Comparison of Kalman and Extended Kalman Filter for State Estimation of Inverted Pendulum

Siddharth Rajagopal Sivakumar Balasubramanian Uday Mathur
Department of Mechanical Engineering
University of Washington,
Seattle, USA

I. Introduction

An inverted pendulum is a self-regulated system with a to-and-fro motion that resembles the balancing of an umbrella or a stick to maintain it upright. This balancing problem is of interest because the system is unstable without control and the pendulum would simply fall over if the cart isn't moved to maintain the upright position. Additionally, the dynamics of the system are nonlinear.

The objective of this paper is to estimate the states of the system to optimize the compensation that is made by the cart to keep the pendulum upright. Randomly generated noise profiles are added to the true measurement data generated using the dynamics of the system. This data is later used for estimation of states of the system using Kalman Filter and Extended Kalman Filter. These results of these estimations are analyzed and compared.

II. Model

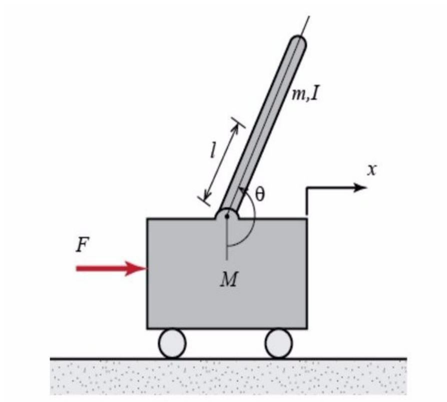


Figure 1 : An inverted pendulum in 2D

According to Jie & Sijing [1], the following equations give the dynamics of the system:

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos\theta - ml\dot{\theta}^2 sin\theta = F$$

$$(I+ml^2)\ddot{\theta} + mglsin\theta = -ml\ddot{x}cos\theta$$

where,

M:	mass of the cart
m:	mass of the pendulum
b :	coefficient of friction for cart
l :	length to pendulum center of mass
I :	mass moment of inertia of the pendulum
F:	force applied to the cart
x :	cart position coordinate
θ:	pendulum angle from vertical (down)

This Continuous nonlinear system is linearized by equating the ODE $\dot{X}$=$f(x_{eq}, u_{eq})$=0 around equilibrium point θ = π. The continuous linear time invariant system is of the form as shown.

$$\dot{X} = AX + Bu$$
$$Y = CX + Du$$

Where the matrices A and B are obtained by taking the Jacobian of $f(x,u)$ with respect to x and u respectively and the matrices C and D are obtained by taking the Jacobian of $h(x,u)$ with respect to x and u respectively around the equilibrium points. The state space system are as shown which also include the identity disturbance observer as assumed by Shimada & Yongyai [2] :

$$X = \begin{bmatrix} x \\ v \\ \theta \\ \omega \end{bmatrix}$$

$v = \dot{x}$: velocity of cart
$\omega = \dot{\theta}$: The angular velocity of pendulum

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & -\frac{(I+ml^2)b}{p} & \frac{(m^2gl^2)}{p} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -\frac{mlb}{p} & \frac{mgl(M+m)}{p} & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ \frac{(I+ml^2)}{p} \\ 0 \\ \frac{ml}{p} \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$D = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

This state space model is derived by linearizing the model around equilibrium point $\theta = \pi$. The A matrix is derived by linearizing the equations of motion around equilibrium point

A. *States*

The X matrix represents the states of the system.

B. *Inputs*

The inputs to the system is a scalar u, which controls the velocity of cart and angular velocity of the pendulum.

C. *Outputs*

The outputs from the system are the position of cart x and angular position of the pendulum θ. The output function is of the form $Y=h(x,u)$.

The system controllability and observability matrices were ensured to be full rank. The following is the observability matrix:

$$\mathcal{O} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.6667 & 0.3333 & 0 \\ 0 & -0.333 & 0.6667 & 0 \\ 0 & 0.444 & -0.222 & 0.333 \\ 0 & 0.2222 & -0.1111 & 0.6667 \end{bmatrix}$$

III. Data

The data required for estimating the states and outputs are simulated in Matlab. Two types of noise are added to the output. One is the Gaussian white noise and the other is a Poisson noise. We also add a Gaussian disturbance $w \sim N(0;Q)$ to the system dynamics, where Q is the covariance of the disturbance. The state space equations for the noisy system are as follows.

$$\dot{X} = AX + Bu + Gw$$
$$Y = CX + Du + Hv$$

where w is the disturbance in the dynamics and v is the noise in the observation. G is the jacobian of the system dynamics with respect to the disturbance w and H is the jacobian of the observation with respect to the noise v. The initial states of the system are [1, 0, 0, 0] and a small constant input of 0.1 is given to the system.

A. *Gaussian White Noise*

We first add Gaussian white noise $v \sim N(0;R)$ to the observation. R is the noise covariance matrix and it is chosen as follows for our system.

$$R = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Figure 2 (a - d) show the simulated observations and states with and without noise and disturbances. We see that the true position and angular position oscillate about zero which is their equilibrium point.

