Algèbre linéaire appliquée

CM/TD, ING1 GM Apprentissage

Enseignant :

Razvan Apredoaei
razvan.apredoaei@cyu.fr

Table des matières

Chapitre 1

Rappels sur les matrices, normes matricielles

Dans ce chapitre, $\mathbb{K}$ désigne $\mathbb{R}$ ou $\mathbb{C}$. Étant donnée une matrice $A \in M_n(\mathbb{K})$, on notera par ${}^tA$ sa **transposée** et par A^* son **adjoint**, i.e.,

$$A^* := {}^t\overline{A}.$$

On rappelle que le produit $AB \in M_{m,p}(\mathbb{K})$ de deux matrices $A \in M_{m,n}(\mathbb{K})$ et $B \in M_{n,p}(\mathbb{K})$ est défini par la formule

$$\forall 1 \leq i \leq m, 1 \leq j \leq p, \quad (AB)_{i,j} = \sum_{k=1}^{n} a_{i,k} b_{k,j}.$$

On dit qu'une matrice $A \in M_n(\mathbb{K})$ est **inversible** s'il existe $A^{-1} \in M_n(\mathbb{K})$ telle que $AA^{-1} = A^{-1}A = I_n$. On note $GL_n(\mathbb{K})$ l'ensemble des matrices carrées inversibles à coefficient dans $\mathbb{K}$.

1.1 Déterminant, trace

On note S_n l'ensemble des permutations de $\{1, \ldots, n\}$.

Définition 1.1. Soit $A \in M_n(\mathbb{K})$, $A = (a_{i,j})1 \leq i, j \leq n$. Le **déterminant** de A est défini par

$$\det A = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{\sigma(i),i},$$

Proposition 1.2 (Caractérisation du déterminant)**.** Le déterminant est l'unique forme multilinéaire alternée en les colonnes sur $M_n(\mathbb{K})$ telle que $\det I_n = 1$, c'est-à-dire

(i) $\det(C_1, \ldots, \lambda C_i + \mu C_i', \ldots C_n) = \lambda \det(C_1, \ldots, C_i, \ldots, C_n) + \mu \det(C_1, \ldots, C_i', \ldots, C_n)$.
(ii) $\det(C_1, \ldots, C_i, \ldots, C_j, \ldots C_n) = -\det(C_1, \ldots, C_j, \ldots, C_i, \ldots C_n)$.
(iii) $\det I_n = 1$.

Proposition 1.3 (Propriétés du déterminant)**.** On vérifie

(i) A est inversible $\Leftrightarrow \det A \neq 0$.
(ii) $\det(AB) = \det(A)\det(B)$
(iii) $\det({}^tA) = \det A$
(iv) $\det(A^*) = \overline{\det(A)}$.

Définition 1.4 (Cofacteurs)**.** Soit $A = (a_{i,j})_{1\leq i,j\leq n} \in M_n(\mathbb{K})$. Pour $1 \leq i, j \leq n$, on définit $\Delta_{i,j}$ le cofacteur de $a_{i,j}$ par

$$\Delta_{i,j} := (-1)^{i+j} \det \begin{pmatrix} a_{1,1} & \dots & a_{1,j-1} & a_{1,j+1} & \dots & a_{1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{i-1,1} & \dots & a_{i-1,j-1} & a_{i-1,j+1} & \dots & a_{i-1,n} \\ a_{i+1,1} & \dots & a_{i+1,j-1} & a_{i+1,j+1} & \dots & a_{i+1,n} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{n,1} & \dots & a_{n,j-1} & a_{n,j+1} & \dots & a_{n,n} \end{pmatrix}.$$

Proposition 1.5 (Développement du déterminant)**.** Pour $A \in M_n(\mathbb{K})$, on vérifie

$$\det A = \sum_{1\leq i\leq n} a_{i,j}\Delta_{i,j} = \sum_{1\leq j\leq n} a_{i,j}\Delta_{i,j}.$$

Remarque 1.6. On rappelle que pour calculer un déterminant, on utilise la méthode du pivot de Gauss (voir Chapitre 2).

Exercice 1.7. Calculer le déterminant $\det \begin{pmatrix} 2 & 1 & 0 & 4 \\ -4 & -2 & 3 & -7 \\ 4 & 1 & -2 & 8 \\ 0 & -3 & -12 & -1 \end{pmatrix}$.

Proposition 1.8 (Formule de la comatrice)**.** Soit $A \in M_n(\mathbb{R})$. On note $\mathrm{com}(A)$ la matrice des cofacteurs de A, définie par

$$\mathrm{com}(A) = (\Delta_{i,j})_{1\leq i,j\leq n} \in M_n(\mathbb{R}),$$

alors on vérifie

$$A\, {}^t\mathrm{com}(A) = (\det A) I_n,$$

et si A est inversible, on a

$$A^{-1} = \frac{1}{\det A}\, {}^t\mathrm{com}(A).$$

On rappelle que la trace d'une matrice $A \in M_n(\mathbb{R})$ est définie par la formule

$$\mathrm{Tr}(A) = \sum_{i=1}^{n} a_{i,i}.$$

Proposition 1.9 (Propriétés de la trace)**.** On a

(i) $\mathrm{Tr}(AB) = \mathrm{Tr}(BA)$

(ii) $\mathrm{Tr}({}^tA) = \mathrm{Tr}(A)$, $\mathrm{Tr}(A^*) = \overline{\mathrm{Tr}(A)}$

1.2 Noyau, Image, Rang

Soit $A \in M_{m,n}(\mathbb{K})$. La matrice A induit une application

$$A : \begin{array}{ccc} \mathbb{K}^n & \longrightarrow & \mathbb{K}^m \\ x & \longmapsto & Ax. \end{array}$$

Définition 1.10. L'**image** de $A \in M_{m,n}(\mathbb{K})$ est définie par

$$\text{Im } A := \{Ax, x \in \mathbb{K}^n\} \subset \mathbb{K}^m,$$

le **noyau** de A est défini par

$$\text{Ker } A := \{x \in \mathbb{K}^n : Ax = 0\}.$$

On appelle **rang** de la matrice A la dimension de $\text{Im}(A)$. On le note $\text{rg}(A) := \dim(\text{Im } A)$.

Proposition 1.11. Le rang d'une matrice $A \in M_{m,n}(\mathbb{K})$ est la dimension du sous-espace vectoriel de $\mathbb{R}^m$ engendré par les colonnes de A.

Proposition 1.12 (Théorème du rang)**.** Soit $A \in M_{m,n}(\mathbb{K})$. On a

$$m = \dim(\text{Ker } A) + \text{rg}(A).$$

Définition 1.13. Soit $A \in M_{m,n}(\mathbb{K})$. Soit $q \in \{1, \ldots, \min(m, n)\}$. On appelle sous-matrice d'ordre q de A toute matrice obtenue en éliminant $m - q$ lignes et $n - q$ colonnes de A.

Proposition 1.14. Le rang de A est égal à la valeur maximale de q pour laquelle il existe une sous-matrice d'ordre q de A de déterminant non nul.

Remarque 1.15. En pratique, pour un calcul de rang, on utilise l'algorithme du pivot de Gauss pour déterminer la dimension de $\text{Ker}A$. On applique ensuite le théorème du rang.

Exercice 1.16. Calculer le rang des matrices suivantes.

$$(i)\ A = \begin{pmatrix} 4 & 0 & -2 \\ 2 & -1 & 1 \end{pmatrix}, \qquad (ii)\ B = \begin{pmatrix} 1 & 2 & 0 \\ 1 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad (iii)\ C = \begin{pmatrix} 1 & -1 & 0 & 0 \\ -1 & 1 & -1 & 0 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}.$$

Proposition 1.17. Dans le cas d'une matrice carrée $A \in M_n(K)$, les propriétés suivantes sont équivalentes.

(i) A est inversible,
(ii) $\det A \neq 0$,
(iii) $\text{Ker}A = \{0\}$,
(iv) $\text{rg}A = n$,
(v) Les colonnes (resp. les lignes) de A sont linéairement indépendantes.

1.3 Spectre d'une matrice et polynôme caractéristique

Dans cette section, on suppose que $A \in M_n(\mathbb{K})$.

Définition 1.18 (Valeur propre, vecteur propre, spectre)**.** On dit que $\lambda \in \mathbb{K}$ est une **valeur propre** de A s'il existe $x \in \mathbb{K}^n \setminus \{0\}$ tel que $Ax = \lambda x$. On dit alors que x est un **vecteur propre** de A associé à la valeur propre λ. Le **spectre** de A est l'ensemble des valeurs propres de A, on le note $\mathrm{Spec}(A)$.

Proposition 1.19. Les points suivants sont équivalents
(i) $\lambda \in \mathrm{Spec}(A)$,
(ii) $\mathrm{Ker}(A - \lambda I_n) \neq \{0\}$,
(iii) $\det(A - \lambda I_n) = 0$
Ainsi, le spectre de A vérifie

$$\mathrm{Spec}(A) := \{\lambda \in \mathbb{K} : \det(A - \lambda I_n) = 0\}.$$

Corollaire 1.20. On a $\mathrm{Spec}({}^tA) = \mathrm{Spec}(A)$, $\mathrm{Spec}(A^*) = \overline{\mathrm{Spec}(A)}$.

Définition 1.21 (Polynôme caractéristique)**.** Le **polynôme caractérisique** de $A \in M_n(\mathbb{K})$ est défini par

$$\chi_A(X) := \det(XI_n - A).$$

On vérifie alors

$$\mathrm{Spec}(A) = \{x \in \mathbb{K} : \chi_A(x) = 0\}.$$

Remarque 1.22. Soit $A \in M_n(\mathbb{R})$. On peut autant voir A comme une matrice à coefficients réels que comme une matrice à coefficients complexes. On note alors $\mathrm{Spec}_{\mathbb{R}}(A) \subset \mathbb{R}$ son **spectre réel**, qui correspond aux **racines réelles** de $\chi_A(X)$ et $\mathrm{Spec}_{\mathbb{C}}(A) \subset \mathbb{C}$ son **spectre complexe**, qui correspond aux **racines complexes** de $\chi_A(X)$.

Proposition 1.23. Soit $A \in M_n(\mathbb{C})$. On vérifie

$$\det A = \prod_{\lambda \in \mathrm{Spec}(A)} \lambda, \qquad \mathrm{Tr}(A) = \sum_{\lambda \in \mathrm{Spec}(A)} \lambda.$$

1.4 Matrices diagonalisables

Définition 1.24. On dit que $A, B \in M_n(\mathbb{K})$ sont **semblables** s'il existe $P \in GL_n(\mathbb{K})$ telle que $A = PBP^{-1}$. Deux matrices semblables ont le même polynôme caractéristique et le même spectre.

Définition 1.25. On dit que $A \in M_n(\mathbb{K})$ est **diagonalisable** si A est semblable à une matrice diagonale, i.e. il existe $P \in GL_n(\mathbb{K})$ et $\lambda_1, \dots, \lambda_n \in \mathbb{K}$ tels que

$$A = P\mathrm{Diag}(\lambda_1, \dots, \lambda_n)P^{-1}.$$

On en particulier $\mathrm{Spec}(A) = \{\lambda_1, \dots, \lambda_n\}$.

Définition 1.26.

1. Une matrice $A \in M_n(\mathbb{K})$ est dite **symétrique** si $A = {}^tA$. Elle est dite **autoadjointe** si $A = A^*$.
2. Une matrice $A \in M_n(\mathbb{K})$ est dite **normale** si $AA^* = A^*A$.

Définition 1.27.

1. Une matrice $O \in M_n(\mathbb{R})$ est dite **orthogonale** si $O^tO = I_n$. On note $O_n(\mathbb{R})$ l'ensemble des matrices orthogonales.
2. Une matrice $U \in M_n(\mathbb{C})$ est dite **unitaire** si $UU^* = I_n$. On note $U_n(\mathbb{C})$ l'ensemble des matrices unitaires.

Proposition 1.28.

1. Si $A \in M_n(\mathbb{R})$ est symmétrique, A est diagonalisable en base orthonormée : il existe $O \in O_n(\mathbb{R})$ et $\lambda_1, \dots, \lambda_n \in \mathbb{R}$ tels que $\mathrm{Spec}(A) = \{\lambda_1, \dots, \lambda_n\}$ et

$$A = O\mathrm{Diag}(\lambda_1, \dots, \lambda_n)^tO.$$

2. Si $A \in M_n(\mathbb{C})$ est normale, A est diagonalisable, et il existe $U \in U_n(\mathbb{C})$ et $\lambda_1, \dots, \lambda_n \in \mathbb{C}$ tels que $\mathrm{Spec}(A) = \{\lambda_1, \dots, \lambda_n\}$ et

$$A = U\mathrm{Diag}(\lambda_1, \dots, \lambda_n)U^*.$$

1.5 Matrices symétriques positives

Définition 1.29. Soit $A \in M_n(\mathbb{R})$ une matrice symétrique.
(i) On dit que A est **positive** si toutes ses valeurs propres sont positives.
(ii) On dit que A est **définie positive** si toutes ses valeurs propres sont strictement positives.

