Skoltech

Skolkovo Institute of Science and Technology

I.Determination of optimal smoothing constant in exponential mean

II.Comparison of methodical errors of exponential and running mean

Experimental Data Processing

Assignment №2

Team №10

Submitted by:
Yunseok Park
Ilya Novikov
Ruslan Kalimullin

Instructor:
Tatiana Podladchikova

MA060238 (Term 1B, 2022-2023)

October 5th, 2022

Contents

1 Introduction **2**

2 Methodology **2**

2.1 Determination of optimal smoothing constant in exponential mean 2

2.2 Comparison of methodical errors of exponential and running mean 3

3 Results **4**

3.1 Determination of optimal smoothing constant in exponential mean 4

3.2 Comparison of methodical errors of exponential and running mean 5

4 Conclusion **7**

1 Introduction

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. Additional important outcome of this exercise is the solution of identification problem of noise statistics that is crucial for reliable estimation.

2 Methodology

2.1 Determination of optimal smoothing constant in exponential mean

Question 1
Generation of true trajectory and measurements of it. True trajectory X_i is defined by using random walk model:

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

w_i represents normally distributed random noise with zero mathematical expectation and variance $\sigma_w^2 = 17$. Two different trajectories are considered which are 300 and 300 steps for this part of the report. Initial condition $X_1 = 10$ is given to generate true trajectory.

Measurements z_i of the process X_i:

$$z_i = X_i + \eta_i \tag{2}$$

η_i - normally distributed random noise with zero mathematical expectation and variance $\sigma_\eta^2 = 10$.

Question 2
Identification of σ_w^2 and σ_η^2 for different size of trajectory (300 and 3000). Comparison of estimation results with true values of variances σ_w^2 and σ_η^2.

Variances are calculated using system of equations:

$$\begin{cases} \nu_i = z_i - z_{i-1} = w_i + \eta_i - \eta_{i-1} \\ \rho_i = z_i - z_{i-2} = w_i + wi - 1 + \eta_i - \eta_{i-2} \\ E[\nu_i^2] = \sigma_w^2 + 2/\sigma_\eta^2 \\ E[\rho_i^2] = 2\sigma_w^2 + 2\sigma_\eta^2 \\ E[\nu_i^2] \approx \frac{1}{n-1}\sum_{k=1}^{N}(\nu_k^2) \\ E[\rho_i^2] \approx \frac{1}{n-2}\sum_{k=3}^{N}(\rho_k^2) \end{cases} \tag{3}$$

Question 3
Determination of optimal smoothing coefficient in exponential smoothing.

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

$$\chi = \frac{\sigma_w^2}{\sigma_\eta^2} \tag{5}$$

Question 4
Perform exponential smoothing with the determined smoothing coefficient calculated in Equation 4. Plot the results and compare add measurements, true values of process and exponentially smoothed data.

2.2 Comparison of methodical errors of exponential and running mean

Question 1
Generation of true trajectory X_i using the random walk model Equation 1. Size of trajectory is 300 points, initial condition - $X_1 = 10$, variance of noise $\sigma_w^2 = 28^2$

Question 2
Generation of measurements z_i of the process X_i using Equation 2. Variance of noise measurement noise is given: $\sigma_\eta^2 = 97^2$

Question 3
Determination of optimal smoothing coefficient α using Equation 3.

Question 4
Determination of the window size that provides equality of running mean and exponential mean using determined smoothing constant α ($\sigma_{RM}^2 = \sigma_{ES}^2$).

Running Mean (RM):

$$\sigma_{RM}^2 = \frac{\sigma_\eta^2}{M} \tag{6}$$

Exponential Smoothing (ES):

$$\sigma_{ES}^2 = \sigma_\eta^2 \frac{\alpha}{2 - \alpha} \tag{7}$$

Question 5
Make visual comparison of results. Make conclusions which method give greater methodical error in conditions of equal errors conditioned by measurement errors for this particular generated trajectory.

3 Results

3.1 Determination of optimal smoothing constant in exponential mean

Question 2
After solving system of equations we get variance for 300 points:

$$\sigma_w^2 = 19.91, \sigma_\eta^2 = 8.88 \tag{8}$$

for 3000 points:

$$\sigma_w^2 = 17.33, \sigma_\eta^2 = 9.17 \tag{9}$$

Comparing two calculated variances σ_w^2 and σ_η^2 we can see that result with 3000 steps are way more closer to result with 300 steps. This indicates that taking bigger steps, so in our case 3000 steps, leads to more accurate result.

