Kalman Filter

Mini Project Report

Jasmin Karki
GE22M019

For the partial fulfillment of requirements for CH5120

October 17, 2022

1 Kalman Filter

1.1 Introduction

Kalman Filter is an estimation algorithm to produce estimates of unknown variables based on the measurements observed over time. In this report, it has been implemented to estimate the level of water in the 4 tanks present in the Quadruple tank experiment.

Kalman Filter is used in two steps. Initial estimates at $k = 0$ is provided to the first step which is called the prediction step, where the state is projected ahead by predicting the states and the error covariance is predicted. In the second step, the Kalman gain is computed; the estimates are updated with the known measurement and finally the error covariance is also updated. The outputs at k are the input to $k+1$ step. These two steps mentioned below are used repeatedly for each step until the heights of the tank are filtered and predicted.

Time Update (Predictor)

1. Project the state ahead
   $\hat{x}_k^- = A\hat{x}_{k-1} + Bu_k$
2. Project the error covariance ahead
   $\hat{P}_k^- = AP_{k-1}A^T + Q$

Measurement Update (Correction)

1. Compute the Kalman Gain
   $K_k = P_k^- H^T (HP_K^- H^T + R)^{-1}$
2. Update the estimate via z_k
   $\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-$
3. Update the error covariance
   $P_k = (I - K_k H)P_k^-$

1.2 Four Tank Problem

In the four tank problem, we have to control the level in the lower two tanks using two pumps. The input voltage to the pumps or process inputs are v_1 and v_2; the corresponding flow to the pumps are $k_1 v_1$ and $k_2 v_2$. The flow to tank 1, tank 2, tank 3 and tank 4 is $\gamma_1 k_1 v_1$, $\gamma_2 k_2 v_2$, $(1-\gamma_2)k_2 v_2$ and $(1-\gamma_1)k_1 v_1$ respectively as mentioned in the equation. Likewise, the flow to the outputs are voltage from level measurement devices, y_1 and y_2.

$$\frac{dh_1}{dt} = -\frac{a_1}{A_1}\sqrt{2gh_1} + \frac{a_3}{A_1}\sqrt{2gh_3} + \frac{\gamma_1 k_1}{A_1}v_1$$

$$\frac{dh_2}{dt} = -\frac{a_2}{A_2}\sqrt{2gh_2} + \frac{a_4}{A_2}\sqrt{2gh_4} + \frac{\gamma_2 k_2}{A_2}v_2$$

$$\frac{dh_3}{dt} = -\frac{a_3}{A_3}\sqrt{2gh_3} + \frac{1-\gamma_2 k_2}{A_3}v_2$$

$$\frac{dh_4}{dt} = -\frac{a_4}{A_4}\sqrt{2gh_4} + \frac{1-\gamma_1 k_1}{A_4}v_1$$

We have non-linear differential equations that are linearized around the operating point (x_0, v_0). The system model is continuous-time dynamic; we discretize it using zero-order hold on the inputs and a sample time of Ts. The minimum phase characteristics of the system has operating points corresponding to following parameter values:

$(h_1^0, h_2^0) = (12.4, 12.7), (h_3^0, h_4^0) = (1.8, 1.4)$,
$(v_1^0, v_2^0) = (3, 3)$,
$(k_1, k_2) = (3.33, 3.35)$ and
$(\gamma_1, \gamma_2) = (0.7, 0.6)$
$(T_1, T_2) = (62, 90)$

Process noise covariance(Q), measurement noise(R) has been tuned in the system. The initial state of the state vector has been guessed.

