Lucrarea 5. Librăria Matplotlib

A. Obiectivele lucrării

1. Familiarizarea cu funcționalitatea și sintaxa librăriei ***Matplotlib.***

B. Introducere

Matplotlib este o librărie ***Python*** utilizată pentru generarea și afișarea graficelor. Această librărie conține un sub-modul numit ***pyplot*** ce oferă un mod de lucru și o sintaxă asemănătoare MATLAB-ului [6]. Întrucât vom folosi doar sub-modulul ***pyplot***, îl vom importa doar pe acesta putându-i da și un alias. În mod uzual se folosește aliasul `plt`:

main.py

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

C. Reprezentări grafice

Pentru reprezentări grafice ale datelor, ***Matplotlib*** pune la dispoziție o multitudine de funcții, din care, cea mai uzuală este funcția `plt.plot` ce are o sintaxă asemănătoare cu funcția `plot` din MATLAB. Astfel, această funcție primește ca parametrii două matrice unidimensionale reprezentând valorile de pe abscisă, respectiv valorile de pe ordonată, și, opțional, un parametru de formatare de tip șir de caractere ce poate specifica atât culoarea liniei, cât și tipul acesteia și tipul de marcator:

Tabelul 1 – Opțiuni de formatare a graficelor realizate cu `plt.plot` [7]

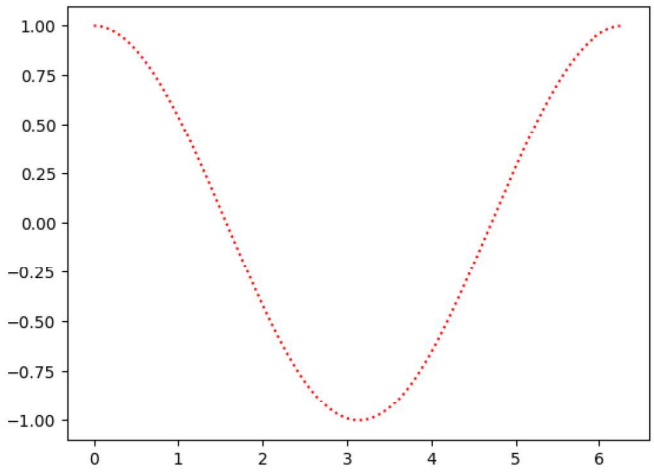
Culoare		Tip linie		Tip marcaj			
Simbol	**Descriere**	**Simbol**	**Descriere**	**Simbol**	**Descriere**	**Simbol**	**Descriere**
y	galben	-	continuă	+	plus	^	triunghi ▲
m	magenta	--	întreruptă	o	cerc	v	triunghi ▼
c	turcoaz	:	punctată	*	steluță	>	triunghi ▶
r	roșu	-.	linie-punct	.	punct	<	triunghi ◀
g	verde			x	x	p	pentagon
b	albastru			s	pătrat	h	hexagon
w	alb			d	romb		
k	negru						

După ce s-au adăugat toate elementele necesare pe grafic, pentru a putea vedea rezultatul, trebuie apelată funcția `plt.show()`. **Atenție**, codul scris după apelarea funcției `plt.show()` se va executa doar după închiderea graficului.

main.py

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 2 * np.pi, 0.01)
y = np.cos(x)