Question 3
From the variances calculated in Equations 4 and 5 we got the window size $\alpha_{300} = 0.74$ for step 300 and $\alpha_{3000} = 0.72$ for step 3000

Question 4

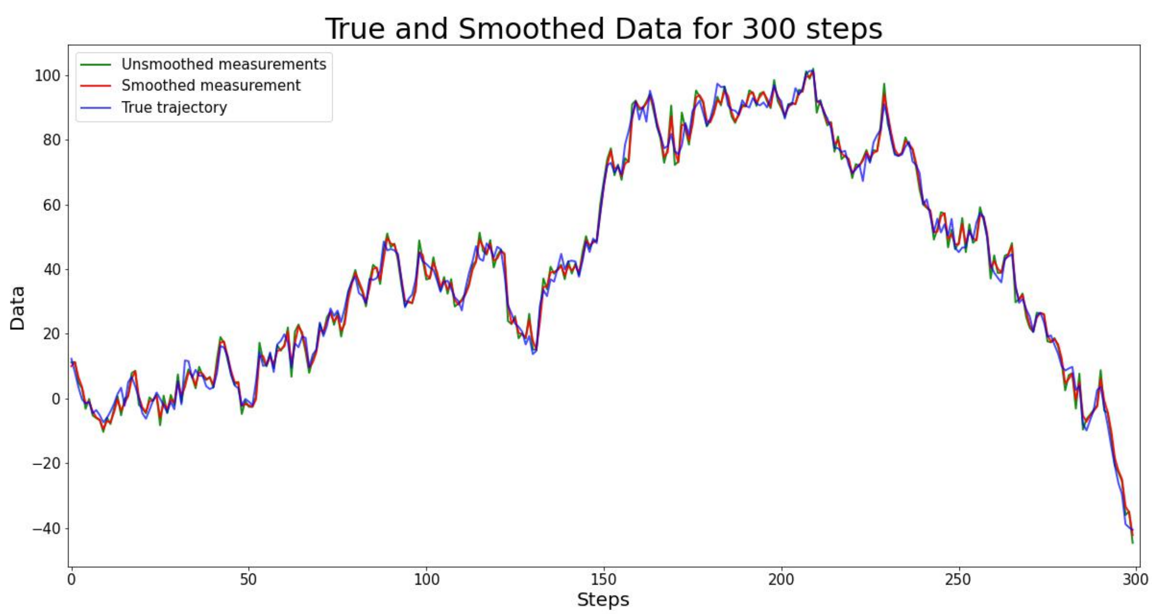


Figure 1: Comparison of True Trajectory, Smoothed Data for 300 steps

As can be seen from Figure 1 and 2, true trajectory, measurements and exponentially smoothed data when 3000 steps used give closer results than 300 step case.

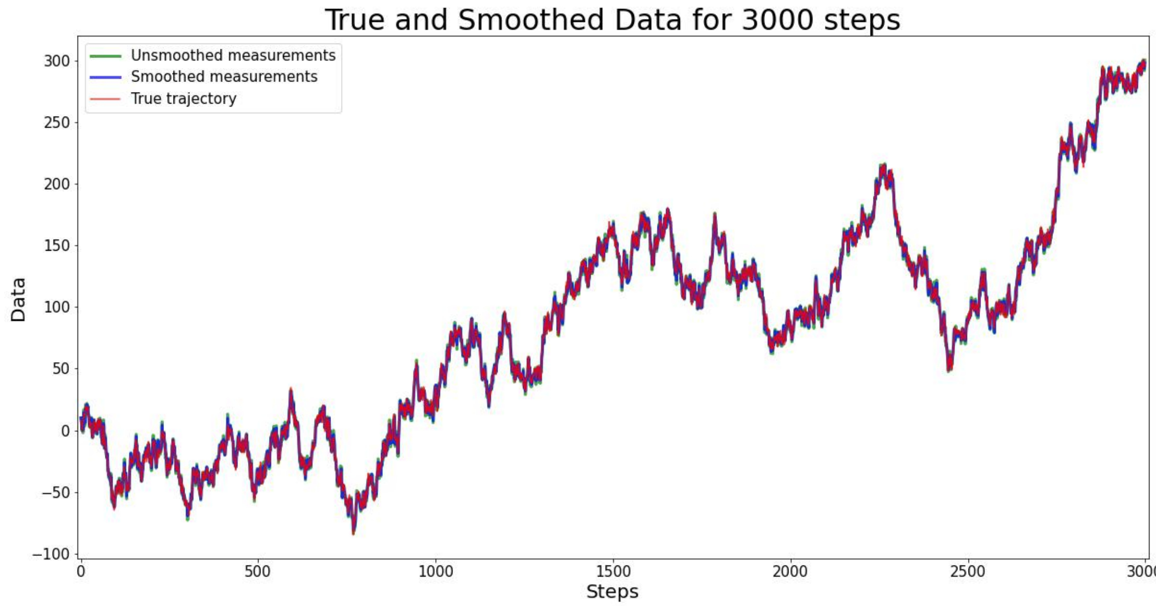


Figure 2: Comparison of True Trajectory, Smoothed Data for 3000 steps

3.2 Comparison of methodical errors of exponential and running mean

Question 3

Optimal smoothing coefficient α is calculated using Equation 3: $\alpha = 0.249$

Question 4

From the variances calculated in Equations 8 and 9 we got the window size $M = 7$.

