MSTAT : Report on Kalman Filter

Course Instructor : **Eric Le Carpentier**

Name : ***Chaitanya Krishna VIRIYALA***

1. Kalman filtering

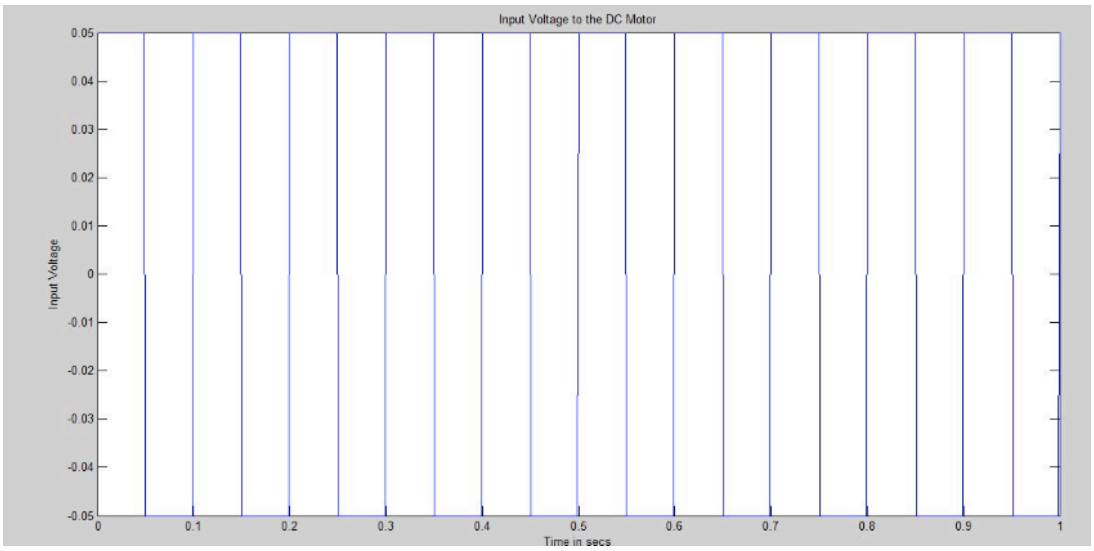
A DC motor is controlled by the armature voltage u(t). The angular position of the rotor θ(t) is measured by an incremental encoder at L = 512 lines per revolution, providing a measurement y(t) of θ(t). Ω(t) is the rotational speed (Ω(t) = θ'(t)). In order to control the speed, we try to estimate online θ(t) and Ω(t), knowing u(t) and y(t).

1.1 Synthesis of system input

The supply voltage u(t) is a period centred slot Δ = 100 ms, peak-to-peak amplitude A = 0.1 V. This signal is sampled at the period Te = 1 ms.

TO DO 1 : Program a MATLAB function to synthesise this sampled input for a duration D, in the form u = input(D, A, Delta, Te), where u is a column vector containing the sampled signal (square).

Given all the informations, we can create the input voltage which gives us a vector having zero mean square wave with delta, sampling time and the input voltage as above. We can have a plot of the same to have a look at how it looks :



Input Signal

1.2 System modelling and simulation

By defining the state vector ; $x(t) = \begin{bmatrix} \theta(t) \\ \Omega(t) \end{bmatrix}$; the following state space representation is obtained:

$$\begin{cases} \dot{x} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{1}{T} \end{bmatrix} x + \begin{bmatrix} 0 \\ \frac{G}{T} \end{bmatrix} u \\ \theta = \begin{bmatrix} 1 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \end{bmatrix} u \end{cases}$$

The input-output signals of the system are sampled at the period T_e. It is assumed that the input u(t) is constant between two sampling instants. The above state representation can therefore be sampled without any approach using the ***index invariance method*** (0-order blocker, MATLAB function c2dm). For any n, and any function f, note f_n = f(nTe).

[Note : Given a continuous-time system governed by the following state representation :

$$\begin{cases} \dot{x}(t) = A\,x(t) + B\,u(t) \\ y(t) = C\,x(t) + D\,u(t) \end{cases}$$

sampling by the method of index invariance at the period Te gives :

$$\begin{cases} x_{n+1} = \tilde{A}\,x_n + \tilde{B}\,u_n \\ y_n = C\,x_n + D\,u_n \end{cases} \text{avec} \begin{cases} \tilde{A} = e^{A\,T_e} \\ \tilde{B} = \int_0^{T_e} e^{A\,\tau}\,B\,d\tau \end{cases}$$

]

The measurement y_n provided by the incremental encoder quantifies the actual angular position θ_n.

