CH5120

Mini Project-B

Arunachalam Ramesh

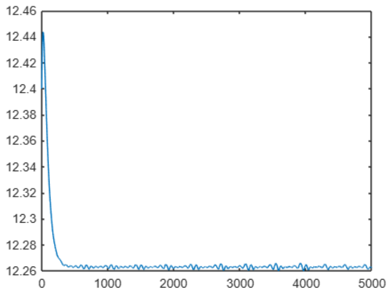
EP22B004

Q1.: Solve the set of non-linear equations of the four-tank system, as given below, using forward difference method or by using ODE-45 of *Matlab*(or equivalent in *Python*,etc.), for generating a typical data set of 10,000. This data set is to be used for comparing the performance of the Kalman Filter & Particle Filter.

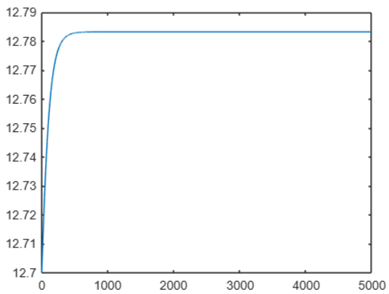
For T_final = 5000s with T_s = 0.5, dataset with 10000 points for 4 tank heights were produced in MATLAB using ode45.

```
% Initialization of all the parameters of the four tank system
A1 = 28; %(cm^2)
A2 = 32;
A3 = 28;
A4 = 32;
a1 = 0.071;
a3 = 0.071; %(cm^2)
a2 = 0.057;
a4 = 0.057;
kc = 0.5; % (V/cm)
g = 981; %(cm/s^2)
gamma1 = 0.7; gamma2 = 0.6; % constants, determined from valve position
k1 = 3.33; k2 = 3.35; %[cm^3/Vs]
kc = 0.5; % [V/cm]
v1 = 3; v2 = 3; % (V)
h0 = [12.4; 12.7; 1.8; 1.4];
% Set the initial conditions and the simulation time span
t0 = 0;
tf = 5000;
ts = tf/10000;
% Call the ODE45 solver
[t, h] = ode45(@four_tank_ode, [t0: ts: tf-ts], h0');
function dh_dt = four_tank_ode(t, h)
 A1 = 28; %(cm^2)
 A2 = 32;
 A3 = 28;
 A4 = 32;
 a1 = 0.071;
 a3 = 0.071; %(cm^2)
 a2 = 0.057;
 a4 = 0.057;
 kc = 0.5; % (V/cm)
 g = 981; %(cm/s^2)
 gamma1 = 0.7; gamma2 = 0.6; % constants, determined from valve position
 k1 = 3.33; k2 = 3.35; %[cm^3/Vs]
 kc = 0.5; % [V/cm]
```

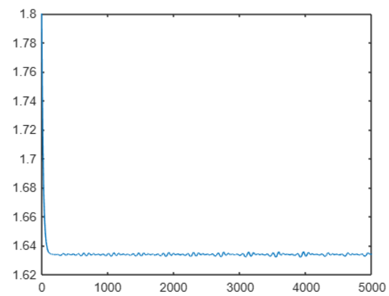
```
 v1 = 3; v2 = 3; % (V)
 h0 = [12.4; 12.7; 1.8; 1.4];

 % Calculate the time derivatives of the state variables
 dh_dt = [
 (-a1 * sqrt(2 * g * h(1)) / A1) + (a3 * sqrt(2 * g * h(3)) / A1) + (gamma1
* k1 * v1 / A1);
 (-a2 * sqrt(2 * g * h(2)) / A2) + (a4 * sqrt(2 * g * h(4)) / A2) + (gamma2
* k2 * v2 / A2);
 (-a3 * sqrt(2 * g * h(3)) / A3) + ((1 - gamma2) * k2 * v2 / A3);
 (-a4 * sqrt(2 * g * h(4)) / A4) + ((1 - gamma1) * k1 * v1 / A4);
 ];
end
figure
plot(t,h(:,1));
```



