МІНІСТЕРСТВО ОСВІТИ, НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ

Лабораторна робота №7
Аналіз даних

Підготував:

студент 4 курсу

групи ФІ-84

Коломієць Андрій Юрійович

E-mail: andrew.kolomiets.work@gmail.com

Київ – 2021

Лабораторна робота №7

Аналіз даних

Завдання на самостійну роботу

1. Встановити на комп‘ютері систему **Python** з підтримкою графічної бібліотеки **matplotlib**.

2. Детально ознайомитись із можливостями бібліотек **matplotlib** і **pywt**.

3. Побудувати вейвлет-скейлограму з іншою базовою вейвлет-функцією. Дослідити різницю вейвлет-скейлограм при різних базових вейвлет-функціях.

Виконання завдання

Інсталяція Python та бібліотек matplotlib та pywt

В **Linux/Unix** системах інсталяція **Python** середовища відбувається командами терміналу.

```
$ sudo apt-get install python3
```

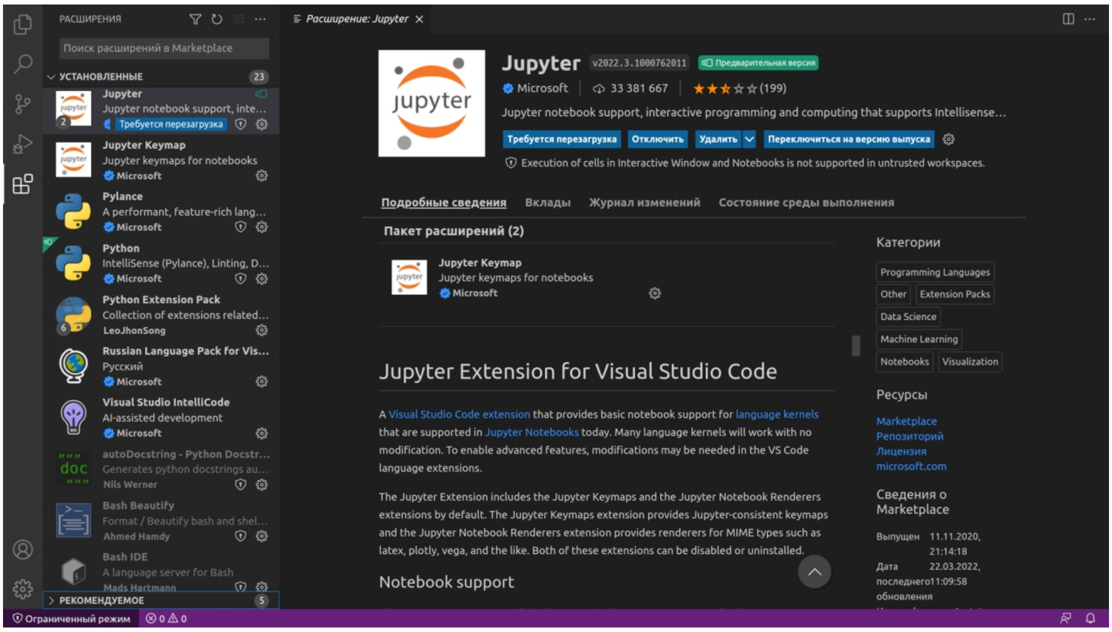
Також необхідний інсталятор програма для модулів середовища **Python.**

```
sudo apt-get install pip3
```

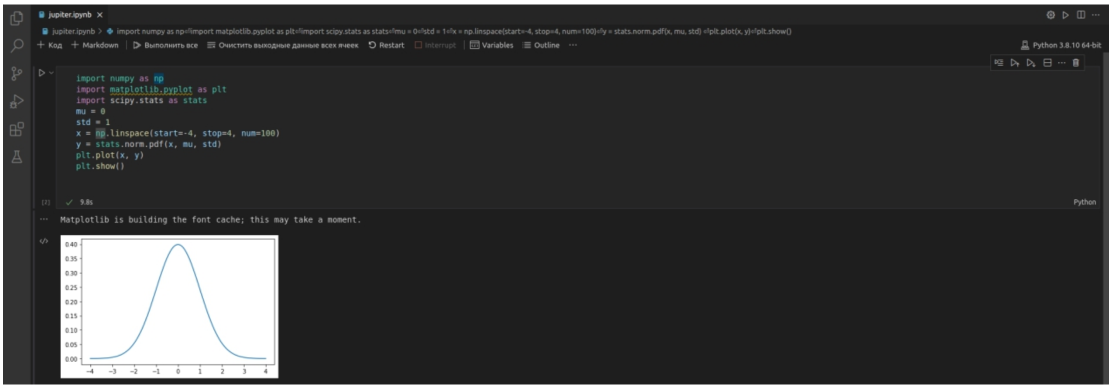
Інсталяція необхідних бібіліотек відбувається наступним шляхом.

```
$ pip3 install matplotlib PyWavelets
```