[Note : An incremental encoder with L lines per revolution has an accuracy of 2 π/L rad. Under MATLAB: y = round(theta*L/2/pi)*2*pi/L]

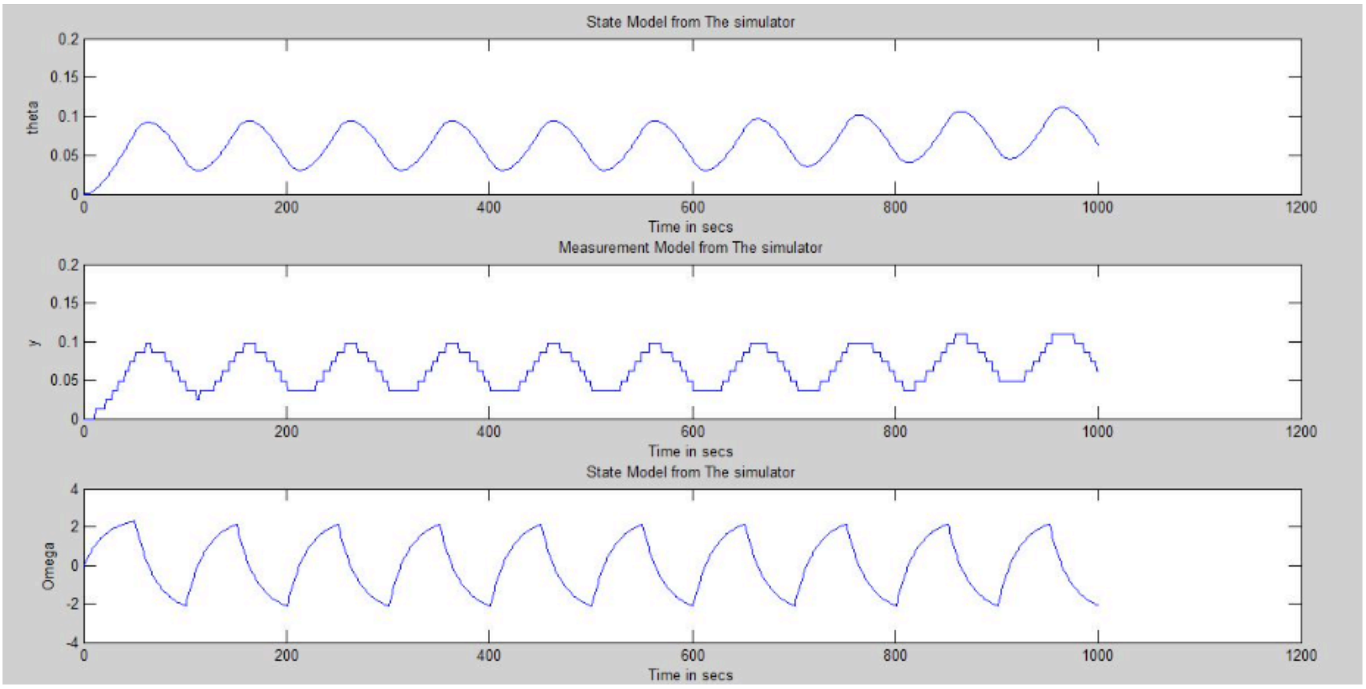
TO DO 2 :

a) Program the simulator (i.e. a function of the shape : [y,x] = simulate(u,G,T,Te,L,x1), where y is a column matrix of the same dimension as u containing the evolution of the measurement y_n, x is a 2-column matrix containing the evolution of the state vector, and x1 is the initialisation of the state vector).

b) Test the simulator by taking G = 50 rad.s-1.V-1 and T = 20 ms.

The simulator Implements the state space model of the system that has been shown above. We first convert the model to a discreet state-space model and then using the input u we had already.

