

Лабораторна робота №1
“Використання бібліотеки matplotlib для візуалізації
експериментальних даних”

Мета роботи: Ознайомитись з можливостями Google Colab та бібліотеки matplotlib для візуалізації експериментально одержаних даних.

Бібліотека matplotlib - бібліотека на мові програмування Python створена для візуалізації даних. Використовуючи **matplotlib** можна створювати високоякісні рисунки різних форматів. Matplotlib є гнучким, легко конфігурованим пакетом, який разом з NumPy, SciPy і IPython надає можливості, подібні до MATLAB. Перевагою є безкоштовність, сумісність з іншими бібліотеками Python та простота використання. [1]

Важливою концепцією у бібліотеці matplotlib є концепція ієрархії об'єктів. “Ієрархія” у нашому випадку означає, що існує певна древовидна структура об'єктів бібліотеки matplotlib, які лежать в основі побудови кожного графіка.

Найвищий в ієрархії об'єктів - об'єкт Figure контейнер для графіки, який може вміщувати багато об'єктів Axes (позначає окремий графік або діаграму). Далі знаходяться менші об'єкти, такі як окремі лінії, легенда, текст та підписи вісей (рис.1). Майже будь який елемент графіку є об'єктом, яким можна керувати.

Інтерфейс **matplotlib.pyplot** дозволяє суттєво спростити роботу об'єктами та використовувати для побудови графіків високорівневі функції.

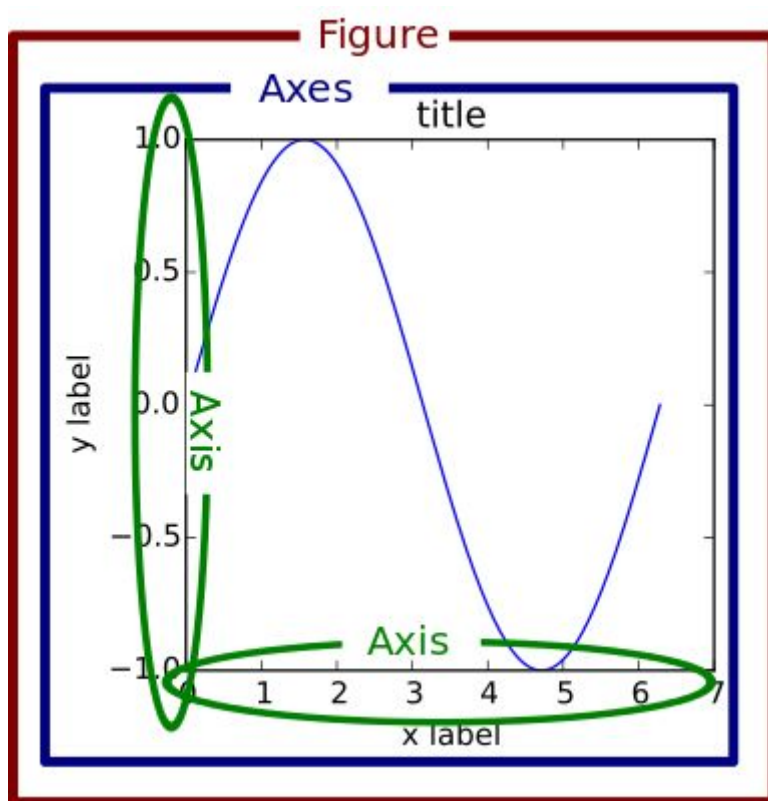


Рис.1. Ієрархія об'єктів в matplotlib

Робота з matplotlib

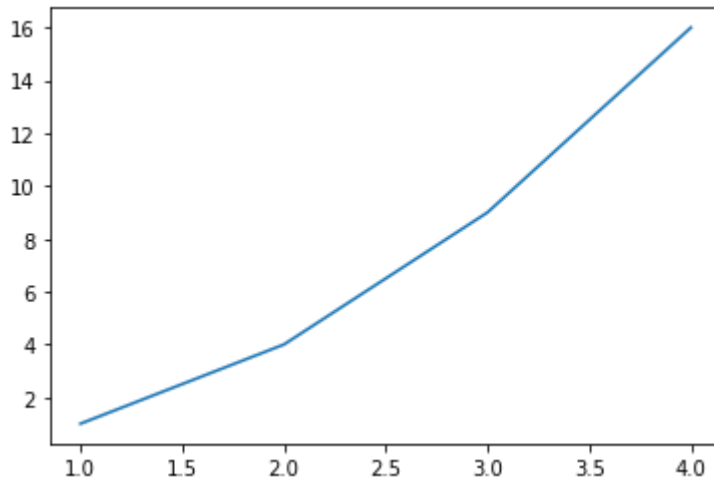
Для початку роботи слід підключити інтерфейс **pyplot**

