Université de Strasbourg **Techniques d'Analyse Numérique (TAN1)**
L3 Mathématiques (S5) **Année 2019-2020**

TD-TP2. Systèmes linéaires

1 Conditionnement

Rappels
Soit $A = (a_{i,j})_{i,j} \in \mathcal{M}_n(\mathbb{R})$, les normes subordonnées aux normes vectorielles classiques $||\cdot||_1$, $||\cdot||_2$ et $||\cdot||_\infty$ vérifient

$$||A||_1 := \sup_{x\in\mathbb{R}^n, x\neq 0} \frac{||Ax||_1}{||x||_1} = \max_{j\in\{1,\dots,n\}} \sum_{i=1}^{n} |a_{i,j}|,$$
$$||A||_2 := \sup_{x\in\mathbb{R}^n, x\neq 0} \frac{||Ax||_2}{||x||_2} = \sqrt{\rho(A^t.A)},$$
$$||A||_\infty := \sup_{x\in\mathbb{R}^n, x\neq 0} \frac{||Ax||_\infty}{||x||_\infty} = \max_{i\in\{1,\dots,n\}} \sum_{j=1}^{n} |a_{i,j}|,$$

où $\rho(\cdot)$ correspond au rayon spectral.

Exercice 1 : Calcul pratique du conditionnement
Soit $A = \begin{pmatrix} 2 & 1 \\ 0 & 1 \end{pmatrix}$. Calculer le conditionnement pour la norme 2 de A.

Exercice 2 : Conditionnement des normes $||\cdot||_1$ et $||\cdot||_\infty$

1. Soit $B \in \mathcal{M}_n(\mathbb{R})$, la matrice suivante : $B = \begin{pmatrix} 1 & -1 & \dots & -1 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & -1 \\ 0 & \dots & 0 & 1 \end{pmatrix}$.
   (a) Calculer B^{-1}.
   (b) En déduire $\text{cond}_1(B)$ et $\text{cond}_\infty(B)$.
2. On considère $A \in \mathcal{M}_n(\mathbb{R})$.
   (a) Montrer que pour tout $x \in \mathbb{R}^n$, on a $||x||_\infty \leq ||x||_1 \leq n||x||_\infty$.
   (b) En déduire que $\frac{1}{n}||A||_\infty \leq ||A||_1 \leq n||A||_\infty$.
   (c) Conclure que $\frac{1}{n^2}\text{cond}_\infty(A) \leq \text{cond}_1(A) \leq n^2\text{cond}_\infty(A)$.

2 Factorisation LU

Rappels
Soit A une matrice dont toutes les sous-matrices $(A^{(k)})_{i,j}$ pour $(i,j) \in \{1,...,k\}^2$ sont inversibles. La factorisation LU revient à décomposer A en produit d'une matrice triangulaire inférieure L à diagonale unitaire ("*lower triangular matrix*" en anglais) et une matrice triangulaire supérieure U ("*upper triangular matrix*" en anglais).
Nous rappelons que la matrice U est obtenue à l'aide de la méthode du pivot de Gauss (sans permutation sur les lignes) et que la matrice L correspond à la matrice

$$L = \begin{pmatrix} 1 & & & \\ \frac{a_{2,1}}{a_{1,1}} & 1 & & 0 \\ \frac{a_{3,1}}{a_{1,1}} & \frac{a_{3,2}}{a_{2,2}} & \ddots & \\ \vdots & \vdots & * & 1 \\ \frac{a_{n,1}}{a_{1,1}} & \frac{a_{n,2}}{a_{2,2}} & & \frac{a_{n,n-1}}{a_{n-1,n-1}} \quad 1 \end{pmatrix}.$$

Exercice 3 : Factorisation LU : Mise en pratique

1. Calculer la factorisation LU de la matrice A :

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2 & 5 \\ 4 & 6 & 8 \end{pmatrix}.$$

2. Même question pour la matrice $\tilde{A}$:

$$\tilde{A} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2+10^{-6} & 5 \\ 4 & 6 & 8 \end{pmatrix}.$$

Exercice 4 (implémentation Python) : Précision de la méthode

1. Calculer les factorisations LU de la matrice A pour plusieurs valeurs de $p > 0$:

$$A = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 2+10^{-p} & 5 \\ 4 & 6 & 8 \end{pmatrix}.$$

En python, on pourra s'aider de la fonction

```
import scipy as sc
sc.linalg.lu
```

