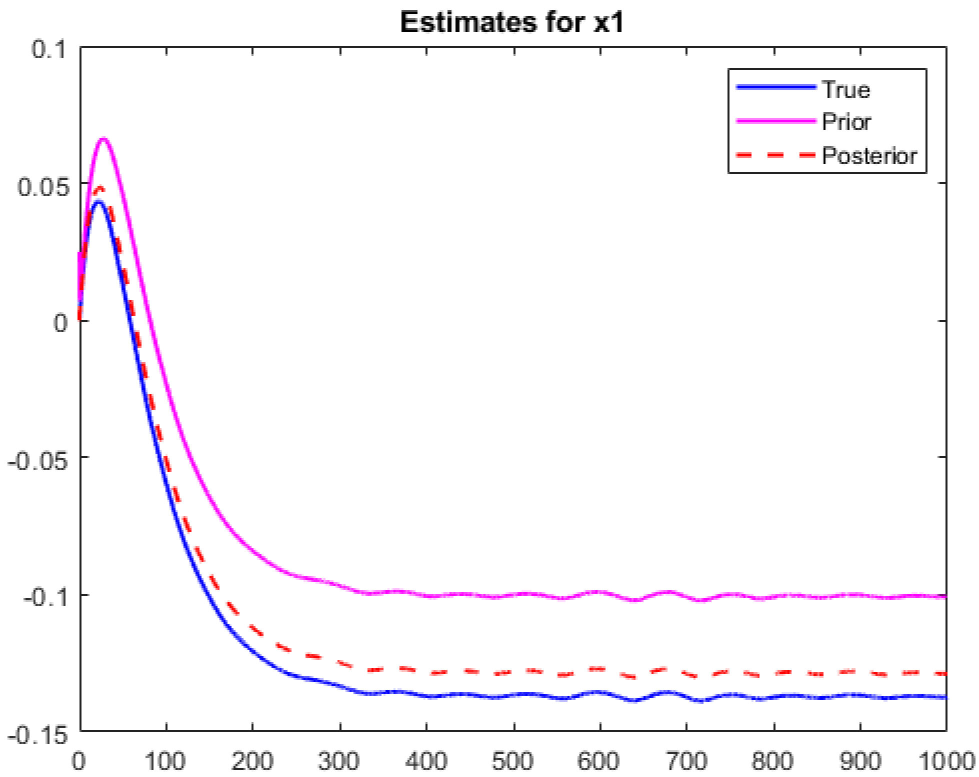
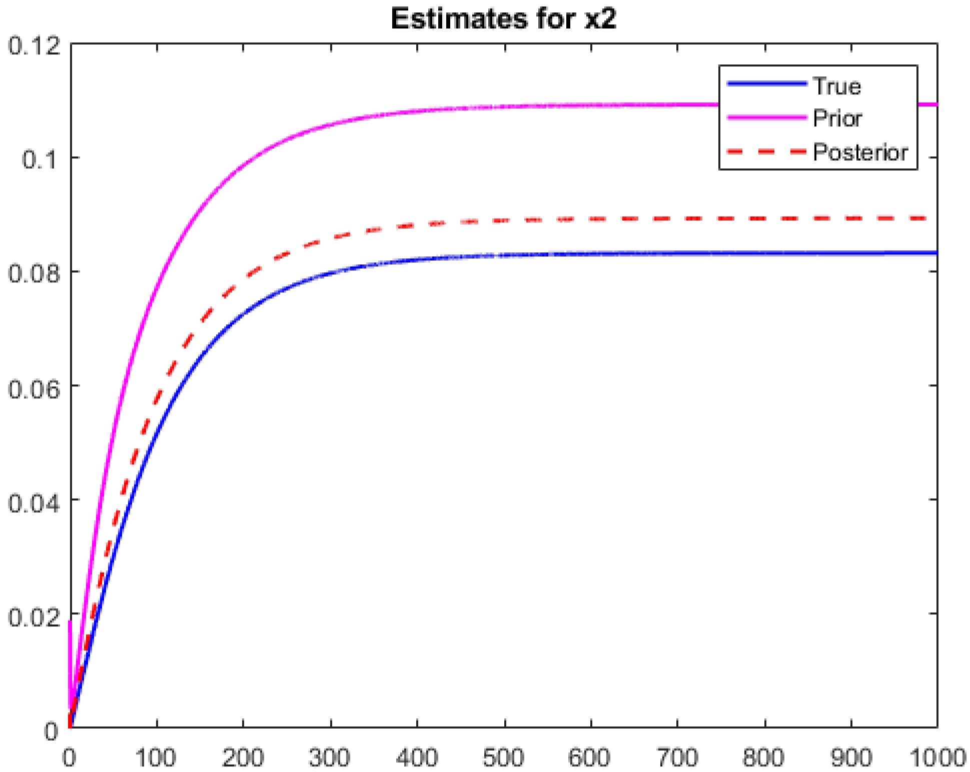
```
% Mini Project Part A
% Arunachalam Ramesh EP22B004
% Q1, Q2, Q3 addressed here

% System parameters given
a1 = 0.071; a2 = 0.057; a3 = 0.071; a4 = 0.057;
A1 = 28; A2 = 32; A3 = 28; A4 = 32;
k1 = 3.33; k2 = 3.35;
g = 981;
gamma1 = 0.7; gamma2 = 0.6;
kc = 0.5;

% Time and initial conditions
dt = 0.1;
h_initial = [12.4; 12.7; 1.8; 1.4];

% Setting up the state space
C = [kc, 0, 0, 0; 0, kc, 0, 0];
A = [
    (-a1/A1) * sqrt(g / (2 * h_initial(1))), 0, (a3/A1) * sqrt(g / (2 *
 h_initial(3))), 0;
    0, (-a2/A2) * sqrt(g / (2 * h_initial(2))), 0, (a4/A2) * sqrt(g / (2 *
 h_initial(4)));
    0, 0, (-a3/A3) * sqrt(g / (2 * h_initial(3))), 0;
    0, 0, 0, (-a4/A4) * sqrt(g / (2 * h_initial(4)))
];
B = [
    gamma1 * k1 / A1, 0;
    0, gamma2 * k2 / A2;
    0, (1 - gamma2) * k2 / A3;
    (1 - gamma1) * k1 / A4, 0
];

Ad = expm(A * dt);
Bd = A \ (Ad - eye(4)) * B;

% Initiation for Kalman Filtering
x_true = [0; 0; 0; 0];
x_hat = x_true;
tspan = [0, 1000];
t_ode = tspan(1):dt:tspan(2);

%After Several rounds of tuning, this set of P,Q,R was obtained. This is
%purely on basis of the outputs the code gave whenever the values where
%changed only. And I have two interpretations of this project. One is to
%get a tuned filter to give closer outputs as the one got from ODE, and the
%other is to build an ideal precise Kalman Filter.
%Precise Kalman Filter
Q = diag([0.01, 0.01, 0.01, 0.01]);
R = diag([0.001, 0.001]);
P = diag([1000, 1000, 1000, 1000]); % Initial error covariance
%Kalman Filter Designed to give close to values of ODE.
```

