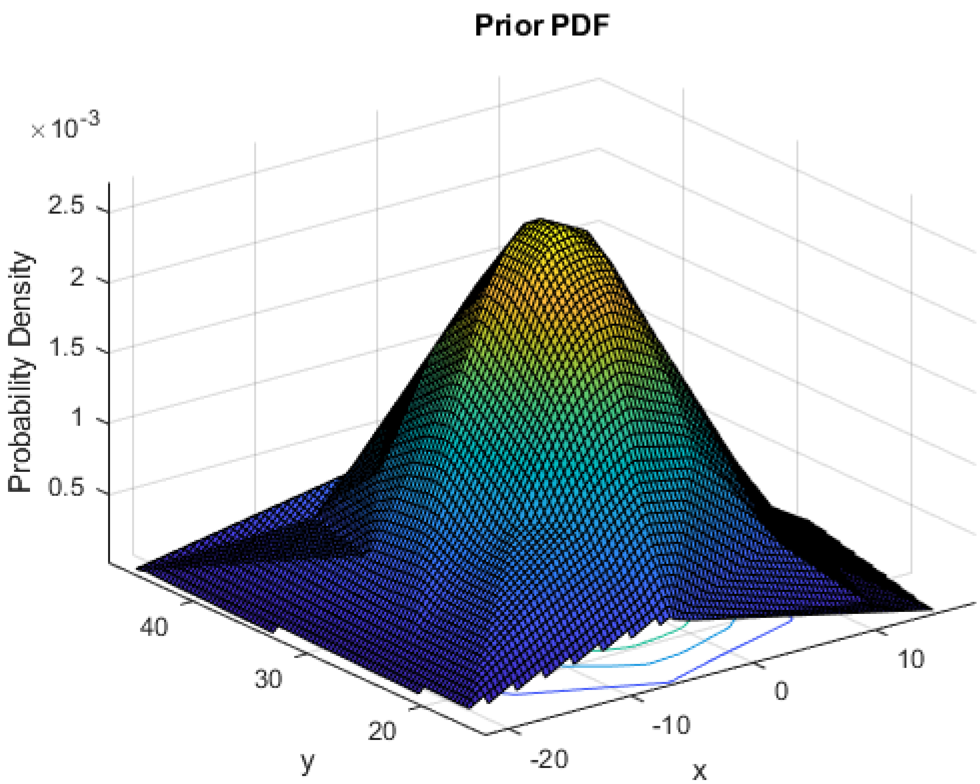
```
close all
clearvars

T = 1; % Time step is one second
time_step = 1
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 0; 0 1 0 0]; % TASK 2 - Complete the measurement matrix
load('data.mat');
input_data = load('data.mat');
measurements_input_data = input_data.measurements;
target_state_input_data = input_data.targetState;
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
           % TASK 4 - Plot the prior pdf using surf and mvnpdf
            X = mvnrnd(priorMean(1:2).',priorCovar(1:2,1:2),100);
            Y = mvnpdf(X, priorMean(1:2).',priorCovar(1:2,1:2));
            xx = X(:,1); yy = X(:,2); zz = Y;
            FF = TriScatteredInterp(xx,yy,zz);
            qx = linspace(min(xx),max(xx),100);
            qy = linspace(min(yy),max(yy),100);
            [qx,qy] = meshgrid(qx,qy);
```