Question 5

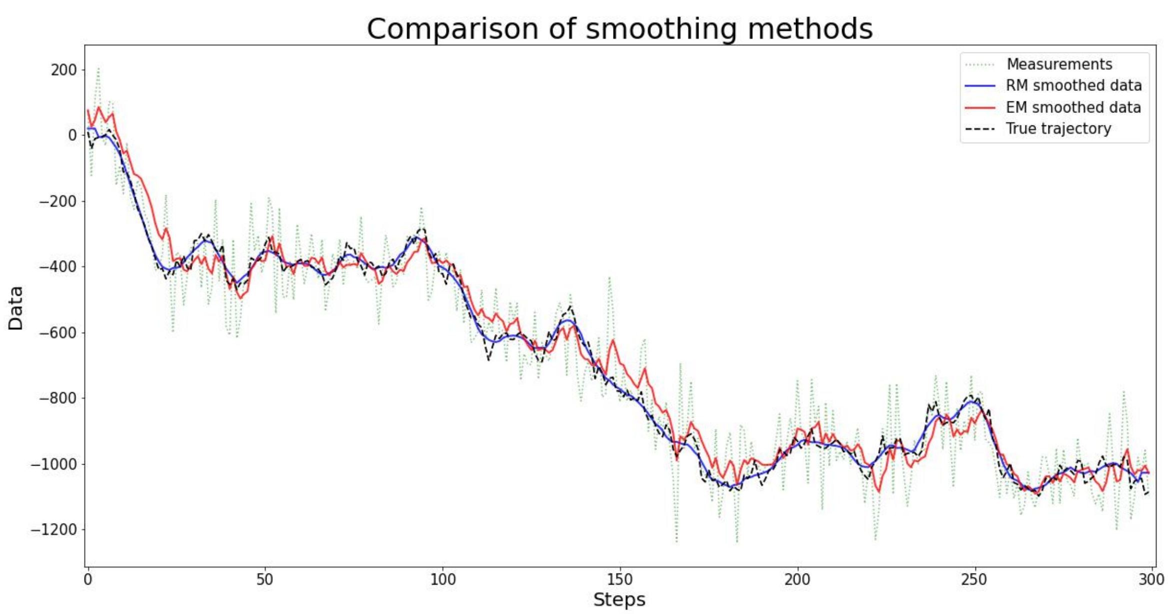


Figure 3: Comparison of True Trajectory, Measurements, Smoothed RM and ES

In Figure 3, two different smoothing method are compared by suing 300 steps. The offset from the true trajectory is much more in the exponentially smoothed data set compared to running mean method. Therefore, running mean smoothing gives closer results to the actual path even it has shifts in measurements, too.

4 Conclusion

What we have learnt and tried:

1. Creating a true trajectory with normally distributed random walk model by using Python.
2. Method to generate measurements from the existing trajectory.
3. Using identification methods to calculate variances after identifying random noise of true data and measurements.

What we have reflected upon:

1. Differences between running mean (RM) and exponential smoothing (ES) methods.
2. How using bigger data set affects smoothing.
3. How using different size of trajectories affects identification.
4. Decreasing offset by increasing size of trajectory
5. Step size affects identification of variances. By increasing step size we get a result closer to actual case.

Contribution of each members:

1. Ilya: writing the code for determining the optimal alpha for exponential mean
2. Ruslan: writing the code to compare the methodical errors of exponential and running mean
3. Yunseok: plotted plots, wrote report

In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import random
```

In [2]:
```python
#Part 1
#Determination of optimal smoothing constant in exponential mean
#Generation a true trajectory Xi using the random walk model & measurements zi of the process Xi
case1 = 300
case2 = 3000
variance_w = 17
variance_eta = 10
```

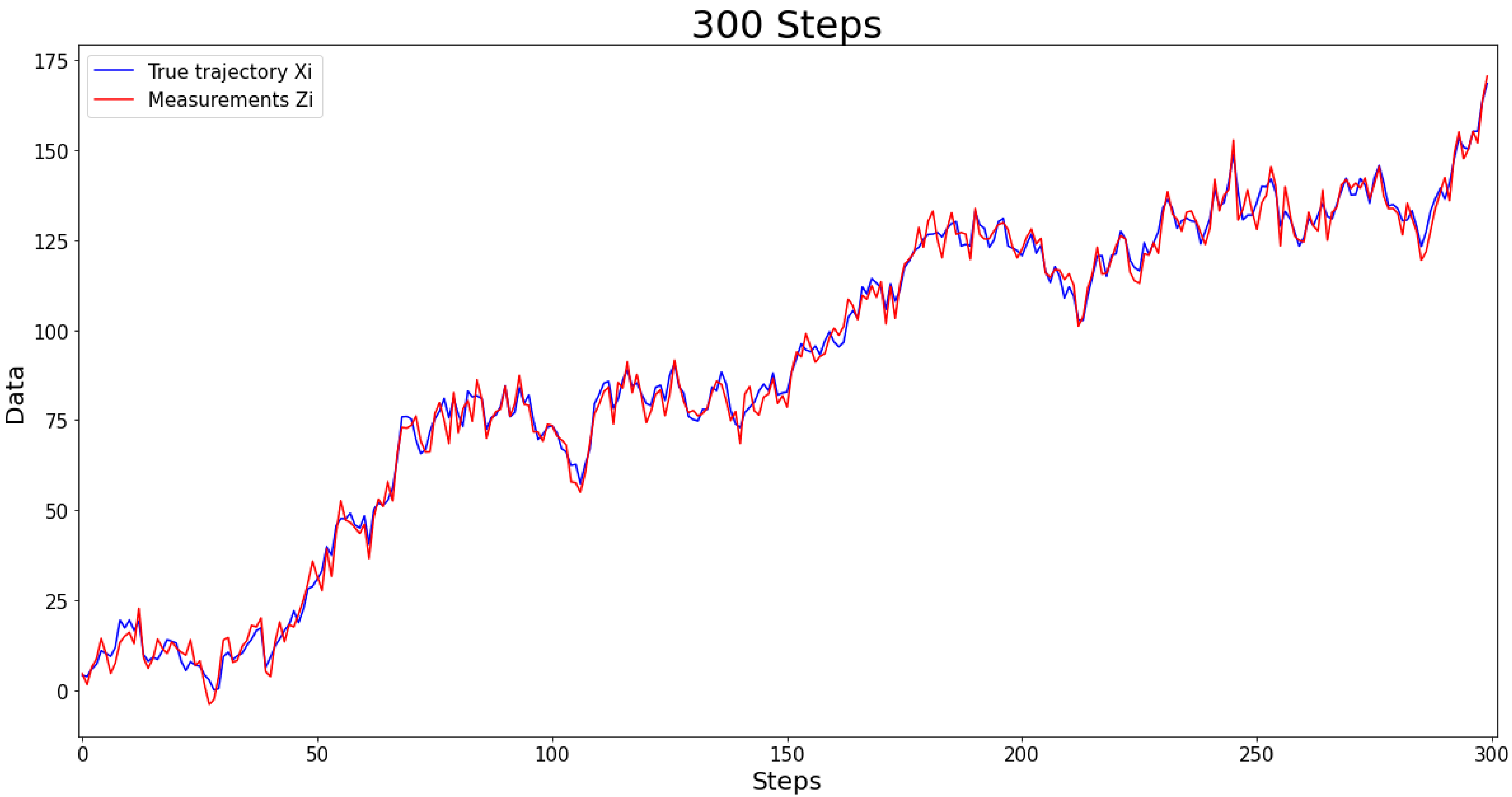
In [3]:
```python
# Trajectory generations for steps 300
true_trajectory_300 = []
noise_path_300 = []
measurements_300 = []
measurement_path_300 = []
residual_v_300 = []
residual_rho_300 = []
step_300 = 10
step1_300 = 10