Proposition 1.30. Soit $A \in M_n(\mathbb{R})$ une matrice symétrique.
(i) A est positive $\Leftrightarrow \forall X \in \mathbb{R}^n \setminus \{0\}, \quad {}^tXAX \geq 0$.
(ii) A est définie positive $\Leftrightarrow \forall X \in \mathbb{R}^n \setminus \{0\}, \quad {}^tXAX > 0$.

Soit $A \in M_n(\mathbb{K})$. Pour $k \in \{1, \dots, n\}$, on définit Δ_k le **k-ème mineur principal** de la matrice A par

$$\Delta_k(A) := \det \begin{pmatrix} a_{1,1} & \dots & a_{1,k} \\ \vdots & & \vdots \\ a_{k,1} & \dots & a_{k,k} \end{pmatrix}.$$

Théorème 1.31 (Critère de Sylvester)**.** Soit $A \in M_n(\mathbb{R})$. Alors A est définie positive si et seulement si ses mineurs principaux sont strictement positif, c'est-à-dire si et seulement si $\forall k \in \{1, \dots, n\}, \Delta_k(A) > 0$.

Exercice 1.32. Montrer que la matrice $A = \begin{pmatrix} 1 & 1/2 & 1/3 & 1/4 \\ 1/2 & 1/3 & 1/4 & 1/5 \\ 1/3 & 1/4 & 1/5 & 1/6 \\ 1/4 & 1/5 & 1/6 & 1/7 \end{pmatrix}$ est définie positive, de deux manières différentes :

(i) En utilisant le critère de Sylvester.

(ii) En montrant que A est diagonalisable de valeurs propres toutes strictement positives.

Remarque 1.33. La matrice A ci-dessus est une **matrice de Hilbert**. Plus généralement, la matrice de Hilbert de taille $n \times n$ est la matrice A définie par $a_{i,j} = \frac{1}{i+j-1}$. C'est un exemple de matrice dite mal conditionnée, ce qui pose souvent des problèmes de convergence en analyse numérique.

1.6 Matrices définies positives, matrices à diagonale dominante

Définition 1.34 (Matrice (semi) définie positive)**.** Une matrice $A \in M_n(\mathbb{R})$ est *définie positive* sur $\mathbb{R}^n$ si

$$\langle Ax, x \rangle > 0, \quad \forall x \in \mathbb{R}^n, x \neq 0.$$

Si l'inégalité au sens strict est remplacée par une inégalité au sens large, alors la matrice est dite *semi-définie positive*.

Définition 1.35 (Partie symétrique, partie antisymétrique)**.** Soit $A \in M_n(\mathbb{R})$. Les matrices

$$A_S = \frac{1}{2}(A + {}^tA), \qquad A_{AS} = \frac{1}{2}(A - {}^tA),$$

sont appelées respectivement *partie symétrique* et *partie antisymétrique* de A. On vérifie $A = A_S + A_{AS}$.

Proposition 1.36. Soit $A \in M_n(\mathbb{R})$. A est définie positive si et seulement si sa parti symétrique A_S est définie positive.

Exercice 1.37. Prouver la proposition.

Proposition 1.38. Une matrice carrée $A \in M_n(\mathbb{R})$ symétrique, définie positive est inversible.

Définition 1.39 (Matrice à diagonale dominante)**.** Une matrice $A \in M_n(\mathbb{R})$ est dite à *diagonale dominante par ligne* si

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, \quad \text{avec } i \in \{1, \ldots, n\},$$

Elle est dite à *diagonale dominante par colonne* si

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ji}|, \quad \text{avec } i \in \{1, \ldots, n\}.$$

Si les inégalité ci-dessus sont strictes, alors la matrice est dite à *diagonale dominante stricte*.

Proposition 1.40. Une matrice à diagonale dominante stricte qui est symétrique est définie positive.

1.7 Produits scalaires et normes

Définition 1.41. Un *produit scalaire* sur un $\mathbb{R}$-espace vectoriel V est une application

$$\langle \cdot, \cdot \rangle : V \times V \longrightarrow \mathbb{R},$$

telle que $\langle \cdot, \cdot \rangle$ est

1. **Linéaire :** $\forall x, z \in V,\ \forall \lambda, \mu \in \mathbb{R},\ \langle \lambda x + \mu y, z \rangle = \lambda \langle x, z \rangle + \mu \langle y, z \rangle$
2. **Symétrique :** $\forall x, y \in V\ \langle x, y \rangle = \langle y, x \rangle,$
3. **Définie positive :** $\forall x \in V, \quad \big(x \neq 0 \Rightarrow \langle x, x \rangle > 0\big) \quad \text{et} \quad \big(\langle x, x \rangle = 0 \Leftrightarrow x = 0\big).$

Exemple 1.42. Le produit scalaire canonique sur $\mathbb{R}^n$ est donné par $\langle X, Y \rangle = {}^tXY,\ X, Y \in \mathbb{R}^n$.

Définition 1.43. Un *produit hermitien* sur un $\mathbb{C}$-espace vectoriel V est une application

$$\langle \cdot, \cdot \rangle : V \times V \longrightarrow \mathbb{C},$$

telle que $\langle \cdot, \cdot \rangle$ est

1. **Sesquilinéaire :**

$$\forall x, z \in V,\ \forall \lambda, \mu \in \mathbb{C},\ \langle \lambda x + \mu y, z \rangle = \lambda \langle x, z \rangle + \mu \langle y, z \rangle$$
$$\forall x, z \in V,\ \forall \lambda, \mu \in \mathbb{C},\ \langle z, \lambda x + \mu y \rangle = \overline{\lambda} \langle z, x \rangle + \overline{\mu} \langle z, y \rangle,$$

2. **Antiymétrique :** $\forall x, y \in V\ \langle x, y \rangle = \overline{\langle y, x \rangle},$
3. **Définie positive :** $\forall x \in V, \quad \big(x \neq 0 \Rightarrow \langle x, x \rangle > 0\big) \quad \text{et} \quad \big(\langle x, x \rangle = 0 \Leftrightarrow x = 0\big).$

Exemple 1.44. Le produit hermitien canonique sur $\mathbb{C}^n$ est donné par $\langle X, Y \rangle = {}^tX\overline{Y},\ X, Y \in \mathbb{C}^n$.

Proposition 1.45. Pour le produit scalaire canonique sur $\mathbb{R}^n$,

$$\forall\, A \in M_n(\mathbb{R}), \forall x, y \in \mathbb{R}^n, \langle Ax, y\rangle = \left\langle x, {}^tAy\right\rangle .$$

Proposition 1.46. On rappelle $A^* = {}^t\overline{A}$. Pour le produit scalaire canonique sur $\mathbb{C}^n$,

$$\forall\, A \in M_n(\mathbb{C}), \forall x, y \in \mathbb{C}^n, \langle Ax, y\rangle = \langle x, A^*y\rangle .$$

1.8 Normes sur un espace vectoriel

Définition 1.47. Soit V un $\mathbb{K}$-espace vectoriel. On dit qu'une application $\|\cdot\|$ est une norme sur V si elle satisfait les propriétés suivantes

1. **Positivité :** $\forall v \in V\ \|v\| \geq 0$,
2. **Homogénéité :** $\forall \alpha \in \mathbb{K},\ \|\alpha v\| = |\alpha|\|v\|$,
3. **Inégalité triangulaire :** $\forall v, w \in V, \|v + w\| \leq \|v\| + \|w\|$,
4. **Séparation :** $\forall v \in V,\ \|v\| = 0 \Rightarrow v = 0$.

Si $\|\cdot\|$ ne vérifie par 4., on parle de semi-norme.

Corollaire 1.48 (de l'inégalité triangulaire)**.** $\forall v, w \in V, \quad |\|v\| - \|w\|| \leq \|v - w\|$.

Définition 1.49 (Norme p)**.** Pour $p \in [1, +\infty[$, on définit la norme p sur $\mathbb{R}^n$ par

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p\right)^{1/p},$$

et pour $p = \infty$, $\|x\|_\infty := \max_{1\leq i\leq n} |x_i|$.

Remarque 1.50.

1. $\|x\|_\infty = \lim\limits_{p\to+\infty} \|x\|_p$
2. Pour $p = 2$, $\|x\|_2 = \left(\sum\limits_{i=1}^{n} |x_i|^2\right)^{1/2} = \sqrt{x\,{}^tx}$. La norme 2 est la norme euclidienne associée au produit scalaire (ou hermitien) canonique.

Proposition 1.51. Inégalité de Hölder

$$\forall p, q \geq 1 : \frac{1}{p} + \frac{1}{q} = 1, \quad \forall x, y \in V,\ |\langle x, y\rangle| \leq \|x\|_p \|x\|_q$$

Remarque 1.52. Pour $p = q = 2$, on retrouve l'*inégalité de Cauchy–Schwarz* $\langle x, y\rangle \leq \|x\|_2\|y\|_2$.

Définition 1.53. Normes équivalentes On dit que deux normes $\|\cdot\|_A$ et $\|\cdot\|_B$ sur un $\mathbb{K}$-espace vectoriel V sont équivalentes si

$$\exists C, C' > 0 : \ \forall x \in V, \quad C\|x\|_A \leq \|x\|_B \leq C'\|x\|_B$$

Proposition 1.54. Si V est de dimension finie, toutes les normes sur V sont équivalentes.

Remarque 1.55. Dans les différents chapitres de ce cours, on travaillera généralement dans le cas $V = \mathbb{R}^n$ ou $V = \mathbb{C}^n$.

Définition 1.56. Soit $(V, \|\cdot\|)$ un espace vectoriel normé. Soit $(x_n)_{n\in\mathbb{N}} \in V^{\mathbb{N}}$. On dit que la suite $(x_n)_{n\in\mathbb{N}}$ converge vers $x \in V$ si $\|x_n - x\| \underset{n\to+\infty}{\longrightarrow} 0$, i.e., si

$$\forall \varepsilon > 0, \ \exists n_0 \in \mathbb{N} : \ \forall n \geq n_0, \ \|x_n - x\| \leq \varepsilon.$$

Remarque 1.57. On suppose V de dimension finie. D'après la Proposition 1.54, si $(x_n)_{n\in\mathbb{N}}$ converge vers $x \in V$ pour une norme sur V, alors $(x_n)_{n\in\mathbb{N}}$ converge vers x pour toutes les autres normes sur V.

1.9 Normes sur les matrices

L'espace $M_n(\mathbb{K})$ des matrices à coefficients dans $\mathbb{K}$ a une structure de $\mathbb{K}$-espace vectoriel, isomorphe à $\mathbb{K}^{n^2}$. On peut donc le munir de normes. Dans le cadre de ce cours, on souhaitera distinguer les normes sur les matrices des normes sur les vecteurs. On notera ainsi par $|||\cdot|||$ les normes sur les matrices, et par $\|\cdot\|$ les normes sur les vecteurs.

Définition 1.58. On dit qu'une norme matricielle $|||\cdot|||$ est compatible ou consistante avec une norme vectorielle $\|\cdot\|$ si

$$\forall\ A \in M_n(\mathbb{K}), \forall v \in \mathbb{K}^n, \quad \|Av\| \leq |||A|||\|v\|.$$

Définition 1.59. On dit qu'une norme matricielle $|||\cdot|||$ est sous-multiplicative si

$$\forall A, B \in M_n(\mathbb{K}), \ |||AB||| \leq |||A|||\,|||B|||.$$

Exemple 1.60.

1. La norme matricielle définie par $|||A|||_\infty := \max_{1\le i,j\le n} |a_{i,j}|$ n'est pas sous-multiplicative. En effet prenons
$$A = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}, \quad \text{alors} \quad AB = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix},$$
et $|||AB|||_\infty = 2 > |||A|||_\infty |||B|||_\infty$.
2. La *norme de Frobenius* est définie par
$$|||A|||_F = \sqrt{\mathrm{Tr}(A^*A)} = \left(\sum_{1\le i,j\le n} |a_{i,j}|^2 \right)^{1/2},$$
et $|||\cdot|||_F$ est compatible avec la norme euclidienne. En effet
$$\begin{aligned} \|Ax\|_2^2 &= \sum_{i=1}^n \left| \sum_{j=1}^n a_{i,j} x_j \right|^2 \\ &\le \sum_{i=1}^n \left(\sum_{j=1}^n |a_{i,j}|^2 \right) \left(\sum_{j=1}^n x_j^2 \right) \\ &= \left(\sum_{1\le i,j\le n} |a_{i,j}|^2 \right) \left(\sum_{j=1}^n x_j^2 \right) \\ &= |||A|||_F^2 \|x\|^2, \end{aligned}$$
on a donc bien $\|Ax\| \le |||A|||_F \|x\|$.

Exercice 1.61. Montrer que $|||\cdot|||_F$ est sous-multiplicative.

Définition 1.62 (Norme matricielle subordonnée)**.** Soit $\|\cdot\|$ une norme vectorielle sur $\mathbb{K}^n$. On définit par
$$|||A||| := \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|},$$
la norme subordonnée à $\|\cdot\|$.