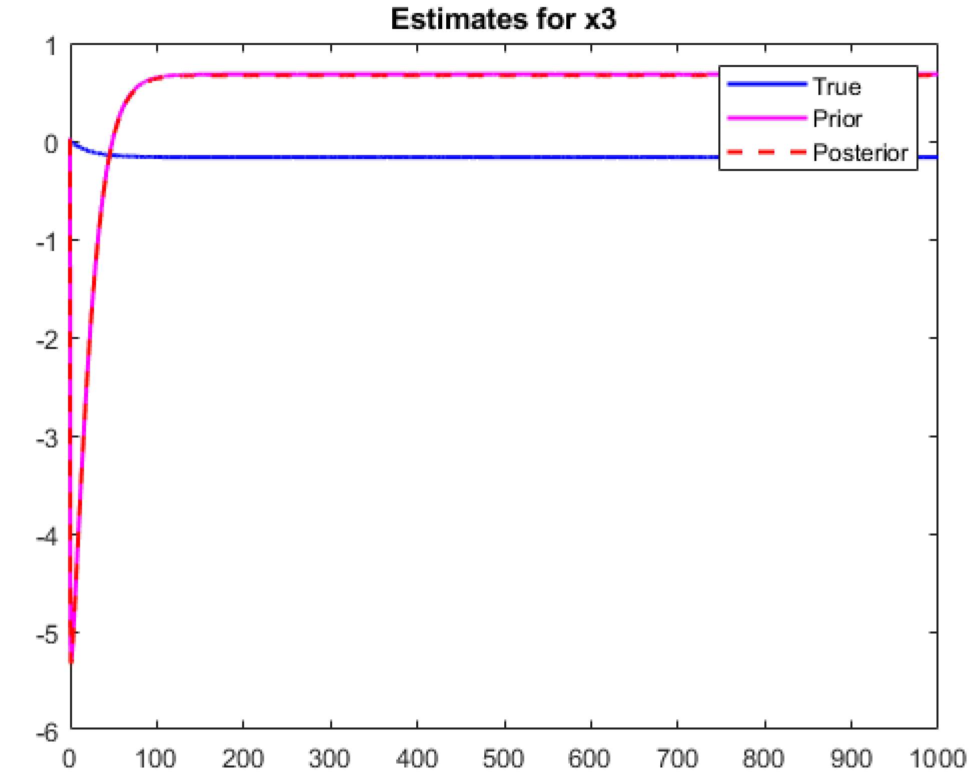
```
 Q1 = diag([10, 10, 16.1, 12.5]);
 R1 = diag([0.5, 0.5]);
 P1 = diag([150, 150, 1500, 1500]);
 %However Lets see the Precise kalman Filter here in this report since
 %there has been no expilicit testing about my tuning. If necessary please
 %use the second set of paramaeters given here as well.

% Pre-allocate arrays
x_hat_array = zeros(length(t_ode), 4);
P_array = zeros(length(t_ode), 4, 4);
K_array = zeros(length(t_ode), 4, 2);
residuals = zeros(length(t_ode), 2);
prior_estimates = zeros(length(t_ode), 4);
innovations_prior = zeros(length(t_ode), 2);
innovations_post = zeros(length(t_ode), 2);
P_prior_array = zeros(length(t_ode), 4, 4);

% Generate the ODE dataset
[~, h_ode] = ode45(@tank_system, t_ode, h_initial);
z_true = h_ode - h_initial';
y_true = C * z_true';

% Kalman Filter loop
for k = 1:length(t_ode)
    x_hat_prior = Ad * x_hat + Bd * [3; 3];
    P_prior = Ad * P * Ad' + Q;
    prior_estimates(k, :) = x_hat_prior';
    P_prior_array(k, :, :) = P_prior;
    y_measured = y_true(:, k);

    % Kalman gain
    K = P_prior * C' / (C * P_prior * C' + R);
    K_array(k, :, :) = K;

    % Update step
    residual = y_measured - C * x_hat_prior;
    x_hat = x_hat_prior + K * residual;
    P = (eye(4) - K * C) * P_prior;

    % Store results
    x_hat_array(k, :) = x_hat';
    residuals(k, :) = residual';
    innovations_prior(k, :) = residual;
    innovations_post(k, :) = y_measured - C * x_hat;
    P_array(k, :, :) = P;

    % Add snapshots for publishing
    if mod(k, 100) == 0 % Adjust snapshot frequency as needed
        snapnow;
    end
end

% Plotting results
```

