Mini Project
Modern Control Theory (CH5120)

Submitted by:
Name: Krishna Sah Teli
Roll No: GE22M018

Part 1: Kalman Filter
Introduction

Kalman filter is an algorithm that provides estimates of some unknown variables given the measurements observed over time. Kalman filters have been demonstrating its usefulness in various applications. It is a classic state estimation technique used in application areas such as signal processing and autonomous control of vehicles. It is now being used to solve problems in computer systems such as controlling the voltage and frequency of processors. It has relatively simple form and require small computational power.

Mathematical Model

Kalman filters are used to estimate states based on linear dynamical systems in state space format. The process model defines the evolution of the state from time k−1 to time k as:

$$X_k = A\,X_{k-1} + BU_{k-1} + W_{k-1}$$

where A is the state transition matrix applied to the previous state vector X_{k-1} , B is the control-input matrix applied to the control vector U_{k-1}, and W_{k-1} is the process noise vector that is assumed to be zero-mean Gaussian with the covariance Q i.e $W_{k-1} \sim N(0, Q)$.

The process model is paired with the measurement model that describes the relationship between the state and the measurement at the current time step k as:

$$Z_k = HX_k + V_k$$

where Z_k is the measurement vector, H is the measurement matrix, and V_k is the measurement noise vector that is assumed to be zero-mean Gaussian with the covariance R , i.e., $V_k \sim N(0, R)$. Note that sometimes the term “measurement” is called “observation”.

The role of the Kalman filter is to provide estimate of X_k at time k , given the initial estimate of X_0 , the series of measurement, $Z_1, Z_2, \ldots . Z_k$, and the information of the system described by A, B, H, Q, and V . Here, Q and R are usually used as tuning parameters that the user can adjust to get desired performance.

It has two stages i.e prediction and measurement updating.

1. Prediction

Predicted state estimate: $\hat{X}_k^{\,-} = A\,\hat{X}_{k-1}^{\,+} + BU_{k-1}$

Predicted error covariance: $P_k^{\,-} = AP_{k-1}^{\,+}A^T + Q$

2. Measurement update

Measurement estimation: $Z_{k_{estimate}} = H\,\hat{X}_k^{\,-}$

Measurement residual: $Z_{k_{error}} = Z_k - Z_{k_{estimate}}$

Kalman gain: $K_k = P_k^{\,-}H^T(R + HP_k^{\,-}H^T)^{-1}$

Update error covariance: $P_k^{\,+} = (I - K_kH)P_k^{\,-}$

$\hat{X}$ is an estimate of x The superscripts – and + denote predicted (prior) and updated (posterior) estimates, respectively

For our 4 tanks problem,
the following information have been provided:
The areas of the tanks

A1 = 28; % cm^2
A2 = 32; % cm^2
A3 = 28; % cm^2
A4 = 32; % cm^2
kc = 0.50; % VM/cm
g=981; %cm/ s^2

a1= 0.071;
a2= 0.057;
a3= 0.071;
a4= 0.057;

% Initial heights and voltage
h10 = 12.4;
h20 = 12.7;
h30 = 1.8;
h40 = 1.4;
v10= 3;
v20= 3;
Xo=[h10; h20; h30; h40];
Uo=[v10; v20];

%Minimum Phase Characteristics Values:
gamma1 = 0.7; % dimensionless
gamma2 = 0.6 ; % dimensionless
k1 = 3.33; % cm^3/Vs
k2 = 3.35; % cm^3/Vs
T1 = 62; % s
T2 = 90; % s
T3 = 23; % s
T4 = 30; % s

Ac = [-1/T1 0 A3/(A1*T3) 0; 0 -1/T2 0 A4/(A2*T4); 0 0 -1/T3 0; 0 0 0 -1/T4];
Bc = [(gamma1*k1)/A1 0 ; 0 (gamma2*k2)/A2; 0 ((1-gamma2)*k2)/A3; ((1-gamma1)*k1)/A4 0];
Cc = [kc 0 0 0; 0 kc 0 0];
Dc=[0];

I have set the initial value for X and P is [1; 1; 1; 1] and 100 * eye(4) respectively.

Goal: To estimate the height of water in the tanks.

Here I have explored the values of Q and R for multiple time in order to find out the suitable values where Kalman filter is converged and predict with more accuracy. That value of Q and R is

Q=3*eye(4);
R=10*eye(2);

