

Analyse Numérique I

Sup'Galilée, Ingénieurs MACS 1ère année & L3 MIM

François Cuvelier

Laboratoire d'Analyse Géométrie et Applications
Institut Galilée
Université Paris XIII.

2017/09/21

Chapitre IV

Résolution de systèmes linéaires

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

Soient $\mathbb{A} \in \mathcal{M}_n(\mathbb{K})$ une matrice inversible et $\mathbf{b} \in \mathbb{K}^n$.

Résoudre

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

Le calcul de la matrice inverse \mathbb{A}^{-1} revient à résoudre n systèmes linéaires.



Pour résoudre un système linéaire, on ne calcule pas la matrice inverse associée.

- **Méthodes directes** : On cherche \mathbb{M} inversible tel que $\mathbb{M}\mathbb{A}$ *facilement* inversible

$$\mathbb{A}\mathbf{x} = \mathbf{b} \iff \mathbb{M}\mathbb{A}\mathbf{x} = \mathbb{M}\mathbf{b}.$$

- **Méthodes itératives** : On cherche \mathbb{B} et \mathbf{c} ,

$$\mathbf{x}^{[k+1]} = \mathbb{B}\mathbf{x}^{[k]} + \mathbf{c}, \quad k \geq 0, \quad \mathbf{x}^{[0]} \text{ donné}$$

en espérant $\lim_{k \rightarrow +\infty} \mathbf{x}^{[k]} = \mathbf{x}$.

Plan

1 Méthodes directes

• Matrices particulières

- Matrices diagonales
- Matrices triangulaires inférieures
- Matrices triangulaires supérieures

• Exercices et résultats préliminaires

• Méthode de Gauss-Jordan

- Ecriture algébrique

• Factorisation LU

- Résultats théoriques
- Utilisation pratique

• Factorisation LDL*

• Factorisation de Cholesky

- Résultats théoriques
- Résolution d'un système linéaire
- Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

Système diagonal

Soit $A \in \mathcal{M}_n(\mathbb{K})$ diagonale inversible et $\mathbf{b} \in \mathbb{K}^n$.

$$x_i = b_i/A_{i,i}, \quad \forall i \in \llbracket 1, n \rrbracket. \quad (1)$$

Algorithme 1 Fonction **RSLMATDIAG** permettant de résoudre le système linéaire à matrice diagonale inversible

$$A\mathbf{x} = \mathbf{b}.$$

Données : A : matrice diagonale de $\mathcal{M}_n(\mathbb{R})$ inversible.
 \mathbf{b} : vecteur de \mathbb{R}^n .

Résultat : \mathbf{x} : vecteur de \mathbb{R}^n .

- 1: **Fonction** $\mathbf{x} \leftarrow$ **RSLMATDIAG** (A, \mathbf{b})
- 2: **Pour** $i \leftarrow 1$ à n **faire**
- 3: $x(i) \leftarrow b(i)/A(i, i)$
- 4: **Fin Pour**
- 5: **Fin Fonction**

Système triangulaire inférieur

Soit $\mathbb{A} \in \mathcal{M}_n(\mathbb{K})$ triangulaire inférieure inversible ($A_{i,j} = 0$ si $i < j$)

$$\mathbb{A}\mathbf{x} = \mathbf{b} \iff \begin{pmatrix} A_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_{n,1} & \dots & \dots & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\mathbb{A} \text{ inversible} \iff$$

Système triangulaire inférieur

Soit $A \in \mathcal{M}_n(\mathbb{K})$ triangulaire inférieure inversible ($A_{i,j} = 0$ si $i < j$)

$$Ax = b \iff \begin{pmatrix} A_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_{n,1} & \dots & \dots & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$A \text{ inversible} \iff A_{i,i} \neq 0, \forall i \in \llbracket 1, n \rrbracket$$

Système triangulaire inférieur

Soit $\mathbb{A} \in \mathcal{M}_n(\mathbb{K})$ triangulaire inférieure inversible ($A_{i,j} = 0$ si $i < j$)

$$\mathbb{A}\mathbf{x} = \mathbf{b} \iff \begin{pmatrix} A_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ A_{n,1} & \dots & \dots & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$

$$\mathbb{A} \text{ inversible} \iff A_{i,i} \neq 0, \forall i \in \llbracket 1, n \rrbracket$$

$$\text{Soit } i \in \llbracket 1, n \rrbracket, \quad (\mathbb{A}\mathbf{x})_i = b_i, \iff \sum_{j=1}^n A_{i,j}x_j = b_i.$$

$$b_i = \sum_{j=1}^{i-1} A_{i,j}x_j + A_{i,i}x_i + \sum_{j=i+1}^n \underbrace{A_{i,j}}_{=0} x_j = \sum_{j=1}^{i-1} A_{i,j}x_j + A_{i,i}x_i$$

$$x_i = \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j}x_j \right), \quad \forall i \in \llbracket 1, n \rrbracket. \quad (2)$$

$$x_i = \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j} x_j \right), \forall i \in \llbracket 1, n \rrbracket.$$

Algorithme 2 \mathcal{R}_0

- 1: Résoudre $Ax = b$ en calculant successivement x_1, x_2, \dots, x_n .

Algorithme 2 \mathcal{R}_1

- 1: Pour $i \leftarrow 1$ à n faire
 2: $x_i \leftarrow \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j} x_j \right)$
 3: Fin Pour

$$x_i = \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j} x_j \right), \quad \forall i \in \llbracket 1, n \rrbracket.$$

Algorithme 2 \mathcal{R}_1

1: Pour $i \leftarrow 1$ à n faire

2: $x_i \leftarrow \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j} x_j \right)$

3: Fin Pour

Algorithme 2 \mathcal{R}_2

1: Pour $i \leftarrow 1$ à n faire

2: $S \leftarrow \sum_{j=1}^{i-1} A_{i,j} x_j$

3: $x_i \leftarrow (b_i - S)/A_{i,i}$

4: Fin Pour

$$x_i = \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j} x_j \right), \forall i \in \llbracket 1, n \rrbracket.$$

Algorithme 2 \mathcal{R}_3

1: Pour $i \leftarrow 1$ à n faire

2: $S \leftarrow \sum_{j=1}^{i-1} A_{i,j} x_j$

3: $x_i \leftarrow (b_i - S)/A_{i,i}$

4: Fin Pour

Algorithme 2 \mathcal{R}_4

1: Pour $i \leftarrow 1$ à n faire

2: $S \leftarrow 0$

3: Pour $j \leftarrow 1$ à $i-1$ faire

4: $S \leftarrow S + A(i,j) * x(j)$

5: Fin Pour

6: $x_i \leftarrow (b_i - S)/A_{i,i}$

7: Fin Pour

$$x_i = \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j} x_j \right), \quad \forall i \in \llbracket 1, n \rrbracket.$$

Algorithme 2 Fonction **RSLTriINF** permettant de résoudre le système linéaire triangulaire inférieur inversible

$$\mathbb{A} \mathbf{x} = \mathbf{b}.$$

Données : \mathbb{A} : matrice triangulaire de $\mathcal{M}_n(\mathbb{K})$ inférieure inversible.
 \mathbf{b} : vecteur de \mathbb{K}^n .

Résultat : \mathbf{x} : vecteur de \mathbb{K}^n .

- 1: **Fonction** $\mathbf{x} \leftarrow \mathbf{RSLTriINF} (\mathbb{A}, \mathbf{b})$
- 2: **Pour** $i \leftarrow 1$ à n **faire**
- 3: $S \leftarrow 0$
- 4: **Pour** $j \leftarrow 1$ à $i - 1$ **faire**
- 5: $S \leftarrow S + A(i, j) * x(j)$
- 6: **Fin Pour**
- 7: $x(i) \leftarrow (b(i) - S) / A(i, i)$
- 8: **Fin Pour**
- 9: **Fin Fonction**

Système triangulaire supérieur

Soit $\mathbb{A} \in \mathcal{M}_n(\mathbb{K})$ triangulaire supérieure inversible ($A_{i,j} = 0$ si $i > j$)

$$\mathbb{A}\mathbf{x} = \mathbf{b} \iff \begin{pmatrix} A_{1,1} & \dots & \dots & A_{1,n} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & A_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ \vdots \\ \vdots \\ b_n \end{pmatrix}$$



Exercice

Ecrire la fonction **RSLTRISUP** permettant de résoudre le système triangulaire supérieure $\mathbb{A}\mathbf{x} = \mathbf{b}$.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

 **Lemme 1.1:** 

Soit $(i, j) \in \llbracket 1, n \rrbracket^2$. On note $\mathbb{P}_n^{[i,j]} \in \mathcal{M}_n(\mathbb{R})$ la matrice identité dont on a permuté les lignes i et j . Alors la matrice $\mathbb{P}_n^{[i,j]}$ est symétrique et orthogonale. Pour toute matrice $A \in \mathcal{M}_n(\mathbb{K})$,

- 1 la matrice $\mathbb{P}_n^{[i,j]}A$ est matrice A dont on a permuté les **lignes** i et j ,
- 2 la matrice $A\mathbb{P}_n^{[i,j]}$ est matrice A dont on a permuté les **colonnes** i et j ,

 **Lemme 1.2:** 

Soit $A \in \mathcal{M}_n(\mathbb{C})$ avec $A_{1,1} \neq 0$. Il existe une matrice $E \in \mathcal{M}_n(\mathbb{C})$ triangulaire inférieure à diagonale unité telle que

$$EA\mathbf{e}_1 = A_{1,1}\mathbf{e}_1 \quad (3)$$

où \mathbf{e}_1 est le premier vecteur de la base canonique de \mathbb{C}^n .



Théorème 2:



Soit $A \in \mathcal{M}_n(\mathbb{C})$. Il existe une matrice unitaire U et une matrice triangulaire supérieure T telles que

$$A = UTU^* \quad (4)$$

Plan

- 1 Méthodes directes
 - Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
 - Exercices et résultats préliminaires
 - **Méthode de Gauss-Jordan**
 - Ecriture algébrique
 - Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
 - Factorisation LDL*
 - Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

- 2 Normes vectorielles et normes matricielles
 - Normes vectorielles
 - Normes matricielles
 - Suites de vecteurs et de matrices
- 3 Conditionnement d'un système linéaire
- 4 Méthodes itératives
 - Principe
 - Notations
 - Méthode de Jacobi
 - Méthode de Gauss-Seidel
 - Méthode de relaxation
 - Etude de la convergence
 - Algorithmes
 - Principe de base
 - Méthode de Jacobi

Algorithme de Gauss-Jordan

$$\mathbf{Ax} = \mathbf{b} \iff \mathbf{Ux} = \mathbf{f}$$

où \mathbf{U} matrice triangulaire supérieure.

Opérations élémentaires sur les matrices :

- $\mathcal{L}_i \leftrightarrow \mathcal{L}_j$ permutation lignes i et j
- $\mathcal{L}_i \leftarrow \mathcal{L}_i + \alpha \mathcal{L}_j$ combinaison linéaire

A l'aide d'opérations élémentaires, on va transformer successivement en $n - 1$ étapes le système. A l'étape j , on va *s'arranger* pour annuler les termes sous-diagonaux de la colonne j de la matrice sans modifier les $j - 1$ premières colonnes.

$$\begin{array}{c}
 \xrightarrow{j-1} \\
 \left(\begin{array}{cccc|cccc}
 \bullet & \bullet & \dots & \bullet & \bullet & \dots & \bullet & \\
 0 & \dots & \dots & & & & & \\
 \vdots & \dots & \dots & & & & & \\
 0 & \dots & 0 & \bullet & \bullet & \dots & \bullet & \\
 \hline
 0 & \dots & \dots & 0 & \bullet & \dots & \bullet & \\
 \vdots & & & & \vdots & & \vdots & \\
 0 & \dots & \dots & 0 & \bullet & \dots & \bullet &
 \end{array} \right) \mathbf{x} = \begin{pmatrix} \bullet \\ \vdots \\ \bullet \\ \vdots \\ \bullet \end{pmatrix}
 \end{array}$$

Etape j



$$\begin{array}{c}
 \xrightarrow{j} \\
 \left(\begin{array}{cccc|cccc}
 \bullet & \bullet & \dots & \bullet & \bullet & \dots & \bullet & \\
 0 & \dots & \dots & & & & & \\
 \vdots & \dots & \dots & & & & & \\
 0 & \dots & 0 & \bullet & \bullet & \dots & \bullet & \\
 \hline
 0 & \dots & \dots & 0 & \bullet & \dots & \bullet & \\
 \vdots & & & & \vdots & & \vdots & \\
 0 & \dots & \dots & 0 & \bullet & \dots & \bullet &
 \end{array} \right) \mathbf{x} = \begin{pmatrix} \bullet \\ \vdots \\ \bullet \\ \vdots \\ \bullet \end{pmatrix}
 \end{array}$$

Algorithme 3 Algorithme de Gauss-Jordan formel pour la résolution de $\mathbb{A}\mathbf{x} = \mathbf{b}$

- 1: **Pour** $j \leftarrow 1$ à $n - 1$ **faire**
 - 2: Rechercher l'indice k de la ligne du pivot (sur la colonne j , $k \in \llbracket j, n \rrbracket$)
 - 3: Permuter les lignes j (\mathcal{L}_j) et k (\mathcal{L}_k) du système si besoin.
 - 4: **Pour** $i \leftarrow j + 1$ à n **faire**
 - 5: Eliminer en effectuant $\mathcal{L}_i \leftarrow \mathcal{L}_i - \frac{A_{ij}}{A_{jj}} \mathcal{L}_j$
 - 6: **Fin Pour**
 - 7: **Fin Pour**
 - 8: Résoudre le système triangulaire supérieur par la méthode de la remontée.
-

Algorithme 4 Algorithme de Gauss-Jordan avec fonctions pour la résolution de $\mathbb{A}\mathbf{x} = \mathbf{b}$

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$ inversible.

\mathbf{b} : vecteur de \mathbb{K}^n .

Résultat : \mathbf{x} : vecteur de \mathbb{R}^n .

```
1: Fonction  $\mathbf{x} \leftarrow \text{RSLGAUSS} (\mathbb{A}, \mathbf{b})$ 
2:   Pour  $j \leftarrow 1$  à  $n - 1$  faire
3:      $k \leftarrow \text{CHERCHEINDPIVOT}(\mathbb{A}, j)$  ▷ à écrire
4:      $[\mathbb{A}, \mathbf{b}] \leftarrow \text{PERMLIGNESYS}(\mathbb{A}, \mathbf{b}, j, k)$  ▷ à écrire
5:     Pour  $i \leftarrow j + 1$  à  $n$  faire
6:        $[\mathbb{A}, \mathbf{b}] \leftarrow \text{COMBLIGNESYS}(\mathbb{A}, \mathbf{b}, j, i, -A(i, j)/A(j, j))$  ▷ à écrire
7:     Fin Pour
8:   Fin Pour
9:    $\mathbf{x} \leftarrow \text{RSLTRISUP}(\mathbb{A}, \mathbf{b})$  ▷ déjà écrite
10: Fin Fonction
```

Algorithme 5 Recherche d'un pivot pour l'algorithme de Gauss-Jordan.

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$.

j : entier, $1 \leq j \leq n$.

Résultat : k : entier, indice ligne pivot

```
1: Fonction  $k \leftarrow$  CHERCHEINDPIVOT (  $\mathbb{A}, j$  )
2:  $k \leftarrow j$ , pivot  $\leftarrow |\mathbb{A}(j, j)|$ 
3: Pour  $i \leftarrow j + 1$  à  $n$  faire
4:   Si  $|\mathbb{A}(i, j)| >$  pivot alors
5:      $k \leftarrow i$ 
6:     pivot  $\leftarrow |\mathbb{A}(i, j)|$ 
7:   Fin Si
8: Fin Pour
9: Fin Fonction
```

Algorithme 6 Permutte deux lignes d'une matrice et d'un vecteur.