for i in range(0, case1):
    noise_path_300.append(random.normalvariate(0, np.sqrt(variance_w)))
    step_300 += noise_path_300[i]
    true_trajectory_300.append(step_300)
    measurement_path_300.append(random.normalvariate(0, np.sqrt(variance_eta)))
    step1_300 = step_300 + measurement_path_300[i]
    measurements_300.append(step1_300)

    if i >= 1:
        residual_v_300.append(noise_path_300[i] + measurement_path_300[i] - measurement_path_300[i - 1])
    if i > 2:
        residual_rho_300.append(noise_path_300[i] + noise_path_300[i - 1] + measurement_path_300[i] - measurement_path_300[i - 2]

fig, ax = plt.subplots(figsize=(20, 10))
ax.set_title('300 Steps', fontsize = 30)
ax.set_ylabel('Data', fontsize = 20)
ax.set_xlabel('Steps', fontsize = 20)
ax.plot(true_trajectory_300, c='blue', label='True trajectory Xi')
ax.plot(measurements_300, c='red', label='Measurements Zi')
```

```
ax.tick_params(axis='both', labelsize=15)
plt.xlim(-1, 301)
ax.legend(fontsize = 15)
plt.savefig("Trajectory_300.jpg")
```



In [4]:

```
# Trajectory generations for steps 3000
true_trajectory_3000 = []
noise_path_3000 = []
measurements_3000 = []
measurement_path_3000 = []
residual_v_3000 = []
residual_rho_3000 = []
step_3000 = 10
```

```
step1_3000 = 10

for i in range(0, case2):
    noise_path_3000.append(random.normalvariate(0, np.sqrt(variance_w)))
    step_3000 += noise_path_3000[i]
    true_trajectory_3000.append(step_3000)
    measurement_path_3000.append(random.normalvariate(0, np.sqrt(variance_eta)))
    step1_3000 = step_3000 + measurement_path_3000[i]
    measurements_3000.append(step1_3000)

    if i >= 1:
        residual_v_3000.append(noise_path_3000[i] + measurement_path_3000[i] - measurement_path_3000[i - 1])
    if i > 2:
        residual_rho_3000.append(noise_path_3000[i] + noise_path_3000[i - 1] + measurement_path_3000[i] - measurement_path_3000[:

fig, ay = plt.subplots(figsize=(20, 10))
ay.set_title('3000 Steps', fontsize = 30)
ay.set_ylabel('Data', fontsize = 20)
ay.set_xlabel('Steps', fontsize = 20)
ay.plot(true_trajectory_3000, c='blue', linewidth = 2, label='True trajectory Xi')
ay.plot(measurements_3000, c='red', alpha = 1, label='Measurements Zi')
ay.tick_params(axis='both', labelsize=15)
plt.xlim(-10, 3010)
ay.legend(fontsize = 15)
plt.savefig("Trajectory_3000.jpg")
```

In [5]:

```
#Identify random noise of true data and measurement for 300
E_v_300 = 0
E_rho_300 = 0
E_v_300 = sum(np.power(residual_v_300, 2)) * 1/(case1 - 1)
E_rho_300 = sum(np.power(residual_rho_300, 2)) * 1/(case1 - 2)

sigma_w_300 = E_rho_300 - E_v_300
sigma_eta_300 = (E_v_300 - sigma_w_300)/2

print('Noise for 300 steps:\nSigma w =',sigma_w_300, '\nSigma eta =', sigma_eta_300)
```

```
Noise for 300 steps:
Sigma w = 19.919970653450612
Sigma eta = 8.880743417929725
```

In [6]:

```
#Identify random noise of true data and measurement for 3000
E_v_3000 = 0
E_rho_3000 = 0
E_v_3000 = sum(np.power(residual_v_3000, 2)) * 1/(case2 - 1)
E_rho_3000 = sum(np.power(residual_rho_3000, 2)) * 1/(case2 - 2)

sigma_w_3000 = E_rho_3000 - E_v_3000
sigma_eta_3000 = (E_v_3000 - sigma_w_3000)/2

print('Noise for 3000 steps:\nSigma w =',sigma_w_3000, '\nSigma eta =', sigma_eta_3000)
```

```
Noise for 3000 steps:
Sigma w = 17.33248594985114
Sigma eta = 9.167367878365589
```

In [7]:

```
#Determine optimal smoothing coefficitent in exponentional smoothing for 300
hi_300 = sigma_w_300 / sigma_eta_300
alpha_300 = (np.power((np.power(hi_300, 2) + 4 * hi_300), 0.5) - hi_300)/2
print('Optimal smoothing coefficitent for 300 step:\nKhi =', hi_300, '\nAlpha =', alpha_300)
```

```
Optimal smoothing coefficitent for 300 step:
Khi = 2.243052154083554
Alpha = 0.7495358929550138
```

In [8]:

```
#Determine optimal smoothing coefficitent in exponentional smoothing for 3000
hi_3000 = sigma_w_3000 / sigma_eta_3000
alpha_3000 = (np.power((np.power(hi_3000, 2) + 4 * hi_3000), 0.5) - hi_3000)/2
print('Optimal smoothing coefficitent for 3000 step:\nKhi =', hi_3000, '\nAlpha =', alpha_3000)
```

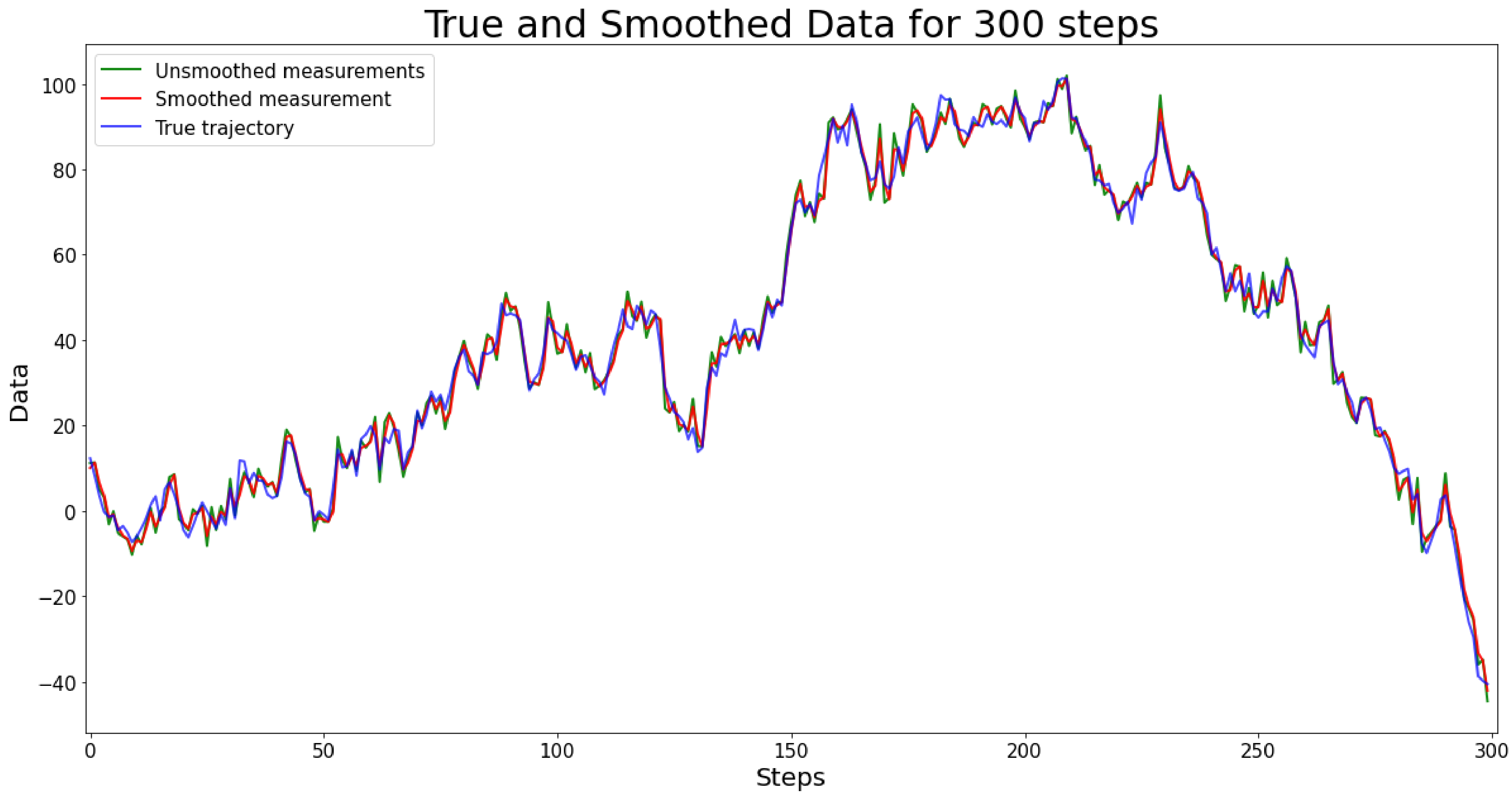
```
Optimal smoothing coefficitent for 3000 step:
Khi = 1.8906720205648904
Alpha = 0.7232958223581832
```

In [9]:

```
#Perfom exponential smoothing with the determined smoothing coefficient for 300
step_300 = 10
step0_300 = 10
step1_300 = 10
true_trajectory_s300 = []
measurements0_s300 = []
measurements_s300 = []
```

```python
for i in range(0, case1):
    step_300 += random.normalvariate(0, np.sqrt(sigma_w_300))
    true_trajectory_s300.append(step_300)
    step0_300 = step_300 + random.normalvariate(0, np.sqrt(sigma_eta_300))
    measurements0_s300.append(step0_300)
    if i == 0:
        measurements0_s300[i] = measurements0_s300[0]
    else:
        step1_300 = measurements0_s300[i - 1] + alpha_300 * (measurements0_s300[i] - measurements0_s300[i - 1])
    measurements_s300.append(step1_300)

fig, ax = plt.subplots(figsize=(20, 10))
ax.set_title('True and Smoothed Data for 300 steps', fontsize = 30)
ax.set_ylabel('Data', fontsize = 20)
ax.set_xlabel('Steps', fontsize = 20)
ax.plot(measurements0_s300, c='green', linewidth = 2, alpha = 0.9, label='Unsmoothed measurements')
ax.plot(measurements_s300, c='red', linewidth = 2, alpha = 0.9, label='Smoothed measurement')
ax.plot(true_trajectory_s300, c='blue', linewidth = 2, alpha = 0.7, label='True trajectory')
ax.tick_params(axis='both', labelsize=15)
plt.xlim(-1, 301)
ax.legend(fontsize = 15)
plt.savefig("Smoothing_300.jpg")
```



In [10]:

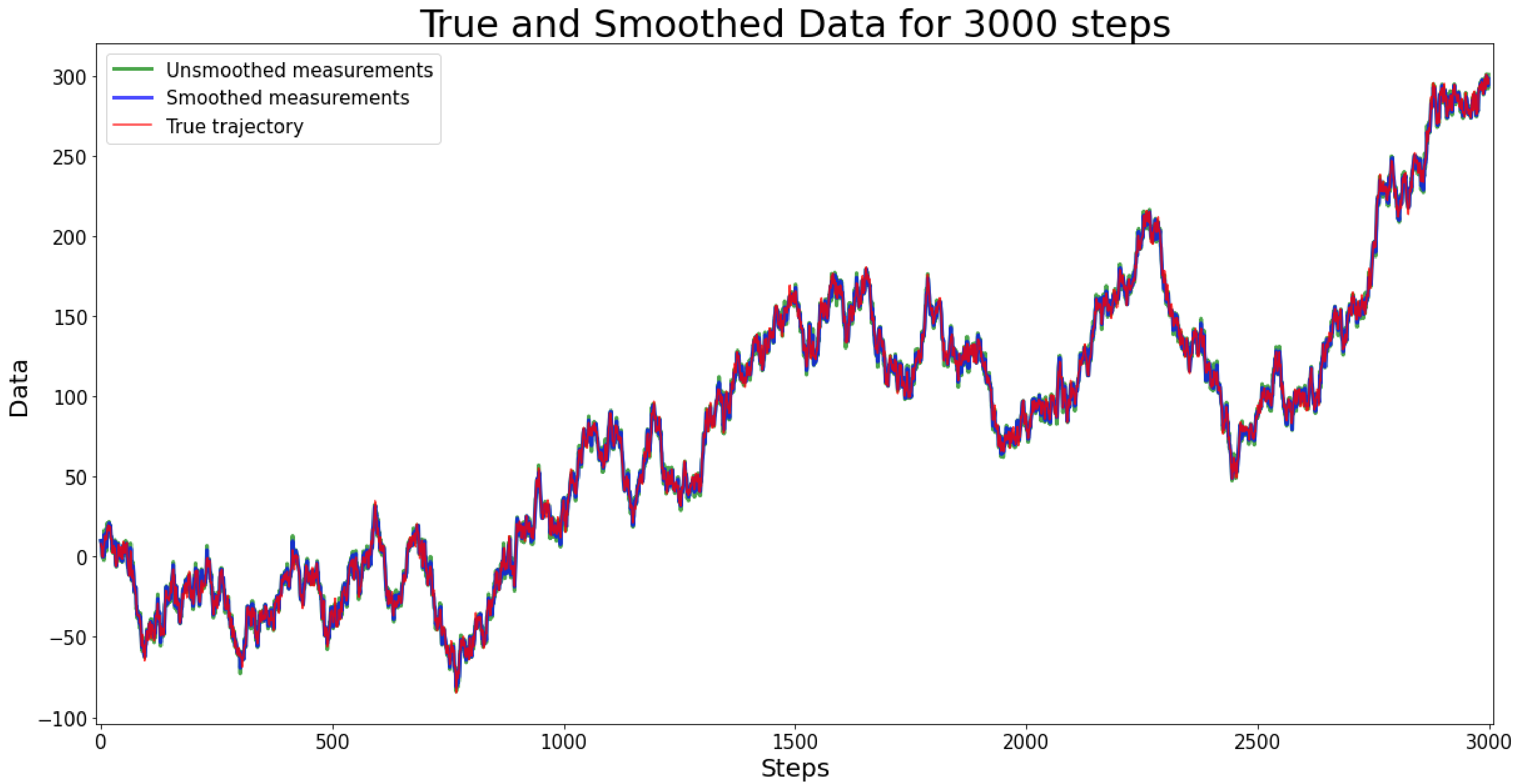
```
#Perfom exponential smoothing with the determined smoothing coefficient for 300
step_3000 = 10
step0_3000 = 10
step1_3000 = 10
true_trajectory_s3000 = []
measurements0_s3000 = []
measurements_s3000 = []
for i in range(0, case2):
    step_3000 += random.normalvariate(0, np.sqrt(sigma_w_3000))
    true_trajectory_s3000.append(step_3000)
    step0_3000 = step_3000 + random.normalvariate(0, np.sqrt(sigma_eta_3000))
    measurements0_s3000.append(step0_3000)
```

