LABORATORY REPORT

STATISTICAL SIGNAL PROCESSING AND ESTIMATION THEORY (STATES)

By:

OKE ADEBAYO OLAYINKA

&

AROMOSE QUDUS AYINDE

SUBMITTED TO: PROF. E. LE CARPENTIER

Program: Robotics and Control

Pogram Specialization: M1 Electric Vehicles Propulsion and Control.

December 20, 2023.

OBJECTIVES:

To estimates online angular position $\theta(t)$ and angular velocity $\Omega(t)$ for DC motor using Kalman's filter.

1. INPUT VOLTAGE SIMULATION

A MATLAB function which provides this sampled input for a duration D: U = inputvoltage(D,A,Delta,Ts), where U is a column vector which contains the sampled input (use square). The input voltage u(t) is a zero-mean square wave with period $\Delta = 0.1$ s, and peak-to-peak amplitude A = 0.1 V. This signal is sampled with sample time Ts = 0.001 s. It is shown in Figure 1.

```
Duration     = 0.5;
Ts           = 0.001;
delta        = 0.1;
amplitude    = 0.1;
T            = 0.020;
G            = 50;
L            = 512;
x1           = [0.1,0];
q            = 1e2;
T_filter           = 25e-3;

function [u,t] = inputvoltage(D,A,Delta,Ts)
    t =(0:Ts:D);
    u = A/2*square(2*pi*t/Delta);
end
```

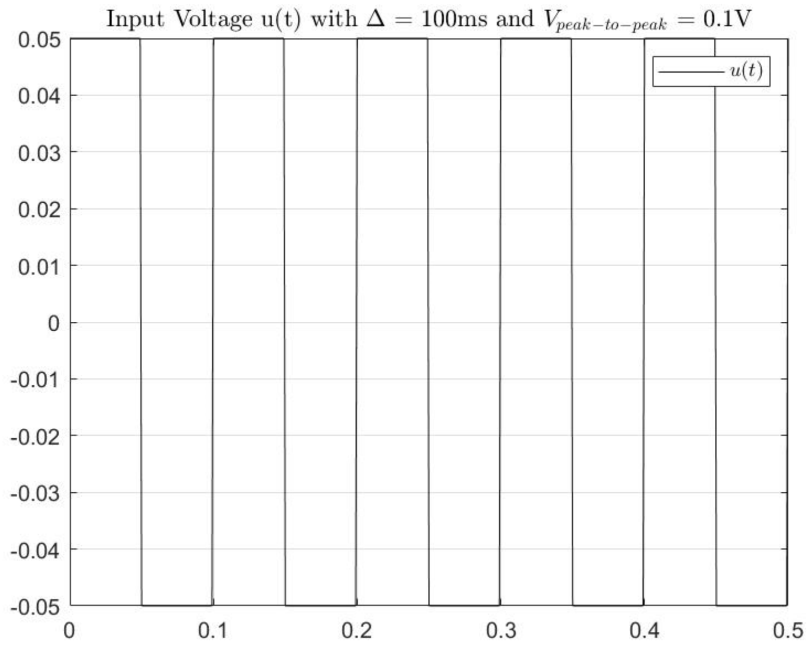

Figure 1: Sampled input for a duration D

2. DETERMINISTIC MODEL AND SIMULATION

The deterministic model is give as with the state given as: $x(t) = \begin{bmatrix} \theta(t) \\ \Omega(t) \end{bmatrix}$

$$\begin{cases} \theta(t) = [1 \quad 0]x(t) + [0]u(t) \\ \dot{x}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 1/T \end{bmatrix} x(t) + \begin{bmatrix} 0 \\ G/T \end{bmatrix} u(t) \end{cases}$$

a. function [Y,X] = simulate(U,G,T,Ts,L,x), where Y is a column matrix (same size as U) which contains the evolution of the output y[n], X evolution of the state vector; x is the initial state vector.

```
function [yt, y,x] =  simulate(u,G,T,Ts,L,x1)
    [sysd,A,B,C]    = getdigitizedsystem(G,T,Ts);
    N               = length(u);
    x               = zeros(2,N);
    y               = zeros(1,N);
    x(:,1)          = x1';


    for n = 1:N-1
        x(:,n+1)    = A* x(:,n) + B*u(n);
        y(n)        = C* x(:,n);

    end
    y(N) = y(N-1);
    yt             = y;
     y             = round(y*L/2/pi)*2*pi/L;

end


function [sysd,A,B,C] = getdigitizedsystem(G,T,Ts)

    A_       = [0 1; 0 -1/T];
    B_       = [0; G/T];
    C_       = [1 0];
    sysc    = ss(A_,B_,C_,0);
    sysd    = c2d(sysc,Ts);
    A       = sysd.A;
    B       = sysd.B;
    C       = sysd.C;

end
```

b. The simulator is tested with G = 50 rad.s −1 .V −1 et T = 0.020 s.

Figure 2 and figure 3 shows the output and states plot.

