```
#Assignment 2
#Comparison of the exponential and running mean for random
#Team 12
#Yaroslav Savotin, Elizaveta Pestova, Selamawit Asfaw
#Skoltech, 2023

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

The objective of this laboratory work is to compare the errors of exponential and running mean to choose the most effective quasi-optimal estimation method in conditions of uncertainty.

This laboratory work consists of two parts: I. Determination of optimal smoothing constant in exponential mean. II. Comparison of methodical errors of exponential and running mean.

Part I: Determination of optimal smoothing constant in exponential mean

1. Please conduct a simulation experiment. First generate a true trajectory, then generate measurements of this true trajectory. 1.1. Generate a true trajectory using the random walk model

$$X_i = X_{i-1} + w_i$$

$$\text{Group 12: } \sigma_w^2 = 19$$

Size of trajectory is (please consider 2 cases)

1. 3000 points
2. 300 points To generate true trajectory use initial condition X1 = 10.

```
#Part 1
#------------№1.1------------
n1 = 3000
n2 = 300
X1 = [0 for _ in range(n1)]
X2 = [0 for _ in range(n2)]
X1[0] = 10
X2[0] = 10

# normally distributed random noise with zero mathematical expectation and variance
mu, sigma1 = 0, 19
w1 = np.random.normal(mu, np.sqrt(sigma1), [n1,1])
w2 = np.random.normal(mu, np.sqrt(sigma1), [n2,1])


for i in range(n1-1):
    X1[i+1] = X1[i] + w1[i+1]

for i in range(n2-1):
    X2[i+1] = X2[i] + w2[i+1]
```

1.2. Generate measurements z_i of the process X_i

$$z_i = X_i + \eta_i \quad (2)$$

η_i – normally distributed random noise with zero mathematical expectation and variance σ_η^2.

```
#------------№1.2------------
#
Z1 = [0 for _ in range(n1)]
Z2 = [0 for _ in range(n2)]

# normally distributed random noise with zero mathematical expectation and variance
mu, sigma2 = 0, 10
nu1 = np.random.normal(mu, np.sqrt(sigma2), [n1,1])
nu2 = np.random.normal(mu, np.sqrt(sigma2), [n2,1])


for i in range(n1):
    Z1[i] = X1[i] + nu1[i]

for i in range(n2):
    Z2[i] = X2[i] + nu2[i]
```

2. Identify σ_w^2 and σ_η^2 using identification method presented on slide 55 (Topic_2_Quasi-optimal approximation under uncertainty.pdf). Perform identification for different size of trajectory (3000 and 300). Compare estimation results with true values of σ_w^2 and σ_η^2. Compare the accuracy of estimation.

Process X_i	$X_i = X_{i-1} + w_i$	(1)
Measurements	$z_i = X_i + \eta_i$	(2)
Residual $\boldsymbol{v_i}$	$v_i = z_i - z_{i-1}$	(3)
Residual ρ_i	$\rho_i = z_i - z_{i-2}$	(4)
Residual $\boldsymbol{v_i}$	$v_i = w_i + \eta_i - \eta_{i-1}$	(5)
Residual ρ_i	$\rho_i = w_i + w_{i-1} + \eta_i$	(6)
Math. expectation	$E[v_i^2] = \sigma_w^2 + 2\sigma_\eta^2$	(7)
Math. expectation	$E[\rho_i^2] = 2\sigma_w^2 + 2\sigma_\eta^2$	(8)

```
#------------№2------------
v1 = [0 for _ in range(n1)]
v2 = [0 for _ in range(n2)]
p1 = [0 for _ in range(n1)]
p2 = [0 for _ in range(n2)]

v1[0] = Z1[0]
v2[0] = Z2[0]
p1[0] = Z1[0]
p2[0] = Z2[0]
p1[1] = Z1[1]
p2[1] = Z2[1]

#Residual v
for i in range(1,n1):
    v1[i] = Z1[i] - Z1[i-1]

for i in range(1,n2):
    v2[i] = Z2[i] - Z2[i-1]

