CODE

CODE

import numpy as np

import matplotlib.pyplot as plt

dox=0.7#started with vague assumptions

doy=0.7#these values were finalised after some fine-tuning

dmx=0.2

dmy=0.2

dovx=5

dovy=0.45

dmvx=5

dmvy=1

lx=[]#stores x and y

ly=[]

lvx=[]#stores velocity in x and y

lvy=[]

lpx=[]#stores predicted x and y

lpy=[]

step=1

ox=0

oy=0

dt=1

A=np.array([[1,0,dt,0],[0,1,0,dt],[0,0,1,0],[0,0,0,1]]) #array to update

X=np.zeros((4,4))

P=np.zeros((4,4))

R=np.zeros((4,4))

Y=np.zeros((4,1))

def set(x,y):#initialize all matrices

global X

global P

global R

global a

global b

a=x

b=y

X[0][0]=x

X[1][0]=y

X[2][0]=0#initial velocities taken 0

X[3][0]=0

R = np.array([[dox \* dox, doy \* dox, dox \* dovx, dox \* dovy], [doy \* dox, doy \* doy, doy \* dovx, doy \* dovy],

[dox \* dovx, doy \* dovx, dovx \* dovx, dovy \* dovx],

[dox \* dovy, doy \* dovy, dovx \* dovy, dovy \* dovy]], dtype=float)#Contains observation errors

#R remains constant

P = np.array([[dmx \* dmx, dmx \* dmy, dmx \* dmvx, dmx \* dmvy], [dmx \* dmy, dmy \* dmy, dmy \* dmvx, dmy \* dmvy],

[dmvx \* dmx, dmvx \* dmy, dmvx \* dmvx, dmvx \* dmvy],

[dmvy \* dmx, dmvy \* dmy, dmvy \* dmvx, dmvy \* dmvy]])#initialize covariance matrix(measurement)

#initialise all matrices

a=0

b=0

def kal(x,y,vx,vy):

global X

global Y

global P

global R

global A

global a

global b

X[0][0]=x #update x

X[1][0]=y #update y

P=np.matmul(A,np.matmul(P,A.transpose())) #find covariance matrix(measurement)

K = np.zeros((4, 4))

K=np.divide(P,np.add(P,R)) #find Kalman gain

a=a+vx #sum of vx till current step

b=b+vy #sum of vy till current step

Y=[[a],[b],[vx],[vy]] #measured state

X=np.add(X,np.multiply(K,np.subtract(Y,X))) #get predicted x and y

lpx.append(X[0][0]) #update lists

lpy.append(X[1][0])

print("x=",X[0][0]," y=",X[1][0]," uncertainity in x=", P[0][0]," uncertainity in y=", P[1][1]) #print required values

I=np.identity(4)

P=(I-K)\*P #update the covariance matrix(measurement)

def plotg():#plot graphs

fig,((ax1,ax2), (ax3,ax4))=plt.subplots(2,2)

ax1.plot(lx,label="x",color="cyan")

ax1.plot(ly, label="y", color="grey")

ax1.legend()

ax2.plot(lpx, label="predicted x",color="blue")

ax2.plot(lpy, label="predicted y", color="black")

ax2.legend()

ax3.plot(lpx, lpy, label="predicted ellipse", color="red")

ax3.plot(lx,ly, label="path",color="orange")

ax3.legend()

ax4.plot(lpx,lpy, label="predicted path",color="red")

ax4.legend()

plt.show()

def main():

given\_file = open('Resources/Kalman.txt', 'r')

lines = given\_file.readlines()#code to extract numbers one by on

#and divide them into x,y,vx,vy

d=0

chk=0

i=0

nch=0

count=-2

step=1

for line in lines:

for c in line:

if(c==' '):

continue

if(c=='-'):

nch=1

elif(c==',' or c=='\n'):

count = count + 1

if(nch==1):

d=-d

if (count == -1):

x=d

elif (count == 0):

y=d

set(x,y)

elif(count%4==1):

x=d

lx.append(d)

elif (count % 4 == 2):

y = d

ly.append(d)

elif (count % 4 == 3):

vx = d

lvx.append(d)

elif (count % 4 == 0):

vy = d

lvy.append(d)

step=step+1

kal(x, y, vx, vy)

d=0

chk=0

i=0

nch=0

elif(c=='.'):

chk=1

elif(chk==0):

d=d\*10+ord(c)-48

else:

i=i+1

d=d+(ord(c)-48)\*pow(0.1,i)

#print(ly)

#print(lpy)

#print(lx)

#print(lpx)

plotg()#plot

if \_\_name\_\_ == "\_\_main\_\_":

# execute main

main()