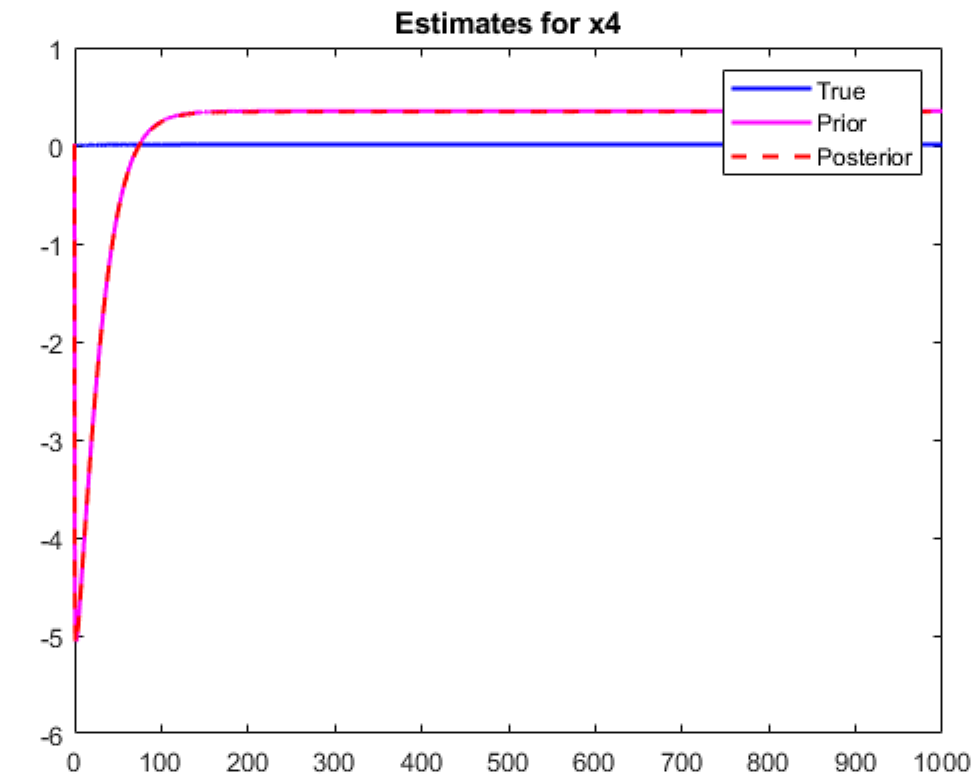
```
plot_results(t_ode, z_true, prior_estimates, x_hat_array, P_prior_array,
 P_array, innovations_prior, innovations_post, K_array);

% Function to handle plotting
function plot_results(t_ode, z_true, prior_estimates, x_hat_array,
 P_prior_array, P_array, innovations_prior, innovations_post, K_array)
    for i = 1:4
        figure;
        plot(t_ode, z_true(:, i), 'b', 'LineWidth', 1.5); hold on;
        plot(t_ode, prior_estimates(:, i), 'm', 'LineWidth', 1.5);
        plot(t_ode, x_hat_array(:, i), 'r--', 'LineWidth', 1.5);
        title(['Estimates for x', num2str(i)]);
        legend('True', 'Prior', 'Posterior');
        snapnow;
    end

    for i = 1:4
        figure;
        plot(t_ode, squeeze(P_prior_array(:, i, i)), 'b', 'LineWidth', 1.5);
 hold on;
        plot(t_ode, squeeze(P_array(:, i, i)), 'r--', 'LineWidth', 1.5);
        title(['Covariance for x', num2str(i)]);
        snapnow;
    end

    figure;
    plot(t_ode, innovations_prior(:, 1), 'g', 'LineWidth', 1.5); hold on;
    plot(t_ode, innovations_post(:, 1), 'c--', 'LineWidth', 1.5);
    plot(t_ode, innovations_prior(:, 2), 'm', 'LineWidth', 1.5);
    plot(t_ode, innovations_post(:, 2), 'k--', 'LineWidth', 1.5);
    title('Residuals for Measurements');
    snapnow;

    for i = 1:2
        figure;
        plot(t_ode, squeeze(K_array(:, :, i)), 'LineWidth', 1.5);
        title(['Kalman Gain for Measurement ', num2str(i)]);
        snapnow;
    end
end

% Define the ODE function (move this to the end of the script)
function dhdt = tank_system(~, h)
    a1 = 0.071; a2 = 0.057; a3 = 0.071; a4 = 0.057;
    A1 = 28; A2 = 32; A3 = 28; A4 = 32;
    k1 = 3.33; k2 = 3.35;
    g = 981;
    gamma1 = 0.7; gamma2 = 0.6;
    v1 = 3; v2 = 3;
    dh1 = -a1/A1 * sqrt(2 * g * h(1)) + a3/A1 * sqrt(2 * g * h(3)) + gamma1 *
 k1/A1 * v1;
    dh2 = -a2/A2 * sqrt(2 * g * h(2)) + a4/A2 * sqrt(2 * g * h(4)) + gamma2 *
 k2/A2 * v2;
    dh3 = -a3/A3 * sqrt(2 * g * h(3)) + (1 - gamma2) * k2/A3 * v2;
```