Додатково інсталюємо середовище **Jupiter** для графічного відображення графіків.



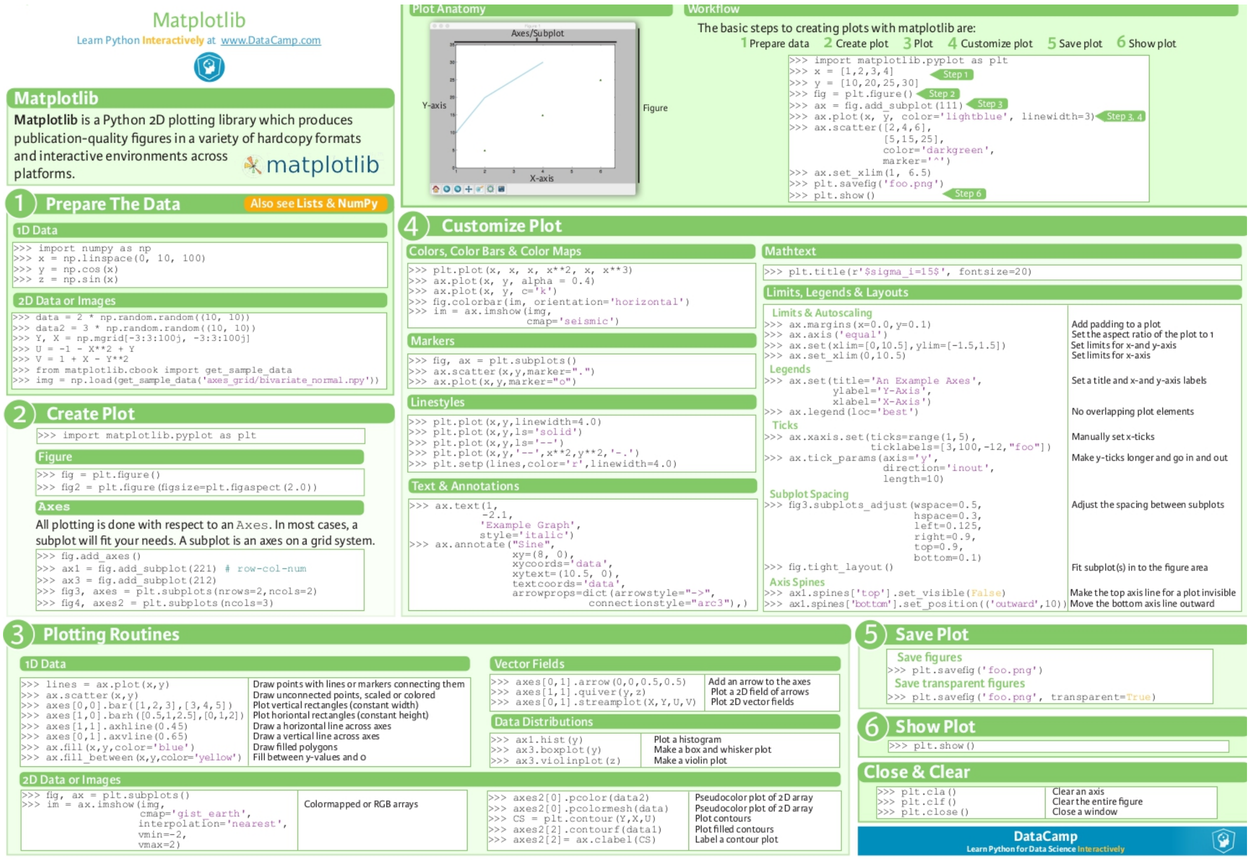
Протестуємо інстальоване середовище.



