

Lucrarea 3. Librăria NumPy

A. Obiectivele lucrării

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

B. Introducere

NumPy (prescurtare de la *Numerical Python*) este o librărie **Python** care oferă posibilitatea de a defini și utiliza matrice multi-dimensionale, oferind de asemenea și o serie de funcții de nivel înalt pentru a lucra cu acestea (funcții pentru manipularea matricelor, funcții pentru generarea matricelor, funcții de algebră liniară, etc) [3].

După instalarea librăriei **NumPy** (așa cum s-a văzut în prima lucrare), pentru a fi folosită într-un program, aceasta trebuie importată scriind la începutul programului instrucțiunea `import` urmată de numele librăriei (`numpy`). Funcțiile și constantele definite în această librărie pot fi accesate scriind numele librăriei (`numpy`), simbolul punct și apoi numele funcției sau constantei dorite. Totuși, pentru a economisi timp, putem să utilizăm un alias mai scurt pentru numele librăriei, completând instrucțiunea `import` cu un alt cuvânt cheie, `as`, urmat de alias-ul dorit. În mod uzual, pentru librăria **NumPy** se folosește alias-ul `np`:

main.py

```
import numpy as np
```

O matrice **NumPy** n-dimensională este o grilă de valori de același tip, putându-se inițializa utilizând funcția `np.array` căreia îi dăm ca parametru liste imbricate (ex.: o matrice bidimensională se poate inițializa cu o listă de liste). Accesarea elementelor se face utilizând tot paranteze drepte între care se pune indexul / indecșii elementului, numărarea indecșilor începând de la 0 [3].

main.py

```
import numpy as np

# matrice unidimensională
X = np.array([3, 2, 1])
print(X)
print(X[1])

# matrice bidimensională
Y = np.array([[3, 2, 1], [6, 5, 4]])
print(Y)
print(Y[0, 1])
```

rezultat

```
[3 2 1]
2
[[3 2 1]
 [6 5 4]]
2
```

Un al doilea parametru care se poate da funcției `np.array` este tipul de date al elementelor. Dacă acesta lipsește, **NumPy** îi va stabili valoarea ca fiind tipul de date cu dimensiune minimă ce poate memora toate elementele date. Tipul de date al elementelor poate lua una din valorile [3]:

Tabelul 1 - Tipuri de date în NumPy

Data Type ¹	Descriere
?	Boolean
b	Signed byte
B	Unsigned byte
i	Signed integer
u	Unsigned integer
f	Floating point
c	Complex floating point
U	Unicode string

Pentru o matrice **NumPy** deja existentă, tipul de date se poate obține accesând proprietatea `dtype`. Utilizând metoda `astype`, valorile unei matrice deja existentă se pot pune într-o altă matrice de alt tip. Exemplu:

main.py

```
import numpy as np

x = np.array([1, 2, 3])
print(f"{x} - {x.dtype}")

x = np.array([1, 2, 3], 'f8')
print(f"{x} - {x.dtype}")

y = x.astype('c8')
print(f"{y} - {y.dtype}")
```

rezultat

```
[1 2 3] - int32
[1. 2. 3.] - float64
[1.+0.j 2.+0.j 3.+0.j] - complex64
```

¹ tipurile i, u, f și c pot fi urmate și de numărul de byți utilizati la stocarea elementelor (ex: i8 este unsigned integer pe 8 byți – int64)

C. Generarea matricelor

a) Generarea matricelor unidimensionale

NumPy oferă trei funcții utile pentru generarea de secvențe de numere sub forma unor matrice unidimensionale [3]:

Tabelul 2 - Funcții pentru generarea secvențelor de numere

Funcția	Descriere
np.arange(amin, amax, step)	generează o matrice unidimensională ale cărei valori vor reprezenta o secvență de numere de la amin la amax cu pasul step. Dacă step lipsește atunci pasul va fi implicit 1. Atenție: elementul cu valoarea amax nu va fi inclus în secvență
np.linspace(amin, amax, N)	generează o matrice unidimensională ale cărei valori vor reprezenta o secvență de N numere egal distanțate în intervalul [amin, amax]
np.logspace(amin, amax, N)	generează o matrice unidimensională ale cărei valori vor reprezenta o secvență de N numere egal distanțate pe scara logaritmică în intervalul $[10^{amin}, 10^{amax}]$