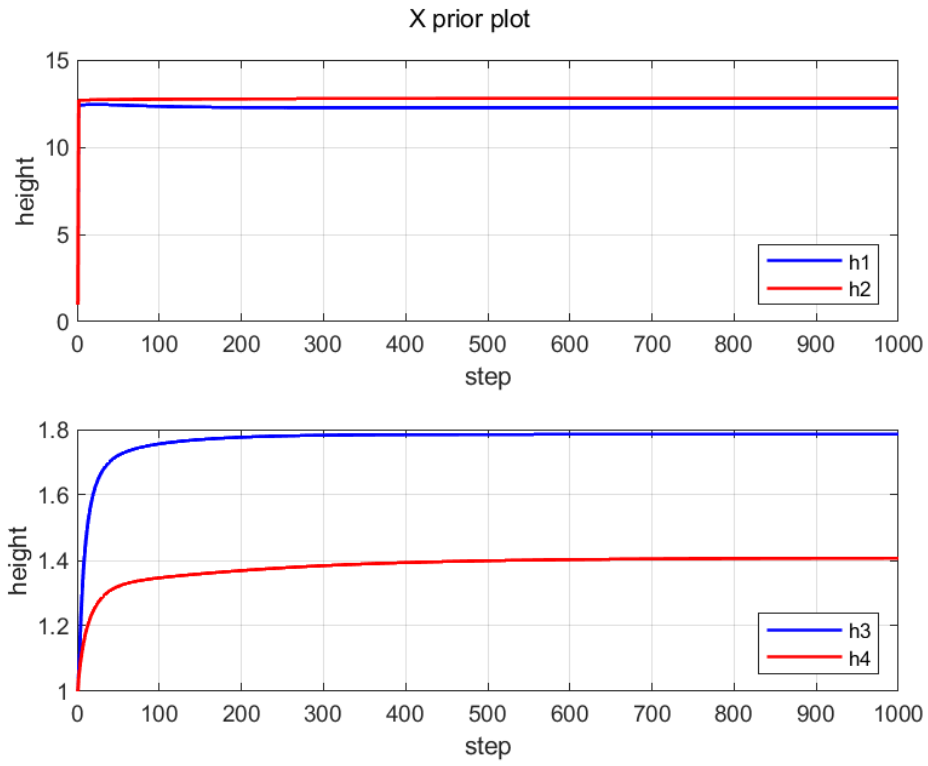
1.3 Code Implementation

```
%% Constants
A1 = 28;    A2 = 32;    A3 = 28;    A4 = 32;
a1 = 0.071; a2 = 0.057; a3 = 0.071; a4 = 0.057;
kc = 0.50;  g=981;

%% Minimum Phase Characteristics
h1o = 12.4;    h2o = 12.7;  h3o = 1.8;    h4o = 1.4;
v1o = 3;    v2o = 3;
Xo = [h1o; h2o; h3o; h4o];
Uo = [v1o; v2o];
T1 = 62;    T2 = 90;    T3 = 23;    T4 = 30;
k1 = 3.33;  k2 = 3.35;  gamma1 = 0.7;  gamma2 = 0.6;
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
```

