Université de Carthage Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul Département mathématique	IPEIN المعهد التحضيري للدراسات الهندسية بنابل Institut Préparatoire aux Etudes d'Ingénieurs de Nabeul	Année universitaire : 2024/2025 Filière : informatique Niveau d'étude : SM/SP/ST

Td les tableaux à une et à deux dimensions

Exercice n°1 : **Problème décomposition de Choleskey**

L'objectif de ce problème est de résoudre un système d'équations linéaire déterminé de la forme :

$$Ax = b \quad (1)$$

Ou $A \in M_n(C)$ est une matrice Hermetienne définie positive et $b \in C^n$ un vecteur à composantes complexes pour $n \in N^*$

- Une matrice $A \in M_n(C)$ est dite Hermitienne si et seulement si elle vérifie la relation décrite par l'équation suivante :

$$A = A^T \Leftrightarrow a_{ij} \ \forall\, i,j \ \in \{1 \ldots n\} \quad (2)$$

Ou A^T est la matrice transconjuguée de A (matrice transposée de la matrice conjuguée).

Par exemple la matrice $A = A^T = \begin{pmatrix} 1 & 1+2i & 3i \\ 1-2i & 2 & 1+i \\ -3i & 1-i & 5 \end{pmatrix}$ est Hermitienne

- Une matrice est définie positive si et seulement si toutes ses valeurs propres sont positives.
- La décomposition de Choleskey permet de décomposer la matrice $A \in M_n(C)$ hermitienne définie positive en un produit faisant intervenir une autre matrice $L \in M_n(C)$ triangulaire inferieur tel que :

$$A = L\,\bar{L}^T \quad (3)$$

Ou $\bar{L}^T$ est la matrice transconjuguée de L. Les coefficients l_{ij} de la matrice L sont décrits par les équations suivantes :

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}\,\bar{l}_{jk}}{l_{jj}} \quad \forall i,j \ tel\ que\ n \geq i > j \geq 1 \quad (4)$$

$$l_{ij} = 0 \quad \forall i,j \ tel\ que\ n \geq j > i \geq 1 \quad (5)$$

$$l_{ii} = \sqrt{a_{ii} - \sum_{k=1}^{i-1} l_{ik}\,\bar{l}_{ik}} \quad \forall i \in \{1 \ldots n\} \quad (6)$$

- Un système d'équations linéaires est dit triangulaire supérieur s'il a la forme suivante :

$$\begin{cases} U_{11}x_1 + U_{12}x_2 + \cdots + U_{1n}x_n = b_1 \\ U_{22}x_2 + \cdots + U_{2n}x_n = b_2 \\ \ddots \\ U_{nn}x_n = b_n \end{cases} \quad (7)$$

Ce système peut s'écrire sous la forme matricielle suivante :

$$U\,x = b\ ou\ U = \begin{pmatrix} U_{11} & \dots & \dots & \dots & U_{1n} \\ 0 & \ddots & & & \vdots \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & U_{nn} \end{pmatrix} \in M_n(C) \quad (8)$$

est une matrice triangulaire supérieur et $b \in C^n$. La résolution d'un système d »équation de forme triangulaire supérieur peut s'effectuer itérativement du bas vers le haut, en remontant et en remplaçant les x_i pour i allant de n vers 1 par les solutions trouvées dans l'étape précédendante.

$$x_i = \frac{b_i - \sum_{k=i+1}^{n} U_{ik} x_k}{U_{ii}} \qquad \forall i\ allant\ de\ n \to 1 \qquad (9)$$

- Un système triangulaire inferieur peut s'écrire sous la forme matricielle suivante :

$$L\,x = b\ ou\ L = \begin{pmatrix} l_{11} & 0 & \dots & \dots & 0 \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & \ddots & & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ l_{n1} & \dots & \dots & \dots & l_{nn} \end{pmatrix} \in M_n(C) \quad (10)$$

est une matrice triangulaire supérieur et $b \in C^n$. La résolution d'un système d'équation de forme triangulaire inferieur s'effectue en appliquant la formule suivante :

$$x_i = \frac{b_i - \sum_{k=1}^{i-1} l_{ik} x_k}{l_{ii}} \qquad \forall i\ allant\ de\ 1 \to n \qquad (11)$$

