Assignment No: 4

Table of Contents

Kalman Update code 1
Kalman Prediction code 1
Main Code 1

Kalman Update code

```
function [posteriorMean,posteriorCovariance,gainX] =
 kalmanUpdate(priorMean,priorCovariance,z,H,R)
    % TASK 3 - Complete this function
    K = priorCovariance * H.' /(H*priorCovariance*H.' + R) ;
    posteriorMean = priorMean + K *(z - H*priorMean);
    posteriorCovariance = (eye(4) - K *H)*priorCovariance;
    gainX = K(1,1);
end
```

Kalman Prediction code

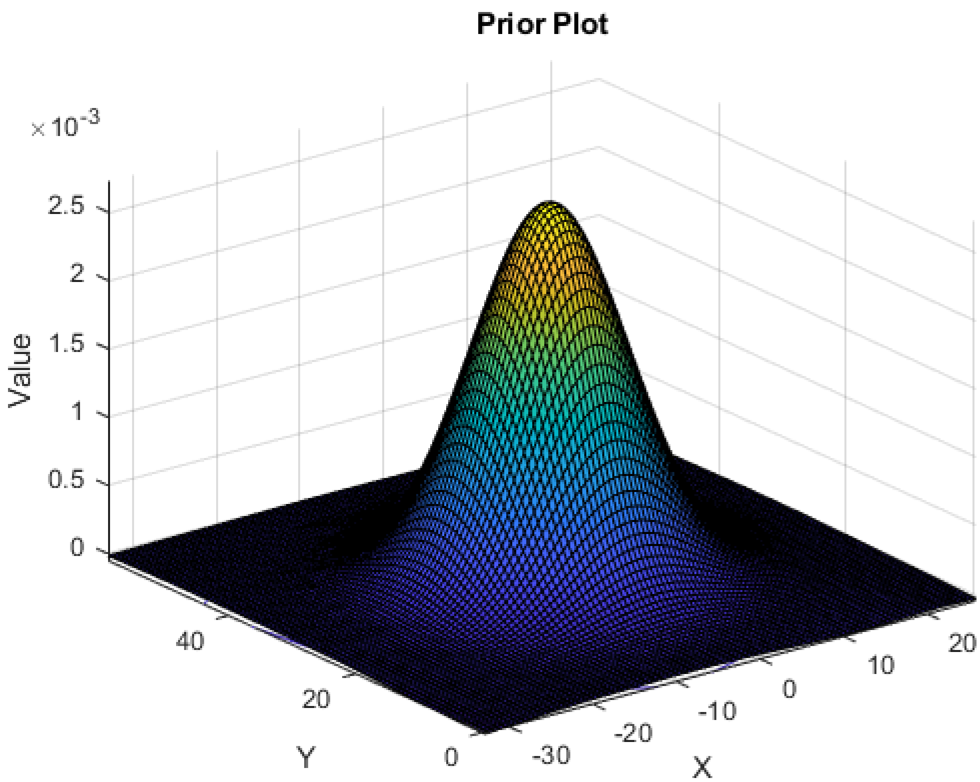
```
function [priorMean,priorCovariance] =
 kalmanPrediction(posteriorMean,posteriorCovar,F,Q)
    %TASK 3 - Complete this function
    priorMean =  F *posteriorMean ;
    priorCovariance = F*posteriorCovar*F.' + Q;
end
```