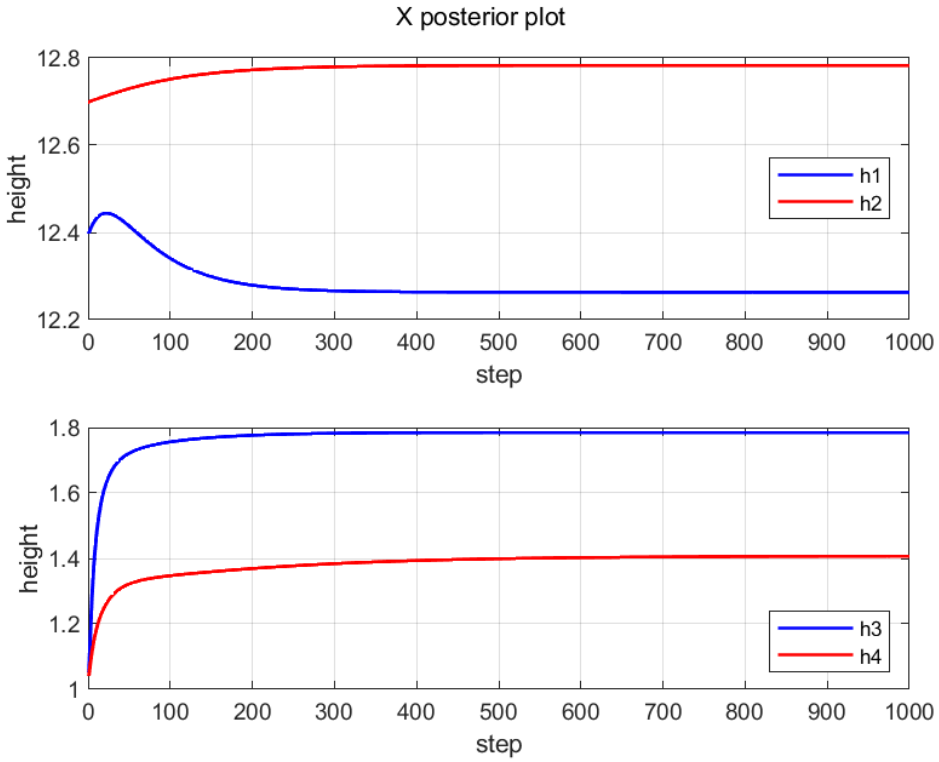
```
        (1-gamma1)*k1/A4        0];
Cc = [kc      0     0     0;
      0       kc    0     0];
Dc = 0;
msrmnts = xlsread('Measurements'); % Original Measurements
X_post = [1;1;1;1];
P_post = 10^5*eye(4);     Q = 20*eye(4);     R = 2*eye(2);

Ts=0.1;           flag = false;
continuousSys = ss(Ac,Bc,Cc,Dc);          % Ct time SS model
discreteSys =   c2d(continuousSys, Ts); % Ct to Dt time
[A, B, H, D] = ssdata(discreteSys);       % SS Data

v1_store = [];                  % flow pump 1
v2_store = [];                  % flow pump 2
X_prior_store = [];             % store prior X
P_prior_store = [];             % store prior covariance
X_post_store = [];              % store post X
P_post_store = [];              % store post covariance
K_store = [];                   % store Kalman Gain
Innov_store = [];               % Msmnts error X prior
Residual_store = [];            % Msmnts error X post
Trace_P_prior = [];             % Store trace of P prior
Trace_P_post = [];              % Store trace of P post

for i=1:1000
% Compute u for each step using forward difference eq
dh1= (msrmnts(i+1,1)-msrmnts(i,1))/ Ts;          % h1 dot
dh2= (msrmnts(i+1,2)-msrmnts(i,2))/ Ts;          % h2 dot
h1 = msrmnts(i,1);      h2 = msrmnts(i,2);
h3 = msrmnts(i,3);      h4 = msrmnts(i,4);

v1 = (dh1 + (a1*sqrt(2*g*h1))/A1 - (a3*sqrt(2*g*h3))/A1)*(A1
    /(gamma1*k1));
v2 = (dh2 + (a2*sqrt(2*g*h2))/A2 - (a4*sqrt(2*g*h4))/A4)*(A2
    /(gamma2*k2));
v1_store = [v1_store v1];                        % Input U(1)
v2_store = [v2_store v2];                        % Input U(2)
U = [v1 v2]';                                    % Input U
```

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

% Kalman Gain calculation
K=P_prior*transpose(H)*(inv(H*P_prior*transpose(H) + R));

% True Z value
Z_true = H*transpose(msrmnts(i,1:4));

% Error between estimated measurement and true measurement
Innovation = Z_true - H*X_prior;

% Update equations
X_post = X_prior + K*Innovation;
P_post = P_prior-K*H*P_prior;

Residual = Z_true - H*X_post;