Implementation

```
clear all
close all

measurementData=readmatrix('Measurements.csv');

% Data from the paper:
A1 = 28;        % cm2
A2 = 32;        % cm2
A3 = 28;        % cm2
A4 = 32;        % cm2
kc = 0.50;      % V/cm
g=981;

% Initial heights and voltage
h10 = 12.4;
h20 = 12.7;
h30 = 1.8;
h40 = 1.4;
v10= 3;
v20= 3;
Xo=[h10; h20; h30; h40];
Uo=[v10; v20];

% syms A1 A2 A3 A4 T1 T2 T3 T4 gamma1
gamma2 k1 k2 kc;
a1= 0.071;
a2= 0.057;
a3= 0.071;
a4= 0.057;

%Minimum Phase Characteristics Values:
gamma1 = 0.7;   % dimensionless
gamma2 = 0.6 ;  % dimensionless
k1 = 3.33;      % cm3/Vs
k2 = 3.35;      % cm3/Vs
T1 = 62;        % s
T2 = 90;        % s
T3 = 23;        % s
T4 = 30;

Ac = [-1/T1 0 A3/(A1*T3) 0; 0 -1/T2 0 A4/(A2*T4);
                    0 0 -1/T3 0; 0 0 0 -1/T4];
Bc = [(gamma1*k1)/A1 0 ; 0 (gamma2*k2)/A2;
                     0 ((1-gamma2)*k2)/A3;
                    ((1-gamma1)*k1)/A4 0];

%% Initial condition of X and P
X_post=[1; 1; 1; 1];
P_post=100 * eye(4);
Q=3*eye(4);
R=10*eye(2);

%% Storage definition and initialization
X_post_store= [];
X_prior_store= [];
P_post_store= [];
P_prior_store= [];
KG_store= [];
Innov_store= [];
Residual_store= [];
Norm_store= [];
v_store=[];
Covariance_post_store= [];
Covariance_prior_store= [];

%% Discretization
Ts=0.1;      %Sampling Time
csys= ss(Ac,Bc,Cc,Dc);
dsys= c2d(csys, Ts);
[A B H D] = ssdata(dsys);

%%
for i=1:10000
   % Computation of input u using forward
difference equation
   h1=measurementData(i,1);
   h2=measurementData(i,2);
   h3=measurementData(i,3);
   h4=measurementData(i,4);

   devH1= (measurementData(i+1,1)-
measurementData(i,1))/ Ts;
   devH2= (measurementData(i+1,2)-
measurementData(i,2))/ Ts;

   v1=(devH1 + (a1*sqrt(2*g*h1))/A1 -
(a3*sqrt(2*g*h3))/A1)*(A1/(gamma1*k1));
   v2=(devH2 + (a2*sqrt(2*g*h2))/A2 -
(a4*sqrt(2*g*h4))/A4)*(A2/(gamma2*k2));

   U=[v1; v2];
```

```
    %Prediction
    X_prior=A*(X_post - Xo) + B*(U-Uo) + Xo;
    P_prior=A*P_post*transpose(A) + Q;

    %Calculate Kalman Gain
    KG=P_prior*transpose(H)*(inv(H*P_prior*
                    transpose(H)+ R));

    %Measurement Data (True data)
    Z_true = H*transpose(measurementData(i,1:4));

    %Estimated Measurement
    Z_est=H*X_prior;

    %Error between estimated and true measurement
    Z_error=Z_true - Z_est;

    %Update with kalman gain
    X_post= X_prior + KG*Z_error;
    P_post=P_prior-KG*H*P_prior;

    norm1= norm(P_prior);
    norm2= norm(P_post);
    if abs(norm2-norm1)<= 5e-3
       disp("This is converged")
    end


    %Storage
    X_prior_store(i, 1:4, 1)= X_prior;
    X_post_store(i, 1:4, 1)= X_post;
    P_prior_store(i, 1:4, 1:4)= P_prior;
    P_post_store(i, 1:4, 1:4)= P_post;
    KG_store(i, 1:4, 1:2)= KG;
    Innov_store(i, 1:2, 1)= Z_error;
    residual= Z_true- H* X_post;
    Residual_store(i, 1:2, 1)= residual;
    Covariance_post_store(i)= trace(P_post);
    Covariance_prior_store(i)= trace(P_prior);
    v_store(i, 1:2)= [v1, v2];

end
```

```
%% plot for x prior
t=10000;
figure(1);
subplot(2,1,1)
plot(X_prior_store(1:1000,1), 'b-', 'LineWidth', 2);
hold on;
plot(X_prior_store(1:1000,2), 'r-', 'LineWidth', 1);
legend("h1", "h2")
xlabel("steps");
ylabel("height");


subplot(2,1,2)
plot(X_prior_store(1:1000,3), 'b.', 'LineWidth', 2);
hold on;
plot(X_prior_store(1:1000,4), 'r.', 'LineWidth', 2);
legend("h3", "h4")
xlabel("steps");
ylabel("height");
sgtitle("The X prior Plot")

%% plot for x post
figure(2);
subplot(2,1,1)
plot(X_post_store(1:1000,1), 'b-', 'LineWidth', 2);
hold on;
plot(X_post_store(1:1000,2), 'r:', 'LineWidth', 1);
legend("h1", "h2")
xlabel("steps");
ylabel("height");

subplot(2,1,2);
plot(X_post_store(1:1000, 3), 'b-', 'LineWidth', 2);
hold on;
plot(X_post_store(1:1000, 4), 'r:', 'LineWidth', 2);
hold on;
plot(X_post_store(1:1000, 4), 'r:', 'LineWidth', 2);
legend("h3", "h4")
xlabel("steps");
ylabel("height");

sgtitle("The X post Plot")
```

```
%% common plot for X proir and x post

figure(3)
subplot(2,1,1);
plot(X_post_store(1:1000,1:2), '*', 'LineWidth', 2);
hold on;
plot(X_prior_store(1:1000,1:2), '.', 'LineWidth', 2);
legend("X post h1", "X post h2", "X prior h1",
        "X prior h2");
xlabel("steps");
ylabel("height");
title("Plot of X post and X prior");

subplot(2,1,2);
plot(X_post_store(1:1000,3:4), '*', 'LineWidth', 2);
hold on;
plot(X_prior_store(1:1000,3:4), '.', 'LineWidth', 2);
legend("X post h3", "X post h4", "X prior h3",
        "X prior h4");
xlabel("steps");
ylabel("height");
title("Plot of X post and X prior");

%% plot of P post

figure(4);
subplot(3,1,1);
plot(Covariance_prior_store(1:1000), 'r.', 'LineWidth',
2);
xlabel("steps");
ylabel("trace");
title("Prior Covariance plot");

subplot(3,1,2);
plot(Covariance_post_store(1:1000), 'b.', 'LineWidth',
1);
xlabel("steps");
ylabel("trace");
title("Posterior Covariance plot");

subplot(3,1,3);
plot(Covariance_prior_store(1:1000), 'r*', 'LineWidth',
2);
hold on;
plot(Covariance_post_store(1:1000), 'b.',
'LineWidth', 1);
xlabel("steps");
ylabel("trace");
legend("Pprior", "Ppost");
title("The Covariance Plot");


%% Kalman gain plot

figure(5);
plot(KG_store(1:100,1,1), 'b-', 'LineWidth', 2);
xlabel("steps");
ylabel("Kalman gain");
title("The plot of 1st row 1st column element of
Kalman Gain matrix of each step");


%% Innovative and residual plot
figure(6);
subplot(2,1,1);
plot(Innov_store(1:100,1), 'b-', 'LineWidth', 2);
hold on;
plot(Residual_store(1:100,1), 'r:', 'LineWidth', 2);
legend("h1 Innov", "h2 Resid")
xlabel("steps");
ylabel("height diff");
subplot(2,1,2);
plot(Innov_store(1:100,2), 'b-', 'LineWidth', 2);
hold on;
plot(Residual_store(1:100,2), 'r:', 'LineWidth', 2);
legend("h1 Innov", "h2 Resid")
xlabel("steps");
ylabel("height diff");

sgtitle("The Innovative and Residual Plot");


%%  The End
```