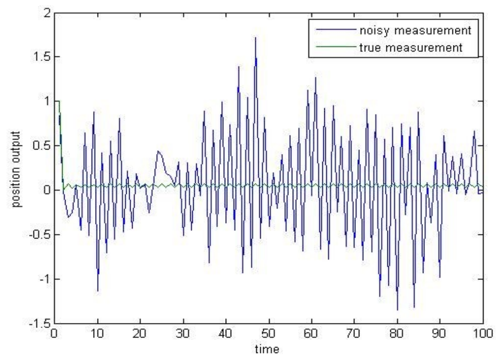


Figure 2a: True and noisy position output measurement

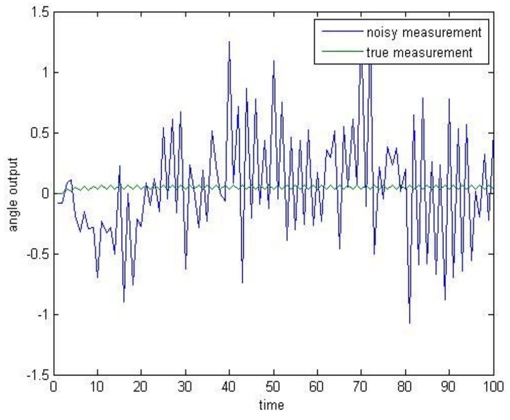


Figure 2b: True and noisy angle output measurement

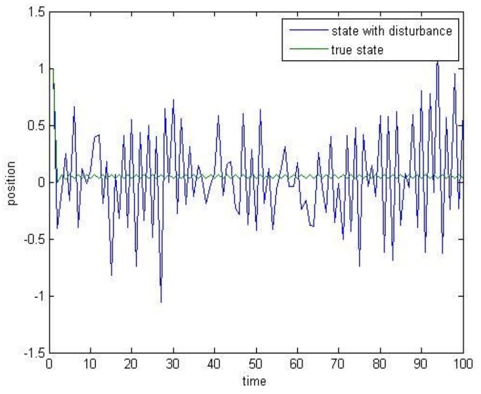


Figure 2c: State (position) with and without disturbances

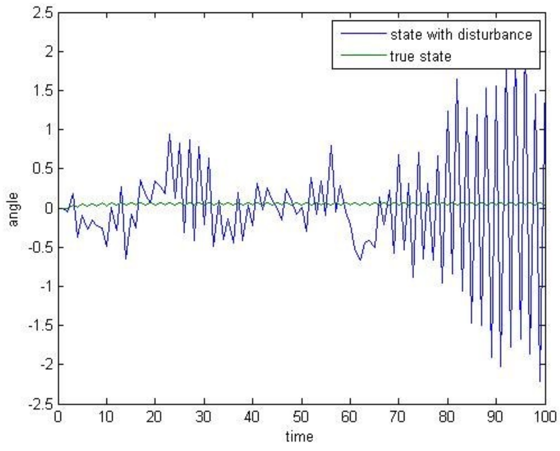


Figure 2d: State (angle) with and without disturbances

B. *Poisson Noise*

In this case we add a Poisson noise with λ value of 0.1 to the output measurements. The plots for the states are the same but the plots for the observations would have the Poisson noise instead of the Gaussian white noise. The plots for the two observations are as shown in figure 3a and 3b.

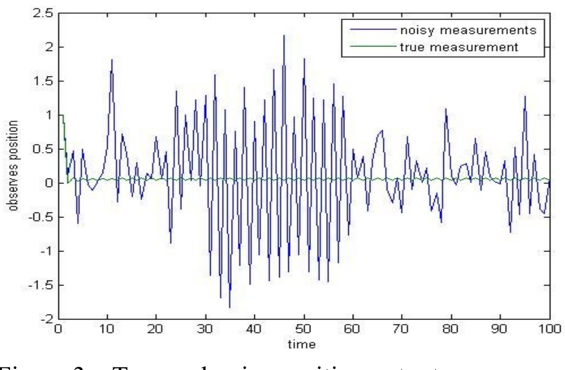


Figure 3a: True and noisy position output measurement

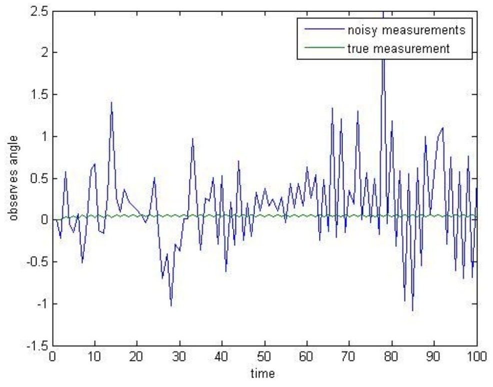


Figure 3b: True and noisy angle output measurement

C. *Non Linear Dynamics*

In actuality, we know that the inverted pendulum is a highly non-linear system and thus we also generate the plots for the nonlinear dynamics of the system. The Nonlinear state equation representing the system is given by:

$$X =$$

$$\begin{bmatrix} v \\ 4\,(0.5(u + w^2\, sin\theta) - (0.5x - 0.25cos\theta sin\theta))/(4 - cos^2\theta) \\ w \\ 4\,(-0.5(sin\theta) - 0.25u(cos\theta) - 0.25w^2 cos\theta sin\theta + 0.25xcos\theta)/(4 - cos\theta sin\theta) \end{bmatrix}$$