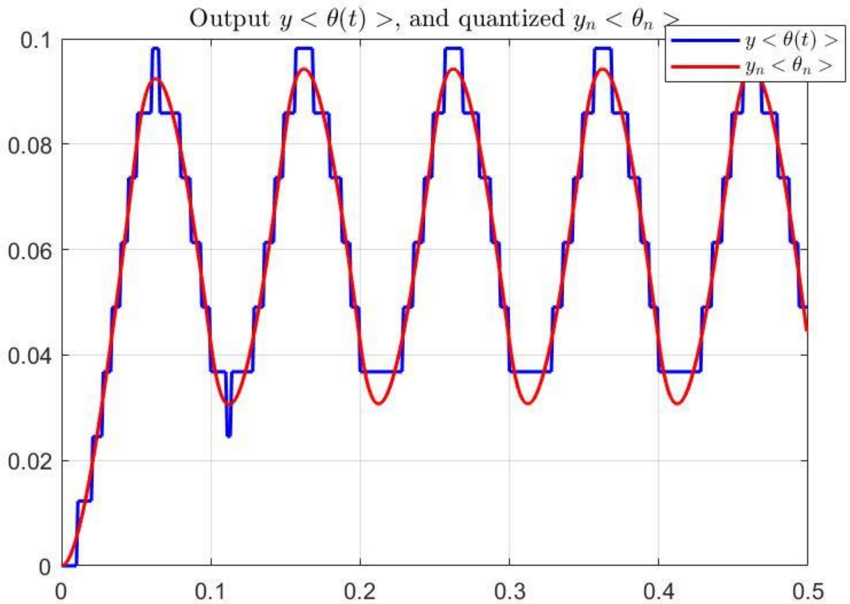

Figure 2: Output y and quantized Output $y(n)$

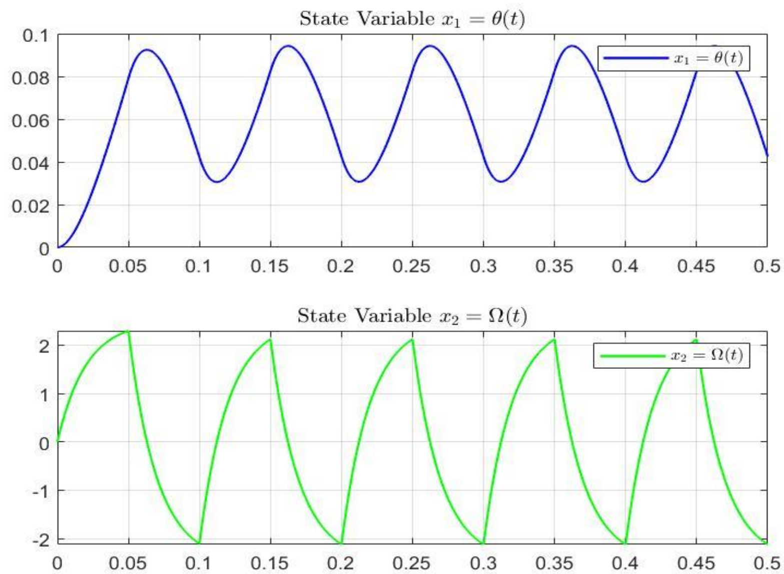

Figure 3: The plot of the position $\theta(t)$and speed $\Omega(t)$ of the DC motor.

STOCHASTIC LINEAR MODEL

a. The model assumed by the Kalman filter is given below:

$$\begin{cases} y[n] = [0 \quad 1]x[n] + \omega[n] \\ x[n+1] = \tilde{A}x[n] + \tilde{B}u[n] + \tilde{B}v[n] \end{cases} \quad and \quad \begin{cases} var(w[n]) = r \\ var(v[n]) = \tilde{B}q\tilde{B}^T \end{cases}$$

b. A uniform random variable is proposed for then value of r error due to quantization and the $var(w[n]) = r = \frac{\left(\frac{2\pi}{L}\right)^2}{12}$ with the PDF shown in figure 4.

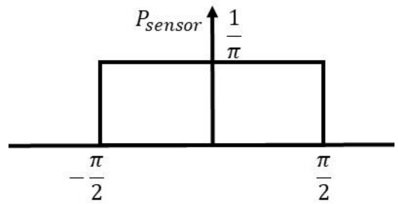

Figure 4: the probability distribution of the error in the position sensor.

KALMAN FILTER

The illustration of the system implementation with kalman filter is shown in Figure 5.

a. Let $\hat{x}^{|0}[1] = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and the process variance $P^{|0}[1] = \begin{bmatrix} \frac{\left(\frac{2\pi}{L}\right)^2}{12} & 0 \\ 0 & 0 \end{bmatrix}$

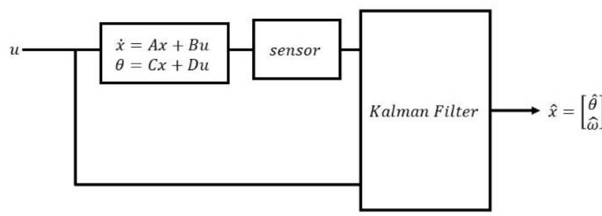

Figure 5: The block diagram for implement the Kalman filter.

Stationary Kalman Filter

```
function xe = stationary_kal(y,u,G,T,Ts,L,x1_0,q)
    y_actual        = y;
    [sysd,A,B,C]    = getdigitizedsystem(G,T,Ts);
    N               = length(u);
    x               = zeros(2,N);
    x(:,1)          = x1_0';
    r               = 4*pi*pi/(12*L*L);

    F               = A;
    H               = C;
    R               = r;
    Q               = B*q*B';

    K=dlqe(A, B, C, q, r)

    %now the Stationary Kalman Loop
```

```
    for n = 1:N-1

        y_pred      = C* x(:,n);

        y_observed  = y_actual(n) ;
        x(:,n)      = x(:,n) + K*(y_observed - y_pred);

        x(:,n+1)    = A*x(:,n) + B*(u(n));
    end

    xe = x;


end
```

