TP3 Python ECG2
Matrices

Il est possible de définir une matrice comme une liste de listes, mais si l'on veut effectuer des calculs matriciels, il est préférable de définir cette matrice comme un tableau `numpy`.

Exemple :

Pour construire les matrice $A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$, $B = \begin{pmatrix} 1 \\ 3 \end{pmatrix}$ et $C = \begin{pmatrix} 1 & 2 \end{pmatrix}$:

```
import numpy as np
A=np.array([[1,2],[3,4]])
B=np.array([[1],[3]])
C=np.array([[1,2]])
```

I)Elément, ligne ou colonne d'une matrice

A[i, j] renvoie le coefficient de la ligne i et de la colonne j de A.

A[i, :] renvoie la ligne i de A.

A[: , j] renvoie la colonne j de A.

> Remarque
> Les lignes et les colonnes sont numérotées à partir de l'indice zéro.

II)Les commandes du module numpy

On importe le module `numpy` d'alias `np`.

np.shape(A) renvoie le format de la matrice A,

np.zeros((n,p)) renvoie la matrice nulle de $\mathscr{M}_{n,p}(\mathbf{R})$,

np.ones((n,p)) renvoie la matrice de $\mathscr{M}_{n,p}(\mathbf{R})$ formée que de 1,

np.eye(n) renvoie la matrice I_n,

np.min(A) renvoie le plus petit élément de la matrice A,

np.max(A) renvoie le plus grand élément de la matrice A,

np.sum(A) renvoie la somme des coefficients de A,

np.transpose(A) renvoie la transposée de A,

np.dot(A,B) renvoie le produit AB

> Remarque
> Pour la somme et la différence, on fait tout simplement $A + B$ et $A - B$.
> En revanche, l'opération $A * B$ renvoie non pas la matrice AB, mais la matrice de coefficient $c_{ij} = a_{ij}b_{ij}$.

III)Les commandes du module numpy.linalg

On importe le module `numpy.linalg` d'alias `al`.

al.inv(A) renvoie la matrice inverse de A,

al.matrix_rank(A) renvoie le rang de A,

al.matrix_power(A,n) renvoie A^n,

al.eig(A) renvoie valeurs propres et vecteurs propres de A,

al.solve(A,B) renvoie la matrice X telle que AX=B où X et B sont des matrices colonnes.

IV)Exercices

On importera les modules `numpy` d'alias `np` et `numpy.linalg` d'alias `al`.

Exercice 1★ ☆ ☆☆

On considère les matrices $A = \begin{pmatrix} 1 & -3 \\ 2 & -7 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 1 \\ -1 & 0 \end{pmatrix}$.

Sur la console, entrer A et B, puis vérifier que ${}^t(AB) = {}^tB\,{}^tA$.

Exercice 2★ ☆ ☆☆

On considère le système (S) : $\begin{cases} x - 2y - 4z = 2 \\ 2x + 3z = 1 \\ -3x + 5y + 9z = 3 \end{cases}$

1)Préciser la matrice A du système (S), puis écrire matriciellement (S) sous la forme $AX = B$.

2)Sur la console, entrer A et B, faire afficher A^{-1}, puis écrire une commande permettant d'afficher l'unique solution de (S).

Exercice 3★ ☆ ☆☆

On considère les suites couplées $(u_n)_{n\in\mathbf{N}}$ et $(v_n)_{n\in\mathbf{N}}$ définies par :

$$\begin{cases} u_{n+1} = \frac{1}{6}u_n + \frac{1}{6}v_n \\ v_{n+1} = u_n + \frac{1}{3}v_n \end{cases}$$

On donne $u_0 = 1$ et $v_0 = 2$. On pose $C_n = \begin{pmatrix} u_n \\ v_n \end{pmatrix}$.

1)Déterminer $A \in \mathscr{M}_2(\mathbf{R})$ telle que $\forall n \in \mathbf{N}, C_{n+1} = AC_n$.

Une récurrence immédiate donne alors : $\forall n \in \mathbf{N}, C_n = A^nC_0$.

2)Sur la console, entrer A et C_0, puis écrire une commande renvoyant la valeur de u_{100} et v_{100}. Conjecturez $\lim\limits_{n\to+\infty} u_n$ et $\lim\limits_{n\to+\infty} v_n$.

Exercice 4★ ☆ ☆☆

Soient $A = \begin{pmatrix} 0 & 1 & 1 \\ 0 & 2 & 0 \\ -2 & 1 & 3 \end{pmatrix}$ et $U = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}$.