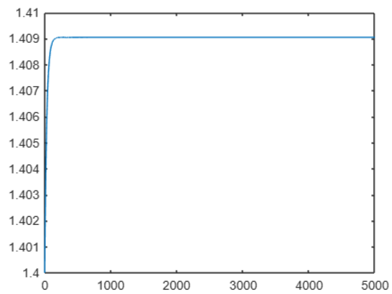

```
figure
plot(t,h(:,2));
```




```
figure
plot(t,h(:,3));
```

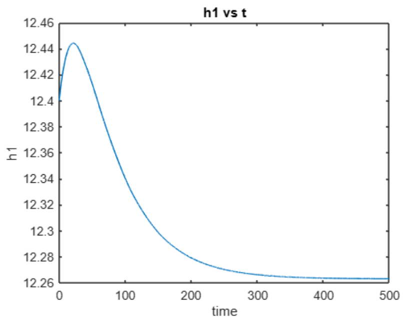


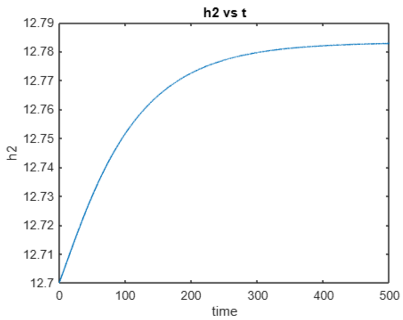
```
figure
plot(t,h(:,4));
```

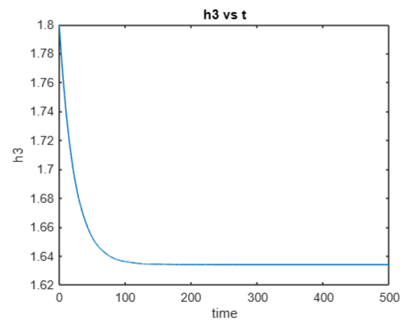


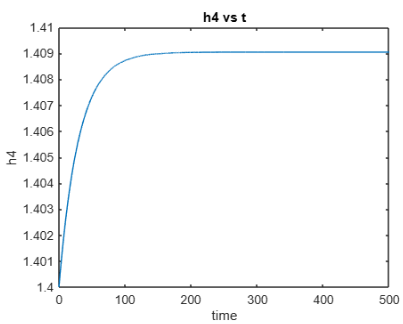
Q4.Formulate a ***Particle Filter*** (use the algorithm, given in ***Slide # 6***) for the above four tank problem for estimating ***h-3 & h-4*** *(not measured)* and obtaining the filtered values for ***h-1 & h-2*** *(measured).Verify the resulting estimate by comparing the same with the generated data set (from Q-1).*

```
A1 = 28; %(cm^2)
A2 = 32;
A3 = 28;
A4 = 32;
a1 = 0.071;
a3 = 0.071; %(cm^2)
a2 = 0.057;
a4 = 0.057;
g = 981; %(cm/s^2)
gamma1 = 0.7; gamma2 = 0.6; % constants, determined from valve position
k1 = 3.33; k2 = 3.35; %[cm^3/Vs]
kc = 0.5; % [V/cm]
v1 = 3; v2 = 3; % (V)
h0 = [12.4; 12.7; 1.8; 1.4];
% Set the initial conditions and the simulation time span
t0 = 0;
tf = 500;
ts = tf/10000;
% Call the ODE45 solver
[t, h] = ode45(@four_tank_ode1, [t0: ts: tf-ts], h0');
for i = [1:4]
 figure;
 plot(t,h(:, i));
 title(['h', num2str(i), ' vs t']);
 xlabel('time');
 ylabel(['h', num2str(i)]);
 % Call the function to save the plot
 save("simulation_h"+i);
end
```