Time Varying Kalman Filter

```
function xe = kal(y,u,G,T,Ts,L,x1_0,p1_0,q)
    y_actual        = y;
    [sysd,A,B,C]    = getdigitizedsystem(G,T,Ts);
    N               = length(u);
    x               = zeros(2,N);
    x(:,1)          = x1_0';
    P               = p1_0';
    r               = 4*pi*pi/(12*L*L);
    for n = 1:N-1

        y_pred      = C* x(:,n);
        Cxy         = P*C';
        Cyy         = C*P*C' + r;

        y_observed  = y_actual(n) ;

        x(:,n)      = x(:,n) + Cxy/Cyy*(y_observed - y_pred);
        P           = P   - Cxy*Cxy'/Cyy;

        K_kal(:,n)=Cxy/Cyy
        x(:,n+1)    = A*x(:,n) + B*(u(n));
        P           = A*P*A'   + B*q*B';

    end
    k_ka=K_kal;
    xe = x;

end

function [yt, y,x] =  simulate(u,G,T,Ts,L,x1)
    [sysd,A,B,C]    = getdigitizedsystem(G,T,Ts);
    N               = length(u);
    x               = zeros(2,N);
    y               = zeros(1,N);
    x(:,1)          = x1';
```

```
    for n = 1:N-1
        x(:,n+1)     = A* x(:,n) + B*u(n);
        y(n)         = C* x(:,n);

    end
    y(N) = y(N-1);
    yt              = y;
     y              = round(y*L/2/pi)*2*pi/L;

end
```

SIMULATIONS OF THE KALMAN FILTER WITH THE DC MOTOR.

TUNING OF q,

- We first simulated the Kalman filter algorithm for $(a)\ q = 0.01$ and$(b)\ q = 25\ (q \longrightarrow \infty)$ to test the algorithm and result is shown in figure for a perfect model, When $\hat{\theta}^{|0}[1] = 0$.

(a) (b)

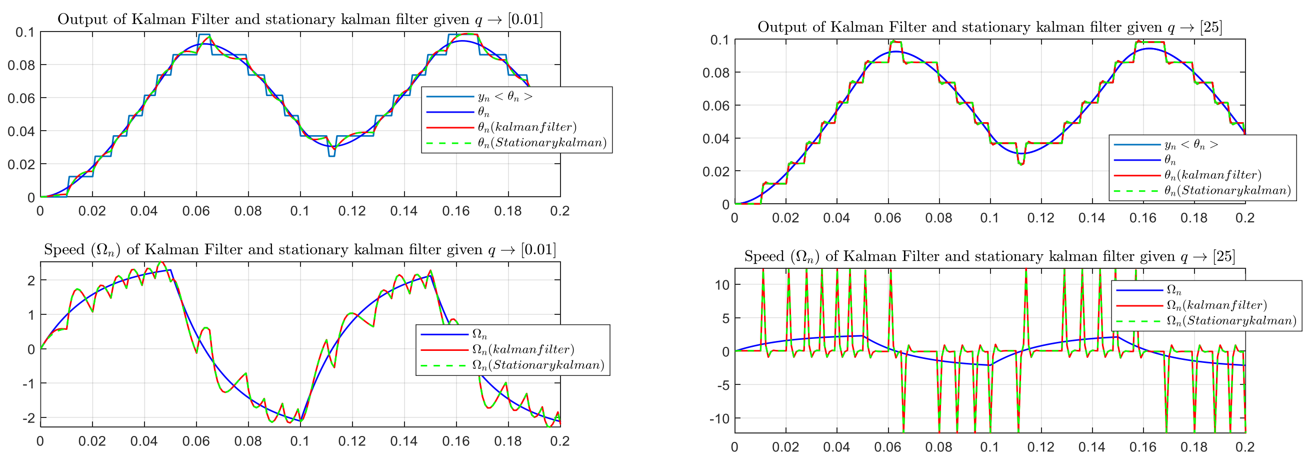

Figure 6: When $\hat{\theta}^{|0}[1] = 0$ for perfect model and $(a)\ q = 0.01$ and$(b)\ q = 25\ (q \longrightarrow \infty)$.

- The estimation of the position and the velocity for both filter values of q (Case1, q=0.001 and Case 2 q=25), (with $\hat{\theta}^{|0}[1] = \theta[1] \pm 0.5$). In our case we simulated for $\hat{\theta}^{|0}[1] = 0.01$ and $\hat{\theta}^{|0}[1] = -0.01$.
- When the model is perfect $\begin{cases} G_{actual} = G_{filter} = 50rad^{-1}V^{-1} \\ T_{actual} = T_{filter} = 0.02s \end{cases}$
- When the model is rough $\begin{cases} G_{actual} = G_{filter} = 50rad^{-1}V^{-1} \\ T_{actual} = 0.02s \\ T_{filter} = 0.025s \end{cases}$

Case 1:

Perfect model $T_{filter} = 0.02s$**:** For (a) $\hat{\theta}^{|0}[1] = 0.01(b)$ $\hat{\theta}^{|0}[1] = -0.01$, result is shown in Figure 7.

(a) (b)