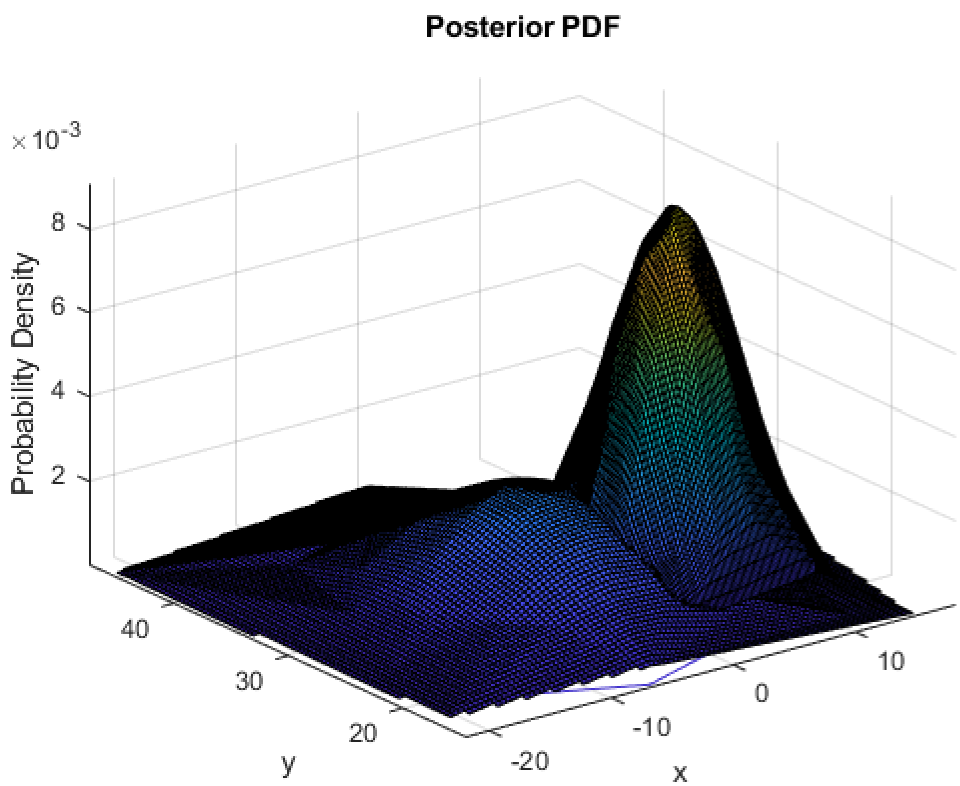
```
        qz = FF(qx,qy);
        figure;

        h = surfc(qx,qy,qz);
        xlabel('x');
        ylabel('y');
        zlabel('Probability Density');
        title('Prior PDF');
        hold on;
    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
        % TASK 4 - Plot the posterior pdf using surf and mvnpdf
        disp('Since we are looking at a random parameter estimation
problem, we are basically choosing between MAP and MMSE estimators. Both
provide good results but since we are using gaussian prior both MAP and MMSe
estimators will give the exact same answer. The results will only differ if
any other prior is used instead')
        X = mvnrnd(mean(1:2).',covar(1:2,1:2),100);
        Y = mvnpdf(X, mean(1:2).',covar(1:2,1:2));
        xx = X(:,1); yy = X(:,2); zz = Y;
```

```
        FF = TriScatteredInterp(xx,yy,zz);
        qx = linspace(min(xx),max(xx),100); % define your grid here
        qy = linspace(min(yy),max(yy),100);
        [qx,qy] = meshgrid(qx,qy);
        qz = FF(qx,qy);
        %figure;

        h = surfc(qx,qy,qz);
        xlabel('x');
        ylabel('y');
        zlabel('Probability Density');
        title('Posterior PDF');


    end

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

Since we are looking at a random parameter estimation problem, we are basically choosing between MAP and MMSE estimators. Both provide good results but since we are using gaussian prior both MAP and MMSe estimators will give the exact same answer. The results will only differ if any other prior is used instead




```
  end
end
```

```
time_step =

     1


% TASK 5 - Plot the true state, the measurements and state estimate
disp('State estimation helps us identify good idea of the sate given the
 measurements. Which measurements alone would not be sufficent to do so and
 will be very noisy due to noise, or other sources.')
figure;
plot(target_state_input_data.', 'DisplayName','True State Values');
xlabel('Time');
ylabel('State Values');
title('True Target States, Measurements, State Estimates (Process Noise -
 0.01)');
legend();

hold on;
time = 1:60;
scatter(time, measurements_input_data, 'x', 'DisplayName','Measurements');
hold on;
plot(estimate.', 'DisplayName','State Estimates');

disp(estimate(:,60));
% TASK 6 - Plot the Kalman filter gain
disp('The kalman gain weights the measurement innovation (difference between
 the measurement and the expected value of measurement), and helps compute
 the posterior mean and covariance. The graph shows the variance of the
 estimator and the decreasing graph shows us that we are approching better
 state estimates with each measurement')

figure;
plot(gain.');
xlabel('Time');
ylabel('Kalman Gain');
title('Kalman Gain v/s Time (Process Noise - 0.01)');
% TASK 7 - Plot the Kalman filter gain
%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 = 20; % 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];

sigmaX = 5; % Measurement error standard deviation in x
sigmaY = 5; % Measurement error standard deviation in y
R = [25 0 ; 0 25];
```

