

Chapitre IV

Résolution de systèmes linéaires

Plan

1 Conditionnement

2 Méthodes directes

- Matrices diagonales
- Matrices triangulaires inférieures
- Matrices triangulaires supérieures
- Ecriture algébrique

- Résultats théoriques
- Utilisation pratique
- Résultats théoriques
- Résolution d'un système linéaire
- Algorithme : Factorisation positive de Cholesky
- La transformation de Householder

3 Méthodes itératives

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

Résoudre

$$Ax = b$$

Le calcul de la matrice inverse 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 M inversible tel que MA *facilement* inversible

$$Ax = b \iff MAx = Mb.$$

- **Méthodes itératives** : On cherche B et c ,

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

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

Conditionnement

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?

Exemple de R.S. Wilson

Soient

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

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

$$Ax = b \iff x^t = (1, 1, 1, 1)$$

$$Au = (b + \Delta b) \iff u^t = \begin{pmatrix} 91 & -9 & 27 & 79 \\ 50 & -25 & 20 & 100 \end{pmatrix} \approx (1.8, -0.36, 1.3, 0.79)$$

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

$$(A + \Delta A)y = (b + \Delta b) \iff y^t = \begin{pmatrix} 18283543 & 31504261 & 3741501 & 5235241 \\ -461600 & 461600 & -230800 & 461600 \end{pmatrix} \approx (-39.61, 68.25, -16.21, 11.34)$$

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 1.1

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

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

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

Proposition:

Soit A une matrice régulière. On a les propriétés suivantes

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

Théorème 2:

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

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

Supposons $b \neq 0$, alors l'inégalité

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

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

Théorème:

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

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

Supposons $b \neq 0$, alors on a

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

Remarque 2.1

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

Plan

- 1 Conditionnement
 - Résultats théoriques
 - Utilisation pratique
- 2 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
- 3 Méthodes itératives
 - Factorisation LDL*
 - Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation positive de Cholesky
 - Factorisation QR
 - La transformation de Householder

Système diagonal

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

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

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

$$Ax = b.$$

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

- 1: **Fonction** $x \leftarrow \text{RSLMATDIAG}(A, b)$
- 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 $A \in \mathcal{M}_n(\mathbb{K})$ 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 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 inversible $\iff A_{i,i} \neq 0, \forall i \in \llbracket 1, n \rrbracket$

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

Soit $i \in \llbracket 1, n \rrbracket$, $(Ax)_i = b_i \iff \sum_{j=1}^n A_{i,j}x_j = b_i$.

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

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

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

<p>Algorithme 2 $\overline{R_0}$</p> <p>1: Résoudre $Ax = b$ en calculant successivement x_1, x_2, \dots, x_n.</p>	<p>Algorithme 2 $\overline{R_1}$</p> <p>1: Pour $i \leftarrow 1$ à n faire</p> <p>2: $x_i \leftarrow \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j}x_j \right)$</p> <p>3: Fin Pour</p>
--	---

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

<p>Algorithme 2 $\overline{R_1}$</p> <p>1: Pour $i \leftarrow 1$ à n faire</p> <p>2: $x_i \leftarrow \frac{1}{A_{i,i}} \left(b_i - \sum_{j=1}^{i-1} A_{i,j}x_j \right)$</p> <p>3: Fin Pour</p>	<p>Algorithme 2 $\overline{R_2}$</p> <p>1: Pour $i \leftarrow 1$ à n faire</p> <p>2: $S \leftarrow \sum_{j=1}^{i-1} A_{i,j}x_j$</p> <p>3: $x_i \leftarrow (b_i - S)/A_{i,i}$</p> <p>4: Fin Pour</p>
---	--

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

Algorithme 2 $\overline{\mathbb{R}_2}$

- 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

Algorithme 2 $\overline{\mathbb{R}_3}$

- 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

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

Algorithme Fonction **RSLTriINF** permettant de résoudre le système linéaire triangulaire inférieur inversible $Ax = b$.

$$Ax = b.$$

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

- 1: Fonction $x \leftarrow$ **RSLTriINF** (A, 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 $A \in \mathcal{M}_n(\mathbb{K})$ triangulaire supérieure inversible ($A_{i,j} = 0$ si $i > j$)

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

 **Exercice**

Ecrire la fonction **RSLTriSUP** permettant de résoudre le système triangulaire supérieure $Ax = b$.