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$.

\mathbf{b} : vecteur de \mathbb{K}^n .

j, k : entiers, $1 \leq j, k \leq n$.

Résultat : \mathbb{A} et \mathbf{b} modifiés.

```
1: Fonction  $[\mathbb{A}, \mathbf{b}] \leftarrow$  PERMLIGNESYS (  $\mathbb{A}, \mathbf{b}, j, k$  )
2: Pour  $l \leftarrow 1$  à  $n$  faire
3:    $t \leftarrow \mathbb{A}(j, l)$ 
4:    $\mathbb{A}(j, l) \leftarrow \mathbb{A}(k, l)$ 
5:    $\mathbb{A}(k, l) \leftarrow t$ 
6: Fin Pour
7:  $t \leftarrow \mathbf{b}(j)$ ,  $\mathbf{b}(j) \leftarrow \mathbf{b}(k)$ ,  $\mathbf{b}(k) \leftarrow t$ 
8: Fin Fonction
```

Algorithme 7 Combinaison linéaire $\mathcal{L}_i \leftarrow \mathcal{L}_i + \alpha \mathcal{L}_j$ appliqué à une matrice et à un vecteur.

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$.

\mathbf{b} : vecteur de \mathbb{K}^n .

j, i : entiers, $1 \leq j, i \leq n$.

alpha : scalaire de \mathbb{K}

Résultat : \mathbb{A} et \mathbf{b} modifiés.

```
1: Fonction  $[\mathbb{A}, \mathbf{b}] \leftarrow$  COMBLIGNESYS (  $\mathbb{A}, \mathbf{b}, j, i, \alpha$  )
2: Pour  $k \leftarrow 1$  à  $n$  faire
3:    $\mathbb{A}(i, k) \leftarrow \mathbb{A}(i, k) + \alpha * \mathbb{A}(j, k)$ 
4: Fin Pour
5:  $\mathbf{b}(i) \leftarrow \mathbf{b}(i) + \alpha \mathbf{b}(j)$ 
6: Fin Fonction
```



Exercice: Méthode de Gauss, écriture algébrique



Soit $\mathbb{A} \in \mathcal{M}_n(\mathbb{C})$ inversible.

Q. 1

Montrer qu'il existe une matrice $\mathbb{G} \in \mathcal{M}_n(\mathbb{C})$ telle que $|\det(\mathbb{G})| = 1$ et $\mathbb{G}\mathbf{e}_1 = \alpha\mathbf{e}_1$ avec $\alpha \neq 0$ et \mathbf{e}_1 premier vecteur de la base canonique de \mathbb{C}^n .

Q. 2

- 1 Montrer par récurrence sur l'ordre des matrices que pour toute matrice $\mathbb{A}_n \in \mathcal{M}_n(\mathbb{C})$ inversible, il existe une matrice $\mathbb{S}_n \in \mathcal{M}_n(\mathbb{C})$ telle que $|\det \mathbb{S}_n| = 1$ et $\mathbb{S}_n\mathbb{A}_n = \mathbb{U}_n$ avec \mathbb{U}_n matrice triangulaire supérieure inversible.
- 2 Soit $\mathbf{b} \in \mathbb{C}^n$. En supposant connue la décomposition précédente $\mathbb{S}_n\mathbb{A}_n = \mathbb{U}_n$, expliquer comment résoudre le système $\mathbb{A}_n\mathbf{x} = \mathbf{b}$.

Q. 3

Que peut-on dire si \mathbb{A} est non inversible?

Indication : utiliser les Lemmes 1.1 et 1.2.

On a donc démontré le théorème suivant



Théorème 3

Soit A une matrice carrée, inversible ou non. Il existe (au moins) une matrice inversible G telle que GA soit triangulaire supérieure.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- **Factorisation LU**
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi



Exercice: Factorisation LU



Soit $A \in \mathcal{M}_n(\mathbb{C})$ une matrice dont les sous-matrices principales d'ordre i , notées Δ_i , $i \in \llbracket 1, n \rrbracket$ (voir Définition B.46, page 176) sont inversibles. Montrer qu'il existe des matrices $E^{[k]} \in \mathcal{M}_n(\mathbb{C})$, $k \in \llbracket 1, n-1 \rrbracket$, triangulaires inférieures à diagonale unité telles que la matrice U définie par

$$U = E^{[n-1]} \dots E^{[1]} A$$

soit triangulaire supérieure avec $U_{i,i} = \det \Delta_i / (U_{1,1} \times \dots \times U_{i-1,i-1})$, $\forall i \in \llbracket 1, n \rrbracket$.



Théorème 4: Factorisation LU



Soit $A \in \mathcal{M}_n(\mathbb{C})$ une matrice dont les sous-matrices principales sont inversibles alors il existe une unique matrice $L \in \mathcal{M}_n(\mathbb{C})$ triangulaire inférieure (*lower triangular* en anglais) à diagonale unité et une unique matrice $U \in \mathcal{M}_n(\mathbb{C})$ triangulaire supérieure (*upper triangular* en anglais) inversible telles que

$$A = LU.$$

preuve :

- **Existence** : exercice précédant $U = E^{[n-1]} \dots E^{[1]}A$

$$L = \left(E^{[n-1]} \dots E^{[1]} \right)^{-1}$$

- **Unicité** : $A = L_1 U_1 = L_2 U_2 \dots$



Corollaire 4.1:



Si $A \in \mathcal{M}_n(\mathbb{C})$ est une matrice hermitienne définie positive alors elle admet une unique factorisation LU.

preuve : A hermitienne définie positive alors toutes ses sous-matrices principales sont définies positives et donc inversibles.

Remarque 4.2

Si la matrice $\mathbb{A} \in \mathcal{M}_n(\mathbb{C})$ est inversible mais que ses sous-matrices principales ne sont pas toutes inversibles, il est possible par des permutations préalables de lignes de la matrice de se ramener à une matrice telle que ses sous-matrices principales soient inversibles.



Théorème 5: Factorisation LU avec permutations



Soit $\mathbb{A} \in \mathcal{M}_n(\mathbb{C})$ une matrice inversible. Il existe une matrice \mathbb{P} , produit de matrices de permutation, une matrice $\mathbb{L} \in \mathcal{M}_n(\mathbb{C})$ triangulaire inférieure à diagonale unité et une matrice $\mathbb{U} \in \mathcal{M}_n(\mathbb{C})$ triangulaire supérieure telles que

$$\mathbb{P}\mathbb{A} = \mathbb{L}\mathbb{U}. \quad (5)$$

Utilisation pratique de la factorisation LU

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU

Trouver $\mathbf{x} \in \mathbb{K}^n$ tel que

$$A\mathbf{x} = \mathbf{b} \iff LU\mathbf{x} = \mathbf{b} \quad (6)$$

est équivalent à

Utilisation pratique de la factorisation LU

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU

Trouver $\mathbf{x} \in \mathbb{K}^n$ tel que

$$A\mathbf{x} = \mathbf{b} \iff LU\mathbf{x} = \mathbf{b} \quad (6)$$

est équivalent à

Trouver $\mathbf{x} \in \mathbb{K}^n$ solution de

$$U\mathbf{x} = \mathbf{y} \quad (7)$$

avec $\mathbf{y} \in \mathbb{K}^n$ solution de

$$L\mathbf{y} = \mathbf{b}. \quad (8)$$

Algorithme de résolution de systèmes linéaire par LU

Algorithme 8 Fonction RSLFactLU permettant de résoudre, par une factorisation LU, le système linéaire

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

où \mathbb{A} une matrice de $\mathcal{M}_n(\mathbb{R})$ définie positive et $\mathbf{b} \in \mathbb{R}^n$.

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{R})$ dont les sous-matrices principales sont inversibles définie positive,
 \mathbf{b} : vecteur de \mathbb{R}^n .

Résultat : \mathbf{x} : vecteur de \mathbb{R}^n .

- 1: **Fonction** $\mathbf{x} \leftarrow \text{RSLFACTLU} (\mathbb{A}, \mathbf{b})$
- 2: $[\mathbb{L}, \mathbb{U}] \leftarrow \text{FACTLU}(\mathbb{A})$ ▷ Factorisation LU
- 3: $\mathbf{y} \leftarrow \text{RSLTRIINF}(\mathbb{L}, \mathbf{b})$ ▷ Résolution du système $\mathbb{L}\mathbf{y} = \mathbf{b}$
- 4: $\mathbf{x} \leftarrow \text{RSLTRISUP}(\mathbb{U}, \mathbf{y})$ ▷ Résolution du système $\mathbb{U}\mathbf{x} = \mathbf{y}$
- 5: **Fin Fonction**

Il nous faut donc écrire la fonction **FACTLU**

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} & \dots & \dots & U_{1,n} \\ 0 & U_{2,2} & \dots & \dots & U_{2,n} \\ 0 & 0 & U_{3,3} & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & U_{n,n} \end{pmatrix}$$

On connaît A , on cherche L et U

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} & \dots & \dots & U_{1,n} \\ 0 & U_{2,2} & \dots & \dots & U_{2,n} \\ 0 & 0 & U_{3,3} & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & U_{n,n} \end{pmatrix}$$

- Etape 1 :

- ▶ On connaît la **première ligne** de $L \implies$ on peut calculer la **première ligne** de U

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} & \dots & \dots & U_{1,n} \\ 0 & U_{2,2} & \dots & \dots & U_{2,n} \\ 0 & 0 & U_{3,3} & \dots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & U_{n,n} \end{pmatrix}$$

• Etape 1 :

- ▶ On connaît la **première ligne** de $L \implies$ on peut calculer la **première ligne** de U
- ▶ On connaît la **première colonne** de $U \implies$ on peut calculer la **première colonne** de L

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} & \dots & \dots & U_{1,n} \\ 0 & U_{2,2} & \dots & \dots & U_{2,n} \\ 0 & 0 & U_{3,3} & \dots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & U_{n,n} \end{pmatrix}$$

- **Etape 1 :**

- ▶ On connaît la **première ligne** de $L \implies$ on peut calculer la **première ligne** de U
- ▶ On connaît la **première colonne** de $U \implies$ on peut calculer la **première colonne** de U

- **Etape 2 :**

- ▶ On connaît la **deuxième ligne** de $L \implies$ on peut calculer la **deuxième ligne** de U car on connaît la première ligne de U

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} & \dots & \dots & U_{1,n} \\ 0 & U_{2,2} & \dots & \dots & U_{2,n} \\ 0 & 0 & U_{3,3} & \dots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & U_{n,n} \end{pmatrix}$$

- **Etape 1 :**

- ▶ On connaît la **première ligne** de $L \implies$ on peut calculer la **première ligne** de U
- ▶ On connaît la **première colonne** de $U \implies$ on peut calculer la **première colonne** de L

- **Etape 2 :**

- ▶ On connaît la **deuxième ligne** de $L \implies$ on peut calculer la **deuxième ligne** de U car on connaît la première ligne de U
- ▶ On connaît la **deuxième colonne** de $U \implies$ on peut calculer la **deuxième colonne** de L car on connaît la première colonne de L

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation LU.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} & \dots & \dots & U_{1,n} \\ 0 & U_{2,2} & \dots & \dots & U_{2,n} \\ 0 & 0 & U_{3,3} & \dots & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & U_{n,n} \end{pmatrix}$$

- **Etape 1 :**

- ▶ On connaît la **première ligne** de $L \implies$ on peut calculer la **première ligne** de U
- ▶ On connaît la **première colonne** de $U \implies$ on peut calculer la **première colonne** de L

- **Etape 2 :**

- ▶ On connaît la **deuxième ligne** de $L \implies$ on peut calculer la **deuxième ligne** de U car on connaît la première ligne de U
- ▶ On connaît la **deuxième colonne** de $U \implies$ on peut calculer la **deuxième colonne** de L car on connaît la première colonne de L

- ...

Soit $A \in \mathcal{M}_n(\mathbb{K})$ admettant une factorisation $\mathbb{L}\mathbb{U}$.

$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \ddots & 0 \\ L_{n,1} & L_{n,2} & \dots & L_{n,n-1} & 1 \end{pmatrix} \begin{pmatrix} U_{1,1} & U_{1,2} & \dots & \dots & U_{1,n} \\ 0 & U_{2,2} & \dots & \dots & U_{2,n} \\ 0 & 0 & U_{3,3} & & \vdots \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \dots & 0 & U_{n,n} \end{pmatrix}$$

- **Etape i :**

On connaît les $i - 1$ premières colonnes de \mathbb{L} et les $i - 1$ premières lignes de \mathbb{U} .

Peut-on calculer la colonne i de \mathbb{L} et la ligne i de \mathbb{U} ?

Par récurrence, en supposant les $i - 1$ premières colonnes de \mathbb{L} et les $i - 1$ premières lignes de \mathbb{U} .

$$\mathbb{A} = \left(\begin{array}{cccc|cccc} \bullet & 0 & \dots & 0 & 0 & \dots & \dots & 0 \\ \bullet & \dots & \dots & \vdots & \vdots & & & \vdots \\ \vdots & \dots & \dots & \vdots & \vdots & & & \vdots \\ \bullet & \dots & \bullet & \bullet & 0 & \dots & \dots & 0 \\ \hline \bullet & \dots & \dots & \bullet & L_{i,i} & 0 & \dots & 0 \\ \vdots & & & \vdots & \bullet & \dots & \dots & \vdots \\ \vdots & & & \vdots & \vdots & \dots & \dots & 0 \\ \bullet & \dots & \dots & \bullet & \bullet & \dots & \bullet & L_{n,n} \end{array} \right) \left(\begin{array}{cccc|cccc} \bullet & \bullet & \dots & \bullet & \bullet & \dots & \dots & \bullet \\ 0 & \dots & \dots & \vdots & \vdots & & & \vdots \\ \vdots & \dots & \dots & \vdots & \vdots & & & \vdots \\ 0 & \dots & 0 & \bullet & \bullet & \dots & \dots & \bullet \\ \hline 0 & \dots & \dots & 0 & U_{i,i} & \bullet & \dots & \bullet \\ \vdots & & & \vdots & 0 & \dots & \dots & \vdots \\ \vdots & & & \vdots & \vdots & \dots & \dots & \bullet \\ 0 & \dots & \dots & 0 & 0 & \dots & 0 & U_{n,n} \end{array} \right)$$

Par récurrence, en supposant les $i - 1$ premières colonnes de \mathbb{L} et les $i - 1$ premières lignes de \mathbb{U} .

$$\mathbb{A} = \left(\begin{array}{cccc|cccc} \bullet & 0 & \dots & 0 & 0 & \dots & \dots & 0 \\ \bullet & \dots & \dots & \vdots & \vdots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \vdots & \vdots & \dots & \dots & \vdots \\ \bullet & \dots & \bullet & \bullet & 0 & \dots & \dots & 0 \\ \hline \bullet & \dots & \dots & \bullet & L_{i,i} & 0 & \dots & 0 \\ \vdots & \dots & \dots & \vdots & \bullet & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \vdots & \vdots & \dots & \dots & 0 \\ \bullet & \dots & \dots & \bullet & \bullet & \dots & \bullet & L_{n,n} \end{array} \right) \left(\begin{array}{cccc|cccc} \bullet & \bullet & \dots & \bullet & \bullet & \dots & \dots & \bullet \\ 0 & \dots & \dots & \vdots & \vdots & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \vdots & \vdots & \dots & \dots & \vdots \\ 0 & \dots & 0 & \bullet & \bullet & \dots & \dots & \bullet \\ \hline 0 & \dots & \dots & 0 & U_{i,i} & \bullet & \dots & \bullet \\ \vdots & \dots & \dots & \vdots & 0 & \dots & \dots & \vdots \\ \vdots & \dots & \dots & \vdots & \vdots & \dots & \dots & \bullet \\ 0 & \dots & \dots & 0 & 0 & \dots & 0 & U_{n,n} \end{array} \right)$$

$$A = \left(\begin{array}{c|ccc}
 \begin{array}{ccc} \bullet & 0 & \dots & 0 \\ \bullet & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \bullet & \dots & \bullet & \bullet \end{array} & 0 & \dots & \dots & 0 \\
 \hline
 \bullet & \dots & \dots & \bullet & L_{i,i} & 0 & \dots & 0 \\
 \vdots & & & \vdots & \bullet & \ddots & \ddots & \vdots \\
 \vdots & & & \vdots & \vdots & \ddots & \ddots & 0 \\
 \bullet & \dots & \dots & \bullet & \bullet & \dots & \bullet & L_{n,n}
 \end{array} \right) \left(\begin{array}{c|ccc}
 \begin{array}{ccc} \bullet & \bullet & \dots & \bullet \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \bullet \end{array} & \dots & \dots & \dots & \bullet & \dots & \dots & \bullet \\
 \hline
 0 & \dots & \dots & 0 & U_{i,i} & \bullet & \dots & \bullet \\
 \vdots & & & \vdots & 0 & \ddots & \ddots & \vdots \\
 \vdots & & & \vdots & \vdots & \ddots & \ddots & \bullet \\
 0 & \dots & \dots & 0 & 0 & \dots & 0 & U_{n,n}
 \end{array} \right)$$

\mathbb{L}
 \mathbb{U}

Connus
Connus

$i-1$
 $i-1$

Par récurrence, on suppose connues les $i - 1$ premières colonnes de \mathbb{L} et les $i - 1$ premières lignes de \mathbb{U} .

