Skoltech

Skolkovo Institute of Science and Technology

I. Comparison of the traditional 13-month running mean with the forward-backward exponential smoothing for approximation of 11-year sunspot cycle

II. 3d surface filtration using forward-backward smoothing

Experimental Data Processing

Assignment №4

Team №10

Submitted by:
Yunseok Park
Ilya Novikov
Ruslan Kalimullin

Instructor:
Tatiana Podladchikova

MA060238 (Term 1B, 2022-2023)

October 6th, 2022

Contents

1 Introduction **2**

2 Work progress **2**

2.1 Comparison of the traditional 13-month running mean with the forward-backward exponential smoothing for approximation of 11-year sunspot cycle 2

2.2 3d surface filtration using forward-backward smoothing 4

3 Conclusion **7**

1 Introduction

The objective of this laboratory work is to determine conditions for which broadly used methods of running and exponential mean provide effective solution and conditions under which they break down. Important outcome of this exercise is getting skill to choose the most effective method in conditions of uncertainty.

2 Work progress

2.1 Comparison of the traditional 13-month running mean with the forward-backward exponential smoothing for approximation of 11-year sunspot cycle

Question 2
Smoothed monthly data by 13-month running mean and by forward-backward exponential smoothing methods was found.

Running mean smoothing was performed using the equation 1:

$$\bar{R} = \frac{1}{24}R_{i-6} + \frac{1}{12}(R_{i-5} + R_{i-4} + \cdots + R_{i-1} + R_i + R_{i+1} + \cdots + R_{i+5}) + \frac{1}{24}R_{i+6} \quad (1)$$

The obtained running mean smoothed results are shown on Figure 1.

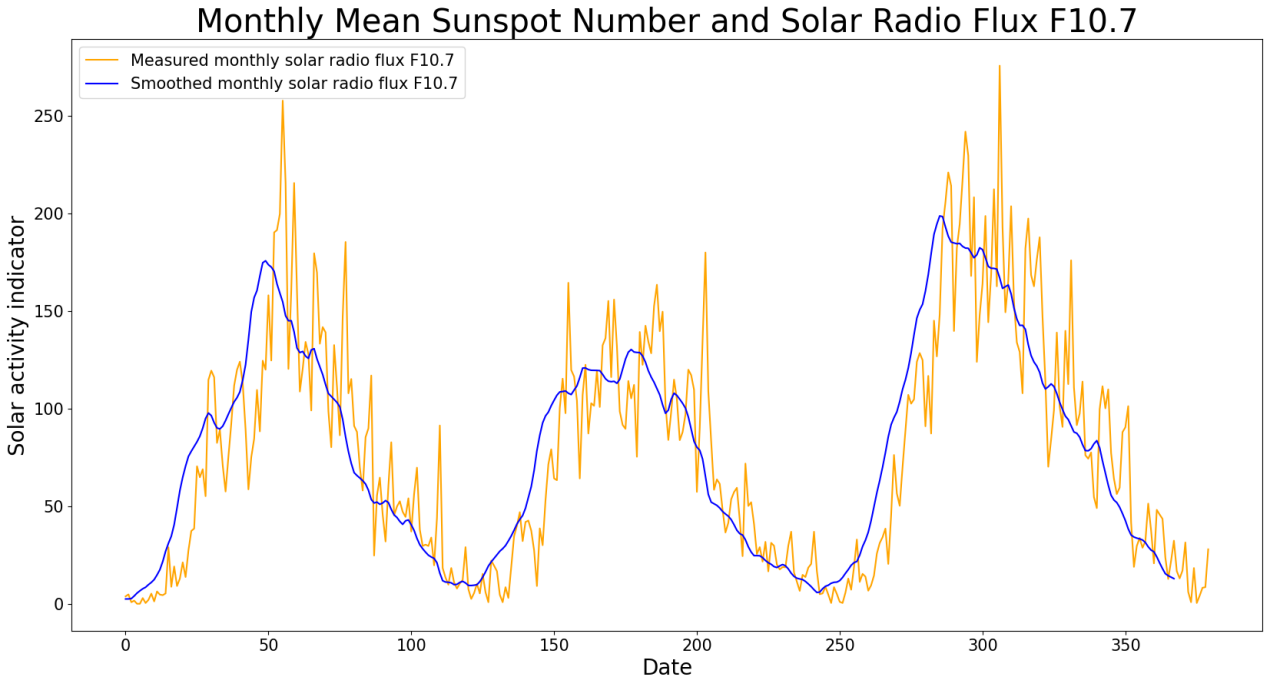


Figure 1: Comparison of measured and smoothed Data

Question 3

For making forward-backward exponential smoothing of monthly mean sunspot number it was necessary to find the constant α to provide better results compared to 13-month running mean. Several alphas have been considered (Figure 2) and it was estimated, that the best $\alpha = 0.12$.

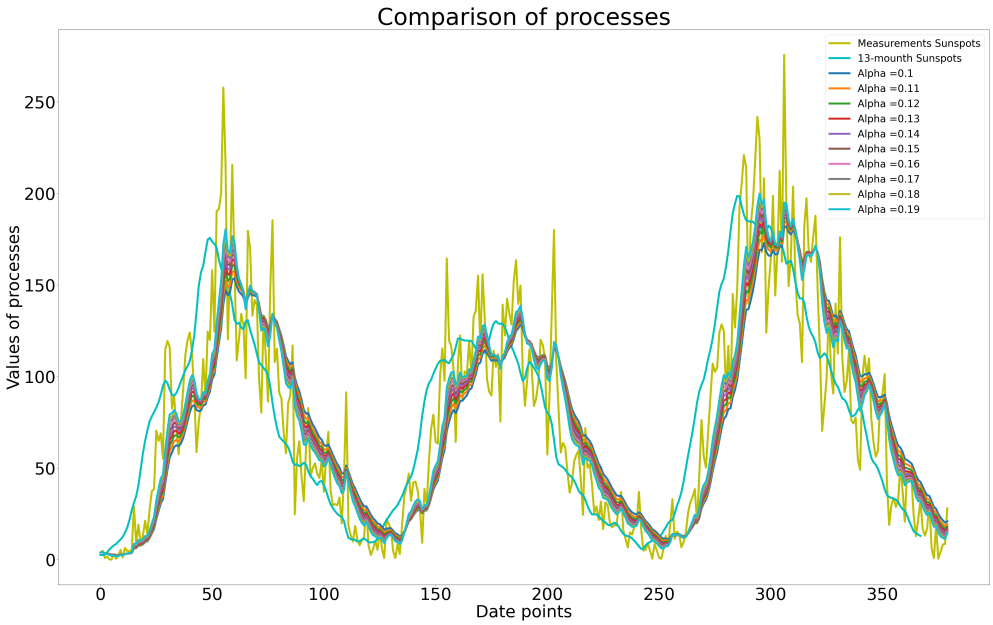


Figure 2: Forward exponential smoothing with different α

After that backward exponential smoothing with obtained α was made and final backward-forward exponential and running mean smoothings were demonstrated on Figure 3. According to deviation and variability indicators it can be concluded, that backward-forward exponential smoothing with $\alpha = 0.12$ provides better results than 13-month running mean according :

$$Dev_{run} = 3461816.504$$
$$Var_{run} = 1484.865$$

$$Dev_{exp} = 3267437.450$$
$$Var_{exp} = 58.744$$

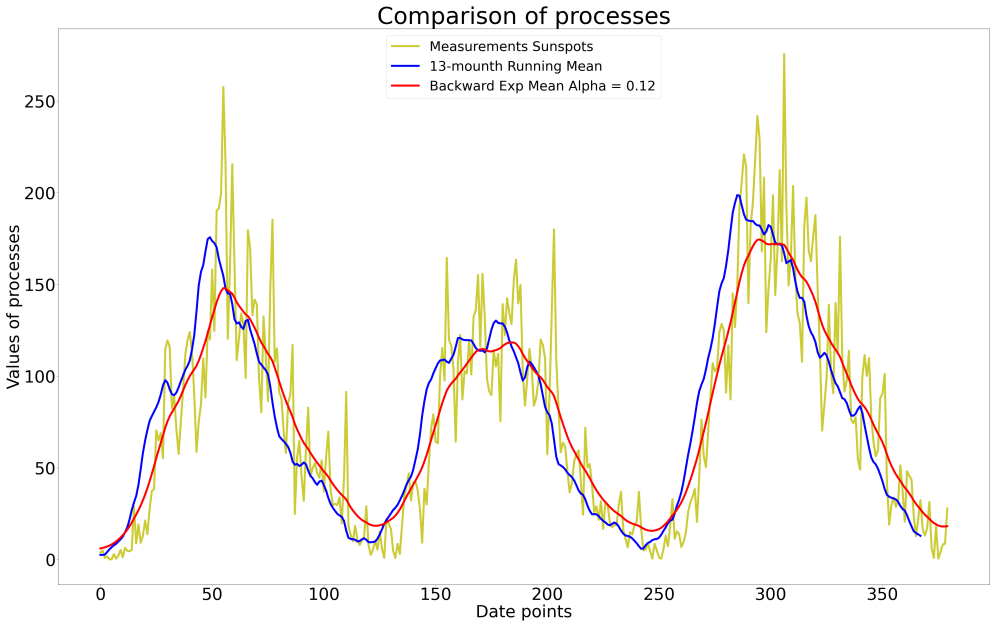


Figure 3: Comparison of measured and smoothed Data

2.2 3d surface filtration using forward-backward smoothing

Question 2

In this question noisy and true surface for visualization purposes was plotted (Figure 4).