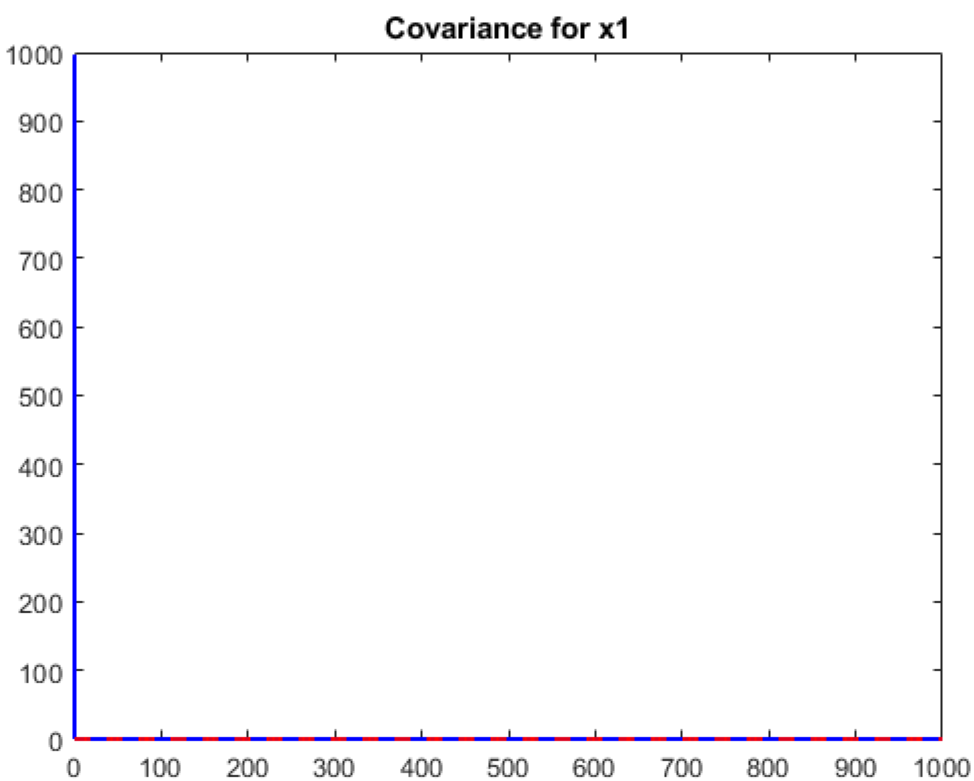
```
    dh4 = -a4/A4 * sqrt(2 * g * h(4)) + (1 - gamma1) * k1/A4 * v1;
    dhdt = [dh1; dh2; dh3; dh4];
end
```