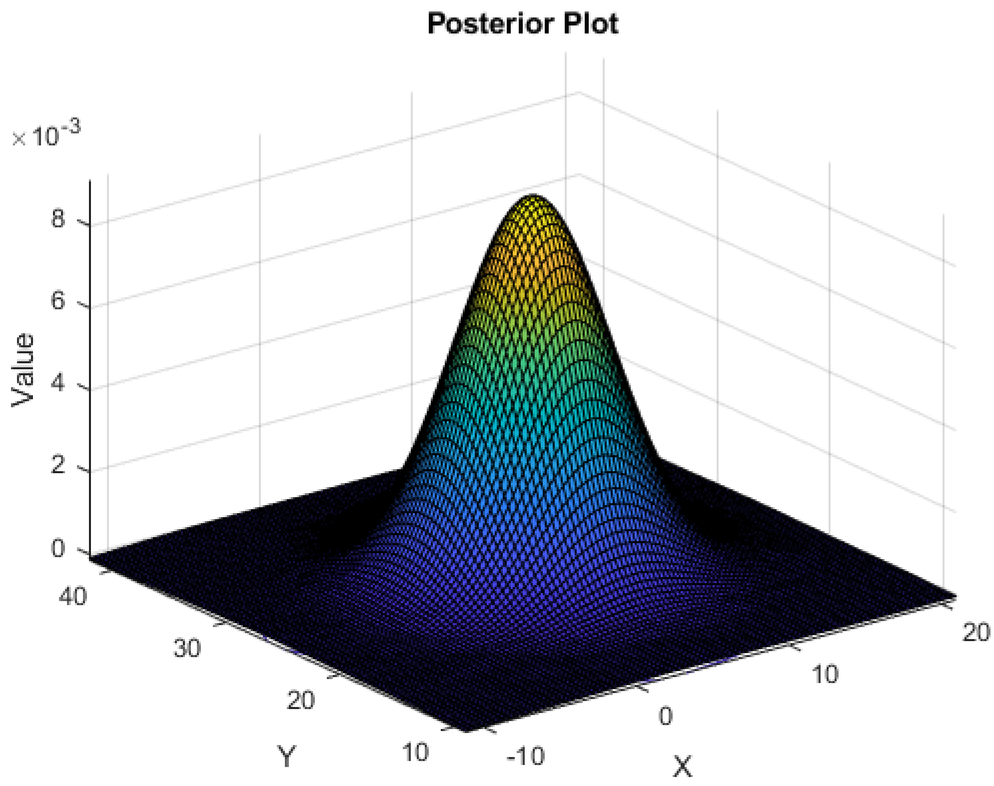
Main Code

```
close all
clearvars

T = 1; % Time step is one second
F = [ 1, 0, T, 0;0, 1, 0, T;0, 0, 1, 0; 0, 0, 0, 1]; % TASK 1 - Complete the
 state transition matrix
proNoise = 0.01; % Process noise intensity q
Q = proNoise*[T^3/3, 0, T^2/2, 0; 0, T^3/3, 0, T^2/2; T^2/2, 0, T, 0; 0,T^2/2,
 0, T];% TASK 1 - Complete the transition noise covariance matrix

sigmaX = 5; % Measurement error standard deviation in x
sigmaY = 5; % Measurement error standard deviation in y
```

```
R = [sigmaX^2, 0;0, sigmaY^2]; % TASK 2 - Complete the measurement error
 covariance matrix
H = [1, 0;0, 1; 0, 0;0, 0].'; % TASK 2 - Complete the measurement matrix

load('data.mat')

estimate = zeros(4,60);
gain = zeros(1,60);
% Indexing for 60 times steps
for i = 1:60
   %Store all the measurements
   z = measurements(:,i);
   if i == 2
      % In the second time step perform initialisation
      mean = [z(1) z(2) z(1)-measurements(1,i-1) z(2)-measurements(2,i-1)]';
      covar = [R(1,1) 0 R(1,1) 0; 0 R(2,2) 0 R(2,2); R(1,1) 0 2*R(1,1) 0; 0
 R(2,2) 0 2*R(2,2)];
      estimate(:,i) = mean;
   elseif i > 2

       % Perform the Kalman filter prediction
       [priorMean, priorCovar] = kalmanPrediction(mean,covar,F,Q);

       if i == 4
           State = mvnrnd(priorMean(1:2)',priorCovar(1:2,1:2),10000);
           Val = mvnpdf(State,priorMean(1:2)',priorCovar(1:2,1:2));
           xAxis = State(:,1); yAxis = State(:,2); zAxis = Val;

           threeD = scatteredInterpolant(xAxis,yAxis,zAxis);
           qx = linspace(min(xAxis),max(xAxis),100);
           qy = linspace(min(yAxis),max(yAxis),100);
           [qx,qy] = meshgrid(qx,qy);
           qz = threeD(qx,qy);
           surfc(qx,qy,qz);
           xlabel('X')
           ylabel('Y')
           zlabel('Value')
           title('Prior Plot')
           % TASK 4 - Plot the prior pdf using surf and mvnpdf
       end

       % Perform the Kalman filter update and log the Kalman gain
       % additionally
       [mean,covar,gain(:,i)] = kalmanUpdate(priorMean,priorCovar,z,H,R);

       if i == 4
           figure
           postState = mvnrnd(mean(1:2).',covar(1:2,1:2),10000);
           postVal = mvnpdf(postState,mean(1:2).',covar(1:2,1:2));
           pxAxis = postState(:,1); pyAxis = postState(:,2); pzAxis = postVal;
           threeD = scatteredInterpolant(pxAxis,pyAxis,pzAxis);
           postx = linspace(min(pxAxis),max(pxAxis),100);
           posty = linspace(min(pyAxis),max(pyAxis),100);
           [postx,posty] = meshgrid(postx,posty);
```

```
            postz = threeD(postx,posty);
            surfc(postx,posty,postz);
            xlabel('X')
            ylabel('Y')
            zlabel('Value')
            title('Posterior Plot')

            % TASK 4 - Plot the posterior pdf using surf and mvnpdf
        end

        % Log the estimate
        estimate(:,i) = mean;
    end
end
```