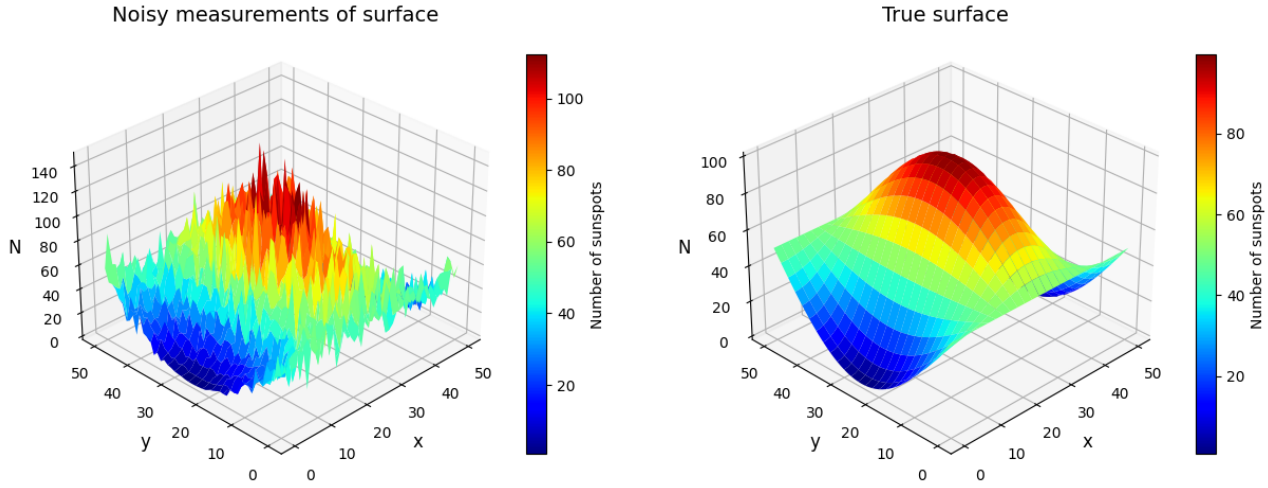


Figure 4: Noisy and true surface

Question 3

At this point the variance of deviation of noisy surface from the true one were determined:

$$Var_{noisy} = 122.016$$

Question 4-6

Forward-backward exponential smoothing was applied to filter noisy surface measurements (Figure 5). Considered smoothing constant was $\alpha = 0.335$.

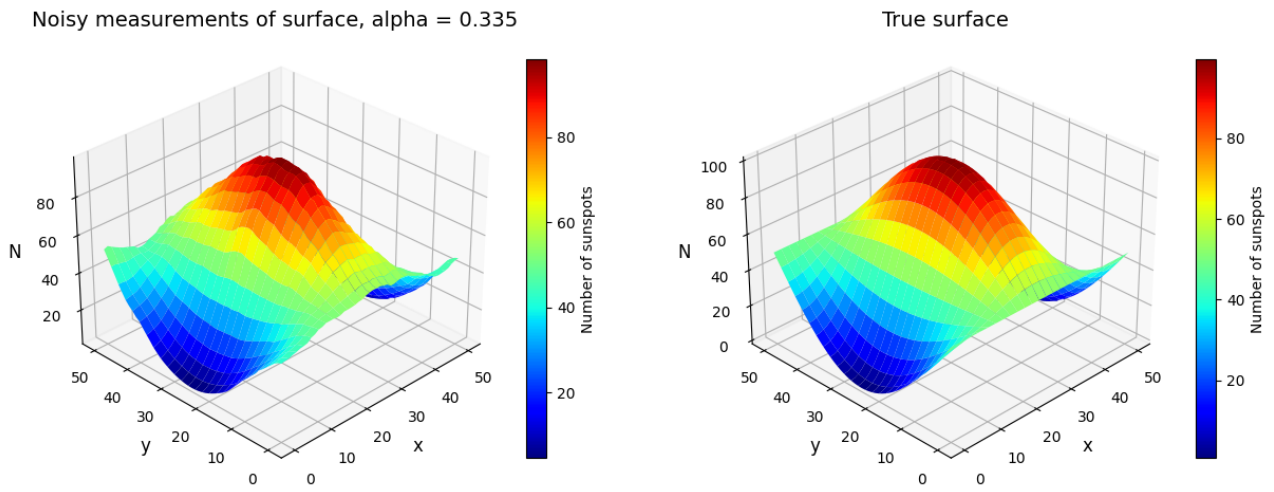


Figure 5: Smoothed and true surface

After plotting the variance of deviation of smoothed surface from the true one was determined:

$$Var_{smoothed} = 6.692$$

After smoothing the variance decreased by more than 18 times showing the method effectiveness.

Question 7

For making smoothing more precise different values of smoothing were considered (Figure 6).

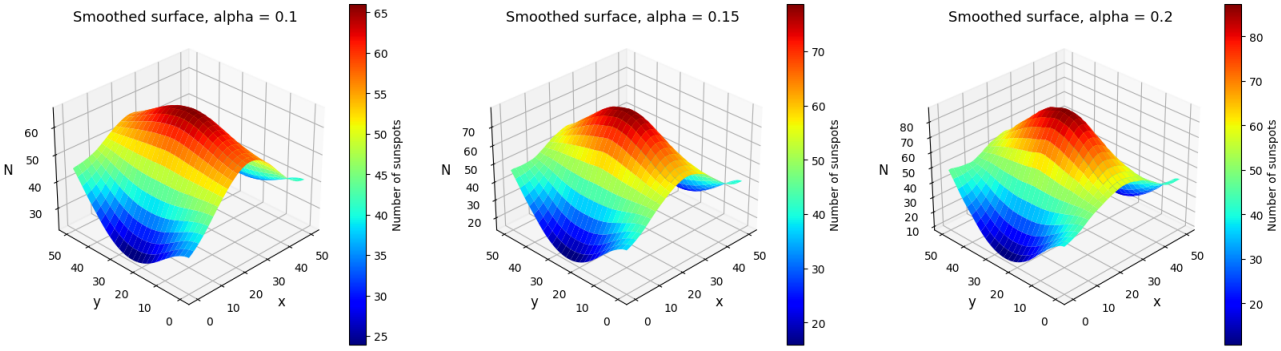


Figure 6: Smoothing examples

After many iterations the best exponential smoothing coefficient was obtained according to the value of variance of deviation of smoothed surface from the true one:

$$\alpha = 0.43$$
$$Var_{smoothed} = 4.302$$

The obtained running mean smoothed results are shown on Figure 7.