Peut-on calculer la colonne i de \mathbb{L} et la ligne i de \mathbb{U} ?

$$\mathbb{A} = \left(\begin{array}{c|ccc}
 \begin{array}{ccc} \bullet & 0 & \dots & 0 \end{array} & 0 & \dots & 0 \\
 \begin{array}{ccc} \bullet & \dots & \dots & \vdots \end{array} & \vdots & & \vdots \\
 \begin{array}{ccc} \vdots & \dots & \dots & \vdots \end{array} & \vdots & & \vdots \\
 \begin{array}{ccc} \bullet & \dots & \bullet & \bullet \end{array} & 0 & \dots & 0 \\
 \begin{array}{ccc} \bullet & \dots & \dots & \bullet \end{array} & \boxed{L_{i,i}} & 0 & \dots & 0 \\
 \begin{array}{ccc} \vdots & & & \vdots \end{array} & \bullet & \dots & \vdots \\
 \begin{array}{ccc} \vdots & & & \vdots \end{array} & \bullet & \dots & 0 \\
 \begin{array}{ccc} \bullet & \dots & \dots & \bullet \end{array} & \bullet & \dots & L_{n,n}
 \end{array} \right) \left(\begin{array}{c|ccc}
 \begin{array}{ccc} \bullet & \bullet & \dots & \bullet \end{array} & \bullet & \dots & \bullet \\
 0 & \dots & \dots & \vdots \\
 \vdots & \dots & \dots & \vdots \\
 0 & \dots & 0 & \bullet & \dots & \bullet \\
 0 & \dots & \dots & 0 & U_{i,i} & \bullet & \dots & \bullet \\
 \vdots & & & \vdots & 0 & \dots & \vdots \\
 \vdots & & & \vdots & \vdots & \dots & \vdots \\
 \vdots & & & \vdots & \vdots & \dots & \bullet \\
 0 & \dots & \dots & 0 & 0 & \dots & 0 & U_{n,n}
 \end{array} \right)$$

\mathbb{L} (top left), \mathbb{U} (bottom right), \mathbb{C}_{onvus} (top left and bottom right), \mathbb{U} (bottom right)

$L_{i,i} = 1$
 \Rightarrow ligne i de \mathbb{L} \mathbb{C}_{onvus}

$$\mathbb{A} = \left(\begin{array}{c|ccc}
 \overbrace{\begin{matrix} \bullet & 0 & \dots & 0 \\ \bullet & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ \bullet & \dots & \bullet & \bullet \\ \bullet & \dots & \dots & \bullet \end{matrix}}^{i-1} & \begin{matrix} 0 \\ \vdots \\ 0 \\ L_{i,i} \\ \vdots \\ \bullet \\ \vdots \\ \bullet \end{matrix} & \dots & \dots \\
 \hline
 & 0 & \dots & 0 \\
 \hline
 & \vdots & \ddots & \vdots \\
 & \vdots & \ddots & 0 \\
 & \bullet & \dots & L_{n,n}
 \end{array} \right) \left(\begin{array}{c|ccc}
 \overbrace{\begin{matrix} \bullet & \bullet & \dots & \bullet \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \bullet \\ 0 & \dots & 0 & \bullet \end{matrix}}^{\text{connus}} & \begin{matrix} \vdots \\ \vdots \\ \vdots \\ U_{i,i} \\ \vdots \\ 0 \\ \vdots \\ 0 \end{matrix} & \dots & \dots \\
 \hline
 & 0 & \dots & 0 \\
 \hline
 & \vdots & \ddots & \vdots \\
 & \vdots & \ddots & \bullet \\
 & 0 & \dots & 0 & U_{n,n}
 \end{array} \right)$$

\mathbb{L}
 \mathbb{U}

On peut calculer la colonne i de \mathbb{L} \Leftrightarrow On connaît la colonne i de \mathbb{U}

On cherche $L_{j,i} \forall j \in \llbracket i+1, n \rrbracket$, ($L_{i,i} = 1$)

$$A_{j,i} \stackrel{\text{def}}{=} \sum_{k=1}^n L_{j,k} U_{k,i} = \sum_{k=1}^{i-1} \overbrace{L_{j,k} U_{k,i}}^{\text{connus}} + L_{j,i} \overbrace{U_{i,i}}^{\text{connu}} + \sum_{k=1}^{i+1} L_{j,k} \overbrace{U_{k,i}}^{=0}$$

$$L_{j,i} = \left(A_{j,i} - \sum_{k=1}^{i-1} L_{j,k} U_{k,i} \right) / U_{i,i}, \quad \forall j \in \llbracket i+1, n \rrbracket.$$

Algorithme 9 \mathcal{R}_0

1: Calculer les matrices \mathbb{L} et \mathbb{U}

Algorithme 9 \mathcal{R}_1

- 1: **Pour** $i \leftarrow 1$ à n faire
- 2: Calculer la ligne i de \mathbb{U} .
- 3: Calculer la colonne i de \mathbb{L} .
- 4: **Fin Pour**

Algorithme 9 \mathcal{R}_1

- 1: Pour $i \leftarrow 1$ à n faire
- 2: Calculer la ligne i de U .
- 3: Calculer la colonne i de L .
- 4: Fin Pour

Algorithme 9 \mathcal{R}_2

- 1: Pour $i \leftarrow 1$ à n faire
- 2: Pour $j \leftarrow 1$ à $i - 1$ faire
- 3: $U(i, j) \leftarrow 0$
- 4: Fin Pour
- 5: Pour $j \leftarrow i$ à n faire
- 6: $U_{i,j} \leftarrow A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} U_{k,j}$
- 7: Fin Pour
- 8: Pour $j \leftarrow 1$ à $i - 1$ faire
- 9: $L_{j,i} \leftarrow 0$
- 10: Fin Pour
- 11: $L_{j,i} \leftarrow 1$
- 12: Pour $j \leftarrow i + 1$ à n faire
- 13: $L_{j,i} \leftarrow \frac{1}{U_{i,i}} \left(A_{j,i} - \sum_{k=1}^{i-1} L_{j,k} U_{k,i} \right)$
- 14: Fin Pour
- 15: Fin Pour

Algorithme 9 \mathcal{R}_2

- 1: Pour $i \leftarrow 1$ à n faire
- 2: Pour $j \leftarrow 1$ à $i-1$ faire
- 3: $U(i,j) \leftarrow 0$
- 4: Fin Pour
- 5: Pour $j \leftarrow i$ à n faire
- 6:
$$U_{i,j} \leftarrow A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} U_{k,j}$$
- 7: Fin Pour
- 8: Pour $j \leftarrow 1$ à $i-1$ faire
- 9: $L_{j,i} \leftarrow 0$
- 10: Fin Pour
- 11: $L_{i,i} \leftarrow 1$
- 12: Pour $j \leftarrow i+1$ à n faire
- 13:
$$L_{j,i} \leftarrow \frac{1}{U_{i,i}} \left(A_{j,i} - \sum_{k=1}^{i-1} L_{j,k} U_{k,i} \right)$$
- 14: Fin Pour
- 15: Fin Pour

Algorithme 9 \mathcal{R}_3

- 1: Pour $i \leftarrow 1$ à n faire
- 2: Pour $j \leftarrow 1$ à $i-1$ faire
- 3: $U(i,j) \leftarrow 0$
- 4: Fin Pour
- 5: Pour $j \leftarrow i$ à n faire
- 6:
$$S_1 \leftarrow \sum_{k=1}^{i-1} L_{i,k} U_{k,j}$$
- 7:
$$U_{i,j} \leftarrow A_{i,j} - S_1$$
- 8: Fin Pour
- 9: Pour $j \leftarrow 1$ à $i-1$ faire
- 10: $L_{j,i} \leftarrow 0$
- 11: Fin Pour
- 12: $L_{i,i} \leftarrow 1$
- 13: Pour $j \leftarrow i+1$ à n faire
- 14:
$$S_2 \leftarrow \sum_{k=1}^{i-1} L_{j,k} U_{k,i}$$
- 15:
$$L_{j,i} \leftarrow \frac{1}{U_{i,i}} (A_{j,i} - S_2)$$
- 16: Fin Pour
- 17: Fin Pour

Algorithme 9 \mathcal{R}_3

```

1: Pour  $i \leftarrow 1$  à  $n$  faire
2:   Pour  $j \leftarrow 1$  à  $i - 1$  faire
3:      $U(i, j) \leftarrow 0$ 
4:   Fin Pour
5:   Pour  $j \leftarrow i$  à  $n$  faire
6:      $S_1 \leftarrow \sum_{k=1}^{i-1} L_{i,k} U_{k,j}$ 
7:      $U_{i,j} \leftarrow A_{i,j} - S_1$ 
8:   Fin Pour
9:   Pour  $j \leftarrow 1$  à  $i - 1$  faire
10:     $L_{j,i} \leftarrow 0$ 
11:   Fin Pour
12:    $L_{i,i} \leftarrow 1$ 
13:   Pour  $j \leftarrow i + 1$  à  $n$  faire
14:     $S_2 \leftarrow \sum_{k=1}^{i-1} L_{j,k} U_{k,i}$ 
15:     $L_{j,i} \leftarrow \frac{1}{U_{i,i}} (A_{j,i} - S_2)$ 
16:   Fin Pour
17: Fin Pour

```

Algorithme 9 \mathcal{R}_4

```

1: Pour  $i \leftarrow 1$  à  $n$  faire
2:   Pour  $j \leftarrow 1$  à  $i - 1$  faire
3:      $U(i, j) \leftarrow 0$ 
4:   Fin Pour
5:   Pour  $j \leftarrow i$  à  $n$  faire
6:      $S_1 \leftarrow 0$ 
7:     Pour  $k \leftarrow 1$  à  $i - 1$  faire
8:        $S_1 \leftarrow S_1 + L_{i,k} * U_{k,j}$ 
9:     Fin Pour
10:     $U_{i,j} \leftarrow A_{i,j} - S_1$ 
11:   Fin Pour
12:   Pour  $j \leftarrow 1$  à  $i - 1$  faire
13:     $L_{j,i} \leftarrow 0$ 
14:   Fin Pour
15:    $L_{i,i} \leftarrow 1$ 
16:   Pour  $j \leftarrow i + 1$  à  $n$  faire
17:     $S_2 \leftarrow 0$ 
18:    Pour  $k \leftarrow 1$  à  $i - 1$  faire
19:       $S_2 \leftarrow S_2 + L_{j,k} * U_{k,i}$ 
20:    Fin Pour
21:     $L_{j,i} \leftarrow \frac{1}{U_{i,i}} (A_{j,i} - S_2)$ 
22:   Fin Pour
23: Fin Pour

```

Algorithme 9 Fonction **FACTLU** permet de calculer les matrices \mathbb{L} et \mathbb{U} dites matrice de factorisation LU associée à la matrice \mathbb{A} , telle que

$$\mathbb{A} = \mathbb{L}\mathbb{U}$$

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$ dont les sous-matrices principales sont inversibles.

Résultat : \mathbb{L} : matrice de $\mathcal{M}_n(\mathbb{K})$ triangulaire inférieure avec $L_{i,i} = 1, \forall i \in \llbracket 1, n \rrbracket$

\mathbb{U} : matrice de $\mathcal{M}_n(\mathbb{K})$ triangulaire supérieure.

```
1: Fonction  $[\mathbb{L}, \mathbb{U}] \leftarrow \text{FACTLU}(\mathbb{A})$ 
2:    $\mathbb{U} \leftarrow \mathbb{0}_n$  ▷  $\mathbb{0}_n$  matrice nulle  $n \times n$ 
3:    $\mathbb{L} \leftarrow \mathbb{I}_n$  ▷  $\mathbb{I}_n$  matrice identité  $n \times n$ 
4:   Pour  $i \leftarrow 1$  à  $n$  faire
5:     Pour  $j \leftarrow i$  à  $n$  faire ▷ Calcul de la ligne  $i$  de  $\mathbb{U}$ 
6:        $S_1 \leftarrow 0$ 
7:       Pour  $k \leftarrow 1$  à  $i - 1$  faire
8:          $S_1 \leftarrow S_1 + L(i, k) * U(k, j)$ 
9:       Fin Pour
10:       $U(i, j) \leftarrow A(i, j) - S_1$ 
11:    Fin Pour
12:    Pour  $j \leftarrow i + 1$  à  $n$  faire ▷ Calcul de la colonne  $i$  de  $\mathbb{L}$ 
13:       $S_2 \leftarrow 0$ 
14:      Pour  $k \leftarrow 1$  à  $i - 1$  faire
15:         $S_2 \leftarrow S_2 + L(j, k) * U(k, i)$ 
16:      Fin Pour
17:       $L(j, i) \leftarrow (A_{j,i} - S_2) / U(i, i)$ 
18:    Fin Pour
19:  Fin Pour
20: Fin Fonction
```

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- **Factorisation LDL***
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

Soit $A \in \mathcal{M}_n(\mathbb{C})$ **hermitienne inversible** admettant une factorisation LU.
On pose

$$D = \text{diag } U \text{ et } R = D^{-1}U.$$

R est alors triangulaire supérieure à diagonale unité. On a alors

$$A = LU = LDD^{-1}U = LDR.$$

$$A \text{ hermitienne } A^* = A \implies A = R^*(D^*L^*) = L(DR)$$

Par unicité de la factorisation LU :

$$R^* = L \text{ et } D^*L^* = DR \implies R^* = L \text{ et } D^* = D$$



Théorème 6: Factorisation LDL*



Soit $A \in \mathcal{M}_n(\mathbb{C})$ une matrice hermitienne inversible admettant une factorisation LU. Alors A s'écrit sous la forme

$$A = LDL^* \quad (9)$$

où $D = \text{diag } U$ est une matrice à coefficients réels.