TASK 4- QUESTION

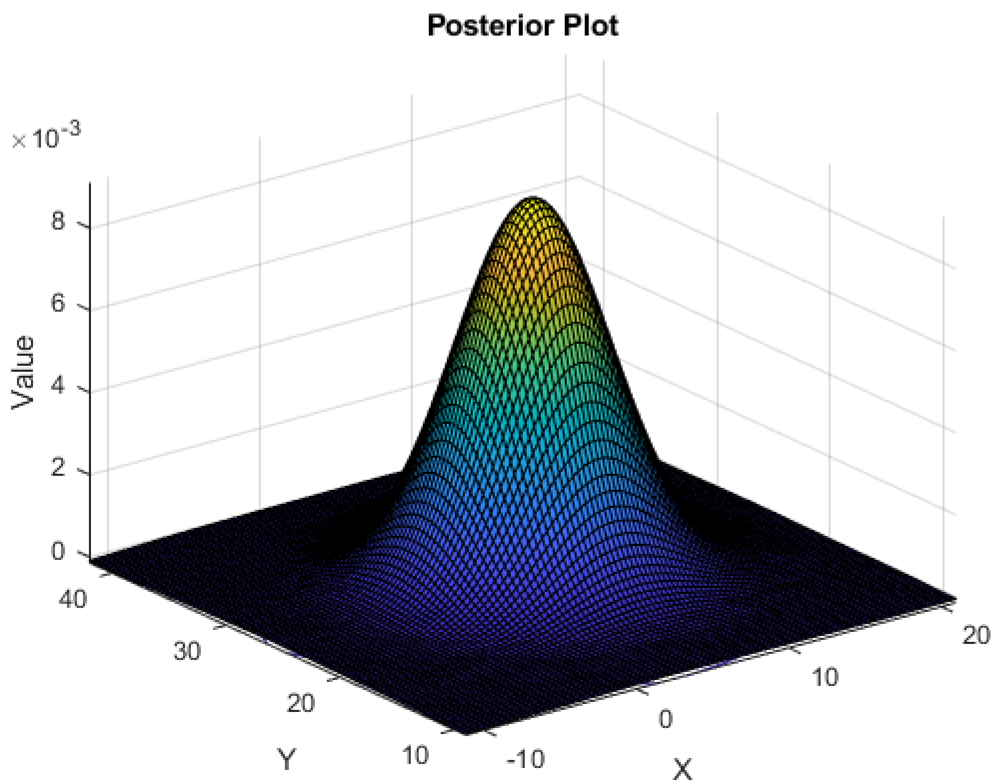
```
%The Best Estimator for this problem would be MAP or MMSE since the state here
 is a Random Vairable that can take value from a given range. Both provide
 similar optimal estimate and here we have a Gaussian Prior, so both give the
 same answer, unlike the other case of Prior being Non-Gaussian
```