Exemple 1.63 (Normes matricielles subordonnées aux normes p)**.** Soit $p \in [1, +\infty[\cup \{+\infty\}$ et $\|\cdot\|_p$ la norme p sur $\mathbb{K}^n$. Pour $A \in M_n(\mathbb{R})$ on a
$$|||A|||_p := \sup_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p},$$
et on vérifie les égalités
$$|||A|||_1 = \max_{1\le j\le n} \sum_{i=1}^n |a_{i,j}|, \qquad |||A|||_\infty = \max_{1\le i\le n} \sum_{j=1}^n |a_{i,j}|.$$

Théorème 1.64. Soit $|||\cdot|||$ une norme matricielle subordonnée à une norme vectorielle $\|\cdot\|$. Alors
1. $\|Ax\| \leq |||A|||\|x\|$, i.e., $|||\cdot|||$ est une norme compatible avec $\|\cdot\|$,
2. $|||I_n||| = 1$,
3. $|||AB||| \leq |||A|||||B|||$, i.e. $|||\cdot|||$ est sous-multiplicative.

Exemple 1.65. La norme de Frobenius définie dans l'exemple 1.60 n'est en général pas une norme subordonnée. En effet

$$|||I_n|||_F = \sqrt{\mathrm{Tr}(I_n^* I_n)} = \sqrt{\mathrm{Tr}(I_n)} = \sqrt{n} \neq 1 \quad \text{pour } n \geq 2.$$

1.10 Relation entre norme matricielle et rayon spectral

1.10.1 Norme 2 et rayon spectral

Définition 1.66 (rayon spectral)**.** On rappelle que $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$. On définit le rayon spectral de $A \in M_n(\mathbb{K})$ comme le module de la plus grande valeur propre complexe de A.

$$\rho(A) := \max_{\lambda \in \mathrm{Spec}_{\mathbb{C}}(A)} |\lambda|.$$

Proposition 1.67 (Décomposition en valeurs singulières)**.** Soit $A \in M_{m,n}(\mathbb{C})$. Il existe $U \in M_m(\mathbb{C})$, $V \in M_n(\mathbb{C})$ et $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p \geq 0$ telles que

$$U^* A V = \mathrm{Diag}(\sigma_1, \ldots, \sigma_p).$$

on appelle cette égalité une décomposition en valeurs singulières de A et les σ_i sont appelées valeurs singulières.

Remarque 1.68. Si $A \in M_n(\mathbb{C})$ est hermitienne, les σ_i correspondent aux modules des valeurs propres de A.

Théorème 1.69. Soit $A \in M_n(\mathbb{K})$. Soit $\sigma_1(A)$ la plus grande valeur singulière de A. Alors

$$|||A|||_2 = \sqrt{\rho(A^* A)} = \sqrt{\rho(A^* A)} = \sigma_1(A),$$

en particulier, si A est hermitienne, on a $|||A|||_2 = \rho(A)$, et si A est unitaire, on a $|||A|||_2 = 1$.

1.10.2 Comparaison des normes p

Proposition 1.70. On vérifie

$$\max_{1 \leq i,j \leq n} |a_{i,j}| \leq |||A|||_2 \leq n \max_{1 \leq i,j \leq n} |a_{i,j}|, \qquad \frac{1}{n}|||A|||_\infty \leq |||A|||_2 \leq \sqrt{n}|||A|||_\infty,$$

$$\frac{1}{\sqrt{n}}|||A|||_1 \leq |||A|||_2 \leq \sqrt{n}|||A|||_1, \qquad |||A|||_2 \leq \sqrt{|||A|||_1 |||A|||_\infty}.$$

1.10.3 Résultats généraux

Proposition 1.71. Si $|||\cdot|||$ est une norme matricielle consistante avec une norme vectorielle $\|\cdot\|$, alors $\forall A \in M_n(\mathbb{K}),\ \rho(A) \leq |||A|||$.

Démonstration. Soit λ une valeur propre de A. Il existe $v \neq 0$ vecteur propre de A tel que $Av = \lambda v$. Alors par consistance

$$|\lambda|\|v\| = \|Av\| \leq |||A|||\|v\|,$$

donc $|\lambda| \leq |||A|||$. □

Définition 1.72. On dit que $(A_k)_{k\in\mathbb{N}} \in M_n(\mathbb{K})^{\mathbb{N}}$ converge vers $A \in M_n(\mathbb{K})$ si

$$|||A_k - A||| \underset{k\to\infty}{\longrightarrow} 0.$$

Remarque 1.73. On rappelle qu'en dimension finie, toutes les normes sont équivalentes. Ainsi, converger pour une norme matricielle, c'est converger pour toutes les autres.

Théorème 1.74. Soit $A \in M_n(\mathbb{K})$. Alors

$$\lim_{k\to\infty} A^k = 0 \Leftrightarrow \rho(A) < 1,$$

de plus, $\sum_{k\geq 0} A^k$ converge si et seulement si $\rho(A) < 1$. Dans ce cas

$$\sum_{k=0}^{\infty} A^k = (I_n - A)^{-1},$$

et il existe une norme matricielle subordonnée $|||\cdot|||$ telle que $|||A||| < 1$ et

$$\frac{1}{1+|||A|||} \leq |||(I_n - A)^{-1}||| \leq \frac{1}{1-|||A|||}.$$

Chapitre 2

Résolution de systèmes linéaires : méthodes directes

On cherche à résoudre des systèmes linéaires de la forme $Ax = b$. On sera amenés à considérer deux types de méthodes : les méthodes dites *directes*, à l'aide desquelles on se ramène à l'étude d'un système $A'x = b'$, plus simple à résoudre, et qui donnent une solution exacte au système, et les méthodes dites *itératives*, qui renvoient une solution approchée x_N après N étapes d'exécution, et telle que Ax_N tend vers b quand N tend vers $+\infty$. Ce chapitre se concentre sur les méthodes directes.

2.1 Rappels sur les sytèmes linéaires, règle de Cramer

Définition 2.1. Un **système linéaire** à m équations et n inconnues est la donnée de $A \in M_{m,n}(\mathbb{R})$ et $b \in \mathbb{R}^n$. Résoudre le système linéaire $Ax = b$, c'est déterminer $\{x \in \mathbb{R}^n : Ax = b\}$. Un système linéaire est dit **de Cramer** si et seulement si la matrice A est carrée ($m = n$) et inversible.

Proposition 2.2 (Règle de Cramer)**.** Soit $Ax = b$ un système linéaire de Cramer, $A \in M_n(\mathbb{R})$, $b \in \mathbb{R}^n$. Il existe un unique $x \in \mathbb{R}^n$ tel que $Ax = b$, dont les coordonnées sont données par

$$x_i = \frac{\det A_i}{\det A},$$

où $\det A_i$ est le déterminant de la matrice où on a remplacé la i-ème colonne de A par le vecteur b.

$$\det A_i = \det \begin{pmatrix} a_{1,1} & \dots & a_{1,i-1} & b_1 & a_{1,i+1} & \dots & a_{1,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{j,1} & \dots & a_{j,i-1} & b_j & a_{j,i+1} & \dots & a_{j,n} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ a_{n,1} & \dots & a_{n,i-1} & b_n & a_{n,i+1} & \dots & a_{n,n} \end{pmatrix}$$

Remarque 2.3. Le coût en calcul de la règle de Cramer est très élevé. Si on développe brutalement les déterminants $\det(A_i)$, il peut aller jusqu'à $\mathcal{O}(n!)$. Si on utilise la méthode du pivot de Gauss pour les calculer, il descend à $\mathcal{O}(n^4)$, car on calcule n déterminants de taille $(n-1)\times(n-1)$. En effet, pour une matrice $n \times n$, la complexité de la méthode du pivot de Gauss est en $\mathcal{O}(n^3)$.

Exemple 2.4. Soient $A = \begin{pmatrix} 2 & 1 \\ 3 & 4 \end{pmatrix}$ et $b = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$. On a $\det A = 5 \neq 0$, le système est donc de Cramer. La règle de Cramer donne

$$x_1 = \frac{1}{5} \det \begin{pmatrix} 1 & 1 \\ 2 & 4 \end{pmatrix} = \frac{2}{5}, \qquad x_2 = \frac{1}{5} \det \begin{pmatrix} 2 & 3 \\ 1 & 2 \end{pmatrix} = \frac{1}{5}.$$

On vérifie que le vecteur $x = \begin{pmatrix} 2/5 \\ 1/5 \end{pmatrix}$ est bien solution de $Ax = b$.

2.2 Le cas des systèmes triangulaires supérieurs et inférieurs

2.2.1 Système triangulaire supérieur

On veut résoudre le système $Ux = b$ avec U triangulaire supérieure, c'est-à-dire de la forme

$$U = \begin{pmatrix} u_{1,1} & \dots & \dots & u_{1,n} \\ 0 & \ddots & & \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & u_{n,n} \end{pmatrix} \in M_n(\mathbb{R}).$$

Proposition 2.5. Le système $Ux = b$ est de Cramer si et seulement si

$$\forall i \in \{1, \dots, n\},\ u_{i,i} \neq 0.$$

Démonstration. Une matrice triangulaire supérieure est inversible si et seulement si tous ses coefficients diagonaux sont non nuls. □

La résolution d'un système triangulaire supérieur est naturelle. En effet de manière générale on vérifie

$$Ux = \begin{pmatrix} u_{1,1} & \dots & \dots & \dots & u_{1,n} \\ 0 & \ddots & & & \vdots \\ \vdots & \ddots & u_{i,i} & \dots & u_{i,n} \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & \dots & 0 & u_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} u_{1,1}x_1 + \dots + u_{1,n}x_n \\ \vdots \\ u_{i,i}x_i + \dots + u_{i,n}x_n \\ \vdots \\ u_{n,n}x_n \end{pmatrix}.$$

Si x est solution de $Ux = b$, alors on a

$$b_i = u_{i,i}x_i + \dots + u_{i,n}x_n,$$

avec $u_{i,i} \neq 0$ puisqu'on a supposé que U est inversible. On en déduit la formule récurrente

$$\begin{cases} x_i = \frac{1}{u_{i,i}} \left(b_i - u_{i,i+1}x_{i+1} - \dots - u_{i,n}x_n \right) \\ x_n = \frac{b_n}{x_{n,n}}. \end{cases}$$

À l'aide de cette formule, on obtient l'algorithme suivant.

Algorithme 2.1 : Résolution d'un système triangulaire supérieur

Entrées : $b \in \mathbb{R}^n$, $U \in M_n(\mathbb{R})$ triangulaire supérieure inversible.
Sorties : $x \in \mathbb{R}^n$ tel que $Ux = b$.
Initialisation $x = (x_1, \dots, x_n) = (0, \dots, 0)$;
$x_n \leftarrow b_n / u_{n,n}$;
pour $i = n-1$ *à* $i = 1$ **faire**
  $x_i \leftarrow b_i$;
  pour $j = i+1$ *à* $j = n$ **faire**
    $x_i \leftarrow x_j - u_{i,j} x_j$;
  fin
  $x_i \leftarrow x_i / u_{i,i}$;
fin
retourner x ;

Exercice 2.6. En appliquant l'Algorithme 2.1, résoudre le système $Ux = b$ pour

$$U = \begin{pmatrix} 1 & 2 & 3 \\ 0 & 4 & 5 \\ 0 & 0 & 6 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Proposition 2.7. La complexité de l'algorithme de résolution d'un système triangulaire supérieur est en $\mathcal{O}(n^2)$.

Exercice 2.8. Prouver la Proposition 2.7.

2.2.2 Système triangulaire inférieur

Le cas d'un système triangulaire inférieur se traite de la même manière. On souhaite résoudre $Lx = b$ avec L triangulaire inférieure, c'est-à-dire

$$L = \begin{pmatrix} l_{1,1} & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ l_{n,1} & \dots & \dots & l_{n,n} \end{pmatrix} \in M_n(\mathbb{R}).$$

De même que précédemment, le système $Lx = b$ est de Cramer si et seulement si $\forall i \in \{1, \dots, n\}, l_{i,i} \neq 0$. Pour résoudre le système $Lx = b$, on effectue les mêmes opérations que pour résoudre le système $Ux = b$, mais en parcourant les lignes du haut vers le bas, plutôt que du bas vers le haut. On en déduit l'algorithme qui suit.

Algorithme 2.2 : Résolution d'un système triangulaire inférieur

Entrées : $b \in \mathbb{R}^n$, $U \in M_n(\mathbb{R})$ triangulaire supérieure inversible.
Sorties : $x \in \mathbb{R}^n$ tel que $Ux = b$.
Initialisation : $x = (x_1, \ldots, x_n) = (0, \ldots, 0)$;
$x_n \leftarrow b_1/l_{1,1}$;
pour $i = 2$ *à* $i = n$ **faire**
  $x_i \leftarrow b_i$;
  pour $j = 1$ *à* $j = i - 1$ **faire**
    $x_i \leftarrow x_j - u_{i,j}x_j$;
  fin
  $x_i \leftarrow x_i/u_{i,i}$;
fin
retourner x ;

Exercice 2.9. En appliquant l'Algorithme 2.2, résoudre le système $Ux = b$ pour

$$U = \begin{pmatrix} 0 & 0 & 6 \\ 0 & 4 & 5 \\ 1 & 2 & 3 \end{pmatrix}, \qquad b = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

2.3 La méthode du pivot de Gauss

La méthode du pivot de Gauss donne un algorithme pour résoudre le système linéaire $Ax = b$, avec dans notre cas $A \in GL_n(\mathbb{R})$, $b \in \mathbb{R}^n$. L'algorithme consiste à transformer le système $Ax = b$ en un système équivalent $Ux = b'$, avec U triangulaire supérieure. Il suffit ensuite d'appliquer l'Algorithme 2.1 de résolution d'un système triangulaire supérieur pour obtenir la solution de $Ax = b$. Le système $Ax = b$ s'écrit également de la manière suivante.