We keep the input to be zero and rest of the parameters the same and simulate the observed position and angle measurements with and without noise. Figure 4 (a – b) show the noisy and actual observed values for Gaussian noise and Figure 4 (c – d) show the noisy and actual observed values for Poisson noise.

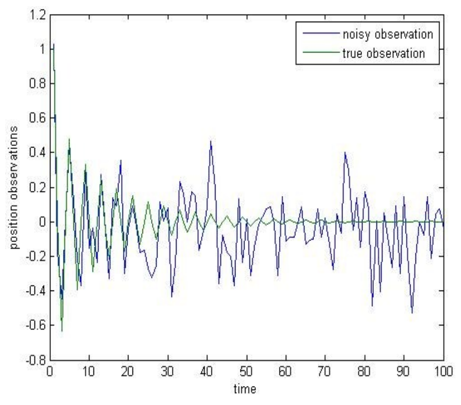


Figure 4a: True and noisy (Gaussian) position output measurement for nonlinear dynamics

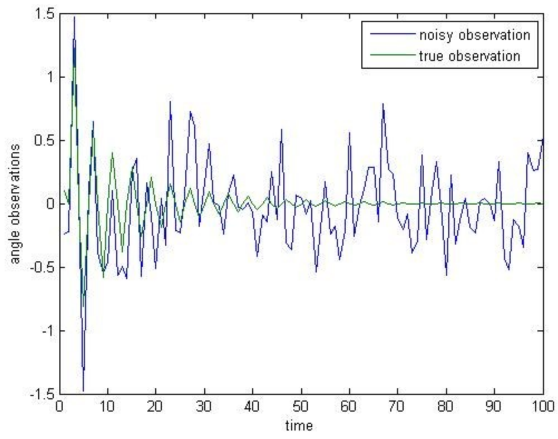


Figure 4b: True and noisy (Gaussian) Angle output measurement for nonlinear dynamics

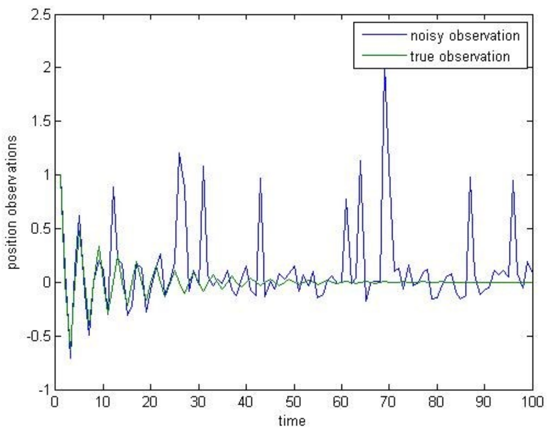


Figure 4c: True and noisy (Poisson) Position output measurement for nonlinear dynamics

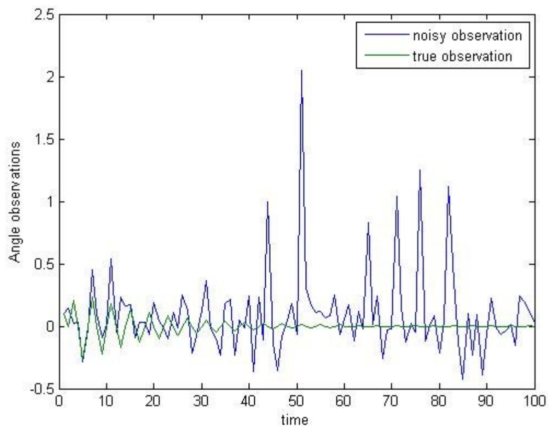


Figure 4d: True and noisy (Poisson) Angle output measurement for nonlinear dynamics

We see that the true values don't oscillate uniformly as seen in the linear case. They have high oscillations initially but it dies down due to the nonlinearity in the system. We will be using the non-linear case for comparing the performance of the Kalman and Extended Kalman filters.

