# TP3

Résolution de systèmes linéaires

Sevault Wolber Lucien et Prince Marc

10/04/2025

## Introduction

Dans ce TP, nous allons comparer plusieurs algorithmes de résolutions de systèmes linéaires, la méthode du pivot de Gauss, la méthode de Cholesky et la méthode de Jacobi.

### 1. Résolution de systèmes simples

On cherche à résoudre le système :

$$Ax = b$$

où A est une matrice, x un vecteur inconnu et b un vecteur.

Pour cela nous allons utiliser plusieurs méthodes permettant de résoudre simplement un tel système.

#### 1.

On récupère les deux fichiers .mat sur moodle et on met ces deux fichiers dans notre répertoire de travail.

#### 2.

On exécute la commande `load matlab_little` pour charger la petite matrice. On remarque que deux variables A et b ont été créées.

Tous les tests seront effectués sur la petite matrice pour vérifier le bon fonctionnement des algorithmes.

### 2. Pivot de Gauss

Dans cette partie, nous allons implémenter la méthode du pivot de Gauss pour résoudre le système.

**1.**

On implémente une fonction `gaussStep1` qui transforme le système augmenté $Ax = b$ en matrice triangulaire supérieure avec une diagonale identité.

Pour cela, on effectue sur chaque ligne de la matrice augmentée :

$$L_i \leftarrow \frac{L_i}{A_{i,i}}$$

$$L_j \leftarrow L_j - L_i \cdot A_{i+1,i+1}$$

**2.**

On implémente une fonction `gaussStep2` qui permet de résoudre le système linéaire à partir d'une matrice triangulaire supérieure et d'un vecteur de constantes.

Pour cela, on effectue pour chaque ligne :

$$x_i = b_i - \sum_{j>i} x_j \cdot A_{i,j}$$

**Résultat**

On trouve :

$$x \approx \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}$$

## 3.a. Cholesky

Dans cette partie, nous allons implémenter la méthode de Cholesky.

La factorisation de Cholesky permet d'obtenir la décomposition d'une matrice en une matrice triangulaire inférieure fois sa transposée.

**1.**

On peut appliquer la factorisation de Cholesky uniquement sur des matrices semi-définies positives ce qui est équivalent à ce que ses valeurs propres soient positives ou nulles.

**2.**

On implémente une fonction `supCholesky` qui prend en paramètre une matrice A et qui renvoie la matrice triangulaire supérieure L obtenue avec la méthode de la factorisation de Cholesky.

Pour déterminer la matrice L, on utilise les formules suivantes :

$$L_{ii} = \sqrt{A_{ii} - \sum_{k=1}^{i-1} L_{ik}^2}$$

$$L_{ji} = \frac{A_{ij} - \sum_{k=1}^{i-1} L_{ik} L_{jk}}{L_{ii}}$$

**3.**

On cherche à résoudre :

$$Ax = b$$

On a :

$$A = L^T \cdot L$$

On doit donc résoudre :

$$L^T L x = b$$

On peut faire un changement de variable :

$$Lx = y$$

Pour résoudre le système il suffit donc de résoudre $L^T y = b$ puis $Lx = y$.

**4.**

On implémente une fonction `resCholesky` qui résout le système $Ax = b$ par la méthode de factorisation de Cholesky.

On trouve :

$$x \approx \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}$$

C'est la même réponse que pour la méthode du pivot de Gauss.

**5.**

La fonction `chol` de Matlab effectue la même opération que notre fonction `supCholesky`.

## 3.b. Algorithme itératif : méthode de Jacobi

Dans cette partie, nous allons implémenter la méthode de Jacobi.

La méthode de Jacobi est une méthode itérative permettant de calculer numériquement x dans le cas où A est une matrice à diagonale dominante, c'est à dire quand pour chaque ligne de A, le module de l'élément diagonal est supérieur au module de chaque élément de la ligne.

On appelle, D la matrice composée des éléments diagonaux de A, U la matrice composée des éléments au dessus de la diagonale de A et L la matrice composée des éléments en dessous de la diagonale de A.