plt.plot(x, y, "r:")
plt.show()
```

rezultat



Afișarea mai multor grafice, se poate face fie în ferestre diferite, fie în cadrul aceleiași ferestre. Pentru primul caz, este necesară apelarea funcției `plt.figure(n)` (unde `n` reprezintă numărul figurii / ferestrei) înainte de apelarea funcției `plt.plot`. În cel de-al doilea caz, pentru a afișa mai multe grafice în aceiași fereastră există trei posibilități [2]:

1. se poate apela funcția `plt.plot` de mai multe ori, pentru fiecare set de date ce se dorește reprezentat grafic; nu este necesară utilizarea unui echivalent pentru `hold on`.
2. se pot da mai mulți parametrii funcției `plt.plot` [6]:
   a. `plt.plot(x1, y1, x2, y2, ...)`
   b. `plt.plot(x1, y1, f1, x2, y2, f2, ...)`
3. se poate utiliza funcția `plt.subplot(m, n, p)` ce va împărți fereastra într-o matrice de sub-ferestre dispuse pe `m` rânduri și `n` coloane, și va selecta sub-fereastra cu numărul `p` pentru a afișa ulterior în ea. Numerotarea sub-ferestrelor se face la fel ca în MATLAB, începând cu numărul 1, de la stânga la dreapta și de sus în jos.

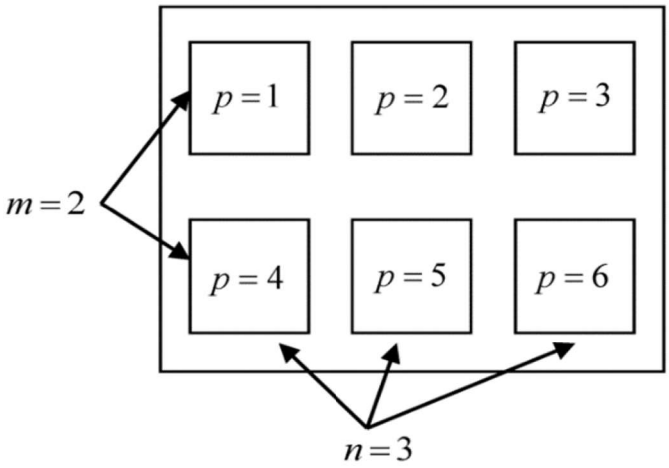
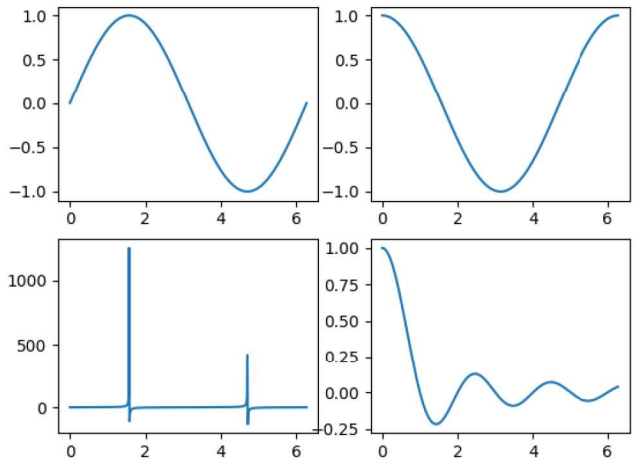


Figura 1 – Exemplu de împărțire și numerotare a sub-ferestrelor [7]

main.py

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 2 * np.pi, 0.01)

y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)
y4 = np.sinc(x)

plt.subplot(2, 2, 1)
plt.plot(x, y1)
plt.subplot(2, 2, 2)
plt.plot(x, y2)
plt.subplot(2, 2, 3)
plt.plot(x, y3)
plt.subplot(2, 2, 4)
plt.plot(x, y4)

plt.show()
```

rezultat



Asemănător mediului MATLAB, ***Matplotlib*** permite reprezentarea grafică și în coordonate logaritmice sau semilogaritmice:

Tabelul 2 – Funcții pentru reprezentarea în coordonate logaritmice și semilogaritmice

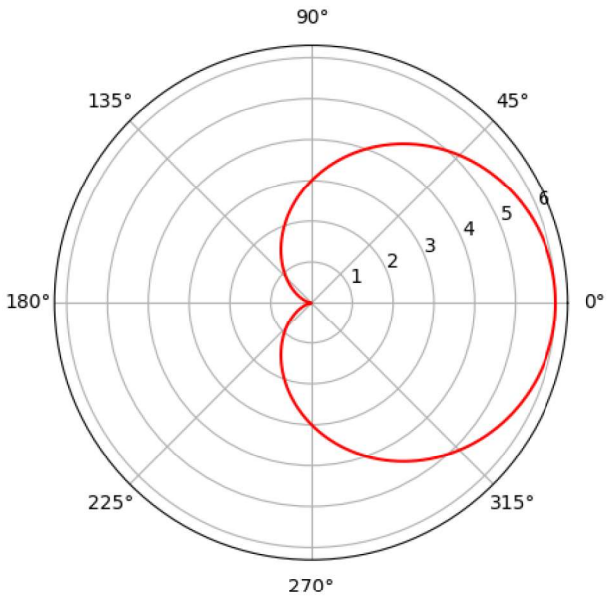
Funcția	**Descriere**
`plt.semilogx(x, y)`	grafic cu axa `x` logaritmică
`plt.semilogy(x, y)`	grafic cu axa `y` logaritmică
`plt.loglog(x, y)`	grafic cu ambele axe logaritmice

Matplotlib mai permite și reprezentarea grafică în coordonate polare utilizând funcția `plt.polar(theta, r)`, unde `theta` și `r` sunt matrice unidimensionale specificând unghiurile, respectiv distanțele [6].

main.py

```
import numpy as np
import matplotlib.pyplot as plt

theta = np.arange(0, 2 * np.pi, 0.01)
r = 3 * (np.cos(theta) + 1)

plt.polar(theta ,r, 'r')
plt.show()
```

rezultat



Alte funcții specializate pentru reprezentări grafice:

Tabelul 3 – Funcții specializate pentru reprezentări grafice [6]

Funcția	**Descriere**
`plt.scatter(x, y)`	reprezentare grafică ca puncte
`plt.bar(x, y)`	reprezentare grafică cu bare
`plt.barh(y, x)`	reprezentare grafică cu bare orizontale
`plt.fill(x, y)`	reprezentare grafică ca poligon
`plt.stem(x, y)`	reprezentare grafică ca eșantioane
`plt.step(x, y)`	reprezentare grafică în trepte
`plt.pie(x, labels)`	reprezentare grafică tip „*pie chart*”
`plt.errorbar(x, y, err)`	reprezentare grafică cu erori

D. Adnotarea graficelor

Totuși, în inginerie, în marea majoritate a cazurilor este necesar ca pe grafic să apară mai multe detalii despre reprezentare, așa că ***Matplotlib*** implementează o serie de funcții pentru cosmetizarca graficului. Dintre aceste, cele mai importante sunt descrise în tabelul următor [6]:

Tabelul 4 – Opțiuni de cosmetizare a graficelor realizate cu `plt.plot` [7]

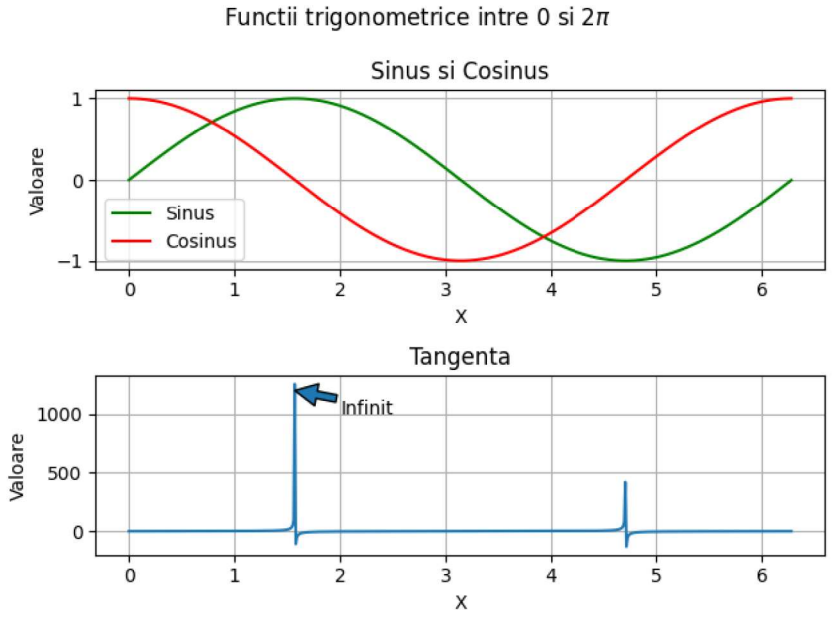
Funcția	**Descriere**
`plt.title(s)`	setează titlul graficului la valoarea șirului de caractere `s`
`plt.suptitle(s)`	setează titlul superior al graficului la valoarea șirului de caractere `s`
`plt.xlabel(s)`	setează eticheta axei `x` la valoarea șirului de caractere `s`
`plt.ylabel(s)`	setează eticheta axei `y` la valoarea șirului de caractere `s`
`plt.grid()`	adaugă grila pe grafic
`plt.xlim(xmin, xmax)`	setează intervalul de afișare al axei `x`
`plt.ylim(ymin, ymax)`	setează intervalul de afișare al axei `y`
`plt.legend(lista)`	adaugă o legendă, fiecărui grafic atribuindu-i-se o etichetă din lista primită ca parametru
`plt.text(x, y, s)`	adaugă textul `s` la coordonatele (`x`, `y`)
`plt.annotate(s, xy, xy2, arrowprops = {})`	adaugă textul `s` la coordonatele specificate de tuplul `xy2` și săgeată spre coordonatele date de tuplul `xy`

Pentru că este posibil ca atunci când adăugăm noi elemente graficului (etichete, titluri, etc) acestea să se suprapună cu graficul sau între ele, se poate apela funcția `plt.tight_layout()` pentru reașezarea automată a elementelor.