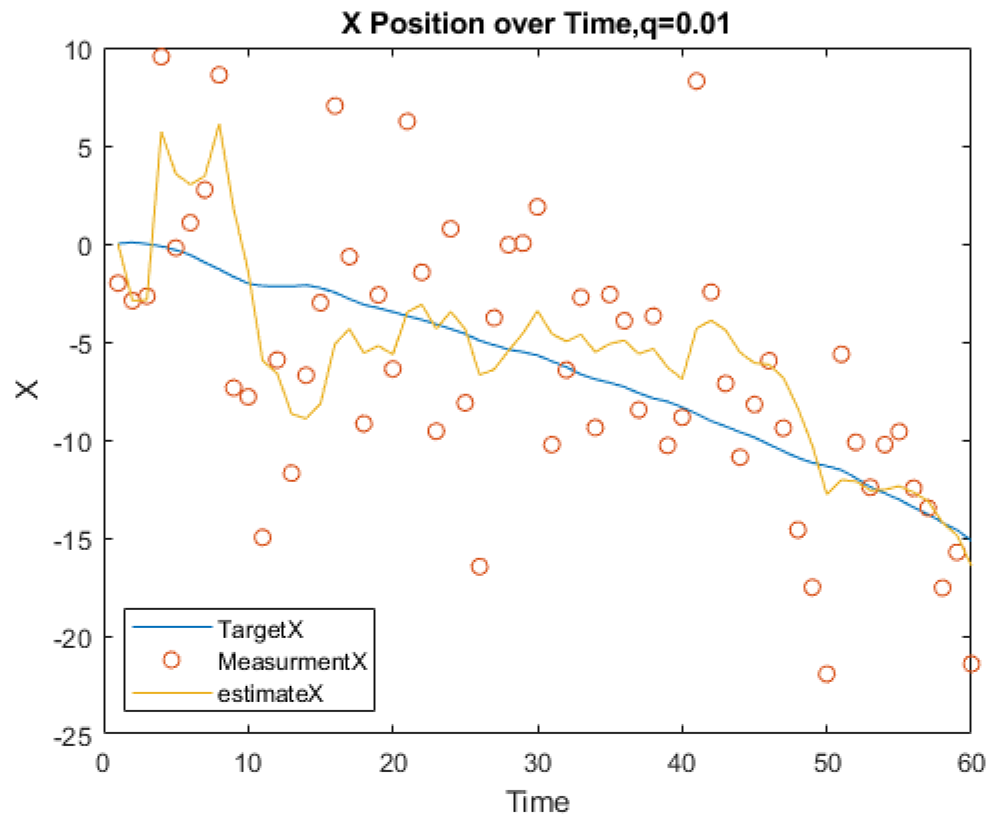
TASK 5 - Plot the true state, the measurements and state estimate