Можливості бібліотек matplotlib і pywt

matplotlib

https://s3.amazonaws.com/assets.datacamp.com/blog_assets/Python_Matplotlib_Cheat_Sheet.pdf



Matplotlib

Learn Python Interactively at www.DataCamp.com

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms.

1 Prepare The Data

Also see Lists & NumPy

1D Data

```
>>> import numpy as np
>>> x = np.linspace(0, 10, 100)
>>> y = np.cos(x)
>>> z = np.sin(x)
```

2D Data or Images

```
>>> data = 2 * np.random.random((10, 10))
>>> data2 = 3 * np.random.random((10, 10))
>>> Y, X = np.mgrid[-3:3:100j, -3:3:100j]
>>> U = -1 - X**2 + Y
>>> V = 1 + X - Y**2
>>> from matplotlib.cbook import get_sample_data
>>> img = np.load(get_sample_data('axes_grid/bivariate_normal.npy'))
```

2 Create Plot

```
>>> import matplotlib.pyplot as plt
```

Figure

```
>>> fig = plt.figure()
>>> fig2 = plt.figure(figsize=plt.figaspect(2.0))
```

Axes

All plotting is done with respect to an `Axes`. In most cases, a subplot will fit your needs. A subplot is an axes on a grid system.

```
>>> fig.add_axes()
>>> ax1 = fig.add_subplot(221) # row-col-num
>>> ax3 = fig.add_subplot(212)
>>> fig3, axes = plt.subplots(nrows=2,ncols=2)
>>> fig4, axes2 = plt.subplots(ncols=3)
```

Plot Anatomy

Workflow

The basic steps to creating plots with matplotlib are:

1 Prepare data 2 Create plot 3 Plot 4 Customize plot 5 Save plot 6 Show plot

```
>>> import matplotlib.pyplot as plt
>>> x = [1,2,3,4]                                        Step 1
>>> y = [10,20,25,30]
>>> fig = plt.figure()                                   Step 2
>>> ax = fig.add_subplot(111)                            Step 3
>>> ax.plot(x, y, color='lightblue', linewidth=3)        Step 3, 4
>>> ax.scatter([2,4,6],
               [5,15,25],
               color='darkgreen',
               marker='^')
>>> ax.set_xlim(1, 6.5)
>>> plt.savefig('foo.png')
>>> plt.show()                                           Step 6
```

4 Customize Plot

Colors, Color Bars & Color Maps

```
>>> plt.plot(x, x, x, x**2, x, x**3)
>>> ax.plot(x, y, alpha = 0.4)
>>> ax.plot(x, y, c='k')
>>> fig.colorbar(im, orientation='horizontal')
>>> im = ax.imshow(img,
                   cmap='seismic')
```

Markers

```
>>> fig, ax = plt.subplots()
>>> ax.scatter(x,y,marker=".")
>>> ax.plot(x,y,marker="o")
```

Linestyles

```
>>> plt.plot(x,y,linewidth=4.0)
>>> plt.plot(x,y,ls='solid')
>>> plt.plot(x,y,ls='--')
>>> plt.plot(x,y,'--',x**2,y**2,'-.')
>>> plt.setp(lines,color='r',linewidth=4.0)
```

Text & Annotations

```
>>> ax.text(1,
            -2.1,
            'Example Graph',
            style='italic')
>>> ax.annotate("Sine",
                xy=(8, 0),
                xycoords='data',
                xytext=(10.5, 0),
                textcoords='data',
                arrowprops=dict(arrowstyle="->",
                                connectionstyle="arc3"),)
```

Mathtext

```
>>> plt.title(r'$sigma_i=15$', fontsize=20)
```

Limits, Legends & Layouts

Limits & Autoscaling

```
>>> ax.margins(x=0.0,y=0.1)                          Add padding to a plot
>>> ax.axis('equal')                                 Set the aspect ratio of the plot to 1
>>> ax.set(xlim=[0,10.5],ylim=[-1.5,1.5])            Set limits for x-and y-axis
>>> ax.set_xlim(0,10.5)                              Set limits for x-axis
```

Legends

```
>>> ax.set(title='An Example Axes',                  Set a title and x-and y-axis labels
           ylabel='Y-Axis',
           xlabel='X-Axis')
>>> ax.legend(loc='best')                            No overlapping plot elements
```