main.py

```
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0, 2 * np.pi, 0.01)

y1 = np.sin(x)
y2 = np.cos(x)
y3 = np.tan(x)

plt.subplot(2, 1, 1)
plt.plot(x, y1, 'g', x, y2, 'r')
plt.xlabel("X")
plt.ylabel("Valoare")
plt.title("Sinus si Cosinus")
plt.grid()
plt.legend(["Sinus", "Cosinus"])


plt.subplot(2, 1, 2)
plt.plot(x, y3)
plt.xlabel("X")
plt.ylabel("Valoare")
plt.title("Tangenta")
plt.annotate("Infinit", (np.pi / 2, 1200), (2, 1000), arrowprops={})

plt.grid()

plt.suptitle("Functii trigonometrice intre 0 si 2$\pi$ ")
plt.tight_layout()
plt.show()
```

rezultat



Toate funcțiile pentru cosmetizarea graficelor ce primesc ca parametru un șir de caractere permit utilizare, în șirul de caractere, a sintaxei TeX între două simboluri `$`. La adresa https://matplotlib.org/stable/tutorials/text/mathtext.html se poate consulta o descriere detaliată a acestei sintaxe.

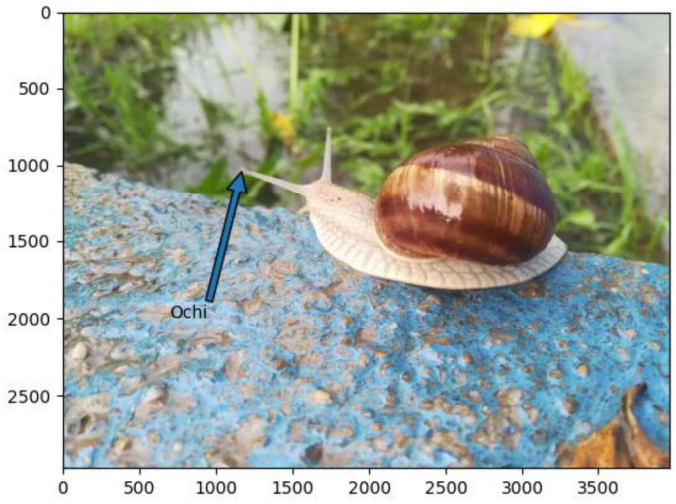
E. Afișarea imaginilor

Matplotlib implementează, în sub-modulul ***pyplot***, funcția `imread` ce primește ca parametru calea spre o imagine și returnează o matrice ***NumPy*** cu eșantioanele imaginii și funcția `imshow` ce primește ca parametru o astfel de matrice și o afișează grafic.

main.py

```
import numpy as np
import matplotlib.pyplot as plt

x = plt.imread("melc.jpg")
plt.imshow(x)
plt.annotate("Ochi", (1170, 1040), (700, 2000), arrowprops={})
plt.show()
```

rezultat



F. Cerințe

1. Pentru $x \in [-5 \quad 5]$, să se reprezinte pe același grafic (suprapuse) funcțiile:
   a) $y_1(x) = \frac{x+1}{x^2-2}$ cu linie roșie continuă
   b) $y_2(x) = x \cdot \log_2(x^2)$ cu linie verde întreruptă

2. Pentru $x \in [0 \quad 8 \cdot \pi]$, să se reprezinte în sub-ferestre diferite funcțiile:
   c) $y_3(x) = \sin(1.5 \cdot x)$
   d) $y_4(x) = \cos(x/2)$

3. Pornind de la ***DataFrame***-ul `df1` rezultat în urma rezolvării cerinței numărul 6 din lucrarea anterioară (Lucrarea 4), rezolvați:

   a) Afișați grafic suprapus evoluția temperaturii minime, medie și maxime pentru stația „*Botoșani*”. Adăugați etichete axelor, titlu graficului și legendă. **Indiciu**: filtrați `df1` astfel încât să rămâneți doar cu valorile înregistrate la acea stație, apoi afișați grafic, pe rând, coloanele `TMIN`, `TMED` și `TMAX`, utilizând pentru axa `x` coloana `DATCLIM`.

   b) Afișați grafic, sub formă de bare orizontale, temperatura minimă înregistrată de fiecare din stațiile meteo. Adăugați etichete axelor și titlu graficului. **Indiciu**: selectați din `df1` coloanele `CODST`, `NUME` și `TMIN`, grupați după coloana `CODST` și calculați minimul. Din rezultat utilizați coloanele `NUME` și `TMIN` pentru afișarea grafică.

   c) Afișați un grafic de tip scatter cu altitudinea pe axa `x` și temperatura minimă înregistrată de fiecare stație pe axa `y`. Adăugați etichete axelor și titlu graficului. Adnotați fiecare punct al graficului cu numele stației. **Indiciu**: selectați din `df1` coloanele `CODST`, `NUME`, `ALT` și `TMIN`, grupați după coloana `CODST` și calculați minimul. Din rezultat utilizați coloanele `ALT` și `TMIN` pentru afișarea grafică. Treceți prin toate rândurile rezultatului și utilizați coloana `NUME` pentru adnotarea graficului.