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n = b_1 & (L_1) \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n = b_2 & (L_2) \\ \vdots & \\ a_{n,1}x_1 + a_{n,2}x_2 + \cdots + a_{n,n}x_n = b_n & (L_n), \end{cases}$$

où on a noté (L_i) les lignes du système. On effectue des opérations sur ces lignes (L_i) pour transformer $Ax = b$ en $Ux = b'$.

Algorithme 2.3 : Méthode du pivot de Gauss

Entrées : $A \in GL_n(\mathbb{R})$, $b \in \mathbb{R}^n$.
Sorties : $x \in \mathbb{R}^n$ solution de $Ax = b$.
pour $i = 1$ à $i = n - 1$ **faire**
  si $a_{i,i} = 0$ **alors**
    p est choisi dans $\{l \in [\![i+1, n]\!] \ : \ a_{i,l} \neq 0\}$; `// p désigne le pivot`
    $L_i \leftrightarrow L_p$;
  fin
  pour $k = i + 1$ à $k = n$ **faire**
    $L_k \leftarrow L_k - (a_{k,i}/a_{i,i})L_i$;
  fin
fin
On obtient un nouveau système $Ux = b'$;
Le résoudre à l'aide de l'Algorithme 2.1;
retourner x solution de $Ux = b'$;

Exercice 2.10. Appliquer l'algorithme du pivot de Gauss au système $Ax = b$ pour

$$A = \begin{pmatrix} 2 & 1 & 0 & 4 \\ -4 & -2 & 3 & -7 \\ 4 & 1 & -2 & 8 \\ 0 & -3 & -12 & -1 \end{pmatrix}, \qquad b = \begin{pmatrix} 2 \\ -9 \\ 2 \\ 2 \end{pmatrix}.$$

Remarque 2.11. Dans le choix du pivot, pour améliorer la stabilité numérique, on prend généralement le pivot p tel que $|a_{i,p}| = \max\{|a_{i,l}|, l \in [\![i+1, n]\!]\}$. On parle de méthode du pivot de Gauss avec **pivot maximal**.

Exercice 2.12. Implémenter en Python l'algorithme du pivot de Gauss avec pivot maximal.

Proposition 2.13. La complexité de l'algorithme du pivot de Gauss est en $\mathcal{O}(n^3)$.

Exercice 2.14. Prouver la Proposition 2.13.

2.4 Opérations élémentaires sur les lignes et pivot de Gauss

Les opérations élementaires effectuées dans la méthodes du pivot de Gauss peuvent se traduire en termes de multiplications par des matrices dites de *transposition* et de *transvection*.

2.4.1 Matrices de transposition

L'opération d'échange des lignes $L_i \leftrightarrow L_j$ correspond à transformer le système $Ax = b$ en $P_{i,j}Ax = P_{i,j}b$, avec $P_{i,j}$ la matrice donnée par

$$(P_{i,j})_{k,l} = \begin{cases} 1 & \text{si} \quad k = l \text{ et } k \notin \{i, j\} \\ 1 & \text{si} \quad \{k, l\} = \{i, j\} \\ 0 & \text{sinon.} \end{cases}$$

Autrement dit on a

$$P_{i,j} = \begin{array}{c} \begin{matrix} & & i & & j & & \\ & & \downarrow & & \downarrow & & \end{matrix} \\ \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & (0) & \\ & & 0 & \dots & 1 & & \\ & & \vdots & \ddots & \vdots & & \\ & & 1 & \dots & 0 & & \\ & (0) & & & & \ddots & \\ & & & & & & 1 \end{pmatrix} \end{array} \begin{matrix} \\ \\ \leftarrow i \\ \\ \leftarrow j \\ \\ \\ \end{matrix} .$$

Remarque 2.15. Pour obtenir la matrice $P_{i,j}$, on a échangé les lignes i et j dans la matrice I_n.

Exercice 2.16. Soient

$$A = \begin{pmatrix} 2 & 1 & 0 & 4 \\ -4 & -2 & 3 & -7 \\ 4 & 1 & -2 & 8 \\ 0 & -3 & -12 & -1, \end{pmatrix}, \quad P_{2,3} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

1. Vérifier en calculant $P_{2,3}A$ que la matrice $P_{2,3}$ échange les lignes 2 et 3 de A. Que se passe-t-il si on calcule $AP_{2,3}$ à la place ?
2. Quelle matrice échange les lignes 1 et 2 ? Les lignes 1 et 4 ?

Proposition 2.17. On vérifie $P_{i,i} = \mathrm{Id}$, $(P_{i,j})^{-1} = P_{j,i} = P_{i,j}$.

2.4.2 Matrices de transvection

On suppose $i \neq j$. Toute opération de transvection $L_i \leftarrow L_i + \alpha L_j$ se traduit également de manière matricielle. Cela correspond à transformer le système $Ax = b$ en $T_{i,j}(\alpha)Ax = T_{i,j}(\alpha)b$, où $T_{i,j}(\alpha)$ est donnée par

$$(T_{i,j}(\alpha))_{k,l} = \begin{cases} 1 & \text{si} \quad k = l \\ \alpha & \text{si} \quad (k,l) = (i,j) \\ 0 & \text{sinon,} \end{cases}$$

autrement dit

$$T_{i,j}(\alpha) = \begin{array}{c} \begin{matrix} & & & j & & \\ & & & \downarrow & & \end{matrix} \\ \begin{pmatrix} 1 & & & & & \\ & \ddots & & & (0) & \\ & & \ddots & & & \\ & \alpha & & \ddots & & \\ (0) & & & & \ddots & \\ & & & & & 1 \end{pmatrix} \end{array} \begin{matrix} \\ \\ \\ \\ \leftarrow i \\ \\ \end{matrix} .$$

Exercice 2.18. Soient

$$A = \begin{pmatrix} 2 & 1 & 0 & 4 \\ -4 & -2 & 3 & -7 \\ 4 & 1 & -2 & 8 \\ 0 & -3 & -12 & -1, \end{pmatrix}, \quad T_{3,2}(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & \alpha & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

1. Vérifier en calculant $T_{3,2}(\alpha)A$ que la matrice $T_{3,2}(\alpha)$ correspond bien à l'opération $L_3 \leftarrow L_3 + \alpha L_2$. Que dire du cas $\alpha = 0$?
2. Calculer $T_{3,2}(-1)T_{3,2}(1)$. Plus généralement, quel est l'inverse de $T_{3,2}(\alpha)$?

Proposition 2.19. Soient $i, j \in \{1, \dots, n\}$, $i \neq j$. On a
(i) $T_{i,j}(0) = \mathrm{Id}$, $(T_{i,j})^{-1}(\alpha) = T_{i,j}(-\alpha)$.
(ii) $T_{i,j}(\alpha)$ est triangulaire inférieure si et seulement si $i > j$.

Exercice 2.20. Soient $T_{i,j}(\alpha), i > j$ et $T_{i',j'}(\beta), i' > j'$ deux matrices de transvection triangulaires inférieures. Montrer que leur produit $T_{i,j}(\alpha)T_{i',j'}(\beta)$ est une matrice triangulaire inférieure dont les coefficients diagonaux sont tous 1. Calculer explicitement $T_{i,j}(\alpha)T_{i',j'}(\beta)$, on fera attention aux valeurs de i, j, i', j'.

Remarque 2.21. On rappelle que le produit de matrices triangulaires inférieures L, L' est une matrice triangulaire inférieure dont les coefficients diagonaux correspondent aux produits $l_{i,i}l'_{i,i}$ des termes diagonaux de L et L', preuve en exercice !

2.4.3 Pivot de Gauss : écriture matricielle

On peut écrire l'Algorithme 2.3 en termes matriciels comme suit.

Algorithme 2.4 : Méthode du pivot de Gauss, écriture matricielle

Entrées : $A \in GL_n(\mathbb{R})$, $b \in \mathbb{R}^n$.
Sorties : $U \in GL_n(\mathbb{R})$ triangulaire supérieure, $b' \in \mathbb{R}^n$.
Initialisation : $U = A$, $b' = b$;
pour $i = 1$ à $i = n - 1$ **faire**
  si $a_{i,i} = 0$ **alors**
    p est choisi dans $\{l \in [\![i + 1, n]\!] \ : \ a_{i,l} \neq 0\}$;
    $U \leftarrow P_{i,p}U$;
    $b' \leftarrow P_{i,p}b'$;
  fin
  pour $k = i + 1$ à $k = n$ **faire**
    $U \leftarrow T_{k,i}(-a_{k,i}/a_{i,i})U$;
    $b' \leftarrow T_{k,i}(-a_{k,i}/a_{i,i})b'$;
  fin
fin
retourner (U, b') ;

2.5 Factorisations $PA = LU$ et $A = LU$

2.5.1 La factorisation $PA = LU$

Dans l'Algorithme 2.4 du pivot de Gauss matriciel, on note U_i la matrice obtenue à la fin de l'étape $i \in \{0, \dots, n-1\}$ de la première boucle **pour**, l'étape 0 correspondant à l'initialisation. Alors on a

$$\begin{cases} U_0 = A \\ U_k = L_k P_k A_{k-1}, \end{cases}$$

où P_k désigne une matrice de transposition, et L_k est un produit de matrices de transvections de la forme $T_{k,i}(\alpha)$. Si U désigne la matrice triangulaire supérieure obtenue à l'issue de l'algorithme, on vérifie

$$U = L_{n-1} P_{n-1} L_{n-2} P_{n-2} \dots L_1 P_1 A. \tag{2.1}$$

On doit donc être capable de calculer le produit d'une matrice de transposition et d'une matrice de transvection.

Lemme 2.22. Soient $i, j, k, l \in \{1, \dots, n\}$. Soit $\tau_{i,j} : \{1, \dots, n\} \to \{1, \dots, n\}$ la transposition qui échange i et j, i.e.,

$$\tau_{i,j}(k) = \begin{cases} i & \text{si} \quad \text{k = j}, \\ j & \text{si} \quad \text{k = i}, \\ k & \text{sinon.} \end{cases}$$

On a

$$P_{i,j} T_{k,l}(\alpha) = T_{\tau_{i,j}(k), \tau_{i,j}(l)}(\alpha) P_{i,j}.$$

Corollaire 2.23. Soient $P_1, \dots, P_{n-1}, L_1, \dots, L_{n-1}$ telles que dans (2.1). Il existe des matrices $L'_1, \dots, L'_{n-1}$, produits de matrices de transvections triangulaires inférieures, telles que

$$L_{n-1} P_{n-1} L_{n-2} P_{n-2} \dots L_1 P_1 = L'_{n-1} \dots L'_1 P_{n-1} \dots P_1.$$

Définition 2.24. On dit qu'une matrice $P \in M_n(\mathbb{R})$ est une **matrice de permutation** si chacune de ses lignes et chacune de ses colonnes n'admet qu'un seul coefficient non nul, qui est égal à 1.

Proposition 2.25. Toute matrice de transposition est une matrice de permutation, et toute matrice de permutation peut s'écrire comme produit de matrices de transpositions.

Théorème 2.26 (Factorisation $PA = LU$)**.** Soit $A \in GL_n(\mathbb{R})$. Il existe une matrice de permutation $P \in GL_n(\mathbb{R})$ (i.e., un produit de matrices de transposition) et deux matrices $L, U \in M_n(\mathbb{R})$ telles que

(i) L est triangulaire inférieure, de coefficients diagonaux tous égaux à 1,
(ii) U est triangulaire supérieure,
(iii) $PA = LU$.

Démonstration. D'après la décomposition (2.1) et le Corollaire 2.23, on vérifie

$$U = L_{n-1}P_{n-1}L_{n-2}P_{n-2}\dots L_1P_1A = L'_{n-1}\dots L'_1P_{n-1}\dots P_1A.$$

Posons $P = P_{n-1}\dots P_1$. L'inverse d'une matrice triangulaire inférieure de coefficients diagonaux 1 étant triangulaire inférieure de coefficients diagonaux 1, posons $L = (L'_{n-1}\dots L'_1)^{-1}$. Alors on a $L^{-1}PA = U$, d'où la factorisation $PA = LU$. □

Remarque 2.27. La matrice $P = P_{n-1}\dots P_1$ est le produit des matrices de transpositions qui correspondent à tous les échanges de lignes dans l'algorithme du pivot de Gauss. Pour la calculer, il suffit de partir de la matrice identité, et de lui appliquer, dans l'ordre, toutes les opérations $L_i \leftrightarrow L_j$ qu'on a effectué (voir aussi la Remarque 2.15).