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

для побудови графіка по точкам можна використати
процедуру `pyplot.plot(X,Y)`

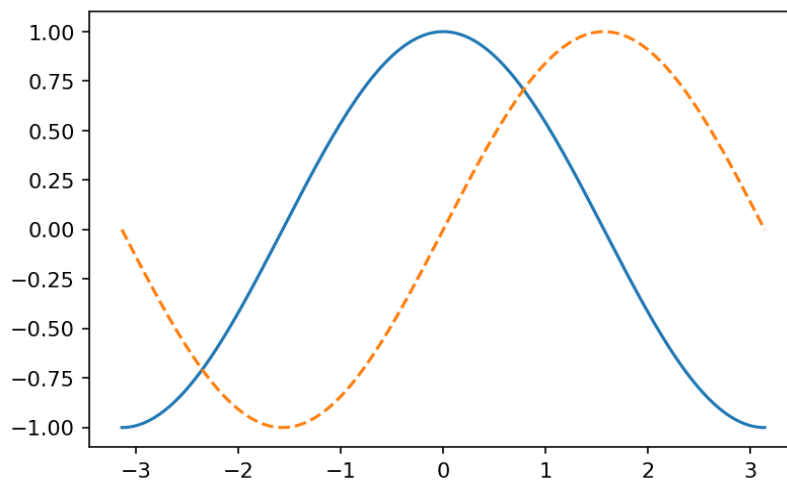
Приклад

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.show()
```



Оскільки, найчастіше доводиться працювати з великими масивами даних для роботи з ними слід використовувати бібліотеку NumPy. Приклад використання та результат роботи коду:

```
import numpy as np (підключаємо бібліотеку numpy)
matplotlib.pyplot.figure(dpi=200) / обираємо якість зображення
X = np.linspace(-np.pi, np.pi, 256) / задаємо кількість точок та
інтервал
C, S = np.cos(X), np.sin(X) (будуємо функцію cos та sin )
plt.plot(X,C)
plt.plot(X,S, '--',color='purple', linewidth=10) ( )
plt.show()
```



Для створення більш інформативних графіків та їх модифікації слід використовувати наступні функції:

pyplot.xlabel (“підпис”) задає підпис вісі x

pyplot.ylabel (“підпис”) задає підпис вісі y

pyplot.axis([x_min, x_max, y_min, y_max,]) - задає діапазон значень, що відображається на рисунку

pyplot.legend() - додаємо легенду до рисунка, у цьому слід дещо змінити функцію **plot(... label='позначення')**

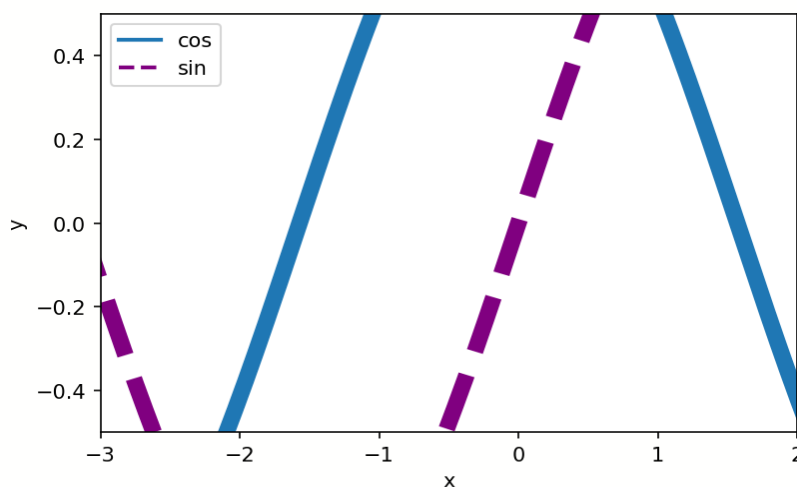
Також за потреби розмістити на одному рисунку декілька графіків можна змінювати параметри кожного окремого графіка(ширина, колір та вид лінії, наявність чи відсутність маркерів таке інше) в функції plot.

Приклад коду та результати його виконання :