IV. Methods

A. *Kalman Filter*

According to Awasthi & Raj[3], the Steady State Kalman filter minimizes the variance of the estimation error. Initially the process state is estimated at some time and then obtains feedback in the form of (noisy) measurements. The equations for the Kalman filter fall into two groups:

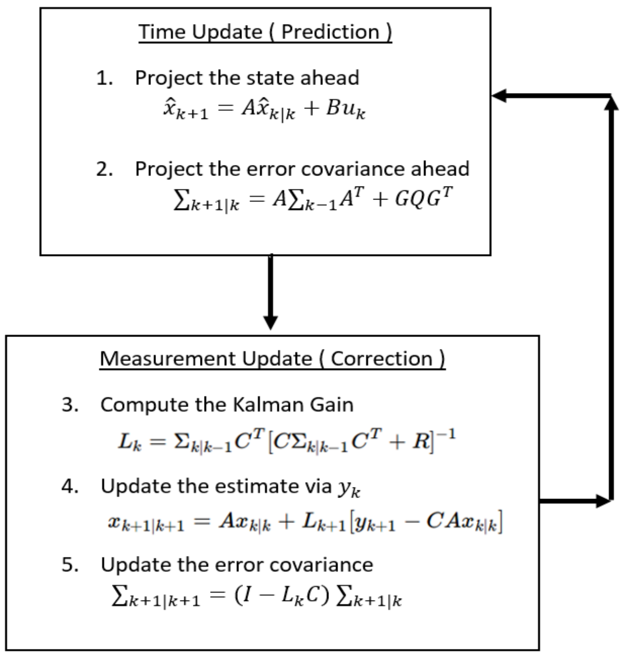
Time update (predictor) equations: which are responsible for projecting forward (in time) the current state and error covariance estimates to obtain the a priori estimates for the next time step.

Measurement update (corrector) equations: which are responsible for the feedback by incorporating a new measurement into the a priori estimate to obtain an improved a posteriori estimate.

In the case of a steady state Kalman filter, the forms of the equations are the same as the linearized equations shown in the above section 2. We consider a time invariant system for prediction. The Kalman gain matrix L, is designed to minimize the estimation error J given by

$$J = y - \hat{y}$$

where, y is the true measurement and $\hat{y}$ is the estimated measurement. The states that are estimated using the time invariant dynamics of the system and are update algorithm is implemented using the following equations:


Time Update (Prediction)

1. Project the state ahead
   $\hat{x}_{k+1} = A\hat{x}_{k|k} + Bu_k$
2. Project the error covariance ahead
   $\sum_{k+1|k} = A\sum_{k-1}A^T + GQG^T$

Measurement Update (Correction)

3. Compute the Kalman Gain
   $L_k = \Sigma_{k|k-1}C^T[C\Sigma_{k|k-1}C^T + R]^{-1}$
4. Update the estimate via y_k
   $x_{k+1|k+1} = Ax_{k|k} + L_{k+1}[y_{k+1} - CAx_{k|k}]$
5. Update the error covariance
   $\sum_{k+1|k+1} = (I - L_kC)\sum_{k+1|k}$

Figure 5: Kalman Filter values for position observation

where,

$\sum_{k+1|k}$: predicted covariance at step k+1, given k data
$\hat{x}_{k+1|k}$: predicted state of at step k+1, given k data
$\sum_{k|k}$: updated covariance at step k, given k data
$\hat{x}_{k|k}$: updated state of at step k, given k data
L_k : Kalman gain at step k

B. *Extended Kalman Filter*

The EKF is a heuristic for the non-linear filtering problem. The dynamics and output functions are linearized at the current time step by finding the Jacobian of the system.

$$A_t = \frac{\partial f_t}{\partial x}(\hat{x}_{t|t})$$

$$C_t = \frac{\partial h_t}{\partial x}(\hat{x}_{t|t-1})$$

$$G_t = g_t(\hat{x}_{t|t})$$

Then an approximate of the conditional expectation and covariance is propagated though each time step t. Kumar & Varaiya[4] corroborate the steps of Kalman Filter as follows:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + L_t(y_t - h_t(\hat{x}_{t|t-1}))$$

$$\hat{x}_{t+1|t} = f_t(\hat{x}_{t|t})$$

$$L_t = \Sigma_{t|t-1} C_t^T [C_t \Sigma_{t|t-1} C_t^T + R_t]^{-1}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{t|t-1} C^T [C_t \Sigma_{t|t-1} C_t^T + R_t]^{-1} C_t \Sigma_{t|t-1}$$

$$\Sigma_{t+1|t} = A_t \Sigma_{t|t} A_t^T + G_t Q_t G_t^T$$

The time update equations project the state and covariance estimates from the previous time step k – 1 to the current time step k. The measurement update equations correct the state and covariance estimates with the measurement Y_t.

V. Results and Discussions

The results obtained for the two noises using the Kalman and Extended Kalman filters are shown. All constants in the dynamics are taken to be unity for simplicity.

A. *Gaussian Noise*

The plots in Figure 5a and 5b show the Kalman and Extended Kalman Filter values for the position observation respectively. The plots in Figure 5c and 5d show the same for the angle observations.