Ticks

```
>>> ax.xaxis.set(ticks=range(1,5),                   Manually set x-ticks
                 ticklabels=[3,100,-12,"foo"])
>>> ax.tick_params(axis='y',                         Make y-ticks longer and go in and out
                   direction='inout',
                   length=10)
```

Subplot Spacing

```
>>> fig3.subplots_adjust(wspace=0.5,                 Adjust the spacing between subplots
                         hspace=0.3,
                         left=0.125,
                         right=0.9,
                         top=0.9,
                         bottom=0.1)
>>> fig.tight_layout()                               Fit subplot(s) in to the figure area
```

Axis Spines

```
>>> ax1.spines['top'].set_visible(False)             Make the top axis line for a plot invisible
>>> ax1.spines['bottom'].set_position(('outward',10))  Move the bottom axis line outward
```

3 Plotting Routines

1D Data

```
>>> lines = ax.plot(x,y)                             Draw points with lines or markers connecting them
>>> ax.scatter(x,y)                                  Draw unconnected points, scaled or colored
>>> axes[0,0].bar([1,2,3],[3,4,5])                   Plot vertical rectangles (constant width)
>>> axes[1,0].barh([0.5,1,2.5],[0,1,2])              Plot horiontal rectangles (constant height)
>>> axes[1,1].axhline(0.45)                          Draw a horizontal line across axes
>>> axes[0,1].axvline(0.65)                          Draw a vertical line across axes
>>> ax.fill(x,y,color='blue')                        Draw filled polygons
>>> ax.fill_between(x,y,color='yellow')              Fill between y-values and 0
```

Vector Fields

```
>>> axes[0,1].arrow(0,0,0.5,0.5)                     Add an arrow to the axes
>>> axes[1,1].quiver(y,z)                            Plot a 2D field of arrows
>>> axes[0,1].streamplot(X,Y,U,V)                    Plot 2D vector fields
```

Data Distributions

```
>>> ax1.hist(y)                                      Plot a histogram
>>> ax3.boxplot(y)                                   Make a box and whisker plot
>>> ax3.violinplot(z)                                Make a violin plot
```

2D Data or Images

```
>>> fig, ax = plt.subplots()                         Colormapped or RGB arrays
>>> im = ax.imshow(img,
                   cmap='gist_earth',
                   interpolation='nearest',
                   vmin=-2,
                   vmax=2)
```

```
>>> axes2[0].pcolor(data2)                           Pseudocolor plot of 2D array
>>> axes2[0].pcolormesh(data)                        Pseudocolor plot of 2D array
>>> CS = plt.contour(Y,X,U)                          Plot contours
>>> axes2[2].contourf(data1)                         Plot filled contours
>>> axes2[2]= ax.clabel(CS)                          Label a contour plot
```

5 Save Plot

Save figures

```
>>> plt.savefig('foo.png')
```

Save transparent figures

```
>>> plt.savefig('foo.png', transparent=True)
```

6 Show Plot

```
>>> plt.show()
```

Close & Clear

```
>>> plt.cla()                                        Clear an axis
>>> plt.clf()                                        Clear the entire figure
>>> plt.close()                                      Close a window
```

DataCamp

Learn Python for Data Science Interactively

pywt

https://pywavelets.readthedocs.io/

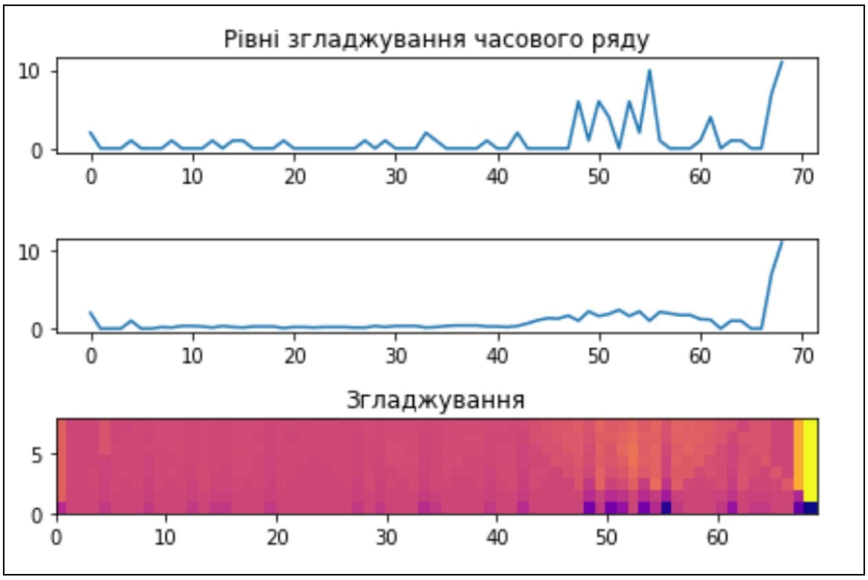
https://pywavelets.readthedocs.io/_/downloads/en/v1.0.0/pdf/