Remarque 2.28. Une stratégie viable pour calculer factorisation $PA = LU$ est la suivante :

(1) On exécute une première fois l'algorithme du pivot de Gauss afin de répertorier tous les échanges de lignes.

(2) On effectue tous les échanges de lignes précédents sur la matrice A, puis on réexécute l'algorithme du pivot de Gauss. On remarque qu'on ne fait plus que des transvections.

Mais cette méthode est coûteuse en calcul. En effet, on doit exécuter l'algorithme du pivot de Gauss deux fois !

Exemple 2.29. On va calculer la factorisation $PA = LU$ de la matrice $A = \begin{pmatrix} 0 & 3 & 1 \\ 4 & 1 & 1 \\ 2 & 2 & 4 \end{pmatrix}$. Les étapes du pivot de Gauss donnent

$$\begin{pmatrix} 0 & 3 & 1 \\ 4 & 1 & 1 \\ 2 & 2 & 4 \end{pmatrix} \xrightarrow[L_1 \leftrightarrow L_3]{} \begin{pmatrix} 2 & 2 & 4 \\ 4 & 1 & 1 \\ 0 & 3 & 1 \end{pmatrix},$$

$$\xrightarrow[L_1 \leftarrow L_2 - 2\times L_1]{} \begin{pmatrix} 2 & 2 & 4 \\ 0 & -3 & -7 \\ 0 & 3 & 1 \end{pmatrix},$$

$$\xrightarrow[L_3 \leftarrow L_3 + 1\times L_2]{} \begin{pmatrix} 2 & 2 & 4 \\ 0 & -3 & -7 \\ 0 & 0 & -6 \end{pmatrix}.$$

Ainsi $T_{3,1}(1)T_{2,1}(-2)P_{1,3}A = U$, avec $U = \begin{pmatrix} 2 & 2 & 4 \\ 0 & -3 & -7 \\ 0 & 0 & -6 \end{pmatrix}$. On obtient $PA = LU$, avec $P = P_{1,3}$, $L = (T_{3,1}(1)T_{2,1}(-2))^{-1} = T_{2,1}(2)T_{3,1}(-1)$, d'où

$$P = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{pmatrix}, \quad L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix}.$$

Exercice 2.30. Calculer la factorisation $PA = LU$ de la matrice $A = \begin{pmatrix} 2 & 6 & -5 \\ 1 & 2 & -3 \\ 1 & -1 & 7 \end{pmatrix}$.

Exercice 2.31. Dans la preuve du Théorème 2.26, on a utilisé que l'inverse d'une matrice triangulaire inférieure inversible L à coefficients diagonaux 1 est triangulaire inférieure à coefficients diagonaux 1. Prouver ce résultat en réécrivant l'Algorithme 2.2 de résolution d'un système triangulaire inférieur à l'aide de matrices de transvections.

2.5.2 La factorisation $A = LU$

Lorsqu'on ne doit effectuer aucune permutation dans l'algorithme du pivot de Gauss, on vérifie $P_{n-1} = P_{n-2} = \dots P_1 = I_n$ dans le produit (2.1). Ce cas correspond exactement au cas où les mineurs principaux de la matrice A sont tous non nuls.

Théorème 2.32 (Factorisation $A = LU$)**.** Soit $A \in GL_n(\mathbb{R})$ telle que pour tout $k \in \{1, \dots, n\}$,

$$\Delta_k(A) = \det \begin{pmatrix} a_{1,1} & \dots & a_{1,k} \\ \vdots & & \vdots \\ a_{k,1} & \dots & a_{k,k} \end{pmatrix} \neq 0.$$

Alors il existe un **unique** couple $(L, U) \in M_n(\mathbb{R}) \times M_n(\mathbb{R})$ avec
(i) L triangulaire inférieure, de coefficients diagonaux tous égaux à 1,
(ii) U triangulaire supérieure,
(iii) $A = LU$.

Remarque 2.33. Il n'y a pas unicité de la factorisation $PA = LU$ car elle dépend des pivots choisis dans l'algorithme du pivot de Gauss. Dans la factorisation $A = LU$, l'hypothèse sur A assure qu'il ne sera pas nécessaire de permuter deux lignes dans l'algorithme, ce qui correspond au cas $P = I_n$.

Démonstration. Nous prouvons l'unicité. Soient L_1, U_1, L_2, U_2 telles que

$$A = L_1U_1 = L_2U_2.$$

Les matrices A, L_1, L_2 étant inversibles, les matrices U_1, U_2 le sont aussi et on obtient l'égalité

$$L_2^{-1}L_1 = U_2U_1^{-1}.$$

L'inverse d'une matrice triangulaire supérieure (resp. inférieure) est triangulaire supérieure (resp. inférieure), donc $L_2^{-1}L_1$ et $U_2U_1^{-1}$ sont triangulaires supérieures et inférieures, donc diagonales. Comme L_2, L_1 sont à diagonale unité, $L_2^{-1}L_1$ est à diagonale unité, donc $L_2^{-1}L_1 = I_n$. On en déduit $L_1 = L_2, U_1 = U_2$. □

2.6 Algorithmes de factorisation « en place »

2.6.1 Décomposition $A = LU$ « en place »

Cet algorithme ne fonctionne que si on ne rencontre aucun pivot dans la méthode du pivot de Gauss pour la matrice $A \in M_n(\mathbb{R})$.

Algorithme 2.5 : Décomposition LU en place

Entrées : $A \in M_n(\mathbb{R})$.
Sorties : Matrice A modifiée (partie inférieure = L, partie supérieure = U).
pour $i = 1$ à $i = n - 1$ **faire**
  si $|a_{i,i}| < \varepsilon$ **alors**
    erreur : pivot nul détecté, arrêt de l'algorithme;
  fin
  pour $k = i + 1$ à $k = n$ **faire**
    $a_{k,i} \leftarrow a_{k,i}/a_{i,i}$; // Stockage de $L_{k,i}$ dans $a_{k,i}$
    pour $j = i + 1$ à $j = n - 1$ **faire**
      $a_{k,j} \leftarrow a_{k,j} - a_{k,i}a_{i,j}$; // Mise à jour de $U_{k,j}$
    fin
  fin
fin
retourner $A = (L - I_n) + U$.

Exercice 2.34. Exécuter l'algorithme, étape par étape, sur la matrice $A = \begin{pmatrix} 2 & 2 & 4 \\ 4 & 1 & 1 \\ 0 & 3 & 1 \end{pmatrix}$. On retrouve les matrices L et U de l'Exemple 2.29.

Proposition 2.35. La complexité de l'algorithme de décomposition LU en place est $\mathcal{O}(n^3)$.

2.6.2 Factorisation $PA = LU$ « en place » avec pivot partiel

Si on exploite la Remarque 2.28, on doit exécuter l'algorithme du pivot de Gauss deux fois, ce qui représente un coût en calcul conséquent. Afin d'éviter ce surcoût, on fait toutes les transformations en même temps. Par ailleurs, pour optimiser le stockage en mémoire, on choisit de stocker les coefficients des transvections (à l'aide desquels on obtiendra la matrice L) dans la partie triangulaire inférieure de la matrice A au fur et à mesure des annulations de coefficients de A par transvection. La partie triangulaire supérieure contiendra la matrice U, la partie triangulaire inférieure la matrice L (on n'a pas besoin de stocker la diagonale faite de 1).

Proposition 2.36. La complexité de l'algorithme de factorisation $PA = LU$ en place avec pivot partiel (page suivante) est $\mathcal{O}(n^3)$.

Algorithme 2.6 : Factorisation $PA = LU$ en place avec pivot partiel

Entrées : $A \in M_n(\mathbb{R})$.
Sorties : Matrice A modifiée, matrice de permutation P.
Initialisation : $P = I_n$;
pour $i = 1$ à $i = n - 1$ **faire**
  `// Recherche du pivot maximum dans la colonne` i
  $p = i$;
  pour $k = i + 1$ à $k = n$ **faire**
    si $|a_{k,i}| > |a_{p,i}|$ **alors**
      $p \leftarrow k$;
    fin
  fin
  si $|a_{p,i}| < \varepsilon$ **alors**
    erreur : matrice singulière ou presque singulière ;
  fin
  `// Échange des lignes dans` A `et dans la matrice` P
  si $p \neq i$ **alors**
    échanger les lignes $A_{i,\cdot}$ et $A_{p,\cdot}$;
    échanger les lignes $P_{i,\cdot}$ et $P_{p,\cdot}$;
  fin
  `// Élimination de Gauss`
  pour $k = i + 1$ à $k = n - 1$ **faire**
    $a_{k,i} \leftarrow a_{k,i}/a_{i,i}$;
    pour $j = i + 1$ à $j = n - 1$ **faire**
      $a_{k,j} \leftarrow a_{k,j} - a_{k,i}a_{i,j}$;
    fin
  fin
fin
retourner A, P.

Exemple 2.37.

	A	P
Initialisation	$\begin{pmatrix} 0 & 2 & -1 & 1 \\ -1 & 1 & 2 & -1 \\ 1 & 1 & -3 & 1 \\ 1 & -1 & -1 & 3 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$L_1 \leftrightarrow L_3$	$\begin{pmatrix} 1 & 1 & -3 & 1 \\ -1 & 1 & 2 & -1 \\ 0 & 2 & -1 & 1 \\ 1 & -1 & -1 & 3 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$L_2 \leftarrow L_2 - (-1) \times L_1$ $L_4 \leftarrow L_4 - 1 \times L_1$	$\begin{pmatrix} 1 & 1 & -3 & 1 \\ -1 & 2 & -1 & 0 \\ 0 & 2 & -1 & 1 \\ 1 & -2 & 2 & 2 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$L_3 \leftarrow L_3 - 1 \times L_2$ $L_4 \leftarrow L_4 - (-1) \times L_2$	$\begin{pmatrix} 1 & 1 & -3 & 1 \\ -1 & 2 & -1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & -1 & 1 & 2 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$L_3 \leftrightarrow L_4$	$\begin{pmatrix} 1 & 1 & -3 & 1 \\ -1 & 2 & -1 & 0 \\ 1 & -1 & 1 & 2 \\ 0 & 1 & 0 & 1 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$

Résultat :

$$P = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 1 & 1 & -3 & 1 \\ 0 & 2 & -1 & 0 \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Exercice 2.38. Refaire la même chose pour la matrice $A = \begin{pmatrix} 1 & -1 & 2 & -1 \\ 2 & -2 & 4 & -3 \\ 1 & 1 & 1 & 0 \\ 1 & -1 & 4 & 3 \end{pmatrix}$.

Exercice 2.39. Implémenter en Python les algorithmes de factorisation $A = LU$ en place et $PA = LU$ en place avec pivot partiel.

2.6.3 Application à la résolution de systèmes

Pour résoudre un système $Ax = b$ avec $A \in GL_n(\mathbb{R})$, $b \in \mathbb{R}^n$, il suffit d'appliquer les étapes suivantes :

(i) Factorisation $PA = LU$ à l'aide de l'Algorithme 2.6.
(ii) On considère le nouveau système $PAx = Pb \Leftrightarrow LUx = Pb$.
(iii) Résolution de $Ly = Pb$ à l'aide de l'Algorithme 2.2, solution $y \in \mathbb{R}^n$.
(iv) Résolution de $Ux = y$ à l'aide de l'Algorithme 2.1.

Exercice 2.40. À l'aide de la méthode ci-dessus, implémenter en Python un algorithme de résolution de systèmes linéaires de la forme $Ax = b$.

2.7 Factorisation de Cholesky

Théorème 2.41 (Factorisation de Cholesky)**.** Soit $A \in M_n(\mathbb{R})$ une matrice symétrique définie positive. Il existe une unique matrice $B \in M_n(\mathbb{R})$ triangulaire inférieure, à coefficients diagonaux tous strictement positifs, telle que

$$A = B^t B.$$

Démonstration. Pour montrer que les mineurs Δ_k de la matrice A sont non nuls, il faut montrer que les sous matrices $(a_{i,j})_{1 \leq i,j \leq k}$ sont définies positives. Pour cela, il suffit d'appliquer critère de Sylvester du Théorème 1.31.