Plan

- 1 Conditionnement
 - Résultats théoriques
 - Utilisation pratique
- 2 Méthodes directes
 - Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
 - Exercices et résultats préliminaires
 - Méthode de Gauss-Jordan
 - Ecriture algébrique
 - Factorisation LU
- 3 Méthodes itératives

 **Lemme 3.1:**

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

- la matrice $P_n^{[i,j]} A$ est matrice A dont on a permuté les **lignes** i et j ,
- la matrice $A P_n^{[i,j]}$ est matrice A dont on a permuté les **colonnes** i et j ,

 **Lemme 3.2:**

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

$$EAe_1 = A_{1,1}e_1 \quad (3)$$

où e_1 est le premier vecteur de la base canonique de \mathbb{C}^n .

 **Théorème 4: Décomposition de Schur**



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 Conditionnement
- 2 Méthodes directes
 - Matrices particulières
 - Matrices diagonales
 - Matrices triangulaires inférieures
 - Matrices triangulaires supérieures
 - Exercices et résultats préliminaires
 - Méthode de Gauss-Jordan
 - Ecriture algébrique
 - Factorisation LU
- 3 Méthodes itératives
 - Résultats théoriques
 - Utilisation pratique
 - Factorisation LDL*
 - Factorisation de Cholesky
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation positive de Cholesky
 - Factorisation QR
 - La transformation de Householder

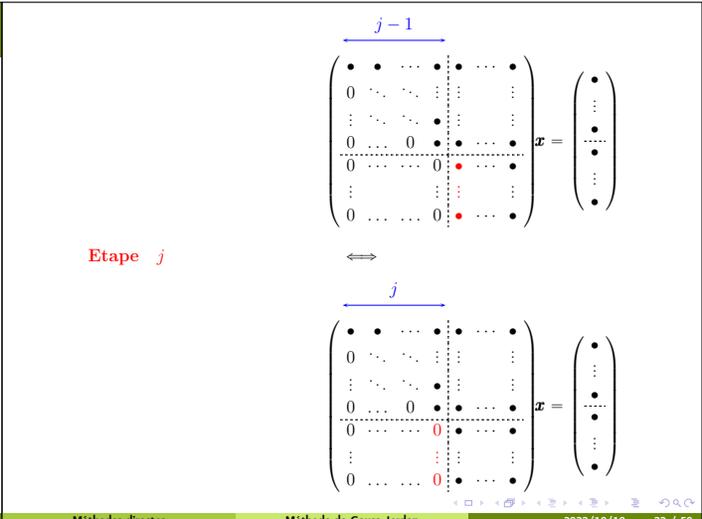
Algorithme de Gauss-Jordan

$$Ax = b \iff Ux = f$$

où 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.



Algorithm Algorithme de Gauss-Jordan formel pour la résolution de $Ax = b$

- 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 $\mathcal{L}_i \leftarrow \mathcal{L}_i - \frac{A_{i,j}}{A_{j,j}} \mathcal{L}_j$
- 6: Fin Pour
- 7: Fin Pour
- 8: Résoudre le système triangulaire supérieur par la méthode de la remontée.

Algorithm Algorithme de Gauss-Jordan avec fonctions pour la résolution de $Ax = b$

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

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

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

Algorithm Recherche d'un pivot pour l'algorithme de Gauss-Jordan

Données : A : matrice de $\mathcal{M}_n(\mathbb{K})$.
 j : entier, $1 \leq j \leq n$.

Résultat : k : entier, indice ligne pivot

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

Algorithm Permutte deux lignes d'une matrice et d'un vecteur

Données : A : matrice de $\mathcal{M}_n(\mathbb{K})$.
 b : vecteur de \mathbb{K}^n .
 j, k : entiers, $1 \leq j, k \leq n$.

Résultat : A et b modifiés.

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

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

Données : A : matrice de $\mathcal{M}_n(\mathbb{K})$.
 b : vecteur de \mathbb{K}^n .
 j, i : entiers, $1 \leq j, i \leq n$.
 α : scalaire de \mathbb{K} .

Résultat : A et b modifiés.

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

Exercice 1: Méthode de Gauss, écriture algébrique

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

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

Q.2

• Montrer par récurrence sur l'ordre des matrices que pour toute matrice $A_n \in \mathcal{M}_n(\mathbb{C})$ inversible, il existe une matrice $S_n \in \mathcal{M}_n(\mathbb{C})$ telle que $|\det S_n| = 1$ et $S_n A_n = U_n$ avec U_n matrice triangulaire supérieure inversible.