```
    if i == 0:
        measurements0_s3000[i] = measurements0_s3000[0]
    else:
        step1_3000 = measurements0_s3000[i - 1] + alpha_3000 * (measurements0_s3000[i] - measurements0_s3000[i - 1])
    measurements_s3000.append(step1_3000)

fig, ay = plt.subplots(figsize=(20, 10))
ay.set_title('True and Smoothed Data for 3000 steps', fontsize = 30)
ay.set_ylabel('Data', fontsize = 20)
ay.set_xlabel('Steps', fontsize = 20)
ay.plot(measurements0_s3000, c='g', linewidth = 3, alpha = 0.7, label='Unsmoothed measurements')
ay.plot(measurements_s3000, c='b', linewidth = 3, alpha = 0.7, label='Smoothed measurements')

ay.plot(true_trajectory_s3000, c='r', linewidth = 1.5, alpha = 0.8, label='True trajectory')

#ay.plot(true_trajectory_s3000, c='#e41a1c', markersize=1, alpha=0.6, label='True trajectory')
ay.tick_params(axis='both', labelsize=15)
plt.xlim(-10, 3010)
ay.legend(fontsize = 15)
plt.savefig("Smoothing_3000.jpg")
```



In [11]:

```
#Part 2
#Comparison of methodical erros of exponential and running man

#Size of trajectory
st = 300

#Intialization of arrays
x = np.zeros((st, 1))
z = np.zeros((st, 1))

#Intial condition of the ture trajectory X
x[0] = 10
```

```python
#Variances
sigma_w2 = 28 ** 2
sigma_eta2 = 97 ** 2

#Generation of normally distributed random noises with zero mathematical expectation
w = np.random.normal(0, np.sqrt(sigma_w2), st - 1)
eta = np.random.normal(0, np.sqrt(sigma_eta2), st)

#Generation of true trajectory X
for i in range(len(x) - 1):
    x[i + 1] = x[i] + w[i]

# Generation of measuerments z of the process x
for i in range(len(z)):
    z[i] = x[i] + eta[i]
```

In [12]:

```python
#Determine optimal smoothing coefficient in exponential smoothing
khi2 = sigma_w2 / sigma_eta2
alpha2 = (np.power((np.power(khi2, 2) + 4 * khi2), 0.5) - khi2)/2
print('Optimal smoothing coefficitent:\nKhi =', khi2, '\nAlpha =', alpha2)
```

```
Optimal smoothing coefficitent:
Khi = 0.08332447656499097
Alpha = 0.24998861233121078
```

In [13]:

```python
#The component of full error that is related to measurement erros
#window size
M = (2 - alpha2) / alpha2
print('Window size M =', M)
```