D'après le théorème de factorisation LU, il existe L triangulaire inférieure à coefficients diagonaux 1 et U triangulaire supérieure telles que

$$A = LU.$$

A est symétrique définie positive, donc inversible. Ainsi, U est inversible, donc $\det U = \prod_i u_{i,i} \neq 0$ et $\forall i,\ u_{i,i} \neq 0$. Considérons $D = \text{Diag}(\sqrt{u_{1,1}}, \ldots, \sqrt{u_{n,n}})$. Alors on vérifie

$$A = LU = (LD)\left(D^{-1}U\right).$$

Notons $B = LD$ et $C = D^{-1}U$. B est triangulaire inférieure, U est triangulaire supérieure. B et C sont de la forme

$$B = \begin{pmatrix} \sqrt{u_{1,1}} & 0 & \ldots & 0 \\ * & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ * & \ldots & * & \sqrt{u_{n,n}} \end{pmatrix}, \qquad C = \begin{pmatrix} \sqrt{u_{1,1}} & * & \ldots & * \\ 0 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & * \\ 0 & \ldots & 0 & \sqrt{u_{n,n}} \end{pmatrix}$$

A est symétrique, $A = {}^tA$, et $A = BC$. Donc ${}^tC{}^tB = BC$. Comme dans la preuve de l'unicité de la factorisation LU, on obtient ${}^tBC^{-1} = ({}^tC)^{-1}B$, avec ${}^tBC^{-1}$ triangulaire supérieure à coefficients diagonaux 1 et $({}^tC)^{-1}B$ triangulaire inférieure à coefficients diagonaux 1, donc les deux matrices sont égales à I_n. Ainsi ${}^tB = C$, donc $A = B{}^tB$. L'unicité se prouve de manière similaire, en exploitant la stricte positivité des coefficients diagonaux. □

Pour calculer une factorisation de Cholesky, on procède comme suit. Soit $A \in M_n(\mathbb{R})$ symétrique positive. On cherche $B \in M_n(\mathbb{R})$ de la forme

$$B = \begin{pmatrix} b_{1,1} & 0 & \dots & \dots & 0 \\ b_{2,1} & b_{2,2} & \ddots & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & 0 \\ b_{n,1} & \dots & \dots & b_{n-1,n} & b_{n,n} \end{pmatrix},$$

et telle que $A = B{}^tB$, ce qui se traduit en

$$(B^tB)_{i,j} = a_{i,j} \Leftrightarrow \sum_{k=1}^{\min(i,j)} b_{i,k}b_{j,k} = a_{i,j}.$$

En particulier, on trouve

$$a_{i,i} = \left(\sum_{k=1}^{i-1} b_{i,k}b_{i,k}\right) + b_{i,i}^2 \Rightarrow b_{i,i} = \sqrt{a_{i,i} - \sum_{k=1}^{i-1} b_{i,k}^2}$$

$$\vdots$$

$$a_{i,j} = \left(\sum_{k=1}^{i-1} b_{i,k}b_{j,k}\right) + b_{i,i}b_{j,i} \Rightarrow b_{j,i} = \frac{1}{b_{i,i}}\left(a_{i,j} - \sum_{k=1}^{i-1} b_{i,k}b_{j,k}\right)$$

$$\vdots$$

$$a_{i,n} = \left(\sum_{k=1}^{i-1} b_{i,k}b_{n,k}\right) + b_{i,i}b_{n,i} \Rightarrow b_{n,i} = \frac{1}{b_{i,i}}\left(a_{i,n} - \sum_{k=1}^{i-1} b_{i,k}b_{n,k}\right).$$

On peut ainsi calculer les coefficients de la matrice B à l'aide de la formule récursive ci-dessus.

Exercice 2.42. Calculer la factorisation de Cholesky de la matrice

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 \\ 1 & 2 & 3 & 4 \end{pmatrix}.$$

On vérifiera auparavant que A est bien symétrique définie positive.

Algorithme 2.7 : Calcul d'une factorisation de Cholesky

Entrées : $A \in M_n(\mathbb{R})$ symétrique définie positive.
Sorties : $B \in M_n(\mathbb{R})$ triangulaire inférieure telle que $A = B^t B$.
Initialisation : $B = O_n$; // O_n désigne la matrice nulle
$b_{1,1} = \sqrt{a_{1,1}}$;
pour $i = 2$ à $i = n$ **faire**
  pour $j = 1$ à $j = i - 1$ **faire**
    $b_{i,j} = a_{i,j}$;
    pour $k = 1$ à $k = j - 1$ **faire**
      $b_{i,j} = b_{i,j} - b_{i,k} b_{j,k}$;
    fin
    $b_{i,j} = b_{i,j} / b_{j,j}$;
  fin
  $b_{i,i} = a_{i,i}$;
  pour $k = 1$ à $k = i - 1$ **faire**
    $b_{i,i} = b_{i,i} - b_{i,k}^2$
  fin
  $b_{i,i} = \sqrt{b_{i,i}}$;
fin
retourner B ;

Exercice 2.43. Implémenter en Python l'algorithme ci-dessus. On introduira des tests d'erreurs appropriés.

Proposition 2.44. La complexité de la factorisation de Cholesky est en $\mathcal{O}(n^3)$.

2.8 D'autres factorisations : QR, Schur

2.8.1 Factorisation QR

On a vu que la décomposition LU découle de la méthode du pivot de Gauss. La décomposition QR découle, quant à elle, du procédé d'orthonormalisation de Gram-Schmidt.

Théorème 2.45 (Factorisation QR)**.** Soit $A \in M_n(\mathbb{K})$, $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$. Il existe une matrice unitaire Q et une matrice R triangulaire supérieure à coefficients diagonaux positifs telles que

$$A = QR.$$

Si la matrice A est inversible, cette décomposition est unique.

Pour résoudre le système $Ax = b$, on se ramène alors au système $Rx = Q^* b$, auquel on applique l'Algorithme 2.1 pour un système triangulaire supérieur.

2.8.2 Factorisation de Schur

Théorème 2.46 (Factorisation de Schur)**.** Soit $A \in M_n(\mathbb{K})$, $\mathbb{K} = \mathbb{R}$ ou $\mathbb{C}$. il existe une matrice unitaire $Q \in M_n(\mathbb{K})$ et une matrice $U \in M_n(\mathbb{K})$ triangulaire supérieure telles que

$$A = QUQ^*.$$

On peut résoudre le système $Ax = b$ à l'aide de la factorisation de Schur $A = QUQ^*$ de la manière suivante :

(i) On résout le système triangulaire supérieur $Uy = Q^*b$

(ii) La solution de $Ax = b \Leftrightarrow UQ^*x = Q^*b$ est donnée par $x = Qy$.

2.9 Factorisation LU des matrices tridiagonales

On reprend [3][p.85-86].

Définition 2.47. Une matrice $A \in M_n(\mathbb{K})$ est une **matrice tridiagonale** si A est de la forme

$$A = \begin{pmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \dots & 0 & a_n & b_n \end{pmatrix}$$

Les matrices tridiagonales apparaissent régulièrement en analyse numérique, notamment dans la discrétisation d'équations aux dérivées partielles par la *méthode des différences finies*. La dérivée seconde par rapport à une variable peut par exemple être décrite à l'aide de la matrice

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

Pour A une matrice tridiagonale comme dans la définition ci-dessus, on note Δ_k son mineur d'ordre $k \in \{1, \dots, n\}$, dont on rappelle qu'il est défini par l'expression

$$\Delta_k = \det \begin{pmatrix} b_1 & c_1 & 0 & \dots & 0 \\ a_2 & b_2 & c_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & a_{k-1} & b_{k-1} & c_{k-1} \\ 0 & \dots & 0 & a_k & b_k \end{pmatrix}$$

En développant le déterminant on obtient la relation

$$\begin{cases} \Delta_0 = 1, \quad \Delta_1 = b_1 \\ \Delta_k = b_k\Delta_{k-1} - a_kc_{k-1}\Delta_{k-2}, \quad 2 \leq k \leq n. \end{cases}$$

Théorème 2.48. Si, pour tout $k \in \{1, \dots, n\}$, $\Delta_k \neq 0$, alors la factorisation LU de la matrice A est donnée par

$$L = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ a_2 \frac{\Delta_0}{\Delta_1} & 1 & 0 & \ddots & \vdots \\ 0 & a_3 \frac{\Delta_1}{\Delta_2} & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 1 & 0 \\ 0 & \dots & 0 & a_n \frac{\Delta_{n-2}}{\Delta_{n-1}} & 1 \end{pmatrix},$$

$$U = \begin{pmatrix} \frac{\Delta_1}{\Delta_0} & c_1 & 0 & \dots & 0 \\ 0 & \frac{\Delta_2}{\Delta_1} & c_2 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \dots & 0 & 0 & \frac{\Delta_n}{\Delta_{n-1}} \end{pmatrix}.$$

2.9.1 Application à la résolution de systèmes linéaires

Résoudre le système linéaire $Av = d$ demande alors beaucoup moins d'opérations que les méthodes précédentes. On considère la matrice $\Delta = \mathrm{Diag}(\Delta_0/\Delta_1, \Delta_1/\Delta_2, \dots, \Delta_n/\Delta_{n-1})$. Alors on vérifie

$$A = (L\Delta)(\Delta^{-1}U) = \begin{pmatrix} c_1/z_1 & 0 & 0 & \dots & 0 \\ a_2 & c_2/z_2 & 0 & \ddots & \vdots \\ 0 & a_3 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & a_n & c_n/z_n \end{pmatrix} \begin{pmatrix} 1 & z_1 & 0 & \dots & 0 \\ 0 & 1 & z_2 & \ddots & \vdots \\ 0 & 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & z_{n-1} \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix},$$

avec

$$z_1 = \frac{c_1}{b_1}, \quad z_k = c_k \frac{\Delta_{k-1}}{\Delta_k}, \quad 2 \leq k \leq n.$$

On résout alors le système linéaire $Av = d$ en résolvant d'abord $L\Delta w = d$, puis $\Delta^{-1}Uv = w$, ce qui correspond à calculer les suites

$$\begin{cases} z_1 = \dfrac{c_1}{b_1}, \quad z_k = \dfrac{c_k}{b_k - a_k z_{k-1}}, \quad k \in \{2, 3, \dots, n\}, \\ w_1 = \dfrac{d_1}{b_1}, \quad w_k = \dfrac{z_k}{c_k}(d_k - a_k w_{k-1}) = \dfrac{d_k - a_k w_{k-1}}{b_k - a_k z_{k-1}}, \quad k \in \{2, 3, \dots, n\} \\ v_n = w_n, v_k = w_k - z_k v_{k-1}, \quad k \in \{n-1, n-2, \dots, 1\}. \end{cases}$$

Cette méthode requiert $2(n-1)$ additions, $3(n-1)$ multiplications et $2n$ divisions, **d'où une complexité en** $\mathcal{O}(n)$, ce qui représente une amélioration significative du $\mathcal{O}(n^3)$ de la méthode du pivot de Gauss.

Exercice 2.49. À l'aide de la méthode ci-dessus, implémenter en Python une méthode de résolution d'un système $Av = d$ lorsque A est tridiagonale.

Chapitre 3

Résolution de systèmes linéaires : méthodes itératives

3.1 Introduction

— On cherche à résoudre un système linéaire de la forme $Ax = b$ en calculant la solution de manière approchée.

— Les méthodes qu'on étudie sont de la forme $x^{(k+1)} = Bx^{(k)} + c$, avec $x^{(0)}$ arbitraire, et B et c construits à partir de A et B.

— On dira qu'une méthode itérative est *convergente* si $\lim_{k\to\infty} x^{(k)} = x$, x solution de $Ax = b$.

3.2 Principe général

Théorème 3.1. Soit $Ax = b$ un système de Cramer tel qu'il existe une matrice inversible M et une matrice carrée N telles que $A = M - N$. Alors toute solution de $Ax = b$ est un point fixe de l'application

$$\varphi : x \mapsto M^{-1}Nx + M^{-1}b,$$

et réciproquement tout point fixe de φ est solution de $Ax = b$.

Démonstration.

$$\begin{aligned} Ax = b &\Leftrightarrow (M - N)x = b \\ &\Leftrightarrow Mx = Nx + b \\ &\Leftrightarrow x = M^{-1}Nx + M^{-1}b \\ &\Leftrightarrow x = \varphi(x). \end{aligned}$$

□

On rappelle que le rayon spectral d'une matrice carrée A est défini par

$$\rho(A) = \sup_{\lambda \in \mathrm{Spec}(A)} |\lambda|$$

Dans le chapitre 1, on a vu l'énoncé qui suit.

Lemme 3.2. Soit $n \in \mathbb{N} \setminus \{0\}$, $A \in M_n(\mathbb{R})$. Alors on a l'équivalence

$$\lim_{k\to+\infty} A^k = 0 \iff \forall x \in \mathbb{R}^n,\ \lim_{k\to+\infty} A^k x = 0 \iff \rho(A) < 1$$
$$\iff \text{Il existe une norme } \|\cdot\| \text{ sur } \mathbb{R}^n \text{ telle que } |||A||| < 1.$$
$$\text{pour la norme } |||\cdot||| \text{ subordonnée associée à } \|\cdot\|.$$

Nous aurons besoin du résultat suivant, appelé théorème du point fixe de Banach–Picard.

Théorème 3.3 (Théorème du point fixe de Banach–Picard)**.** Soit $(V, \|\cdot\|)$ un espace vectoriel normé de dimension finie, et $f : V \to V$ une application contractante, c'est-à-dire qu'il exite $0 < k < 1$ tel que, pour tout $x, y \in V$,

$$\|f(x) - f(y)\| \leq k\|x - y\|,$$

alors f possède un unique point fixe l. De plus, toute suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 \in V$ et $u_{n+1} = f(u_n), n \geq 0$, converge vers l et on a les estimations suivantes

1. $\forall n \in \mathbb{N}, \|u_n - l\| \leq k^n\|u_0 - l\|$,
2. $\forall n \in \mathbb{N}, \|u_n - l\| \leq \frac{k}{1-k}\|u_n - u_{n-1}\|$.

Remarque 3.4. Le « vrai » cadre pour énoncer le théorème de Banach–Picard est pour (E, d) un espace métrique complet, et $f : E \to E$ contractante.