We see the profile of the state x and the measurement y below:



Plot of y, θ and Ω

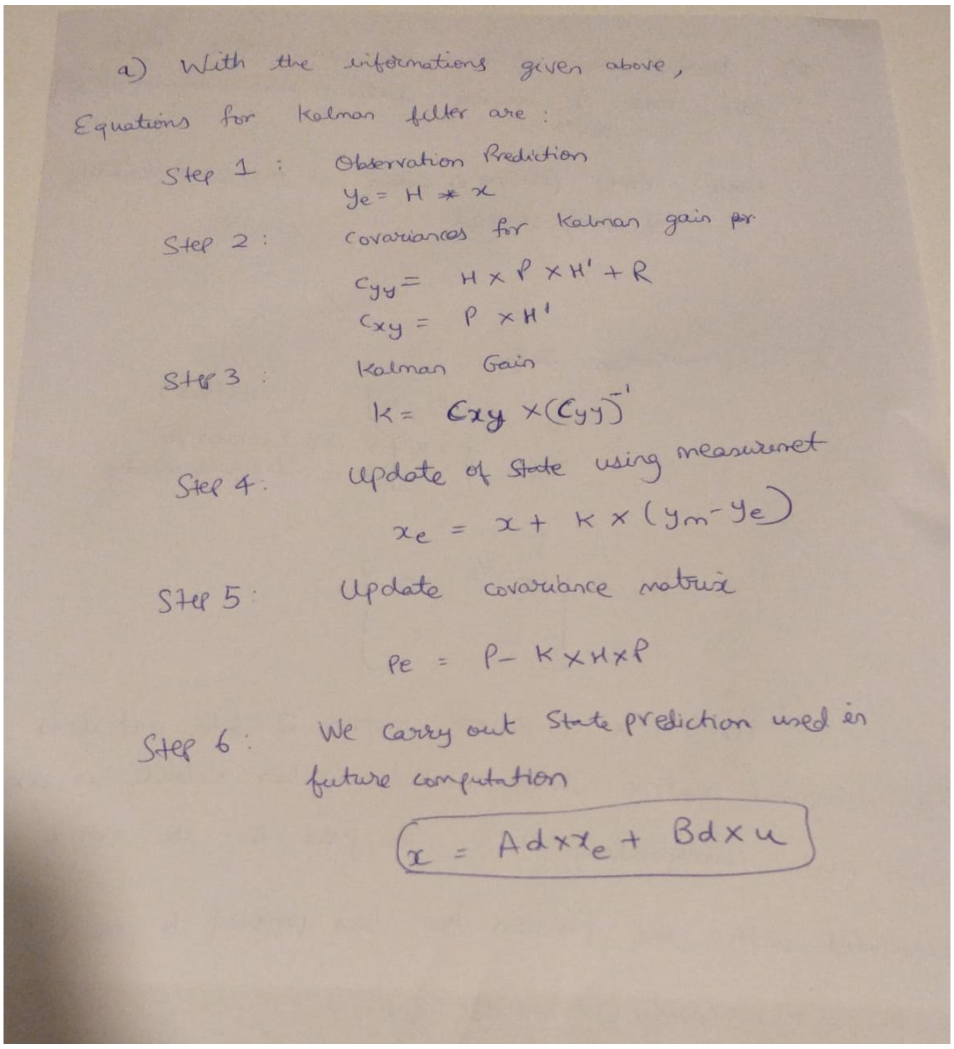
1.3 Estimation by Kalman filtering

The assumed white measurement noise of the incremental encoder (due to the quantization error) is called w_n, r its variance (y_n = θ_n + w_n). To account for possible modelling errors, the actual input to the system is assumed to be u_n+ v_n, where v_n is white noise of variance q, independent of w_n.

In order to carry out the rotor speed control, we try to estimate X_n as well as possible using a Kalman filter. It is known that the motor is not powered and at standstill before the first sampling time, but no information is available on the initial angular position.

TO DO 3 :

a) Write the stochastic state representation of the model of input system one and output system y_n.



a) With the informations given above,

Equations for Kalman filter are :