#Residual p
for i in range(2,n1):
    p1[i] = Z1[i] - Z1[i-2]

for i in range(2,n2):
    p2[i] = Z2[i] - Z2[i-2]
```

```
#for 3000-size dataset
Ev1 = 0
Ep1 = 0
for i in range(2,n1):
    Ev1 += (v1[i])**2
Ev1 = Ev1/(n1-1)

for i in range(3,n1):
    Ep1 += (p1[i])**2
Ep1 = Ep1/(n1-2)

sigmaW1 = Ep1 - Ev1
sigmaN1 = (Ev1 - sigmaW1)/2
print('sigmaW1 =', sigmaW1)
print('sigmaN1 =', sigmaN1)
```

```
sigmaW1 = [19.78057613]
sigmaN1 = [10.24762868]
```

```
#for 300-size dataset
Ev2 = 0
Ep2 = 0
for i in range(2,n2):
    Ev2 += (v2[i])**2
Ev2 = Ev2/(n2-1)

for i in range(3,n2):
    Ep2 += (p2[i])**2
Ep2 = Ep2/(n2-2)

sigmaW2 = Ep2 - Ev2
sigmaN2 = (Ev2 - sigmaW2)/2
print('sigmaW2 =', sigmaW2)
print('sigmaN2 =', sigmaN2)
```

```
sigmaW2 = [20.11036696]
sigmaN2 = [12.21255286]
```

```
# Comparison

#3000
print('N = 3000')
print('Error of consistent estimate sigmaW1 = ',np.abs(sigmaW1 - sigma1))
print('Error of consistent estimate sigmaN1 = ',np.abs(sigmaN1 - sigma2))
print()

#300
print('N = 300')
print('Error of consistent estimate sigmaW1 = ',np.abs(sigmaW2 - sigma1))
print('Error of consistent estimate sigmaN1 = ',np.abs(sigmaN2 - sigma2))
```

```
N = 3000
Error of consistent estimate sigmaW1 =  [0.78057613]
Error of consistent estimate sigmaN1 =  [0.24762868]

N = 300
Error of consistent estimate sigmaW1 =  [1.11036696]
Error of consistent estimate sigmaN1 =  [2.21255286]
```

Conclusion:

The estimated variance which we have identified is much closer to the actual value which tells us the accuracy of estimation.

There fore from this identification we conclude that when there is more steps our estimated variance is more closer to the actual.

3. Determine optimal smoothing coefficient in exponential smoothing

$$\alpha = \frac{-\chi+\sqrt{\chi^2+4\chi}}{2}$$

$$\chi = \frac{\sigma_w^2}{\sigma_\eta^2}$$

```
#------------№3------------
ksi = sigmaW1/sigmaN1
alfa = (-ksi + np.sqrt(ksi**2 + 4*ksi))/2
print('alfa =', alfa)
```

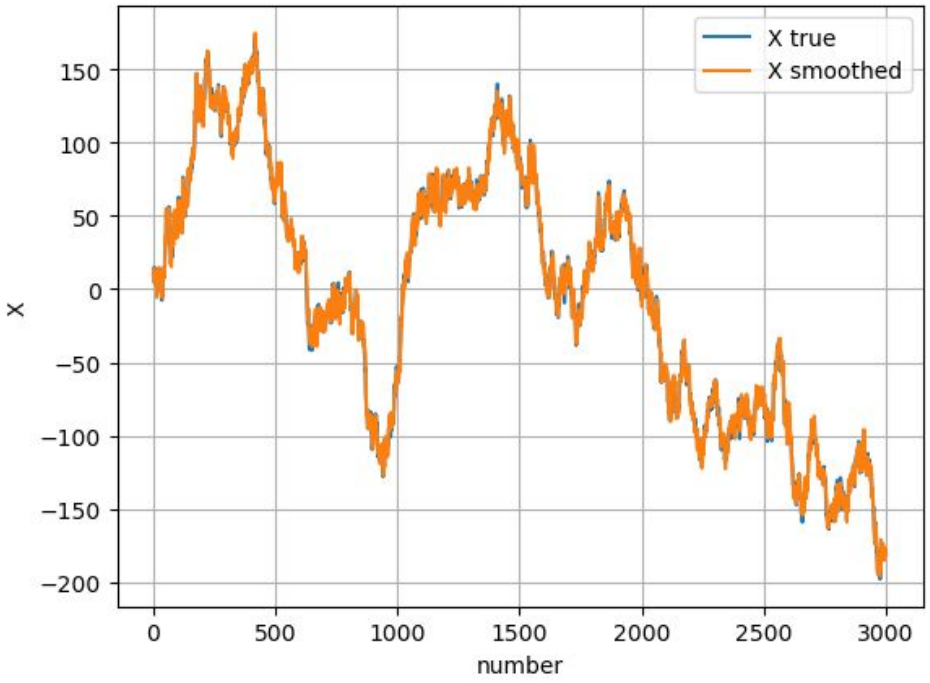
```
alfa = [0.72653652]
```

4. Perform exponential smoothing with the determined smoothing coefficient. Plot results. For the comparison add measurements, true values of process and exponentially smoothed data.