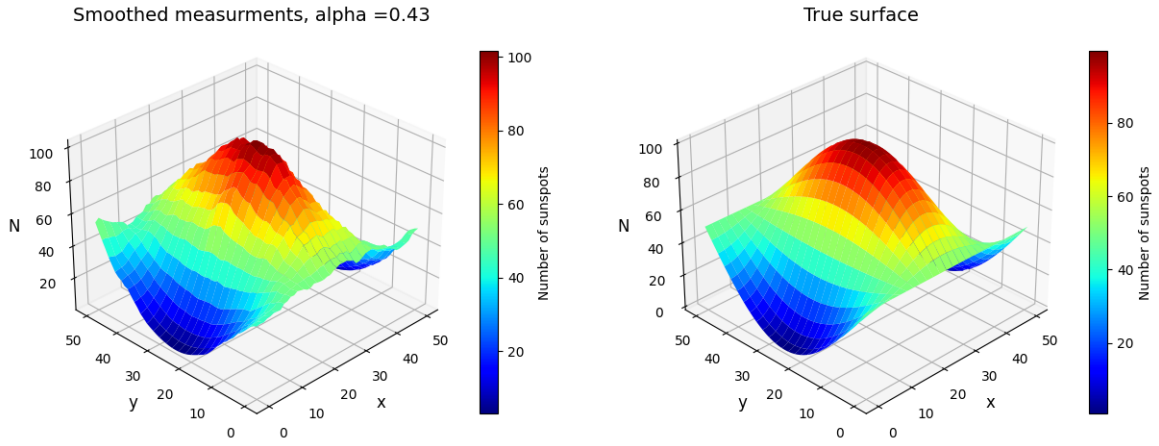


Figure 7: Smoothing with $\alpha = 0.43$

3 Conclusion

What we have learnt and tried:

1. Reconstruct the 3D surface on the basis noisy measurements of the surface in conditions of uncertainty
2. 3d surface filtration using forward-backward smoothing
3. Way to choose the optimal α

What we have reflected upon:

1. It is possible to use the indicators to choose the better method if it's hard to determine it visually
2. Backward exponential smoothing is also appropriate for 3D surfaces reconstruction
3. Via variance indicator it's better to check the accuracy of estimations than visually

Contribution of each members:

1. Ilya: wrote the code and plotted 3d plots
2. Ruslan: wrote the code and plotted forward-backward exponential and running mean smoothing
3. Yunseok: wrote the code and the report

In [82]:
```
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
from matplotlib import cm
from matplotlib.ticker import LinearLocator, FormatStrFormatter
import numpy as np
```

In [83]:
```
def exp_mean(alpha, sunspots):
    exp = np.zeros((len(sunspots),1))
    exp[0] = sunspots[0]
    for i in range(1, len(sunspots)):
        exp[i] = exp[i-1] + alpha*(sunspots[i] - exp[i-1])
    return(exp)
```

PART I

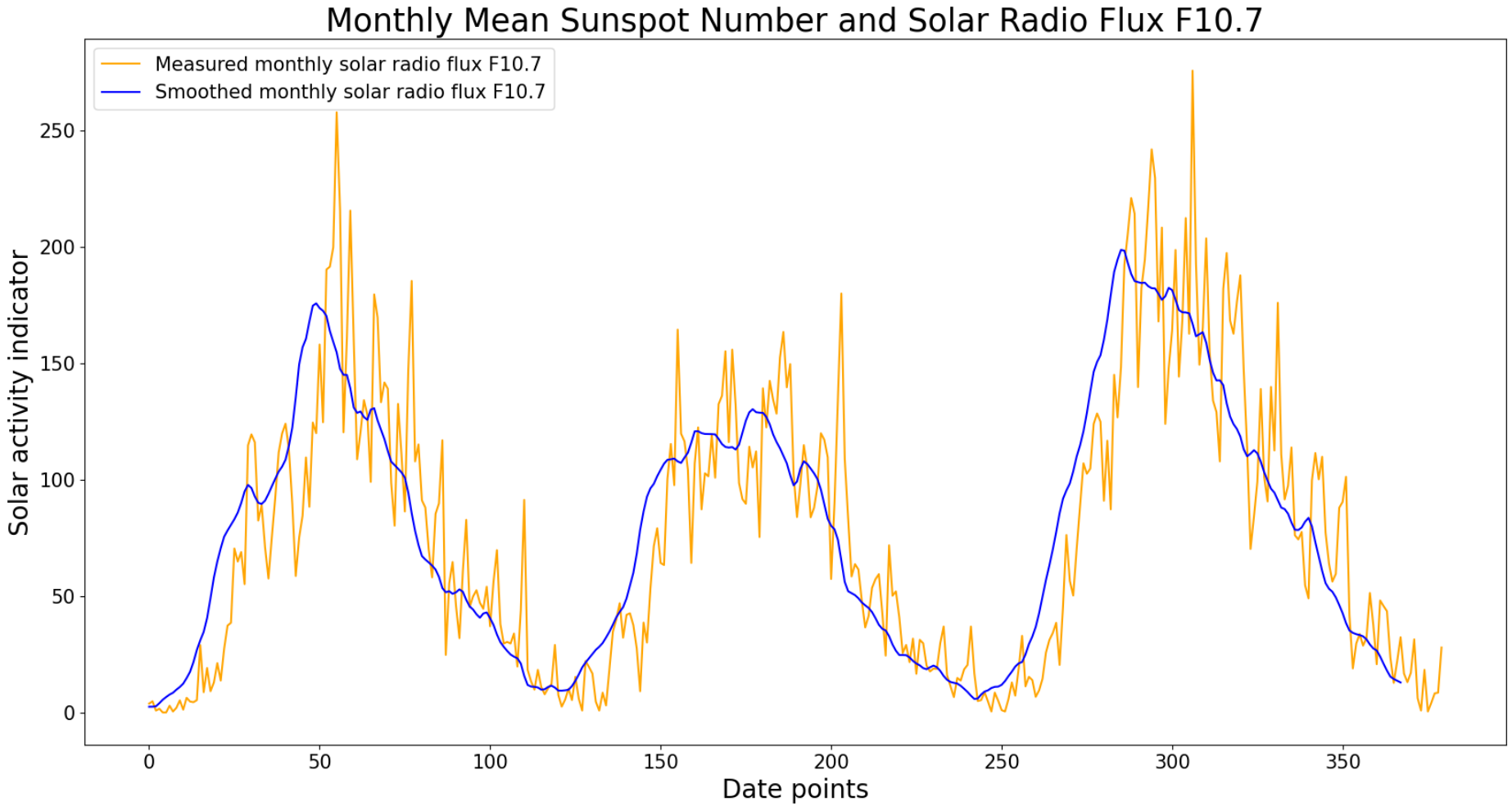
In [84]:
```
data = np.loadtxt('data_group3.txt')
```

In [85]:
```
years = data[:, 0]
months = data[:, 1]
sunspots = data[:, 2]
count = len(months)
```

In [86]:
```
r_sunspots = np.zeros((count - 12, 1))  # smoothed sunspots array

for n in range(r_sunspots.shape[0]):
    r_sunspots[n, 0] = 1/24 * (data[n, 2] + data[n + 12, 2]) + 1/12 * sum(data[n + 1:n + 12, 2])
```

In [87]:
```
#Plot for monthly mean sunspot number and solar radio flux F10.7 cm
fig, ax = plt.subplots(figsize=(20,10))
ax.set_title('Monthly Mean Sunspot Number and Solar Radio Flux F10.7', fontsize = 25)
ax.set_ylabel('Solar activity indicator', fontsize = 20)
ax.set_xlabel('Date points', fontsize = 20)
ax.plot(data[:, 2], 'orange', label='Measured monthly solar radio flux F10.7')
ax.plot(r_sunspots, 'b', label='Smoothed monthly solar radio flux F10.7')

#ax.set_xticks([0, 100, 200, 300, 400], labels = ['1947', '1955', '1963', '1972', '1980'])
ax.tick_params(axis='both', labelsize=15)
ax.legend(fontsize = 15)
```