Віконне згладжування

Code-1

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import LogNorm

D=[2,0,0,0,1,0,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,2,1,0,
0,0,0,1,0,0,2,0,0,0,0,0,6,1,6,4,0,6,2,10,1,0,0,0,1,4,0,1,1,0,0,7,11]

M = 8
N = len(D)
Z = np.random.rand(M, N)
C = D
for i in range(N):
    for j in range(M):
        Z[j][i] = D[i]

for j in range(M):
    for k in range(j, N-j):
        Z[j][k] = 0
        for l in range(1, j):
            Z[j][k] = Z[j][k]+D[k-l]
            Z[j][k] = Z[j][k]+D[k+l]
        Z[j][k] = Z[j][k]-D[k]
        Z[j][k] = Z[j][k]/(2*j+1)

fig, (ax0, ax1, ax2) = plt.subplots(3, 1)

ax0.plot(D)

ax0.set_title('Рівні згладжування часового ряду')

fig.tight_layout()

for i in range(N):
    C[i] = Z[7][i]

ax1.plot(C)

fig.tight_layout()
```

```
ax2.pcolor(Z, cmap='plasma')

ax2.set_title('Згладжування')

fig.tight_layout()

plt.show()
```

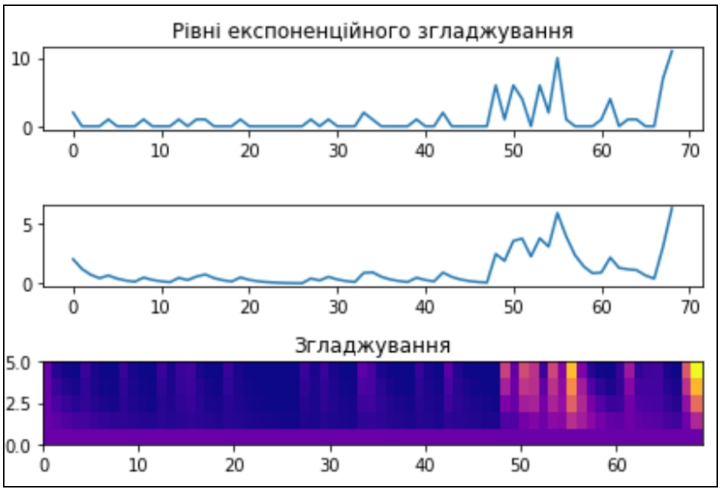


Експоненціальне згладжування

Code-2

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib.colors import LogNorm

D=[2,0,0,0,1,0,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,2,1,0,0,0,0,
      1,0,0,2,0,0,0,0,0,6,1,6,4,0,6,2,10,1,0,0,0,1,4,0,1,1,0,0,7,11]

M = 5

N = len(D)

Z = np.random.rand(M, N)

C = D
```

