Feuille_info_17 : Matrices et `numpy`.

Toutes les réponses doivent être regroupées dans un unique fichier Python exécutable.
Chaque fonction doit être testée par au moins un appel.

Rappel de quelques commandes élémentaires de `numpy` :

- Importation du module numpy : `import numpy as np`
- Initialisation d'un tableau : `A = np.array( [[1, 3, 5], [2, 5, 6], [1, 0, 2]] )`
- `M[i, j]` : le coefficient d'indice (i, j) de la matrice `M`
- `M[i, :]` : la ligne numéro i de la matrice `M`
- `M[:, j]` : la colonne numéro j de la matrice `M`
- `M[a:b, c:d]` : matrice extraite de M constituée de lignes `a` à `b-1` et des colonnes `c` à `d-1`.
- `np.zeros((n, p))` renvoie une matrice à n lignes et p colonnes remplie de zéros.
- `np.ones((n, p))` renvoie une matrice à n lignes et p colonnes remplie de uns.
- `np.eye(n)` renvoie la matrice identité de taille n.
- `np.shape(M)` renvoie le couple (n, p) où n est le nombre de lignes et p le nombre de colonnes.
- `np.copy(T)` renvoie une copie en profondeur du tableau `T`.
- `np.linalg.eig(M)` renvoie les valeurs propres et la matrice ce passage de vecteurs propres associés

Les tableaux `numpy` sont des objets mutables, leurs éléments doivent tous être du même type (entiers, flottants, listes, ...) et on peut modifier les éléments, les lignes, les colonnes, ... mais on ne peut pas ajouter des éléments.

Dans ce TD, on se concentre sur les représentations avec le module `numpy`.
(sans oublier que certains préfèrent les matrices représentées par des listes de listes).

1) Ecrire une fonction `test_carre(A)` qui prend en entrée une matrice A et qui renvoie `True` si la matrice est carre et `False` sinon.

2) Ecrire une fonction `test_symétrique(A)` qui prend en entrée une matrice A et qui renvoie `True` si la matrice est symétrique et `False` sinon.

3) On dit qu'une matrice carrée réelle $M = (m_{i,j})$ est *diagonale à dominante stricte par lignes* si :

$$\text{Pour tout indice } i, \quad |m_{i,i}| > \sum_{j \neq i} |m_{i,j}|$$

Écrire une fonction Python `diago_dominante_stricte(M)` qui renvoie `True` si une matrice M vérifie la condition de dominance stricte par lignes et `False` sinon.

On ne vérifiera pas que la matrice est carrée.

4) Déterminer avec un programme les éléments propres des matrices suivantes, puis vérifier par le calcul.

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \qquad M_2 = \begin{pmatrix} 1 & 2 \\ 5 & 4 \end{pmatrix} \qquad M_3 = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 2 \end{pmatrix}$$

5) Ecrire une fonction `puissance(A, n)` qui prend en entrée une matrice A et un entier n et qui renvoie A^n.
Attention : `A**n` ne fait ce que vous pourriez penser en `numpy`.

6) Ecrire une fonction `valeur_dominante(A)` qui prend en entrée une matrice A et qui renvoie sa valeur propre de module dominant.

7) Ecrire une fonction `vecteurs_dominants(A)` qui prend en entrée une matrice A et qui renvoie la liste des vecteurs propres associés à la valeur propre de module dominant.

8) On appelle norme euclidienne d'un vecteur u de $\mathbb{R}^n$ le réel : $||u||_2 = \sqrt{\sum_{k=1}^{n} u_k^2}$

a) Ecrire une fonction `norme_2(u)` qui renvoie la norme euclidienne d'un vecteur `u` modélisé par une liste de flottants.

b) Déterminer les vecteurs propres par `np.linalg.eig` pour la matrice $\begin{pmatrix} 4 & -1 & 0 & 0 \\ -1 & 4 & -1 & 0 \\ 0 & -1 & 4 & -1 \\ 0 & 0 & -1 & 4 \end{pmatrix}$.

c) Vérifier que les vecteurs précedents sont tous de norme 1.

9) On fixe un entier $n \geq 2$ et on considère la matrice $A \in \mathscr{M}_n(\mathbb{R})$ définie par

$$a_{i,j} = \begin{cases} 4 & \text{si } i = j, \\ -1 & \text{si } |i-j| = 1, \\ 0 & \text{sinon.} \end{cases}$$

a) Écrire une fonction `matrice_A(n)` renvoyant la matrice A sous forme d'un tableau `numpy`.

b) Afficher la matrice obtenue pour $n = 5$.

c) Calculer les valeurs propres de A pour $n = 5$.

d) Représenter l'ensemble des valeurs propres de A pour n allant de 2 à 30.

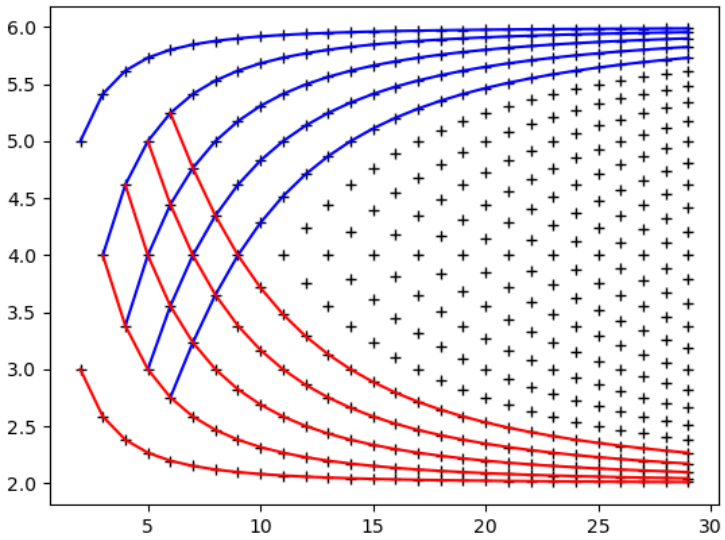

Figure 1 : Le spectre de A en fonction de n entre 2 et 30

Des extraits du formulaire de l'agro :

```
import numpy as np
np.array() ------------- Transforme une liste en matrice numpy
np.linspace(a,b,n) ----- Crée une matrice ligne de n valeurs
                         uniformément réparties entre a et b (inclus)
np.zeros([n,m]) -------- Crée la matrice nulle de taille n x m
np.eye(n) -------------- Crée la matrice identité de taille n
np.diag(L) ------------- Crée la matrice diagonale dont les termes
                         diagonaux sont les éléments de la liste L
np.transpose(M) -------- Renvoie la transposée de M
np.dot(M,P) ------------ Renvoie le produit matriciel MP
np.sum(M) -------------- Renvoie la somme de tous les éléments de M
np.prod(M) ------------- Renvoie le produit de tous les éléments de M
np.max(M) -------------- Renvoie le plus grand élément de M
np.min(M) -------------- Renvoie le plus petit élément de M
np.shape(M) ------------ Renvoie dans un couple le format de la matrice M
np.size(M) ------------- Renvoie le nombre d'éléments de M


import numpy.linalg as la
la.inv(M) -------------- Renvoie l'inverse de la matrice M si elle est inversible
la.eigvals(M) ---------- Renvoie la liste des valeurs propres de M
la.eig(M) -------------- Renvoie un couple L, P où L est la liste des valeurs
                         propres de M et P la matrice de passage associée
la.matrix_rank(M) ------- Renvoie le rang de M
```