2. A l'aide du conditionnement des matrices L et U, conclure sur la précision des méthodes.

Exercice 5 : Une autre factorisation
Soient

$$A_1 = \begin{pmatrix} 1 & -2 & 0 \\ -2 & 8 & -6 \\ 0 & -6 & 25 \end{pmatrix} \quad \text{et} \quad A_2 = \begin{pmatrix} 4 & 0 & 12 & -6 \\ 0 & 1 & 2 & 1 \\ 12 & 2 & 49 & -4 \\ -6 & 1 & -4 & 51 \end{pmatrix}.$$

Montrer que les matrices A_1 et A_2 peuvent s'écrire $A_i = B_i B_i^T$ avec B_i une matrice triangulaire inférieure avec des éléments diagonaux strictement positifs.
Indication : on pourra utiliser la décomposition LU de A_i ainsi que la matrice $D = diag(\sqrt{u_{1,1}}, \sqrt{u_{2,2}}, \ldots, \sqrt{u_{n,n}})$.
Remarque. Une telle décomposition est appelée la décomposition de Cholesky. La matrice B existe (et est unique) dès que A est symétrique définie positive.

Exercice 6 (implémentation Python) : Décomposition LU sans utiliser `sc.linlag.lu`

1. Construire une fonction `DecompositionLU(A)` qui calcule les matrices L et U de la décomposition LU de la matrice A.
2. Appliquer cette fonction aux matrices

$$A_1 = \begin{pmatrix} 9 & 8 & 6 \\ 7 & 6 & 12 \\ 9 & 3 & 9 \end{pmatrix} \quad \text{et} \quad A_2 = \begin{pmatrix} 11 & 8 & 3 & 13 \\ 2 & 12 & 7 & 10 \\ 3 & 3 & 17 & 13 \\ 11 & 2 & 12 & 7 \end{pmatrix}.$$

   Les réponses attendues pour U par exemple sont

$$U_1 = \begin{pmatrix} 9 & 8 & 6 \\ 0 & -\frac{2}{9} & \frac{22}{3} \\ 0 & 0 & -162 \end{pmatrix} \quad \text{et} \quad U_2 = \begin{pmatrix} 11 & 8 & 3 & 13 \\ 0 & 10.5454545 & 6.45454545 & 7.63636364 \\ 0 & 0 & 15.6810345 & 8.86206897 \\ 0 & 0 & 0 & -8.81693238 \end{pmatrix}.$$

3. Soient A une matrice triangulaire inférieure et x une solution du système $Ax = b$ (si cette solution existe). Ecrire une relation de récurrence pour calculer les x_i. Cette récurrence s'appelle *l'algorithme de descente*.
4. Écrire de même une relation de récurrence entre les x_i si A est une matrice trianguaire supérieure (on retrouve ainsi *l'algorithme de remontée*).
5. Construire une fonction `Descente(A,b)` correspondant à l'algorithme de descente.
6. Construire de même une fonction `Remontee(A,b)` correspondant à l'algorithme de remontée.
7. À l'aide des fonctions `DecompositionLU`, `Remontee` et `Descente`, écrire une fonction qui résoud un système linéaire $Ax = b$. L'utiliser pour les systèmes $A_1 x = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ et $A_2 x = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$.
8. Tester votre code. Vérifier votre programme en comparant la solution avec ce que fournit la fonction `np.linalg.solve(A,b)` de Python.

Exercice 7 : Matrice inversible

Soient a, b, c et d des nombres réels. On considère la matrice $A = \begin{pmatrix} a & a & a & a \\ a & b & b & b \\ a & b & c & c \\ a & b & c & d \end{pmatrix}$.

1. Déterminer la décomposition LU de A (si elle existe).
2. Donner les conditions sur a, b, c et d pour que la matrice A soit inversible.

Exercice 8 (implémentation Python) : Matrice du laplacien

On rappelle que la matrice du laplacien est définie par

$$A = \begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{pmatrix}$$

1. Comment déterminer l'inverse de A à partir de la résolution de $Ax = b$ pour des vecteurs b bien choisis ?

2. En prenant les vecteurs b de la question précédente et à l'aide des fonctions `DecompositionLU`, `Remontee` et `Descente` de l'exercice 6 calculer numériquement l'inverse de la matrice A (sans utiliser la fonction `np.linalg.inv(A)`).
3. Comparer le résultat avec ce que retourne `np.linalg.inv(A)`.