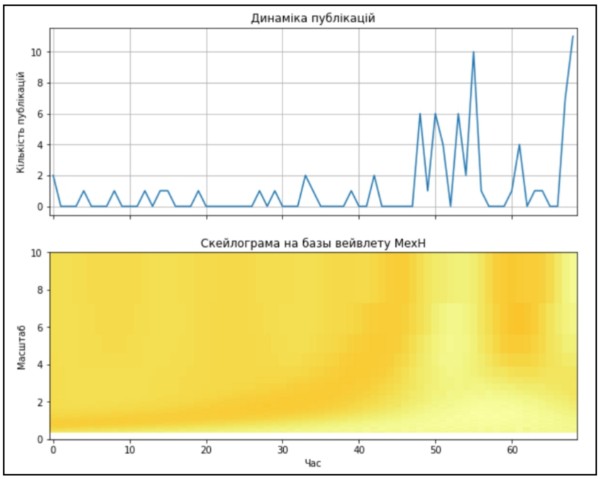
```
for i in range(N):
    for j in range(M):
        Z[j][i] = D[i]

for j in range(M):
    alf = j/M
    for k in range(1, N):
        Z[j][k] = D[k]*alf+Z[j][k-1]*(1-alf)

fig, (ax0, ax1, ax2) = plt.subplots(3, 1)

ax0.plot(D)

ax0.set_title('Рівні експоненційного згладжування')

fig.tight_layout()

for i in range(N):
    C[i] = Z[2][i]

ax1.plot(C)

fig.tight_layout()

ax2.pcolor(Z, cmap='plasma')

ax2.set_title('Згладжування')

fig.tight_layout()

plt.show()
```



Побудува вейвлет-скейлограми з іншою базовою вейвлет-функцією. Дослідження різниці вейвлет-скейлограм при різних базових вейвлет-функціях

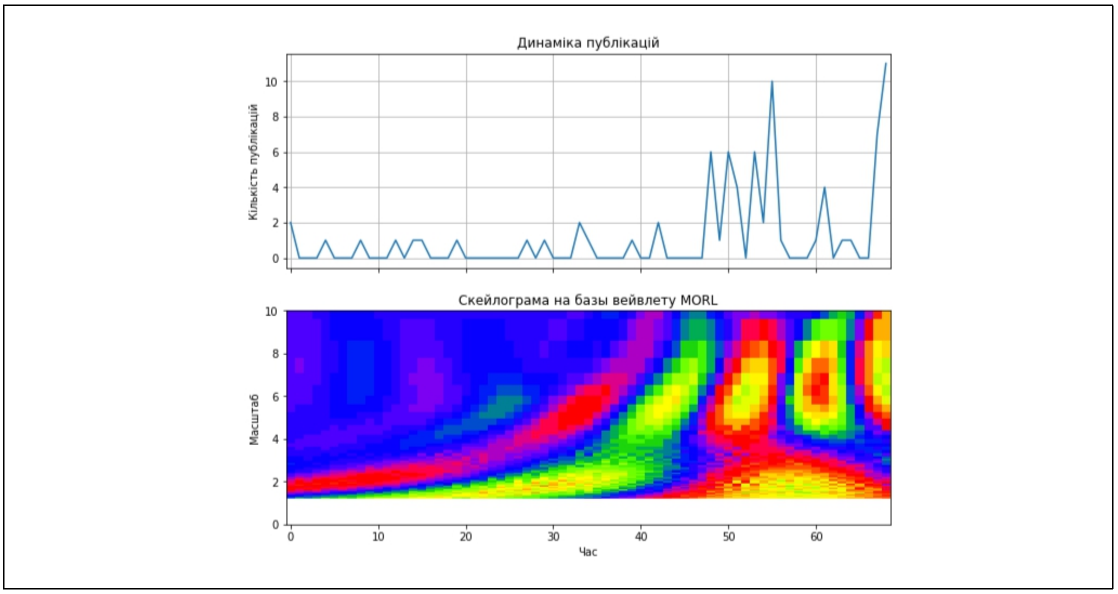
Code-lab

```
import matplotlib.pyplot as plt
import pywt

f_s = 100
# Sampling rate
x=[2,0,0,0,1,0,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,2,1,0,0,0,0,
      1,0,0,2,0,0,0,0,0,6,1,6,4,0,6,2,10,1,0,0,0,1,4,0,1,1,0,0,7,11]
N = len(x)
t = range(N)
# Visualization
fig, (ax1, ax4) = plt.subplots(2, 1, sharex=True, figsize=(10, 8))
# Signal
ax1.plot(t, x)
ax1.grid(True)
ax1.set_ylabel("Кількість публікацій")
ax1.set_title("Динаміка публікацій")
# Wavelet transform, i.e. scaleogram
cwtmatr, freqs = pywt.cwt(x, range(1, N), "mexh", sampling_period=1
/ f_s)
ax4.pcolormesh(t, freqs, cwtmatr, vmin=-100, cmap="inferno")
ax4.set_ylim(0, 10)
ax4.set_ylabel("Масштаб")
ax4.set_xlabel("Час")
ax4.set_title("Скейлограма на базы вейвлету MexH")
# plt.savefig("./fourplot.pdf")
plt.show()
```