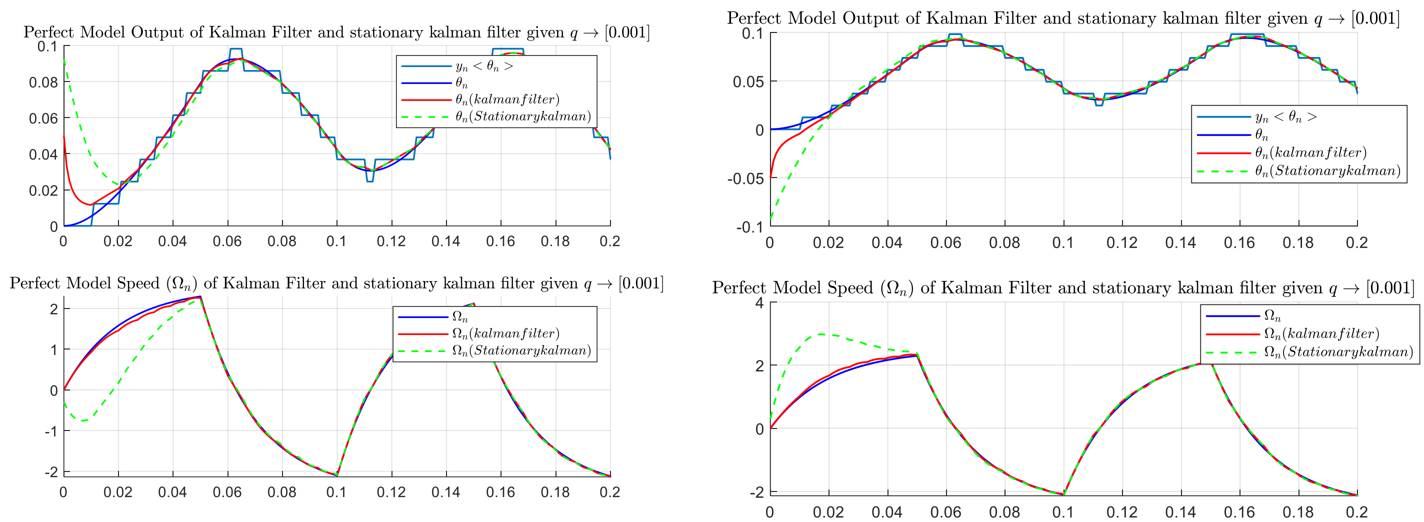

Figure 7: When q=0.001 for perfect model,(a) $\hat{\theta}^{|0}[1] = 0.01(b)$ $\hat{\theta}^{|0}[1] = -0.01$

Rough Model $T_{filter} = 0.025s$**:** For (a) $\hat{\theta}^{|0}[1] = 0.01(b)$ $\hat{\theta}^{|0}[1] = -0.01$, result is shown in Figure 8.

(a) (b)

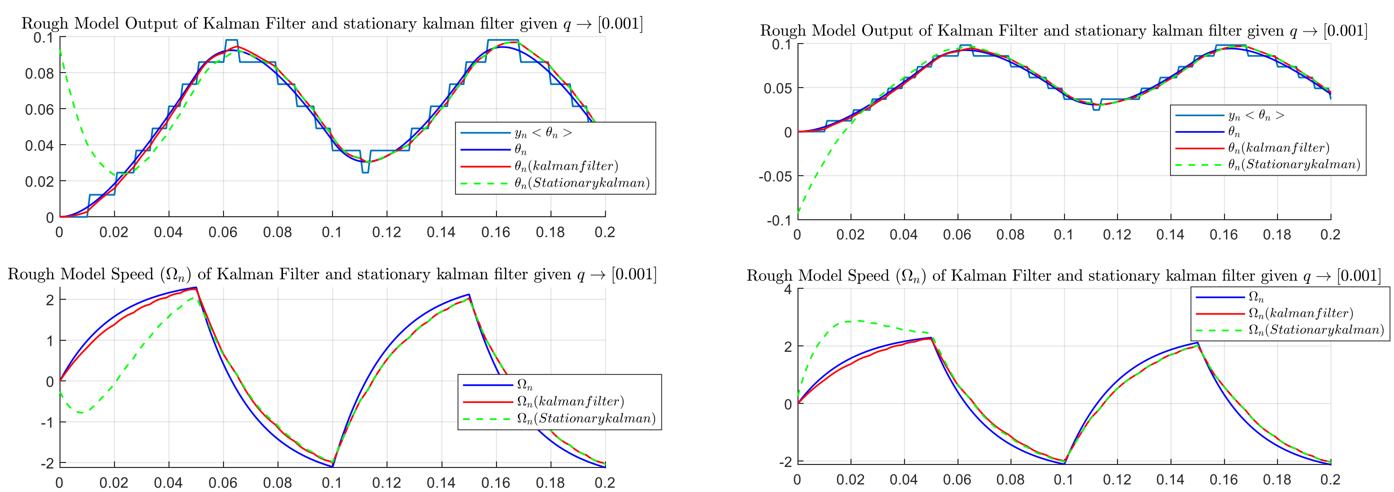

Figure 8: When q=0.001 for rough model,(a) $\hat{\theta}^{|0}[1] = 0.01(b)$ $\hat{\theta}^{|0}[1] = -0.01$.

CASE 2:

Perfect model $T_{filter} = 0.02s$**:** For $(a)\ \hat{\theta}^{|0}[1] = 0.01(b)\ \hat{\theta}^{|0}[1] = -0.01$

When, $q \longrightarrow \infty$, for this case (the sensor is broken) we use a large number of q=25, and result is shown in Figure 9.

(a) (b)