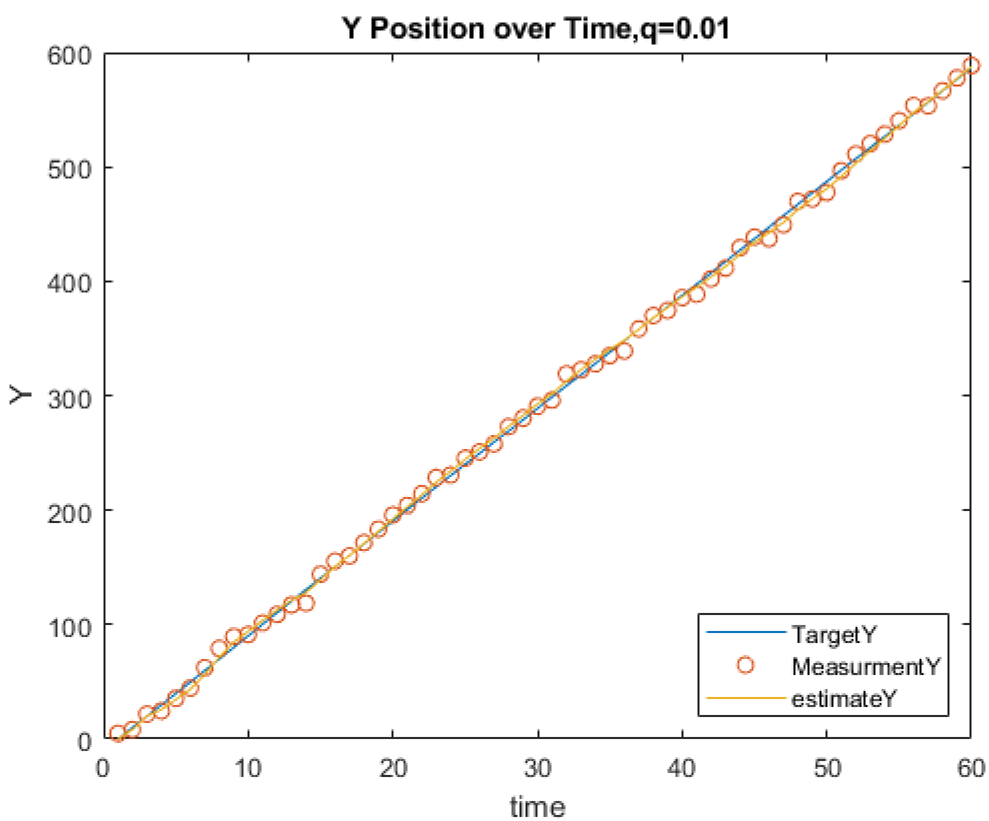
```
figure
tim = 1:60;
plot(tim,targetState(1,:));
hold on;
scatter(tim,measurements(1,:));
hold on;
plot(tim,estimate(1,:));
title('X Position over Time,q=0.01')
ylabel('X')
xlabel('Time')
legend({'TargetX','MeasurmentX','estimateX'},'Location','southwest')

figure
tim = 1:60;
plot(tim,targetState(2,:));
hold on;
scatter(tim,measurements(2,:));
hold on;
```

```
plot(tim,estimate(2,:));
title('Y Position over Time,q=0.01')
ylabel('Y')
xlabel('time')
legend({'TargetY','MeasurmentY','estimateY'},'Location','southeast')

Noise1LastEstimate = estimate(:,60)
```

```
Noise1LastEstimate =

  -16.4385
  587.6455
   -0.5741
   10.2068
```




