Module: Analyse Numérique **Classes :** $3^{\text{ème}}$ année **AU: 2024 / 2025**

TP1: Résolution numérique de systèmes d'équations linéaires

Objectif du TP:

L'objectif de ce TP est l'implémentation des méthodes itératives de Jacobi et de Gauss-Seidel pour la résolution d'un système d'équations linéaires (SEL).
On aura besoin des modules numpy et matplotlib.pyplot. Il faudra charger ces modules, en tapant les commandes suivantes

```
import numpy as np
import matplotlib.pyplot as plt
```

1 Rappels de Cours

1.1 Méthodes itératives pour la résolution d'un SEL

Les méthodes itératives consistent à avoir une suite récurrente de vecteurs, qui converge vers la solution du SEL. Cette solution est donnée une fois qu'une condition d'arrêt imposée à l'algorithme, est satisfaite.

1.1.1 Principe

On considère le SEL (S): $AX = b$ avec

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix}, X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} \; et \; b = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}.$$

On suppose que (S) est de Cramer.
Les méthodes itératives se basent sur une décomposition de A sous la forme $A = M - N$ où M est une matrice inversible. Une suite récurrente de solutions $X^{(k)}$, $k \geq 0$, est générée comme suit :

$$\begin{cases} X^{(0)} \in \mathbb{R}^n \text{ donné} \\ \\ X^{(k+1)} = M^{-1}NX^{(k)} + M^{-1}b. \end{cases}$$

avec $X^{(0)}$ un vecteur initial.
Cette suite converge, sous certaines conditions, vers la solution exacte X du système (S).

1.1.2 Méthodes de Jacobi et de Gauss-Seidel

Les méthodes itératives de Jacobi et de Gauss-Seidel pour résoudre (S) : $AX = b$, consistent en premier lieu à décomposer A sous la forme :

$$A = D - E - F$$

avec:
D : matrice diagonale dont les coefficients diagonaux sont ceux de la matrice A.
E : matrice triangulaire inférieure dont les coefficients diagonaux sont nuls.
F : matrice triangulaire supérieure dont les coefficients diagonaux sont nuls.
Plus précisément, étant donnée une matrice $A = (a_{i,j}) \in M_n(\mathbb{R})$,

$$A = \underbrace{\begin{pmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{pmatrix}}_{D} - \underbrace{\begin{pmatrix} 0 & \cdots & \cdots & 0 \\ -a_{2,1} & \ddots & & 0 \\ \vdots & \ddots & \ddots & 0 \\ -a_{n,1} & \cdots & -a_{n,n-1} & 0 \end{pmatrix}}_{E} - \underbrace{\begin{pmatrix} 0 & -a_{1,2} & \cdots & -a_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & -a_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix}}_{F}$$

Méthode de Jacobi: $M = D$ et $N = E + F$.
Méthode de Gauss-Seidel: $M = D - E$ et $N = F$.

1.1.3 Convergence

Si la matrice A est à diagonale strictement dominante, alors les méthodes de Jacobi et de Gauss-Seidel convergent quelque soit le choix du vecteur initial $X^{(0)}$.

1.1.4 Critère d'arrêt

Soit $A = (a_{i,j})_{1 \leq i,j \leq n} \in M_n(\mathbb{R})$ avec $a_{ii} \neq 0$ pour tout $i = 1, \cdots, n$ et soit ε une tolérance donnée. Parmi les critères d'arrêt pour les méthodes itératives de Jacobi et Gauss-Seidel, on cite

$$||AX^{(k)} - b|| \leq \varepsilon.$$

2 Applications numériques

Soient $n \geq 4$ et $a_i \in \mathbb{R}$ pour $i = 1, 2, 3$.
On s'intéresse à la résolution numérique du problème (S_n): $AX = b$ avec

$$A = A(a_1, a_2, a_3) = \begin{pmatrix} a_1 & a_2 & 0 & \dots & \dots & 0 \\ a_3 & a_1 & a_2 & \ddots & & \vdots \\ 0 & a_3 & a_1 & a_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & \ddots & a_2 \\ 0 & \dots & \dots & 0 & a_3 & a_1 \end{pmatrix}, \; X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{pmatrix} \quad \text{et} \quad b = \begin{pmatrix} 2 \\ 2 \\ \vdots \\ \vdots \\ 2 \\ 2 \end{pmatrix}.$$

1. (a) Écrire une fonction `tridiag(a1,a2,a3,n)` qui renvoie la matrice tridiagonale $A = A(a_1, a_2, a_3)$ de taille n et le second membre b.