Different plots

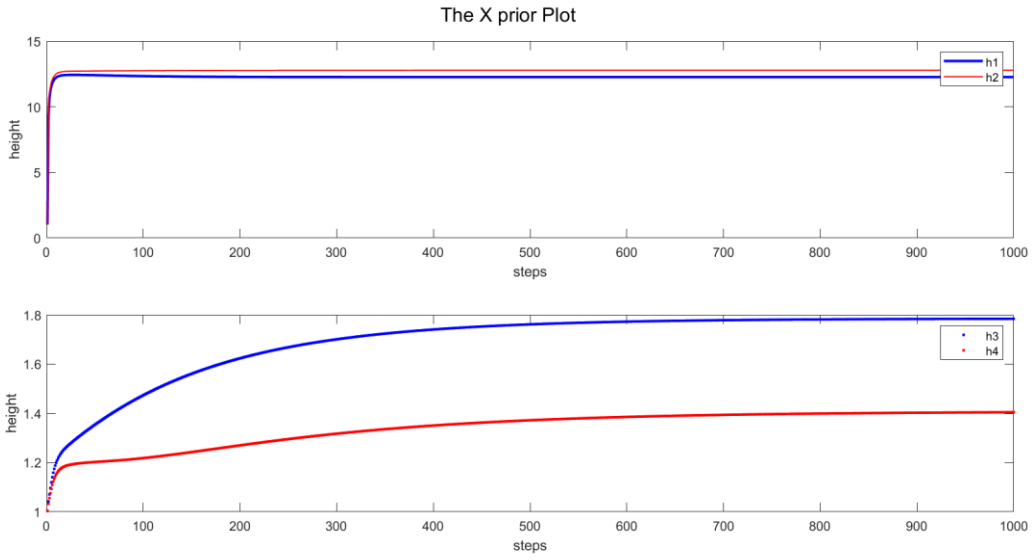

fig. 1 Plot of X_prior

It shows that the predicted value (i.e height of water in tanks) has converged to 12.8 cm for h1, to 12.28 cm for h2, to 1.78 cm for h3 and 1.4 cm for h3.

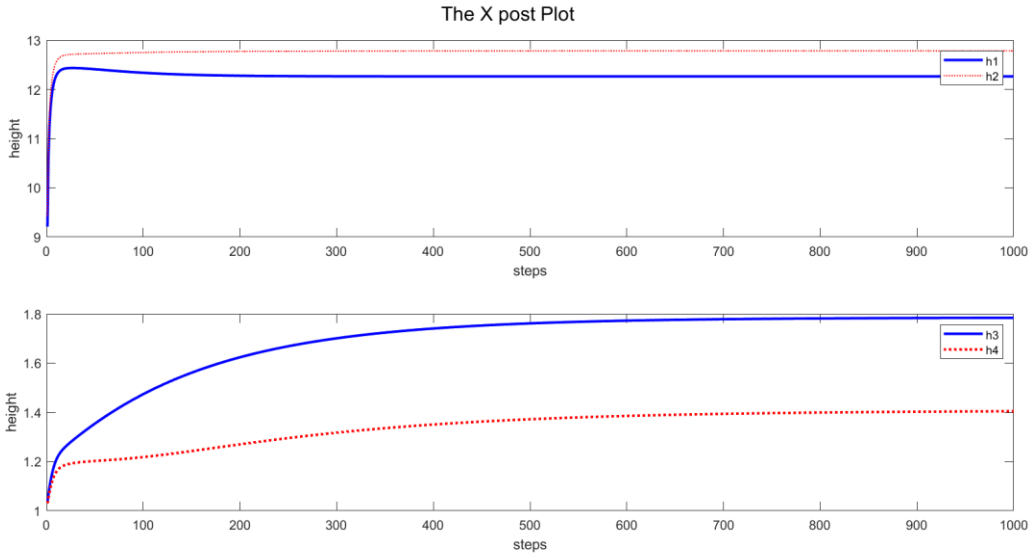

fig. 2 Plot of X post

It shows that the updated value for the heights of tanks have converged similar to that of predicted values. It signifies that the prediction is more accurate and closer to its true value. As we can see in fig.3, there is overlapping between the plots of predicted and updating values of height. Initially, they are little differed from each other but as the steps have proceeded ahead, predicted and updated values have becoming approximately same.