```
H = [1 0 0 0; 0 1 0 0];
load('data.mat');
input_data = load('data.mat');
measurements_input_data = input_data.measurements;
target_state_input_data = input_data.targetState;
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);
       % Perform the Kalman filter update and log the Kalman gain
       % additionally
       [mean,covar,gain(:,i)] = kalmanUpdate(priorMean,priorCovar,z,H,R);
       % Log the estimate
       estimate(:,i) = mean;
   end
end

figure;
plot(target_state_input_data.', 'DisplayName','True State Values');
xlabel('Time');
ylabel('State Values');
title('True Target States, Measurements, State Estimates (Process Noise-20)'
 );
legend();

hold on;
time = 1:60;
scatter(time, measurements_input_data, 'x', 'DisplayName','Measurements');
hold on;
plot(estimate.', 'DisplayName','State Estimates');

disp(estimate(:,60))

figure;
plot(gain.');
xlabel('Time');
ylabel('Kalman Gain');
title('Kalman Gain v/s Time (Process Noise - 20)');
disp('The track plot clearly shows that there is more noise in the state
 estimates when the process noise intensity is more. Also becoause of this,
 the kalman filter gain is converging to a higher value since the state
```

```
 estimates have inherently more noise,  and the Kalman Filter update is unable
 to reduce it below a certain limit.')
```

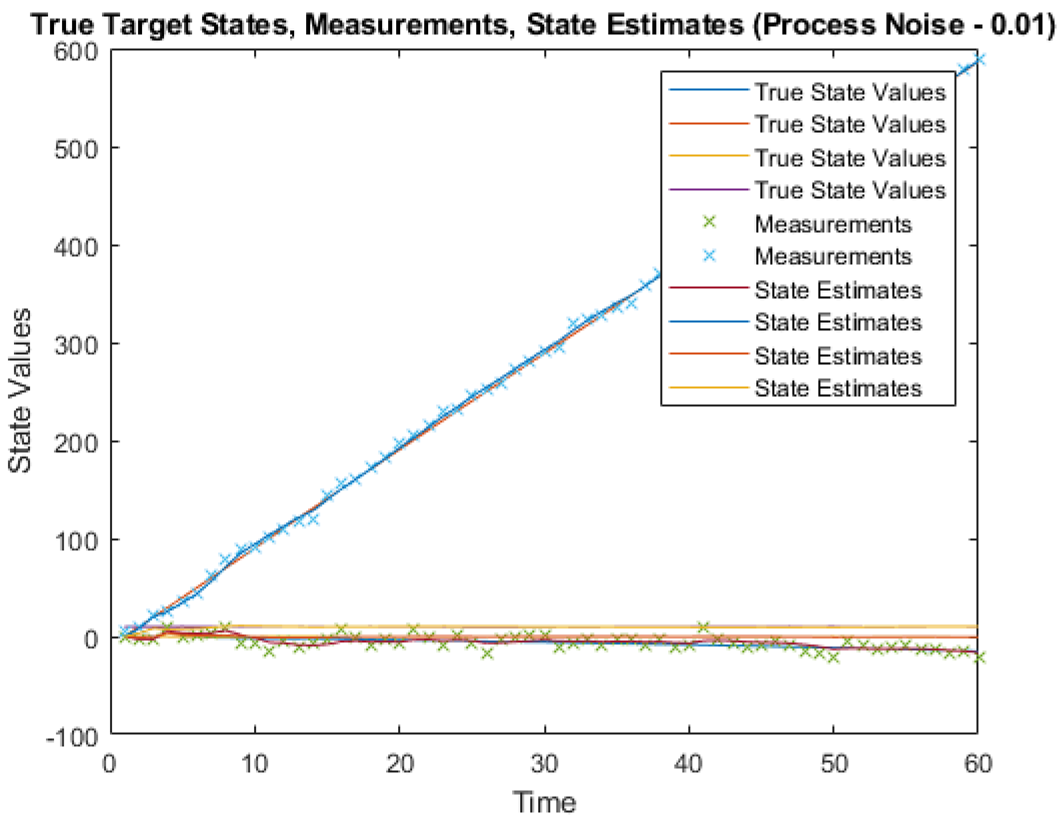
State estimation helps us identify good idea of the sate given the measurements. Which measurements alone would not be sufficent to do so and will be very noisy due to noise, or other sources.

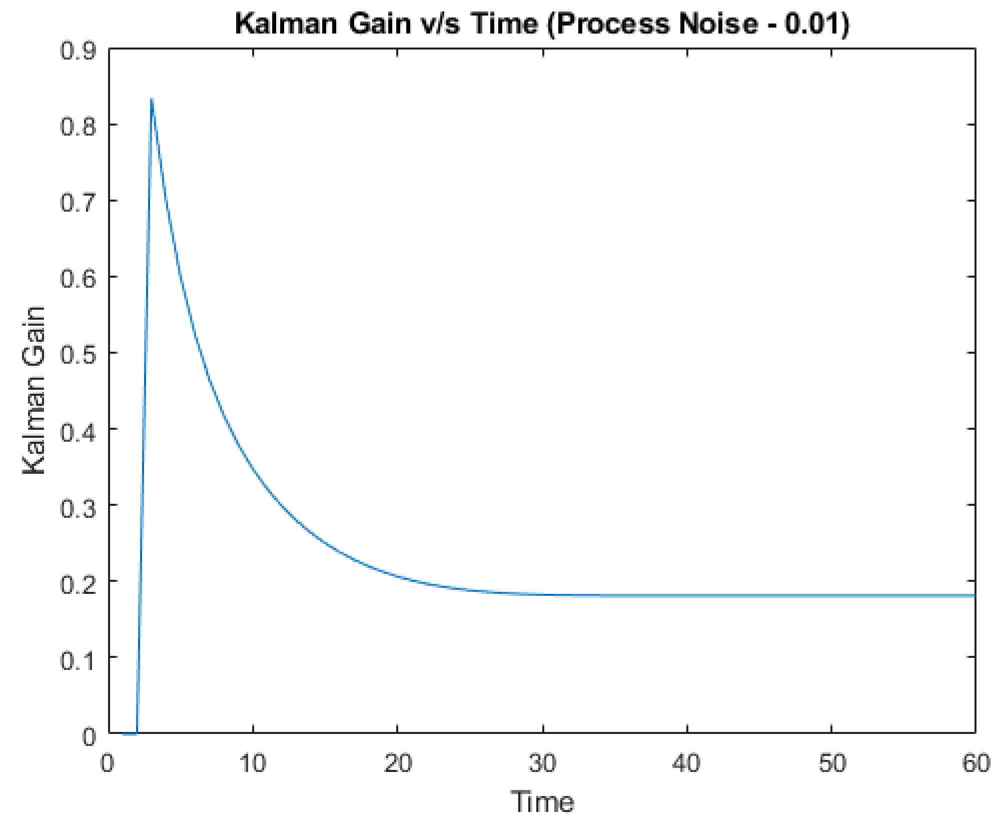
```
  -16.4385
  587.6455
   -0.5741
   10.2068
```

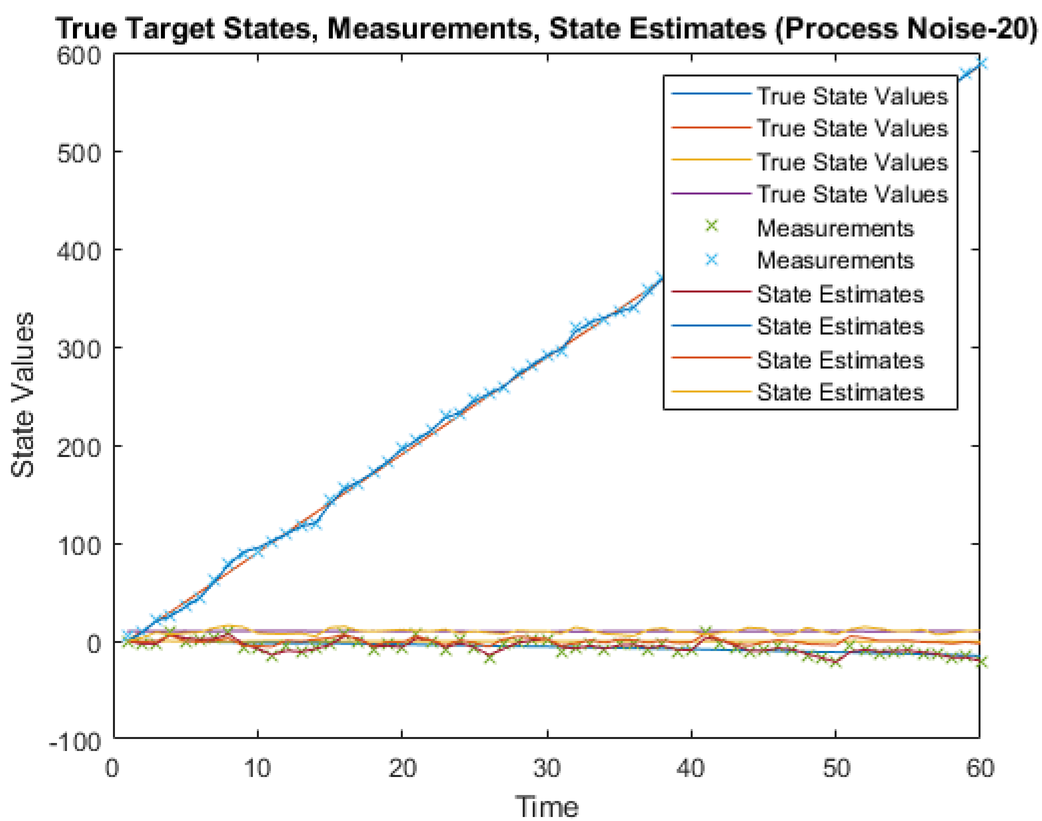
The kalman gain weights the measurement innovation (difference between the measurement and the expected value of measurement), and helps compute the posterior mean and covariance. The graph shows the variance of the estimator and the decreasing graph shows us that we are approching better state estimates with each measurement