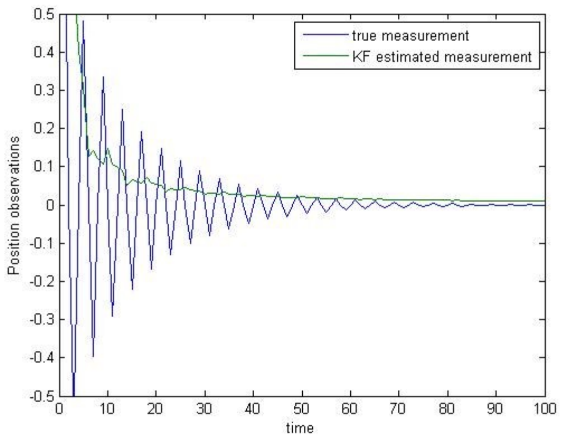


Figure 6a: Kalman Filter values for position observation

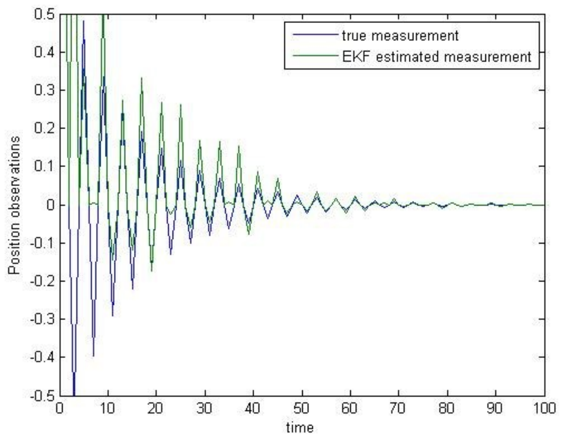


Figure 6b: Extended Kalman Filter values for position observation

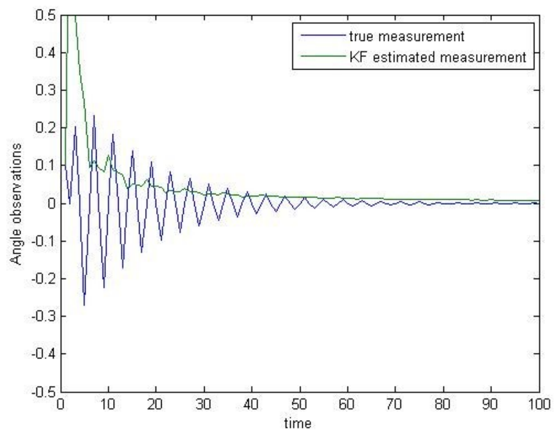


Figure 6c: Kalman Filter values for angle observation

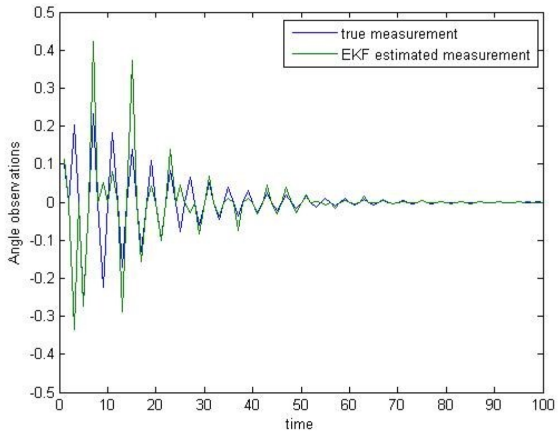


Figure 6d: Extended Kalman Filter values for angle observation

We see that the extended Kalman filter does a better job in converging to the true values as it does a first order linear approximation of the non-linear dynamics at each time step. This makes the mean and covariance converge much faster than the Kalman case which uses the linear model for calculating the mean and covariance.

B. *Poisson Noise*

Now we try to implement the filters for the nonlinear dynamics with Poisson noise. Figure 6a and 6b show the Kalman and Extended Kalman filter values for the position observation respectively. The plots in Figure 6c and 6d show the same for the angle observations.