Vérifier sur la console que U est un vecteur propre de A.

Exercice 5★ ☆ ☆☆

Soit $A \in \mathscr{M}_n(\mathbf{R})$. On rappelle que λ est VP de $A \iff rg(A - \lambda I_n) < n$.

On prend $A = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$.

A l'aide de la fonction `matrix_rank`, vérifier sur la console que -1 est une valeur propre de A.

Exercice 6★ ★ ☆☆

Dans le module `numpy`, les fonctions `hstack` et `vstack` permettent de concaténer deux matrices A et B suivant une direction horizontale ou verticale.

1)Recopier et exécuter le programme suivant :

```
import numpy as np
A=np.array([[1,2],[3,4]])
B=np.array([[5],[6]])
C=np.array([[7,8]])
print("concaténation horizontale de A et B:")
H=np.hstack((A, B))
print(A,B,H,sep="\n")
print("concaténation verticale de A et C:")
V=np.vstack((A, C))
print(A,C,V,sep="\n")
```

2)Ecrire la matrice D renvoyée par le programme suivant :

```
import numpy as np
A=np.eye(4)
L=np.zeros((1,4))
C=np.ones((5,1))
B=np.vstack((L,A))
D=np.hstack((C,B))
```

3)Soit $M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$.

Ecrire un programme affichant M à l'aide de concaténations de matrices de type ones et zeros.

Exercice 7★ ★ ☆☆
On considère le programme suivant :

```
import numpy as np
A=np.zeros((5,5))
for i in range(5):
    for j in range(5):
        if j==i+1:
            A[i,j]=1
B=A+np.transpose(A)
print(B)
```

Sans exécuter le programme, deviner quelle est la matrice B affichée par le programme. B est-elle diagonalisable ?

Exercice 8★ ★ ☆☆
Ecrivez une fonction de trois arguments (a et b réels, n entier) qui renvoie la matrice de $\mathscr{M}_n(\mathbf{R})$ dont tous les coefficients valent b sauf ceux de la diagonale qui valent a.

Exercice 9★ ★ ☆☆
Ecrire une fonction prenant en argument un entier $n \in \mathbf{N}^*$ et renvoyant la matrice de $\mathscr{M}_n(\mathbf{R})$ telle que pour tout $i \in [\![1, n]\!]$, les coefficients de la i-ème ligne valent i.

Exercice 10★ ★ ☆☆
On dit qu'une matrice carrée est *stochastique* si :
– ses coefficients sont positifs ou nuls,
– sur chaque ligne, la somme de ses coefficients fait 1.

1)Trouver un exemple de matrice stochastique.

2)Ecrire une fonction dont l'argument est une matrice carrée A et qui retourne `True` si A est stochastique et `False` sinon.

Exercice 11★ ★ ☆☆
1)Ecrire une fonction f d'argument A matrice carrée et qui renvoie le nombre d'éléments nuls de A.

2)Ecrire une fonction g d'argument A matrice carrée et qui renvoie $True$ si A est une matrice diagonale et $False$ sinon.

Exercice 12★ ★ ★☆
1)Soit la fonction mystere d'arguments $T \in \mathscr{M}_n(\mathbf{R})$ et $p \in \mathbf{N}$.

```
import numpy as np
def mystere(T,p):
    temp=np.eye(len(T))
    for k in range(p):
        temp=np.dot(temp,T)
    return temp
```

Que renvoie-t-elle ? Par quelle commande simple pouvait-on obtenir le même résultat ?

2)A l'aide de la fonction mystere, vérifier sur un exemple de matrice le théorème suivant :

« si tous les éléments diagonaux d'une matrice triangulaire T sont dans l'intervalle $]-1,1[$, alors T^p tend vers 0 quand $p \to +\infty$ ».

3)On souhaite démontrer ce théorème dans le cas particulier où $T = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}$ avec a,b,c réels tels que $0<c\leq a< 1$ et $b\geq 0$.

a)Montrer par récurrence que pour tout entier $p\geq 0$, il existe un réel b_p tel que $T^p = \begin{pmatrix} a^p & b_p \\ 0 & c^p \end{pmatrix}$ où $0 \leq b_p \leq pa^{p-1}b$.

b)Conclure.

4)Démontrer le théorème dans le cas où T est une matrice triangulaire diagonalisable de $\mathscr{M}_2(\mathbf{R})$.