X_prior_store = [X_prior_store X_prior];
P_prior_store = [P_prior_store P_prior];
X_post_store = [X_post_store X_post];
P_post_store = [P_post_store P_post];
K_store = [K_store  K];
Innov_store = [Innov_store Innovation];
Residual_store = [Residual_store Residual];
Trace_P_post = [Trace_P_post trace(P_post)];
Trace_P_prior = [Trace_P_prior trace(P_prior)];
norm_X = norm(X_post-X_prior);
norm_P_prior = abs(norm(trace(P_prior)));
norm_P_post = abs(norm(trace(P_post)));
if flag == false & norm_X < 5e-3
    flag = true;
    sprintf("Converges %d", i)
end
end
```

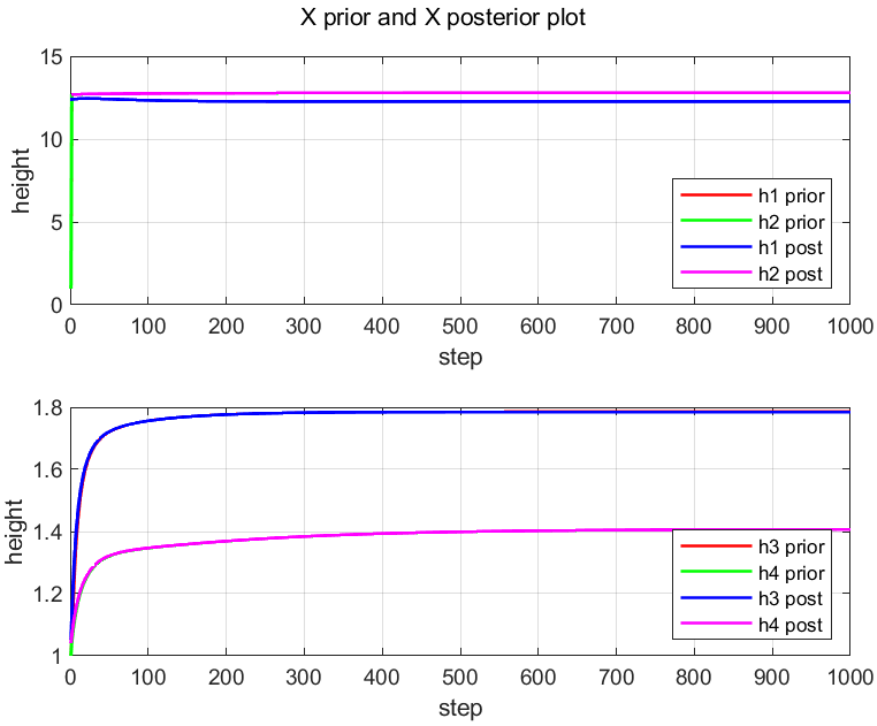
The code for visualizations has not been mentioned in the report.



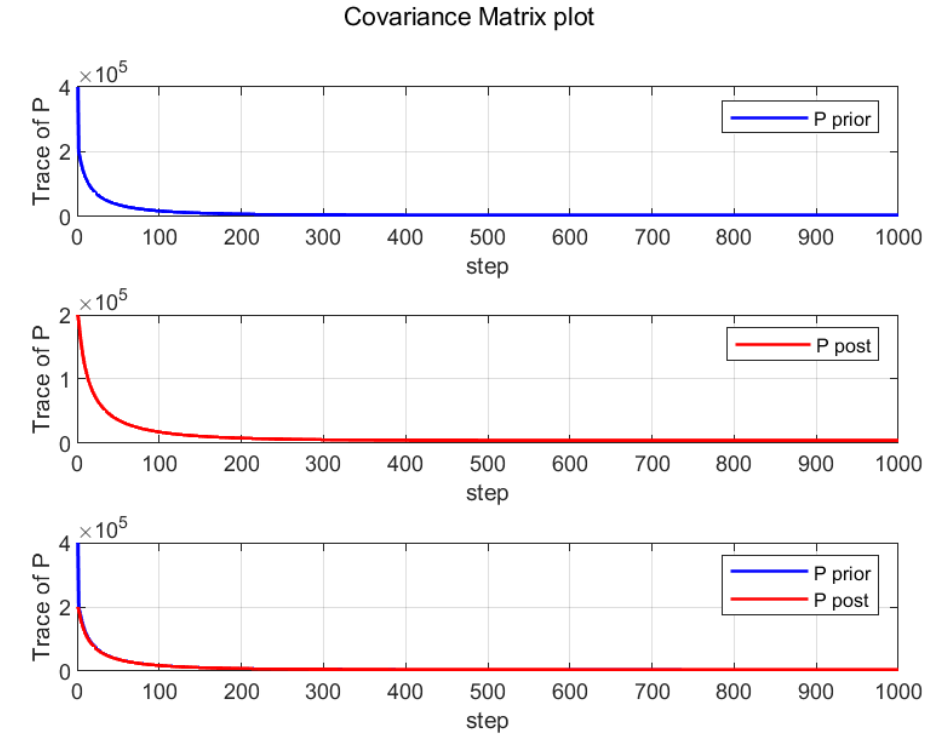
(a) X prior plot



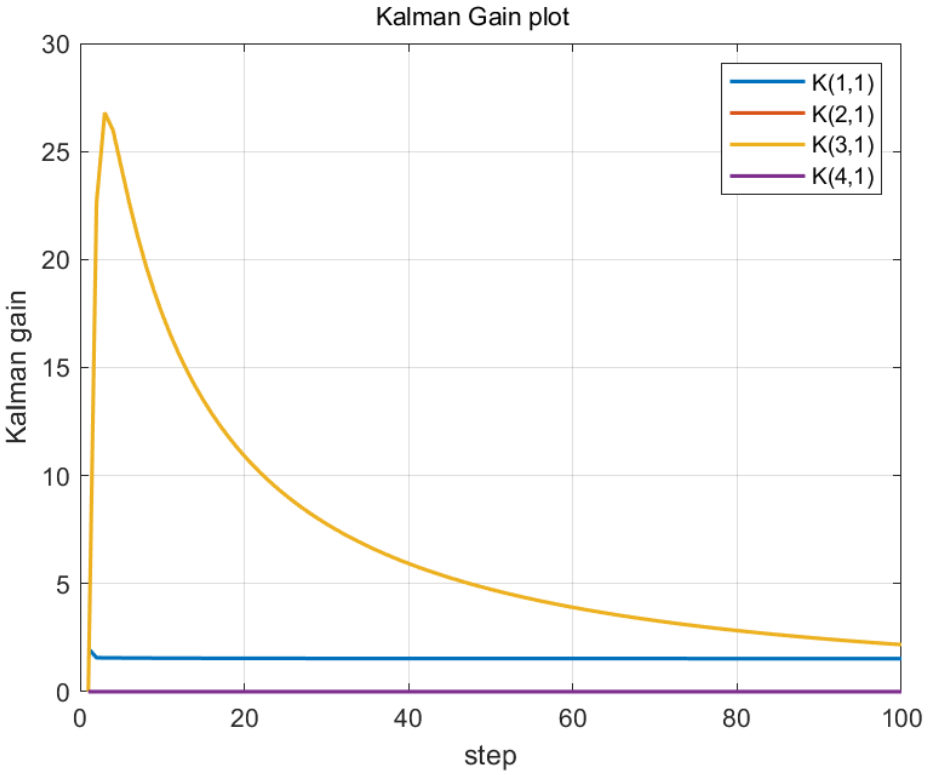
(b) X posterior plot



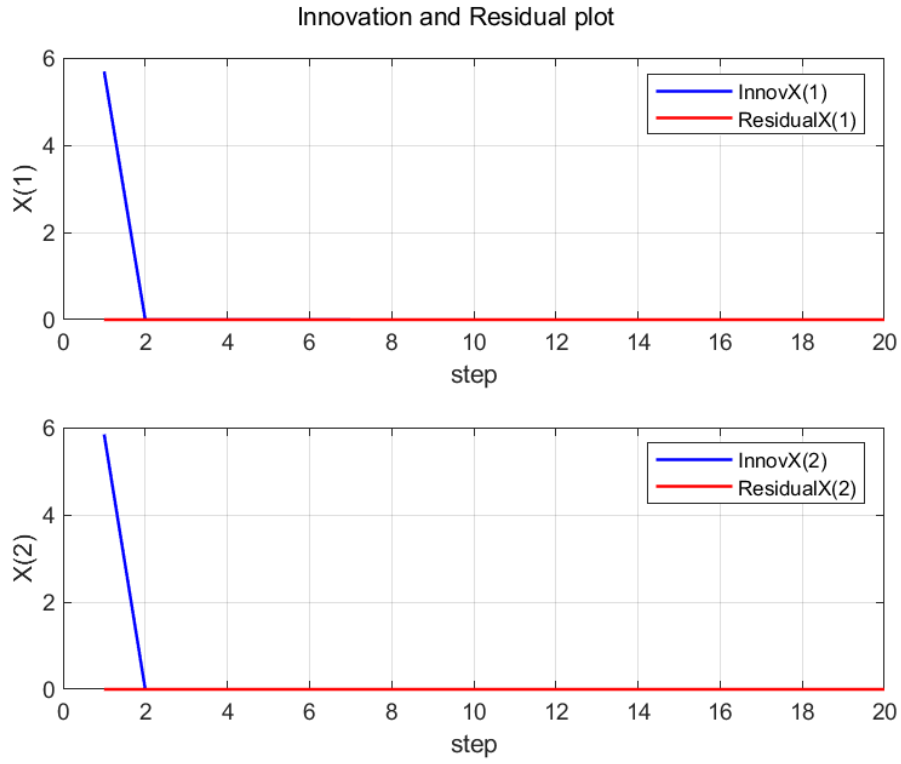
(a) X prior and posterior graph



(b) Covariance Matrix plot



(a) Kalman Gain (100 steps)



(b) Innovation and Residual Plot