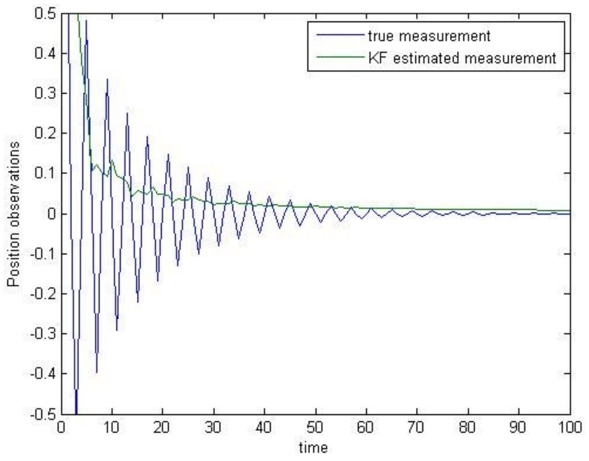


Figure 7a: Kalman Filter values for position observation

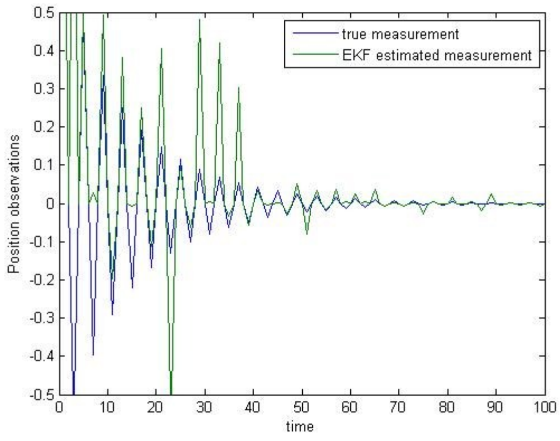


Figure 7b: Extended Kalman Filter values for position observation

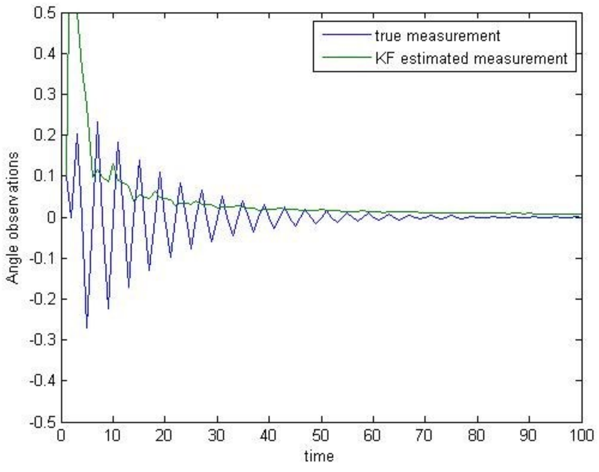


Figure 7c: Kalman Filter values for angle observation

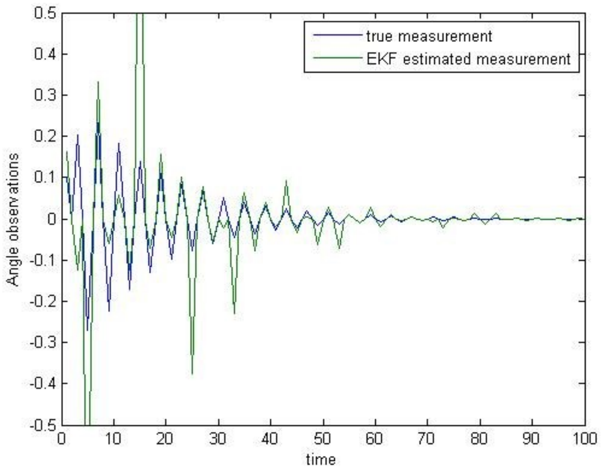


Figure 7d: Extended Kalman Filter values for angle observation

We see that the Kalman filter doesn't change much for the Poisson noise compared to the Gaussian noise case. The Extended Kalman filter takes more time to converge to the true value compared to the Gaussian noise case. In few cases it also oscillates again after converging. We can better the performance by tuning the Noise covariance matrix. But the EKF does not perform well consistently as it did for the white Gaussian noise case.

When we use other initial conditions we can find that even the EKF may be insufficient for getting estimates close to the true values and thus we can go for the Unscented Kalman filter or the particle filter. They are used for cases where there is high amount of non-linearity in the system dynamics. We should also note that EKF is not an optimal filter as compared to the Kalman filter.

VI. References

[1] Z. Jie and R. Sijing, "Sliding mode control of inverted pendulum based on state observer," in *2016 Sixth International Conference on Information Science and Technology (ICIST)*, 2016, pp. 322–326.

[2] A. Shimada and C. Yongyai, "Motion control of inverted pendulum robots using a kalman filter based disturbance observer," *SICE J. Control Meas. Syst. Integr.*, vol. 2, no. 1, pp. 50–55, 2009.

[3] V. Awasthi and K. Raj, "A Comparison of Kalman Filter and Extended Kalman Filter in State Estimation."

[4] P. R. Kumar and P. Varaiya, *Stochastic Systems : Estimation, Identification, and Adaptive Control.* .