Out[87]: <matplotlib.legend.Legend at 0x2569f424220>

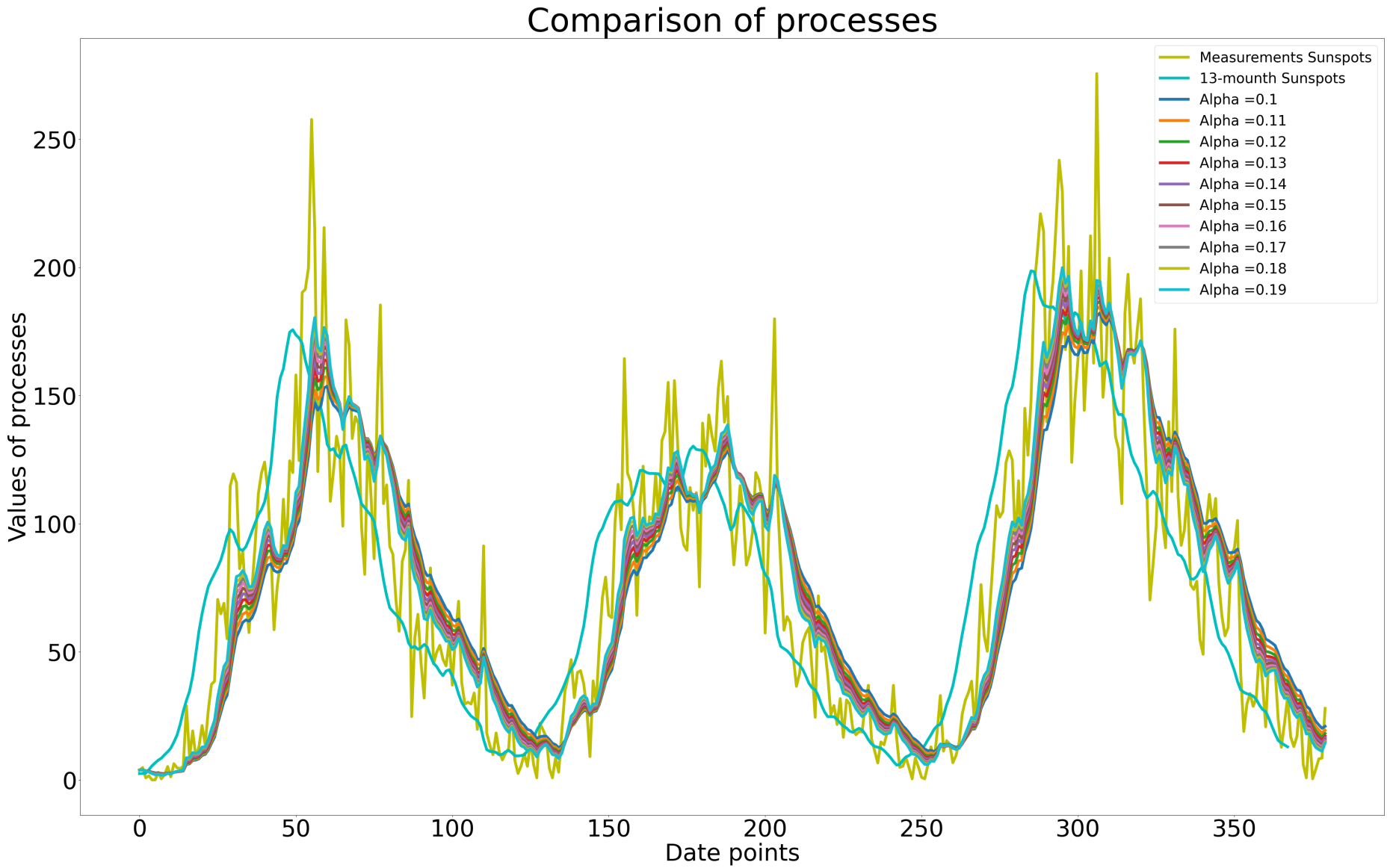


In [88]:
```
alfa = np.arange(0.1, 0.2, 0.01)
```

In [89]:
```
fig, ax = plt.subplots(figsize=(50, 30))
ax.set_title("Comparison of processes", fontsize = 70)
ax.set_ylabel("Values of processes", fontsize = 50)
ax.set_xlabel("Date points", fontsize = 50)
ax.plot(sunspots, linewidth = 6, label = "Measurements Sunspots", color = "y")
ax.plot(r_sunspots, linewidth = 6, label = "13-mounth Sunspots", color = "c")
#ax.plot(running_mean(2, z3), linewidth = 6, label = 'Running Mean', color = 'r')
for i in range(len(alfa)):
```

```
    ax.plot(exp_mean(alfa[i], sunspots), linewidth = 6, label = 'Alpha ='+ str(round(alfa[i], 2)))

ax.tick_params(labelsize = 50)
ax.legend(fontsize = 30)
```

Out[89]: <matplotlib.legend.Legend at 0x2569cc4c640>



Alpha best result for Forward Exponential Mean = 0.12

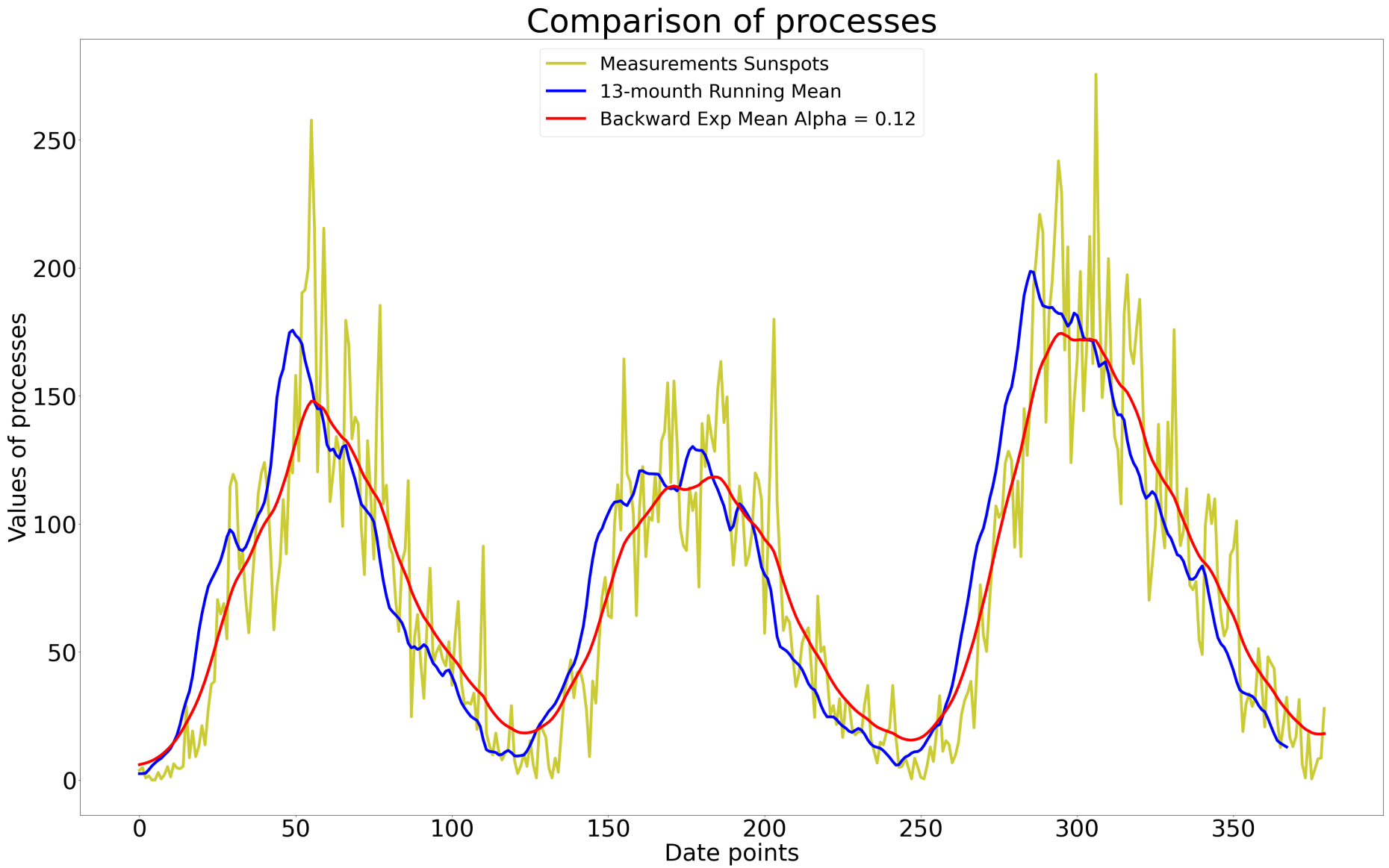
In [90]:
```
measurements_smoothed_b = np.zeros((len(sunspots), 1))
measurements_smoothed_b[len(sunspots)-1] = exp_mean(0.12,sunspots)[len(sunspots)-1]
for i in range(1, len(sunspots)):
     measurements_smoothed_b[len(sunspots)-i-1] = measurements_smoothed_b[len(sunspots)-i] + 0.12*(exp_mean(0.12,sunspots)[len(su
```