• Soit $\mathbf{b} \in \mathbb{C}^n$. En supposant connue la décomposition précédente $S_n A_n = U_n$, expliquer comment résoudre le système $A_n \mathbf{x} = \mathbf{b}$.

Q.3 Que peut-on dire si A est non inversible?

Indication : utiliser les Lemmes 3.1 et 3.2.

On a donc démontré le théorème suivant

Théorème 5

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

Plan

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

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.48, page 207) 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 6: 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 précédent $U = E^{[n-1]} \dots E^{[1]} A$

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

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

Exercice 2

Montrer que si A inversible admet une factorisation LU alors toutes ses sous-matrices principales sont inversibles.

Corollaire 6.1:



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 6.2

Si la matrice $A \in \mathcal{M}_n(\mathbb{C})$ 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 7: Factorisation LU avec permutations ☆☆☆☆☆

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

$$PA = LU. \quad (5)$$

Utilisation pratique de la factorisation LU

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

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

$$Ax = b \iff LUx = b \quad (6)$$

est équivalent à

Utilisation pratique de la factorisation LU

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

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

$$Ax = b \iff LUx = b \quad (6)$$

est équivalent à

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

$$Ux = y \quad (7)$$

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

$$Ly = b. \quad (8)$$

Algorithme de résolution de systèmes linéaire par LU

Algorithm Fonction RSLFactLU permettant de résoudre, par une factorisation LU, le système linéaire $Ax = b$ où A est une matrice de $\mathcal{M}_n(\mathbb{K})$, dont toutes les sous-matrices principales sont inversibles, et $b \in \mathbb{K}^n$.

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

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

- 1: **Fonction** $x \leftarrow \text{RSLFactLU}(A, b)$
- 2: $[L, U] \leftarrow \text{FactLU}(A)$ ▷ Factorisation LU
- 3: $y \leftarrow \text{RSLTriInf}(L, b)$ ▷ Résolution du système $Ly = b$
- 4: $x \leftarrow \text{RSLTriSup}(U, y)$ ▷ Résolution du système $Ux = y$
- 5: **Fin Fonction**

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

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

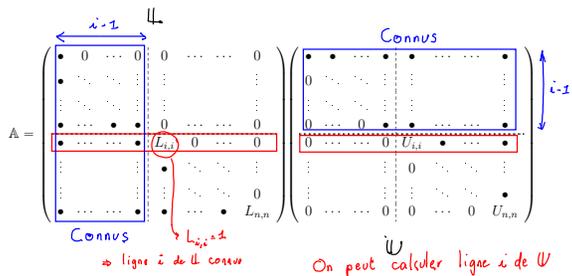
$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots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

On connaît A , on cherche L et U

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

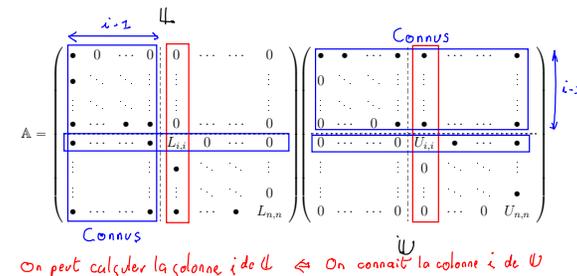
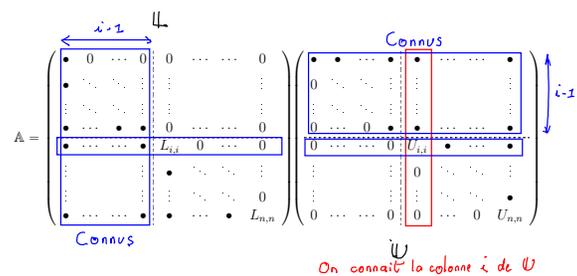
$$\begin{pmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{n,1} & A_{n,2} & \dots & A_{n,n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \dots & \dots & 0 \\ L_{2,1} & 1 & 0 & \dots & 0 \\ L_{3,1} & L_{3,2} & \ddots & \ddots & \vdots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

- **Etape 1 :**
 - ▷ On connaît la première ligne de $L \implies$ on peut calculer la première ligne de U



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

$$U_{i,j} = A_{i,j} - \sum_{k=1}^{i-1} L_{i,k} U_{k,j}, \quad \forall j \in [i, n].$$



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

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

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

Algorithme 9 $\overline{R_0}$

1: Calculer les matrices L et U

Algorithme 9 $\overline{R_1}$

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 $\overline{R_1}$

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 $\overline{R_2}$

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 $\overline{R_2}$

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 $\overline{R_3}$

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: $L_{j,i} \leftarrow \frac{1}{U_{i,i}} \left(A_{j,i} - S_2 \right)$