```
# exponential smoothing with calculated alfa
Xsm = [0 for _ in range(n1)]
Xsm[0] = X1[0]
for i in range(1,n1):
    Xsm[i] = Xsm[i-1] + alfa*(Z1[i] - Xsm[i-1])
```

```
#preparation for plotting
k = [1 for _ in range(n1)]
for i in range(n1-1):
    k[i+1] = k[i] + 1
for i in range(1,n1):
    Xsm[i] = Xsm[i][0]
for i in range(1,n1):
    X1[i] = X1[i][0]
#create the plot for comparison
y1 = X1
y2 = Xsm
x = k
plt.plot(x,y1,label = 'X true')
plt.plot(x,y2,label = 'X smoothed')
plt.xlabel('number')
plt.ylabel('X')
plt.suptitle('Comparison of true values of process and exponentially smoothed data.')
plt.legend()
plt.grid(True)
```

Comparison of true values of process and exponentially smoothed data.




```
#Conclusion: as the main parametr of smoothing - alpha is predetermined,
#the quality of smoothing is very low as smoothed line almost repeats the line of real measurments
```

Here is the recommended procedure for part II:
Comparison of methodical errors of exponential and running mean.

1. Generate a true trajectory X_i using the random walk model (1).
   Size of trajectory is 300 points.
   Initial condition $X_1 = 10$.
   Variance of noise w_i, $\sigma_w^2 = 28^2$

2. Generate measurements z_i of the process X_i using equation (2)
   Variance of noise measurement noise η_i, $\sigma_\eta^2 = 97^2$

3. Determine optimal smoothing coefficient α using equation (3).
   (There is no need to identify it again, just use equation for α from part I with given σ_w^2 and σ_η^2).

4. The component of full error that is related to measurement errors is determined as
   (from slide 37, Topic_2_Quasi-optimal approximation under uncertainty.pdf)

Running mean (RM):

$$\sigma_{RM}^2 = \frac{\sigma_\eta^2}{M} \quad (4)$$

Exponential smoothing (ES):

$$\sigma_{ES}^2 = \sigma_\eta^2 \frac{\alpha}{2-\alpha} \quad (5)$$

Determine the window size M (use round values) that provides equality of σ_{RM}^2 and σ_{ES}^2 using determined smoothing constant α

$$\sigma_{ES}^2 = \sigma_{RM}^2$$

5. Apply running mean using determined window size M and exponential mean (see, for instance, page 30, Topic_2_Quasi-optimal approximation under uncertainty.pdf) using determined smoothing constant α to measurements z_i. Plot true trajectory X_i, measurements z_i, running and exponential mean.

6. Make visual comparison of results. Make conclusions which methods give greater

```
#Part 2
#------------№1------------
n2 = 300
X2 = [0 for _ in range(n2)]
X2[0] = 10

# normally distributed random noise with zero mathematical expectation and variance
mu, sigma1 = 0, 28**2
w2 = np.random.normal(mu, np.sqrt(sigma1), [n2,1])

for i in range(n2-1):
    X2[i+1] = X2[i] + w2[i+1]

#------------№2------------
Z2 = [0 for _ in range(n2)]

# normally distributed random noise with zero mathematical expectation and variance
mu, sigma2 = 0, 97**2
nu2 = np.random.normal(mu, np.sqrt(sigma2), [n2,1])

for i in range(n2):
    Z2[i] = X2[i] + nu2[i]

