TP2 – Python

Pour toute le TP, on pourra entrer les commandes

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

1 Tableaux et matrices

Exercice 1

1. Écrire la matrice
$$A = \begin{bmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{bmatrix}$$
Calculez son déterminant ainsi que son inverse.
2. Entrez les vecteurs
$$b_1 = \begin{bmatrix} 32 \\ 23 \\ 33 \\ 31 \end{bmatrix} \quad b_2 = \begin{bmatrix} 32.1 \\ 22.9 \\ 33.1 \\ 30.9 \end{bmatrix}$$
et résolvez les sytèmes linéaires $Ax = b_1$ et $Ax = b_2$. Qu'y a-t-il d'étonnant ? Calculer le conditionnement de A (utiliser `np.linalg.cond`).

Exercice 2

Tester les commandes `np.prod`, `np.sum`, `np.cumprod`, `np.cumsum`, `np.triu` et `np.tril` sur la matrice
$$A = \begin{bmatrix} 1 & 3 & 5 \\ 3 & 6 & 9 \\ 4 & 3 & 2 \end{bmatrix}$$

Exercice 3

1. Définir le vecteur $V = [0, 1, 2, 3, \ldots, 49, 50]$. Définir le vecteur W contenant les cinq premiers éléments de V, et le vecteur X contenant les cinq premiers et les cinq derniers éléments. Définir ensuite le vecteur $Z = [0, 2, 4, \ldots, 48, 50]$ à partir de V.
2. Définir une matrice M aléatoire à trois lignes et sept colonnes (utiliser `np.random.rand`). Combien de nombres dans cette matrice sont plus grand que $0,5$? Où sont-ils situés ? (utiliser `np.nonzero` ou `np.argwhere`).
3. Construire alors la matrice P obtenue à partir de la matrice M en remplaçant tous les nombres de M inférieurs à $0,5$ par 0, et ceux supérieurs à $0,5$ par 1.

Exercice 4

1. Définir C matrice nulle carrée d'ordre n, avec par exemple $n = 6$.
2. Grâce à une double boucle sur i et j, remplir la matrice C de telle sorte que, pour $1 \leq i \leq n$ et $1 \leq j \leq i$, on ait $C_{i,j} = C_{i-1}^{j-1}$ (coefficient binômial) en utilisant la relation de récurrence
$$C_i^j = C_{i-1}^{j-1} + C_{i-1}^j$$
3. En utilisant `np.mod`, construire la matrice B carrée de taille n telle que
$$B_{i,j} = \begin{cases} 1 \text{ si } C_{i,j} \text{ est impair,} \\ 0 \text{ sinon.} \end{cases}$$

Exercice 5

1. On considère $n = 10$
   (a) Construire la matrice du Laplacien
$$A_n = \begin{bmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & -1 & 2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \ldots & 0 & -1 & 2 \end{bmatrix}$$
   en utilisant `np.diag` et `np.ones`.
   (b) Vérifier (mathématiquement) que les valeurs propres de A sont les $\lambda_k = 4\sin^2\left(\frac{k\pi}{2(n+1)}\right)$, $1 \leq k \leq n$ avec pour vecteur propre associé $p_k = \left[\sin\left(\frac{jk\pi}{(n+1)}\right)\right]_{1 \leq j \leq n}$.
   (c) Calculer les valeurs propres et les vecteurs propres numériques de A (`np.linalg.eig`).
   (d) Calculer le conditionnement de la matrice.
   (e) Ré-ordonner le vecteur des valeurs propres (`np.sort`) de A et déterminer l'erreur numérique maximale commise sur les valeurs propres.
2. Recommencer pour $n = 50$, $n = 100$ et $n = 1000$. Que peut-on constater sur le conditionnement ?

---

2 Approximation numérique

Exercice 1 - Intégration numérique

L'intégrale I d'une fonction f entre a et b peut être approchée en utilisant une approche naïve (mais efficace) de l'intégrale. Les formules correspondant à quelques unes de ces méthodes sont les suivantes :

— Rectangles à gauche : $I \approx h \sum_{0 \leq i \leq n-1} f(x_i)$

— Rectangles à droite : $I \approx h \sum_{1 \leq i \leq n} f(x_i)$

— Trapèzes : $I \approx h\Big(\frac{f(a)+f(b)}{2} + \sum_{1 \leq i \leq n-1} f(x_i)\Big)$

où n est le nombre de points utilisés pour calculer l'intégrale, $h = (b-a)/n$ et $x_i = a + (i-1)h$ pour $i \in [\![0, n]\!]$.

1. Pour chacune des approches ci-dessus, écrire une fonction qui calcule l'intégrale d'une fonction f entre a et b.
2. Tester les fonctions avec, $a = 0$, $b = \pi/2$, $n = 100$, $h = (b-a)/n$ et $f(x) = \sin(x)$. Quelle est la méthode la plus efficace ?
3. Le package permettant de faire des intégrations approchées est `integrate`, dans la bibliothèque `scipy`. Entrer la commande `import scipy.integrate as integ` et tester la fonction `integ.quad`.

Exercice 2 - Interpolation polynomiale

1. Les polynômes peuvent être représentés avec la commande `np.poly1d`.
2. Quelles commandes permettent de récupérer
   — le degré d'un polynome P ?
   — son coefficient X^k ?
   — ses racines ?
   — la valeur de P en un point ?
3. Tracer le polynôme $P(x) = 3x^2 - 5x + 2$ sur l'intervalle $[-1, 2]$.
4. Le package permettant de faire de l'interpolation est `interpolate`, dans la bibliothèque `scipy`.
   (a) Construire le polynôme de Lagrange passant par les points $(0, 2)$, $(1, 4)$ et $(2, -1)$. Pour cela, entrer la commande `import scipy.interpolate as interp`, puis utiliser la fonction `interp.lagrange`.
   (b) Tracer le polynôme obtenu sur $[-1, 3]$ en représentant les points d'interpolation par des croix rouges.

---

3 Figures

Exercice 1 - Courbes en 2D

On considère les fonctions $f(x) = \sin(x)$ et $g(x) = x\cos(x)$ pour $x \in [0, 2]$.

1. Définir un tableau X contenant une discrétisation en $n = 50$ points de $[0, 2]$ (utiliser `np.linspace`), puis définir deux vecteurs F et G contenant les valeurs de f et g en ces points.
2. Tracer F et G en fonction de X sur un même graphe avec des couleurs et des types de traits différents (voir l'aide).
3. Nommer les axes de la figure, donner un titre à la figure et à chaque courbe.

Exercice 2 - Courbes polaires

En utilisant la fonction `plt.pyplot.polar`, tracer la courbe du trifolium $r_1 = 1 + \cos(\beta)$, pour $\beta \in [-\pi, \pi]$.

En ajoutant l'option `projection="polar"` à la fonction `plt.pyplot.subplot`, tracer les courbes suivantes sur la même figure

— Cardioïde $r_1 = 1 + \cos(\beta)$, pour $\beta \in [-\pi, \pi]$.
— Limaçon $r_2 = 3/2 + \cos(\beta)$, pour $\beta \in [-\pi, \pi]$.
— Rosace $r_3 = \sin(6\beta)$, pour $\beta \in [-\pi, \pi]$.
— Spirale d'Archimède $r_4 = 2\beta$, pour $\beta \in [0, 4\pi]$.

Donner un titre à chaque courbe.