Corollaire 6.1:



Une matrice $A \in \mathcal{M}_n(\mathbb{C})$ admet une factorisation LDL* avec $L \in \mathcal{M}_n(\mathbb{C})$ matrice triangulaire inférieure à diagonale unité et $D \in \mathcal{M}_n(\mathbb{R})$ matrice diagonale à coefficients diagonaux strictement positifs **si et seulement si** la matrice A est hermitienne définie positive.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- **Factorisation de Cholesky**
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

♥ Definition

Une **factorisation régulière de Cholesky** d'une matrice $\mathbb{A} \in \mathcal{M}_n(\mathbb{C})$ est une factorisation $\mathbb{A} = \mathbb{B}\mathbb{B}^*$ où \mathbb{B} est une matrice triangulaire inférieure inversible.

Si les coefficients diagonaux de \mathbb{B} sont positifs, on parle alors d'une **factorisation positive de Cholesky**.



Théorème: Factorisation de Cholesky



La matrice $\mathbb{A} \in \mathcal{M}_n(\mathbb{C})$ admet une factorisation régulière de Cholesky **si et seulement si** la matrice \mathbb{A} est hermitienne définie positive. Dans ce cas, elle admet une unique factorisation positive.

Soit $A \in \mathcal{M}_n(\mathbb{C})$ hermitienne définie positive et $\mathbf{b} \in \mathbb{C}^n$. On note B la matrice de factorisation positive de Cholesky de A .

Trouver $\mathbf{x} \in \mathbb{C}^n$ tel que

$$A\mathbf{x} = \mathbf{b} \quad (\iff) \quad BB^*\mathbf{x} = \mathbf{b} \quad (10)$$

est équivalent à

Soit $A \in \mathcal{M}_n(\mathbb{C})$ hermitienne définie positive et $\mathbf{b} \in \mathbb{C}^n$. On note B la matrice de factorisation positive de Cholesky de A .

Trouver $\mathbf{x} \in \mathbb{C}^n$ tel que

$$A\mathbf{x} = \mathbf{b} \quad (\iff B B^* \mathbf{x} = \mathbf{b}) \quad (10)$$

est équivalent à

Trouver $\mathbf{x} \in \mathbb{C}^n$ solution de

$$B^* \mathbf{x} = \mathbf{y} \quad (11)$$

avec $\mathbf{y} \in \mathbb{C}^n$ solution de

$$B\mathbf{y} = \mathbf{b}. \quad (12)$$

Algorithme de résolution de systèmes linéaire par Cholesky

Algorithme 10 Fonction **RSLCHOLESKY** permettant de résoudre, par une factorisation de Cholesky positive, le système linéaire

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

où \mathbb{A} une matrice hermitienne de $\mathcal{M}_n(\mathbb{C})$ définie positive et $\mathbf{b} \in \mathbb{C}^n$.

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{C})$ symétrique définie positive,
 \mathbf{b} : vecteur de \mathbb{C}^n .

Résultat : \mathbf{x} : vecteur de \mathbb{C}^n .

- 1: **Fonction** $\mathbf{x} \leftarrow \mathbf{RSLCHOLESKY}(\mathbb{A}, \mathbf{b})$
- 2: $\mathbb{B} \leftarrow \mathbf{CHOLESKY}(\mathbb{A})$ ▷ Factorisation positive de Cholesky
- 3: $\mathbf{y} \leftarrow \mathbf{RSLTRIINF}(\mathbb{B}, \mathbf{b})$ ▷ Résolution du système $\mathbb{B}\mathbf{y} = \mathbf{b}$
- 4: $\mathbb{U} \leftarrow \mathbf{MATADJOINTE}(\mathbb{B})$ ▷ Calcul de la matrice adjointe de \mathbb{B}
- 5: $\mathbf{x} \leftarrow \mathbf{RSLTRISUP}(\mathbb{U}, \mathbf{y})$ ▷ Résolution du système $\mathbb{B}^*\mathbf{x} = \mathbf{y}$
- 6: **Fin Fonction**

Il nous faut donc écrire la fonction **CHOLESKY**

Soit $A \in \mathcal{M}_n(\mathbb{C})$ hermitienne définie positive. Il existe une unique matrice $B \in \mathcal{M}_n(\mathbb{C})$ triangulaire inférieure avec $B_{i,i} \in \mathbb{R}^{+*}$, $\forall i \in \llbracket 1, n \rrbracket$, telle que

$$A = BB^*$$

c'est à dire

$$\begin{pmatrix} A_{1,1} & \dots & A_{1,n} \\ \vdots & \ddots & \dots \\ A_{n,1} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} B_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ B_{n,1} & \dots & \dots & B_{n,n} \end{pmatrix} \begin{pmatrix} \overline{B_{1,1}} & \dots & \dots & \overline{B_{n,1}} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \overline{B_{n,n}} \end{pmatrix}.$$

Soit $A \in \mathcal{M}_n(\mathbb{C})$ hermitienne définie positive. Il existe une unique matrice $B \in \mathcal{M}_n(\mathbb{C})$ triangulaire inférieure avec $B_{i,i} \in \mathbb{R}^{+*}$, $\forall i \in \llbracket 1, n \rrbracket$, telle que

$$A = BB^*$$

c'est à dire

$$\begin{pmatrix} A_{1,1} & \dots & A_{1,n} \\ \vdots & \ddots & \dots \\ A_{n,1} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} B_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ B_{n,1} & \dots & \dots & B_{n,n} \end{pmatrix} \begin{pmatrix} \overline{B_{1,1}} & \dots & \dots & \overline{B_{n,1}} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \overline{B_{n,n}} \end{pmatrix}.$$

- Calcul de $B_{1,1}$ (la 1ère ligne de B est donc déterminée)
 \implies calcul 1ère colonne de B .

Soit $A \in \mathcal{M}_n(\mathbb{C})$ hermitienne définie positive. Il existe une unique matrice $B \in \mathcal{M}_n(\mathbb{C})$ triangulaire inférieure avec $B_{i,i} \in \mathbb{R}^{+*}$, $\forall i \in \llbracket 1, n \rrbracket$, telle que

$$A = BB^*$$

c'est à dire

$$\begin{pmatrix} A_{1,1} & \dots & A_{1,n} \\ \vdots & \ddots & \dots \\ A_{n,1} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} B_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ B_{n,1} & \dots & \dots & B_{n,n} \end{pmatrix} \begin{pmatrix} \overline{B_{1,1}} & \dots & \dots & \overline{B_{n,1}} \\ 0 & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & \overline{B_{n,n}} \end{pmatrix}.$$

- Calcul de $B_{1,1}$ (la 1ère ligne de B est donc déterminée)
 \implies calcul 1ère colonne de B .
- Puis calcul de $B_{2,2}$ (la 2ème ligne de B est donc déterminée)
 \implies calcul 2ème colonne de B .
- Etc...

Soit $i \in \llbracket 1, n \rrbracket$. On suppose connues les $i - 1$ premières colonnes de \mathbb{B} .

Peut-on calculer la colonne i de \mathbb{B} ?

$$\mathbb{A} = \mathbb{B}\mathbb{B}^* \implies A_{i,i} = \sum_{k=1}^n B_{i,k}(\mathbb{b}^*)_{k,i} = \sum_{k=1}^n B_{i,k} \overline{B_{i,k}}$$

Or \mathbb{B} triangulaire inférieure (i.e. $B_{i,j} = 0$ si $j > i$)

$$A_{i,i} = \sum_{k=1}^{i-1} |B_{i,k}|^2 + |B_{i,i}|^2$$

et donc

$$B_{i,i} = \left(A_{i,i} - \sum_{j=1}^{i-1} |B_{i,j}|^2 \right)^{1/2} .$$

Il reste à déterminer $B_{j,i}$, $\forall j \in \llbracket i + 1, n \rrbracket$.

$$A_{j,i} = \sum_{k=1}^n B_{j,k} (\mathbb{B}^*)_{k,i} = \sum_{k=1}^n B_{j,k} \overline{B_{i,k}}, \quad \forall j \in \llbracket i + 1, n \rrbracket$$

Comme \mathbb{L} est triangulaire inférieure on obtient

$$A_{j,i} = \sum_{k=1}^i B_{j,k} \overline{B_{i,k}} = \sum_{k=1}^{i-1} B_{j,k} \overline{B_{i,k}} + B_{j,i} \overline{B_{i,i}}, \quad \forall j \in \llbracket i + 1, n \rrbracket$$

Or $B_{i,i} > 0$ connu et les $i - 1$ premières colonnes de \mathbb{B} aussi.

$$B_{j,i} = \frac{1}{B_{i,i}} \left(A_{j,i} - \sum_{k=1}^{i-1} B_{j,k} \overline{B_{i,k}} \right), \quad \forall j \in \llbracket i + 1, n \rrbracket$$

$$B_{j,i} = 0, \quad \forall j \in \llbracket 1, i - 1 \rrbracket.$$

Algorithme 11 \mathcal{R}_0

1: Calculer la matrice \mathbb{B}

Algorithme 11 \mathcal{R}_1

1: **Pour** $i \leftarrow 1$ à n **faire**
2: Calculer $B_{i,i}$, connaissant les $i-1$ premières colonnes de \mathbb{B} .
3: Calculer la $i^{\text{ème}}$ colonne de \mathbb{B} .
4: **Fin Pour**

Algorithme 11 $\overline{\mathcal{R}}_1$

- 1: Pour $i \leftarrow 1$ à n faire
- 2: Calculer $B_{i,i}$, connaissant les $i - 1$ premières colonnes de \mathbb{B} .
- 3: Calculer la $i^{\text{ème}}$ colonne de \mathbb{B} .
- 4: Fin Pour

Algorithme 11 $\overline{\mathcal{R}}_2$

- 1: Pour $i \leftarrow 1$ à n faire

- 2:
$$B_{i,i} \leftarrow \left(A_{i,i} - \sum_{j=1}^{i-1} |B_{i,j}|^2 \right)^{1/2}$$

- 3: Pour $j \leftarrow 1$ à $i - 1$ faire

- 4: $B_{j,i} \leftarrow 0$

- 5: Fin Pour

- 6: Pour $j \leftarrow i + 1$ à n faire

- 7:
$$B_{j,i} \leftarrow \frac{1}{B_{i,i}} \left(A_{j,i} - \sum_{k=1}^{i-1} B_{j,k} \overline{B_{i,k}} \right).$$

- 8: Fin Pour

- 9: Fin Pour

Algorithme 11 \mathcal{R}_2

1: Pour $i \leftarrow 1$ à n faire

$$2: B_{i,i} \leftarrow \left(A_{i,i} - \sum_{j=1}^{i-1} |B_{i,j}|^2 \right)^{1/2}$$

3: Pour $j \leftarrow 1$ à $i-1$ faire

$$4: B_{j,i} \leftarrow 0$$

5: Fin Pour

6: Pour $j \leftarrow i+1$ à n faire

$$7: B_{j,i} \leftarrow \frac{1}{B_{i,i}} \left(A_{j,i} - \sum_{k=1}^{i-1} B_{j,k} \overline{B_{i,k}} \right)$$

8: Fin Pour

9: Fin Pour

Algorithme 11 \mathcal{R}_3

1: Pour $i \leftarrow 1$ à n faire

$$2: S_1 \leftarrow \sum_{j=1}^{i-1} |B_{i,j}|^2$$

$$3: B_{i,i} \leftarrow (A_{i,i} - S_1)^{1/2}$$

4: Pour $j \leftarrow 1$ à $i-1$ faire

$$5: B_{j,i} \leftarrow 0$$

6: Fin Pour

7: Pour $j \leftarrow i+1$ à n faire

$$8: S_2 \leftarrow \sum_{k=1}^{i-1} B_{j,k} \overline{B_{i,k}}$$

$$9: B_{j,i} \leftarrow \frac{1}{B_{i,i}} (A_{j,i} - S_2)$$

10: Fin Pour

11: Fin Pour

Algorithme 11 \mathcal{R}_3

```

1: Pour  $i \leftarrow 1$  à  $n$  faire
2:    $S_1 \leftarrow \sum_{j=1}^{i-1} |B_{i,j}|^2$ 
3:    $B_{i,i} \leftarrow (A_{i,i} - S_1)^{1/2}$ 
4:   Pour  $j \leftarrow 1$  à  $i-1$  faire
5:      $B_{j,i} \leftarrow 0$ 
6:   Fin Pour
7:   Pour  $j \leftarrow i+1$  à  $n$  faire
8:      $S_2 \leftarrow \sum_{k=1}^{i-1} B_{j,k} \overline{B_{i,k}}$ 
9:      $B_{j,i} \leftarrow \frac{1}{B_{i,i}} (A_{j,i} - S_2)$ 
10:  Fin Pour
11: Fin Pour
  
```

Algorithme 11 \mathcal{R}_4

```

1: Pour  $i \leftarrow 1$  à  $n$  faire
2:    $S_1 \leftarrow 0$ 
3:   Pour  $j \leftarrow 1$  à  $i-1$  faire
4:      $S_1 \leftarrow S_1 + |B_{i,j}|^2$ 
5:   Fin Pour
6:    $B_{i,i} \leftarrow (A_{i,i} - S_1)^{1/2}$ 
7:   Pour  $j \leftarrow 1$  à  $i-1$  faire
8:      $B_{j,i} \leftarrow 0$ 
9:   Fin Pour
10:  Pour  $j \leftarrow i+1$  à  $n$  faire
11:     $S_2 \leftarrow 0$ 
12:    Pour  $k \leftarrow 1$  à  $i-1$  faire
13:       $S_2 \leftarrow S_2 + B_{j,k} \overline{B_{i,k}}$ 
14:    Fin Pour
15:     $B_{j,i} \leftarrow \frac{1}{B_{i,i}} (A_{j,i} - S_2)$ 
16:  Fin Pour
17: Fin Pour
  
```

Algorithme 11 Fonction **CHOLESKY** permettant de calculer la matrice \mathbb{B} , dite matrice de factorisation positive de Cholesky associée à la matrice \mathbb{A} , telle que

$$\mathbb{A} = \mathbb{B}\mathbb{B}^*.$$

Données : \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{C})$ hermitienne définie positive.

Résultat : \mathbb{B} : matrice de $\mathcal{M}_n(\mathbb{C})$ triangulaire inférieure
avec $B(i, i) > 0, \forall i \in \llbracket 1, n \rrbracket$

```
1: Fonction B ← CHOLESKY ( A )
2:   Pour i ← 1 à n faire
3:     S1 ← 0
4:     Pour j ← 1 à i - 1 faire
5:       S1 ← S1 + |B(i, j)|2
6:     Fin Pour
7:     B(i, i) ← SQRT(A(i, i) - S1)
8:     Pour j ← 1 à i - 1 faire
9:       B(j, i) ← 0
10:    Fin Pour
11:    Pour j ← i + 1 à n faire
12:      S2 ← 0
13:      Pour k ← 1 à i - 1 faire
14:        S2 ← S2 + B(j, k) *  $\overline{B(i, k)}$ 
15:      Fin Pour
16:      B(j, i) ← (A(j, i) - S2)/B(i, i).
17:    Fin Pour
18:  Fin Pour
19: Fin Fonction
```



Exercice

Proposer une méthode permettant de tester la fonction `CHOLESKY` .

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

♥ Definition: Matrice élémentaire de Householder

Soit $\mathbf{u} \in \mathbb{C}^n$ tel que $\|\mathbf{u}\|_2 = 1$. On appelle **matrice élémentaire de Householder** la matrice $\mathbb{H}(\mathbf{u}) \in \mathcal{M}_n(\mathbb{C})$ définie par

$$\mathbb{H}(\mathbf{u}) = \mathbb{I} - 2\mathbf{u}\mathbf{u}^*. \quad (13)$$

 Propriété:



Toute matrice élémentaire de Householder est hermitienne et unitaire.