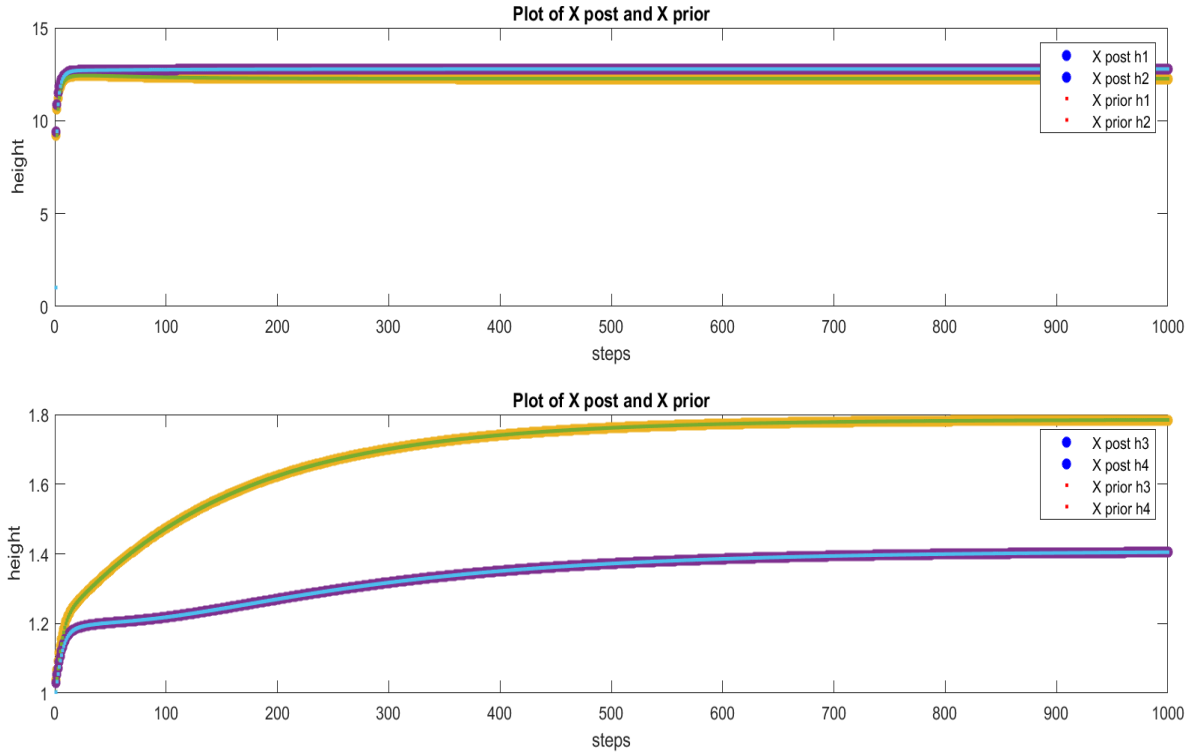

Fig. 3 Common plot for predicted and updating values of X.

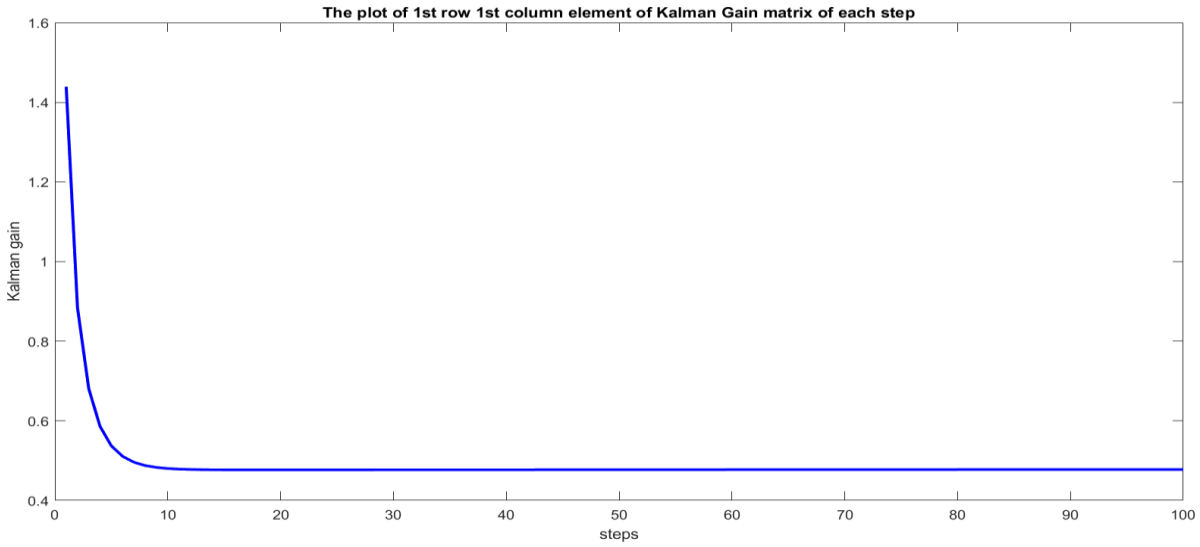

Fig.4 Kalman gain plot for different steps

As we can see in fig.4 (This plot is for K_store(1,1)), the Kalman gain has converged from 1.48 to 0.46. It signifies that the model has started depending on predicted values rather than true.