Step 1 : Observation Prediction

$y_e = H * x$

Step 2 : Covariances for Kalman gain per

$C_{yy} = H \times P \times H' + R$

$C_{xy} = P \times H'$

Step 3 : Kalman Gain

$K = C_{xy} \times (C_{yy})^{-1}$

Step 4 : update of State using measuremet

$x_e = x + K \times (y_m - y_e)$

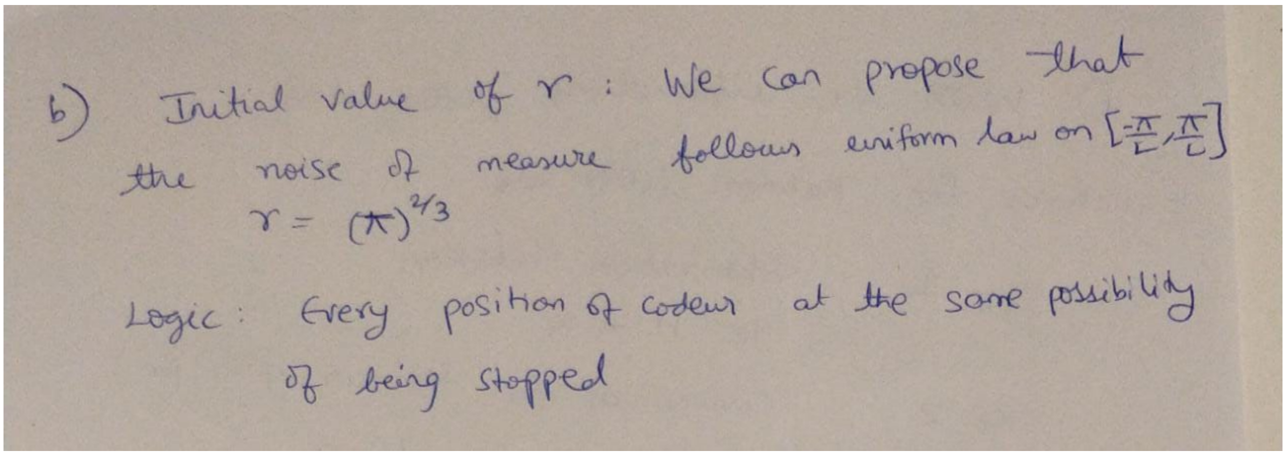
Step 5 : Update covariance matrix

$P_e = P - K \times H \times P$

Step 6 : We carry out State prediction used in future computation

$\boxed{x = Ad \times x_e + Bd \times u}$

b) Propose an r-value based on physical considerations for the incremental encoder.

