Feuille_info_19 : Autour du DS 5.

Le but de ce TD est de revenir sur des questions du DS 5 avec de nouvelles fonctions du module `numpy` *données en annexe.*

Vous mettrez votre travail sur cahier de prépa.

1) Dans la partie B on montre que quelle que soit la population de départ X_0 alors la répartition dans les trois classes d'âges se stabilise à long terme. (*57% de jeunes, 24% d'adultes jeunes et 19% d'adultes agés*)
   Illustrer ce résultat dans un programme Python en regardant les effectifs de la population après 10 ou 20 générations. (On prendra plusieurs valeurs de X_0)

2) Dans la partie C.
   a) Ecrire une fonction testant si une matrice est stochastique par lignes avec la caractérisation
   $$A \geqslant 0 \ \text{ et } \ A\mathbb{1}_n = \mathbb{1}_n.$$
   b) Ecrire une fonction testant si une matrice est stochastique par colonnes avec la caractérisation
   $$A \geqslant 0 \ \text{ et } \ \mathbb{1}_n^T A = \mathbb{1}_n^T.$$

3) Dans la partie A.
   a) Trouver les racines de $P : x \longmapsto 10x^3 - 4x^2 - 10x + 1$ avec l'algorithme de dichotomie.
   b) Trouver les valeurs propres de L avec `np.linalg.eigvals(L)` pour $L = \begin{pmatrix} 0 & 2 & 1 \\ \frac{1}{2} & 0 & 0 \\ 0 & \frac{3}{5} & \frac{2}{5} \end{pmatrix}$
   c) Comparer les résultats des deux questions précédentes.

4) Dans la partie D on montre que quelle que soit la population de départ X_0 alors l'effectif de chaque classe converge, (X_n) converge vers T_0X^*

   X^* est le vecteur propre de $\mathcal{S}$ associé à la valeur propre 1 de $L = \begin{pmatrix} 1/2 & 1/3 & 1/4 & 1 \\ 1/2 & 0 & 0 & 0 \\ 0 & 2/3 & 0 & 0 \\ 0 & 0 & 3/4 & 0 \end{pmatrix}$.

   a) Déterminer X^* avec `np.linalg.eig`. *Le calcul donne* $X^* = \begin{pmatrix} 48\% \\ 24\% \\ 16\% \\ 12\% \end{pmatrix}$
   b) On prend $X_0 = \begin{pmatrix} 1000 \\ 0 \\ 0 \\ 0 \end{pmatrix}$
      i. Interpréter biologiquement le choix de X_0
      ii. Que vaut T_0 ?
      iii. Vérifier qu'après plusieurs générations (X_n) converge vers T_0X^*.

5) Dans la partie E.

   Ecrire une fonction Python qui vérifie qu'une matrice P vérifie $\exists m \in \mathbb{N},\ P^m > 0$.
   On admettra le résultat : si un tel m existe alors il vérifie $m \leqslant (n-1)^2 + 1$.

Annexe : **Commandes Python**

On suppose que les modules `math` et `numpy` sont importés via `import math as m` et `import numpy as np` .

Interprétation	Python
Transforme une liste `L` en tableau `numpy`	`np.array(L)`
Matrice nulle de taille $n \times p$	`np.zeros([n,p])`
Matrice remplie de 1 de taille $n \times p$	`np.ones([n,p])`
Matrice identité de taille n	`np.eye(n)`
Coefficient d'indice (i, j) de la matrice A	`A[i, j]`
Ligne d'indice i de la matrice A	`A[i, :]`
Colonne d'indice j de la matrice A	`A[:, j]`
Transposée de la matrice A	`np.transpose(A)`
Somme des éléments de la matrice A	`np.sum(A)`
Taille de la matrice A	`d = np.shape(A)`
- nombre de lignes	`d[0]`
- nombre de colonnes	`d[1]`
Pour A et B matrices de tailles compatibles :	
- addition et soustraction	`A + B` , `A - B`
- multiplication matricielle	`np.dot(A, B)` ou `A @ B`
Création de matrices de booléens (coefficient par coefficient) :	
- Coefficients égaux à une constante c	`A == c`
- Coefficients strictement supérieurs à une constante c	`A > c`
- Coefficients strictement inférieurs à une constante c	`A < c`
- Coefficients inférieurs ou égaux à une constante c	`A <= c`
- Coefficients supérieurs ou égaux à une constante c	`A >= c`
Opérations sur des matrices de booléens (*toujours avec des parenthèses*) :	
- Conjonction logique	`(0 < A) & ( A < 1)`
- Disjonction logique	`(A == 0) | (A == 2)`
- Négation logique	`~(A > 0)`
Pour A une matrice de booléens :	
- quantificateur universel	`np.all(A)`
- quantificateur existentiel	`np.any(A)`
Valeur absolue d'un réel x	`abs(x)`
Valeur absolue d'un complexe z	`abs(z)`

```
np.linalg.eigvals(M) --- Renvoie la liste des valeurs propres de M
np.linalg.eig(M) ------- Renvoie un couple L, P où L est la liste des valeurs
                                 propres de M et P la matrice de passage associée
```