#------------№3------------
ksi = sigma1/sigma2
alfa = (-ksi + np.sqrt(ksi**2 + 4*ksi))/2
alfa = round(alfa, 2)
print('alfa =', alfa)
```

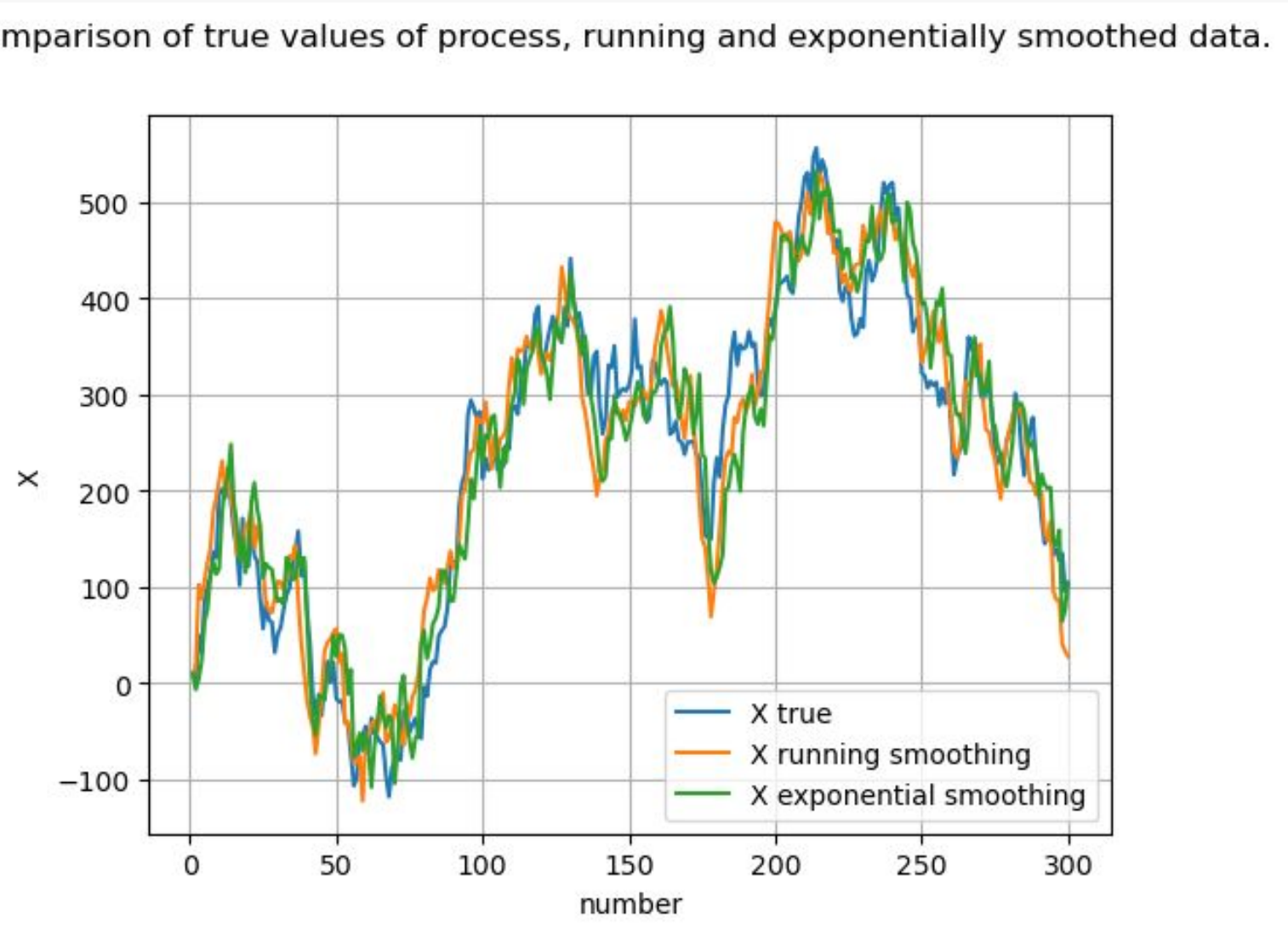
```
alfa = 0.25
```