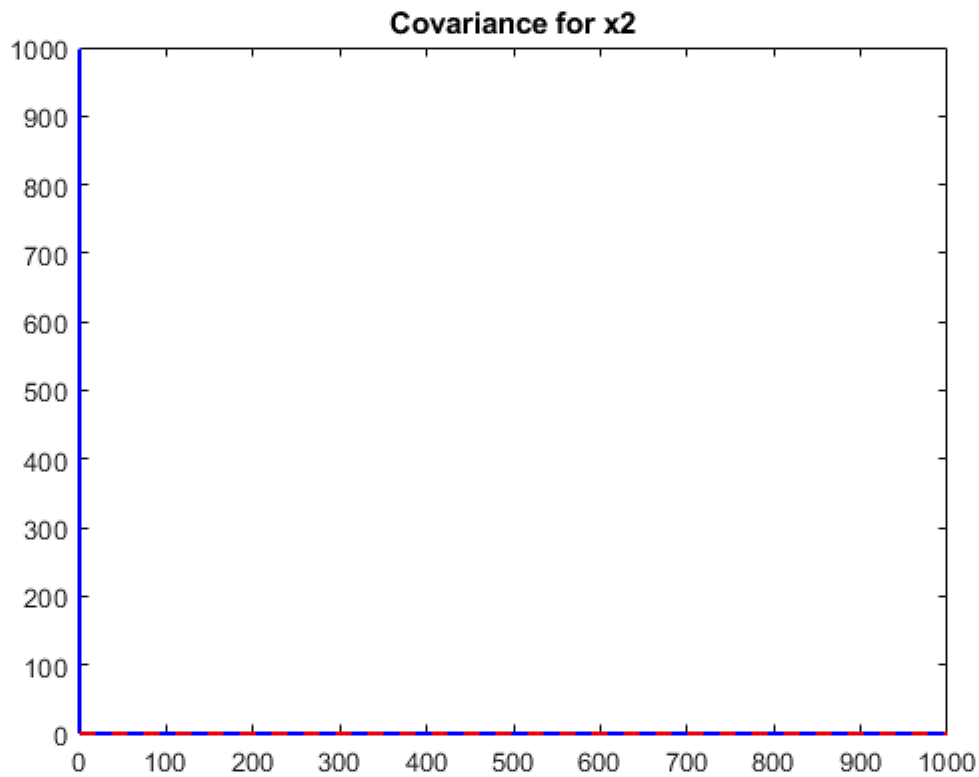


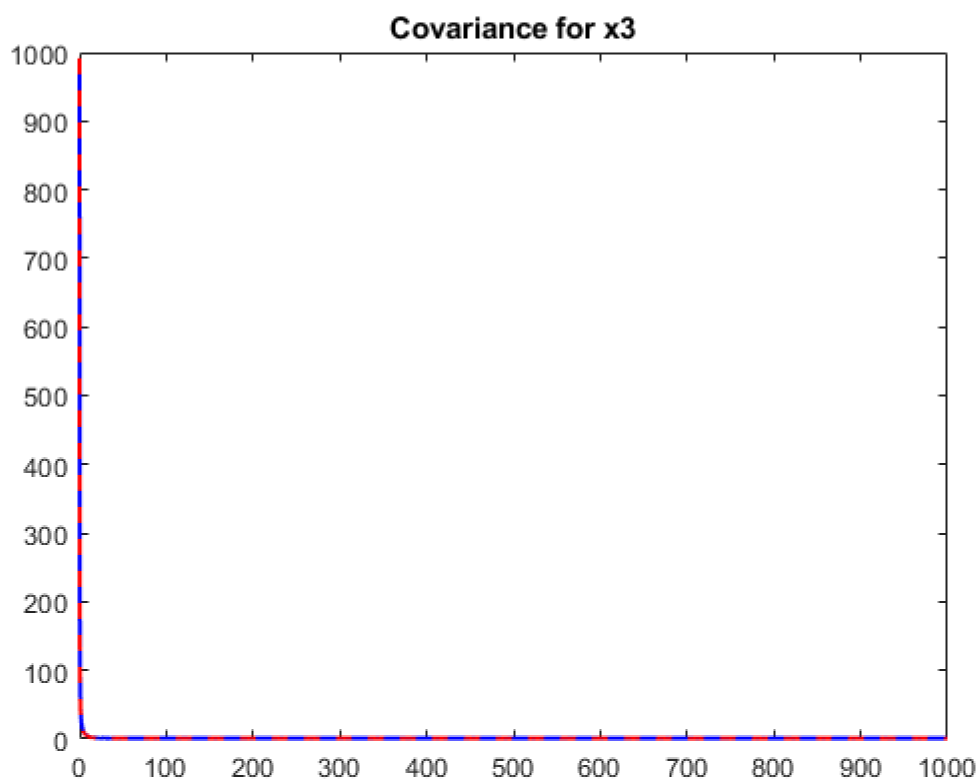

Estimates for x2
True
Prior
Posterior
0.12
0.1
0.08
0.06
0.04
0.02
0
0
100
200
300
400
500
600
700
800
900
1000