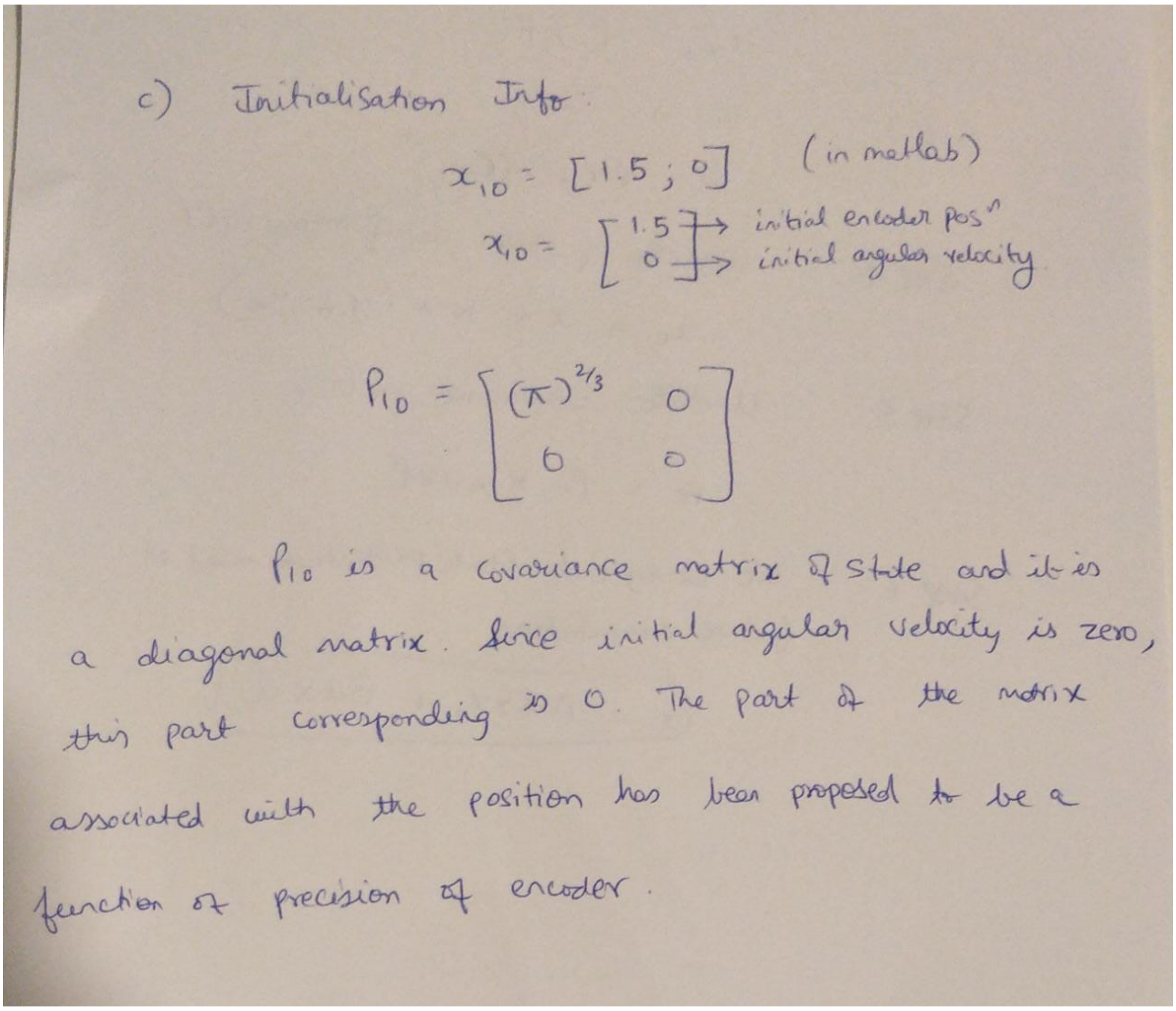


b) Initial value of r : We can propose that the noise of measure follows uniform law on $[\frac{-\pi}{L}, \frac{\pi}{L}]$

$r = (\pi)^{2/3}$

Logic: Every position of codeur at the same possibility of being stopped

c) Propose an initialisation of the prediction of the state Xˆ_1/0 and of the variance of the prediction error P_1/0, based on a priori knowledge.



c) Initialisation Info:

$x_{10} = [1.5 ; 0]$ (in matlab)

$$x_{10} = \begin{bmatrix} 1.5 \\ 0 \end{bmatrix} \begin{array}{l} \rightarrow \text{initial encoder pos}^n \\ \rightarrow \text{initial angular velocity} \end{array}$$

$$P_{10} = \begin{bmatrix} (\pi)^{2/3} & 0 \\ 0 & 0 \end{bmatrix}$$

P_{10} is a covariance matrix of state and it is a diagonal matrix. Since initial angular velocity is zero, this part corresponding is 0. The part of the matrix associated with the position has been proposed to be a function of precision of encoder.

d) Program optimal Kalman filters (time-dependent gain) and stationary (constant gain, dlqe), i.e. two functions of the type xe = kal(y,u,G,T,Te,L,x1_0,P1_0,q), where xe is a 2-column matrix containing the evolution of the state vector estimate, and where x1_0 and P1_0 denote respectively the initialization of the prediction of the state Xˆ_1/0 and the variance of the prediction error P_1/0.

Please refer to the last section for the code

1.4 Simulations

TO DO 4 :

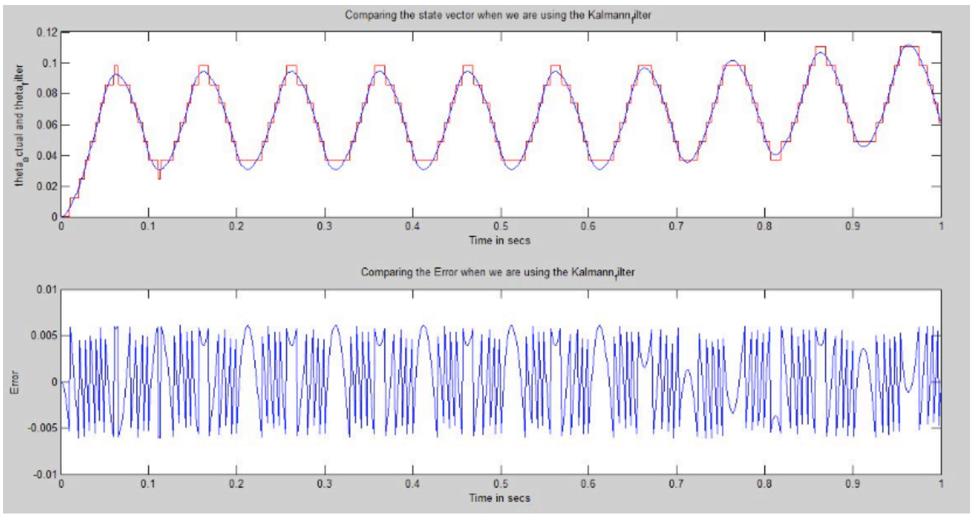
Compare on the estimation of position and speed the performances of the two filters for different values of q (place yourself in the case θˆ_1/0 - θ = ±0, 05), in the following cases :