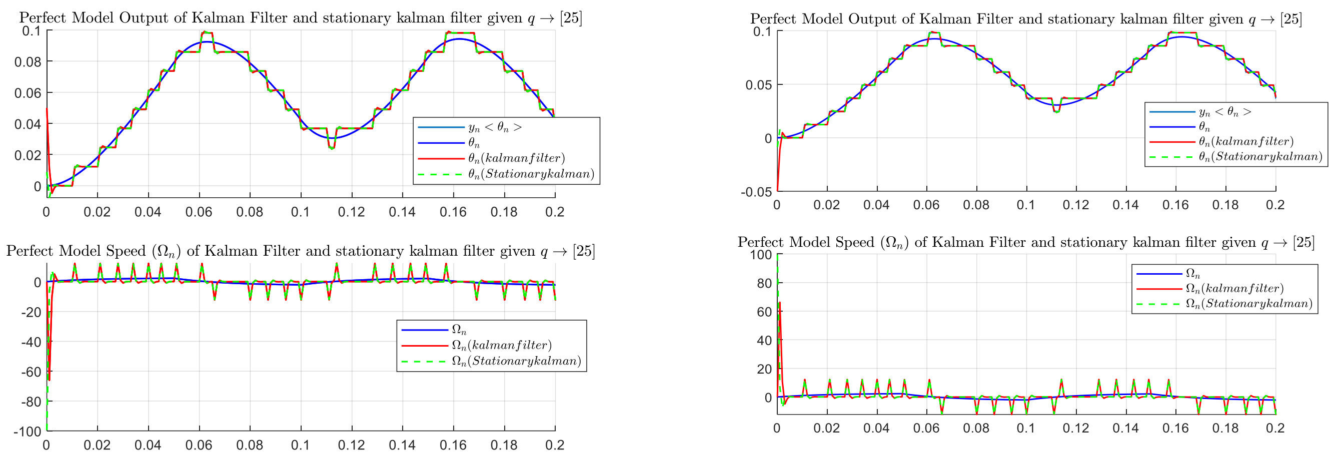

Figure 9: When q=25 for perfect model,$(a)\ \hat{\theta}^{|0}[1] = 0.01(b)\ \hat{\theta}^{|0}[1] = -0.01$.

Rough Model $T_{filter} = 0.025s$**:** For $(a)\ \hat{\theta}^{|0}[1] = 0.01(b)\ \hat{\theta}^{|0}[1] = -0.01$ and result is shown in Figure 7.

(a) (b)

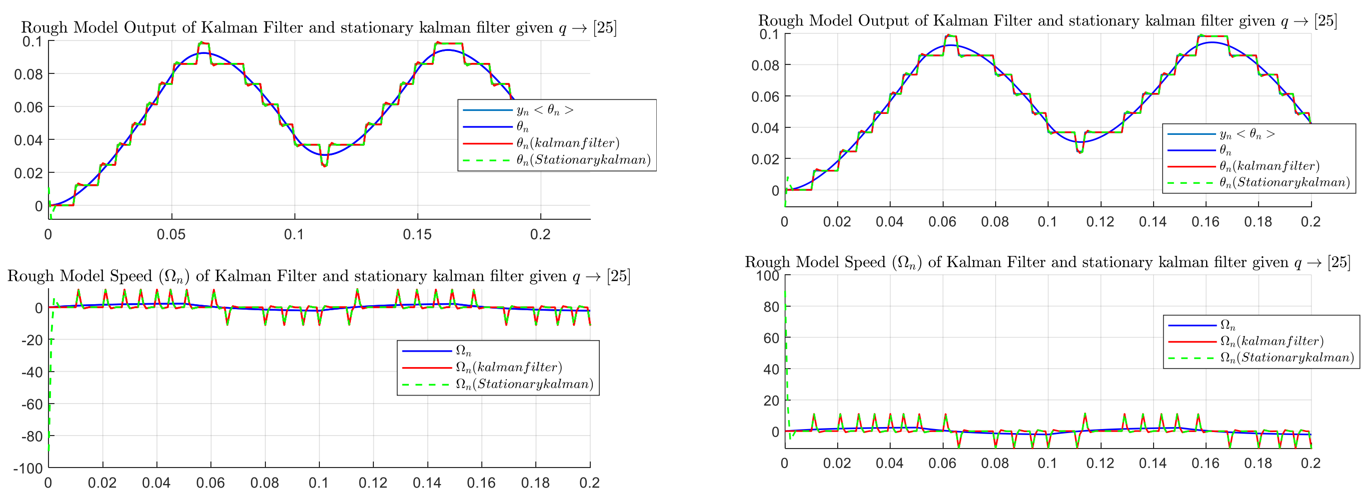

Figure 10: When q=25 for rough model,$(a)\ \hat{\theta}^{|0}[1] = 0.01(b)\ \hat{\theta}^{|0}[1] = -0.01$.

CONCLUSION ON TUNING q:

- The Kalman filter gain for time varying is equal to kalman filter stationary gain as time $t \rightarrow \infty$.
- When q is very small such as $q = 0.001$, the kalman filter choose a value close to the model
- The change in the initial value prediction of the state within the limit $\hat{\theta}^{|0}[1] = \theta[1] \pm 0.5$ does not affect the Kalman filter performance for Stationary and time varying Kalman filter.
- When q is very large such as $q = 25$, the kalman filter chooses the signal from the position sensor because it does not trust the observer model and kalman gain K$\approx$ 1. In this case the position sensor only measures the position and not the velocity hence there is noise in the speed estimation.
- The result of rough model and perfect model produces the same response, Hence within the limit of variation in T_{filter}. Kalman filter filters the parametric variations.
- When q is small such as $q = 0.01$, the kalman filter choose a value in between the model and position sensor as shown in figure 11 for initial value prediction $\hat{\theta}^{|0}[1] = -0.01$.

