<u>Localisation Task</u>

<u>Task 1:Kalman Filter</u>

The objective of the task is to implement the Kalman Filter Algorithm(KF) to estimate the robot’s location.
Before proceeding further, watch the first five videos of the Kalman Filter playlist by MATLAB(link in repo).
With the Kalman filter, we can combine the prediction from our motion model with the sensor measurement to get a better state estimate.

Remember the assumptions used in Kalman filter:

1. Linear dynamics with Gaussian noise
2. Linear measurement model with Gaussian noise
3. Gaussian prior

In the Kalman filter, there are two main steps that we need to perform. These are the prediction and update steps. We perform the prediction step to compute our new belief state after we apply a control signal *ut* to our robot following the motion model. After we apply a control signal, ut, we assume to have received a sensor measurement, zt, and we perform the update step to update the belief state based on the sensor measurement.

Since the model is gaussian, therefore the prediction will also be gaussian. Gaussian state is parameterised by mean and covariance so we need to estimate the new mean and covariance.
Mean determines where we believe the robot to be.
Covariance determines the uncertainty.

Prediction step:
The goal of the prediction is to predict the state of the robot based on the motion model. Done in two steps:

1. Compute the predicted mean

$$\mu_{t|t-1} = A\mu_{t-1|t-1} + B\mathbf{u}_{t-1}$$

where,
u=velocity vector, μ=mean position, A,B=motion model

$$u_t = [u_x , u_y]$$
$$\mu = [x \quad , y]$$

2. Compute the predicted covariance

$$\Sigma_{t|t-1} = A\Sigma_{t-1|t-1}A^T + Q$$

where,
Q=noise, Σ= covariance

$$\sum = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}$$

Update step:
The goal of the update step is to update using the prediction. Done in 3 steps:
1. Compute the mean and covariance of the prediction residual(or, the difference between the sensor measurement and our latest prediction)

$$\delta_\mu = \mathbf{z}_t - H\mu_{t|t-1}$$

$$\delta_\Sigma = H\Sigma_{t|t-1}H^T + R$$

R=measurement noise

2. Compute the Kalman gain.(Determines whether we trust the measurement the more or the prediction)

$$K_t = \Sigma_{t|t-1} H^T \delta_{\Sigma}^{-1}$$

K=Kalman Gain

3. Compute the updated mean and covariance.

$$\mu_{t|t} = \mu_{t|t-1} + K\delta_{\mu}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - KH\Sigma_{t|t-1}$$

Use the following,

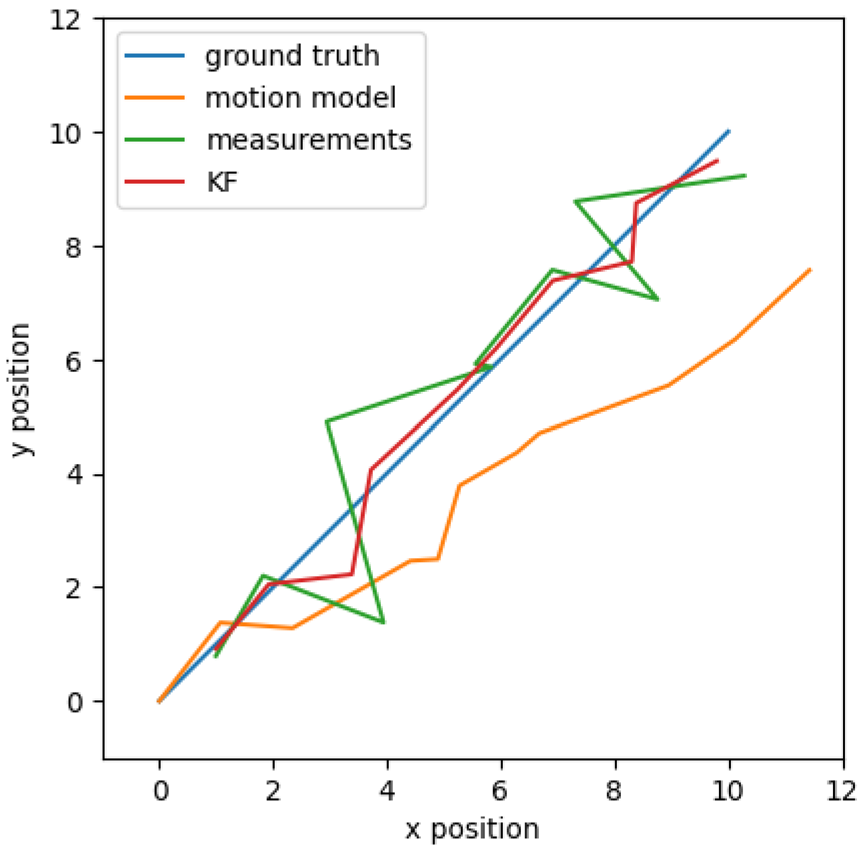
$$u_0 = [0, 0], \mu_0 = [0, 0]$$

$z_t = Hx_t + n_t$, where nt=random noise

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$Q = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}, H = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, R = \begin{bmatrix} 0.75 & 0 \\ 0 & 0.6 \end{bmatrix}$$

Now you have to code the KF algorithm in Python. You are only allowed to use NumPy, Matplotlib, and Math libraries in python for all the mathematical operations. Given the above initial conditions, you have to code the algorithm for a robot to estimate where the robot is at each time step!.



This is how the plot would look if I were to plot x=[0,10] and y=[0,10] with a step size =1(i.e. ux=1,uy=1).
Do the same and try playing around with the code and see what happens!

Happy filtering!