Exemple:

main.py
<pre>import numpy as np print(np.arange(1, 11, 2)) print(np.linspace(0, 10, 3)) print(np.logspace(0, 3, 4))</pre>
rezultat
<pre>[1 3 5 7 9] [0. 5. 10.] [1.e+00 1.e+01 1.e+02 1.e+03]</pre>

b) Generarea matricelor multidimensionale

NumPy oferă o serie de funcții utile pentru generarea de matrice multidimensionale uzuale [3]:

Tabelul 3 - Funcții pentru generarea de matrice uzuale

Funcția	Descriere	Exemplu	Rezultat
np.zeros(t)	generează o matrice cu toate elementele având valoarea 0, ale cărei dimensiuni sunt specificate de tuplul t	np.zeros((2, 2))	[[0. 0.] [0. 0.]]
np.ones(t)	generează o matrice cu toate elementele având valoarea 1, ale cărei dimensiuni sunt specificate de tuplul t	np.ones((1, 2))	[[1. 1.]]
np.full(t, z)	generează o matrice cu toate elementele având valoarea z, ale cărei dimensiuni sunt specificate de tuplul t	np.full((3, 2), 7)	[[7 7] [7 7] [7 7]]
np.eye(x)	generează o matrice identitate cu x linii și x coloane	np.eye(2)	[[1. 0.] [0. 1.]]
np.diag(x)	dacă x este unidimensional, va genera o matrice bidimensională cu valorile din x pe diagonala principală, și restul elementelor 0	np.diag([3, 2])	[[3. 0.] [0. 2.]]

D. Indexarea elementelor matricelor

Indexarea elementelor unei matrice unidimensionale **NumPy** se poate face în mai multe moduri, conform tabelului [3]:

Tabelul 4 - Moduri de indexare ale matricilor NumPy

Indexare	Descriere
X[i]	selectează elementul cu indexul i din matricea X
X[i:j]	selectează elementele cu indecsii de la i la j - 1 din matricea X
X[:i]	selectează elementele cu indecsii de la 0 la i - 1 din matricea X
X[j:]	selectează elementele cu indecsii de la j la ultimul din matricei X
X[[i, j, k]]	selectează elementele cu indecsii i, j și k din matricea X
X[:]	selectează toate elementele din matricea X

În cazul matricelor multidimensionale, indecsii aferenți fiecarei dimensiuni se vor pune între aceleasi paranteze pătrate, separați de virgulă. Pentru indecsii fiecarei dimensiuni se pot utiliza toate metodele prezentate în tabelul anterior. Spre exemplu, pentru o matrice

bidimensională X , dacă dorim să selectăm elementele care se află pe primele 3 linii (toate coloanele), se va utiliza: $A[:3, :]$.

Pe lângă indexarea utilizând indecșii elementelor, **NumPy** mai permite și indexarea booleană, cu ajutorul căreia se pot selecta elementele unei matrice care satisfac o anumită condiție. Spre exemplu:

main.py	
<pre>import numpy as np A = np.array([[8, 7, 6, 1], [5, 4, 3, 7], [2, 1, 0, 4]]) print(A[A > 2])</pre>	
rezultat	[8 7 6 5 4 3 7 4]

E. Operații matematice

a) Operații aritmetice

Operațiile aritmetice uzuale (adunare, scădere, înmulțire, împărțire și ridicare la putere) se pot efectua:

1. între două matrice **NumPy** – operația se va efectua element cu element; **atenție**: este necesar ca matricele să aibă aceleași dimensiuni, altfel se va returna o eroare.
2. între o matrice **NumPy** și un scalar – operația se va realiza între fiecare element al matricei și scalarul respectiv

Exemple:

main.py	
<pre>import numpy as np A = np.array([[1,2],[3,4]]) B = np.array([[5,6],[7,8]]) print(A + B) # adunare element cu element print(A - B) # scadere element cu element print(A * B) # inmultire element cu element print(A / B) # impartire element cu element print(A ** B) # ridicare la patrat element cu element print(A + 5) # se adaugă valoarea 5 fiecarui element print(A * 5) # se multiplică cu 5 fiecare element</pre>	

b) Funcții matematice

Întrucât funcțiile matematice disponibile în librăria **math** care au fost prezentate în prima lucrare se pot folosi doar pentru scalari, **NumPy** pune la dispoziție propriul set de funcții matematice care se aplică asupra tuturor elementelor unei matrice [3]:

Tabelul 5 - Funcții matematice din NumPy

Funcții pentru reprezentarea numerelor	
np.ceil (X)	Rotunjirea tuturor elementelor la cel mai mic întreg mai mare
np.floor (X)	Rotunjirea tuturor elementelor la cel mai mare întreg mai mic
np.round (X, d)	Rotunjirea tuturor elementelor matricei X la d zecimale
np.absolute (X)	Modulul tuturor elementelor matricei
Funcții exponențiale și logaritmice	
np.exp (X)	e la puterea fiecărui element din X
np.log (X)	Logaritmul natural al fiecărui element al matricei
np.log2 (X)	Logaritmul în bază 2 al fiecărui element al matricei
np.log10 (X)	Logaritmul în bază 10 al fiecărui element al matricei
np.sqrt (X)	Rădăcina pătrată a fiecărui element al matricei
Funcții trigonometrice	
np.sin (X)	Sinusul fiecărui element al matricei (în radiani)
np.cos (X)	Cosinusul fiecărui element al matricei (în radiani)
np.tan (X)	Tangenta fiecărui element al matricei (în radiani)
np.arcsin (X)	Arc sinusul în radiani al fiecărui element al matricei (rezultatul între $-\pi/2$ și $\pi/2$)
np.arccos (X)	Arc cosinusul în radiani al fiecărui element al matricei (rezultatul între 0 și π)
np.arctan (X)	Arc tangenta în radiani a fiecărui element al matricei (rezultatul între $-\pi/2$ și $\pi/2$)
np.arctan2 (B, A)	Arc tangenta în radiani a fiecărui element al matricei B/A (rezultatul între $-\pi$ și π)
Funcții de conversie unghiulară	
np.rad2deg (X)	Convertește valorile matricei din radiani în grade
np.deg2rad (X)	Convertește valorile matricei din grade în radiani
Alte funcții	
np.minimum (X, Y)	Returnează o matrice calculând minimul valorilor din X și Y element cu element
np.maximum (X, Y)	Returnează o matrice calculând maximul valorilor din X și Y element cu element
np.amin (X, axis)	Dacă parametrul axis lipsește va returna valoarea minimă din matrice. Altfel, va returna o matrice cu valorile minime pe axa axis a matricei (axis = 0 – minimul fiecărei coloane; axis = 1 – minimul fiecărui rând)

np.amax(X, axis)	Dacă parametrul axis lipsește va returna valoarea maxima din matrice. Altfel, va returna o matrice cu valorile maxime pe axa axis a matricei
np.sum(X, axis)	Dacă parametrul axis lipsește va returna suma tuturor elementelor matricei. Altfel, va returna o matrice cu suma elementelor de pe axa axis a matricei
np.prod(X, axis)	Dacă parametrul axis lipsește va returna produsul tuturor elementelor matricei. Altfel, va returna o matrice cu produsul elementelor de pe axa axis a matricei

Pe lângă funcțiile uzuale, **NumPy** oferă și două constante matematice de bază: np.pi (constanta matematică π) și np.e (numărul lui Euler).

Astfel, dacă dorim să calculăm cosinusul tuturor elementelor unei matrice unidimensionale ce conține 10 valori de la 0 la $\frac{\pi}{2}$ vom proceda astfel:

main.py	<pre>import numpy as np A = np.linspace(0, np.pi / 2, 10) print(np.sin(A))</pre>
rezultat	<pre>[0. 0.17364818 0.34202014 0.5 0.64278761 0.76604444 0.8660254 0.93969262 0.98480775 1.]</pre>

c) Algebră liniară

Librăria **NumPy** permite utilizarea matricelor în rezolvarea problemelor de algebră liniară oferind o serie de funcționalități și funcții dedicate. Astfel, putem determina transpusa unei matrice accesând proprietatea T și putem înmulți două matrice în sens algebraic utilizând operatorul @. Exemplu:

main.py	<pre>import numpy as np X = np.array([[1, 2, 3], [4, 5, 6]]) print(X @ X.T) # X înmultit cu transpusa acestuia</pre>
rezultat	<pre>[[14 32] [32 77]]</pre>