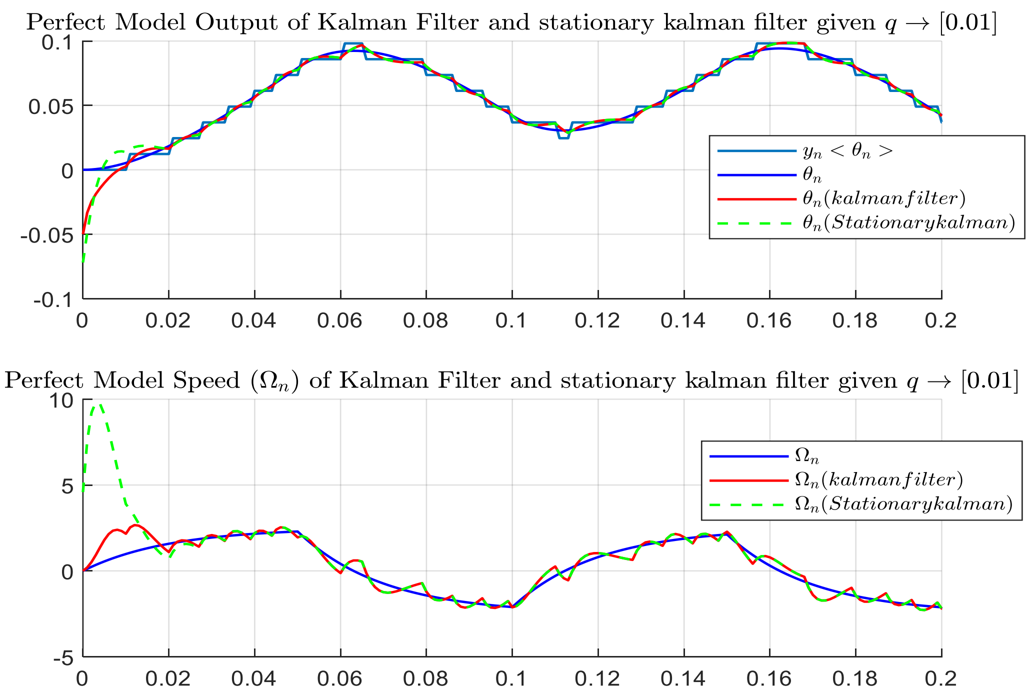

Figure 11: When for perfect model and q=0.01, $\hat{\theta}^{|0}[1] = -0.01$

TUNING r:

If we tuned q=0.001**,** $\hat{\theta}^{|0}[1] = 0.01$,very large $\boldsymbol{r = 10^{10}}$**,** a case where the sensor is broken $r \rightarrow \infty$, result is shown in Figure 12.

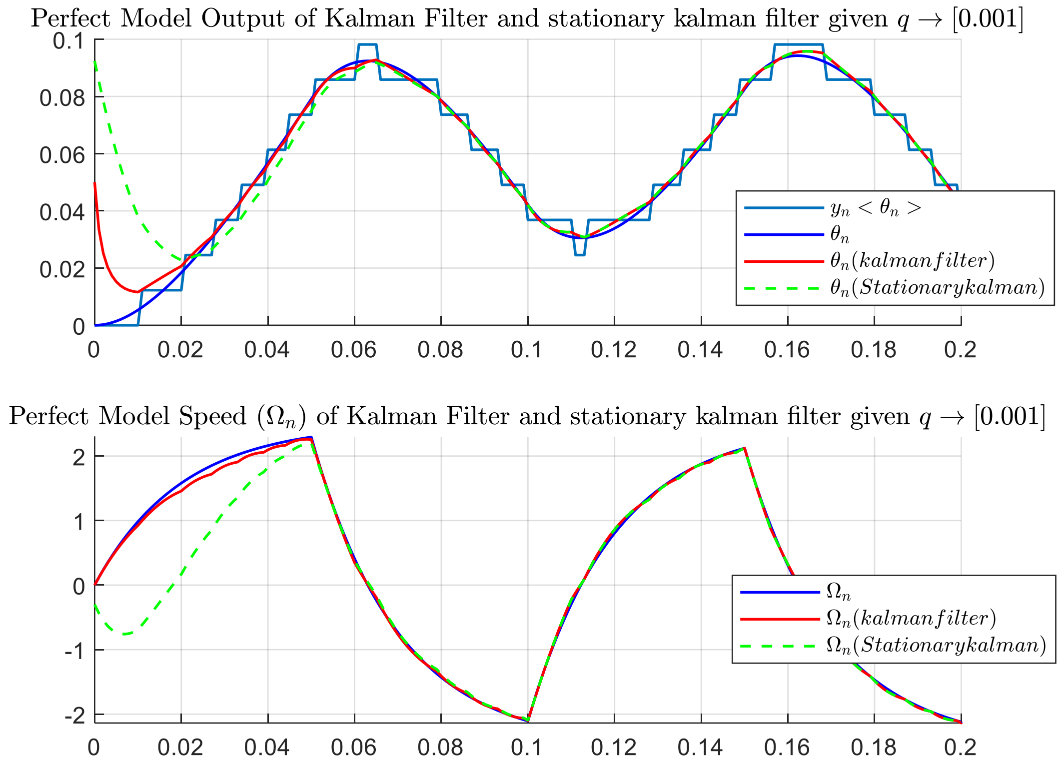

Figure 12: When for perfect model q=0.001 and $r = 10^{10}$, $\hat{\theta}^{|0}[1] = -0.01$

CONCLUSION ON TUNING r:

- When r is large the kalman filter chooses the signals from the observation model for estimation and filters the signal coming from the sensor because of sensor's fault.
- However, the stationary filter did not initially follow the observation model as compared to the asymptotic kalman filter.

TUNING P:

If we tuned $\boldsymbol{P^{|0}[1]} = \begin{bmatrix} \frac{\left(\frac{2\pi}{L}\right)^2}{12} & \mathbf{0} \\ \mathbf{0} & \mathbf{1000} \end{bmatrix}$**,** $\hat{\theta}^{|0}[1] = 0.01$, q=0.001, and $r = \frac{\left(\frac{2\pi}{L}\right)^2}{12}$, a case where we make a wrong prediction of the process variance. , result is shown in Figure 13.