Théorème 3.5. Soit $n \in \mathbb{N} \setminus \{0\}$ et (A, b) un système linéaire de Cramer d'ordre n tel qu'il existe $M \in M_n(\mathbb{R})$ et $N \in M_n(\mathbb{R})$ telles que $A = M - N$. Considérons la suite définie par la relation de récurrence

$$\begin{cases} x^{(0)} \in \mathbb{R}^n \\ \forall\ k \in \mathbb{N},\ x^{(k+1)} = \varphi(x^{(k)}) \end{cases} \tag{3.1}$$

où on rappelle que

$$\varphi(x^{(k)}) = M^{-1}Nx^{(k)} + M^{-1}b = x^{(k)} + M^{-1}(b - Ax^{(k)}).$$

Si $\rho(M^{-1}N) < 1$, alors $(x^{(k)})_{k\in\mathbb{N}}$ converge vers la solution du système linéaire (A, b).

Démonstration. Il s'agit de remarquer que si $\rho(M^{-1}N) < 1$, l'application $\varphi(x) = M^{-1}Nx + M^{-1}b$ est contractante pour la norme subordonnée $|||\cdot|||$ du Lemme 3.2. En effet

$$\begin{aligned} \|\varphi(x) - \varphi(y)\| &= \|M^{-1}Nx + M^{-1}b - M^{-1}Ny + M^{-1}b\| \\ &= \|M^{-1}N(x - y)\| \\ &\leq |||M^{-1}N|||\,\|x - y\|, \end{aligned}$$

avec $|||M^{-1}N||| < 1$. D'après le théorème du point fixe de Banach, la suite définie par la relation (3.1) converge pour n'importe quel $x^{(0)} \in \mathbb{R}^n$. On note $x \in \mathbb{R}^n$ sa limite. Par continuité de φ, on a $\varphi(x^{(k)}) \underset{k\to\infty}{\longrightarrow} \varphi(x)$. Or $\varphi(x^{(k)}) = x^{(k+1)} \underset{k\to+\infty}{\longrightarrow} x$, donc $\varphi(x) = x$. D'après le Théorème 3.1, x est solution de $Ax = b$. □

Dans la suite de ce chapitre, pour $A \in M_n(\mathbb{R})$, on écrira $A = D - E - F$ avec

$$A = \begin{pmatrix} \ddots & & -F \\ & D & \\ -E & & \ddots \end{pmatrix},$$

où D désigne la partie diagonale de A, $-E$ sa partie triangulaire inférieure stricte et $-F$ sa partie triangulaire supérieure stricte.

Exemple 3.6. Par exemple pour $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$, on a

$$D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 9 \end{pmatrix}, \quad -E = \begin{pmatrix} 0 & 0 & 0 \\ 4 & 0 & 0 \\ 7 & 8 & 0 \end{pmatrix}, \quad \text{et} \quad -F = \begin{pmatrix} 0 & 2 & 3 \\ 0 & 0 & 6 \\ 0 & 0 & 0 \end{pmatrix}.$$

Nous allons à présent décrire deux méthodes itératives pour résoudre le système linéaire $Ax = b$, qui reposent sur le principe général que l'on vient de décrire. Il s'agit, dans chaque cas, de choisir une bonne décomposition de A de la forme $A = M - N$ pour laquelle la propriété $\rho(M^{-1}N) < 1$ est vérifiée.

3.3 Méthode de Jacobi

Soit $A = D - E - F \in GL_n(\mathbb{R})$. Dans la méthode de Jacobi, on pose

$$M = D \quad \text{et} \quad N = E + F.$$

On a bien sûr $A = M - N$.

3.3.1 Méthode de Jacobi

Définition 3.7 (Définition de la méthode de Jacobi)**.** Soit $A \in GL_n(\mathbb{R})$. Soit D la matrice diagonale dont les coefficients diagonaux sont ceux de A. On suppose D **inversible**, i.e. tous les coefficients diagonaux de A sont non nuls. Soit $b \in \mathbb{R}^n$. On cherche à construire une solution approchée de $Ax = b$. La *méthode de Jacobi associée au système* $Ax = b$ est la suite $x^{(k)}$ définie par la relation de récurrence du Théorème 3.5, i.e.,

$$\begin{cases} x_0 \in \mathbb{R}^n \\ \forall\, k \in \mathbb{N},\ x^{(k+1)} = D^{-1}(E+F)x^{(k)} + D^{-1}b \\ \qquad i.e.,\ x^{(k+1)} = x^{(k)} + D^{-1}(b - Ax^{(k)}). \end{cases} \tag{3.2}$$

Remarque 3.8. D'après le Théorème 3.5, la méthode converge si $\rho(M^{-1}N) < 1$. S'il est aisé de déterminer le spectre d'une matrice 2×2 ou 3×3, il est généralement difficile de calculer le rayon spectral d'une matrice $n \times n$. On utilisera donc plutôt le résultat suivant.

Théorème 3.9 (Convergence de la méthode de Jacobi)**.** La méthode de Jacobi converge

1. Si A est à diagonale strictement dominante, i.e.,

$$\forall i \in \{1, \dots, n\}, \quad a_{i,i} > \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j},$$

2. Ou si A et $2D - A$ sont symétriques définies positives.

D'après (3.2), on vérifie

$$x^{(k+1)} - x^{(k)} = -D^{-1}Ax^{(k)} + D^{-1}b = D^{-1}(-Ax^{(k)} + b),$$

si on note $x^{(k)} = \left(x_1^{(k)}, \dots, x_n^{(k)}\right) \in \mathbb{R}^n$, alors on obtient

$$x_i^{(k+1)} - x_i^{(k)} = \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{n} a_{ij} x_j^{(k)} \right],$$

d'où l'algorithme de calcul de la méthode de Jacobi en page suivante.

Algorithme 3.1 : Méthode de Jacobi

Entrées : $A \in M_n(\mathbb{R})$ inversible, $b \in \mathbb{R}^n$, N nombre d'étapes.
Sorties : $x^{(N)} \in \mathbb{R}^n$ solution approchée de $Ax = b$.
Initialisation : $x = (0, \dots, 0)$; `// On choisit` $x^{(0)} = (0, \dots, 0)$.
pour $k = 1$ *à* $k = N$ **faire**
  $y = (0, \dots, 0)$;
  pour $i = 1$ *à* $i = n$ **faire**
    $S = b_i$;
    pour $j = 1$ *à* $j = n$ **faire**
      $S = S - a_{i,j} x_j$;
    fin
    $y_i = x_i + S/a_{i,i}$;
  fin
  pour $i = 1$ *à* $i = n$ **faire**
    $x_i = y_i$;
  fin
fin
retourner x ; `//` *À l'étape* k, *calcul de* $x^{(k)}$ *dans* (3.2). *L'algorithme renvoie* $x^{(N)}$.

Exercice 3.10. Calculer la complexité de l'algorithme.

Exercice 3.11. Implémenter l'algorithme en Python. On ajoutera un test de convergence, c'est-à-dire qui arrête l'algorithme dès que $|Ax^{(k)} - b| < \varepsilon$ pour $\varepsilon > 0$ donné en argument. N désignera alors un nombre maximal d'étapes.

3.3.2 Méthode de Jacobi relaxée

La méthode de Jacobi relaxée est une perturbation de la méthode de Jacobi standard.

Définition 3.12. Soit $A = D \in GL_n(\mathbb{R})$. Soit D la matrice diagonale dont les coefficients diagonaux sont ceux de A. On suppose D **inversible**, i.e. tous les coefficients diagonaux de A sont non nuls. Soit $b \in \mathbb{R}^n$. On cherche à construire une solution approchée de $Ax = b$. Soit $\omega \in \mathbb{R} \setminus \{0\}$. La *méthode de Jacobi relaxée de paramètre ω associée à $Ax = b$* est la suite récurrente

$$\begin{cases} x_0 \in \mathbb{R}^n \\ \forall k \in \mathbb{N},\ x^{(k+1)} = x^{(k)} + \omega D^{-1}(b - Ax^{(k)}), \end{cases} \tag{3.3}$$

ω est appelé le *paramètre de relaxation de la méthode.*

Remarque 3.13. Dans le cas $\omega = 1$, on retrouve la méthode de Jacobi standard.

Proposition 3.14 (Convergence de la méthode de Jacobi relaxée)**.** Soit $Ax = b$ un système linéaire de Cramer tel que A soit symétrique définie positive. On note D la matrice diagonale dont les coefficients diagonaux sont ceux de A. La méthode de Jacobi relacée de paramètre ω associée à (A, b) converge pour tout $\omega \in \left]0, \dfrac{2}{\rho(D^{-1}A)}\right[$.

Remarque 3.15. Retenir qu'en pratique, il est préférable de prendre ω assez petit.

3.4 Méthode de Gauss-Seidel

Soit $A = D - E - F \in GL_n(\mathbb{R})$. Dans la méthode de Gauss-Seidel, on pose

$$M = D - E \quad \text{et} \quad N = F.$$

On a bien sûr $A = M - N$.

3.4.1 Méthode de Gauss–Seidel

Définition 3.16 (Définition de la méthode de Gauss-Seidel)**.** Soit $A \in GL_n(\mathbb{R})$.

Soit $D - E$ la matrice triangulaire inférieure dont les coefficients correspondent à la partie triangulaire inférieure de A. On suppose $D - E$ **inversible**, i.e. tous les coefficients diagonaux de A sont non nuls.

Soit F matrice triangulaire supérieure stricte correspondant à la partie triangulaire supérieure stricte de A.

Soit $b \in \mathbb{R}^n$. On cherche à construire une solution approchée de $Ax = b$. La *méthode de Gauss-Seidel associée au système $Ax = b$* est la suite $x^{(k)}$ définie par la relation de récurrence

$$\begin{cases} x^{(0)} \in \mathbb{R}^n \\ \forall k \in \mathbb{N},\ x^{(k+1)} = (D - E)^{-1}Fx^{(k)} + (D - E)^{-1}b \\ \quad i.e.,\ x^{(k+1)} = x^{(k)} + (D - E)^{-1}(b - Ax^{(k)}) \end{cases} \tag{3.4}$$

Théorème 3.17 (Convergence de la méthode de Gauss-Seidel)**.** La méthode de Gauss-Seidel converge

1. Si A est à diagonale strictement dominante, i.e.,

$$\forall i \in \{1, \dots, n\}, \quad a_{i,i} > \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{i,j},$$

2. Ou si A symétrique définie positive.

À l'aide de (3.4), on obtient l'égalité $(D - E)x^{(k+1)} = Fx^{(k)} + b$ qui donne, après calcul,

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)} \right), \tag{3.5}$$

Si on note X le vecteur défini par

$$X = \begin{pmatrix} x_1^{(k+1)} \\ \vdots \\ x_{i-1}^{(k+1)} \\ x_i^{(k)} \\ \vdots \\ x_n^{(k)} \end{pmatrix}$$

alors la relation (3.5) devient

$$x_i^{(k+1)} = x_i^{(k)} + \frac{1}{a_{ii}} \left[b_i - \sum_{j=1}^{n} a_{ij} X_j^{(k)} \right],$$

d'où l'algorithme de calcul pour la méthode de Gauss–Seidel :

Algorithme 3.2 : Méthode de Gauss-Seidel

Entrées : $A \in M_n(\mathbb{R})$ inversible, $b \in \mathbb{R}^n$, N nombre d'étapes.
Sorties : $x^{(N)} \in \mathbb{R}^n$ solution approchée de $Ax = b$.
Initialisation : $x = (0, \dots, 0)$; `// On choisit` $x^{(0)} = (0, \dots, 0)$.
pour $k = 1$ *à* $k = N$ **faire**
  pour $i = 1$ *à* $i = n$ **faire**
    $S = 0$;
    pour $j = 1$ *à* $j = n$ **faire**
      $S = S + a_{i,j} x_j$;
    fin
    $x_i = x_i + (b_i - S)/a_{ii}$;
  fin
fin
retourner x ; `//` À l'étape k, `calcul de` $x^{(k)}$ `dans (3.2). L'algorithme renvoie` $x^{(N)}$.

Exercice 3.18. Calculer la complexité de l'algorithme.

Exercice 3.19. Implémenter l'algorithme en Python. On ajoutera un test de convergence, c'est-à-dire qui arrête l'algorithme dès que $|Ax^{(k)} - b| < \varepsilon$ pour $\varepsilon > 0$ donné en argument. N désignera alors un nombre maximal d'étapes.

3.4.2 Méthode de Gauss–Seidel relaxée

Définition 3.20 (Définition de la méthode de Gauss-Seidel relaxée)**.** Soit $A \in GL_n(\mathbb{R})$.

Soit $D - E$ la matrice triangulaire inférieure dont les coefficients correspondent à la partie triangulaire inférieure de A. On suppose $D - E$ **inversible**, i.e. tous les coefficients diagonaux de A sont non nuls.

Soit F matrice triangulaire supérieure stricte correspondant à la partie triangulaire supérieure stricte de A.