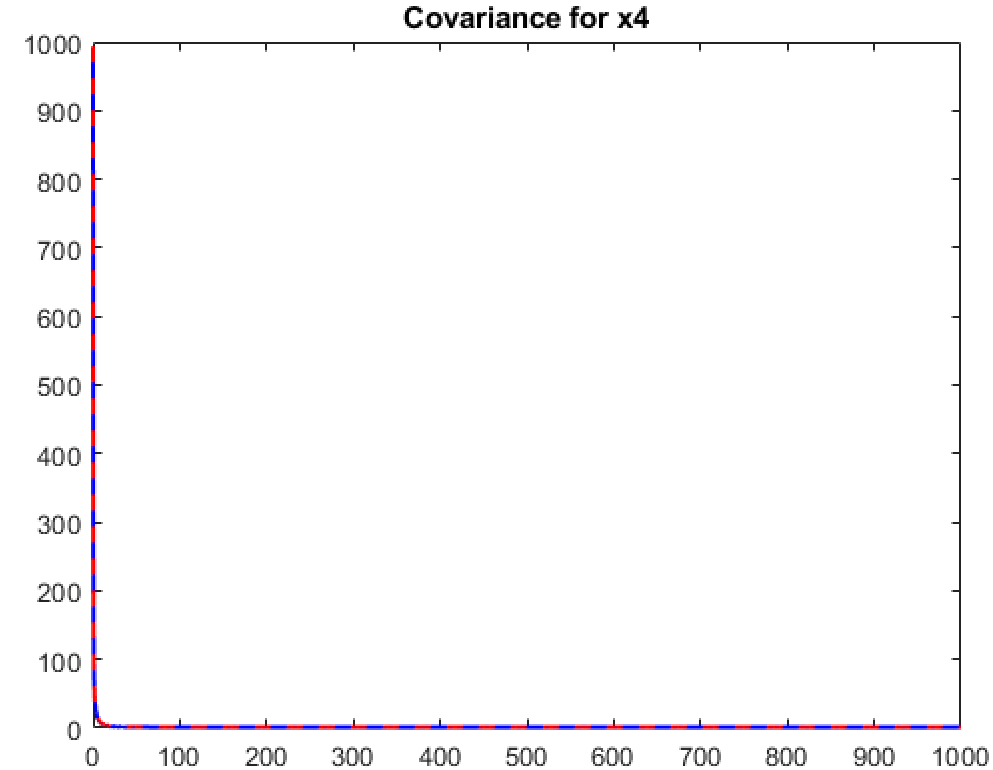

Estimates for x3
True
Prior
Posterior
1
0
-1
-2
-3
-4
-5
-6
0
100
200
300
400
500
600
700
800
900
1000