On a :

$$\begin{aligned} Ax &= b \\ \Leftrightarrow (L+D+U)x &= b \\ \Leftrightarrow Dx &= -(L+U)x + b \\ \Leftrightarrow x &= D^{-1}(-(L+U)x + b) \end{aligned}$$

On arrive à la relation suivante :

$$x^{(k)} = D^{-1}(-(L+U)x^{(k-1)} + b)$$

**1.**

On implémente une fonction decompJacobi qui prend en paramètre une matrice A et retourne les matrices L, D et U.

**2.**

On implémente une fonction `resJacobi(A, b, tol, itermax)` qui résout le système $Ax = b$ avec la méthode de Jacobi. On arrête d'itérer la fonction quand l'erreur est inférieur à la tolérance où quand le nombre d'itérations maximale est atteint. Pour créer cette fonction, on utilise la relation définie précédemment.

**3.**

On teste notre fonction dans le script `testJacobi.m` en résolvant $Ax = b$. Pour une tolérance de $10^{-3}$, on obtient :

$$x = \begin{bmatrix} 1.0001 \\ 1.9998 \\ -0.9998 \\ 0.9998 \end{bmatrix}$$

On semble s'approcher de la bonne réponse.

Pour une tolérance de $10^{-5}$, on obtient :

$$x = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}$$

C'est la même réponse que pour la méthode du pivot de Gauss.

## 4. Comparaison de méthodes

**1.**

On réalise un script `testMethodes.m` dans lequel on compare la performance de chaque méthode implémentée.

**2.**

On utilise les fonctions `tic` et `toc` pour calculer la vitesse d'exécution de chaque méthode.

On obtient comme temps d'exécution pour la méthode de Gauss, la méthode de Cholesky et la méthode de Jacobi respectivement : $5,1188 \cdot 10^{-4}$, $6,3300 \cdot 10^{-4}$ et $5,1594 \cdot 10^{-4}$.

La méthode la plus rapide est donc la méthode de Gauss.

**3.**

Même pour des sytèmes aussi simples, il semble compliqué de trouver la vrai valeur car des estimations numériques sont faites pour beaucoup de calculs impliquant une erreur. On trouve pour chaque méthode :

$$x \approx \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}$$

En supposant que :

$$x = \begin{bmatrix} 1 \\ 2 \\ -1 \\ 1 \end{bmatrix}$$

On trouve bien Ax = b.

**4.**

On calcule la norme infinie de l'erreur pour les 3 méthodes, on obtient respectivement pour la méthode de Gauss, la méthode de Cholesky et la méthode de Jacobi, $1,1102 \cdot 10^{-16}$, $8,8818 \cdot 10^{-16}$ et $8,8818 \cdot 10^{-16}$.

La méthode la plus précise est la méthode de Gauss.

**5.**

On recommence les expériences avec la matrice de taille moyenne. Pour effectuer nos tests, on a à notre disposition le résultat exact stocké dans la variable `ans`.

On obtient pour respectivement pour la méthode de Gauss, la méthode de Cholesky et la méthode de Jacobi un temps d'exécution de $2,5454$, $59,8041$ et $1,01$.

La méthode la plus rapide semble de loin être la méthode de Jacobi.

On obtient pour respectivement pour la méthode de Gauss, la méthode de Cholesky et la méthode de Jacobi une erreur de $4.4298 \cdot 10^{-14}$, $3.3168 \cdot 10^{-14}$ et $4.4451 \cdot 10^{-14}$.

La méthode la plus précise semble être la méthode de Cholesky.

# Conclusion

Pour conclure, nous avons vu dans ce TP plusieurs méthodes pour résoudre des systèmes linéaires. On remarque que dû à des approximations numériques on ne peut pas obtenir le résultat exact. En observant les résultats de nos test, on remarque que la méthode de Cholesky devient absolument inefficace pour des matrices très grandes et que la méthode du pivot de Gauss et surtout la méthode de Jacobi fonctionnent particulièrement bien.