X Position over Time,q=0.01
X
Time
TargetX
MeasurmentX
estimateX

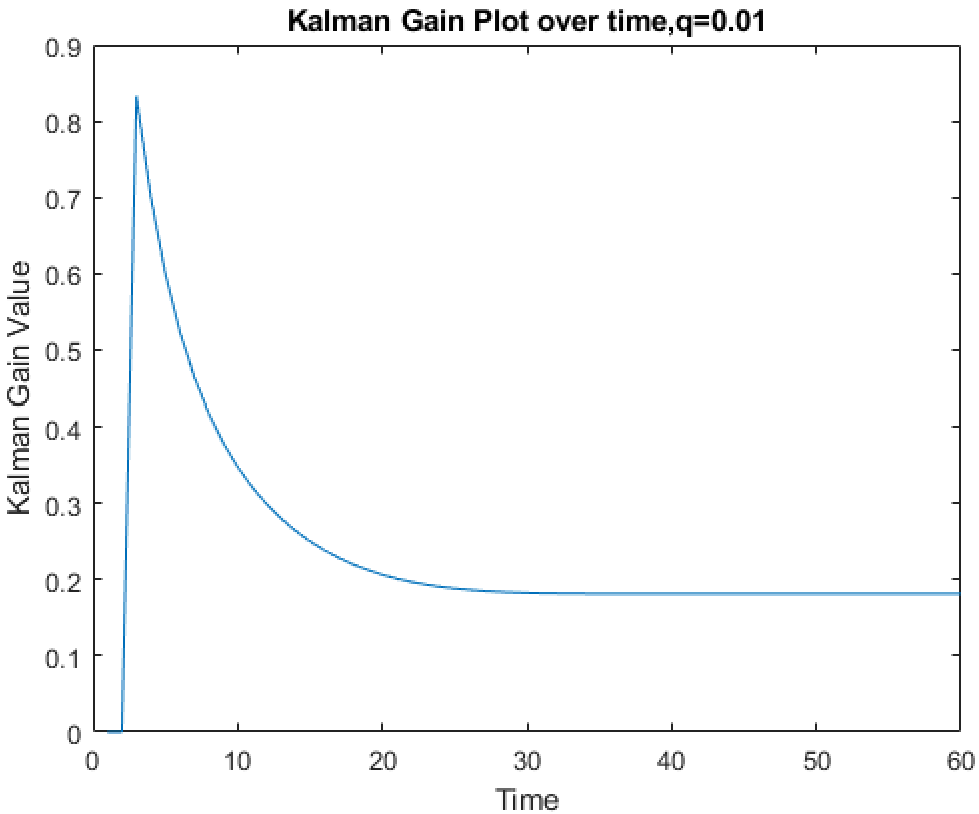

Y Position over Time,q=0.01
Y
time
TargetY
MeasurmentY
estimateY

TASK 5 - QUESTION

```
%State Estimation helps us gain a good idea about all the states of the
%system given the Measurement which can be very erratic due to sensor or
%object motion errors. Thus tracking an object becomes very informative for
%any other task that the system must do.
```

TASK 6 - Plot the Kalman filter gain

```
figure
plot(gain(1,:))
ylabel('Kalman Gain Value')
xlabel('Time')
title('Kalman Gain Plot over time,q=0.01')
```



TASK 6 - QUESTION

```
%Kalman filter gain helps us identify the Measurement Innovation values for
%our model. This represents the variance of our state estimator vs
 measurement,
%and the graph shows that we are approaching a better state estimate with each
 measurement as time progresses.
```