1.4 Visualization

The predicted and estimated states of first 1000 steps along with their comparison is visualized in figure 1a 1b 2a. Likewise, the predicted and estimated covariance matrix of 1000 steps in figure 2b, 100 steps of Kalman gain and innovation-residual in figure 3a and 3b respectively.

1.5 Inference

With the initial covariance P of 10^5, process noise Q of 20 and measurement noise R of 2. Since the system dynamics is governed by the non-linear differential equations, the process noise is greater than the measurement noise. Tuning the value of Q and R based on this criteria, it was found that the R doesn't impact much on the convergence as compared to Q. When Q was increased, the convergence required more steps. Likewise the Kalman gain has lower value as R gets smaller for constant Q.

The system converges in about 28 steps with $Q = 20$ and $R = 2$. The predicted states (X prior) in figure 1a and estimated states (X posterior) in figure 1b both converge to the actual measurements i.e. height of water level in tanks. Likewise, the X prior and X posterior seem to overall as shown in figure 2a. Figure 2b shows the plot of trace of covariance matrix P_{prior} and $P_{posterior}$. The graph shows that the covariance is decreasing and converged after around 300 steps. The value of $P_{posterior}$ is less than P_{prior} signifying that at each step, the predicted covariance is further reduced while estimating the updated covariance. The Kalman gain as shown in figure 3a reduced from higher value to lower value and remained same throughout the remaining steps. Finally the innovation and residual plot is represented by figure 3b. The residual value is smaller than innovation value. Initially, the predicted state is high resulting a high value of innovation. As the predicted state is updated using Kalman filter, the residual is reduced.

2 Particle Filter

2.1 Code Implementation