In [105…
```
Id_exp = sum((sunspots[:] - measurements_smoothed_b[0, :])**2)
Id_r = sum((sunspots[:] - r_sunspots[0, :])**2)

Iv_r = 0
for i in range(0, len(r_sunspots) -2):
    Iv_r += (r_sunspots[i+2,0] - 2*r_sunspots[i+1, 0] + r_sunspots[i, 0])**2

Iv_exp = 0
for i in range(0, len(r_sunspots) -2):
    Iv_exp += (measurements_smoothed_b[i+2,0] - 2*measurements_smoothed_b[i+1, 0] + measurements_smoothed_b[i, 0])**2

print('Running mean deviation =', Id_r)
print('Exponentional smoothing deviation =', Id_exp)
print('Running mean variability indicator =', Iv_r)
print('Exponentional smoothing variability indicator =', Iv_exp)
```

```
Running mean deviation = 3461816.504097223
Exponentional smoothing deviation = 3267437.4501358005
Running mean variability indicator = 1484.8649305555564
Exponentional smoothing variability indicator = 58.744135787460436
```

In [91]:
```
fig, ax = plt.subplots(figsize=(50,30))
ax.set_title("Comparison of processes", fontsize = 70)
ax.set_ylabel("Values of processes", fontsize = 50)
ax.set_xlabel("Date points", fontsize = 50)
ax.plot(sunspots, linewidth=6, alpha = 0.8, label = "Measurements Sunspots", c = 'y')
ax.plot(r_sunspots, linewidth = 6, label = "13-mounth Running Mean", c = 'b')
ax.plot(measurements_smoothed_b, 'r', linewidth = 6, label = "Backward Exp Mean Alpha = 0.12")

ax.tick_params(labelsize = 50)
ax.legend(fontsize = 40)
```

Out[91]:
```
<matplotlib.legend.Legend at 0x2569d052500>
```



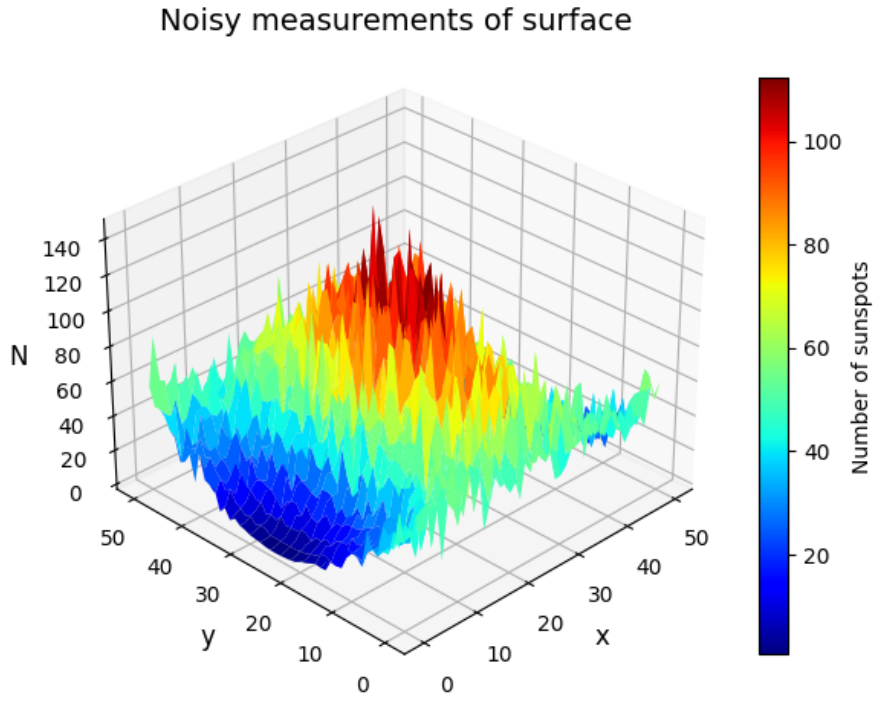
PART II

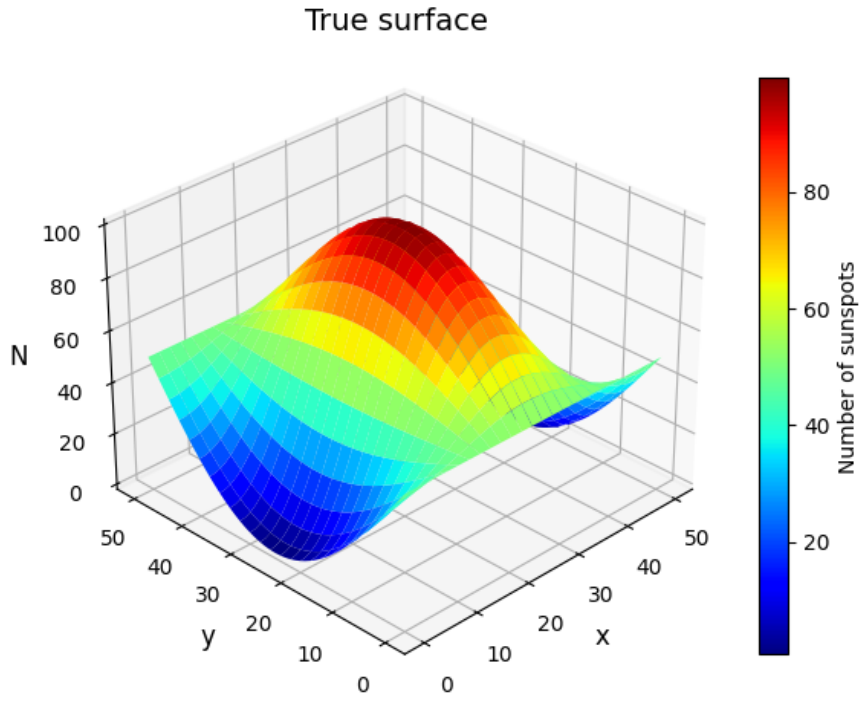
In [92]:
```
#Part 2
#3d surface filtration using forward-backward smoothing
data_true = np.loadtxt('true_surface.txt')
```

```
data_noisy = np.loadtxt('noisy_surface.txt')

x_coordinate, y_coordinate = np.meshgrid(np.arange(len(data_true[0,:])), np.arange(len(data_true[0,:])))
```

In [93]:

```
from cProfile import label

fig1 = plt.figure(figsize = (15, 7))

ax1 = fig1.add_subplot(1, 2, 2, projection = '3d')
ax1.set_title('True surface', fontsize = 14)
ax1.set_ylabel('y', fontsize = 12)
ax1.set_xlabel('x', fontsize = 12)
ax1.set_zlabel('N', fontsize = 12)

im1 = ax1.plot_surface(x_coordinate, y_coordinate, data_true, cmap=cm.jet)
ax1.view_init(30, -135)
plt.colorbar(im1, shrink = 0.7, label = 'Number of sunspots')

ax2 = fig1.add_subplot(1, 2, 1, projection='3d')
ax2.set_title('Noisy measurements of surface', fontsize = 14)
ax2.set_ylabel('y', fontsize = 12)
ax2.set_xlabel('x', fontsize = 12)
ax2.set_zlabel('N', fontsize = 12)

im2 = ax2.plot_surface(x_coordinate, y_coordinate, data_noisy, cmap=cm.jet)
ax2.view_init(30, -135)
plt.colorbar(im2, shrink = 0.7, label = 'Number of sunspots')
plt.show()
```





In [94]:
```
#The variance of deviation of noisy surface from the true one

Dev_n_t = np.zeros((len(data_noisy), len(data_noisy)))
Dev_n_t[:, :] = data_noisy[:, :] - data_true[:, :]
Dev_n_t = np.reshape(Dev_n_t, int(len(data_noisy)**2))
#надо делить на n или n-1?
Var_n_t = sum(Dev_n_t[:]**2)/(len(Dev_n_t))
print('Variance of deviation =', Var_n_t)
```

```
Variance of deviation = 122.01549845246063
```