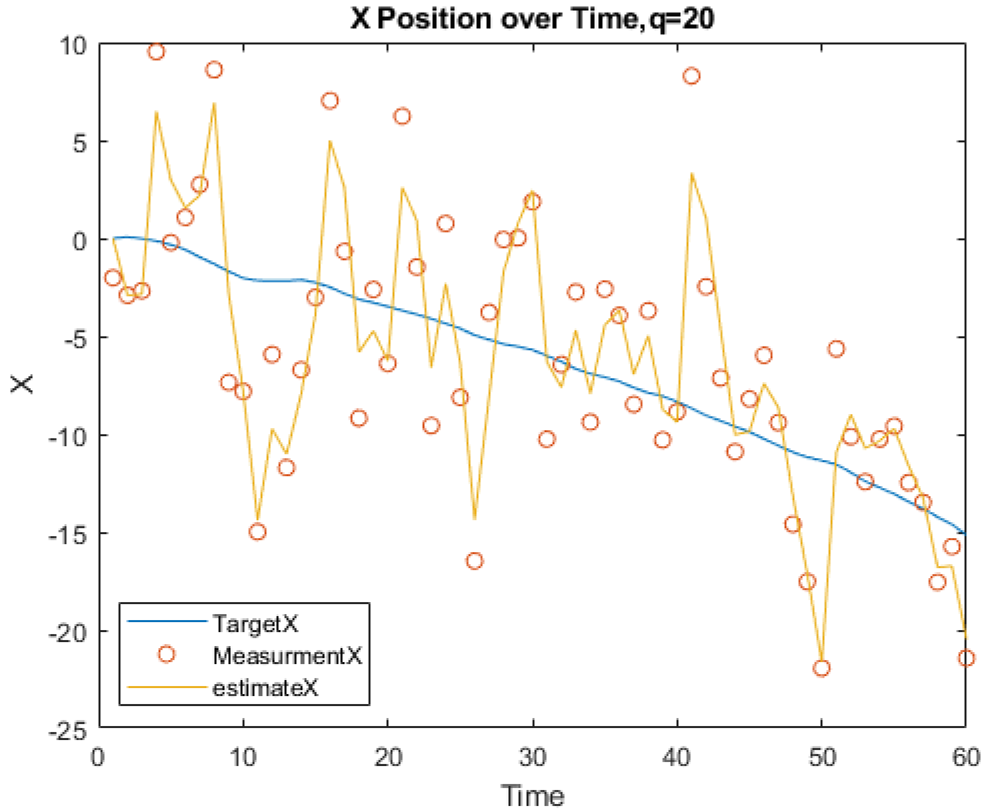
TASK 7

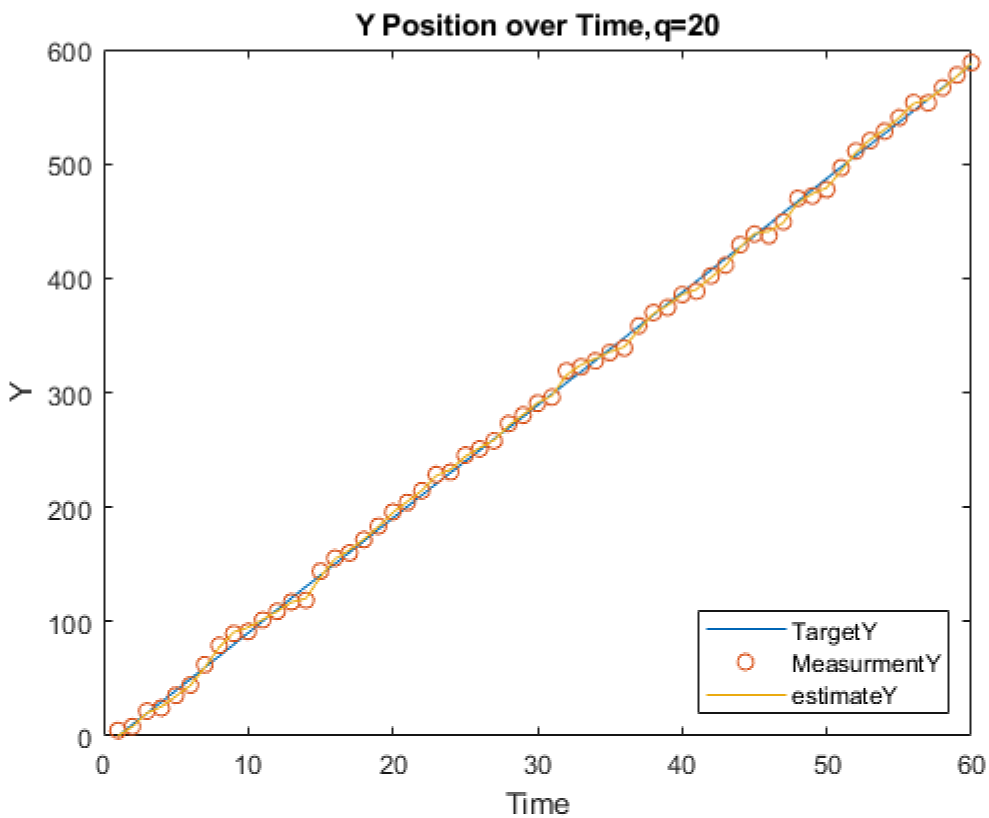
```
proNoise2 = 20; % Process noise intensity q
Q2 = proNoise2*[T^3/3, 0, T^2/2, 0; 0, T^3/3, 0, T^2/2; T^2/2, 0, T, 0;
 0,T^2/2, 0, T];% TASK 1 - Complete the transition noise covariance matrix
```

```
sigmaX = 5; % Measurement error standard deviation in x
sigmaY = 5; % Measurement error standard deviation in y
R = [sigmaX^2, 0;0, sigmaY^2]; % TASK 2 - Complete the measurement error
 covariance matrix
H = [1, 0;0, 1; 0, 0;0, 0].'; % TASK 2 - Complete the measurement matrix

load('data.mat')

estimate = zeros(4,60);
gain = zeros(1,60);
% Indexing for 60 times steps
for i = 1:60
   %Store all the measurements
   z = measurements(:,i);
   if i == 2
      % In the second time step perform initialisation
      mean = [z(1) z(2) z(1)-measurements(1,i-1) z(2)-measurements(2,i-1)]';
      covar = [R(1,1) 0 R(1,1) 0; 0 R(2,2) 0 R(2,2); R(1,1) 0 2*R(1,1) 0; 0
 R(2,2) 0 2*R(2,2)];
      estimate(:,i) = mean;
   elseif i > 2

       % Perform the Kalman filter prediction
       [priorMean, priorCovar] = kalmanPrediction(mean,covar,F,Q2);

       if i == 4
           State = mvnrnd(priorMean(1:2)',priorCovar(1:2,1:2),10000);
           Val = mvnpdf(State,priorMean(1:2)',priorCovar(1:2,1:2));
           % TASK 4 - Plot the prior pdf using surf and mvnpdf
       end

       % Perform the Kalman filter update and log the Kalman gain
       % additionally