Soit $b \in \mathbb{R}^n$. On cherche à construire une solution approchée de $Ax = b$. Soit $\omega \in \mathbb{R} \setminus \{0\}$. La *méthode de Gauss-Seidel relaxée de paramètre ω associée au système $Ax = b$* est la suite $x^{(k)}$ définie par la relation de récurrence

$$\begin{cases} x^{(0)} \in \mathbb{R}^n \\ \forall k \in \mathbb{N},\ x^{(k+1)} = x^{(k)} + \omega(D - E)^{-1}(b - Ax^{(k)}). \end{cases} \tag{3.6}$$

ω est appelé le *paramètre de relaxation* de la méthode.

Remarque 3.21. La méthode de Gauss-Seidel standard correspond à une méthode relaxée de paramètre $\omega = 1$.

On énonce les deux résultats de convergence suivants :

Proposition 3.22 (Condition nécessaire de convergence)**.** Soit (A, b) un système linéaire de Cramer tel que les coefficients diagonaux de A sont tous non nuls. Si la méthode de Gauss-Seidel relaxée de paramètre ω converge, alors $\omega \in]0, 2[$.

Proposition 3.23 (Condition nécessaire et suffisante de convergence)**.** Soit (A, b) un système linéaire de Cramer tel que A soit symétrique définie positive. Alors la méthode de Gauss-Seidel relaxée de paramètre ω associée à (A, b) converge si et seulement si $\omega \in]0, 2[$.

3.5 Exercices

Exercice 3.24. On considère le système $Ax = b$ avec les valeurs de A et b suivantes

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix} \qquad \text{et} \qquad b = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}.$$

1. *Méthode de Jacobi*
   (a) Écrire la méthode de Jacobi pour la résolution du système $Ax = b$ sous la forme $x^{(k+1)} = B_J x^{(k)} + c_J$.
   (b) Calculer le rayon spectral de B_J et en déduire que la méthode de Jacobi converge.
   (c) Calculer $x^{(1)}$ et $x^{(2)}$ pour les choix de $x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ et $x^{(0)} = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$.
2. *Méthode de Gauss-Seidel*
   (a) Écrire la méthode de Gauss–Seidel pour la résolution du système $Ax = b$ sous la forme $x^{(k+1)} = B_{GS} x^{(k)} + c_{GS}$.
   (b) Calculer le rayon spectral de B_{GS} et en déduire que la méthode de Gauss–Seidel converge. Laquelle de ces deux méthodes converge le plus rapidement ?
   (c) Calculer $x^{(1)}$ et $x^{(2)}$ pour les choix de $x^{(0)} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$ et $x^{(0)} = \begin{pmatrix} 0 \\ 1 \\ 2 \end{pmatrix}$.

Exercice 3.25. Le but de cet exercice est de montrer qu'on ne peut rien dire en général de la comparaison de deux méthodes itératives. On note respectivement B_J et B_{GS} les matrices d'itération de Jacobi et de Gauss-Seidel.

1. Pour $A = \begin{pmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{pmatrix}$, montrer que $\rho(B_J) < 1 < \rho(B_{GS})$.
2. Pour $B = \begin{pmatrix} 2 & -1 & 1 \\ 2 & 2 & 2 \\ -1 & -1 & 2 \end{pmatrix}$, montrer que $\rho(B_{GS}) < 1 < \rho(B_J)$.

Chapitre 4

Représentation des nombres en machine : le standard IEEE-754

4.1 Représentation d'un nombre réel en base b

4.1.1 Représentation des entiers naturels

Théorème 4.1. Soit $b \in \mathbb{N}, b \geq 2$. Pour tout $n \in \mathbb{N}$, il existe un unique $p \in \mathbb{N} \setminus \{0\}$ et un unique p-uplet $(a_0, \ldots, a_{p-1}) \in [\![0, b-1]\!]^p$ tels que

$$n = \sum_{k=0}^{p-1} a_k b^k.$$

La donnée de p et de $(a_0, \ldots, a_{p-1})$ est appelée *représentation de n en base b* et on écrit

$$n = \overline{a_{p-1} \ldots a_1 a_0}^{(b)}$$

Exemple 4.2. Quelques bases usuelles

- $b = 2$: représentation binaire
- $b = 8$: représentation octale (utilisée dans les transpondeurs)
- $b = 10$: représentation décimale
- $b = 12$: représentation duodécimale (mois de l'année)
- $b = 16$: représentation hexadécimale (usage en informatique)
- $b = 60$: réprésentation sexagésimale (minutes, secondes)

Exemple 4.3. Représentation de 13 en bases $2, 4, 8$ et 16.

$13 = 2 \times 6 + \mathbf{1}$
$6 = 2 \times 3 + \mathbf{0}$
$3 = 2 \times 1 + \mathbf{1}$
$1 = 2 \times 0 + \mathbf{1}$
Donc $13 = \overline{1101}^{(2)}$

$13 = 8 \times 1 + \mathbf{5}$
$5 = 8 \times 0 + \mathbf{1}$
Donc $13 = \overline{15}^{(8)}$

$13 = 16 \times 0 + \mathbf{13}$
Donc $13 = \overline{D}^{(16)}$

4.1.2 Représentation des réels de $[0, 1[$

Théorème 4.4. Soit $b \in \mathbb{N},\ b \geq 2$. Pour tout $x \in [0, 1[$, il existe une unique suite $(a_k)_{k\geq 1} \in [\![0, b-1]\!]^{\mathbb{N}}$ telle que

$$x = \sum_{k=1}^{+\infty} a_k b^{-k}.$$

La donnée de $(a_k)_{k\geq 1}$ est appelée représentation de x en base b. On écrit $x = \overline{0, a_1 a_2 \dots}^{(b)}$.

Remarque 4.5. Une telle représentation est potentiellement infinie.

Exemple 4.6. Représenation de $0,1875$ en base $2, 4, 8$ et 16.

$0,1875 \times 2 = \mathbf{0},375$
$0,375 \times 2 = \mathbf{0},75$
$0,75 \times 2 = \mathbf{1},5$
$0,5 \times 2 = \mathbf{1},0$
$0,1875 = \overline{0,0011}^{(2)}$

$0,1875 \times 8 = \mathbf{1},5$
$0,5 \times 8 = \mathbf{4},0$
$0,1875 = \overline{0,14}^{(8)}$

$0,1875 \times 16 = \mathbf{3},0$
$0,1875 = \overline{0,3}^{(16)}$

4.1.3 Représentation des réels : exercices

Exercice 4.7. Donner la représentation des nombres suivants en binaire

1. 15, 275
2. 2,37
3. 0,1678
4. 9,564

Exercice 4.8. Donner la représentation des nombres suivants en hexadécimal

1. 15, 275
2. 358,937
3. 387,62
4. 5233,618

4.2 Description de la norme `IEEE-754`

4.2.1 Principe général

L'`IEEE-754` est une norme de représentation des nombres à virgule flottante utilisée pour le calcul en machine. Dans cette norme, on représente un nombre x de la manière suivante

$$x = s \times m \times 2^{e-b},$$

Avec

- s le **signe** : $s = 0$ si $x \geq 0$, $s = 1$ si $x < 0$,
- m la **mantisse**,
- $e - b$ est l'**exposant**,
- e l'**exposant biaisé**,
- $b = 2^{N_e - 1} - 1$ le **biais**, où N_e désigne le nombre de bits utilisés pour stocker l'exposant (voir plus loin).

Remarque 4.9. On peut voir cette représentation comme une version en binaire de l'écriture scientifique en base 10. La présence du biais b permet de stocker un exposant biaisé e qui est toujours positif, l'exposant $e - b$ pourra quant à lui prendre des valeurs négatives.

En machine, on dispose d'une capacité limitée pour représenter les nombres réels (généralement 32 bits ou 64 bits). On doit donc imposer des restrictions sur le nombre de bits pour stocker l'exposant e et la mantisse m et représenter le nombre x. Nous supposerons donc que

— Le signe s est stocké sur 1 bit,

— L'exposant $e = \overline{e_{N_e-1} \dots e_0}^{(2)}$ est stocké sur N_e bits,

— La mantisse $m = \overline{m_{N_m-1} \dots m_0}^{(2)}$ est stockée sur N_m bits.

Ainsi le nombre x peut-être représenté par le tableau suivant :

Signe	**Exposant**			**Mantisse**		
s	e_{N_e-1}	$\dots$	e_0	m_{N_m-1}	$\dots$	m_0

4.2.2 Valeurs spéciales

Une fois ce format de stockage décrit, on souhaite faire des opérations sur les flottants (addition, soustraction, multiplication, division). Or lors de ces opérations, il arrive d'atteindre des valeurs très grandes ($+\infty$, $-\infty$), ou des erreurs, qui doivent être sockées dans la représentation. On dispose des valeurs spéciales suivantes :

— « Zéro positif » : $\boxed{0 \mid 0 \ \dots \ 0 \mid 0 \ \dots \ 0}$;

— « Zéro négatif » : $\boxed{1 \mid 0 \ \dots \ 0 \mid 0 \ \dots \ 0}$;

— « Infini positif » : $\boxed{0 \mid 1 \ \dots \ 1 \mid 0 \ \dots \ 0}$;

— « Infini négatif » : $\boxed{1 \mid 1 \ \dots \ 1 \mid 0 \ \dots \ 0}$;

— « NaN » (Not a Number) : $\boxed{0 \mid 1 \ \dots \ 1 \mid \neq 0}$ ou $\boxed{1 \mid 1 \ \dots \ 1 \mid \neq 0}$.

4.2.3 Format simple précision (32 bits)

Le format simple précision correspond au stockage des flottants sur 32 bits. Dans ce cas on a $N_e = 8$ et $N_m = 23$. Un nombre au format simple précision sera donc de la forme :

Signe	**Exposant**			**Mantisse**		
s	e_7	$\dots$	e_0	m_{22}	$\dots$	m_0

Nous allons illustrer par un exemple comment calculer la représentation d'un nombre au format 32 bits/simple précision. L'exemple est tiré de la page WIKIPÉDIA du format IEEE-754 https://fr.wikipedia.org/w/index.php?title=IEEE_754&oldid=233526448.

Exemple 4.10. Nous allons représenter le nombre $x = -118,625$ au format `IEEE-754`, simple précision.

1. $x < 0$ donc $s = 1$.
2. On commence par écrire $118,625$ en binaire. On trouve $118,625 = \overline{1110110,101}^{(2)}$,
3. On décale la virgule : $\overline{1,110110101}^{(2)} \times 2^6$.
4. La mantisse est la partie à droite de la virgule à laquelle on ajoute des 0. On obtient
$$m = 110\ 1101\ 0100\ 0000\ 0000\ 0000,$$
5. L'exposant est $e - b = 6$. Comme $b = 2^{N_e - 1} - 1 = 2^7 - 1 = 127$, on obtient $e = 6 + 127 = 133$. En binaire, cela donne $e = \overline{10000101}^{(2)}$. Conclusion :
$$-118,625 = 1 \quad 1000\ 0101 \quad 110\ 1101\ 0100\ 0000\ 0000\ 0000.$$

4.2.4 Format double précision (64 bits)

Le format double précision correspond au stockage des flottants sur 64 bits. Dans ce cas on a $N_e = 11$ et $N_m = 52$. Un nombre au format double précision sera donc de la forme :

Signe	**Exposant**			**Mantisse**		
s	e_{10}	$\dots$	e_0	m_{51}	$\dots$	m_0

Nous allons reprendre l'exemple précédent et en calculer la représentation au format 64 bits/double précision.

Exemple 4.11. Nous allons représenter le nombre $x = -118,625$ au format `IEEE-754`, double précision.

1. $x < 0$ donc $s = 1$.
2. On commence par écrire $118,625$ en binaire. On trouve $118,625 = \overline{1110110,101}^{(2)}$,
3. On décale la virgule : $\overline{1,110110101}^{(2)} \times 2^6$.
4. La mantisse est la partie à droite de la virgule à laquelle on ajoute des 0. On obtient
$$m = 110\ 1101\ 0100\ \dots\ 0,$$
5. L'exposant est $e - b = 6$. Comme $b = 2^{N_e - 1} - 1 = 2^{10} - 1 = 1023$, on obtient $e = 6 + 1023 = 1029$. En binaire, cela donne $e = \overline{100\ 0000\ 0101}^{(2)}$. Conclusion :
$$-118,625 = 1 \quad 100\ 0000\ 0101 \quad 110\ 1101\ 0100\ \dots\ 0.$$

Exercice 4.12. Calculer la représentation

1. de $5,75$ au format `IEEE-754`, simple précision puis double précision,
2. Même question pour $0,1$.

Bibliographie

[1] P. Lascaux, R. Théodor. *Analyse numérique matricielle appliquée à l'art de l'ingénieur*. Tome 1 : Méthodes directes. Dunod, 2000.

[2] P. Lascaux, R. Théodor. *Analyse numérique matricielle appliquée à l'art de l'ingénieur*. Tome 2 : Méthodes itératives. Dunod, 2000.

[3] P.G. Ciarlet. *Introduction à l'analyse numérique matricielle et à l'optimisation, troisième tirage*. Masson, 1988.

[4] Romain Dujol et Laurence Lamoulie. *Algèbre linéaire appliquée, Cycle ingénieur, première année mathématique*. Cours et TD. CY Tech Cergy-Pau.