

T.P. VII - Matrices

Code Capytale : aa20-2467643

I - Ce qu'il faut savoir

Le module `numpy`, importé via la ligne de commande `import numpy as np` permet de manipuler les matrices avec Python.

Définition de matrices.

- * `A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])` permet de définir une matrice ligne par ligne et d'obtenir ainsi la matrice $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ qui sera ici stockée dans la variable `A`.
- * `np.arange(a, b, p)` crée un vecteur ligne contenant les valeurs `a, a + p, a + 2p, ...` tant que `a + kp < b`.
- * `np.zeros((n, p))` crée une matrice à `n` lignes et `p` colonnes ne contenant que des 0.
- * `np.ones((n, p))` crée une matrice à `n` lignes et `p` colonnes ne contenant que des 1.
- * `np.eye(n)` crée la matrice identité de taille `n`.
- * Si `M` est une matrice, alors `M[i, j]` est l'élément situé à la i^{e} ligne et j^{e} colonne, la numérotation commençant à 0.

Opérations sur les matrices.

- * `3 * A` permet de multiplier `A` par le nombre 3.
- * Si `A` et `B` sont des matrices de mêmes tailles, `A + B` permet d'en calculer la somme.
- * Si `A` et `B` sont des matrices de tailles compatibles, `np.dot(A, B)` permet de multiplier les matrices `A` et `B`.
- * Si `x` est une variable contenant la matrice ligne $X = (x_1 \cdots x_n)$, alors `np.cumsum(x)` renvoie le vecteur ligne contenant la somme cumulée des éléments : $(x_1 \quad (x_1 + x_2) \quad (x_1 + x_2 + x_3) \quad \cdots \quad (x_1 + \cdots + x_n))$.

Exercice 1. [D'après BCE ESCP - 2020 - Exercice 1] On considère la matrice $A = \begin{pmatrix} 1 & 2 & 1 \\ 1/2 & 1 & 2 \\ 1 & 1/2 & 1 \end{pmatrix}$. On admet que

$$\forall n \geq 3, A^n = 3A^{n-1} + \frac{9}{4}A^{n-3}.$$

Modifier la suite d'instructions suivante pour qu'elle calcule et affiche A^n pour $n = 10$.

```
import numpy as np

n = ...
I = np.eye(3)
A = np.array(...)
B = np.dot(A, A)

for k in range(3, n+1):
    C = 3 * B + 9/4 * I
    I = ...
    A = ...
    B = ...

print(B)
```

Exercice 2. [D'après Ecricome - 2019 - Exercice 2] On définit sur $]0, +\infty[$ la fonction g par :

$$g(x) = 2x - 1 + \ln\left(\frac{x}{x+1}\right).$$

On admet que g s'annule en unique point α , en changeant de signe, sur l'intervalle $[0,5,1]$.

1. Compléter l'algorithme de dichotomie ci-dessous afin qu'il affiche un encadrement de α à 10^{-2} .

```
import numpy as np

def g(x):
    return ...

a = 0.5
b = 1

while b - a ... :
    m = ...
    if g(a) * g(m) <= 0:
        b = ...
    else:
        ...

print(...)
```

2. On considère la suite (u_n) définie par $u_0 = 0$ et

$$\forall n \geq 1, u_n = (2n - 1) - g(n).$$

La suite d'instructions suivante construit un vecteur u contenant les 50 premiers termes de la suite $(u_n)_{n \geq 1}$.

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

U = np.zeros((1, 51))

for n in range(1, 51):
    U[0,n] = (2 * n - 1) - g(n)

X = np.arange(0, 51)
S = np.cumsum(U)

plt.plot(X, S, 'r+')
plt.show()
```

Interpréter le contenu du vecteur S . Que conjecturer à l'aide du graphique précédent ?

II - Suites récurrentes

Exercice 3. [D'après Ecricome - 2020 - Exercice 1] On considère les suites (u_n) et (v_n) définies par $u_0 = 0, v_0 = 1$ et

$$\forall n \in \mathbb{N}, \begin{cases} u_{n+1} = -2u_n + v_n \\ v_{n+1} = 3u_n \end{cases}$$

On pose $A = \begin{pmatrix} -2 & 1 \\ 3 & 0 \end{pmatrix}$ et $C_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$. On constate que

$$\forall n \in \mathbb{N}, C_{n+1} = AC_n.$$

Compléter le script suivant pour qu'il calcule et affiche les termes u_n et v_n pour $n = 20$.