VII. Appendix

MATLAB Code:

```
clc, clear all, close all;
%constants initialised to one.
M = 1;
m = 1;
b = 1;
I = 1;
g = 1;
l = 1;

p = I*(M+m)+M*m*l^2; %denominator for the
A and B matrices

A = [0      1              0           0;
     0 -(I+m*l^2)*b/p  (m^2*g*l^2)/p   0;
     0      0              0           1;
     0 -(m*l*b)/p       m*g*l*(M+m)/p
0];
B = [     0;
     (I+m*l^2)/p;
          0;
        m*l/p];
C = [1 0 0 0;
     0 0 1 0];
D = [0;
     0];

 dt=0.1;
 tfin = 5;
 tspan = 0:dt:tfin;

 sysc = ss(A,B,C,D);
 sysd = c2d(sysc,dt);
 %discretisation
 Ad = sysd.A;
Bd = sysd.B;
Cd = sysd.C;
Dd = sysd.D;

%Simulation
mu = [1 0 0.1 0]';
sigma = eye(4);

u =0;
w1 = normrnd(0,0.1,100,1);
w3 = normrnd(0,0.2,100,1);
w2 = normrnd(0,0.1,100,1);
w4 = normrnd(0,0.3,100,1);
%disturbance
wn= [w1,w2,w3,w4];
%disturbance covariance matrix
Q = [0.1 0 0 0;
    0 0.2 0 0;
    0 0 0.1 0;
    0 0 0 0.3;];

%gaussian noise
 %v1 = normrnd(0,.2,100,1);
 %v3 = normrnd(0,.2,100,1);
%poisson noise
 v1=poissrnd(0.1,100,1);
 v3=poissrnd(0.1,100,1);
vn= [v1,v3];
% R = [.2 0;
%      0 .2];
 R=cov(vn);
G=eye(4);

H = [1 0;0 1];
%linear simulations
% x(:,1) = mu;
% for i=1 :100
%      x(:,i+1) = Ad*(x(:,i))+Bd*u +
(G*w(i,:)');
%      y(:,i) = Cd*x(:,i) + H*v(i,:)';
% end
% xreal(:,1) = mu;
% for i=1 :100
%     xreal(:,i+1) =
Ad*(xreal(:,i))+Bd*u;
%     yreal ?,i) = Cd*(xreal(:,i));
% end

%non-linear simulations
xreal(:,1) = mu;
for i=1 :100
     t= xreal(3,i);
  x= xreal(1,i);
 v= xreal(2,i);
  w= xreal(4,i);
    xreal(:,i+1)=[v;
(4*(0+w^2*sin(t)*0.5 - 0.5*x +
cos(t)*sin(t)*0.25))/(4-cos(t)^2);
w;
(4*((-0.5*sin(t)) - 0 -
w^2*cos(t)*sin(t)*0.25 +
x*cos(t)*0.25))/(4-cos(t)*sin(t))];
    yreal(:,i) = C*(xreal(:,i));
end
xn(:,1) = mu;

%noisy observation simulation
for i=1 :100
     xn(:,i+1) = xreal(:,i+1)+wn(i,:)';
     y(:,i) = C*xn(:,i) + H*vn(i,:)';
end
%  xreal(:,1) = mu;
% for i=1 :100
%     xreal(:,i+1) = A*(xreal(:,i))+B*u;
%     yreal ?,i) = C*(xreal(:,i));
% end

%
plot(1:100,y(2,:),1:100,yreal(2,:)),legen
d('noisy observation','true
observation'),xlabel('time'),ylabel('Angl
e observations')
%plot(1:100,y(1,:),1:100,yreal(1,:))
```

```
% %% kalman filter

%Q = [1 0; 0 2];  % process noise w
G = eye(4); % is the matrix associated
with process noise w_k (like A)
%R = 0.1; % measurement noise v
K = [];


xest_b(:,1) =mu;
xest_c(:,1)= mu;

Pest_b(:,:,1) = sigma;
Pest_c(:,:,1) = sigma;

for n=1:100

xest_b(:,n+1) = A*xest_c(:,n)+B*u;
%estimate with data before current
Pest_b(:,:,n+1) = A*Pest_c(:,:,n)*A' + Q;
% estimate with data before current

K(:,:,n) =
Pest_c(:,:,n)*(C'*inv(C*Pest_c(:,:,n)*C'
+ R));

xest_c(:,n+1) =
xest_c(:,n)+(K(:,:,n)*(yreal(n)-
C*xest_c(:,n)));
Pest_c(:,:,n+1) = (eye(4)-
K(:,:,n)*C)*Pest_c(:,:,n);
yest(:,n) = C*xest_c(:,n);

end

 plot(1:100,yreal(1,:),1:100,yest(1,:))
   legend('true measurement','KF
estimated measurement'), axis([0 100 -0.5
0.5]),xlabel('time'),ylabel('Position
observations')
```