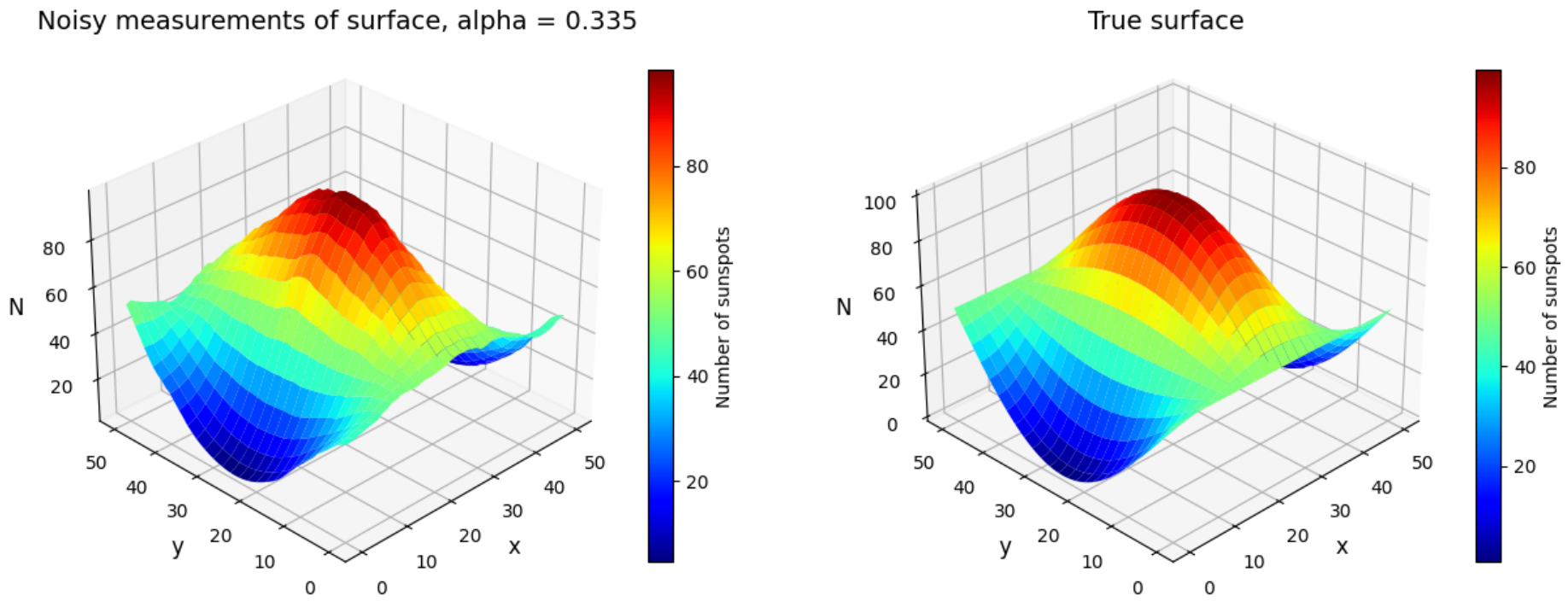
In [95]:
```
#Forward-backward exponential smoothing alpha = 0.335
alpha = 0.335
Points = len(data_noisy)
fex_str = np.zeros((Points, Points))
bfex_str = np.zeros((Points, Points))
fex_col = np.zeros((Points, Points))
bfex_col = np.zeros((Points, Points))

#Strings smoothing
for n in range(0, Points):
```

```
    for m in range(0, Points):
        if m == 0:
            fex_str[n, m] = data_noisy[n, m]
        else:
            fex_str[n, m] = fex_str[n, m-1] + alpha*(data_noisy[n, m] - fex_str[n, m-1])

    bfex_str[n, Points-1] = fex_str[n, Points-1]
    for i in range(1, Points):
        bfex_str[n, Points-i-1] = bfex_str[n, Points-i] + alpha*(fex_str[n, Points-1-i] - bfex_str[n, Points-i])

#Columns smoothing
for n in range(0, Points):
    for i in range(0, Points):
        if i == 0:
            fex_col[Points-1, n] = bfex_str[Points-1, n]
        else:
            fex_col[Points-i-1, n] = fex_col[Points-i, n] + alpha*(bfex_str[Points-1-i, n] - fex_col[Points-i, n])

    bfex_col[0, n] = fex_col[0, n]

    for m in range(1, Points):
        bfex_col[m, n] = bfex_col[m-1, n] + alpha*(fex_col[m, n] - bfex_col[m-1, n])
```

In [96]:

```
fig1 = plt.figure(figsize = (15, 7))

ax1 = fig1.add_subplot(1, 2, 2, projection = '3d')
ax1.set_title('True surface', fontsize = 14)
ax1.set_ylabel('y', fontsize = 12)
ax1.set_xlabel('x', fontsize = 12)
ax1.set_zlabel('N', fontsize = 12)

im1 = ax1.plot_surface(x_coordinate, y_coordinate, data_true, cmap=cm.jet)
ax1.view_init(30, -135)
plt.colorbar(im1, shrink = 0.7, label = 'Number of sunspots')

ax2 = fig1.add_subplot(1, 2, 1, projection='3d')
ax2.set_title('Noisy measurements of surface, alpha = 0.335', fontsize = 14)
ax2.set_ylabel('y', fontsize = 12)
ax2.set_xlabel('x', fontsize = 12)
ax2.set_zlabel('N', fontsize = 12)

im2 = ax2.plot_surface(x_coordinate, y_coordinate, bfex_col, cmap=cm.jet)
ax2.view_init(30, -135)
```

```
plt.colorbar(im2, shrink = 0.7, label = 'Number of sunspots')
plt.show()
```



In [97]:
```
#The variance of deviation of smoothed surface from the true one

Dev_s_t = np.zeros((len(bfex_col), len(bfex_col)))
Dev_s_t[:, :] = bfex_col[:, :] - data_true[:, :]
Dev_s_t = np.reshape(Dev_s_t, int(len(bfex_col)**2))
#надо делить на n или n-1?
Var_s_t = sum(Dev_s_t[:]**2)/(len(Dev_s_t))
print('Variance of deviation of smoothed surface from the true one =', round(Var_s_t, 3))
```

```
Variance of deviation of smoothed surface from the true one = 6.692
```

In [98]:
```
alpha = np.arange(0, 0.61, 0.05)
Points = len(data_noisy)

fex_str = np.zeros((Points, Points, len(alpha)))
bfex_str = np.zeros((Points, Points, len(alpha)))
fex_col = np.zeros((Points, Points, len(alpha)))
bfex_col = np.zeros((Points, Points, len(alpha)))
```

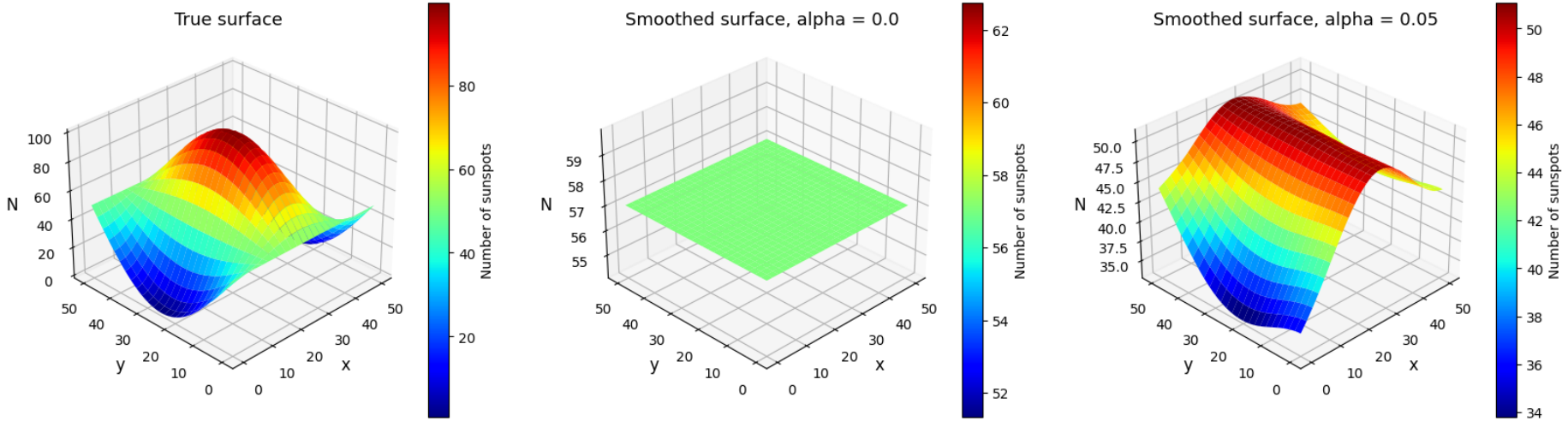
```
#Strings smoothing
for f in range(len(alpha)):
    for n in range(0, Points):
        for m in range(0, Points):
            if m == 0:
                fex_str[n, m, f] = data_noisy[n, m]
            else:
                fex_str[n, m, f] = fex_str[n, m-1, f] + alpha[f]*(data_noisy[n, m] - fex_str[n, m-1, f])

        bfex_str[n, Points-1, f] = fex_str[n, Points-1, f]
        for i in range(1, Points):
            bfex_str[n, Points-i-1, f] = bfex_str[n, Points-i, f] + alpha[f]*(fex_str[n, Points-1-i, f] - bfex_str[n, Points-i, f

#Columns smoothing
for f in range(len(alpha)):
    for n in range(0, Points):
        for i in range(0, Points):
            if i == 0:
                fex_col[Points-1, n, f] = bfex_str[Points-1, n, f]
            else:
                fex_col[Points-i-1, n, f] = fex_col[Points-i, n, f] + alpha[f]*(bfex_str[Points-1-i, n, f] - fex_col[Points-i, n,

        bfex_col[0, n, f] = fex_col[0, n, f]

        for m in range(1, Points):
            bfex_col[m, n, f] = bfex_col[m-1, n, f] + alpha[f]*(fex_col[m, n, f] - bfex_col[m-1, n, f])
```