-
- the system is perfectly modelled : $\begin{cases} G_{\text{exact}} = G_{\text{filtre}} = 50 \text{ rad.s}^{-1}.\text{V}^{-1} \\ T_{\text{exact}} = T_{\text{filtre}} = 20 \text{ ms} \end{cases}$

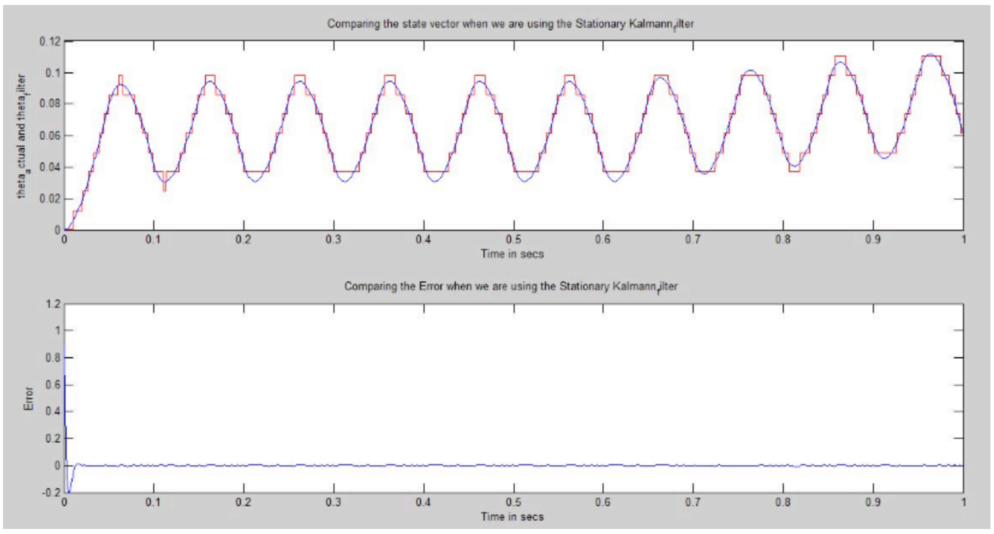
- the model of the system is imprecise $\begin{cases} G_{\text{exact}} = G_{\text{filtre}} = 50 \text{ rad.s}^{-1}.\text{V}^{-1} \\ T_{\text{exact}} = 20 \text{ ms} \\ T_{\text{filtre}} = 25 \text{ ms} \end{cases}$

The "exact" index indicates the value used for the simulation, the "filter" index indicates the value used for the estimation.