Propriété:



Soient $\mathbf{x} \in \mathbb{K}^n$ et $\mathbf{u} \in \mathbb{K}^n$, $\|\mathbf{u}\|_2 = 1$. On note $\mathbf{x}_{\parallel} = \text{proj}_{\mathbf{u}}(\mathbf{x}) \stackrel{\text{def}}{=} \langle \mathbf{u}, \mathbf{x} \rangle \mathbf{u}$ et $\mathbf{x}_{\perp} = \mathbf{x} - \mathbf{x}_{\parallel}$. On a alors

$$\mathbb{H}(\mathbf{u})(\mathbf{x}_{\perp} + \mathbf{x}_{\parallel}) = \mathbf{x}_{\perp} - \mathbf{x}_{\parallel}. \quad (14)$$

et

$$\mathbb{H}(\mathbf{u})\mathbf{x} = \mathbf{x}, \quad \text{si } \langle \mathbf{x}, \mathbf{u} \rangle = 0. \quad (15)$$



Théorème:



Soient \mathbf{a}, \mathbf{b} deux vecteurs non colinéaires de \mathbb{C}^n avec $\|\mathbf{b}\|_2 = 1$. Soit $\alpha \in \mathbb{C}$ tel que $|\alpha| = \|\mathbf{a}\|_2$ et $\arg \alpha = -\arg \langle \mathbf{a}, \mathbf{b} \rangle [\pi]$. On a alors

$$\mathbb{H}\left(\frac{\mathbf{a} - \alpha \mathbf{b}}{\|\mathbf{a} - \alpha \mathbf{b}\|_2}\right) \mathbf{a} = \alpha \mathbf{b}. \quad (16)$$



Exercice:



Soient \mathbf{a} et \mathbf{b} deux vecteurs non nuls et non colinéaires de \mathbb{C}^n avec $\|\mathbf{b}\|_2 = 1$.

Q. 1

Ecrire la fonction algorithmique **HOUSEHOLDER** permettant de retourner une matrice de Householder \mathbb{H} et $\alpha \in \mathbb{C}$ tels que $\mathbb{H}(\mathbf{u})\mathbf{a} = \alpha\mathbf{b}$. Le choix du α est fait par le paramètre δ (0 ou 1) de telle sorte que $\arg \alpha = -\arg(\langle \mathbf{a}, \mathbf{b} \rangle) + \delta\pi$ avec $|\alpha| = \|\mathbf{a}\|_2$. Des fonctions comme **DOT**(\mathbf{a}, \mathbf{b}) (produit scalaire de deux vecteurs), **NORM**(\mathbf{a}) (norme d'un vecteur), **ARG**(z) (argument d'un nombre complexe), **MATPROD**(\mathbb{A}, \mathbb{B}) (produit de deux matrices), **CTRANSPOSE**(\mathbb{A}) (adjoint d'une matrice), ... pourront être utilisées

Q. 2

Proposer un programme permettant de tester cette fonction. On pourra utiliser la fonction **VECRAND**(n) retournant un vecteur aléatoire de \mathbb{C}^n les parties réelles et imaginaires de chacune de ses



Corollaire 6.2:



Soit $\mathbf{a} \in \mathbb{C}^n$ avec $a_1 \neq 0$ et $\exists j \in \llbracket 2, n \rrbracket$ tel que $a_j \neq 0$. Soient $\theta = \arg a_1$ et

$$\mathbf{u}_{\pm} = \frac{\mathbf{a} \pm \|\mathbf{a}\|_2 e^{i\theta} \mathbf{e}_1}{\|\mathbf{a} \pm \|\mathbf{a}\|_2 e^{i\theta} \mathbf{e}_1\|}$$

Alors

$$\mathbb{H}(\mathbf{u}_{\pm})\mathbf{a} = \mp \|\mathbf{a}\|_2 e^{i\theta} \mathbf{e}_1 \quad (17)$$

où \mathbf{e}_1 désigne le premier vecteur de la base canonique de \mathbb{C}^n .



Théorème 7:



Soit $A \in \mathcal{M}_n(\mathbb{C})$ une matrice. Il existe une matrice unitaire $Q \in \mathcal{M}_n(\mathbb{C})$ produit d'au plus $n - 1$ matrices de Householder et une matrice triangulaire supérieure $R \in \mathcal{M}_n(\mathbb{C})$ telles que

$$A = QR. \quad (18)$$

Si A est réelle alors Q et R sont aussi réelles et l'on peut choisir Q de telle sorte que les coefficients diagonaux de R soient positifs. De plus, si A est inversible alors la factorisation est unique.



Exercice: Algorithmique



Q. 1

Ecrire une fonction **FACTQR** permettant de calculer la factorisation QR d'une matrice $A \in \mathcal{M}_n(\mathbb{C})$.

On pourra utiliser la fonction **HOUSEHOLDER** (voir Exercice 49, page 75).

Q. 2

Ecrire un programme permettant de tester cette fonction.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

♥ Definition

Une **norme** sur un espace vectoriel V est une application $\|\bullet\| : V \rightarrow \mathbb{R}^+$ qui vérifie les propriétés suivantes

- ◇ $\|\mathbf{v}\| = 0 \iff \mathbf{v} = 0$,
- ◇ $\|\alpha\mathbf{v}\| = |\alpha| \|\mathbf{v}\|$, $\forall \alpha \in \mathbb{K}$, $\forall \mathbf{v} \in V$,
- ◇ $\|\mathbf{u} + \mathbf{v}\| \leq \|\mathbf{u}\| + \|\mathbf{v}\|$, $\forall (\mathbf{u}, \mathbf{v}) \in V^2$ (inégalité triangulaire).

Une norme sur V est également appelée **norme vectorielle**. On appelle **espace vectoriel normé** un espace vectoriel muni d'une norme.



Proposition

Soit $\mathbf{v} \in \mathbb{K}^n$. Pour tout nombre réel $p \geq 1$, l'application $\|\bullet\|_p$ définie par

$$\|\mathbf{v}\|_p = \left(\sum_{i=1}^n |v_i|^p \right)^{1/p}$$

est une norme sur \mathbb{K}^n .

Normes usitées :

$$\|\mathbf{v}\|_1 = \sum_{i=1}^n |v_i|, \quad \|\mathbf{v}\|_2 = \left(\sum_{i=1}^n |v_i|^2 \right)^{1/2}, \quad \|\mathbf{v}\|_\infty = \max_{i \in [1, n]} |v_i|.$$



Lemme 8.1: Inégalité de Cauchy-Schwarz



$\forall \mathbf{x}, \mathbf{y} \in \mathbb{K}^n$

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\|_2 \|\mathbf{y}\|_2. \quad (19)$$

Cette inégalité s'appelle l'**inégalité de Cauchy-Schwarz**. On a égalité si et seulement si \mathbf{x} et \mathbf{y} sont colinéaires.



Lemme 8.2: Inégalité de Hölder



Pour $p > 1$ et $\frac{1}{p} + \frac{1}{q} = 1$, on a $\forall \mathbf{u}, \mathbf{v} \in \mathbb{K}^n$

$$\sum_{i=1}^n |u_i v_i| \leq \left(\sum_{i=1}^n |u_i|^p \right)^{1/p} \left(\sum_{i=1}^n |v_i|^q \right)^{1/q} = \|\mathbf{u}\|_p \|\mathbf{v}\|_q. \quad (20)$$

Cette inégalité s'appelle l'**inégalité de Hölder**.

♥ Definition 8.3

Deux **normes** $\|\bullet\|$ et $\|\bullet\|'$, définies sur un même espace vectoriel V , sont **équivalentes** s'il existe deux constantes C et C' telles que

$$\|\mathbf{v}\|' \leq C \|\mathbf{v}\| \quad \text{et} \quad \|\mathbf{v}\| \leq C' \|\mathbf{v}\|' \quad \text{pour tout } \mathbf{v} \in V. \quad (21)$$

Proposition

Sur un espace vectoriel de dimension finie toutes les normes sont équivalentes.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- **Normes matricielles**
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

♥ Definition 8.4

Une **norme matricielle** sur $\mathcal{M}_n(\mathbb{K})$ est une application $\|\bullet\| : \mathcal{M}_n(\mathbb{K}) \rightarrow \mathbb{R}^+$ vérifiant

- 1 $\|A\| = 0 \iff A = 0,$
- 2 $\|\alpha A\| = |\alpha| \|A\|, \forall \alpha \in \mathbb{K}, \forall A \in \mathcal{M}_n(\mathbb{K}),$
- 3 $\|A + B\| \leq \|A\| + \|B\|, \forall (A, B) \in \mathcal{M}_n(\mathbb{K})^2$ (inégalité triangulaire)
- 4 $\|AB\| \leq \|A\| \|B\|, \forall (A, B) \in \mathcal{M}_n(\mathbb{K})^2$

Peut-on étendre cette définition sur $\mathcal{M}_{m,n}(\mathbb{K})$?

 **Proposition:** 

Etant donné une norme vectorielle $\|\bullet\|$ sur \mathbb{C}^n , l'application $\|\bullet\|_s : \mathcal{M}_n(\mathbb{C}) \rightarrow \mathbb{R}^+$ définie par

$$\|\mathbb{A}\|_s = \sup_{\substack{\mathbf{v} \in \mathbb{C}^n \\ \mathbf{v} \neq 0}} \frac{\|\mathbb{A}\mathbf{v}\|}{\|\mathbf{v}\|} = \sup_{\substack{\mathbf{v} \in \mathbb{C}^n \\ \|\mathbf{v}\| \leq 1}} \|\mathbb{A}\mathbf{v}\| = \sup_{\substack{\mathbf{v} \in \mathbb{C}^n \\ \|\mathbf{v}\|=1}} \|\mathbb{A}\mathbf{v}\|, \quad (22)$$

est une norme matricielle, appelée **norme matricielle subordonnée** (à la norme vectorielle donnée).

De plus

$$\|\mathbb{A}\mathbf{v}\| \leq \|\mathbb{A}\|_s \|\mathbf{v}\| \quad \forall \mathbf{v} \in \mathbb{C}^n \quad (23)$$

et la norme $\|\mathbb{A}\|$ peut se définir aussi par

$$\|\mathbb{A}\|_s = \inf \{ \alpha \in \mathbb{R} : \|\mathbb{A}\mathbf{v}\| \leq \alpha \|\mathbf{v}\|, \forall \mathbf{v} \in \mathbb{C}^n \}. \quad (24)$$

Il existe au moins un vecteur $\mathbf{u} \in \mathbb{C}^n$ tel que

$$\mathbf{u} \neq 0 \quad \text{et} \quad \|\mathbb{A}\mathbf{u}\| = \|\mathbb{A}\|_s \|\mathbf{u}\|. \quad (25)$$

Enfin une norme subordonnée vérifie toujours

$$\|\mathbb{0}\|_s = 1 \quad (26)$$



Théorème 9:



Soit $A \in \mathcal{M}_n(\mathbb{K})$. On a

$$\|A\|_1 \stackrel{\text{def}}{=} \sup_{\substack{\mathbf{v} \in \mathbb{K}^n \\ \mathbf{v} \neq 0}} \frac{\|A\mathbf{v}\|_1}{\|\mathbf{v}\|_1} = \max_{j \in \llbracket 1, n \rrbracket} \sum_{i=1}^n |a_{ij}| \quad (27)$$

$$\|A\|_2 \stackrel{\text{def}}{=} \sup_{\substack{\mathbf{v} \in \mathbb{K}^n \\ \mathbf{v} \neq 0}} \frac{\|A\mathbf{v}\|_2}{\|\mathbf{v}\|_2} = \sqrt{\rho(A^*A)} = \sqrt{\rho(AA^*)} = \|A^*\|_2 \quad (28)$$

$$\|A\|_\infty \stackrel{\text{def}}{=} \sup_{\substack{\mathbf{v} \in \mathbb{K}^n \\ \mathbf{v} \neq 0}} \frac{\|A\mathbf{v}\|_\infty}{\|\mathbf{v}\|_\infty} = \max_{i \in \llbracket 1, n \rrbracket} \sum_{j=1}^n |a_{ij}| \quad (29)$$

La norme $\|\bullet\|_2$ est invariante par transformation unitaire :

$$UU^* = I \implies \|A\|_2 = \|AU\|_2 = \|UA\|_2 = \|U^*AU\|_2. \quad (30)$$



Corollaire 9.1

- 1 Si une matrice \mathbb{A} est hermitienne, ou symétrique (donc normale), on a $\|\mathbb{A}\|_2 = \rho(\mathbb{A})$.
- 2 Si une matrice \mathbb{A} est unitaire, ou orthogonale (donc normale), on a $\|\mathbb{A}\|_2 = 1$.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- **Suites de vecteurs et de matrices**

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

♥ Definition 9.2

Soit V un espace vectoriel muni d'une norme $\|\bullet\|$, on dit qu'une suite (\mathbf{v}_k) d'éléments de V **converge vers un élément** $\mathbf{v} \in V$, si

$$\lim_{k \rightarrow \infty} \|\mathbf{v}_k - \mathbf{v}\| = 0$$

et on écrit

$$\mathbf{v} = \lim_{k \rightarrow \infty} \mathbf{v}_k.$$



Théorème 10: admis

Soit \mathbb{B} une matrice carrée. Les conditions suivantes sont équivalentes :

- 1 $\lim_{k \rightarrow \infty} \mathbb{B}^k = 0,$
- 2 $\lim_{k \rightarrow \infty} \mathbb{B}^k \mathbf{v} = 0$ pour tout vecteur $\mathbf{v},$
- 3 $\rho(\mathbb{B}) < 1,$
- 4 $\|\mathbb{B}\| < 1$ pour au moins une norme matricielle subordonnée $\|\bullet\|.$



Théorème 11: admis

Soit \mathbb{B} une matrice carrée, et $\|\bullet\|$ une norme matricielle quelconque. Alors

$$\lim_{k \rightarrow \infty} \left\| \mathbb{B}^k \right\|^{1/k} = \rho(\mathbb{B}).$$

Soient $\mathbb{A} \in \mathcal{M}_n(\mathbb{K})$ inversible et $\mathbf{b} \in \mathbb{K}^n$.

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

De petites erreurs sur les données engendrent-elles de petites erreurs sur la solution?

Exemple de R.S. Wilson

Soient

$$\mathbb{A} = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix}, \quad \Delta\mathbb{A} = \begin{pmatrix} 0 & 0 & \frac{1}{10} & \frac{1}{5} \\ \frac{2}{25} & \frac{1}{25} & 0 & 0 \\ 0 & -\frac{1}{50} & -\frac{11}{100} & 0 \\ -\frac{1}{100} & -\frac{1}{100} & 0 & -\frac{1}{50} \end{pmatrix}$$

et $\mathbf{b}^t = (32, 23, 33, 31)$, $(\Delta\mathbf{b})^t = (\frac{1}{100}, -\frac{1}{100}, \frac{1}{100}, -\frac{1}{100})$. Des calculs exacts donnent

$$\mathbb{A}\mathbf{x} = \mathbf{b} \iff \mathbf{x}^t = (1, 1, 1, 1)$$

$$\mathbb{A}\mathbf{u} = (\mathbf{b} + \Delta\mathbf{b}) \iff \mathbf{u}^t = \left(\frac{91}{50}, -\frac{9}{25}, \frac{27}{20}, \frac{79}{100}\right) \\ \approx (1.8, -0.36, 1.3, 0.79)$$

$$(\mathbb{A} + \Delta\mathbb{A})\mathbf{v} = \mathbf{b} \iff \mathbf{v}^t = (-81, 137, -34, 22)$$

$$(\mathbb{A} + \Delta\mathbb{A})\mathbf{y} = (\mathbf{b} + \Delta\mathbf{b}) \iff \mathbf{y}^t = \left(-\frac{18283543}{461600}, \frac{31504261}{461600}, -\frac{3741501}{230800}, \frac{5235241}{461600}\right) \\ \approx (-39.61, 68.25, -16.21, 11.34)$$

Soient $A \in \mathcal{M}_n(\mathbb{K})$ inversible et $b \in \mathbb{K}^n$.

$$Ax = b$$

De petites erreurs sur les données engendrent-elles de petites erreurs sur la solution?

