Kalman Filter Implementation: CH5120 Mini Project

Krthan Musugunthan - AE21B039

October 2024

1 Introduction

Based on the paper *"The Quadruple-Tank Process: A Multivariable Laboratory Process with an Adjustable Zero",* by **Karl Henrik Johansson** we use the setup described to implement a Kalman filter and a Particle filter to estimate the measured and unmeasured states in the presence of state and measurement noise.

2 Setup

Four tanks are named according to the schematic shown below is arranged, there is a reservoir collecting from the openings of Tank 1 and 2. Two pumps, A (pumps into Tank 1 and 4 through a distribution valve 1) and B (pumps into tank 2 and 3 through a distribution valve 2) circulates liquid collected in the reservoir.
In the paper, the author performs his study using 2 operating points based on the phase characteristics ($P_{\ominus}$) and ($P_{\oplus}$). However, this project on focuses on the $P_{\ominus}$ operating point.

3 System dynamics

The system's dynamical equation has been given by the following non-linear equation:

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}\nu_1$$
$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_4}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}\nu_2$$
$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{(1-\gamma_2)k_2}{A_3}\nu_2$$
$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{(1-\gamma_1)k_1}{A_4}\nu_1$$

where, A_i is the area of cross section of tank i; a_i is the area of cross-section of the outlet hole ; h_i is the water level; γ_i is the valve position; ν_i is the voltage applied to pump i;$k_i\nu_i$ is the corresponding flow; g is the acceleration of gravity.

The values of the parameters of the system are presented below: The operating points used in the study are:

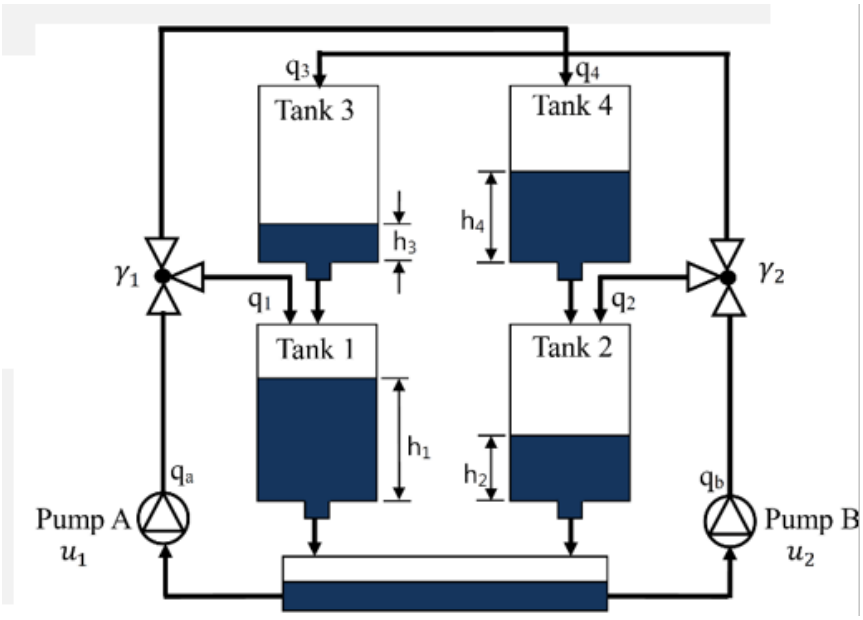


Figure 1: Setup of the experiment

3.1 Steady state solution

Using the non-linear equations, with an initial conditions/operating point we can get the steady state solutions of the height of fluid in tanks. A MATLAB [6] code was implemented, the ode45 solver was used to solve the differential equations. The following the code script used for defining the non-linear equations

```
function dxdt = fourtank_non_lin(t,x, params)

    A1 = params(1); A2 = params(2); A3 = params(3); A4 = params(4);

    a1 = params(5); a2 = params(6); a3 = params(7); a4 = params(8);

    kc = params(9); g = params(10);

    gamma1 = params(11); gamma2 = params(12);

    k1 = params(13); k2 = params(14); v1 = params(15); v2 = params
        (16);

    h1 = x(1); h2 = x(2); h3 = x(3); h4 = x(4);

    dh1 = (-a1/A1) * sqrt(2*g*h1) + (a3/A1) * sqrt(2*g*h3) + (gamma1
         * k1 * v1)/A1;
```

A_1, A_3	[cm^2]	28
A_2, A_4	[cm^2]	32
a_1, a_3	[cm^2]	0.071
a_2, a_4	[cm^2]	0.057
k_c	[V/cm]	0.50
g	[cm/s^2]	981.

