PART B

Solution 4

A particle filter was formulated, values of h1, h2, h3 and h4 after 100 time steps and with 1000 particles are as follows:
h1=11.8552
h2=10.5393
h3=1.9553
h4=1.8636
These values are close to the values obtained by the ode45 solver in question 1 given below.
h1=12.27cm
h2=12.78cm
h3=1.64cm
h4=1.409cm
Therefore, it validates our algorithm for particle filter.

Matlab code is as follows:

```
% Initialization of all the parameters of the four-tank system
clc; clear all; close all;
% Tank and system parameters
A1 = 28; A2 = 32; A3 = 28; A4 = 32;  % Cross-sectional areas of the tanks (cm^2)
a1 = 0.071; a2 = 0.057; a3 = 0.071; a4 = 0.057;  % Outlet areas (cm^2)
kc = 0.5;  % Sensor gain (V/cm)
g = 981;  % Gravitational acceleration (cm/s^2)
% Valve parameters
gamma1 = 0.7; gamma2 = 0.6;  % Constants determined by valve positions
% Pump parameters
k1 = 3.33; k2 = 3.35;  % Pump constants (cm^3/V/s)
v1 = 3; v2 = 3;  % Voltage input to the pumps (V)
% Initial conditions
h0 = [12.4; 12.7; 1.8; 1.4];  % Initial water levels in the tanks (cm)
x0 = h0;
x_post = x0;
n = 4;
N = 1000;  % Number of particles
T = 100;   % Total number of time steps
dt = 1;   % Time step size
% Process and measurement noise covariance matrices
Q = [80, 30, 20, 10;  % Tank 1
    30, 80, 15, 5;   % Tank 2
    0, 15, 10, 0;    % Tank 3
    10, 5, 25, 8];    % Tank 4
```

```
R =  30* eye(2);
% Initial particles setup
P = 10^5 * eye(4);
L = chol(P);
particles = (x0 * ones(1, N))' + randn(N, n) * L;
% Separate the particles for each state variable
x1_post = particles(:, 1);
x2_post = particles(:, 2);
x3_post = particles(:, 3);
x4_post = particles(:, 4);
% Time loop for particle filter operations from t = 1 to T
for t = 1:T
   % Add process noise (for prior roughening)
   w = chol(Q) * randn(n, N);
   w1 = w(1, :);
   w2 = w(2, :);
   w3 = w(3, :);
   w4 = w(4, :);

   % Prediction Step
   for i = 1:N
       % Prediction for Tank 1
       x1_pri(i) = -a1/A1 * sqrt(2 * g * x1_post(i)) + a3/A1 * sqrt(2 * g *
x3_post(i)) ...
                   + (gamma1 * k1 * v1) / A1 + w1(i);

       % Prediction for Tank 2
       x2_pri(i) = -a2/A2 * sqrt(2 * g * x2_post(i)) + a4/A2 * sqrt(2 * g *
x4_post(i)) ...
                   + (gamma2 * k2 * v2) / A2 + w2(i);

       % Prediction for Tank 3
       x3_pri(i) = -a3/A3 * sqrt(2 * g * x3_post(i)) + (1 - gamma2) * k2 * v2 / A3
+ w3(i);

       % Prediction for Tank 4
       x4_pri(i) = -a4/A4 * sqrt(2 * g * x4_post(i)) + (1 - gamma1) * k1 * v1 / A4
+ w4(i);
   end

   % Ensure non-negative water levels by taking absolute values
   x1_pri = abs(x1_pri);
   x2_pri = abs(x2_pri);
   x3_pri = abs(x3_pri);
   x4_pri = abs(x4_pri);
   % True measurements (simulated) for Tanks 1 and 2 (e.g., can vary over time)
   z1 = 12.4;  %  measured value for Tank 1
```

```
    z2 = 12.7;  %  measured value for Tank 2
    z = [z1; z2];
    z_true = z * ones(1, N);  %  measurements for all particles
    % Measurement matrix
    C = [kc 0 0 0; 0 kc 0 0];
    % Predicted measurement based on the predicted state
    x_pri = [x1_pri; x2_pri; x3_pri; x4_pri];
    z_est = C * x_pri;
    % Measurement residual (difference between true and estimated measurements)
    v = z_true - z_est;
    % Importance Weights (Likelihood Function)
    for i = 1:N
        q(i) = exp(-0.5 * (v(:,i)' / R * v(:,i)));  % Calculate likelihood weight
for each particle
    end

    % Normalize the importance weights
    q_sum = sum(q);  % Sum of all importance weights
    if q_sum > 0
        wt = q / q_sum;  % Normalize the weights
    else
        wt = ones(1, N) / N;  % If sum is zero, assign equal weights
    end
    % Resampling
    M = length(wt);
    Q_cumsum = cumsum(wt);
    indx = zeros(1, N);
    T_values = linspace(0, 1 - 1 / N, N) + rand(1) / N;
    i = 1; j = 1;
    while (i <= N && j <= M)
        while (Q_cumsum(j) < T_values(i))
            j = j + 1;
        end
        indx(i) = j;
        x1_post(i) = x1_pri(indx(i));
        x2_post(i) = x2_pri(indx(i));
        x3_post(i) = x3_pri(indx(i));
        x4_post(i) = x4_pri(indx(i));
        i = i + 1;
    end
end
% The state estimates for h1, h2, h3, and h4 at each time step T can be obtained by
% averaging the particles
h1_estimate = mean(x1_post);
h2_estimate = mean(x2_post);
h3_estimate = mean(x3_post);
h4_estimate = mean(x4_post);
```

```
disp(['Estimated h1 at time T: ', num2str(h1_estimate)]);
disp(['Estimated h2 at time T: ', num2str(h2_estimate)]);
disp(['Estimated h3 at time T: ', num2str(h3_estimate)]);
disp(['Estimated h4 at time T: ', num2str(h4_estimate)]);
```