```
N = 500;                % particle size
n = 4;                  % states
L = chol(P);            % cholesky factor of P
% Adding covariance to each particle
x = (z10K(1,:)'*ones(1,N))'+ randn(N,n)*L;
x1 = x(:,1);    x2 = x(:,2);    x3 = x(:,3);    x4 = x(:,4);
%% Process Noise
```

```
Q = 100*eye(4);                 % process noise
w = chol(Q)*randn(n,N);         % roughening of the prior
w1 = w(1,:); w2 = w(2,:); w3 = w(3,:); w4 = w(4,:);
%% Roughening the Prior
x1 = x1 + w1'; x2 = x2 + w2'; x3 = x3 + w3'; x4 = x4 + w4';

% Initialize Prediction values
xpred = zeros(N,n);
x1pred = xpred(:,1);     x2pred = xpred(:,2);
x3pred = xpred(:,3);     x4pred = xpred(:,4);

%% Prediction Step
for i = 1:N
      x1pred(i) = -a1/A1*sqrt(2*g*x1(i)) + a3/A1*sqrt(2*g*x3(
          i)) + (gamma1*k1*v1)/A1 + w1(i);
      x2pred(i) = -a2/A2*sqrt(2*g*x2(i)) + a4/A2*sqrt(2*g*x4(
          i)) + (gamma2*k1*v2)/A2 + w2(i);
      x3pred(i) = -a3/A3*sqrt(2*g*x3(i)) + (1 - gamma2)*k2*v2
          /A3  +  w3(i);
      x4pred(i) = -a4/A4*sqrt(2*g*x4(i)) + (1 - gamma1)*k1*v1
          /A4  +  w4(i);
end
xpred  = abs(xpred);
x1pred = abs(x1pred);    x2pred = abs(x2pred);
x3pred = abs(x3pred);    x4pred = abs(x4pred);

% Importance Weights (Likelihood Function)
z1 = z10K(1,1);      z2 = z10K(1,2);
z = [z1; z2];
z_true = z * ones(1,N);
R = 100 * eye(2);
z_est  =  C*xpred';
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(i) = x1pred(j);       x2(i) = x2pred(j);
     x3(i) = x3pred(j);       x4(i) = x4pred(j);
    i = i + 1;
end

x1_pri_cum = cumsum(x1pred);      x2_pri_cum = cumsum(x2pred);
x3_pri_cum = cumsum(x3pred);      x4_pri_cum = cumsum(x4pred);

x1_cum = cumsum(x1);      x2_cum = cumsum(x2);
x3_cum = cumsum(x3);      x4_cum = cumsum(x4);

x_post = [x1 x2 x3 x4];

x1_est = mean(x1);
x2_est = mean(x2);
x3_est = mean(x3);
```

2.2 Visualization

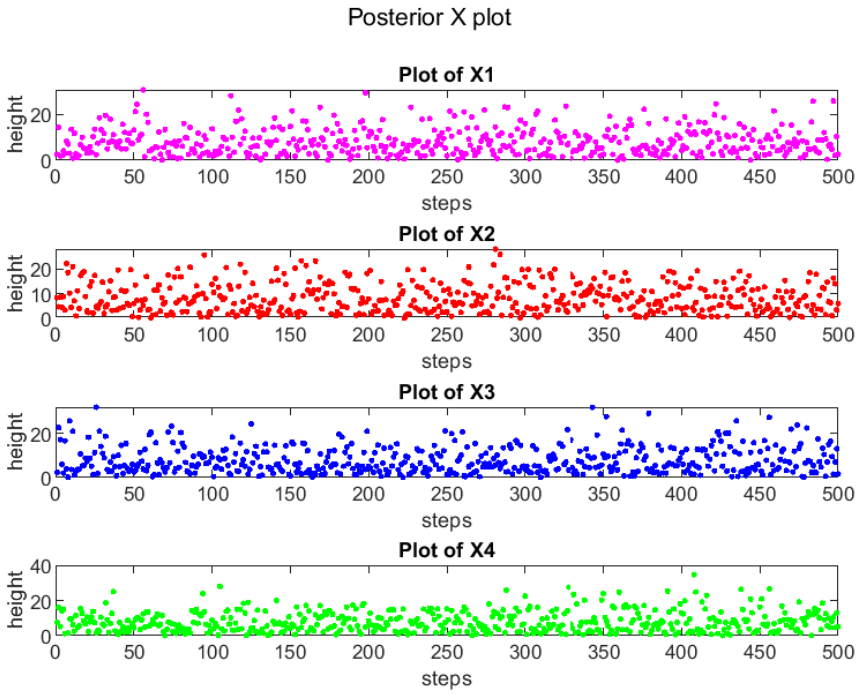


Figure 4: Particle Filter X prior Plot