- Etant donnée une matrice $A \in M_n(C)$ supposée hermitienne et définie positive, soit $L \in M_n(C)$ la matrice triangulaire supérieur résultante de sa décomposition de Choleskey $A = L\,\overline{L}^T$. Il est possible de transformer la résolution du système (1) en une résolution de deux systèmes linéaires (un système inférieur puis un système supérieur)

$$A\,x = b \Leftrightarrow \begin{cases} L\,y = b \\ \overline{L}^T\,x = y \end{cases} \qquad (12)$$

Travail demandée

Ecrire en python les fonctions suivantes :

1) La fonction booléenne nommée **Hermitienne** qui prend en entrée une matrice **A** et teste si A est une matrice hermitienne ou non.

2) La fonction booléenne nommée **Positive_definite** qui teste une matrice **A**, passée en paramètre, si elle est définie positive.

3) La fonction **Choleskey** qui prend en entrée une matrice **A** supposée hermitienne et définie positive et retourne la matrice L qui vérifie la relation décrite par l »équation (3).

4) La fonction **Solve_upper** qui prend en paramètre une matrice **U** de la forme triangulaire supérieure et un vecteur **b**, la fonction permet de résoudre le système (8) et retourner le vecteur solution ***x*** en appliquant la formule décrite par l'équation (9).

5) La fonction **Solve_lower** qui prend en paramètre une matrice **L** de la forme triangulaire supérieure et un vecteur **b**, la fonction permet de résoudre le système (10) et retourner le vecteur solution ***x*** en appliquant la formule décrite par l'équation (11).

6) La fonction Solve_Choleskey qui prend en entrée :
   - A une matrice supposée hermitienne définie positive.
   - b un vecteur

   la fonction doit renvoyer x la solution du système $Ax = b$ en appliquant la démarche décrite par l'équation (12)

Exercice n°2 : problème de technique de TOPSIS

La technique de TOPSIS est une technique d'aide à la décision multi-critères. L'idée fondamentale de cette technique consiste à choisir parmi un ensemble de'alternatives. La meilleur alternative selon des critères qu'on cherche à maximiser (J) et d'autres critères qu'on cherche à minimiser (J') . L'alternative idéale est la solution qui se rapproche le plus possible de la solution idéale (A^+) et qui s'éloigne le plus possible de la pire solution(A^-).

Cette technique peut-être formalisée en utilisant une matrice décrivant m alternatives en lignes et n critères en colonnes. La matrice de décision associée est notée X. On note x_{ij} un élément à la ligne i et à la colonne j dans X $(1 \leq i \leq m \; et \; 1 \leq j \leq n)$.

Les étapes de la technique TOPSIS sont décrites comme suit :

- **étape 1** : Transformer la matrice de décision X en une matrice de décision normalisée X_n ou (n_{ij}) est un élément de la matrice X_n :

$$n_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}} \qquad 1 \leq i \leq m \;\; et \;\; 1 \leq j \leq n$$

- **étape 2** : Transformer la matrice de décision normalisée X_n en une matrice de décision normalisée et pondérée X_p , ou (p_{ij}) est un élément de la matriceX_p :

$$p_{ij} = w_j n_{ij} \quad 1 \leq i \leq m \;\; et \;\; 1 \leq j \leq n$$

  et ou w_j est le poids du j-ième critère et $\sum_{j=1}^{n} w_j = 1$.

- **étape 3** : La solution idéale (A^+) et la pire solution (A^-) sont enquite définies comme suit :

$$A^+ = \{p_1^+, \dots, p_n^+\}$$

$$A^+ = \{(maxp_{ij}, j \in J), (minp_{ij;}\, j \in J')\}, \qquad 1 \le i \le m$$

$$A^- = \{p_1^-, \dots, p_n^-\}$$
$$A^- = \{(minp_{ij}, j \in J), (maxp_{ij;}\, j \in J')\}, \qquad 1 \le i \le m$$

- A^+ : minimise les critères qu'on cherche à réduire (J') et maximise les critères qu'on cherche à maximiser (J).
- A^- : minimise les critères qu'on cherche à réduire (J) et maximise les critères qu'on cherche à maximiser (J').