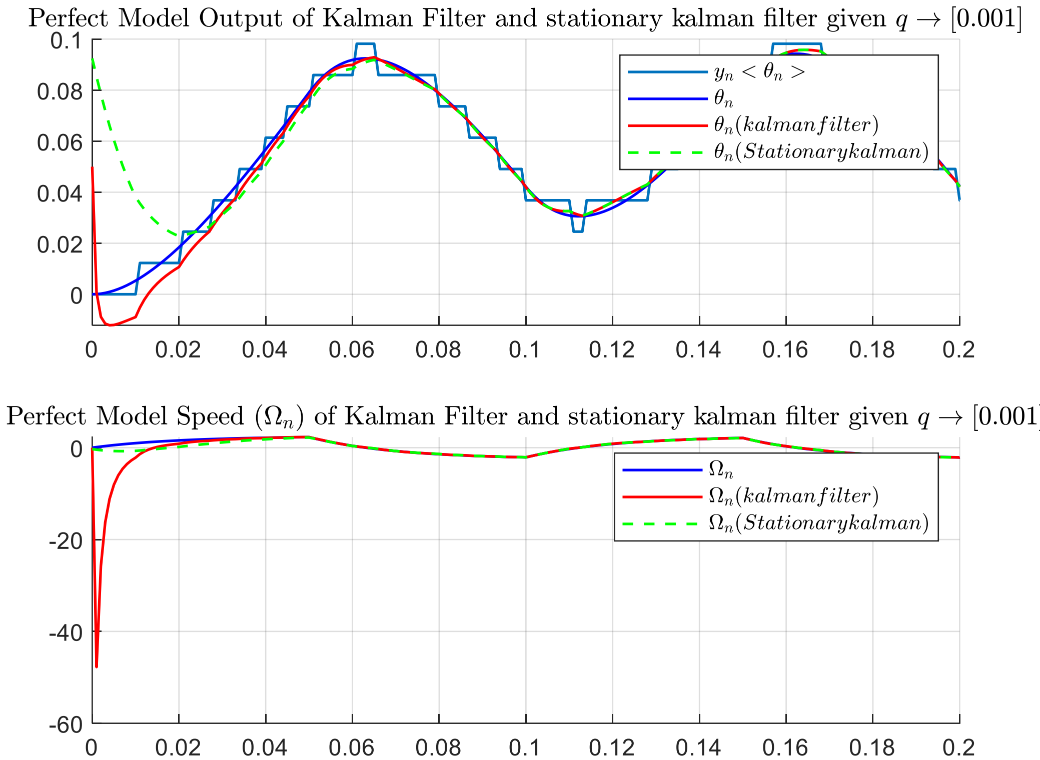

Figure 13: perfect model when $P^{|0}[1] = \begin{bmatrix} \frac{\left(\frac{2\pi}{L}\right)^2}{12} & 0 \\ 0 & 1000 \end{bmatrix}$ q=0.001 and $r = \frac{\left(\frac{2\pi}{L}\right)^2}{12}$, $\hat{\theta}^{|0}[1] = 0.01$

CONCLUSION ON TUNING P:

- When the process variance is tuned from the predicted value, the time varying kalman filter adjusts the $P^{|n-1}[1]$, because it adjusts its kalman gain.

The initial choice of the process variance, Initally affects the performance accuracy of the kalman filter but later adjust for both the stationary kalman filter and the time varying kalman filter. For example we chose $\boldsymbol{P^{|0}[1]} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$, result is shown in Figure 14.

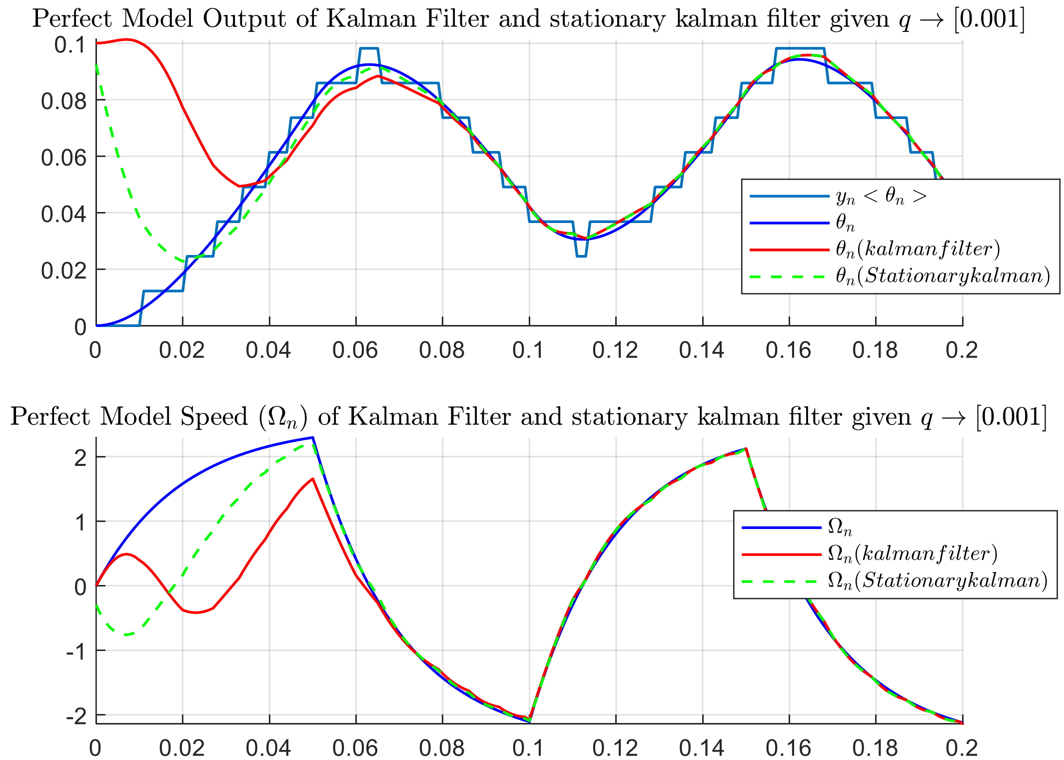

Figure 14: perfect model q=0.001 and $r = 10^{10}$, $\hat{\theta}^{|0}[1] = -0.01$

CONCLUSION ON KALMAN FILTER

- The time varying kalman filter is an asymptotic kalman filter because the kalman gain tend to the kalman gain of the stationary kalman filter and time t tend to ∞.
- Kalman Filter is a recursive filter that linearly optimized the choice of selecting the signal between observation model and sensor signal based on the variance of the sensor and the model observation.
- The kalman filter can be tuned with $\boldsymbol{P}^{|\mathbf{0}}[\mathbf{1}], \boldsymbol{r}, \mathbf{q}$.

