TP 2e année IPT : Pivot de Gauss et Applications

Dans tout le TP on utilisera le module `numpy` que l'on importera comme suit : `import numpy as np`. On représente les matrices et vecteurs par des tableaux de type `np.array`. On ne travaillera qu'avec des matrices carrées inversibles.

Exercice 1 : Algorithme du pivot de Gauss

1. Ecrire les fonctions `echange(A,i,j)` et `transvection(A,i,j,t)` qui réalisent respectivement l'échange des lignes i et j et la transvection $L_i \leftarrow L_i + t.L_j$ sur la matrice A. Ces fonctions modifient la matrice A sans créer de nouvelle matrice et ne renvoient rien.

2. Ecrire la fonction `pivot_partiel(A,j)` qui cherche dans la colonne j (et les lignes $\geqslant j$) un pivot de valeur absolue maximale.

3. Ecrire la fonction `triangularisation(A,b)` qui transforme le système linéaire $Ax = b$ en système triangulaire équivalent $Tx = v$. La fonction devra renvoyer le couple (T, v) sans modifier A et b (utiliser `copy()`).

4. Ecrire la fonction `remontee(T,v)` qui renvoie la solution x du système triangulaire $Tx = v$. En déduire la fonction `gauss(A,b)` qui résout le système linéaire $Ax = b$. Comparer avec la fonction `solve` du module `numpy.linalg`.

Exercice 2 : Applications

1. Ecrire une fonction `determinant(A)` qui calcule le déterminant de A après l'avoir triangularisée (attention au signe !). Comparer avec la fonction `det` du module `numpy.linalg`.

2. Ecrire une fonction `inverse(A)` qui calcule l'inverse de A en utilisant l'algorithme du pivot de Gauss (on pourra modifier la fonction de remontée pour qu'elle traite le cas des systèmes $AX = B$ où B est une matrice carrée). Comparer avec la fonction `inv` du module `numpy.linalg`.

Exercice 3 : Décomposition LU

On suppose que A a tous ses mineurs principaux (c'est-à-dire les déterminants des sous-matrices "calées dans le coin gauche") non nuls. Dans ce cas il existe une matrice triangulaire supérieure inversible U et une matrice triangulaire inférieure L avec des 1 sur la diagonale telles que $A = LU$. La matrice U n'est autre que la triangularisée de A avec le pivot de Gauss (sans faire de permutations !) et la matrice L est le produit des inverses des matrices de transvections que l'on applique au cours de l'algorithme (s'en convaincre !). Calculer la décomposition LU a la même complexité que le pivot de Gauss (qui est ?) mais est très utile si l'on souhaite ensuite résoudre plusieurs systèmes linéaires successifs associés à A, en effet chaque système se ramène alors à résoudre deux systèmes triangulaires successifs (quelle en est la complexité ?)

1. Quelle est la matrice associée à la transvection $L_i \leftarrow L_i + t.Lj$? Quelle est son inverse ?

2. Ecrire une fonction `decompositionLU(A)` qui renvoie les matrices L et U définies ci-dessus. Par exemple, pour la matrice $A = \begin{pmatrix} 1 & 1 & -2 \\ 2 & -1 & -1 \\ 1 & 1 & 1 \end{pmatrix}$, la fonction doit retourner $L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}$ et $U = \begin{pmatrix} 1 & 1 & -2 \\ 0 & -3 & 3 \\ 0 & 0 & 3 \end{pmatrix}$.