Implementation of Kalman Filter

AIM

To Construct a Python Code to implement the Kalman filter to predict the position and velocity of an object.

ALGORITHM

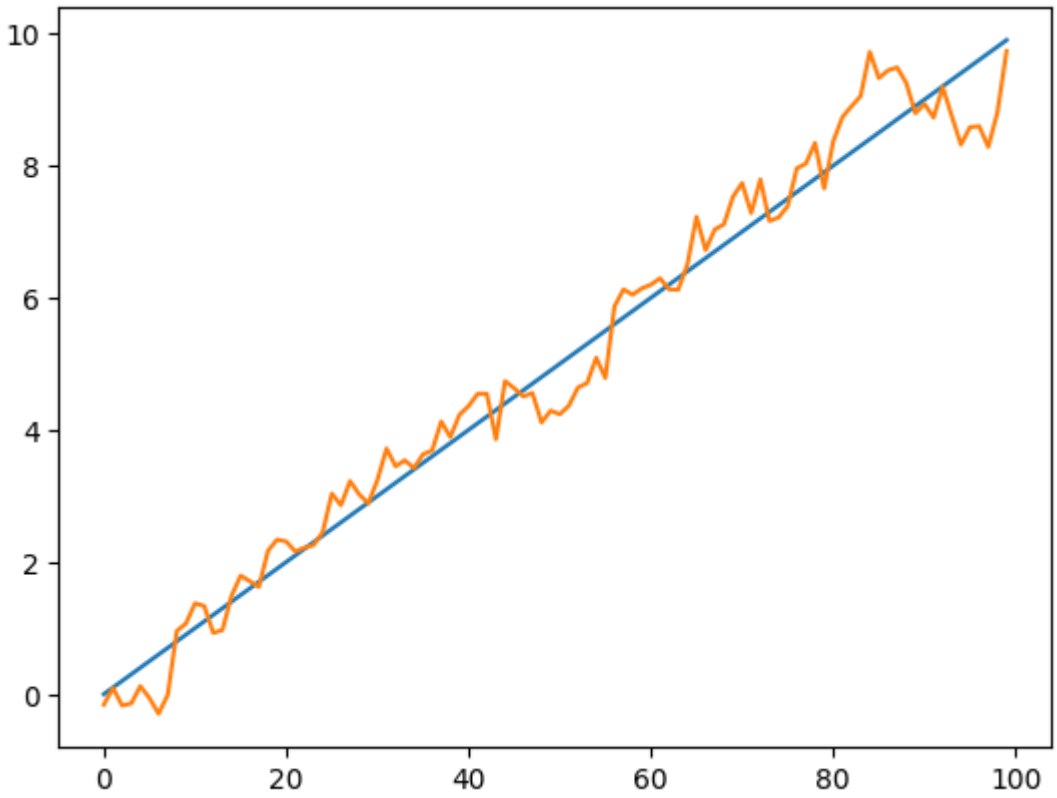
- Step 1: Define the state transition model F, the observation model H, the process noise covariance Q, the measurement noise covariance R, the initial state estimate x0, and the initial error covariance P0.
- Step 2: Create a KalmanFilter object with these parameters.
- Step 3: Simulate the movement of the object for a number of time steps, generating true states and measurements.
- Step 4: For each measurement, predict the next state using kf.predict().
- Step 5: Update the state estimate based on the measurement using kf.update().
- Step 6: Store the estimated state in a list.
- Step 7: Plot the true and estimated positions.

PROGRAM

```
import numpy as np

class KalmanFilter:
  def __init__(self,F,H,Q,R,x0,P0):
    self.F=F
    self.H=H
    self.Q=Q
    self.R=R
    self.X=x0
    self.P=P0

  def predict(self):
    self.X=np.dot(self.F,self.X)
    self.P=np.dot(np.dot(self.F,self.P),self.F.T)+self.Q

  def update(self,z):
    y=z-np.dot(self.H,self.X)
    S=np.dot(np.dot(self.H,self.P),self.H.T)+self.R
    K=np.dot(np.dot(self.P,self.H.T),np.linalg.inv(S))
    self.X=self.X+np.dot(K,y)
    self.P=np.dot(np.eye(self.F.shape[0])-np.dot(K,self.H),self.P)

dt=0.1
F=np.array([[1,dt],[0,1]])
H=np.array([[1,0]])
Q=np.diag([0.1,0.1])
R=np.array([[1]])
x0=np.array([0,0])
P0=np.diag([1,1])

kf=KalmanFilter(F,H,Q,R,x0,P0)

true_states=[]
measurements=[]
for i in range(100):
  true_states.append([i*dt,1])
  measurements.append(i*dt+np.random.normal(scale=1))

est_states=[]
for z in measurements:
  kf.predict()
  kf.update(np.array([z]))
  est_states.append(kf.X)

import matplotlib.pyplot as plt
plt.plot([s[0] for s in true_states],label='true')
```

```
plt.plot([s[0] for s in est_states],label='estimate')
plt.legend
plt.show()
```

OUTPUT



RESULT

Thus, Kalman filter to predict the position and velocity of an object is implemented successfully.