```
#------------№4------------
M = int((2-alfa)/alfa)
sigmaRM = sigma2/M
sigmaES = (sigma2*alfa)/(2-alfa)
print('sigmaRM**2 =', sigmaRM)
print('sigmaES**2 =', sigmaES)
print('window size M =', M)
```

```
sigmaRM**2 = 1344.142857142857
sigmaES**2 = 1344.142857142857
window size M = 7
```

```
#------------№5------------
#apply running mean using M and exponential mean using alfa

R = [0 for _ in range(n2)]
sum = 0
r = int((M-1)/2) #3

for j in range(3,n2-3):
    for i in range(r+1):
        if i == 0:
            sum += Z2[j-i]
        else:
            sum += Z2[j-i] + Z2[j+i]
    R[j] = sum/M
    sum = 0

for j in range(r):
    for i in range(r):
        sum += Z2[j+i]
    R[j] = sum/r
    sum = 0

for j in range(n2-r,n2):
    for i in range(r):
        sum += Z2[j-i]
    R[j] = sum/r
    sum = 0

#Exponential mean
Xsm = [0 for _ in range(n2)]
Xsm[0] = X2[0]
for i in range(1,n2):
    Xsm[i] = Xsm[i-1] + alfa*(Z2[i] - Xsm[i-1])
```

```
#preparation for plotting
k = [1 for _ in range(n2)]
for i in range(n2-1):
    k[i+1] = k[i] + 1
for i in range(1,n2):
    Xsm[i] = Xsm[i][0]
for i in range(1,n2):
    X2[i] = X2[i][0]
for i in range(0,n2):
    R[i] = R[i][0]

#plot
y1 = X2
y2 = R
y3 = Xsm
x = k
plt.plot(x,y1,label = 'X true')
plt.plot(x,y2,label = 'X running smoothing')
plt.plot(x,y3,label = 'X exponential smoothing')
plt.xlabel('number')
plt.ylabel('X')
plt.suptitle('Comparison of true values of process, running and exponentially smoothed data.')
plt.legend()
plt.grid(True)
```




```
#Conclusion: this plot nicely depicts small, but existing error between real measurements and postprocessed data
```

Conclusion

Visualizing from the sample of comparison of results in the above figure we can conclude that at the same down shoot point. we have already applied running and exponential mean (which is actually forward exponential mean) methods to random walk model with noise statistics

In these conditions it can be truly seen that the results of exponential smoothing demonstrated significant shift (delay) of estimations.Bases on visual analysis, there is a tendency of shifting in Forward Exponential Smoothing method.

The running mean method has much less shifting error. By calculating variances of both methods we obtained exact result: running mean performs more accurately.

From first part we noticed very important thing: exponential smoothing performs better for larger datasets, keeping other parameters the same.

In this lab, we learned about two methods of data processing: running mean and exponential smoothing. The second method has an advantage over the first: the averaging takes into account all previous points with different weights varying exponentially, while the first method takes into account measurements only by window size. For different signals, the same window size will have a different effect on the quality of averaging and noise elimination. However, the disadvantage of exponential smoothing is the shift relative to the actual trajectory and measurements.

This methodical error can lead to delay or advance of a signal. The running mean method has almost no such methodological error. By changing the smoothing factor, it is possible to adapt such a filter to different input signals: smoother or frequently changing.

LEARNING LOG

1. Lisa: I learned how to determine optimal smoothing coefficient in exponential smoothing, the window size M that provides equality of running and exponential mean and apply a exponential smoothing.
2. Yaroslav: first time I met with random walk model in programming, tried differrent ways of creating sets of normally distributed numbers,but couldn sutisfy task requirements, but all in all Lisa showed me the right one. Going through the main part of work, appllying running and exponential smoothing I made simple conclusion for myself: on the long run running mean creates better prediction of results, but with every step we approaching very recent data, exponential mean is better predictor for actual measurments.
3. Selamawit Asfaw: I have learnt two methods of data processing techniques that are running mean and exponential smoothing. From the analyzed data I have observed applying forward exponential smoothing and then backward one yields better results than running mean smoothing. From first part I have undestood that exponential smoothing performs better for larger datasets, keeping other parameters the same.