Non, pas forcément!

Le système linéaire précédent est **mal conditionné**.

On dit qu'un système linéaire est **bien conditionné** ou qu'il a un **bon conditionnement** si de petites perturbations des données n'entraînent qu'une variation *raisonnable* de la solution.

Est-il possible de "mesurer" le **conditionnement** d'une matrice?

♥ Definition 12.1

Soit $\|\cdot\|$ une norme matricielle subordonnée, le conditionnement d'une matrice régulière \mathbb{A} , associé à cette norme, est le nombre

$$\text{cond}(\mathbb{A}) = \|\mathbb{A}\| \|\mathbb{A}^{-1}\|.$$

Nous noterons $\text{cond}_p(\mathbb{A}) = \|\mathbb{A}\|_p \|\mathbb{A}^{-1}\|_p$.



Proposition:



Soit A une matrice régulière. On a les propriétés suivantes

- 1 $\forall \alpha \in \mathbb{K}^*$, $\text{cond}(\alpha A) = \text{cond}(A)$.
- 2 $\text{cond}_p(A) \geq 1$, $\forall p \in \llbracket 1, +\infty \rrbracket$.
- 3 $\text{cond}_2(A) = 1$ si et seulement si $A = \alpha Q$ avec $\alpha \in \mathbb{K}^*$ et Q matrice unitaire



Théorème 13:



Soit \mathbb{A} une matrice inversible. Soient \mathbf{x} et $\mathbf{x} + \Delta\mathbf{x}$ les solutions respectives de

$$\mathbb{A}\mathbf{x} = \mathbf{b} \quad \text{et} \quad \mathbb{A}(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b} + \Delta\mathbf{b}.$$

Supposons $\mathbf{b} \neq \mathbf{0}$, alors l'inégalité

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x}\|} \leq \text{cond}(\mathbb{A}) \frac{\|\Delta\mathbf{b}\|}{\|\mathbf{b}\|}$$

est satisfaite, et c'est la meilleure possible : pour une matrice \mathbb{A} donnée, on peut trouver des vecteurs $\mathbf{b} \neq \mathbf{0}$ et $\Delta\mathbf{b} \neq \mathbf{0}$ tels qu'elle devienne une égalité.



Théorème:



Soient \mathbb{A} et $\mathbb{A} + \Delta\mathbb{A}$ deux matrices inversibles. Soient \mathbf{x} et $\mathbf{x} + \Delta\mathbf{x}$ les solutions respectives de

$$\mathbb{A}\mathbf{x} = \mathbf{b} \text{ et } (\mathbb{A} + \Delta\mathbb{A})(\mathbf{x} + \Delta\mathbf{x}) = \mathbf{b}.$$

Supposons $\mathbf{b} \neq \mathbf{0}$, alors on a

$$\frac{\|\Delta\mathbf{x}\|}{\|\mathbf{x} + \Delta\mathbf{x}\|} \leq \text{cond}(\mathbb{A}) \frac{\|\Delta\mathbb{A}\|}{\|\mathbb{A}\|}.$$

Remarque 13.1

Une matrice est donc **bien conditionnée** si son conditionnement est proche de 1.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

Méthodes itératives pour la résolution du système linéaire

$$\mathbb{A}\mathbf{x} = \mathbf{b}.$$

Trouver une **matrice d'itération** \mathbb{B} et d'un vecteur \mathbf{c} telles que

$$\mathbf{x}^{[k+1]} = \mathbb{B}\mathbf{x}^{[k]} + \mathbf{c}, \quad k \geq 0, \quad \mathbf{x}^{[0]} \text{ arbitraire}$$

vérifie

$$\lim_{k \rightarrow \infty} \mathbf{x}^{[k]} = \tilde{\mathbf{x}} \quad \text{avec} \quad \tilde{\mathbf{x}} = \mathbb{A}^{-1}\mathbf{b}$$

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- **Notations**
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

Soit $A \in \mathcal{M}_n(\mathbb{K})$ une matrice régulière, avec $\forall i \in \llbracket 1, n \rrbracket, A_{i,i} \neq 0$

$$\begin{aligned}
 A &= \begin{pmatrix} A_{1,1} & 0 & \cdots & 0 \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & A_{n,n} \end{pmatrix} + \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ A_{2,1} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \vdots \\ A_{n,1} & \cdots & A_{n,n-1} & 0 \end{pmatrix} + \begin{pmatrix} 0 & A_{1,2} & \cdots & A_{1,n} \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & A_{n-1,n} \\ 0 & \cdots & \cdots & 0 \end{pmatrix} \\
 &= D - E - F = \begin{pmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{pmatrix}
 \end{aligned}$$

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

$$\mathbf{Ax} = \mathbf{b} \iff \forall i, b_i = \sum_{j=1}^n A_{ij}x_j = \sum_{j=1}^{i-1} A_{ij}x_j + A_{i,i}x_i + \sum_{j=i+1}^n A_{ij}x_j$$

La méthode itérative de **Jacobi** :

$$b_i = \sum_{j=1}^{i-1} A_{ij}x_j^{[k]} + A_{i,i}x_i^{[k+1]} + \sum_{j=i+1}^n A_{ij}x_j^{[k]} \quad \forall i \in \llbracket 1, n \rrbracket$$

ou encore

$$x_i^{[k+1]} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n A_{ij}x_j^{[k]} \right) \quad \forall i \in \llbracket 1, n \rrbracket$$

$$\mathbb{A}\mathbf{x} = \mathbf{b} \iff \forall i, b_i = \sum_{j=1}^n A_{i,j}x_j = \sum_{j=1}^{i-1} A_{i,j}x_j + A_{i,i}x_i + \sum_{j=i+1}^n A_{i,j}x_j$$

La méthode itérative de **Jacobi** :

$$b_i = \sum_{j=1}^{i-1} A_{i,j}x_j^{[k]} + A_{i,i}x_i^{[k+1]} + \sum_{j=i+1}^n A_{i,j}x_j^{[k]} \quad \forall i \in \llbracket 1, n \rrbracket$$

$$\mathbf{b} = \mathbb{D}\mathbf{x}^{[k+1]} - \mathbb{E}\mathbf{x}^{[k]} - \mathbb{F}\mathbf{x}^{[k]}$$

ou encore

$$x_i^{[k+1]} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n A_{ij}x_j^{[k]} \right) \quad \forall i \in \llbracket 1, n \rrbracket$$

$$\mathbf{x}^{[k+1]} = \mathbb{D}^{-1}(\mathbb{E} + \mathbb{F})\mathbf{x}^{[k]} + \mathbb{D}^{-1}\mathbf{b}$$

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- **Méthode de Gauss-Seidel**
- Méthode de relaxation
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

$$\mathbf{Ax} = \mathbf{b} \iff \forall i, b_i = \sum_{j=1}^n A_{ij}x_j = \sum_{j=1}^{i-1} A_{ij}x_j + A_{i,i}x_i + \sum_{j=i+1}^n A_{ij}x_j$$

La méthode itérative de **Gauss-Seidel** :

$$b_i = A_{i,i}x_i^{[k+1]} + \sum_{j=1}^{i-1} A_{ij}x_j^{[k+1]} + \sum_{j=i+1}^n A_{ij}x_j^{[k]} \quad \forall i \in \llbracket 1, n \rrbracket$$

ou encore

$$x_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{[k+1]} - \sum_{j=i+1}^n A_{ij}x_j^{[k]} \right) \quad \forall i \in \llbracket 1, n \rrbracket$$

$$\mathbb{A}\mathbf{x} = \mathbf{b} \iff \forall i, b_i = \sum_{j=1}^n A_{i,j}x_j = \sum_{j=1}^{i-1} A_{i,j}x_j + A_{i,i}x_i + \sum_{j=i+1}^n A_{i,j}x_j$$

La méthode itérative de **Gauss-Seidel** :

$$b_i = A_{i,i}x_i^{[k+1]} + \sum_{j=1}^{i-1} A_{i,j}x_j^{[k+1]} + \sum_{j=i+1}^n A_{i,j}x_j^{[k]} \quad \forall i \in \llbracket 1, n \rrbracket$$

$$\mathbf{b} = \mathbb{D}\mathbf{x}^{[k+1]} - \mathbb{E}\mathbf{x}^{[k+1]} - \mathbb{F}\mathbf{x}^{[k]}$$

ou encore

$$x_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{[k+1]} - \sum_{j=i+1}^n A_{ij}x_j^{[k]} \right) \quad \forall i \in \llbracket 1, n \rrbracket$$

$$\mathbf{x}^{[k+1]} = (\mathbb{D} - \mathbb{E})^{-1}\mathbb{F}\mathbf{x}^{[k]} + (\mathbb{D} - \mathbb{E})^{-1}\mathbf{b}$$

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- **Méthode de relaxation**
- Etude de la convergence
- Algorithmes
 - Principe de base
 - Méthode de Jacobi

Soit $w \in \mathbb{R}^*$.

$$x_i^{[k+1]} = w\hat{x}_i^{[k+1]} + (1-w)x_i^{[k]}$$

où $\hat{x}_i^{[k+1]}$ est obtenu à partir de l'une des deux méthodes précédentes.
Avec la méthode de Gauss-Seidel : méthode S.O.R. (successive over relaxation)

$$x_i^{[k+1]} = \frac{w}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{[k+1]} - \sum_{j=i+1}^n A_{ij}x_j^{[k]} \right) + (1-w)x_i^{[k]} \quad \forall i \in \llbracket 1, n \rrbracket$$



Exercice 14.1:



Déterminer la matrice d'itération \mathbb{B} et le vecteur \mathbf{c} tels que

$$\mathbf{x}^{[k+1]} = \mathbb{B}\mathbf{x}^{[k]} + \mathbf{c}$$

en fonction de \mathbb{D} , \mathbb{E} , \mathbb{F} , et \mathbf{b} .



Proposition:



On pose $\mathbb{L} = \mathbb{D}^{-1}\mathbb{E}$ et $\mathbb{U} = \mathbb{D}^{-1}\mathbb{F}$.

La matrice d'itération de la méthode de Jacobi, notée \mathbb{J} , est donnée par

$$\mathbb{J} = \mathbb{D}^{-1}(\mathbb{E} + \mathbb{F}) = \mathbb{L} + \mathbb{U}, \quad (31)$$

La matrice d'itération de la méthode S.O.R., notée \mathcal{L}_w , est donnée par

$$\mathcal{L}_w = \left(\frac{\mathbb{D}}{w} - \mathbb{E} \right)^{-1} \left(\frac{1-w}{w} \mathbb{D} + \mathbb{F} \right) = (\mathbb{I} - w\mathbb{L})^{-1} ((1-w)\mathbb{I} + w\mathbb{U}). \quad (32)$$

et elle vérifie

$$\rho(\mathcal{L}_w) \geq |w - 1|. \quad (33)$$

La matrice d'itération de Gauss-Seidel est \mathcal{L}_1 et elle correspond à

$$\mathcal{L}_1 = (\mathbb{D} - \mathbb{E})^{-1}\mathbb{F} = (\mathbb{I} - \mathbb{L})^{-1}\mathbb{U}. \quad (34)$$

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- **Etude de la convergence**
- Algorithmes
 - Principe de base
 - Méthode de Jacobi



Théorème:



Soit \mathbb{A} une matrice régulière décomposée sous la forme $\mathbb{A} = \mathbb{M} - \mathbb{N}$ avec \mathbb{M} régulière. On pose

$$\mathbb{B} = \mathbb{M}^{-1}\mathbb{N} \quad \text{et} \quad \mathbf{c} = \mathbb{M}^{-1}\mathbf{b}.$$

Alors la suite définie par

$$\mathbf{x}^{[0]} \in \mathbb{K}^n \quad \text{et} \quad \mathbf{x}^{[k+1]} = \mathbb{B}\mathbf{x}^{[k]} + \mathbf{c}$$

converge vers $\bar{\mathbf{x}} = \mathbb{A}^{-1}\mathbf{b}$ quelque soit $\mathbf{x}^{[0]}$ si et seulement si $\rho(\mathbb{B}) < 1$.



Corollaire:



Soit A une matrice vérifiant $A_{i,i} \neq 0 \forall i$. Une condition nécessaire de convergence pour la méthode S.O.R. est que $0 < w < 2$.



Théorème: voir Lascaux-Théodor, vol.2, Théorème 19 et 20, pages 346 à 349

Soit A une matrice à diagonale strictement dominante ou une matrice inversible à diagonale fortement dominante alors

- la méthode de Jacobi est convergente,
- si $w \in]0, 1]$ la méthode de Relaxation est convergente.



Théorème

Soit A une matrice hermitienne inversible en décomposée en $A = M - N$ où M est inversible. Soit $B = I - M^{-1}A$, la matrice de l'itération. Supposons que $M^* + N$ (qui est hermitienne) soit définie positive. Alors $\rho(B) < 1$ si et seulement si A est définie positive.



Théorème

Soit A une matrice hermitienne définie positive, alors la méthode de relaxation converge si et seulement si $w \in]0, 2[$.

Plan

1 Méthodes directes

- Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
- Exercices et résultats préliminaires
- Méthode de Gauss-Jordan
 - Ecriture algébrique
- Factorisation LU
 - Résultats théoriques
 - Utilisation pratique
- Factorisation LDL*
- Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation

2 Normes vectorielles et normes matricielles

- Normes vectorielles
- Normes matricielles
- Suites de vecteurs et de matrices

3 Conditionnement d'un système linéaire

4 Méthodes itératives

- Principe
- Notations
- Méthode de Jacobi
- Méthode de Gauss-Seidel
- Méthode de relaxation
- Etude de la convergence

Algorithme

- Principe de base
- Méthode de Jacobi

Principe de base

Résoudre :

$$\mathbb{A}\mathbf{x} = \mathbf{b}$$

Méthodes itératives :

$$\mathbf{x}^{[0]} \in \mathbb{K}^n \text{ et } \mathbf{x}^{[k+1]} = \mathbb{B}\mathbf{x}^{[k]} + \mathbf{c}$$

Algorithme :

$\mathbf{x}^{[0]}$ donné

Pour $k = 0, 1, \dots$ faire

$\mathbf{x}^{[k+1]} \leftarrow \mathbb{B}\mathbf{x}^{[k]} + \mathbf{c}$

Fin Pour

Critère d'arrêt? Stockage de tous les $\mathbf{x}^{[k]}$?

La convergence de ces méthodes n'est pas assurées et si il y a convergence le nombre d'itération nécessaire n'est (à priori) pas connu.

⇒ boucle **Tantque**

Critères d'arrêt :

- nombre maximum d'itérations
- $\varepsilon > 0$ permet l'arrêt des calculs si $\mathbf{x}^{[k]}$ suffisamment proche de $\bar{\mathbf{x}} = \mathbb{A}^{-1}\mathbf{b}$

Comment choisir le critère d'arrêt pour la convergence?