Marea majoritate a funcțiilor dedicate algebrei liniare se află în sub-modulul `linalg`. Cele mai uzuale dintre acestea sunt descrise în tabelul următor [3]:

Tabelul 6 - Funcții uzuale de algebră liniară în NumPy

Funcția	Descriere
<code>np.linalg.det(X)</code>	Calculează determinantul matricei X
<code>np.linalg.inv(X)</code>	Calculează inversa matricei X
<code>np.linalg.matrix_rank(X)</code>	Calculează rangul matricei X
<code>np.linalg.svd(X)</code>	Descompunerea pe valori singulare a matricei X
<code>np.linalg.eig(X)</code>	Calculează valorile și vectorii proprii ai matricei X
<code>np.linalg.solve(A, B)</code>	Calculează soluțiile sistemelor de ecuații scrise în formă matriceală: $A \cdot x = B$

Exemplu de calcul al determinantului și al inversei unei matrice:

main.py

```
import numpy as np

X = np.array([[1, 2], [3, 4]])

print(np.linalg.det(X))
print(np.linalg.inv(X))
```

rezultat

```
-2.0000000000000004
[[-2.    1. ]
 [ 1.5 -0.5]]
```

Exemplu de rezolvare a unui sistem de ecuații $\begin{cases} 2 \cdot x_1 - 3 \cdot x_2 = 5 \\ -x_1 + 2 \cdot x_2 = 8 \end{cases}$ utilizând algebra liniară.

În primul rând va trebui să scrie sistemul de ecuații în forma matriceală:

$$\begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix} \text{ și vom nota } A = \begin{bmatrix} 2 & -3 \\ -1 & 2 \end{bmatrix} \text{ și } B = \begin{bmatrix} 5 \\ 8 \end{bmatrix}. \text{ Atunci:}$$

main.py

```
import numpy as np

A = np.array([[2, -3], [-1, 2]])
B = np.array([[5], [8]])
print(np.linalg.solve(A, B))
```

rezultat

```
[ [34.]
 [21.]]
```

O altă funcție utilă definită de **NumPy** este funcția `np.roots(P)` care calculează rădăcinile polinomului ai căruia coeficienți sunt definiți de matricea unidimensională `P`. Astfel, pentru a calcula rădăcinile polinomului $x^2 - 5 \cdot x + 6$, matricea `P` va fi $[1, -5, 6]$:

main.py

```
import numpy as np

print(np.roots(np.array([1, -5, 6])))
```

Rezultat

```
[3. 2.]
```

F. Alte funcționalități utile

Pentru a determina dimensiunea unei matrice, se va accesa proprietatea `shape` a acesteia. Această proprietate este un tuplu ce specifică dimensiunile matricei. Funcția `np.reshape` ne permite să schimbăm formă unei matrice. Exemplu:

main.py

```
import numpy as np

A = np.array([[1, 2, 3, 4, 5, 6], [7, 8, 9, 10, 11, 12]])
print(A)
print(A.shape)

B = np.reshape(A, (3, 4))
print(B)
print(B.shape)
```

Rezultat

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]
(2, 6)
[[ 1  2  3  4]
 [ 5  6  7  8]
 [ 9 10 11 12]]
(3, 4)
```

Concatenarea a două sau mai multe matrice se va face cu funcțiile `np.hstack` (concatenare pe orizontală a acestora) și `np.vstack` (concatenarea pe verticală a acestora). Aceste funcții vor primi ca parametru un tuplu ce conține matricele ce se doresc concatenate. Exemplu:

main.py

```
import numpy as np

X = np.array([1, 2])
Y = np.array([3, 4])

A = np.hstack((X, Y))
print(A)

B = np.vstack((X, Y))
print(B)
```

rezultat

```
[1 2 3 4]
[[1 2]
 [3 4]]
```

Salvarea matricelor **NumPy** într-un fișier CSV se face utilizând funcția `np.savetxt`, în timp ce încărcarea unei matrice **NumPy** dintr-un fișier CSV se face utilizând funcția `np.loadtxt`. Aceste două funcții nu implică deschiderea prealabilă a fișierului cu ajutorul structurii `with`. Exemplu de utilizare a acestor funcții:

main.py

```
import numpy as np

X = np.array([[1, 2, 3], [4, 5, 6]])