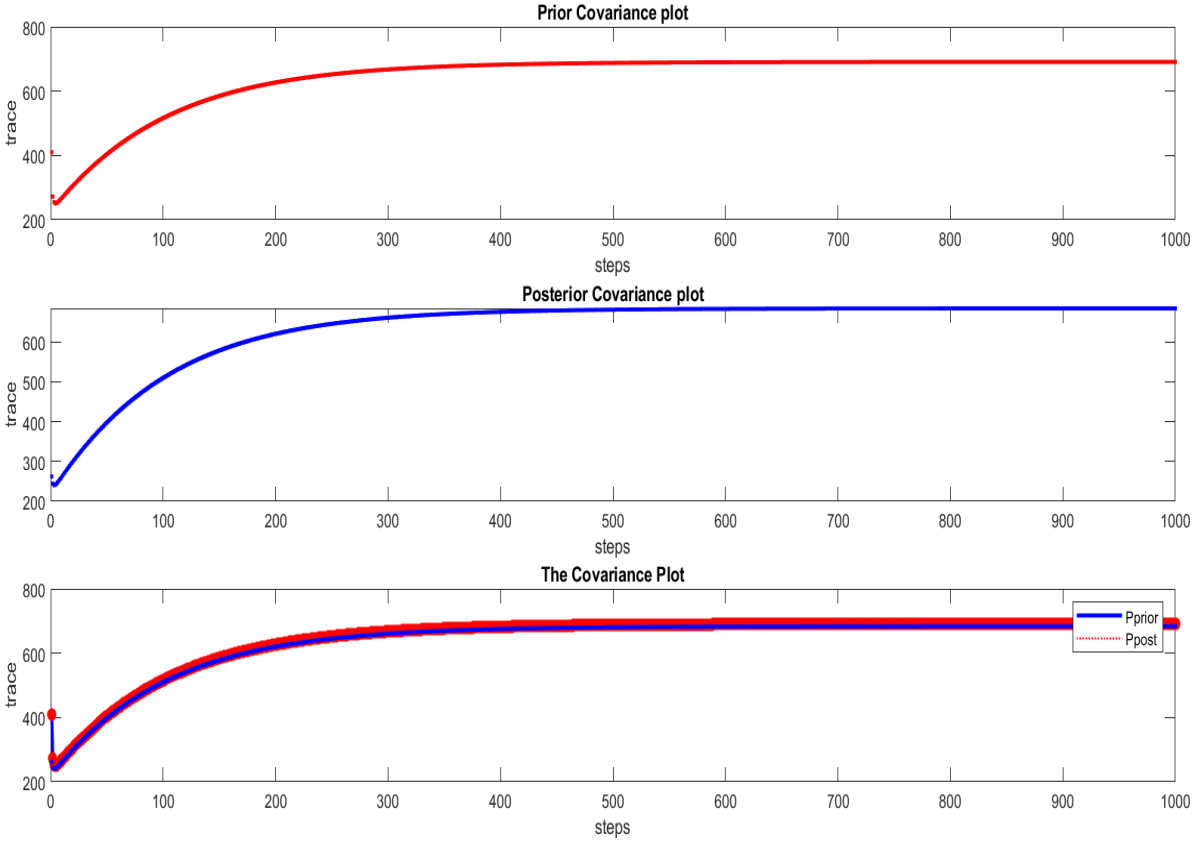

Fig. 5 Covariance plot

Fig.5 shows that the difference in covariance for post and prior has ended up with very insignificant value i.e less than 5x10^-3.

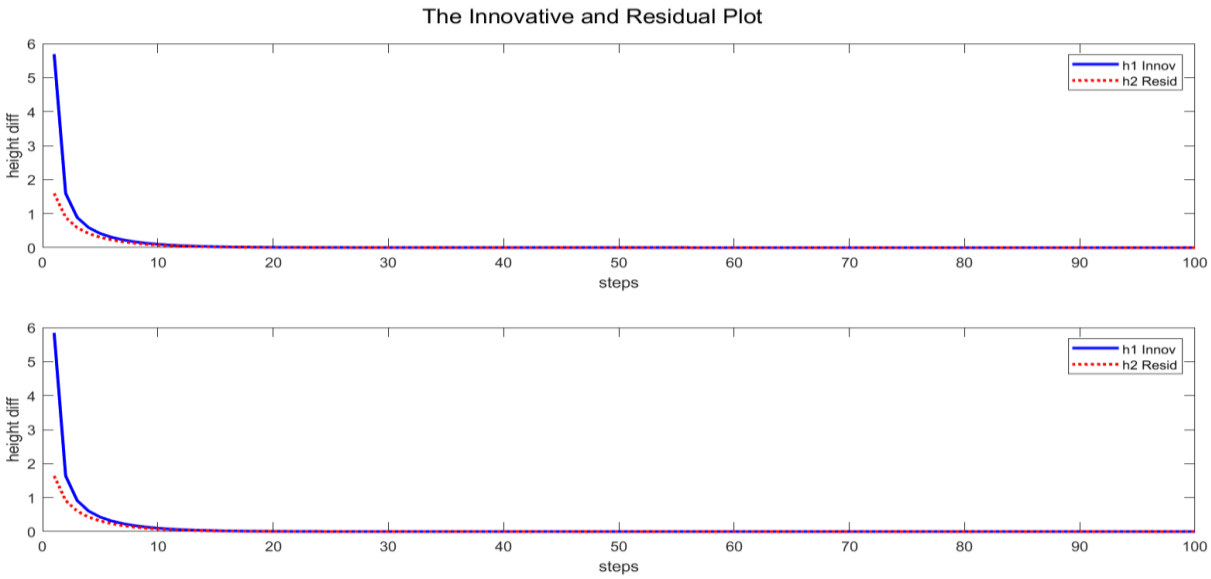

Fig. 6 Innovative and Residual plots

Here in fig.6, we can see that residual graph is always lower than innovative graph. They have converged to zero after some steps. Since their values have converged to zero, hence we can conclude that the accuracy Kalman with the value of Q as 3 and R as 10 is high. Hence this value of Q and R is suitable.

Part 2: Particle Filter

Introduction

Particle filter is the advance filter technique over Kalman filter. Kalman filter is applicable when noise is gaussian distribution and model is linear dynamics otherwise it does not work. Hence to overcome it, particle filter was introduced and is used to approximate the value rather than finding exact value as the randomness and noise is non-gaussian distribution and system is non-linear.