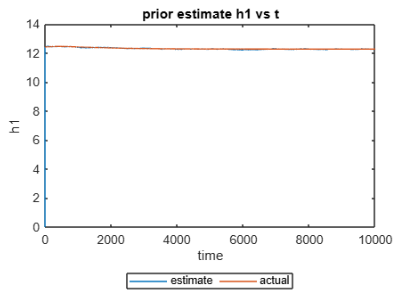


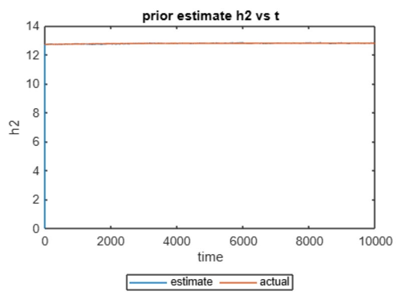


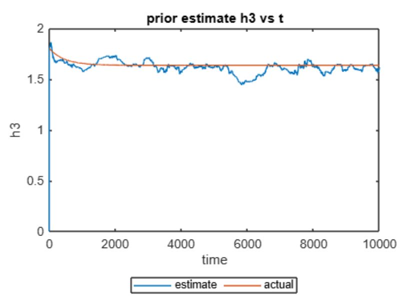

```
C = [
 kc 0 0 0;
 0 kc 0 0;
];
Ts = 0.1;
% Defining the disturbances
P0 = 10^-2 * eye(4);
Q = 10^-5 * eye(4);
R = 2 * 10^-2 * eye(2);
% Initialization of states
x0 = h0;
u = [
 3;
```

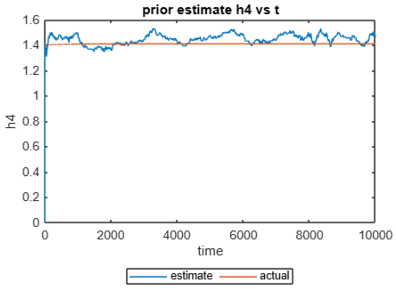
```
 3;
];
N = 400;
L = chol(P0);
rng(32);
x0 = x0 * ones(1, N) + L * randn(4, N);
wr = chol(Q) * randn(4, N);
% Prior roughening
for i = [1:4]
 x_post(i, :) = x0(i, :) + wr(i, :);
end
x_pri_mean=zeros(4, 10000);
x_post_mean=zeros(4, 10000);
j = 2;
while j <= 10000
 w = chol(Q) * randn(4, N);
 x_dt = zeros(4, N);
 for i = [1:N]
 x_dt(1, i) = -a1/A1 * sqrt(2 * g * (x_post(1, i))) + a3/A1 * sqrt(2 * g *
(x_post(3, i))) + gamma1 * k1 * v1/A1;
 x_dt(2, i) = -a2/A2 * sqrt(2 * g * (x_post(2, i))) + a4/A2 * sqrt(2 * g *
(x_post(4, i))) + gamma2 * k1 * v2/A2;
 x_dt(3, i) = -a3/A3 * sqrt(2 * g * (x_post(3, i))) + (1 - gamma2) * k2 *
v2/A3;
 x_dt(3, i) = -a4/A4 * sqrt(2 * g * (x_post(4, i))) + (1 - gamma1) * k1 *
v1/A4;
 end
 x_pri = x_post + Ts * x_dt + w;
 % Importance weights
 z_tru = C * h(j, :)' *ones(1, N);
 z_est = C * x_pri;
 v = z_tru - z_est;
 q = zeros(1, N);
 wt = zeros(1, N);
 for i = [1:N]
 q(i) = exp(-0.5 * (v(:, i)' * inv(R) * v(:, i)));
 end
 % Normalized weights
 for i = [1:N]
 wt(i) = q(i)/sum(q);
 end
 cum_wt = cumsum(wt);
 for i = [1:N]
 r = rand;
 ind = find(r <= cum_wt, 1);
 x_post(:, i) = x_pri(:, ind);
 end
 x_pri_mean(:, j)=mean(x_pri, 2);
 x_post_mean(:, j)=mean(x_post, 2);
 j = j+1;
```

```
end
x_pri_mean = abs(x_pri_mean);
x_post_mean = abs(x_post_mean);
t_span = [1: 10000];
for i=1:4
 figure;
 plot(t_span,x_pri_mean(i, 1:10000), t_span, h(1:10000, i))
 legend('estimate', 'actual', 'Location', 'southoutside', 'Orientation',
'horizontal')
 title(['prior estimate h', num2str(i), ' vs t'])
 xlabel('time')
 ylabel(['h', num2str(i)])
 save("Prior state estimation" + num2str(i));
end
```

