Kalman Filter

A Kalman filter is an optimal estimator which is used to infer parameters of interest from indirect, inaccurate, and uncertain observations. Kalman filter is used for state estimation that is estimating the next state based on the previous state and the input vector which can be given or calculated. It works based on the given below equations. There are basically two steps state prediction and update.

$\mathbf{X_k = A * X_{k-1} + B*U_k + W_{k-1}}$

$\mathbf{Y_k = H * X_k + V_k}$

The random variables $\mathbf{W_{k-1}}$ and $\mathbf{V_k}$ represent process and measurement noise respectively, they are assumed to be independent of each other, white, and normally distributed with probabilities **p(W) = N(0, Q), p(V) = N(0, R).** The matrices **Q** and **R** are equal to **c*I** and **d*I** respectively where **c and d are scalers**.

Working of Kalman filter

- **Prediction**

  $X_k = A_{k-1}*X_{k-1} + B_k*U_k$

  $P_k = A_{k-1} * P_{k-1} * A^T_{k-1} + Q_{k-1}$
- **Updation**

  $V_k = Y_k - H, * X_k$

  $S_k = H_k * P_k * H^T_k + R_k$

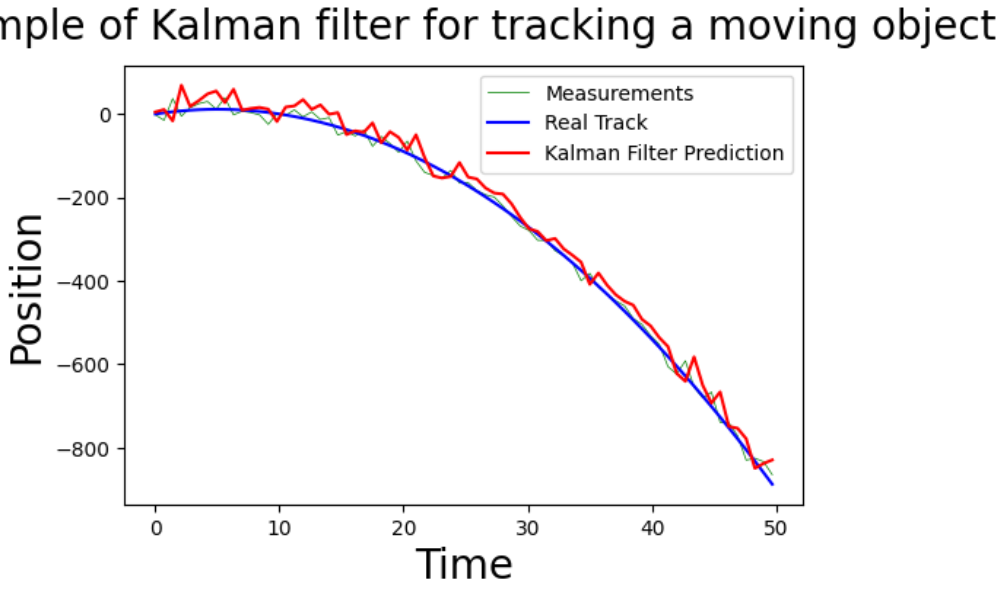
  $K_k = P_k * H^T_k * S^{-1}_k$

  $X_k = X_k + K_k*V_k$

  $P_k = P_k - K_k * S_k * K^T_k$

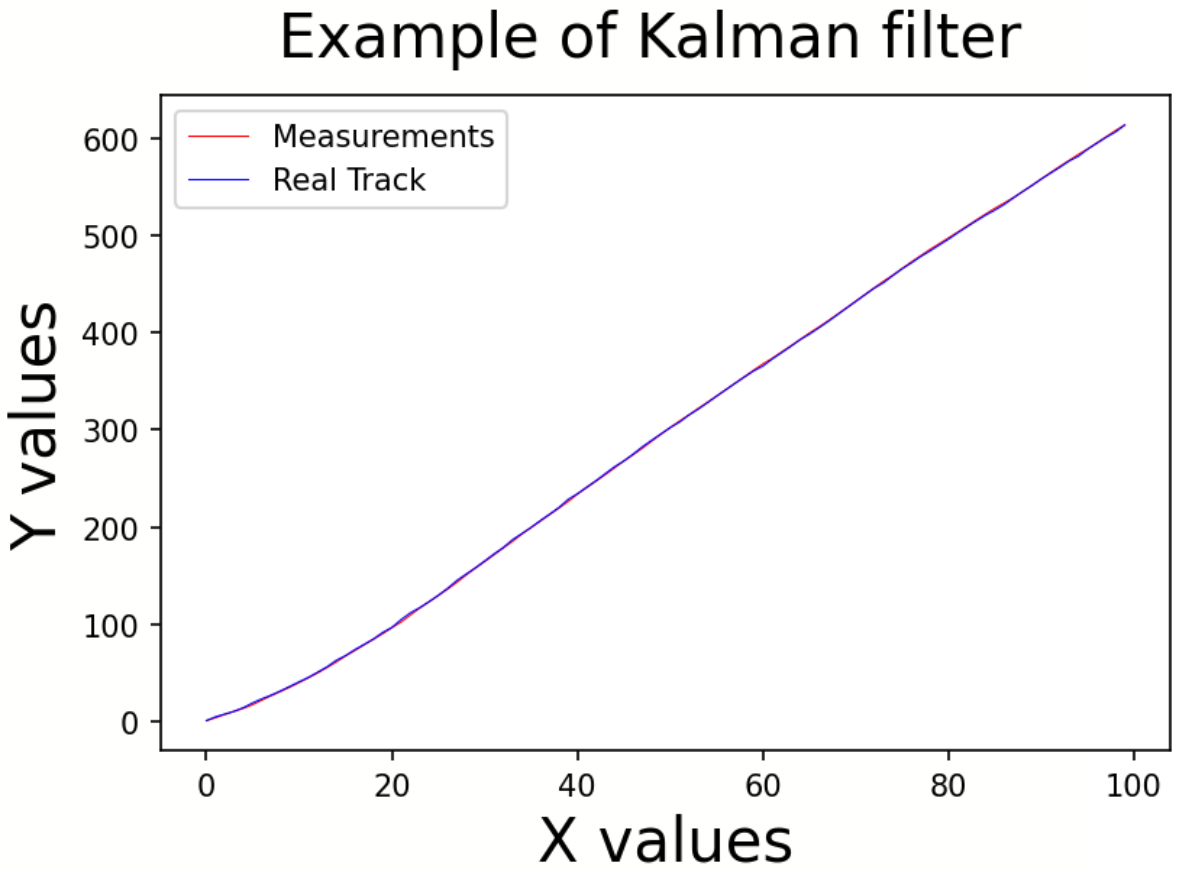
Implementing Kalman filter for 1D data (Track recognization)

- A = [[1, dt],[0, 1]]
- B = [0.5*(dt**2) , dt]
- H = [1, 0]
- Q = [[(dt**4)*0.25,(dt**3)*0.5],[(dt**3)*0.5, (dt**2)]] * (sigma_x**2)
- X is taken based on the equation of a track like parabola etc. Example y = 0.1*(10*x-x**2)
- The Kalman filter is run and output is noted.
- The value of the measurement is taken as X + nomal_random.