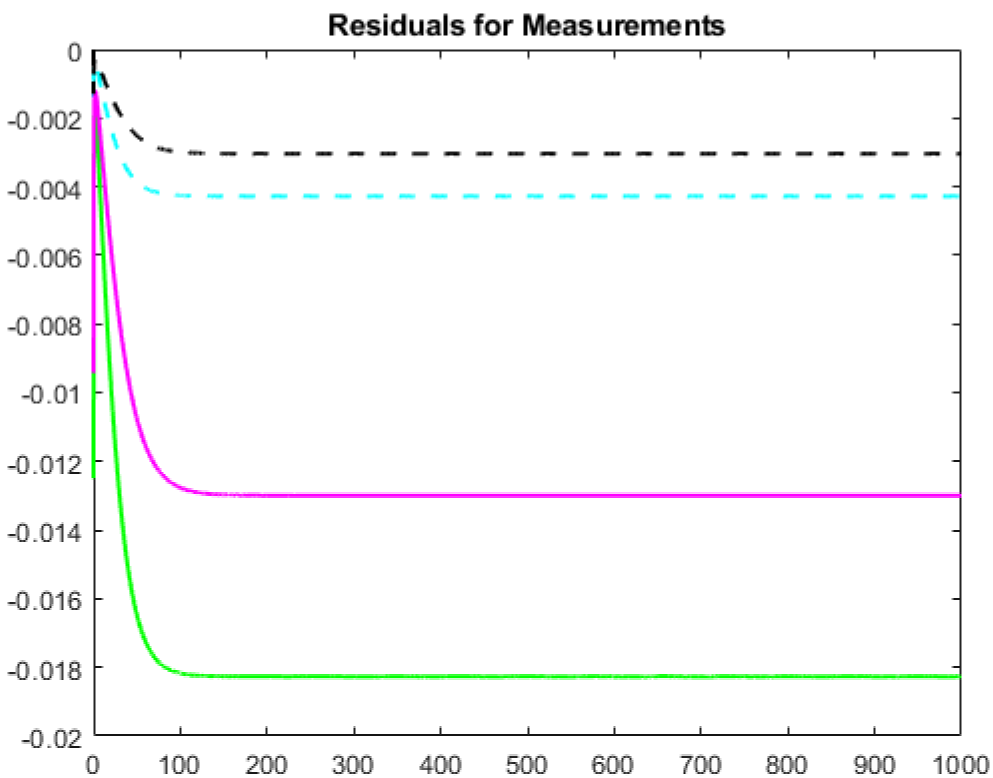

Estimates for x4
True
Prior
Posterior

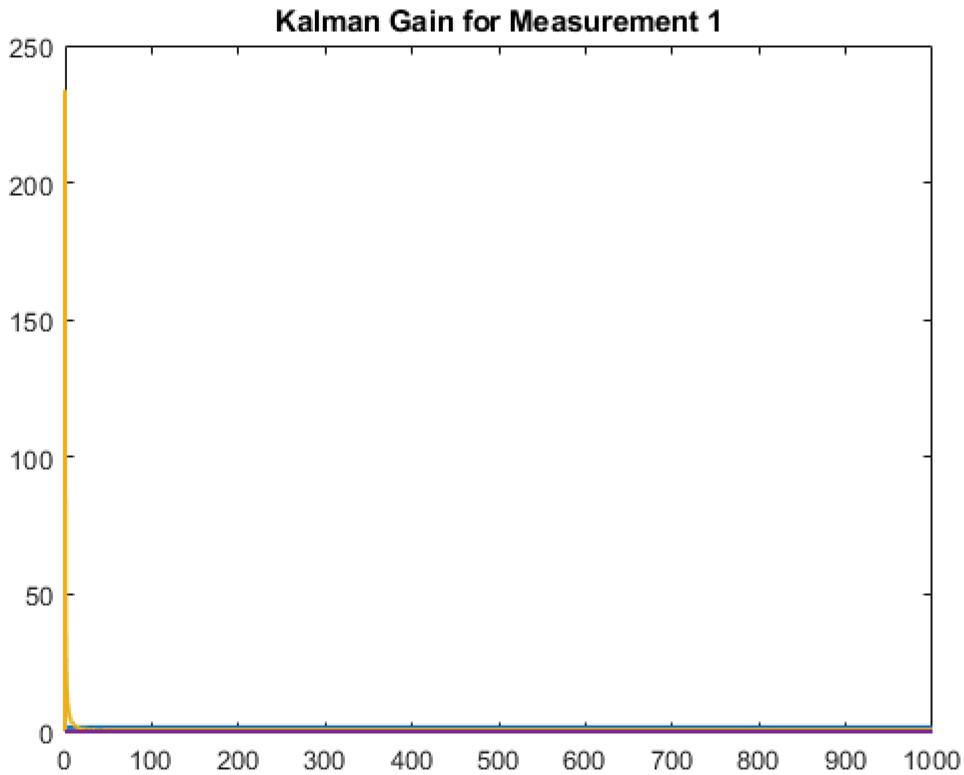

Covariance for x1

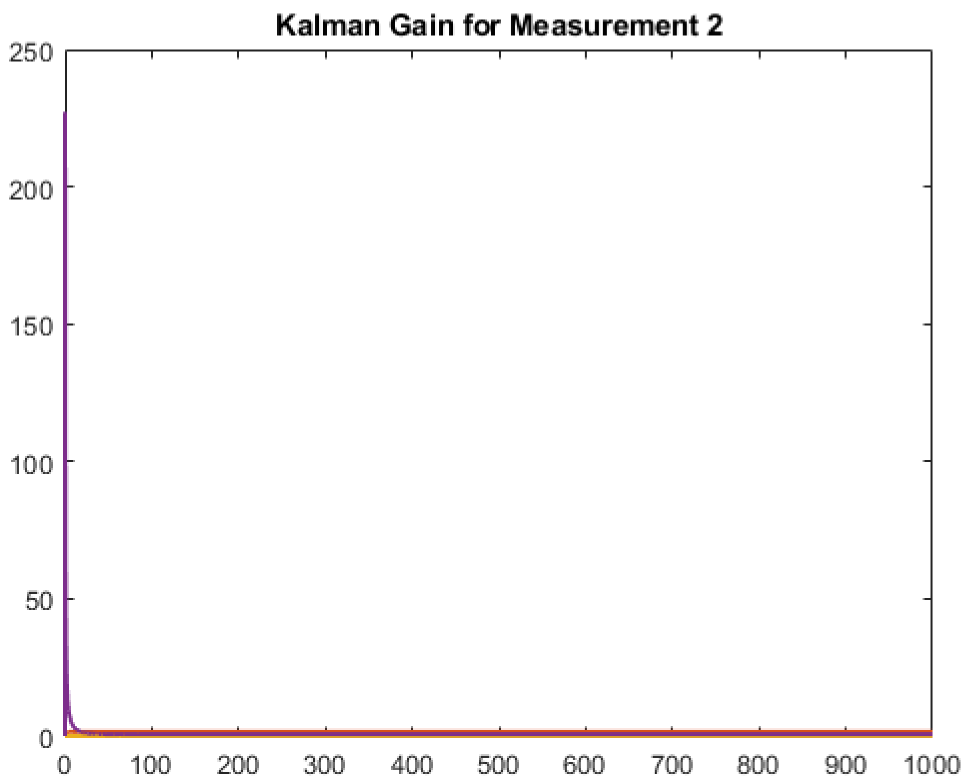

Covariance for x2
1000
900
800
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000


Covariance for x3
1000
900
800
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000


Covariance for x4
1000
900
800
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000


Residuals for Measurements
0
-0.002
-0.004
-0.006
-0.008
-0.01
-0.012
-0.014
-0.016
-0.018
-0.02
0
100
200
300
400
500
600
700
800
900
1000


Kalman Gain for Measurement 1
250
200
150
100
50
0
0
100
200
300
400
500
600
700
800
900
1000


Kalman Gain for Measurement 2
250
200
150
100
50
0
0
100
200
300
400
500
600
700
800
900
1000

Published with MATLAB® R2023a