Figure 2: Parameters of the system

		$P\_$
(h_1^0, h_2^0)	[cm]	(12.4, 12.7)
(h_3^0, h_4^0)	[cm]	(1.8, 1.4)
(v_1^0, v_2^0)	[V]	(3.00, 3.00)
(k_1, k_2)	[cm^3/Vs]	(3.33, 3.35)
(γ_1, γ_2)		(0.70, 0.60)

Figure 3: Operating point

```
    dh2 = (-a2/A2) * sqrt(2*g*h2) + (a4/A2) * sqrt(2*g*h4) + (gamma2
        * k2 * v2)/A2;

    dh3 = (-a3/A3) * sqrt(2*g*h3) + ((1 - gamma2) * k2 * v2)/A3;

    dh4 = (-a4/A4) * sqrt(2*g*h4) + ((1 - gamma1) * k1 * v1)/A4;

    dxdt = [dh1; dh2; dh3; dh4];

end
```

For generating data, another script was used, this script generates the measurement values, state values and stores it in a text file which can later be read as a matrix for implementing the Kalman filter. the steady state points are Tank 1: 12.2631, Tank 2: 12.7828, Tank 3: 1.6339, Tank 4: 1.4090.

```
%% Data Generation code for the Quadruple-Tank problem
% Variable initialization
```

```
A1 = 28; A2 = 32; A3 = 28; A4 = 32; %(in cm^2)

a1 = 0.071; a2 = 0.057; a3 = 0.071; a4 = 0.057; %(in cm^2)

kc = 0.5;

g = 981; % in cm/s^2

gamma1 = 0.7; gamma2 = 0.6;

k1 = 3.33; k2 = 3.35;

v1 = 3; v2 = 3;

h0 = [12.4; 12.7; 1.8; 1.4]; %operating point/inital condition

params = [A1, A2, A3, A4, a1, a2, a3, a4, kc, g, gamma1, gamma2, k1,
    k2, v1, v2]; %combinign all variables for input into the
   function

step_size = 0.05;
tspan = 0:step_size:500; %10000 time values for 500 seconds

[t,x] = ode45(@(t,x)fourtank_non_lin(t,x,params), tspan, h0); %
   function call, diff eqn solver ode45
%%
%plots the variation of heights in the tanks
figure;
plot(x(1:10000,1))
hold on;
plot(x(1:10000,2))
hold on;
plot(x(1:10000,3))
hold on;
plot(x(1:10000,4))
hold off;
title('Tank Height')
xlabel('Iterations')
ylabel('Height in each tank (in cm)')
legend('Tank 1', 'Tank 2', 'Tank 3', 'Tank 4')

%%

x(end,:); % steady state values
C = [kc 0 0 0; 0 kc 0 0]; %measurement matrix
```

```
z_true = (C * x')'; %measurement values
time_state_meas_true_array = [t z_true x]; %making a matrix to be
    written in a file
writematrix(time_state_meas_true_array, 'measurement_state_time_true
    ', 'Delimiter', '\t') %writing the data file
```

