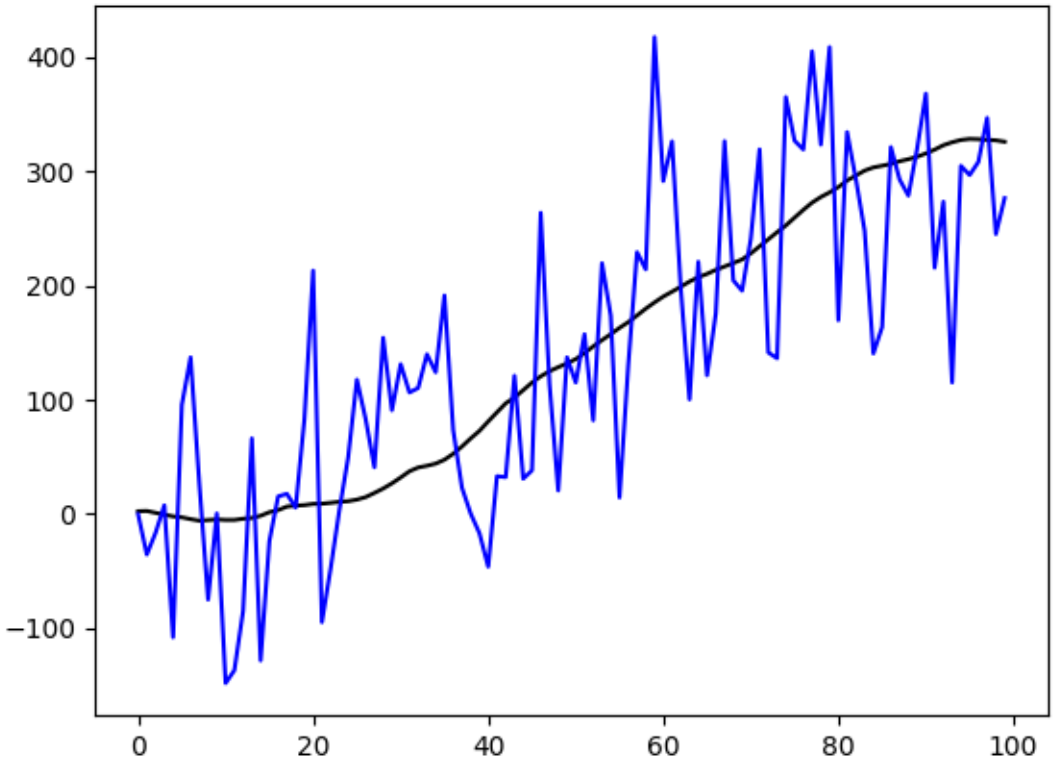
ואדים ליטבינוב

314552365

הגרף הנדרש:



כאשר הקו השחור זה המיקום האמיתי של החלקיק, והקו הכחול אלה ההערכות של מסנן קלמן.

שגיאת המדידה הממוצעת הייתה 10,708.2 לעומת שגיאה ממוצעת של 7,286.9 של מסנן קלמן.

היחס ביניהם הוא 1.47.

<u>הקוד:</u>

```
# Vadim Litvinov

import matplotlib.pyplot as plt

import numpy as np


# variance of x measurements
```

```
R = 100
# variance of the noise of particle location (after multiplication it only changes the speed)
Q = np.array([[0, 0],
[0, 1]])
# matrix that will give us the new location and new velocity
A = np.array([[1, 1],
[0, 0.98]])
# matrix that concerns only location and zeros the velocity
C = np.array([1, 0])
SAMPLES_NUM = 100


def generateParticles():
# generate process noises from N(0,1)
noises = np.random.randn(2, SAMPLES_NUM)
# multiply and leave only velocity noises (location has no noise v)
v = np.matmul(Q, noises)
del noises
# generate measurement noise from N(0,R)
w = np.random.randn(SAMPLES_NUM) * R
# initialization of arrays of our z and predicted locations
x = np.zeros((2, SAMPLES_NUM))
y = np.zeros(SAMPLES_NUM)
# the first x is taken from N(0,1)
x[0:2, 0] = np.random.randn(2, 1).transpose()
# now generate all other with respect to parameters
for i in range(1, SAMPLES_NUM):
# generate true process
x[:, i] = np.matmul(A, x[:, i-1]) + v[:, i]
# generate measurements
```

```
        y[i] = np.matmul(C, x[:, i]) + w[i]
    return y, x


def kalmanFilter(y):
    x_pred = np.zeros((2, 100))
    x_pred[:, 0] = np.array([y[0], 0]).T
    identity = np.ones((2, 2))
    xt_t1 = np.matmul(A, x_pred[:, 0])
    Pt_t1 = identity

    for t in range(100):
        Kt = Pt_t1 @ C.T / (C @ Pt_t1 @ C.T + R)
        print(Kt)
        Pt_t = (identity - Kt @ C) @ Pt_t1 @ (identity - Kt @ C) + R * Kt @ Kt.T
        print(Pt_t)
        xt_t = xt_t1 + Kt * (y[t] - C @ xt_t1)
        print(xt_t)
        x_pred[:, t] = xt_t
        Pt1_t1 = Pt_t
        xt1_t1 = xt_t
        Pt_t1 = A @ Pt1_t1 @ A.T + Q
        xt_t1 = np.matmul(A, xt1_t1)
    return x_pred


def plot(x_pred, x):
    plt.figure()
    plt.plot(range(0, x_pred.shape[1]), x[0, :], 'k', label='True location')
```

```
plt.plot(range(0, x_pred.shape[1]), x_pred[0, :], '-b', label='KF location')
plt.show()


if __name__ == '__main__':
    y, x = generateParticles()
    x_pred = kalmanFilter(y)
    plot(x_pred, x)

    # now we shall calculate the errors
    y_err = np.sum((y - x[0, :]) ** 2) / x_pred.shape[1]
    pred_err = np.sum((x_pred[0, :] - x[0, :]) ** 2) / x_pred.shape[1]
    # print the errors
    print(y_err, pred_err)
    # print the error ratio
    print(y_err / pred_err)
```