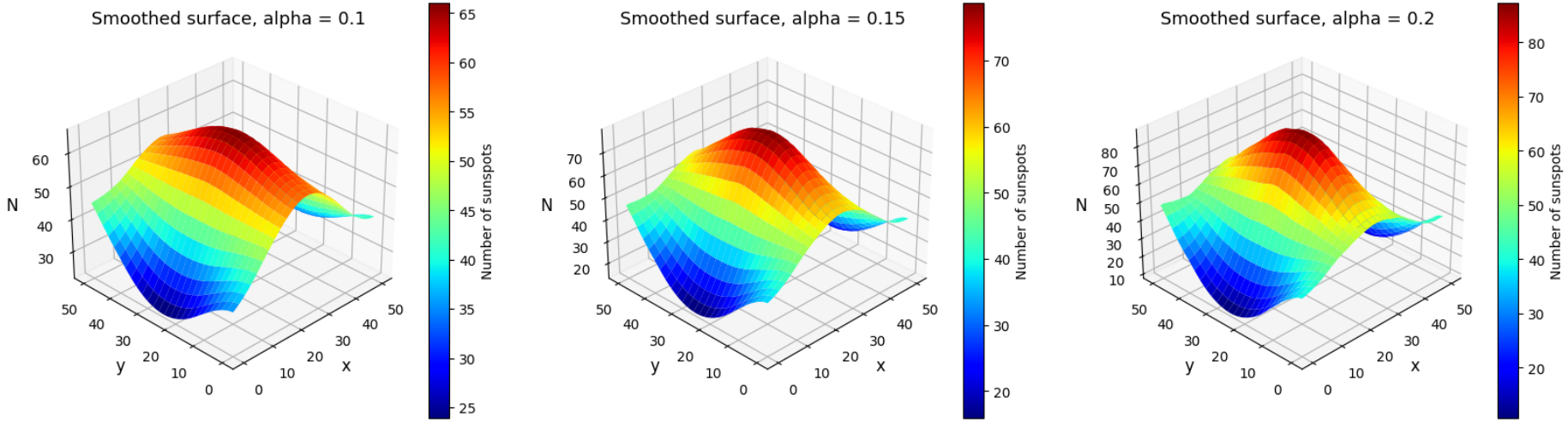
In [99]:
```
print(len (alpha))
```

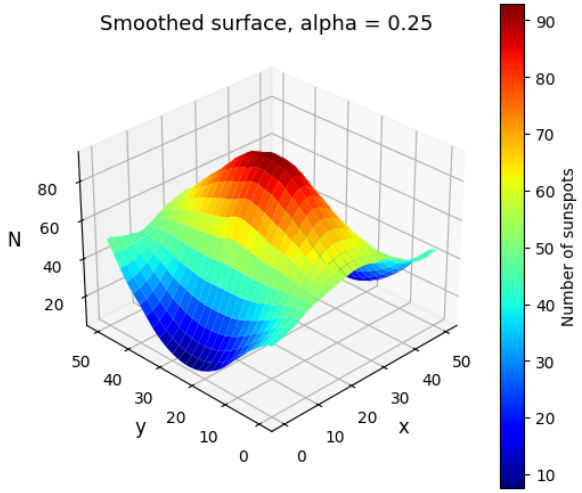
```
13
```

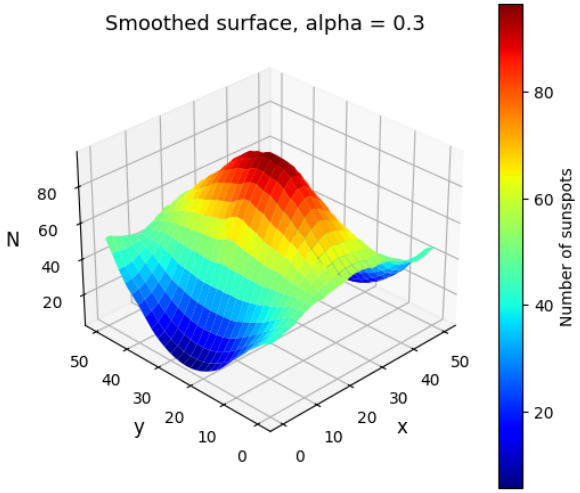
In [100…
```
fig1 = plt.figure(figsize = (20, 80))
fig1. tight_layout ()

ax1 = fig1.add_subplot(5, 3, 1, projection = '3d')
ax1.set_title('True surface', fontsize = 13)
ax1.set_ylabel('y', fontsize = 12)
ax1.set_xlabel('x', fontsize = 12)
ax1.set_zlabel('N', fontsize = 12)

im1 = ax1.plot_surface(x_coordinate, y_coordinate, data_true, cmap=cm.jet)
ax1.view_init(30, -135)
```

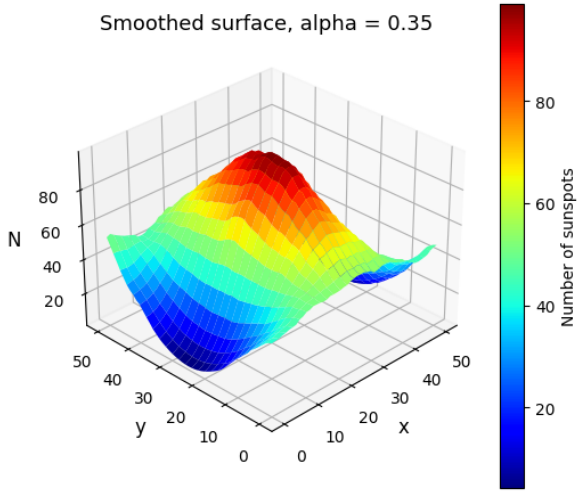
```
plt.colorbar(im1, shrink = 0.4, label = 'Number of sunspots')

for i in range(5):
    for j in range(3):
        if i*3+j == 13:
            break
        ax1 = plt.subplot(5, 3, i*3+j+2, projection = '3d')
        ax1.set_title('Smoothed surface, alpha = ' + str(round(alpha[i*3+j], 2)), fontsize = 13)
        ax1.set_ylabel('y', fontsize = 12)
        ax1.set_xlabel('x', fontsize = 12)
        ax1.set_zlabel('N', fontsize = 12)
        im1 = ax1.plot_surface(x_coordinate, y_coordinate, bfex_col[:, :, i*3+j], cmap=cm.jet)
        ax1.view_init(30, -135)
        plt.colorbar(im1, shrink = 0.4, label = 'Number of sunspots')
#plt.subplots_adjust(wspace=0, hspace=0)
plt.show()
```