   (b) Tester la fonction `tridiag(a1,a2,a3,n)` pour $a_1 = 4$, $a_2 = a_3 = 1$ et $n = 10$.

2. (a) Écrire une fonction `matrice_diag_dominante(A)` prenant en entrée une matrice A, une matrice carrée d'ordre n, qui vérifie si cette matrice est à diagonale strictement dominante ou non.

   (b) Tester la fonction `matrice_diag_dominante(A)` avec les matrices $A = A(4, 1, 1)$ et $B = A(1, 1, 1)$ pour $n = 10$.

3. (a) Définir une fonction nommée `jacobi(A, b, X0, epsilon)` prenant en entrée une matrice carrée A d'ordre n, un second membre b, une valeur initiale $X^{(0)}$ et une tolérence `epsilon`, et qui renvoie une solution approchée du SEL $AX = b$ par la méthode de Jacobi et le nombre d'itérations effectuées.
   La tolérance `epsilon` est utilisée pour le critère d'arrêt: $||AX^{(k)} - b|| \leq$ `epsilon`.
   On testera, à l'aide de la fonction `matrice_diag_dominante(A)`, si A est à diagonale strictement dominante, dans le cas contraire, on renverra le message

suivant: "A n'est pas à diagonale strictement dominante". Compléter la fonction pour quelle soit correctement implémentée.

```
def ..............................

    etat = ..........................

    if  .................. :
        return ("A n'est pas à diagonale strictement dominante")
    else:
        D= ..................     #matrice diagonale de A

        E= ..................  #matrice tringulaire inférieure

        F= ..................  #matrice tringulaire supérieure

        M= ..................

        N= ..................

        k=0

        ............................ :

            ....................................

            ....................................

        return X0,k
```

(b) En utilisant la fonction `tridiag(a1,a2,a3,n)`, tester la fonction `jacobi`, avec les paramètres et arguments suivants :
$n = 10$, $a1 = 4$, $a2 = a3 = 1$, $X^{(0)} = (1, 1, \ldots, 1)^t \in \mathcal{M}_{10,1}(\mathbb{R})$ et `epsilon` $= 10^{-6}$.

(c) En utilisant la fonction $np.linalg.solve(A, b)$, résoudre (S_{10}) et trouver l'erreur commise par la méthode de Jacobi en norme euclidienne.

4. (a) Écrire une fonction `Gauss(A, b, X0, epsilon)` qui renvoie une solution approchée du SEL $AX = b$ par la méthode de Gauss-Seidel et le nombre d'itérations effectuées pour atteindre la convergence en testant si A est à diagonale strictement dominante.

(b) Tester la fonction de Gauss-Seidel pour le même exemple considéré dans 3.b). Trouver, ensuite, l'erreur commise en norme euclidienne.

5. Comparer les résultats obtenus par les deux méthodes.

6. Représenter, maintenant, sur un même graphe, le nombre d'itérations par les deux méthodes itératives : Jacobi et Gauss-Seidel, en fonction de la précision `epsilon`. On considère `epsilon` $\in \{10^{-3}, 10^{-6}, 10^{-9}, 10^{-12}\}$ et $n = 10$. Interpréter les résultats obtenus.

7. (Question Asychrone) Soit la fonction `Niter(epsilon)` définie comme suit

```
def Niter(epsilon):
    N=np.arange(5,31,5)
    Niter_J=[]
    Niter_GS=[]
    for n in N:
        A=tridiag(4,1,1,n)[0]
        b=tridiag(4,1,1,n)[1]
        X0=np.ones((n,1))
        Niter_J.append(jacobi(A,b,X0,epsilon)[1])
        Niter_GS.append(gauss_seidel(A,b,X0,epsilon)[1])
    return Niter_J, Niter_GS
```

Tester cette fonction pour `epsilon`=10^{-6}. Expliquer les résultats obtenus.