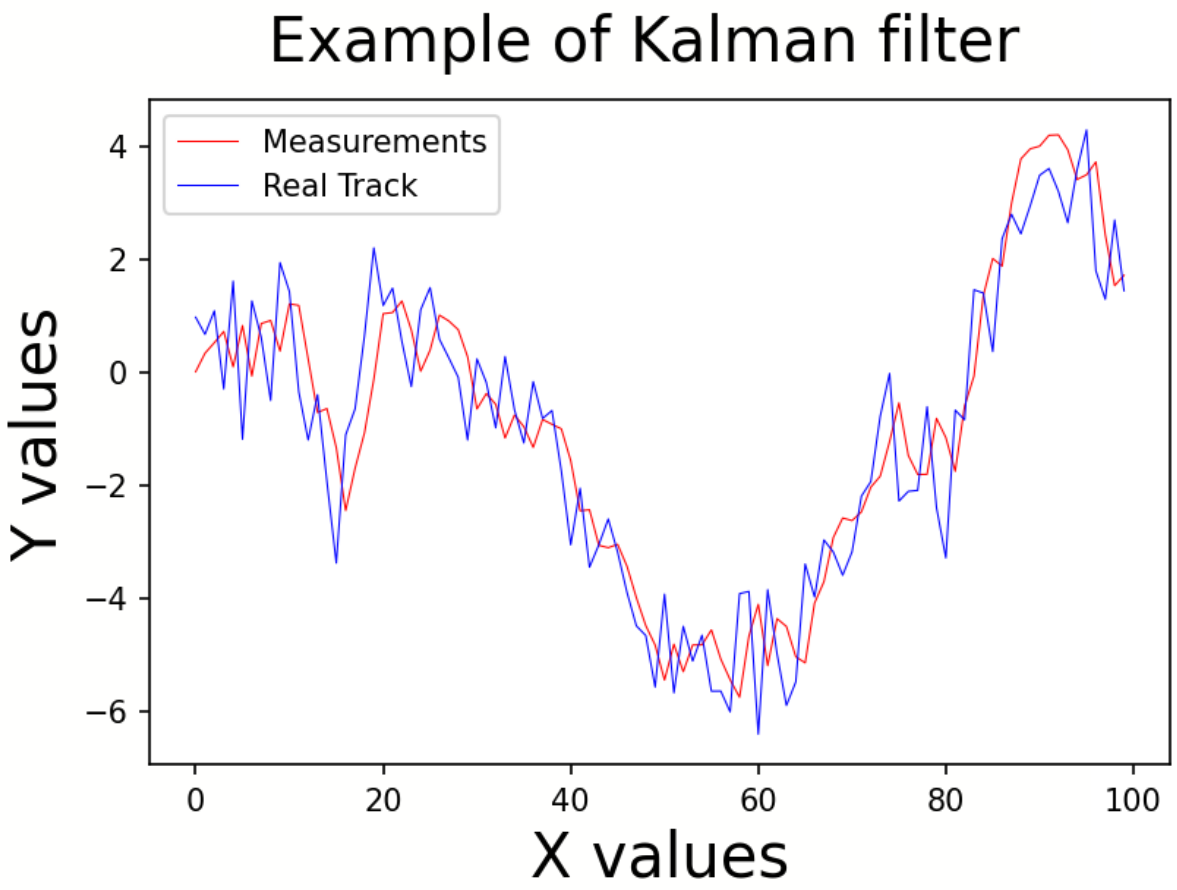


Implementing Kalman filter for 2D data:

- In the case of 2D the value of x' is also required with the value of x and thus the values of A, Q, R, and others change.
- X = [xk, yk, x'k, y'k]
- A = [[1, 0, dt, 0],[0, 1, 0, dt],[0, 0, 1, 0],[0, 0, 0, 1]]
- B = [[(dt**2)*0.5, 0],[0, (dt**2)*0.5],[dt ,0],[0, dt]]
- H = [[1, 0 ,0 ,0],[0, 1, 0, 0],[0, 0, 1, 0],[0, 0, 0, 1]]
- Q = [[(dt**4)/4, 0, (dt**3)/2, 0],[0, (dt**4)/4, 0, (dt**3)/2], [(dt**3)/2, 0, dt**2, 0],[0, (dt**3)/2, 0, dt**2]] * std_acc**2
- R = [[sigma_x**2, 0],[0, sigmay**2]]
- X values are calculated based on the measurement in which a simple increment is done in the value of x randomly and thus the values of y are calculated.
- Values of dt = 0.55, c = 3, d = 7, number of steps = 100.
- Example of Kalman filter when Y = X + abs(random(0,1))
- Here the dimension of x is 4 x 1 and that of y is 4 x 1.