Particle filtering uses a set of particles (also called samples) to represent the posterior distribution of a stochastic process given the noisy and/or partial observations. The state-space model can be nonlinear and the initial state and noise distributions can take any form required. Particle filter techniques provide a well-established methodology for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions. However, these methods do not perform well when applied to very high-dimensional systems.

Particle filters update their prediction in an approximate (statistical) manner. The samples from the distribution are represented by a set of particles; each particle has a likelihood weight assigned to it that represents the probability of that particle being sampled from the probability density function. Weight disparity leading to weight collapse is a common issue encountered in these filtering algorithms, however, it can be mitigated by including a resampling step before the weights become uneven. Several adaptive resampling criteria can be used including the variance of the weights and the relative entropy with respect to the uniform distribution. In the resampling step, the particles with negligible weights are replaced by the new particles in the proximity of the particles with higher weights.

Mathematical Model

It is used to solve Hidden Markov Model and non-linear filtering problems. It has three stages:

1. Initialization (Prior step)

A random sample is generated containing noise.

For $i = 1, \ldots, N$, sample $x(0) \sim p\{x(0)\}$ & set $t = 1$

2. Sampling

It is an iterative process where prediction, likelihood and selection steps have involved.

2.1 Prediction Step

For $i = 1,\ldots,N$, sample $x(t,i) \sim p\{x(t)|x(t-1, i)\}$

2.2 Likelihood Step

For $i = 1,\ldots,N$, evaluate importance weights : $w(t,i) = p\{z(t)|x(t,i)$

Normalise the Importance Weights

2.3 Selection

It is also called update Step/Resampling Step

Resample with replacement, according to Importance Weights

Set $t = t + 1$ and return to Prediction Step

For 4 tank problem,

The random sample of dimension (500, 4) has been generated corresponding to the heights of water in 4 different tanks and it was converged to the approximate true values.

Implementation

```
clear all
% The prior information
data = readmatrix('Measurements.csv');
step = 100;
A1 = 28;        % cm2
A2 = 32;        % cm2
A3 = 28;        % cm2
A4 = 32;        % cm2

kc = 0.50;      % V/cm
T1 = 62;        % s
T2 = 90;        % s
T3 = 23;        % s
T4 = 30;        % s

a1 = 0.071;     % cm2
a2 = 0.057;     % cm2
a3 = 0.071;     % cm2
a4 = 0.057;     % cm2

g = 981;        % cm/s2
gamma1 = 0.7;   % dimensionless
gamma2 = 0.6 ;  % dimensionless
k1 = 3.33;      % cm3/Vs
k2 = 3.35;      % cm3/Vs
kc = 0.5;       % V/cm

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/A1 0];
C = [1/kc 0 0 0; 0 1/kc 0 0];
D= [0];
v1 = 3;
v2 = 3;
uk = [v1; v2];
P = 10^5*eye(4);

%% Discretization
Ts=0.1;      %Sampling Time
csys= ss(A,B,C,D);
dsys= c2d(csys, Ts);
[A B C D] = ssdata(dsys);

%%

N = 500;           % particle size
n = 4;             % states

% cholesky factor of P
L = chol(P);
% Adding covariance to each particle
x = (data(1,:)'*ones(1,N))'+ randn(N,n)*L;
x1_post = x(:,1);
x2_post = x(:,2);
x3_post = x(:,3);
x4_post = x(:,4);

%% Process Noise
Q = 3*eye(4);

% roughening of the prior
w = chol(Q)*randn(n,N);
w1 = w(1,:);
w2 = w(2,:);
w3 = w(3,:);
w4 = w(4,:);

%% Roughening the Prior
x1_post = x1_post + w1';
x2_post = x2_post + w2';
x3_post = x3_post + w3';
x4_post = x4_post + w4';

% Initialize Prediction values
xpri = zeros(N,n);
x1pri = xpri(:,1);
x2pri = xpri(:,2);
x3pri = xpri(:,3);
x4pri = xpri(:,4);

%% Prediction Step
for i = 1:N
    x1pri(i) = -a1/A1*sqrt(2*g*x1_post(i)) +
          a3/A1*sqrt(2*g*x3_post(i)) +
          (gamma1*k1*v1)/A1
          + w1(i);
    x2pri(i) = -a2/A2*sqrt(2*g*x2_post(i)) +
           a4/A2*sqrt(2*g*x4_post(i)) +
          (gamma2*k1*v2)/A2 + w2(i);
    x3pri(i) = -a3/A3*sqrt(2*g*x3_post(i)) +
           (1 - gamma2)*k2*v2/A3  +  w3(i);
    x4pri(i) = -a4/A4*sqrt(2*g*x4_post(i)) +
          (1 - gamma1)*k1*v1/A4  +  w4(i);
end
xpri  = abs(xpri);
x1pri = abs(x1pri);
```

```
x2pri = abs(x2pri);
x3pri = abs(x3pri);
x4pri = abs(x4pri);

% Importance Weights (Likelihood
Function)
z1 = data(1,1);
z2 = data(1,2);
z = [z1; z2];
z_true = z * ones(1,N);
R = 10 * eye(2);
z_est  =  C*xpri';
v = z_true - z_est;
for i = 1:N
   q(i) = exp(-0.5 * (v(:,i)' * inv(R) *
v(:,i)));
end

%% Normalizing the weights
 for i = 1:N
   wt(i) = q(i)/sum(q);
end

%% Resampling
M = length(wt);
Q = cumsum(wt);
indx = zeros(1, N);
T = linspace(0,1-1/N,N) + rand/N;
i = 1; j = 1;
while(i<=N && j<=M)
   while Q(j) < T(i)
      j = j + 1;
   end
   indx(i) = j;
    x1_post(i) = x1pri(j);
    x2_post(i) = x2pri(j);
    x3_post(i) = x3pri(j);
    x4_post(i) = x4pri(j);
   i = i + 1;
end

x1_pri_cum = cumsum(x1pri);
x2_pri_cum = cumsum(x2pri);
x3_pri_cum = cumsum(x3pri);
x4_pri_cum = cumsum(x4pri);

x1_cum = cumsum(x1_post);
x2_cum = cumsum(x2_post);
x3_cum = cumsum(x3_post);
x4_cum = cumsum(x4_post);

x_post = [x1_post x2_post x3_post x4_post];

%% Finding Centroid
x1_est = mean(x1_post);
x2_est = mean(x2_post);
x3_est = mean(x3_post);
x4_est = mean(x4_post);

%% plotting
figure(1);
subplot(2,1,1);
plot(x_post(:,1), 'LineWidth', 1);
xlabel("steps");
ylabel("Post height");

subplot(2,1,2);
plot(x1pri, "LineWidth",1);
plot(x_post(:,2), 'LineWidth', 1);
xlabel("steps");
ylabel("prior height");

sgtitle("Plot of h1 cloud");

figure(2);
subplot(2,1,1);
plot(x_post(:,2), 'LineWidth', 1);
xlabel("steps");
ylabel("Post height");

subplot(2,1,2);
plot(x2pri, "LineWidth",1);
plot(x_post(:,2), 'LineWidth', 1);
xlabel("steps");
ylabel("prior height");

sgtitle("Plot of h2 cloud");

figure(3);
subplot(2,1,1);
plot(x_post(:,3), 'LineWidth', 1);
xlabel("steps");
ylabel("Post height");

subplot(2,1,2);
plot(x3pri, "LineWidth",1);
plot(x_post(:,2), 'LineWidth', 1);
xlabel("steps");
ylabel("prior height");
sgtitle("Plot of h3 cloud");
```

```
figure(4);
subplot(2,1,1);
plot(x_post(:,4), 'LineWidth', 1);
xlabel("steps");
ylabel("Post height");

subplot(2,1,2);
plot(x4pri, "LineWidth",1);
plot(x_post(:,2), 'LineWidth', 1);
xlabel("steps");
ylabel("prior height");

sgtitle("Plot of h4 cloud");


%%
```