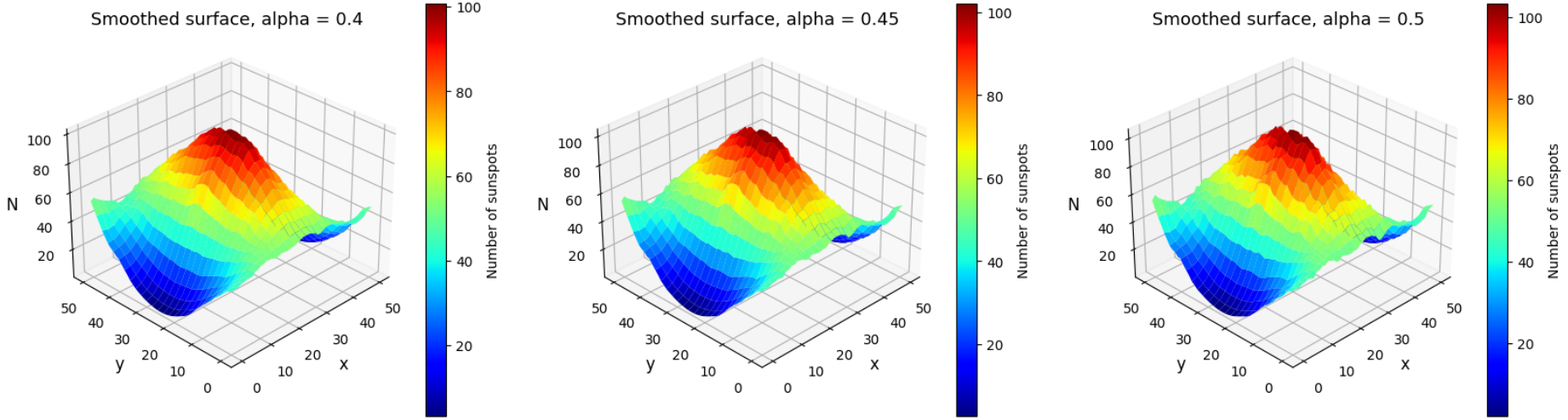

True surface
N
y
x
Number of sunspots
Smoothed surface, alpha = 0.0
N
y
x
Number of sunspots
Smoothed surface, alpha = 0.05
N
y
x
Number of sunspots

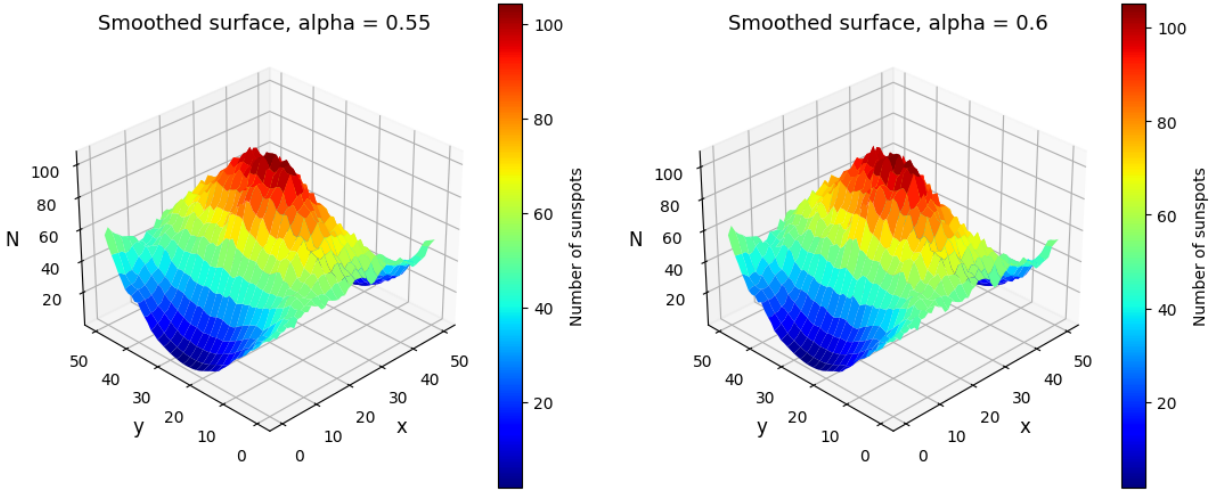

Smoothed surface, alpha = 0.1
Smoothed surface, alpha = 0.15
Smoothed surface, alpha = 0.2
N
x
y
Number of sunspots


Smoothed surface, alpha = 0.25
N
y
x
Number of sunspots


Smoothed surface, alpha = 0.3
N
y
x
Number of sunspots


Smoothed surface, alpha = 0.35
N
y
x
Number of sunspots


Smoothed surface, alpha = 0.4
Smoothed surface, alpha = 0.45
Smoothed surface, alpha = 0.5
N
x
y
Number of sunspots



In [101…
```
#The variance of deviation of smoothed surface from the true one for different alpha
def deviation(alpha_in):
    a = np.zeros((len(bfex_col), len(bfex_col), len(alpha)))
    a = bfex_col[:, :, alpha_in] - data_true[:, :]
    a = np.reshape(a, int(len(bfex_col)**2))
    Var = sum(a[:]**2)/(len(a))
    return Var

Var_alpha = np.zeros(len(alpha))
for i in range(len(alpha)):
    Var_alpha[i] = deviation(i)
alpha_min = alpha[np.where(Var_alpha[:] == np.min(Var_alpha))]
```

In [102…
```
alpha_precised = np.arange(alpha_min-0.05, alpha_min+0.06, 0.01)
Points = len(data_noisy)

fex_str_p = np.zeros((Points, Points, len(alpha_precised)))
bfex_str_p = np.zeros((Points, Points, len(alpha_precised)))
fex_col_p = np.zeros((Points, Points, len(alpha_precised)))
bfex_col_p = np.zeros((Points, Points, len(alpha_precised)))

#Strings smoothing
for f in range(len(alpha_precised)):
    for n in range(0, Points):
```

```
        for m in range(0, Points):
            if m == 0:
                fex_str_p[n, m, f] = data_noisy[n, m]
            else:
                fex_str_p[n, m, f] = fex_str_p[n, m-1, f] + alpha_precised[f]*(data_noisy[n, m] - fex_str_p[n, m-1, f])

        bfex_str_p[n, Points-1, f] = fex_str_p[n, Points-1, f]
        for i in range(1, Points):
            bfex_str_p[n, Points-i-1, f] = bfex_str_p[n, Points-i, f] + alpha_precised[f]*(fex_str_p[n, Points-1-i, f] - bfex_str

#Columns smoothing
for f in range(len(alpha_precised)):
    for n in range(0, Points):
        for i in range(0, Points):
            if i == 0:
                fex_col_p[Points-1, n, f] = bfex_str_p[Points-1, n, f]
            else:
                fex_col_p[Points-i-1, n, f] = fex_col_p[Points-i, n, f] + alpha_precised[f]*(bfex_str_p[Points-1-i, n, f] - fex_c

        bfex_col_p[0, n, f] = fex_col_p[0, n, f]

        for m in range(1, Points):
            bfex_col_p[m, n, f] = bfex_col_p[m-1, n, f] + alpha_precised[f]*(fex_col_p[m, n, f] - bfex_col_p[m-1, n, f])
```

In [103…

```
def deviation_p(alpha_in):
    a = np.zeros((len(bfex_col_p), len(bfex_col_p), len(alpha_precised)))
    a = bfex_col_p[:, :, alpha_in] - data_true[:, :]
    a = np.reshape(a, int(len(bfex_col_p)**2))
    Var = sum(a[:]**2)/(len(a))
    return Var

Var_alpha_final = np.zeros(len(alpha_precised))
for i in range(len(alpha_precised)):
    Var_alpha_final[i] = deviation_p(i)
alpha_final = alpha_precised[np.where(Var_alpha_final[:] == np.min(Var_alpha_final))]
print('Variance =', Var_alpha_final[np.where(Var_alpha_final[:] == np.min(Var_alpha_final))][0], '\nThe most precise alpha =', r
```

```
Variance = 4.301900836887329
The most precise alpha = 0.43
```

In [104…

```
fig1 = plt.figure(figsize = (15, 7))

ax1 = fig1.add_subplot(1, 2, 2, projection = '3d')
```

```
ax1.set_title('True surface', fontsize = 14)
ax1.set_ylabel('y', fontsize = 12)
ax1.set_xlabel('x', fontsize = 12)
ax1.set_zlabel('N', fontsize = 12)

im1 = ax1.plot_surface(x_coordinate, y_coordinate, data_true, cmap=cm.jet)
ax1.view_init(30, -135)
plt.colorbar(im1, shrink = 0.7, label = 'Number of sunspots')

ax2 = fig1.add_subplot(1, 2, 1, projection='3d')
ax2.set_title('Smoothed measurments, alpha =' + str(round(alpha_final[0], 2)), fontsize = 14)
ax2.set_ylabel('y', fontsize = 12)
ax2.set_xlabel('x', fontsize = 12)
ax2.set_zlabel('N', fontsize = 12)

im2 = ax2.plot_surface(x_coordinate, y_coordinate, bfex_col_p[:, :, np.where(Var_alpha_final[:] == np.min(Var_alpha_final))[0][0]
ax2.view_init(30, -135)
plt.colorbar(im2, shrink = 0.7, label = 'Number of sunspots')
plt.show()
```