Case 1 Plot showing Comparison of State and Error when we use the Kalman Filter

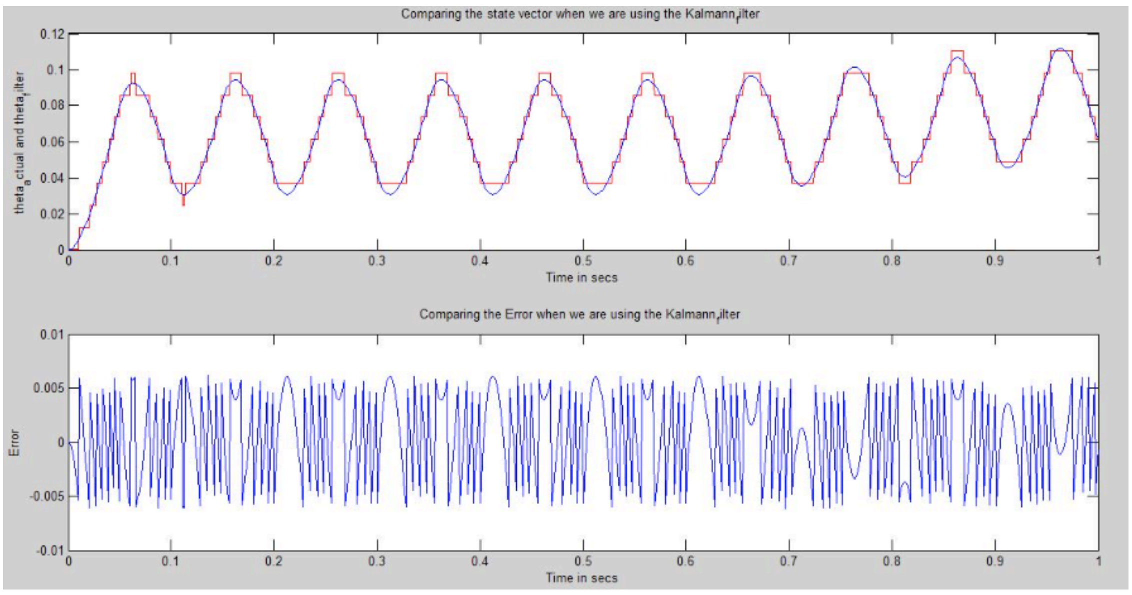


Plot showing Comparison of State and Error when we use the Kalman Filter

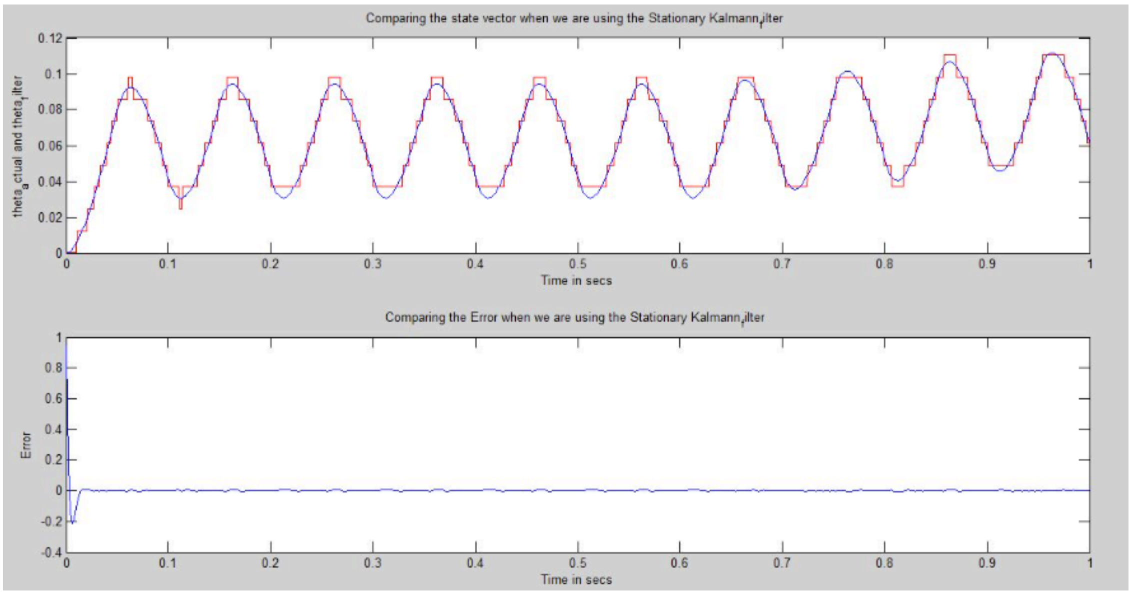


Remark : In the stationary Kalman filter we see that there is an initial error which slowly converges while in the case of the Kalman filter, the convergence is faster and the error stays within a steady state band.

Case 2 : Plot showing Comparison of State and Error when we use the Kalman Filter



Plot showing Comparison of State and Error when we use the Kalman Filter



Remark : When we look at the stationary Kalman filter we see that there is an initial error which slowly converges while in the case of the Kalman filter, the convergence is faster and the error stays within a steady state band.

Code (order is functions first and main later)

Input Voltage :

```
function [u] = input1(D1,A1,Delta,Te)
t=0:Te:D1;
f = 1/Delta;
u = A1*0.5*(square(2*pi*f*t));
% axis([0 0.1 -0.1 0.1]);
plot(t,u);
 ylabel('Input Voltage')
 xlabel('Time in secs')
 title('Input Voltage to the DC Motor')
u = u';
end
```

Simulate Function :

```
function [y, X, Ad,Bd,Cd] = simulate1(u,G,T,Te,L,x1)
x = x1
A = [0 1; 0 -1/T];
B =  [0;G/T];
C = [1 0];
D = [0];
system=ss(A,B,C,D);
sysd=c2d(system,Te,'zoh');
[Ad,Bd,Cd,Dd]= ssdata(sysd);
X = zeros(2, length(u));
y = zeros(1, length(u));
for i = 1 : length(u)
    y(i) = Cd*x;
    X(:,i) = x;
    x = Ad*x + Bd*u(i);
end
y = round(y*L/2/pi)*2*pi/L;
```

```
figure(1)
subplot(3,1,1)
plot(X(1,:));
 ylabel('theta')
 xlabel('Time in secs')
 title('State Model from The simulator')
 subplot(3,1,2)
plot(y);
 ylabel('y')
 xlabel('Time in secs')
  title('Measurement Model from The simulator')
 subplot(3,1,3)
  plot(X(2,:));
 ylabel('Omega')
 xlabel('Time in secs')
 title('State Model from The simulator')
end
```