```
import numpy as np
plt.figure(dpi=150) / задаємо якість рисунку
X = np.linspace(-np.pi, np.pi, 256)
C, S = np.cos(X), np.sin(X)

plt.xlabel('x')
plt.ylabel('y')
plt.axis([-3, 2, -0.5, 0.5,])
plt.plot(X, C, label='cos', linewidth=8)
plt.plot(X, S, '--', label='sin', color='purple', linewidth=8)/
змінили ширину лінії, колір та додали ярлик для легенди
leg = plt.legend()

leg.get_lines()[0].set_linewidth(2)
leg.get_lines()[1].set_linewidth(2)
plt.show()
```



Часто виникає потреба комбінувати декілька рисунків, найпростіше це реалізовувати використовуючи **pyplot.subplots()**.

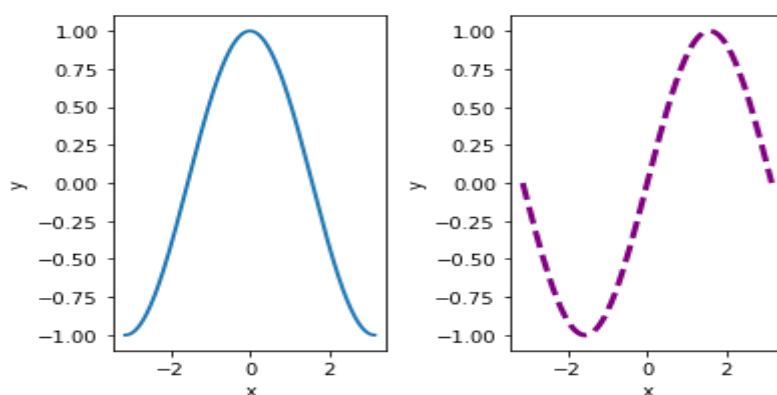
Можливим є декілька підходів до створення композиції декількох рисунків:

1) створювати об'єкти класу `Axes` та додавати їх до раніше створеного об'єкту `Figure` функцією `figure.add_subplot(кількість рядків, кількість стовпців, номер рисунку)`, при цьому нумерація рисунків відбувається з лівого верхнього кута по рядкам. Після цього окремо модифікувати об'єкт класу `Axes` для отримання потрібного результату.

Приклад

```
fig = plt.figure()
X = np.linspace(-np.pi, np.pi, 256)
C, S = np.cos(X), np.sin(X)

ax_1 = fig.add_subplot(1, 2, 1)
ax_2 = fig.add_subplot(1, 2, 2)
ax_1.set_xlabel('x')
ax_2.set_ylabel('y')
ax_2.set_xlabel('x')
ax_1.set_ylabel('y')
ax_1.plot(X, C, label='cos', linewidth=2)
ax_2.plot(X, S, '--', label='sin', color='purple', linewidth=3)
plt.subplots_adjust(wspace = 0.35) / змінюємо відстань між окремими графіками
plt.show()
```



2) використати функцію `plt.subplots(nrows=, ncols=)`, що створює об'єкт класу `Figure` та масив об'єктів класу `Axes`, а потім працюючи з кожним з цих об'єктів окремо, звертаючись до них як до елементів масиву.

Аналогічний результат можна одержати замінивши у попередньому прикладі

`fig, axes = plt.subplots(nrows = 1, ncols =2)` та замінивши `ax_1` `ax_2` на `axes[0]` `axes[1]`.

Слід також відзначити, що можна задати всі параметри одночасно використовуючи функцію `Axes.set()`, зокрема у попередньому прикладі можна замінити

```
ax_1.set_xlabel ('x')
```

```
ax_1.set_ylabel ('y')
```

на

```
ax_1.set(xlabel= 'x', ylabel ='y')
```

Завдання

- 1) Використовуючи `google colab` зчитати експериментальні фотоemisійні спектри з файлу формату `csv` за допомогою функції з бібліотеки `Numpy`
`genfromtxt('Table1.csv', delimiter=',')`
- 2) За допомогою `matplotlib` побудувати на одному рисунку графіки залежності для 3 різних значень енергії, оформити рисунок відповідно до вимог для наукових публікацій.
- 3) Використовуючи `pyplot.subplots()` побудувати набір з 3 рисунків залежність при сталому куті, при сталій енергії та весь спектр цілком.
- 4) Зберегти усі рисунки.

Використані джерела

1. <https://realpython.com/python-matplotlib-guide/>
2. https://pyprog.pro/mpl/mpl_main_components.html
3. <https://matplotlib.org/3.3.3/api>