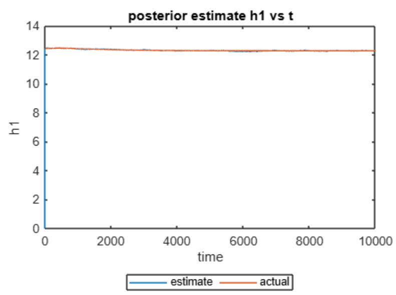


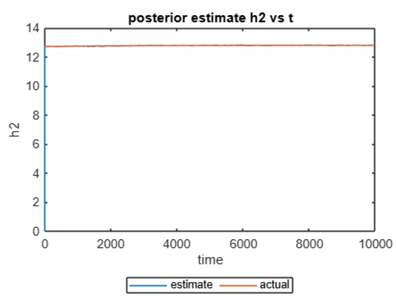


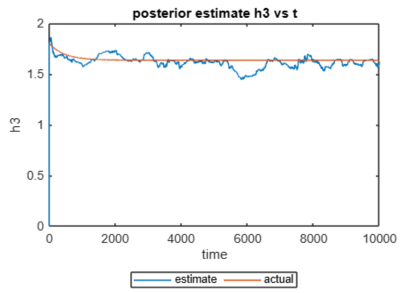


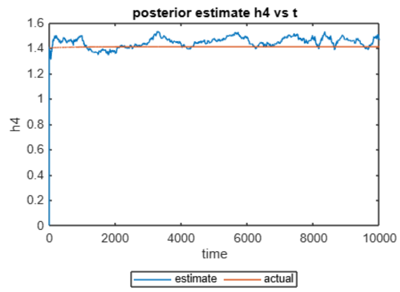



```
rms = zeros(4, 1);
for i=1:4
 figure;
 plot(t_span,x_post_mean(i, 1:10000), t_span, h(1:10000, i))
 legend('estimate', 'actual', 'Location', 'southoutside', 'Orientation',
'horizontal')
 title(['posterior estimate h', num2str(i), ' vs t'])
 xlabel('time')
 ylabel(['h', num2str(i)])
 save("Postrior state estimation" + num2str(i));

 % Call the function to calculate rmse
 rms(i) = rmse(h(1:10000, i)', x_post_mean(i,1:10000));
end
```










```
% Function to save the plot
function save(title)
 validFileName = sprintf(title);
 fileExtension = 'png';
 fileName = sprintf('%s.%s', validFileName, fileExtension);
 saveas(gcf, fileName);
end
% Function to calculate rmse
function rms = rmse(a, b)
 rms = sqrt(mean((a-b).^2));
end
function dh_dt = four_tank_ode1(t, h)
A1 = 28; %(cm^2)
 A2 = 32;
 A3 = 28;
 A4 = 32;
 a1 = 0.071;
 a3 = 0.071; %(cm^2)
 a2 = 0.057;
 a4 = 0.057;
 g = 981; %(cm/s^2)
 gamma1 = 0.7; gamma2 = 0.6; % constants, determined from valve position
 k1 = 3.33; k2 = 3.35; %[cm^3/Vs]
 kc = 0.5; % [V/cm]
 v1 = 3; v2 = 3; % (V)
 h0 = [12.4; 12.7; 1.8; 1.4];

 % Calculate the time derivatives of the state variables
 dh_dt = [
 (-a1 * sqrt(2 * g * h(1)) / A1) + (a3 * sqrt(2 * g * h(3)) / A1) + (gamma1
* k1 * v1 / A1);
```

```
 (-a2 * sqrt(2 * g * h(2)) / A2) + (a4 * sqrt(2 * g * h(4)) / A2) + (gamma2
* k2 * v2 / A2);
 (-a3 * sqrt(2 * g * h(3)) / A3) + ((1 - gamma2) * k2 * v2 / A3);
 (-a4 * sqrt(2 * g * h(4)) / A4) + ((1 - gamma1) * k1 * v1 / A4);
 ];
end
```