La convergence de ces méthodes n'est pas assurées et si il y a convergence le nombre d'itération nécessaire n'est (à priori) pas connu.

⇒ boucle **Tantque**

Critères d'arrêt :

- nombre maximum d'itérations
- $\varepsilon > 0$ permet l'arrêt des calculs si $\mathbf{x}^{[k]}$ suffisamment proche de $\bar{\mathbf{x}} = \mathbb{A}^{-1}\mathbf{b}$

Comment choisir le critère d'arrêt pour la convergence?

Exemple de critère d'arrêt pour la convergence :

Soit $\mathbf{r}^{[k]} = \mathbf{b} - \mathbb{A}\mathbf{x}^{[k]}$ le résidu.

$$\frac{\|\mathbf{r}^{[k]}\|}{\|\mathbf{b}\|} \leq \varepsilon$$

La convergence de ces méthodes n'est pas assurées et si il y a convergence le nombre d'itération nécessaire n'est (à priori) pas connu.

⇒ boucle **Tantque**

Critères d'arrêt :

- nombre maximum d'itérations
- $\varepsilon > 0$ permet l'arrêt des calculs si $\mathbf{x}^{[k]}$ suffisamment proche de $\bar{\mathbf{x}} = \mathbb{A}^{-1}\mathbf{b}$

Comment choisir le critère d'arrêt pour la convergence?

Exemple de critère d'arrêt pour la convergence :

Soit $\mathbf{r}^{[k]} = \mathbf{b} - \mathbb{A}\mathbf{x}^{[k]}$ le résidu.

$$\frac{\|\mathbf{r}^{[k]}\|}{\|\mathbf{b}\|} \leq \varepsilon$$

Car dans ce cas, on a avec $\mathbf{e}^{[k]} = \bar{\mathbf{x}} - \mathbf{x}^{[k]}$

$$\frac{\|\mathbf{e}^{[k]}\|}{\|\bar{\mathbf{x}}\|} \leq \varepsilon \text{cond}(\mathbb{A})$$

Algorithme 12 Méthode itérative pour la résolution d'un système linéaire $\mathbb{A}\mathbf{x} = \mathbf{b}$

Données :

- \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$,
- \mathbf{b} : vecteur de \mathbb{K}^n ,
- \mathbf{x}^0 : vecteur initial de \mathbb{K}^n ,
- ε : la tolérance, $\varepsilon \in \mathbb{R}^+$,
- kmax : nombre maximum d'itérations, kmax $\in \mathbb{N}^*$

Résultat :

\mathbf{x}^{tol} : un vecteur de \mathbb{K}^n si convergence, sinon \emptyset

- 1: $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$
- 2: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x}$,
- 3: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
- 4: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq \text{kmax}$ **faire**
- 5: $k \leftarrow k + 1$
- 6: $\mathbf{p} \leftarrow \mathbf{x}$ ▷ \mathbf{p} contient le vecteur précédent
- 7: $\mathbf{x} \leftarrow$ calcul de l'itérée suivante en fonction de $\mathbf{p}, \mathbb{A}, \mathbf{b}, \dots$
- 8: $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x}$,
- 9: **Fin Tantque**
- 10: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors** ▷ Convergence
- 11: $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$
- 12: **Fin Si**

Pour Jacobi , la suite des itérées est définie par

$$x_i^{[k+1]} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n A_{ij} x_j^{[k]} \right), \quad \forall i \in \llbracket 1, n \rrbracket.$$

Algorithme 13 \mathcal{R}_0

- 1: $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$
- 2: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$
- 3: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
- 4: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\max}$ **faire**
- 5: $k \leftarrow k + 1$
- 6: $\mathbf{p} \leftarrow \mathbf{x}$
- 7:
- 8: $\mathbf{x} \leftarrow$ calcul par Jacobi
- 9: $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$
- 10: **Fin Tantque**
- 11: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors** \triangleright Convergence
- 12: $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$
- 13: **Fin Si**

Algorithme 13 \mathcal{R}_1

- 1: $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$
- 2: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$
- 3: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
- 4: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\max}$ **faire**
- 5: $k \leftarrow k + 1$
- 6: $\mathbf{p} \leftarrow \mathbf{x}$
- 7: **Pour** $i \leftarrow 1$ à n **faire**
- 8: $x_i \leftarrow \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n A_{ij} p_j \right)$
- 9: **Fin Pour**
- 10: $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$
- 11: **Fin Tantque**
- 12: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors** \triangleright Convergence
- 13: $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$
- 14: **Fin Si**

Algorithme 13 \mathcal{R}_1

```

1:  $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$ 
2:  $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$ 
3:  $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$ 
4: Tantque  $\|\mathbf{r}\| > \text{tol}$  et  $k \leq k_{\text{max}}$  faire
5:    $k \leftarrow k + 1$ 
6:    $\mathbf{p} \leftarrow \mathbf{x}$ 
7:   Pour  $i \leftarrow 1$  à  $n$  faire
8:
9:     
$$x_i \leftarrow \frac{1}{A_{ii}} \left( b_i - \sum_{j=1, j \neq i}^n A_{ij} p_j \right)$$

10:  Fin Pour
11:   $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$ 
12: Fin Tantque
13: Si  $\|\mathbf{r}\| \leq \text{tol}$  alors
14:    $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$ 
15: Fin Si

```

Algorithme 13 \mathcal{R}_2

```

1:  $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$ 
2:  $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$ 
3:  $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$ 
4: Tantque  $\|\mathbf{r}\| > \text{tol}$  et  $k \leq k_{\text{max}}$  faire
5:    $k \leftarrow k + 1$ 
6:    $\mathbf{p} \leftarrow \mathbf{x}$ 
7:   Pour  $i \leftarrow 1$  à  $n$  faire
8:
9:     Pour  $j \leftarrow 1$  à  $n$  ( $j \neq i$ ) faire
10:       $S \leftarrow S + A_{ij} p_j$ 
11:    Fin Pour
12:     $x_i \leftarrow \frac{1}{A_{ii}} (b_i - S)$ 
13:  Fin Pour
14:   $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{A}\mathbf{x},$ 
15: Fin Tantque
16: Si  $\|\mathbf{r}\| \leq \text{tol}$  alors
17:    $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$ 
18: Fin Si

```

Algorithme 13 Méthode itérative de Jacobi pour la résolution d'un système linéaire $\mathbb{A}\mathbf{x} = \mathbf{b}$

Données :

- \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$,
- \mathbf{b} : vecteur de \mathbb{K}^n ,
- \mathbf{x}^0 : vecteur initial de \mathbb{K}^n ,
- ε : la tolérance, $\varepsilon \in \mathbb{R}^+$,
- kmax : nombre maximum d'itérations, kmax $\in \mathbb{N}^*$

Résultat :

\mathbf{X} : un vecteur de \mathbb{K}^n

```
1: Fonction  $\mathbf{X} \leftarrow \text{RSLJACOBI}$  (  $\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, \text{kmax}$  )
2:    $k \leftarrow 0, \mathbf{X} \leftarrow \emptyset$ 
3:    $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x}$ ,
4:   tol  $\leftarrow \varepsilon(\|\mathbf{b}\| + 1)$ 
5:   Tantque  $\|\mathbf{r}\| > \text{tol}$  et  $k \leq \text{kmax}$  faire
6:      $k \leftarrow k + 1$ 
7:      $\mathbf{p} \leftarrow \mathbf{x}$ 
8:     Pour  $i \leftarrow 1$  à  $n$  faire
9:        $S \leftarrow 0$ 
10:      Pour  $j \leftarrow 1$  à  $n$  ( $j \neq i$ ) faire
11:         $S \leftarrow S + A(i, j) * p(j)$ 
12:      Fin Pour
13:       $x(i) \leftarrow (b(i) - S)/A(i, i)$ 
14:    Fin Pour
15:     $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x}$ ,
16:  Fin Tantque
17:  Si  $\|\mathbf{r}\| \leq \text{tol}$  alors
18:     $\mathbf{X} \leftarrow \mathbf{x}$ 
19:  Fin Si
20: Fin Fonction
```

Pour Gauss-Seidel, la suite des itérées est définie par

$$x_i^{(k+1)} = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j^{[k+1]} - \sum_{j=i+1}^n A_{ij}x_j^{[k]} \right) \quad \forall i \in \llbracket 1, n \rrbracket$$

Algorithme 14 $\overline{\mathcal{R}_0}$

- 1: $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$
- 2: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x},$
- 3: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
- 4: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\text{max}}$ **faire**
- 5: $k \leftarrow k + 1$
- 6: $\mathbf{p} \leftarrow \mathbf{x}$
- 7:
- 8: $\mathbf{x} \leftarrow$ calcul par Gauss-Seidel
- 9: $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x},$
- 10: **Fin Tantque**
- 11: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors**
- 12: $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$
- 13: **Fin Si**

Algorithme 14 $\overline{\mathcal{R}_1}$

- 1: $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$
- 2: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x},$
- 3: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
- 4: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\text{max}}$ **faire**
- 5: $k \leftarrow k + 1$
- 6: $\mathbf{p} \leftarrow \mathbf{x}$
- 7: **Pour** $i \leftarrow 1$ à n **faire**
- 8: $x_i \leftarrow \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j - \sum_{j=i+1}^n A_{ij}p_j \right)$
- 9: **Fin Pour**
- 10: $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x},$
- 11: **Fin Tantque**
- 12: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors**
- 13: $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$
- 14: **Fin Si**

Algorithme 14 \mathcal{R}_1

1: $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$
 2: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \Delta \mathbf{x},$
 3: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
 4: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\text{max}}$ **faire**
 5: $k \leftarrow k + 1$
 6: $\mathbf{p} \leftarrow \mathbf{x}$
 7: **Pour** $i \leftarrow 1$ à n **faire**
 8:
 9:
$$x_i \leftarrow \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j - \sum_{j=i+1}^n A_{ij}p_j \right)$$

 10: **Fin Pour**
 11: $\mathbf{r} \leftarrow \mathbf{b} - \Delta \mathbf{x},$
 12: **Fin Tantque**
 13: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors**
 14: $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$
 15: **Fin Si**

Algorithme 14 \mathcal{R}_2

1: $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$
 2: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \Delta \mathbf{x},$
 3: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
 4: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\text{max}}$ **faire**
 5: $k \leftarrow k + 1$
 6: $\mathbf{p} \leftarrow \mathbf{x}$
 7: **Pour** $i \leftarrow 1$ à n **faire**
 8: $S \leftarrow 0$
 9: **Pour** $j \leftarrow 1$ à $i - 1$ **faire**
 10: $S \leftarrow S + A_{i,j}x_j$
 11: **Fin Pour**
 12: **Pour** $j \leftarrow i + 1$ à n **faire**
 13: $S \leftarrow S + A_{i,j}p_j$
 14: **Fin Pour**
 15: $x_i \leftarrow \frac{1}{A_{ii}} (b_i - S)$
 16: **Fin Pour**
 17: $\mathbf{r} \leftarrow \mathbf{b} - \Delta \mathbf{x},$
 18: **Fin Tantque**
 19: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors**
 20: $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$
 21: **Fin Si**

Algorithme 14 Méthode itérative de Gauss-Seidel pour la résolution d'un système linéaire $\mathbb{A}\mathbf{x} = \mathbf{b}$ **Données :**

- \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$,
- \mathbf{b} : vecteur de \mathbb{K}^n ,
- \mathbf{x}^0 : vecteur initial de \mathbb{K}^n ,
- ε : la tolérance, $\varepsilon \in \mathbb{R}^+$,
- kmax : nombre maximum d'itérations, kmax $\in \mathbb{N}^*$

Résultat : \mathbf{X} : un vecteur de \mathbb{K}^n

```
1: Fonction  $\mathbf{X} \leftarrow \text{RSLGAUSSSEIDEL} (\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, \text{kmax})$ 
2:  $k \leftarrow 0, \mathbf{X} \leftarrow \emptyset$ 
3:  $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x}$ ,
4: tol  $\leftarrow \varepsilon(\|\mathbf{b}\| + 1)$ 
5: Tantque  $\|\mathbf{r}\| > \text{tol}$  et  $k \leq \text{kmax}$  faire
6:    $k \leftarrow k + 1$ 
7:    $\mathbf{p} \leftarrow \mathbf{x}$ 
8:   Pour  $i \leftarrow 1$  à  $n$  faire
9:      $\mathbf{S} \leftarrow 0$ 
10:    Pour  $j \leftarrow 1$  à  $i - 1$  faire
11:       $\mathbf{S} \leftarrow \mathbf{S} + A(i, j) * x(j)$ 
12:    Fin Pour
13:    Pour  $j \leftarrow i + 1$  à  $n$  faire
14:       $\mathbf{S} \leftarrow \mathbf{S} + A(i, j) * p(j)$ 
15:    Fin Pour
16:     $x(i) \leftarrow (b(i) - \mathbf{S}) / A(i, i)$ 
17:  Fin Pour
18:   $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x}$ ,
19: Fin Tantque
20: Si  $\|\mathbf{r}\| \leq \text{tol}$  alors
21:    $\mathbf{X} \leftarrow \mathbf{x}$ 
22: Fin Si
23: Fin Fonction
```

Fonction $X \leftarrow \text{RSLJACOBI} (\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, k_{\max})$

$k \leftarrow 0, X \leftarrow \emptyset$

$\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$

$\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$

Tantque $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\max}$ faire

$k \leftarrow k + 1$

$\mathbf{p} \leftarrow \mathbf{x}$

Pour $i \leftarrow 1$ à n faire

$S \leftarrow 0$

Pour $j \leftarrow 1$ à n ($i \neq j$) faire

$S \leftarrow S + A(i, j) * p(j)$

Fin Pour

$x(i) \leftarrow (b(i) - S) / A(i, i)$

Fin Pour

$\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$

Fin Tantque

Si $\|\mathbf{r}\| \leq \text{tol}$ alors

$X \leftarrow \mathbf{x}$

Fin Si

Fin Fonction

Fonction $X \leftarrow \text{RSLGAUSSSEIDEL} (\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, k_{\max})$

$\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$

Tantque $\|\mathbf{r}\| > \text{tol}$ et $k \leq k_{\max}$ faire

$k \leftarrow k + 1$

$\mathbf{p} \leftarrow \mathbf{x}$

Pour $i \leftarrow 1$ à n faire

$S \leftarrow 0$

Pour $j \leftarrow 1$ à $i - 1$ faire

$S \leftarrow S + A(i, j) * x(j)$

Fin Pour

Pour $j \leftarrow i + 1$ à n faire

$S \leftarrow S + A(i, j) * p(j)$

Fin Pour

$x(i) \leftarrow (b(i) - S) / A(i, i)$

Fin Pour

$\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$

Fin Tantque

Si $\|\mathbf{r}\| \leq \text{tol}$ alors

$X \leftarrow \mathbf{x}$

Fin Si

Fin Fonction

```

k ← 0. X ← ∅
tol ← ε(||b|| + 1)
k ← k + 1
p ← x
Pour i ← 1 à n faire
  S ← 0
  Pour i ← 1 à n (i ≠ i) faire
    S ← S + A(i,j) * p(j)
  Fin Pour
  x(i) ← (b(i) - S)/A(i,i)
Fin Pour
r ← b - A * x,
Fin Tantque
Si ||r|| ≤ tol alors
  X ← x
Fin Si
Fin Fonction

```

Fonction **X** ← RSLGAUSSSEIDEL (**A**, **b**, **x**⁰, ε, kmax)

```

k ← 0, X ← ∅
x ← x0, r ← b - A * x,
tol ← ε(||b|| + 1)
Tantque ||r|| > tol et k ≤ kmax faire
  k ← k + 1