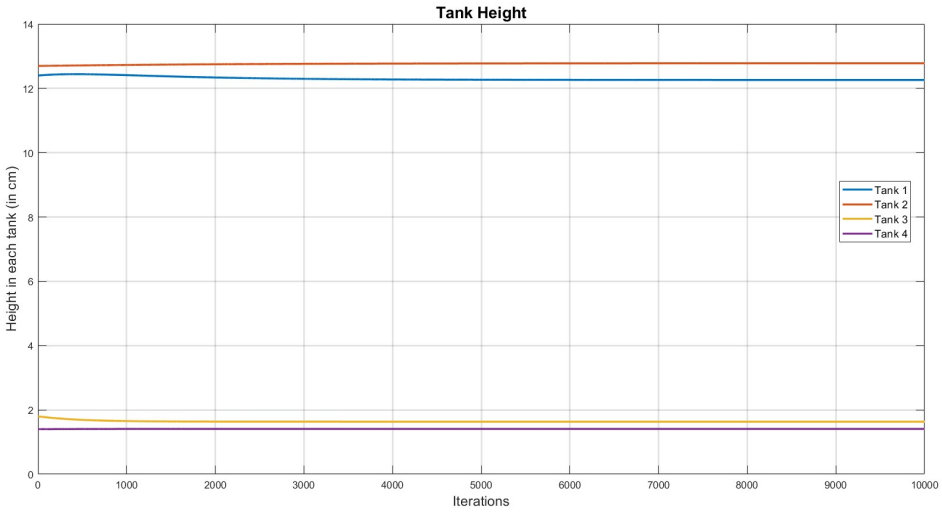


Figure 4: Generated data using non-linear dynamics

The measurements, and state data points are written into a file named $'measurement\_state\_time\_true.txt'$

4 Kalman Filter

4.1 Linearizing the model

For implementing a Kalman filter, we need to linearize the model, the model has been linearized in the paper and we use the state space formulation for our Kalman filter implementation. The linearized fomulation :

$$\frac{dx}{dt} = \begin{bmatrix} -\frac{1}{T_1} & 0 & \frac{A_3}{A_1 T_3} & 0 \\ 0 & -\frac{1}{T_2} & 0 & -\frac{A_4}{A_2 T_4} \\ 0 & 0 & -\frac{1}{T_3} & 0 \\ 0 & 0 & 0 & -\frac{1}{T_4} \end{bmatrix} x + \begin{bmatrix} \frac{\gamma_1 k_1}{A_1} & 0 \\ 0 & \frac{\gamma_2 k_2}{A_2} \\ 0 & \frac{(1-\gamma_2)k_2}{A_3} \\ \frac{(1-\gamma_1)k_1}{A_4} & 0 \end{bmatrix} u$$

$$y = \begin{bmatrix} k_c & 0 & 0 & 0 \\ 0 & k_c & 0 & 0 \end{bmatrix} x$$

where the constants are given by:

$$T_i = \frac{A_i}{a_i}\sqrt{\frac{2h_i^0}{g}}, \qquad i = 1, 2, 3, 4$$

4.2 Implementation

Using the $P_{\ominus}$ operating point, we get the A and B matrices. The following MATLAB$^{\circ}$code was implemented for this.

```
clc; close all; clear all;

%parameters of the system
A1 = 28; A2 = 32; A3 = 28; A4 = 32;
a1 = 0.071; a2 = 0.057; a3 = 0.071; a4 = 0.057;
kc = 0.5; g = 981;

%Operating point parameters
h1_o = 12.4; h2_o = 12.7; h3_o = 1.8; h4_o = 1.4;
h_op = [h1_o; h2_o; h3_o; h4_o]; %operating point
T1 = (A1/a1) * sqrt(2*h1_o/981); T2 = (A2/a2)* sqrt(2*h2_o/981); T3
   = (A3/a3) * sqrt(2 * h3_o /981); T4 = (A4/a4) * sqrt(2 * h4_o
   /981);
v1 = 3; v2 = 3;
k1 = 3.33; k2  = 3.35;
gamma1 = 0.70; gamma2 = 0.60;

% defining the A, B and H matrices
A = [-1/T1 0 A3/(A1*T3) 0; 0 -1/T2 0 A4/(A2*T4); 0 0 -1/T3 0; 0 0 0
   -1/T4 ];

B = [gamma1*k1/A1 0; 0 gamma2*k2/A2; 0 (1-gamma2)*k2/A3; (1-gamma1)*
   k1/A4 0];

H = [kc 0 0 0; 0 kc 0 0 ];

N = 10000; % number of iterations

%% Kalman filter
X_post(:,1) = h_op; % intialization
P_0 = 1 * eye(4); % initial co-variance matrix
P_post(:,:,1) = P_0; % intial covariance
P_post_value(1) = det(P_post(:,:,1));
Q = 0.1 * eye(4) ; R = 0.01 * eye(2);
true_values = readmatrix("measurement_state_time_true.txt"); %read
   values from data generated
z_meas = true_values(:,2:3)'; %measured values
x_true = true_values(:,4:7)'; %state values
```

```
rng(10, 'twister'); %random number generation setting

for i = 2:N+1
%measurement noise is  10 times lesser compared to process noise is
    the
%assumption
    w = sqrt(0.1) * randn(size(A,1),1); %state/process noise
    v = sqrt(0.01) * rand(size(H,1),1); %measurement noise

    %Prediction steps start here
    X_pri(:,i) = A * X_post(:,i-1) + B * [v1;v2] + h_op + w; %
       apriori  state estimate

    P_pri(:,:,i) = A * P_post(:,:,i-1) * A' + Q; %apriori covariance

    P_pri_value(i) = det(P_pri(:,:,i));

    K(:,:,i) = P_pri(:,:,i) * H' * inv(H * P_pri(:,:,i) * H' + R); %
       kalman gain

    z_est(:,i) = H * X_pri(:,i); %measurement estimate from apriori
       estimate

    error_pri(:,i) = z_meas(:,i) + v - z_est(:,i); %Innovation/
       apriori residue

    %prediction steps end here

    %correction steps start here

    X_post(:,i) = X_pri(:,i) + K(:,:,i) * error_pri(:,i); %
       Aposteriori state estimate

    error_post(:,i) = z_meas(:,i) + v - (H * X_post(:,i)); %
       aposteriori residue

    P_post(:,:,i) = P_pri(:,:,i) - K(:,:,i) * H * P_pri(:,:,i); %
       aposteriori covariance

    %correction steps end here

    P_post_value(i) = det(P_post(:,:,i));

end

X_pri(:,1) = [];
```

```
%% Tank 1
% This section plots Posterior, true and prior states estimated by
    the
% Kalman filter
figure;
plot(X_pri(1,1:50)', 'LineWidth', 1.5)
hold on;
plot(x_true(1,1:50),'LineStyle', '-')
hold on;
plot(X_post(1,1:50)', 'LineWidth', 1.5)
title('Tank 1')
legend('Priori Estimate', 'True Value', 'Posteriori Estimate')
hold off;
%% Tank 2
% This section plots Posterior, true and prior states estimated by
    the
% Kalman filter
figure;
plot(X_pri(2,1:50)', 'LineWidth', 1.5)
hold on;
plot(x_true(2,1:50), 'LineStyle', '-')
hold on;
plot(X_post(2,1:50)', 'LineWidth', 1.5)
title('Tank 2')
legend('Priori Estimate', 'True Value', 'Posteriori Estimate')
hold off;

%% Tank 3
% This section plots Posterior, true and prior states estimated by
    the
% Kalman filter
figure;
plot(X_pri(3,1:50)', 'LineWidth', 1.5)
hold on;
plot(x_true(3,1:50),'LineStyle', '-')
hold on;
plot(X_post(3,1:50)', 'LineWidth', 1.5)
title('Tank 3')
legend('Priori Estimate', 'True Value', 'Posteriori Estimate')
hold off;

%% Tank 4
% This section plots Posterior, true and prior states estimated by
    the
% Kalman filter
figure;
```