SIMULATION AND COMPARISON OF EXTENDED KALMAN, UNSCENTED KALMAN FILTER AND BOOSTSTRAP PARTICLE FILTER

Given the model :

y[n] = h(x[n], n) + w[n]

x[n+1] = f(x[n], n) + v[n]

v[n], w[n], x[1] ARE NORMALLY DISTRIBUTED AND ZERO-MEAN

The functions and derivatives:

f = @(x, n) 0.5*x + 25*x./(1+x.*x)+8*cos(1.2*n)

h = @(x,n) x.*x/20

df = @(x, n) 0.5 + 25*(1-x.*x)./(1+x.*x)./(1+x.*x) % derivative of f

dh = @(x, n) x/10 % derivative of h

P1 = var(x[1]) = 0.1;

Q = Var(v[n]) = 10;

R = Var(w[n]) = 1;

g = @(v, nu) 1/sqrt(2*pi*nu)*exp(-v.*v/2/nu)   % zero mean variance nu gaussian PDF

Figure 15 shows the state trajectory and the observation model.

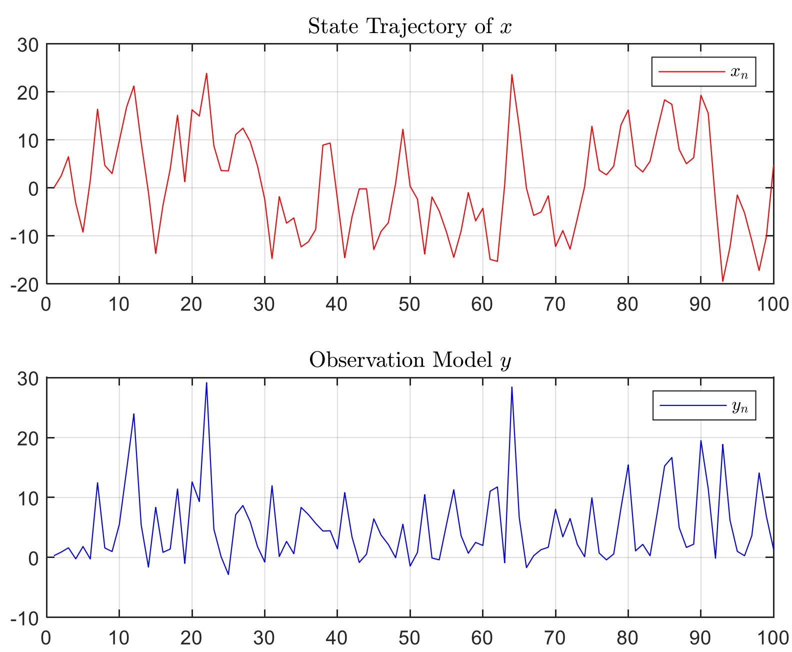

Figure 15: Plot of State Trajectory and Observation Model.

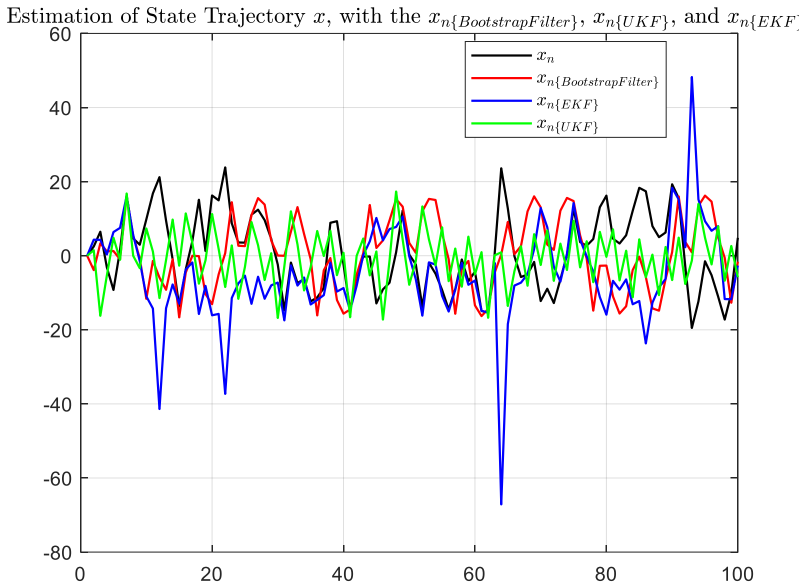

Figure 16: State estimation with Bootstrap Filter, UKF and EKF.

CONCLUSION AND COMPARISON ON PERFORMANCES

According to figure 16, the following can be deduced:

- The booststrap particle filter has the best estimation.
- The Unscented Kalman Filter has a better estimation
- However, the extended Kalman Filter diverges at some points, this divergence reduces it performance.
- These filters can be used to estimate nonlinear functions and the average error from less to high are in the following order: Boostrap Particle filter, Unscented Kalman Filter and Extended Kalman Filter.