Code-other-my-choice-function

```
import matplotlib.pyplot as plt

import pywt

f_s = 100

# Sampling rate

x=[2,0,0,0,1,0,0,0,1,0,0,0,1,0,1,1,0,0,0,1,0,0,0,0,0,0,0,1,0,1,0,0,0,2,1,0,0,0,0,
      1,0,0,2,0,0,0,0,0,6,1,6,4,0,6,2,10,1,0,0,0,1,4,0,1,1,0,0,7,11]

N = len(x)

t = range(N)

# Visualization

fig, (ax1, ax4) = plt.subplots(2, 1, sharex=True, figsize=(10, 8))

# Signal

ax1.plot(t, x)

ax1.grid(True)

ax1.set_ylabel("Кількість публікацій")

ax1.set_title("Динаміка публікацій")

# Wavelet transform, i.e. scaleogram

cwtmatr, freqs = pywt.cwt(x, range(1, N), "morl", sampling_period=1

/ f_s)

ax4.pcolormesh(t, freqs, cwtmatr, vmin=-100, cmap="prism")

ax4.set_ylim(0, 10)

ax4.set_ylabel("Масштаб")

ax4.set_xlabel("Час")

ax4.set_title("Скейлограма на базы вейвлету МехН")

# plt.savefig("./fourplot.pdf")

plt.show()
```



Питання до практичної роботи

У чому полягає суть віконного згладжування у статистичній обробці даних?

Згладжування допомагає виявити суттєві тенденції в динаміці ряду, приховавши при цьому шум і різні особливості, які проявляються при невеликих масштабах. Існують різноманітні методи згладжування. При використанні віконного згладжування, чим більше ширина інтервалу згладжування, тим більш гладкою вийде результуюча функція. Ширина інтервалу згладжування - кількість елементів, за якими розраховується середнє. Просте ковзне середнє дорівнює середньому арифметичному значенню елементів ряду з інтервалу заданої довжини, а саме:

$$S = \{s_t\}, \quad s_t = \frac{1}{w} \sum_{k=-[w/2]}^{[w/2]} d_{t+k}$$

де w – ширина інтервалу згладжування, s_t *– значення віконного ковзного середнього в точці t.*

У деяких випадках корисно розглядати більш гладку версію вихідного часового ряду $D=\{d_t\}$, *де* $t=1..N$.

Що таке експоненціальне згладжування у статистичній обробці даних?

Інший часто використовуваний метод згладжування рядів - це експоненційне згладжування. Попередні значення ряду враховуються з ваговими значеннями, що зменшуються експоненціально.

Які переваги має вейвлет-аналіз (вейвлет-перетворення) при дослідженні часових рядів і інтерпретації результатів аналізу «великих даних»?

Особливо ефективний у тих випадках, коли крім загальних спектральних характеристик потрібно виявляти локальні в часі особливості поведінки процесу, що досліджується. Основою вейвлет-аналізу є вейвлет-перетворення, яке є особливим типом лінійного перетворення, базисні функції якого (вейвлети) мають специфічні властивості. Аналіз даних з використанням вейвлет- перетворень є зручним, надійним і потужним інструментом дослідження часових рядів і дозволяє представити результати у наочному вигляді, зручному для інтерпретації. Разом з тим, усі вейвлети мають вигляд коротких хвильових пакетів з нульовим інтегральним значенням, локалізованих на часовій осі, які є інваріантними до зсуву і до масштабування.

Назвати найбільш поширені дійсні базисні вейвлет-функції?

Найбільш поширеними дійсними базисними вейвлет-функціями є:

- *Бета вейвлет*
- *Ермітів вейвлет*
- *Вейвлет Мейєра*
- *Мексиканський капелюх вейвлет*
- *Пуассоновський вейвлет*
- *Вейвлет Шеннона*
- *Сплайн вейвлет*
- *Вейвлет Стрьомберга*

Документація по вейвлет-функціям Python

https://pywavelets.readthedocs.io/en/latest/ref/