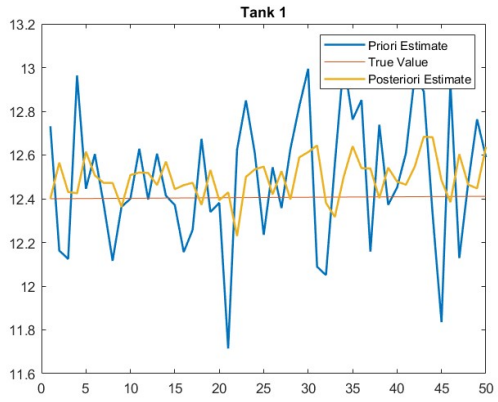
```
plot(X_pri(4,1:50)', 'LineWidth', 1.5)
hold on;
plot(x_true(4,1:50),'LineStyle', '-')
hold on;
plot(X_post(4,1:50)', 'LineWidth', 1.5)
title('Tank 4')
legend('Priori Estimate', 'True Value', 'Posteriori Estimate')
hold off;

%% Determinent of covariance
% This section plots the determinant of the posterior covariance and
    prior
% covariance for the first 10 iterations
figure()
plot(P_post_value(1:10), 'LineWidth', 2);
hold on;
plot(P_pri_value(2:10), 'LineWidth', 2);
xlabel('Iterations')
ylabel('Determinent of Covaraince matrices')
legend('Aposteriori Covariance', 'Apriori Covariance')
hold off;

%%
% This section plots the prior residues or Innovations for the first
    50
% iterations
figure()
plot(error_pri(:,1:50)', 'LineWidth', 2);
xlabel('Iterations')
ylabel('error in cm')
title('Apriori residue/Innovations')

%%
% This section plots the posterior residues for the first 50
   iterations
figure()
error_post(:,1) = [];
plot(error_post(:,1:50)', 'LineWidth', 2)
xlabel('Iterations')
ylabel('error in cm')
title('Aposteriori residue')

%%
%This section plots the Kalman gains of each state
figure;
K(:,:,1) = [];
```

```
for i = 1:size(K, 1) % Loop through each state
    subplot(size(K, 1), 1, i); % Create a subplot for each state
    plot(1:N, squeeze(K(i, :, :)), 'LineWidth', 2);
    title(['Kalman Gain for State ' num2str(i)]);
    xlabel('Iteration');
    ylabel('Kalman Gain');
    legend('Gain 1', 'Gain 2');
    grid on;
end
```

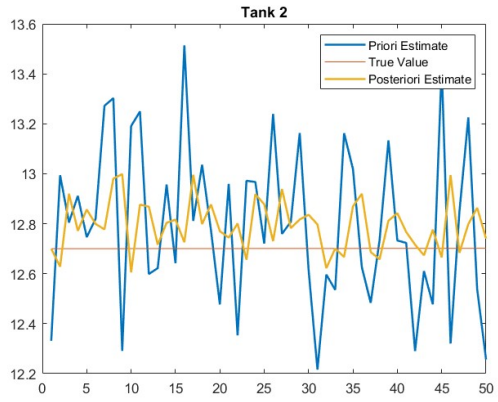
4.3 Results

With the following initial conditions,

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad Q = 0.1\ I_{4\times4} \qquad R = 0.01\ I_{2\times2}$$

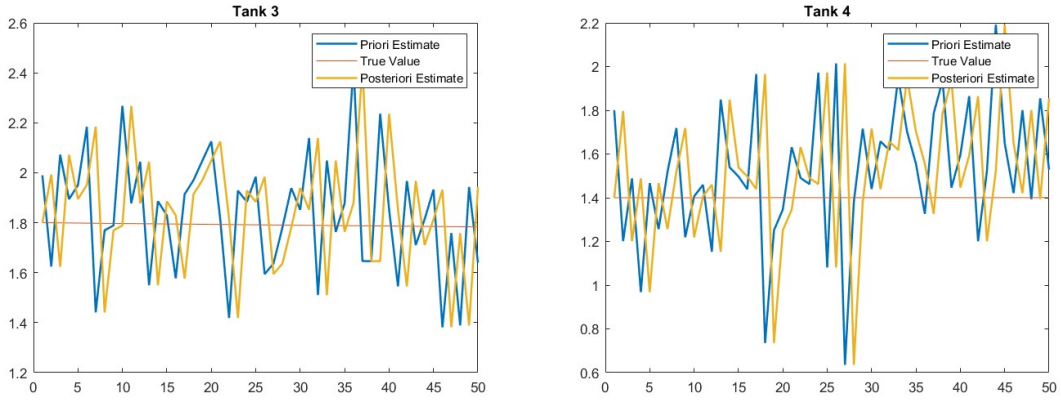


(a) Tank 1 estimate



(b) Tank 2 estimate

For the measured states, there is a correction/filtering that happens from the prior to posterior step



(a) Tank 3 estimate

(b) Tank 4 estimate

For the states that aren't measured, the prior and posterior estimates seems to be similar.