16: Fin Pour

17: Fin Pour

```

Algorithme 9  $\overline{R_3}$ 
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  $j \leftarrow i+1$  à  $n$  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  $\overline{R_4}$ 
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  $j \leftarrow i+1$  à  $n$  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
  
```

Algorithm Fonction **FactLU** permet de calculer les matrices L et U dites matrice de factorisation LU associée à la matrice A, telle que $A = LU$

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

Résultat : L : matrice de $M_n(\mathbb{K})$ triangulaire inférieure avec $L_{i,i} = 1, \forall i \in [1, n]$
 U : matrice de $M_n(\mathbb{K})$ triangulaire supérieure.

```

1: Fonction [L, U]  $\leftarrow$  FactLU ( A )
2:   U  $\leftarrow$  0n
3:   L  $\leftarrow$  In
4:   Pour  $i \leftarrow 1$  à  $n$  faire
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  $j \leftarrow i+1$  à  $n$  faire
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 Fonction
  
```

Plan

- Conditionnement
- 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 positive de Cholesky
 - Factorisation QR
 - La tranformation de Householder
- Méthodes itératives

Soit $A \in M_n(\mathbb{C})$ hermitienne inversible admettant une factorisation LU. On pose $D = \text{diag } U$ et $R = D^{-1}U$.

R est alors triangulaire supérieure à diagonale unité. On a alors $A = LU = LDD^{-1}U = LDR$.

A 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 8: Factorisation LDL* ★★★★★

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

$$A = LDL^* \tag{9}$$

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

Corollaire 8.1: ★★★★★

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

Plan

- Conditionnement
- 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 de Cholesky**
 - Résultats théoriques
 - Résolution d'un système linéaire
 - Algorithme : Factorisation positive de Cholesky
 - Factorisation QR
 - La tranformation de Householder
- Méthodes itératives

♥ **Definition**

Une **factorisation régulière de Cholesky** d'une matrice $A \in \mathcal{M}_n(\mathbb{C})$ est une factorisation $A = BB^*$ où B est une matrice triangulaire inférieure inversible. Si les coefficients diagonaux de B sont positifs, on parle alors d'une **factorisation positive de Cholesky**.

📖 **Théorème: Factorisation de Cholesky** ★★★★★

La matrice $A \in \mathcal{M}_n(\mathbb{C})$ admet une factorisation régulière de Cholesky **si et seulement si** la matrice 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 $b \in \mathbb{C}^n$. On note B la matrice de factorisation positive de Cholesky de A .

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

$$Ax = b \iff BB^*x = b \tag{10}$$

est équivalent à

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

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

$$Ax = b \iff BB^*x = b \tag{10}$$

est équivalent à

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

$$B^*x = y \tag{11}$$

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

$$By = b. \tag{12}$$

Algorithme de résolution de systèmes linéaire par Cholesky

Algorithm Fonction `RSLCHOLESKY` permettant de résoudre, par une factorisation de Cholesky positive, le système linéaire

$$Ax = b$$

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

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

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

- 1: **Fonction** $x \leftarrow \text{RSLCHOLESKY}(A, b)$
- 2: $B \leftarrow \text{CHOLESKY}(A)$ ▷ Factorisation positive de Cholesky
- 3: $y \leftarrow \text{RSLTRIINF}(B, b)$ ▷ Résolution du système $By = b$
- 4: $U \leftarrow \text{MATADJOINTE}(B)$ ▷ Calcul de la matrice adjointe de B
- 5: $x \leftarrow \text{RSLTRISUP}(U, y)$ ▷ Résolution du système $B^*x = y$
- 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 $B_{i,i} \in \mathbb{R}^{+*}, \forall i \in [1, n]$, telle que

$$A = BB^*$$

c'est à dire

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

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 $B_{i,i} \in \mathbb{R}^{+*}, \forall i \in [1, n]$, telle que

$$A = BB^*$$

c'est à dire

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

- 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 $B_{i,j} \in \mathbb{R}^{+*}, \forall i \in \llbracket 1, n \rrbracket$, telle que

$$A = BB^*$$

c'est à dire

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

- 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 B .
Peut-on calculer la colonne i de B ?

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

Or 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 $B_{j,i}, \forall j \in \llbracket i + 1, n \rrbracket$.