# sciere in fisier
np.savetxt("test.csv", X, delimiter = ",")

# citire din fisier
Y = np.loadtxt("test.csv", delimiter = ",")

print(Y)
```

rezultat

```
[[1. 2. 3.]
 [4. 5. 6.]]
```

G. Cerințe

1. Folosind funcția `np.arange` să se genereze și afișeze următoarele matrice:
 - a) $[1 \ 3 \ 5 \ 7 \ \dots \ 49]$
 - b) $[12 \ 11 \ 10 \ \dots \ -10 \ -11 \ -12]$
 - c) $\left[0 \ \frac{1}{2} \ \frac{2}{3} \ \frac{3}{4} \ \dots \ \frac{59}{60}\right]$

2. Folosind funcția `np.linspace` să se genereze și afișeze următoarele matrice:
 - a) $[4 \ 6 \ 8 \ 10]$
 - b) $[-2 \ -6 \ -10 \ -14 \ -18]$
 - c) $[12 \ 10 \ 8 \ 6 \ 4 \ 2]$

3. Să se rezolve următorul sistem liniar $\begin{cases} 2 \cdot x - 3 \cdot y = 8 \\ 3 \cdot x - 2 \cdot y = -5 \end{cases}$ folosind **NumPy**. Afipați rezultatele acestuia.

4. Să dau următoarele cerințe:
 - a) Creați și afipați o matrice **NumPy** cu valori întregi.
 - b) Utilizați proprietatea `"dtype"` pentru a afipa tipul de date al matricei.
 - c) Creați și afipați o matrice **NumPy** cu valori în virgulă mobilă.
 - d) Utilizați metoda `„astype”` pentru a crea o altă matrice cu aceleași valori, dar de tip întreg.

5. Folosind **NumPy** convertiți următoarea listă de valori numerice, `lista_1 = [15.56, 56.00, 52.36, 56, 58]` într-o matrice **NumPy** unidimensională.

6. Să se genereze o matrice cu dimensiunea 3×3 cu valori cuprinse între 5 și 14.

7. Fie matricele $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$ și $B = \begin{pmatrix} 8 & 10 \\ 8 & 5 \\ 1 & 5 \end{pmatrix}$. Calculați și afipați:
 - a) $A + 5, A - 5, A * 5, \frac{A}{5}, A + B^T, A - B^T, A * B^T, A / B^T$
 - b) $A @ B, (A @ B) @ (A @ B)^{-1}$

8. Fie matricea $C = \begin{pmatrix} 2 & -4 & 1 \\ 6 & 3 & 2 \\ 3 & -2 & 4 \end{pmatrix}$. Să se calculeze și afișeze dimensiunea, transpusa, determinantul, inversa și rangul matricei C .

9. Să se calculeze și să se afișeze sinusul, cosinusul și tangenta elementelor matricei unidimensionale $y = [0^\circ \ 20^\circ \ 40^\circ \ \dots \ 180^\circ]$.

10. Fie matricea $D = \begin{pmatrix} 6 & 5 & 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ -6 & -5 & -4 & -3 & -2 & -1 \\ 4 & 5 & 6 & 7 & 8 & 9 \\ 8 & 7 & 6 & 5 & 4 & 3 \end{pmatrix}$
 - a) Să se afișeze dimensiunea matricei.
 - b) Să se extragă și afișeze submatricea E de dimensiune 3×3 , ce constă din elementele situate pe ultimele 3 linii și primele 3 coloane ale matricei D .

- c) Să se extragă și afișeze submatricea F de dimensiune 3×3 , ce constă din elementele situate pe primele 3 linii și ultimele 3 coloane ale matricei D .
 - d) Să se concateneze matricele E și F sub forma unei matrice de dimensiune 3×6 .
 - e) Să se concateneze matricele E și F sub forma unei matrice de dimensiune 6×3 .
 - f) Să se redimensioneze matricea D din 5×6 într-o matrice 10×3 și să se salveze conținutul într-un fișier CSV.
11. Fie fișierul „`date.csv`”. Încărcați fișierul într-o matrice **NumPy**.
- a) Să se afișeze dimensiunea matricei.
 - b) Calculați și afișați minimul, maximul fiecărei coloane din matrice.
 - c) Calculați și afișați minimul, maximul și suma fiecărei linii din matrice