TP: matrice et pivot de Gauss
Informatique commune

Comme toujours, toutes les fonctions doivent être testées.

Rappels sur les tableaux:

- `array` (tableau) est un type similaire à `list`, défini dans le module `numpy`. Il faut donc écrire `import numpy as np` pour l'utiliser.
- `np.array(L)` créé un tableau à partir d'une liste `L`. Par exemple `T = np.array([1, 7, 4])` créé un tableau `T` contenant les éléments 1, 7 et 4.
- Un tableau doit contenir des éléments tous de même type.
- Il est impossible de modifier la taille d'un tableau (sa taille est fixée à sa création).
- **On peut effectuer n'importe qu'elle opération arithmétique sur deux tableaux (de même taille), ce qui donne le tableau obtenu en appliquant l'opération case par case.**
  Exemple: `np.array([7, 1]) + np.array([2, 5])` donne le tableau `np.array([9, 6])`.
- On peut accéder/modifier le $i$ème élément d'un tableau `T` en écrivant `T[i]` (comme pour les listes).

Une ligne d'une matrice est représentée par un tableau. On rappelle qu'une matrice est représentée par un tableau de ses lignes (donc un tableau de tableaux). Par exemple, la matrice $\begin{pmatrix} 1 & 2 \\ 4 & 3 \\ 2 & 0 \end{pmatrix}$ est définie par `np.array([[1, 2], [4, 3], [2, 0]])`.

1. Écrire une fonction `transpose` renvoyant la transposée d'une matrice. On fera en sorte que cette fonction marche même pour des matrices qui ne sont pas carrés. On pourra utiliser `np.zeros((n, p))` pour créer une matrice de taille $n \times p$ avec que des zéros.
2. Écrire une fonction `symetrique` déterminant si une matrice est symétrique, c'est à dire égale à sa transposée.
3. Écrire une fonction `inverse` renvoyant l'inverse d'une matrice 2×2, en supposant que cet inverse existe. On rappelle que l'inverse de $\begin{pmatrix} a & b \\ c & d \end{pmatrix}$ est $\frac{1}{ad - bc} \begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$.
   Vérifier que votre fonction marche en calculant le produit d'une matrice inversible `M` par `inverse(M)`. Pour cela vous pouvez utiliser `np.dot(A, B)` qui renvoie le produit matriciel de `A` par `B`.
4. Écrire une fonction `dilatation` telle que `dilatation(M, i, lambda)` multiplie la ligne `i` de la matrice `M` par `lambda`, c'est à dire qui effectue l'opération $L_i \leftarrow \lambda L_i$.
5. Écrire une fonction `transvection` telle que `transvection(M, i, j, lambda)` effectue l'opération $L_i \leftarrow L_i + \lambda L_j$.
6. Écrire une fonction `echange` telle que `echange(M, i, j)` échange les lignes i et j de `M`.
7. Écrire une fonction `pivot` telle que `pivot(M, j)` renvoie un pivot sur la `j`ème colonne, c'est à dire un indice $i \geq j$ tel que l'élément $m_{i,j}$ de `M` soit non nul.
8. En déduire une fonction `descente(M)` réalisant la descente du pivot de Gauss sur une matrice augmentée `M`, de façon à obtenir une matrice échelonnée (avec des 0 en dessous de la diagonale).

Il faut donc, pour chaque colonne j de `M` (sauf la dernière, qui correspond au second membre de la matrice augmentée):

- Soit k le numéro de ligne renvoyé par `pivot(M, j)`. Échanger la $j$ème ligne avec la $k$ème ligne de façon à ce que le coefficient $m_{j,j}$ de `M` soit non nul.
- Pour toute ligne $i > j$, effectuer une transvection de façon à mettre un 0 sur la ligne i, colonne j de `M`.

9. De façon similaire, écrire une fonction `remontee(M)` parcourant les colonnes de `M` de droite à gauche (sauf la dernière) en mettant des 0 au dessus de la diagonale. On supposera dans cette fonction que tous les coefficients diagonaux sont non nuls: il n'y a donc pas besoin d'appeller `pivot`.

10. Écrire une fonction `gauss(M)` prenant en argument une matrice augmentée `M` d'un système $AX = B$ (c'est à dire `M` $= (A|B)$) et appliquant la méthode du pivot de Gauss sur `M`.

    Vérifier que $\begin{pmatrix} 3 \\ -5 \\ 8 \end{pmatrix}$ est l'unique solution du système dont la matrice augmentée est `M1`.

11. Écrire une fonction `inverse` telle que `inverse(A)` renvoie la matrice inverse de `A` si elle existe et affiche un message d'erreur sinon.