       [mean,covar,gain(:,i)] = kalmanUpdate(priorMean,priorCovar,z,H,R);

       if i == 4
           postState = mvnrnd(mean(1:2).',covar(1:2,1:2),10000);
           postVal = mvnpdf(postState,mean(1:2).',covar(1:2,1:2));

           % TASK 4 - Plot the posterior pdf using surf and mvnpdf
       end

       % Log the estimate
       estimate(:,i) = mean;
   end
end


% TASK 5 - Plot the true state, the measurements and state estimate
figure
tim = 1:60;
plot(tim,targetState(1,:));
hold on;
```

```
scatter(tim,measurements(1,:));
hold on;
plot(tim,estimate(1,:));
title('X Position over Time,q=20')
ylabel('X')
xlabel('Time')
legend({'TargetX','MeasurmentX','estimateX'},'Location','southwest')

figure
tim = 1:60;
plot(tim,targetState(2,:));
hold on;
scatter(tim,measurements(2,:));
hold on;
plot(tim,estimate(2,:));
title('Y Position over Time,q=20')
ylabel('Y')
xlabel('Time')
legend({'TargetY','MeasurmentY','estimateY'},'Location','southeast')

Noise2LastEstimate = estimate(:,60)


% TASK 6 - Plot the Kalman filter gain
figure
plot(gain(1,:))
ylabel('Kalman Gain Value')
xlabel('Time')
title('Kalman Gain Plot over time,q=20')
```

```
Noise2LastEstimate =

  -20.4568
  588.0735
   -2.6916
   10.8939
```


X Position over Time, q=20
10
5
0
-5
-10
-15
-20
-25
X
0
10
20
30
40
50
60
Time
TargetX
MeasurmentX
estimateX


Y Position over Time, q=20
600
500
400
300
200
100
0
Y
0
10
20
30
40
50
60
Time
TargetY
MeasurmentY
estimateY

Published with MATLAB® R2021b