```
import numpy as np

n = ...
A = np.array(...)
C = np.array([[0.], [1.]])

for k in range(1, n+1):
    C = ...

print(C[0,0], C[1,0])
```

Exercice 4. [Inspiré de Ecricome - 2021 - Exercice 1] Soit $A = \begin{pmatrix} 1 & 1 \\ 2 & 0 \end{pmatrix}$. On définit les suites $(u_n)_{n \in \mathbb{N}}$ et $(v_n)_{n \in \mathbb{N}}$ par

$$u_0 = 1, v_0 = 0 \text{ et } \forall n \in \mathbb{N}, \begin{cases} u_{n+1} = u_n + v_n \\ v_{n+1} = 2u_n \end{cases}.$$

Pour tout n entier naturel, on note $C_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$. On constate que

$$\forall n \in \mathbb{N}, C_{n+1} = AC_n.$$

Compléter le script suivant pour qu'il calcule et affiche les termes u_{12} et v_{12} .

```
import numpy as np

n = ...
A = np.array(...)
C = np.array(...)

for k in range(..., n+1):
    C = ...

print(...)
```

Exercice 5. [D'après BCE BSB - 2016 - Exercice 1] On définit les suites (a_n) , (b_n) et (c_n) par $a_0 = 0$, $b_0 = 1$, $c_0 = 2$ et pour tout n entier naturel :

$$\begin{cases} a_{n+1} = 4a_n - 6b_n + 2c_n \\ b_{n+1} = 2a_n - 4b_n + 2c_n \\ c_{n+1} = -2a_n + 2b_n \end{cases}$$

On pose $A = \begin{pmatrix} 4 & -6 & 2 \\ 2 & -4 & 2 \\ -2 & 2 & 0 \end{pmatrix}$ et $U_n = \begin{pmatrix} a_n \\ b_n \\ c_n \end{pmatrix}$. On constate alors que

$$U_n = A^n U_0.$$

Compléter la suite d'instructions suivante pour qu'elle calcule et affiche la valeur de a_{10} .

```
import numpy as np

A = ...
U = np.arange([[0.], [1.], [2.]])
for i in range(1, 11):
    U = ...

print(...)
```

Exercice 6. [D'après BCE BSB - 2018 - Exercice 1] On définit la suite $(u_n)_{n \in \mathbb{N}}$ par $u_0 = -1$, $u_1 = 1$ et pour tout n entier naturel,

$$u_{n+2} = 3u_{n+1} + 2u_n - 4.$$

1. Compléter la suite d'instructions suivantes pour qu'elle calcule et affiche la valeur de u_{20} .

```
n = ...
v = -1
u = 1

for i in range(2, n+1):
    a = u
    ...
    v = a

print(u)
```

2. On pose $B = \begin{pmatrix} -4 \\ 0 \end{pmatrix}$, $A = \begin{pmatrix} 3 & 2 \\ 1 & 0 \end{pmatrix}$ et $X_n = \begin{pmatrix} u_{n+1} \\ u_n \end{pmatrix}$. On constate que

$$\forall n \in \mathbb{N}, X_{n+1} = AX_n + B.$$

Compléter la suite d'instructions suivante pour qu'elle calcule et affiche la valeur de u_{20} .

```
import numpy as np

n = ...
X = np.array([[1.], [-1.]])
A = np.array([[3., 2.], [1., 0.]])
B = np.array([[4.], [0.]])

for i in range(1, n+1):
    X = ...

print(X[1, 0])
```

Exercice 7. [D'après BCE BSB - 2017 - Exercice 1] On définit les suites (u_n) , (v_n) et (w_n) par $u_0 = 1$, $v_0 = 0$, $w_0 = 2$ et pour tout n entier naturel :

$$\begin{cases} u_{n+1} = u_n \\ v_{n+1} = v_n + 2w_n \\ w_{n+1} = 2u_n + w_n \end{cases}$$

On pose $A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 2 \\ 2 & 0 & 1 \end{pmatrix}$ et $X_n = \begin{pmatrix} 1 \\ v_n \\ w_n \end{pmatrix}$. On admet que

$$\forall n \in \mathbb{N}, X_{n+1} = AX_n.$$

Compléter le programme suivant pour qu'il calcule et mémorise les premiers termes des suites (u_n) , (v_n) et (w_n) puis effectue un tracé de ces points.

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

A = ...
u = np.zeros((11, 1))
v = np.zeros((11, 1))
w = np.zeros((11, 1))

u[0] = 1
v[0] = 0
w[0] = 2
X = np.array([[1.], [0.], [2.]])

for i in range(1, 11):
    X = np.dot(A, X)
    u[i] = 1
    ...
    ...

X = np.arange(0, 11)

plt.figure()
plt.plot(..., ..., 'r.') # Trace u avec des points rouges
plt.plot(..., ..., 'go') # Trace v avec des points verts
plt.plot(..., ..., 'b+') # Trace w avec des + bleus
plt.show()
```