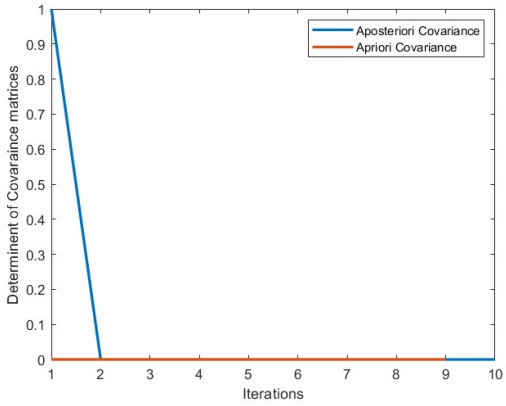
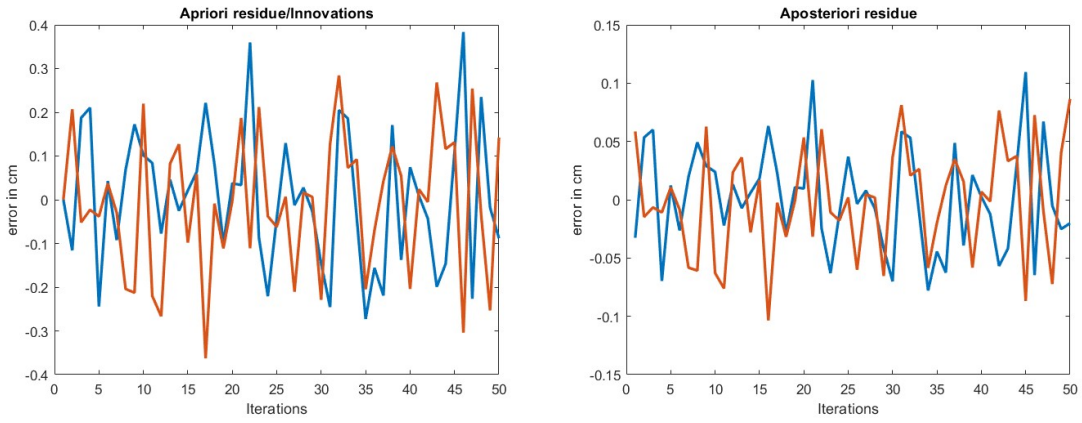


Figure 5: Posteriori and Priori covariance

The co-variance matrix of posterior and prior maintain a non-zero steady state. The innovations is larger compared to the posteriori residue which is characteristic of estimating and correcting properties of the Kalman filter.



(a) Innovations/ Priori residue

(b) Posteriori residue

For the states 3 and 4, the Kalman gain is negative to show there is presence of correction that is happening to the estimates 3 and 4, the gain values also show that states 1 and 2 has been more reliant in the estimation process.

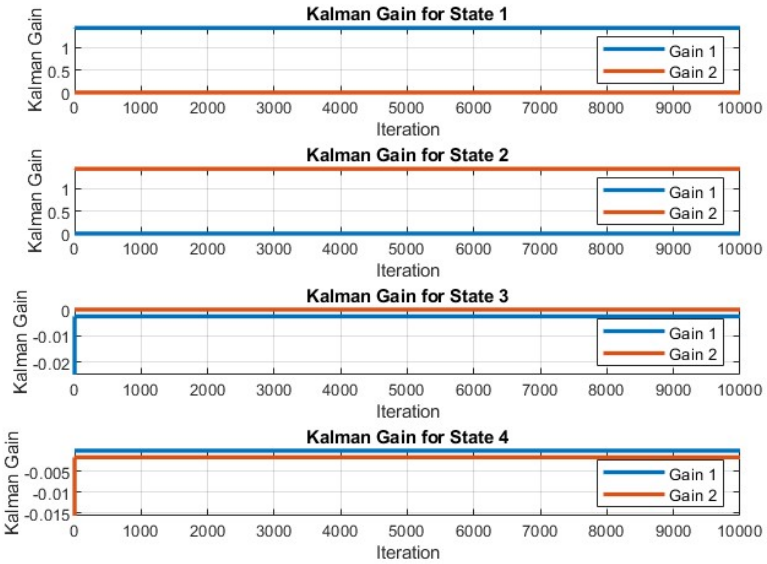


Figure 6: Kalman Gain of all states

5 Conclusion

The Kalman filter can give a faster estimate of the states with reduced noise effects. The states that are being measured shows a reduction in the prior estimate from the posterior estimate. However, the states that are not being measured have same posterior and prior estimates.