TP 11

Décomposition LU

Exercice 1 – Décomposition LU

Soit A une matrice carrée telle que toute sous-matrice principale (sous-matrice carrée calée dans le coin supérieur gauche) soit inversible.

On définit les matrices de transvections :

$$\forall \alpha \in \mathbb{R},\ \forall i,j \in \mathbb{N}_n \,/\, i \neq j,\ T(i,j,\alpha) = I_n + \alpha E_{i,j}$$

où $E_{i,j}$ est la matrice où tous les coefficients sont nuls sauf le numéro (i,j) qui vaut α.

On rappelle qu'étant donnée une matrice »»»»

- $T(i,j,\alpha) \times A$ revient à faire l'opération $L_i \leftarrow L_i + \alpha L_j$ sur les lignes de A ;
- $A \times T(i,j,\alpha)A$ revient à faire l'opération $C_j \leftarrow C_j + \alpha C_i$ sur les colonnes de A.
- $T(i,j,\alpha)$ est inversible d'inverse $T(i,j,-\alpha)$.

On peut alors montrer que la matrice A admet une décomposition de la forme $A = LU$ avec L (lower) une matrice triangulaire inférieure et U (upper) une matrice triangulaire supérieure.

La méthode consiste à appliquer la méthode de Gauss classique sur les lignes de A, sans permutation de lignes afin de trianguler A. Ainsi il existe des matrices de transvections $T_1, ..., T_p$ telles que

$$(T_p \times T_{p-1} \times ... \times T_1) \times A = U \text{ triangulaire supérieure.}$$

On suffit alors de poser

$$L = (T_p \times T_{p-1} \times ... \times T_1)^{-1} = T_1^{-1} \times T_2^{-1} \times ... \times T_p^{-1} \text{ triangulaire inférieure}$$

soit $A = LU$ cherchée.

L'intérêt de cette méthode consiste à ramener la résolution d'un système linéaire $AX = B$ à la résolution de deux systèmes triangulaires faciles à résoudre :

$$\begin{cases} UX = Y \\ LY = B \end{cases}$$

L'objet de ce TP consiste à écrire un programme permettant de déterminer cette décomposition LU et la comparer à l'algorithme de Gauss classique. Pour cela, nous allons écrire les matrices sous forme de tableaux bi-dimensionnels de type `array`, en utilisant le package `numpy` dont voici quelques exemples d'utilisations :

```
import numpy as np
A=np.array([[1,2],[3,4]]
B=np.eye(3)  # (matrice identité I_3)
B=np.zeros(n,p)  # (matrice nulle d'ordre (n,p))
B[1,0]=2
C=A.dot(A)+2B.dot(A)  # (pour le calcul de C=A^2+2B x A).
B[:,j]     # (représente la colonne numéro j de B)
```

Nous utiliserons aussi les fonctions définies dans le fichier `random_matrix`, disponible sur le site de la classe ainsi que la fonction `clock` du module `time` pour la mesure des temps d'éxécution.

Question 1 *Écrire une fonction* `trans_ligne` *qui à un quadruplet* `(A,i,j,alpha)`*, où A est la matrice d'ordre n, $i,j \in [\![0,n]\!]$, et $\alpha \in \mathbb{R}$, associe le résultat du produit de la matrice de transvection $T(i,j,\alpha)$ d'ordre n par A.*

Par souci d'efficacité, il est préférable de voir cette opération comme une opération sur les lignes plutôt que comme un produit matriciel. On s'arrangera autant que possible pour éviter les calculs inutiles sur les coefficients que l'on sait être nuls.

Question 2 *Écrire une fonction* `trans_colonne` *qui à un quadruplet* `(A,i,j,alpha)`*, où A est la matrice d'ordre n, $i,j \in [\![0,n]\!]$, et $\alpha \in \mathbb{R}$, associe le résultat du produit de A par l'inverse de la matrice de transvection $T(i,j,\alpha)$ d'ordre n.*

Question 3 *Écrire une fonction* `LU` *retournant deux matrices L et U de la décomposition $A = LU$ pour une matrice A donnée.*

Question 4 *Écrire une fonction* `resolution_sup` *de résolution d'un système triangulaire supérieur.*

Question 5 *Écrire une fonction* `resolution_inf` *de résolution d'un système triangulaire inférieur.*

Soient $A \in M_{100}(\mathbb{R})$ et $b_0, ..., b_{24} \in M_{100,1}(\mathbb{R})$ choisis aléatoirement (les coefficients étant pris dans $[\![0,100]\!]$).

Question 6 *Écrire pour une matrice A donnée et la matrice B constituée des colonnes $b_0, ..., b_{24}$:*

1. *une fonction* `temps_LU(A,B)` *qui calcule le temps de résolution de 25 systèmes associés à A, en calculant une seule fois la décomposition LU de A, puis en résolvant les 2 systèmes triangulaires pour chaque second membre B ;*
2. *une fonction* `temps_pivot(A,B)` *qui calcule le temps de résolution des 25 systèmes associés à A, en reprenant un pivot complet pour chaque système (on utilisera la fonction* `resout` *du programme* `pivot` *sur le site de la classe).*
3. *une fonction* `temps_inverse(A,B)` *qui calcule le temps de résolution de 25 systèmes associés à A, en calculant A^{-1} par la méthode du pivot puis en calculant $A^{-1} \times B$ pour chaque second membre B.*

Question 7 *Plus généralement, écrire une fonction* `trace_temps(A,B,nom_de_fichier)` *qui trace, pour $k \in [\![0,25]\!]$, les 2 courbes correspondant au temps de réponse pour la résolution de k systèmes associés à $A \in M_{100}(\mathbb{R})$ par décomposition LU et par calcul de A^{-1}. Commenter.*