```
Window size M = 7.000364422000923
```

In [14]:

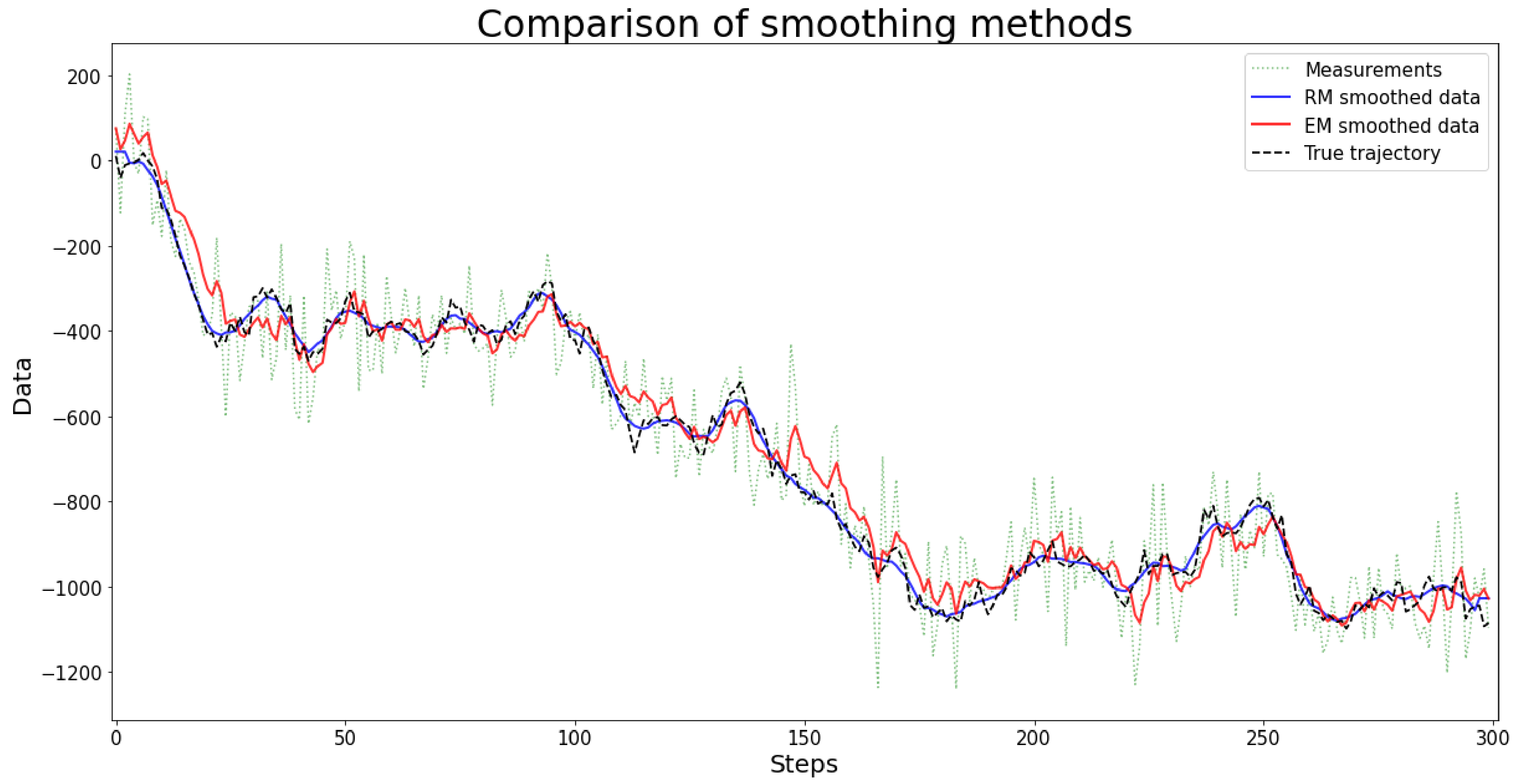
```python
#Running Mean
RM = np.zeros((st, 1))
beg_RM = np.sum(z[:3]) / len(z[:3])
end_RM = np.sum(z[len(z) - 3:]) / len(z[len(x) - 3:])
for i in range(len(z)):
    if i <= 2:
        RM[i] = beg_RM
    elif i >= len(z) - 3:
        RM[i] = end_RM
    else:
        RM[i] = np.sum(x[i - 3: i + 4]) / len(z[i - 3: i + 4])
```

In [15]:
```
#Exponential mean
EM = np.zeros((st, 1))
EM[0] = z[0]

for i in range(1, len(z)):
    EM[i] = EM[i - 1] + alpha2 * (z[i] - EM[i - 1])
```

In [16]:
```
#Plotting of measurements, true values of porcess, running and expontential mean
fig, af = plt.subplots(figsize=(20, 10))
af.set_title('Comparison of smoothing methods', fontsize = 30)
af.set_ylabel('Data', fontsize = 20)
af.set_xlabel('Steps', fontsize = 20)
af.plot(z, ':g', alpha = 0.5, label='Measurements')
af.plot(RM, 'b', linewidth = 2, alpha = 0.8, label='RM smoothed data')
af.plot(EM, 'r', linewidth = 2, alpha = 0.8, label='EM smoothed data')
af.plot(x, '--k', linewidth = 1.7, label='True trajectory')

af.tick_params(axis='both', labelsize=15)
plt.xlim(-1, 301)
af.legend(fontsize = 15)
plt.savefig("Compare.jpg")
```



Comparison of smoothing methods
Measurements
RM smoothed data
EM smoothed data
True trajectory
Data
Steps
200
0
−200
−400
−600
−800
−1000
−1200
0
50
100
150
200
250
300