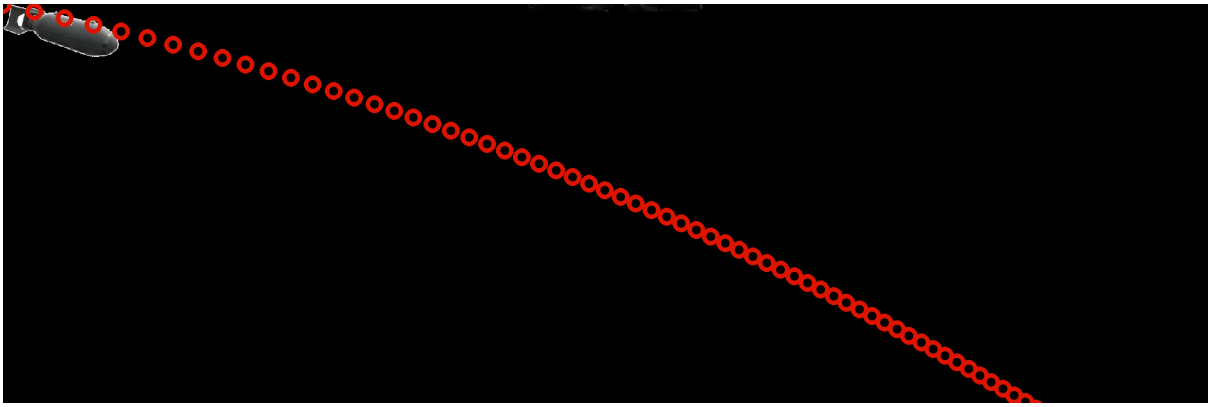
- Example of Kalman Filter when Y = X + random(0,1)



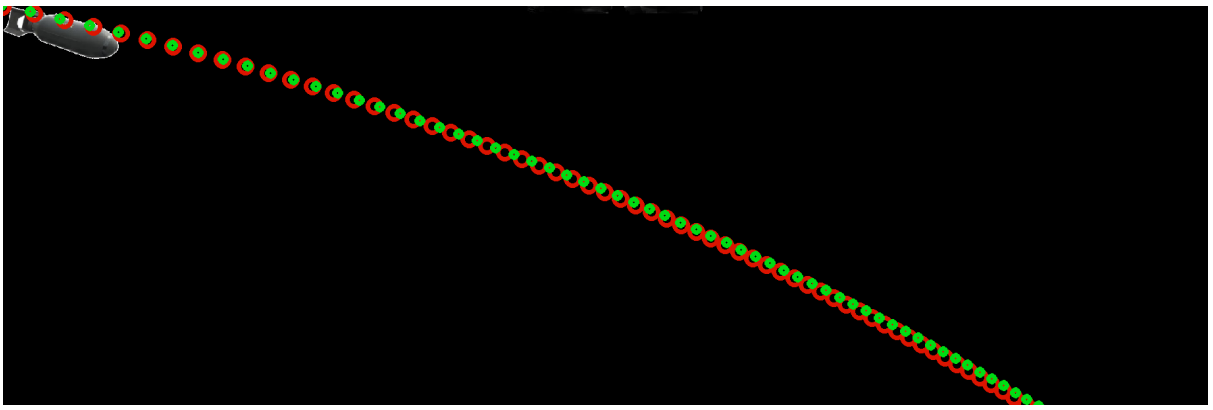
Implementing Kalman Filter for Projectile Tracking:

For projectile tracking the value of All the matrices are the same as that was used for 2-D tracking. The equation of the projectile is $y^2 = R^2 - (x - a)^2$ where R and a are constants. Values of Ux = 1, Uy = 1 , dt = 1 , sigma_x = 0.1, sigma_y = 0.1. In this the prediction is based on the actual values not the measurement.

The actual path is RED color



The Predicted path using the Kalman filter is GREEN color.



Observations and Results:

- While using the values of measurements for prediction of the next state it was observed that the filter was deviated by a significant margin than using the actual values for the prediction.
- While running the filter for small time steps less than 30, the value of prediction based on the measurement was more deviated than for larger time steps.
- When using the Kalman filter to predict the next state using the actual values rather than measurements the accuracy was about 100% as can be seen in the last example.
- When changing the values of c and d scalers for Q and R matrices the result was not deviated from than previous.
- For changing the dimensionality of state and input vector we need to change U, A, P, Q, and R also. For these two examples, one of 1D and the second of 2D is shown and their state space and other matrices are also defined.

Running the code:

- For running the code the **img.jpeg** file need to be uploaded.

References used:

- https://www.researchgate.net/publication/222095635_Implementation_of_Kalman_Filter_with_Python_Language
- https://machinelearningspace.com/2d-object-tracking-using-kalman-filter