Kalman Filter :

```
function [X E] = kalman_filter(y,u,G,T,Te,L,x1_0,p1_0,q)
  % This function is used to prepare the kalman filter
x=x1_0;
P=p1_0;
X_E = zeros(2, length(u));
A = [0 1; 0 -1/T];
B =  [0;G/T];
C = [1 0];
D = [0];
R = (2*pi/L)^2/12;


system=ss(A,B,C,D);
sysd=c2d(system,Te,'zoh');
[Ad,Bd,Cd,Dd]= ssdata(sysd);
H = Cd;
for i = 1 : length(u)
% State prediction
% Observation Prediction y e = H*x;
Cyy = H*P*H' + R;
```

```
Cxy = P*H';
%  Compute Kalman Gain
K = Cxy*inv(Cyy);
% Update the state estimate with Measurement
x_e = x + K*(y(i)-y e); X_E(:,i) = x_e;
% Update the error in co-variance
p_e = P - K*H*P;
%state Prediction
x = Ad*x_e + Bd*u(i);
P = Ad*p_e*Ad' + Bd'*q*Bd;
end


end
```

Stationary Kalman Filter :

```
function [X E2] =
Stationary_Kalman(y,u,G,T,Te,L,x1_0,p1_0,q)
 % This function is used to prepare the kalmann filter
x=x1_0;
P=p1_0;
X_E2 = zeros(2, length(u));
A = [0 1; 0 -1/T];
B =  [0;G/T];
C = [1 0];
D = [0];
R = (*pi/)^2/3;


system=ss(A,B,C,D);
sysd=c2d(system,Te,'zoh');
[Ad,Bd,Cd,Dd]= ssdata(sysd);
[K,P,~,~] = dlqe(Ad,Bd,Cd,q,R); H = Cd;


for i = 1 : length(u)
% State prediction
% Observation Prediction
y_e = H*x;
% Update the state estimate with Measurement
x_e = x + K*(y(i)-y_e);
```

```
X_E2(:,i) = x_e;
% Update the error in co-variance
p_e = P - K*H*P;
%state Prediction
x = Ad*x_e + Bd*u(i);
end


end
```

Main Function :

```
%Clear All variables
clear all
close all
clc


% Declare global variables
A1 = 0.1;
Te =1*10^-3;
Delta = 100*10^-3;
D1 = 1;
G = 50;
T = 20*10^-3;
Tf = 25*10^-3;
L = 512;
q = 0.05;
x1 = [0;0];
x1_0 = [1.5;0];
p1_0 = (pi)^2/3;
p1_0= [p1_0 0; 0 0];


% Testing functions and Plots
u = input1(D1,A1,Delta,Te);
[y, X, Ad,Bd,Cd] = simulate1(u,G,T,Te,L,x1);
[X_E] = kalmann_filter(y,u,G,Tf,Te,L,x1_0,p1_0,q)
errortheta = X_E(1,:) - X(1,:)
figure(2)
subplot(2,1,1)
plot ((0:Te:D1),X_E(1,:),'r');
```

```
hold on
plot ((0:Te:D1),X(1,:));
ylabel('theta actual and theta filter')
 xlabel('Time in secs')
title('Comparing the state vector when we are using the
Kalman filter ')
subplot(2,1,2)
plot ((0:Te:D1),errortheta);
ylabel('Error')
xlabel('Time in secs')
title('Comparing the Error when we are using the Kalman
filter ')
[X_E2] = Stationary_Kalman(y,u,G,Tf,Te,L,x1_0,p1_0,q);
figure(3)
subplot(2,1,1)
plot ((0:Te:D1),X_E(1,:),'r');
hold on
plot ((0:Te:D1),X(1,:));
ylabel('theta actual and theta filter')
 xlabel('Time in secs')
title('Comparing the state vector when we are using the
Stationary Kalman filter ')
errortheta2 = X_E2(1,:) - X(1,:);
 subplot(2,1,2)
plot ((0:Te:D1),errortheta2);
ylabel('Error')
xlabel('Time in secs')
title('Comparing the Error when we are using the Stationary
Kalman filter ')
```

References :

Class Notes

DeepL for translation of questions from French to English

https://www.mathworks.com/help/predmaint/ug/Fault-Detection-Using-an-Extended-Kalman-Filter.html

http://www.aessweb.com/pdf-files/jasr-3(12)-1157-1172.pdf