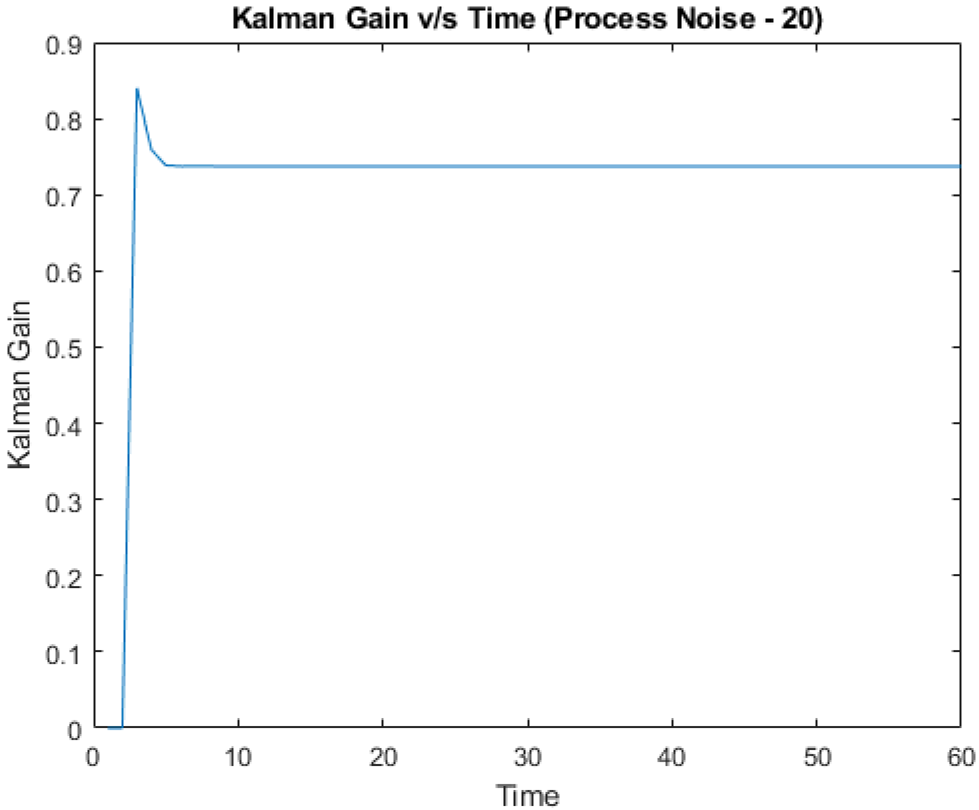
```
  -20.4568
  588.0735
   -2.6916
   10.8939
```

The track plot clearly shows that there is more noise in the state estimates when the process noise intensity is more. Also becoause of this, the kalman filter gain is converging to a higher value since the state estimates have inherently more noise, and the Kalman Filter update is unable to reduce it below a certain limit.




Kalman Gain v/s Time (Process Noise - 0.01)
Kalman Gain
Time
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
10
20
30
40
50
60


True Target States, Measurements, State Estimates (Process Noise-20)
State Values
Time
True State Values
True State Values
True State Values
True State Values
Measurements
Measurements
State Estimates
State Estimates
State Estimates
State Estimates
600
500
400
300
200
100
0
-100
10
20
30
40
50
60



Published with MATLAB® R2021b