% State values
    x = double(X_pred(1));
    xdot = double(X_pred(2));
    h = double(X_pred(3));
```

```
    %% Updating jacobians
    Fx = double(subs(Fx));
    Gx = double(subs(Gx));

    %% State covariance prediction
    S_pred = Fx*S*Fx' + Fa*Q*Fa';

    %% Kalman gain
    K = (S_pred*Gx')/(Gx*S_pred*Gx' + Gn*R*Gn');

    %% Correction
    X_pred = X_pred+K*(Y(i)-h);
    S = (eye(n_states)-K*Gx)*S_pred;

    %% Storing estimation
    X_est(:,i) = X_pred;
end

%% Plot of estimate and true state
figure
hold on
plot(X_est(3,:),Color=[0.4660 0.6740 0.1880])
plot(true_state,Color= [1 0 0])
legend('Estimate', 'True state')
hold off
title('Plot of the estimate and true state')
xlabel('Timeseries')
ylabel('Amplitude')
```