- **étape 4** : Calculer pour chaque alternative ***i*** , la distance euclidienne par rapport à la solution idéale (notée d_i^+) ainsi que la distance euclidienne par rapport à la pire solution (notée d_i^-) :

$$\mathrm{d_i^+} = \sqrt{\sum_{\mathrm{j=1}}^{\mathrm{n}} (\mathrm{p_{ij}} - \mathrm{p_j^+})^2} \ , \ 1 \le \mathrm{i} \le \mathrm{m}.$$

$$\mathrm{d_i^-} = \sqrt{\sum_{\mathrm{j=1}}^{\mathrm{n}} (\mathrm{p_{ij}} - \mathrm{p_j^-})^2} \ , \ 1 \le \mathrm{i} \le \mathrm{m}.$$

- **étape 5** : Calculer coefficient de mesure de rapprochement de chaque alternative ***i*** par rapport à la solution idéale (notéeC_i) :

$$C_i = \frac{d_i^-}{d_i^+ + d_i^-} \ , \ 1 \le i \le m$$

- **étape 6** : Ranger les alternatives en fonction des valeurs décroissantes de C_i.
  La valeur C_i est entre 0 et 1 : plus cette valeur est proche de 1, plus l'alternative *i* est meilleure.

Travail demandée

Ecrire en python les fonctions suivantes :

1) La fonction **Saisir** qui permet de saisir et retourner la matrice **M** de réels de taille m * n.

2) La fonction **Normaliser** qui permet de normaliser une matrice **M** telle **qu'indiquer** à l'étape 1.
   NB : on cherche à normaliser la matrice M et non pas à créer une nouvelle matrice normalisée.

3) Etant donnée un vecteur $W = \{w_1, w_2, w_3, \dots, w_n\}$ dont les éléments sont des valeurs réelles de somme égale à 1 et ou chaque w_j représente le j-ème critère.

Ecrire une fonction **Pondérer** qui permet de pondérer une matrice **M** telle qu'indiquée à l'étape 2. (la fonction prend aussi en argument le vecteur W)
NB : on ne vérifie pas la somme des poids des critères.

4) Une fonction **Solutions** qui permet de calculer et retourner, à partir de la matrice **M** pondérée , deux tableaux python à une dimension **A1** et **A2** représentants respectivement la solution idéale (A^+) et la pire solution (A^-) , tel que indiquée à l'étape 3.
NB : pour simplifier le problème, on suppose que :
   - $J' = \emptyset$
   - La solution (A^+) consiste en l'ensemble des valeurs maximales des critères de l'ensemble J.
   - La solution (A^-) consiste en l'ensemble des valeurs minimales des critères de l'ensemble J.

5) Une fonction **Trace_sol** qui permet d'afficher la figure nommée « Solutions » représentant les deux courbes :
   - Courbe 1 : nommée « val max », colorée en rouge, représentant la variation des solutions maximales (A^+) de différents alternatives.
   - Courbe 2 : nommée « val min », colorée en vert, avec un linewidth égale à 4 représentant la variation des solutions minimales (A^-) de différentes alternatives.

   La fonction prend en entrée les deux vecteurs A1 etA2.

6) La fonction **Distances** qui, à partir de la matrice **M** pondérée, un tableau **A1** des solutions (A^+) et un tableau **A2** des solutions(A^-), permet de calculer et retourner pour chaque alternative ***i*** , la distance (d_i^+) ainsi que la distance (d_i^-) telle qu'indiquée à l'étape 4. Le résultat doit être sous la forme d'un dictionnaire python nommée **Dis** tel que pour chaque élément de Dis, la clé est l'indice de l'alternative et la valeur est un tuple de la forme (d_i^+, d_i^-).

7) Une fonction **Coefficient** qui permet de calculer les coefficients de mesure de rapprochement des différentes alternatives par rapport à la solution idéale, telle que indiquée à l'étape 5. La fonction prend en paramètre un dictionnaire **Dis** des distances et retourne comme résultat un dictionnaire **Dc** qui a comme clé l'indice de l'alternative ***i*** et comme valeur le coefficient de rapprochement C_i.

8) Une fonction **Classer** qui permet à partir du dictionnaire des coefficients de rapprochement **Dc** des différentes alternatives par rapport à la solution idéale, d'afficher les différentes alternatives (de la meilleure vers la pire alternative), telle qu'indiquée à l'étape 6. L'affichage consiste le numéro de l'alternative ainsi que son coefficient de rapprochement.