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

Comme B est triangulaire inférieure on obtient

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

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

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

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

Algorithme 11 $\overline{\mathcal{R}_0}$

1: Calculer la matrice B

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 B .

3: Calculer la $i^{\text{ème}}$ colonne de 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 B .

3: Calculer la $i^{\text{ème}}$ colonne de 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 $\overline{\mathcal{R}_2}$

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 $\overline{\mathcal{R}_3}$

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  $\mathbb{R}_3$ 
1: Pour  $i \leftarrow 1$  à  $n$  faire
2:    $S_1 \leftarrow \sum_{j=1}^{i-1} |B_{ij}|^2$ 
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  $\mathbb{R}_4$ 
1: Pour  $i \leftarrow 1$  à  $n$  faire
2:    $S_1 \leftarrow 0$ 
3:   Pour  $j \leftarrow 1$  à  $i-1$  faire
4:      $S_1 \leftarrow S_1 + |B_{ij}|^2$ 
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 Fonction CHOLESKY permettant de calculer la matrice  $B$ , dite matrice de factorisation positive de Cholesky associée à la matrice  $A$ , telle que  $A = BB^*$ .
Données :  $A$  : matrice de  $\mathcal{M}_n(\mathbb{C})$  hermitienne définie positive.
Résultat :  $B$  : matrice de  $\mathcal{M}_n(\mathbb{C})$  triangulaire inférieure avec  $B(i,i) > 0, \forall i \in [1, n]$ 

1: Fonction  $B \leftarrow$  CHOLESKY ( $A$ )
2:  Pour  $i \leftarrow 1$  à  $n$  faire
3:     $S_1 \leftarrow 0$ 
4:    Pour  $j \leftarrow 1$  à  $i-1$  faire
5:       $S_1 \leftarrow S_1 + |B(i,j)|^2$ 
6:    Fin Pour
7:     $B(i,i) \leftarrow \text{sqrt}(A(i,i) - S_1)$ 
8:    Pour  $j \leftarrow 1$  à  $i-1$  faire
9:       $B(j,i) \leftarrow 0$ 
10:   Fin Pour
11:   Pour  $j \leftarrow i+1$  à  $n$  faire
12:      $S_2 \leftarrow 0$ 
13:     Pour  $k \leftarrow 1$  à  $i-1$  faire
14:        $S_2 \leftarrow S_2 + B(j,k) * \overline{B(i,k)}$ 
15:     Fin Pour
16:      $B(j,i) \leftarrow (A(j,i) - S_2) / B(i,i)$ 
17:   Fin Pour
18: Fin Fonction

```

Exercice 3

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

Plan

- 1 Conditionnement
- 2 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 positive de Cholesky
- Factorisation QR
 - La transformation de Householder
- 3 Méthodes itératives

♥ Définition: Matrice élémentaire de Householder

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

$$H(u) = I - 2uu^* \tag{13}$$

Propriété:

Toute matrice élémentaire de Householder est hermitienne et unitaire.

Propriété:

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

$$H(u)(x_{\perp} + x_{\parallel}) = x_{\perp} - x_{\parallel} \tag{14}$$

et

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

Théorème:

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

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

 **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(a, b)` (produit scalaire de deux vecteurs), `NORM(a)` (norme 2 d'un vecteur), `ARG(z)` (argument d'un nombre complexe), `MATPROD(A, B)` (produit de deux matrices), `TRANSPOSE(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 composantes étant dans $]0, 1[$ (loi uniforme).

Q.3 Proposer un programme permettant de vérifier que $\delta = 1$ est le "meilleur" choix.

 **Corollaire 8.2:** 

Soit $\mathbf{a} \in \mathbb{C}^n$ avec $a_1 \neq 0$ et $\exists j \in [2, n]$ 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 9:**  ★★★★★

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

$$\mathbb{A} = \mathbb{Q}\mathbb{R}. \quad (18)$$

Si \mathbb{A} est réelle alors \mathbb{Q} et \mathbb{R} sont aussi réelles et l'on peut choisir \mathbb{Q} de telle sorte que les coefficients diagonaux de \mathbb{R} soient positifs. De plus, si \mathbb{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 $\mathbb{A} \in \mathcal{M}_n(\mathbb{C})$. On pourra utiliser la fonction `HOUSEHOLDER` (voir Exercice 55, page 79).

Q.2 Ecrire un programme permettant de tester cette fonction.

Plan

- 1 Conditionnement
- 2 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 positive de Cholesky
 - Factorisation QR
 - La transformation de Householder
- 3 Méthodes itératives