```

```

Fin Pour
r ← b - A * x,
Fin Tantque
Si ||r|| ≤ tol alors
  X ← x
Fin Si
Fin Fonction

```

Même ossature puisque toutes deux basées sur l'Algorithme générique
 Peut-on simplifier, clarifier et raccourcir les codes?

```

k ← 0, X ← ∅
x ← x0, r ← b - A * x,
tol ← ε(||b|| + 1)
Tantque ||r|| > tol et k ≤ kmax faire
  k ← k + 1

```

```

Pour i ← 1 à n faire
  S ← 0
  Pour j ← 1 à n (j ≠ i) faire
    S ← S + A(i, j) * p(j)
  Fin Pour
  x(i) ← (b(i) - S)/A(i, i)

```

```

Fin Pour
r ← b - A * x,

```

```

Fin Tantque
Si ||r|| ≤ tol alors
  X ← x
Fin Si
Fin Fonction

```

Algorithme 15 itération de Jacobi : calcul de \mathbf{x} tel que

$$x_i = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n A_{ij} y_j \right), \quad \forall i \in \llbracket 1, n \rrbracket.$$

Données :

A : matrice de $\mathcal{M}_n(\mathbb{K})$,
b : vecteur de \mathbb{K}^n ,
y : vecteur de \mathbb{K}^n ,

Résultat :

x : un vecteur de \mathbb{K}^n

```

1: Fonction x ← ITERJACOBI ( A, b, y )
2:   Pour i ← 1 à n faire
3:     S ← 0
4:     Pour j ← 1 à n (j ≠ i) faire
5:       S ← S + A(i, j) * y(j)
6:     Fin Pour
7:     x(i) ← (b(i) - S)/A(i, i)
8:   Fin Pour
9: Fin Fonction

```

Fonction $X \leftarrow \text{RSLJACOBI2} (\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, \text{kmax})$

$k \leftarrow 0, X \leftarrow \emptyset$

$\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$

$\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$

Tantque $\|\mathbf{r}\| > \text{tol}$ et $k \leq \text{kmax}$ faire

$k \leftarrow k + 1$

$\mathbf{p} \leftarrow \mathbf{x}$

$\mathbf{x} \leftarrow \text{ITERJACOBI}(\mathbb{A}, \mathbf{b}, \mathbf{p})$

$\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$

Fin Tantque

Si $\|\mathbf{r}\| \leq \text{tol}$ alors

$X \leftarrow \mathbf{x}$

Fin Si

Fin Fonction

Algorithme 16 itération de Jacobi : calcul de \mathbf{x} tel que

$$x_i = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1, j \neq i}^n A_{ij} y_j \right), \quad \forall i \in \llbracket 1, n \rrbracket.$$

Données :

A : matrice de $\mathcal{M}_n(\mathbb{K})$,

b : vecteur de \mathbb{K}^n ,

y : vecteur de \mathbb{K}^n ,

Résultat :

x : un vecteur de \mathbb{K}^n

1: Fonction $x \leftarrow \text{ITERJACOBI} (\mathbb{A}, \mathbf{b}, \mathbf{y})$

2: Pour $i \leftarrow 1$ à n faire

3: $S \leftarrow 0$

4: Pour $j \leftarrow 1$ à n ($j \neq i$) faire

5: $S \leftarrow S + A(i, j) * y(j)$

6: Fin Pour

7: $x(i) \leftarrow (b(i) - S)/A(i, i)$

8: Fin Pour

9: Fin Fonction

Fonction $\mathbf{X} \leftarrow \text{RSLGAUSSSEIDEL2} (\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, \text{kmax})$

$k \leftarrow 0, \mathbf{X} \leftarrow \emptyset$

$\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$

$\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$

Tantque $\|\mathbf{r}\| > \text{tol}$ et $k \leq \text{kmax}$ faire

$k \leftarrow k + 1$

$\mathbf{p} \leftarrow \mathbf{x}$

$\mathbf{x} \leftarrow \text{ITERGAUSSSEIDEL}(\mathbb{A}, \mathbf{b}, \mathbf{p})$

$\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$

Fin Tantque

Si $\|\mathbf{r}\| \leq \text{tol}$ alors

$\mathbf{X} \leftarrow \mathbf{x}$

Fin Si

Fin Fonction

Algorithme 18 Itération de Gauss-Seidel : calcul de \mathbf{x}
tel que

$$x_i = \frac{1}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij}x_j - \sum_{j=i+1}^n A_{ij}y_j \right), \quad \forall i \in \llbracket 1, n \rrbracket.$$

Données :

\mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$,

\mathbf{b} : vecteur de \mathbb{K}^n ,

\mathbf{y} : vecteur de \mathbb{K}^n ,

Résultat :

\mathbf{x} : un vecteur de \mathbb{K}^n

1: Fonction $\mathbf{x} \leftarrow \text{ITERGAUSSSEIDEL} (\mathbb{A}, \mathbf{b}, \mathbf{y})$

2: Pour $i \leftarrow 1$ à n faire

3: $S \leftarrow 0$

4: Pour $j \leftarrow 1$ à $i - 1$ faire

5: $S \leftarrow S + A(i, j) * x(j)$

6: Fin Pour

7: Pour $j \leftarrow i + 1$ à n faire

8: $S \leftarrow S + A(i, j) * y(j)$

9: Fin Pour

10: $x(i) \leftarrow (b(i) - S)/A(i, i)$

11: Fin Pour

12: Fin Fonction

```

Fonction  $\mathbf{X} \leftarrow \text{RSLGAUSSSEIDEL2}$  (  $\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, k_{\max}$ 
)
   $k \leftarrow 0, \mathbf{X} \leftarrow \emptyset$ 
   $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$ 
   $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$ 
  Tantque  $\|\mathbf{r}\| > \text{tol}$  et  $k \leq k_{\max}$  faire
     $k \leftarrow k + 1$ 
     $\mathbf{p} \leftarrow \mathbf{x}$ 
     $\mathbf{x} \leftarrow \text{ITERGAUSSSEIDEL}(\mathbb{A}, \mathbf{b}, \mathbf{p})$ 
     $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$ 
  Fin Tantque
  Si  $\|\mathbf{r}\| \leq \text{tol}$  alors
     $\mathbf{X} \leftarrow \mathbf{x}$ 
  Fin Si
Fin Fonction

```

```

Fonction  $\mathbf{X} \leftarrow \text{RSLJACOBI2}$  (  $\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, k_{\max}$  )
   $k \leftarrow 0, \mathbf{X} \leftarrow \emptyset$ 
   $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$ 
   $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$ 
  Tantque  $\|\mathbf{r}\| > \text{tol}$  et  $k \leq k_{\max}$  faire
     $k \leftarrow k + 1$ 
     $\mathbf{p} \leftarrow \mathbf{x}$ 
     $\mathbf{x} \leftarrow \text{ITERJACOBI}(\mathbb{A}, \mathbf{b}, \mathbf{p})$ 
     $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A} * \mathbf{x},$ 
  Fin Tantque
  Si  $\|\mathbf{r}\| \leq \text{tol}$  alors
     $\mathbf{X} \leftarrow \mathbf{x}$ 
  Fin Si
Fin Fonction

```

Les deux codes sont fortement similaires!

Peut-on éviter les copier/coller et gagner encore en lisibilité?

Ecriture Algorithme générique sous forme d'une fonction et on ajoute aux paramètres d'entrées une fonction formelle **ITERFONC** calculant une itérée :

$$\mathbf{x} \leftarrow \text{ITERFONC}(\mathbb{A}, \mathbf{b}, \mathbf{y}).$$

Algorithme 18 Méthode itérative pour la résolution d'un système linéaire $\mathbb{A}\mathbf{x} = \mathbf{b}$

Données :

- \mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$,
- \mathbf{b} : vecteur de \mathbb{K}^n ,
- ITERFONC** : fonction de paramètres une matrice d'ordre n ,
et deux vecteurs de \mathbb{K}^n . retourne un vecteur de \mathbb{K}^n .
- \mathbf{x}^0 : vecteur initial de \mathbb{K}^n ,
- ε : la tolérance, $\varepsilon \in \mathbb{R}^+$,
- kmax : nombre maximum d'itérations, kmax $\in \mathbb{N}^*$

Résultat :

- \mathbf{x}^{toi} : un vecteur de \mathbb{K}^n si convergence, sinon \emptyset

- 1: **Fonction** $\mathbf{X} \leftarrow \text{RSLMETHITER}(\mathbb{A}, \mathbf{b}, \text{ITERFONC}, \mathbf{x}^0, \varepsilon, \text{kmax})$
- 2: $k \leftarrow 0, \mathbf{x}^{\text{toi}} \leftarrow \emptyset$
- 3: $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x}$,
- 4: $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$
- 5: **Tantque** $\|\mathbf{r}\| > \text{tol}$ et $k \leq \text{kmax}$ **faire**
- 6: $k \leftarrow k + 1$
- 7: $\mathbf{p} \leftarrow \mathbf{x}$
- 8: $\mathbf{x} \leftarrow \text{ITERFONC}(\mathbb{A}, \mathbf{b}, \mathbf{p})$
- 9: $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x}$,
- 10: **Fin Tantque**
- 11: **Si** $\|\mathbf{r}\| \leq \text{tol}$ **alors**
- 12: $\mathbf{x}^{\text{toi}} \leftarrow \mathbf{x}$
- 13: **Fin Si**
- 14: **Fin Fonction**

Fonction $X \leftarrow \text{RSLJACOBI3} (\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, \text{kmax})$
 $X \leftarrow \text{RSLMETHITER}(\mathbb{A}, \mathbf{b}, \text{ITERJACOBI}, \mathbf{x}^0, \varepsilon, \text{kmax})$
 Fin Fonction

Fonction $X \leftarrow \text{RSLGAUSSSEIDEL3} (\mathbb{A}, \mathbf{b}, \mathbf{x}^0, \varepsilon, \text{kmax})$
 $X \leftarrow \text{RSLMETHITER}(\mathbb{A}, \mathbf{b}, \text{ITERGAUSSSEIDEL}, \mathbf{x}^0, \varepsilon, \text{kmax})$
 Fin Fonction

Algorithme 18 Méthode itérative pour la résolution d'un système linéaire $\mathbb{A}\mathbf{x} = \mathbf{b}$

Données :

\mathbb{A} : matrice de $\mathcal{M}_n(\mathbb{K})$,
 \mathbf{b} : vecteur de \mathbb{K}^n ,
ITERFONC : fonction de paramètres une matrice d'ordre n ,
 et deux vecteurs de \mathbb{K}^n . retourne un vecteur de \mathbb{K}^n .
 \mathbf{x}^0 : vecteur initial de \mathbb{K}^n ,
 ε : la tolérance, $\varepsilon \in \mathbb{R}^+$,
 kmax : nombre maximum d'itérations, $\text{kmax} \in \mathbb{N}^*$

Résultat :

\mathbf{x}^{tol} : un vecteur de \mathbb{K}^n si convergence, sinon \emptyset

```

1: Fonction  $X \leftarrow \text{RSLMETHITER} (\mathbb{A}, \mathbf{b}, \text{ITERFONC}, \mathbf{x}^0, \varepsilon, \text{kmax})$ 
2:  $k \leftarrow 0, \mathbf{x}^{\text{tol}} \leftarrow \emptyset$ 
3:  $\mathbf{x} \leftarrow \mathbf{x}^0, \mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x}$ ,
4:  $\text{tol} \leftarrow \varepsilon(\|\mathbf{b}\| + 1)$ 
5: Tantque  $\|\mathbf{r}\| > \text{tol}$  et  $k \leq \text{kmax}$  faire
6:    $k \leftarrow k + 1$ 
7:    $\mathbf{p} \leftarrow \mathbf{x}$ 
8:    $\mathbf{x} \leftarrow \text{ITERFONC}(\mathbb{A}, \mathbf{b}, \mathbf{p})$ 
9:    $\mathbf{r} \leftarrow \mathbf{b} - \mathbb{A}\mathbf{x}$ ,
10: Fin Tantque
11: Si  $\|\mathbf{r}\| \leq \text{tol}$  alors
12:    $\mathbf{x}^{\text{tol}} \leftarrow \mathbf{x}$ 
13: Fin Si
14: Fin Fonction
```

Méthode de relaxation utilisant Gauss-Seidel, avec $w \in \mathbb{R}^*$,

$$x_i^{[k+1]} = \frac{w}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{[k+1]} - \sum_{j=i+1}^n A_{ij} x_j^{[k]} \right) + (1-w)x_i^{[k]} \quad \forall i \in \llbracket 1, n \rrbracket$$

Algorithme 19 itération S.O.R.

Données :

- A : matrice de $\mathcal{M}_n(\mathbb{K})$,
- b : vecteur de \mathbb{K}^n ,
- y : vecteur de \mathbb{K}^n ,
- w : réel non nul.

Résultat :

- x : un vecteur de \mathbb{K}^n

- 1: **Fonction** x ← **ITERSOR** (A, b, y, w)
- 2: **Pour** i ← 1 à n **faire**
- 3: S ← 0
- 4: **Pour** j ← 1 à i - 1 **faire**
- 5: S ← S - A(i, j) * x(j)
- 6: **Fin Pour**
- 7: **Pour** j ← i + 1 à n **faire**
- 8: S ← S - A(i, j) * y(j)
- 9: **Fin Pour**
- 10: x(i) ← w * (b(i) - S)/A(i, i) + (1 - w) * x(i)
- 11: **Fin Pour**
- 12: **Fin Fonction**

Paramètre w "en trop" dans l'appel de la fonction **ITERSOR** pour pouvoir utiliser la fonction générique **RSLMETHITER** !

Méthode de relaxation utilisant Gauss-Seidel, avec $w \in \mathbb{R}^*$,

$$x_i^{[k+1]} = \frac{w}{A_{ii}} \left(b_i - \sum_{j=1}^{i-1} A_{ij} x_j^{[k+1]} - \sum_{j=i+1}^n A_{ij} x_j^{[k]} \right) + (1-w)x_i^{[k]} \quad \forall i \in \llbracket 1, n \rrbracket$$

Algorithme 20 Itération S.O.R.

Données :

- A : matrice de $\mathcal{M}_n(\mathbb{K})$,
- b : vecteur de \mathbb{K}^n ,
- y : vecteur de \mathbb{K}^n ,
- w : réel non nul.

Résultat :

- x : un vecteur de \mathbb{K}^n

- 1: **Fonction** x \leftarrow **ITERSOR** (A, b, y, w)
- 2: **Pour** i \leftarrow 1 à n **faire**
- 3: S \leftarrow 0
- 4: **Pour** j \leftarrow 1 à i - 1 **faire**
- 5: S \leftarrow S - A(i, j) * x(j)
- 6: **Fin Pour**
- 7: **Pour** j \leftarrow i + 1 à n **faire**
- 8: S \leftarrow S - A(i, j) * y(j)
- 9: **Fin Pour**
- 10: x(i) \leftarrow w * (b(i) - S)/A(i, i) + (1 - w) * x(i)
- 11: **Fin Pour**
- 12: **Fin Fonction**

Paramètre w "en trop" dans l'appel de la fonction **ITERSOR** pour pouvoir utiliser la fonction générique **RSLMETHITER** !

Fonction X \leftarrow **RSLSOR3** (A, b, w, x⁰, ε, kmax)
ITERFUN \leftarrow ((M, r, s) \mapsto **ITERSOR**(M, r, s, w))
X \leftarrow **RSLMETHITER**(A, b, **ITERFUN**, x⁰, ε, kmax)
Fin Fonction