Different plots

By setting the Q value as 3 and R value as 10, different graphs have been plotted out.

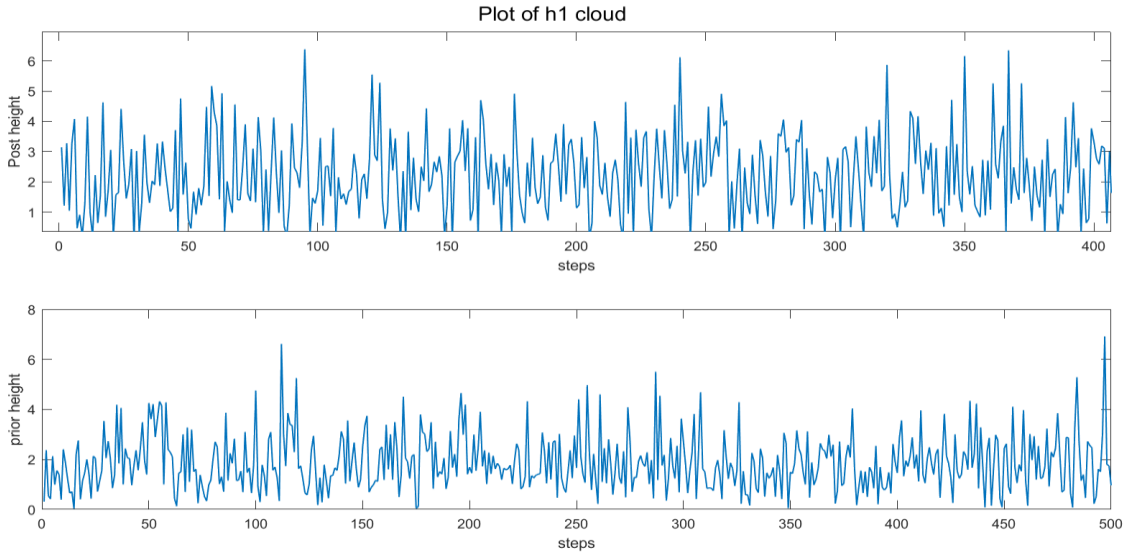

Fig. 7 Plot of posterior and prior value of h1 cloud

As we can see in the fig7, fig.8, fig.9 and fig.10,

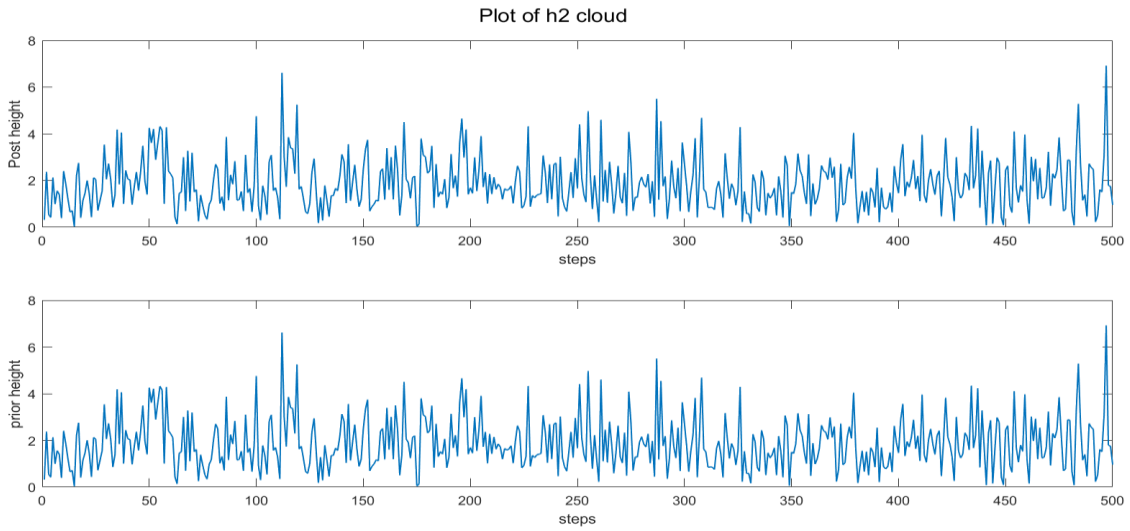

Fig. 8 Plot of posterior and prior value of h2 cloud

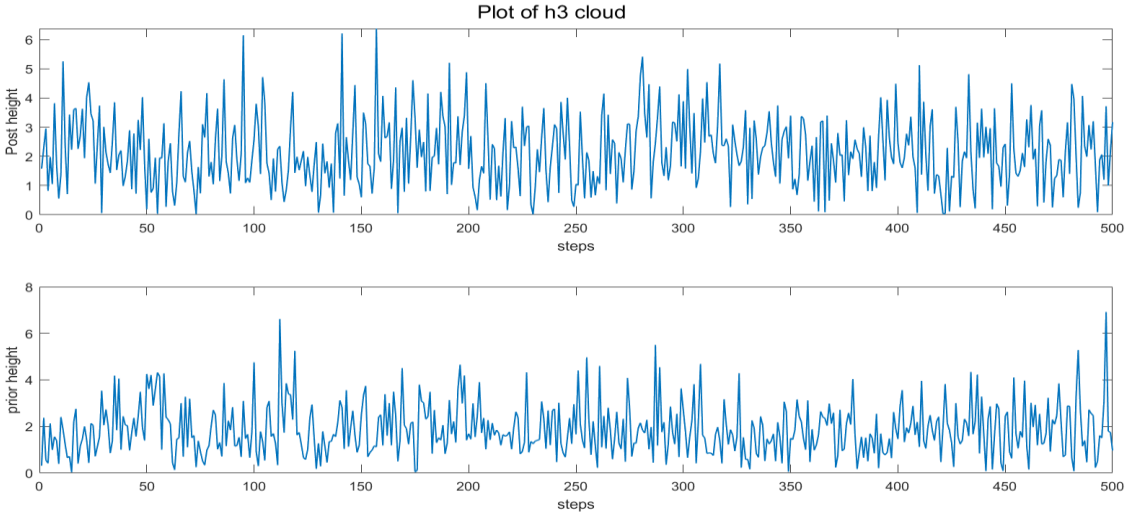

Fig. 9 Plot of posterior and prior value of h3

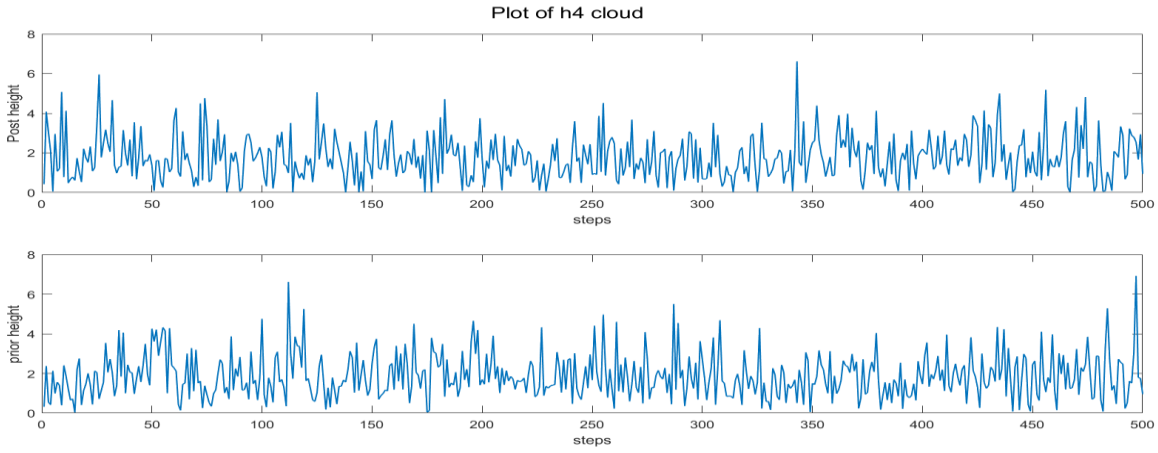

Fig. 10. Plot of posterior and prior value of h4

Part 3: Result Discussion

The Kalman filter with the value of Q as 3*eye(4) and R as 10*eye(4) worked well and predicted values are approximately equal to the true value of measurement. The difference in true measurement of height and predicted height has converged to zero as shown in fig.6. Similarly, the tolerance of the model has been checked and it has been found out to be less than 5* 10^-3. Kalman gain has also converged towards zero as we can see in fig.4 which implies that model has relied on prediction than measurement values as it can predict accurate values as in fig.1 and fig.2. Similarly, posterior values are superior over prior value at the beginning. However, with the passage of steps, both become approximate same as seen in fig.3.

Particle filter is started with random sample of dimension (500, 4) representing the random heights of water in the tanks. After applying the particle filter, it has converged but is not satisfactory.