Extended Kalman Filter :

```
clc, clear all, close all;
%constants initialised to one.
M = 1;
m = 1;
b = 1;
I = 1;
g = 9.8;
l = 1;

p = I*(M+m)+M*m*l^2; %denominator for the
A and B matrices

A = [0      1              0           0;
     0 -(I+m*l^2)*b/p  (m^2*g*l^2)/p   0;
     0      0              0           1;
     0 -(m*l*b)/p       m*g*l*(M+m)/p
0];
B = [     0;
     (I+m*l^2)/p;
          0;
        m*l/p];
C = [1 0 0 0;
     0 0 1 0];
D = [0;
     0];

 dt=0.1;
 tfin = 5;
 tspan = 0:dt:tfin;

 sysc = ss(A,B,C,D);
 sysd = c2d(sysc,dt);

 Ad = sysd.A;
Bd = sysd.B;
Cd = sysd.C;
Dd = sysd.D;

%Simulation
mu = [1 0 0.1 0]';
sigma = eye(4);

u =0;
w1 = normrnd(0,0.1,100,1);
w3 = normrnd(0,0.2,100,1);
w2 = normrnd(0,0.1,100,1);
w4 = normrnd(0,0.3,100,1);
%disturbances
wn= [w1,w2,w3,w4];
%disturbance covariance matrix
Q = [0.1 0 0 0;
    0 0.2 0 0;
    0 0 0.1 0;
    0 0 0 0.3];

%gaussian noise
 v1 = normrnd(0,.2,100,1);
 v3 = normrnd(0,.2,100,1);
 %poisson noise
v1=poissrnd(0.1,100,1);
 v3=poissrnd(0.1,100,1);
vn= [v1,v3]; % as sensor readings are
taken ony from two output states
  R = [0.1 0;
      0 0.1];
%  R=cov(vn);
G=eye(4);

H = [1 0;0 1];

%non-linear system simulation
xreal(:,1) = mu;
for i=1 :100
     t= xreal(3,i);
  x= xreal(1,i);
 v= xreal(2,i);
  w= xreal(4,i);
    xreal(:,i+1)=[v;
(4*(0+w^2*sin(t)*0.5 - 0.5*x +
cos(t)*sin(t)*0.25))/(4-cos(t)^2);
w;
(4*((-0.5*sin(t)) - 0 -
w^2*cos(t)*sin(t)*0.25 +
x*cos(t)*0.25))/(4-cos(t)*sin(t))];
    yreal(:,i) = C*(xreal(:,i));
end

%noisy system
xn(:,1) = mu;
for i=1 :100
     xn(:,i+1) = xreal(:,i+1)+wn(i,:)';
     y(:,i) = C*xn(:,i) + H*vn(i,:)';
end

% %% Extended kalman filter
x_p=zeros(101,4);
x_p(1,:)=mu;
s_p=zeros(4,4,101);
s_p(:,:,1)=eye(4);
s_u=zeros(4,4,100);
x_u=zeros(100,4);

for i=1:100
  c(:,:,i)=[x_p(i,1) 0 0 0;
      0 0 x_p(i,3) 0];

 s_u(:,:,i)=s_p(:,:,i)-
s_p(:,:,i)*c(:,:,i)'*inv(c(:,:,i)*s_p(:,:
,i)*c(:,:,i)'+R)*c(:,:,i)*s_p(:,:,i);

x_u(i,:)=x_p(i,:)+(s_p(:,:,i)*c(:,:,i)'*i
nv(c(:,:,i)*s_p(i)*c(:,:,i)'+R)*(y(:,i)-
(c(:,:,i)*x_p(i,:)')))';



 t= x_u(i,3);
  x= x_u(i,1);
 v= x_u(i,2);
  w= x_u(i,4);

 a = [0, 1,
0,
0;
                2/(cos(t)^2 - 4), 0,
- (2*w^2*cos(t) + cos(t)^2 -
sin(t)^2)/(cos(t)^2 - 4) -
(2*cos(t)*sin(t)*(2*sin(t)*w^2 + 0 - 2*x
+ cos(t)*sin(t)))/(cos(t)^2 - 4)^2,
-(4*w*sin(t))/(cos(t)^2 - 4);
                               0, 0,
0,
1;
 -cos(t)/(4*(cos(t)*sin(t) - 4)), 0,
(2*cos(t) + (w^2*cos(t)^2)/4 -
(w^2*sin(t)^2)/4 - 0 +
(x*sin(t))/4)/(cos(t)*sin(t) - 4) -
((cos(t)^2 -
sin(t)^2)*((cos(t)*sin(t)*w^2)/4 +
2*sin(t) + 0 -
(x*cos(t))/4))/(cos(t)*sin(t) - 4)^2,
(w*cos(t)*sin(t))/(2*(cos(t)*sin(t) -
4))];

 x_p(i+1,:)=[v;
(4*(0+w^2*sin(t)*0.5 - 0.5*x +
cos(t)*sin(t)*0.25))/(4-cos(t)^2);
w;
(4*((-0.5*sin(t)) - 0 -
w^2*cos(t)*sin(t)*0.25 +
x*cos(t)*0.25))/(4-cos(t)*sin(t))]';



   s_p(:,:,i+1)=a*s_u(:,:,i)*a' + Q;
   yk(:,i)=C*x_u(i,:)';
end


plot(1:100,yreal(2,:),1:100,yk(2,:))
legend('true measurement','EKF estimated
measurement'), axis([0 100 -0.5
0.5]),xlabel('time'),ylabel('Position
observations')
```